Package 'gmvarkit'

September 17, 2021

Title Estimate Gaussian Mixture Vector Autoregressive Model

Version 1.5.0

Description
Unconstrained and constrained maximum likelihood estimation of structural and reduced form Gaussian mixture vector autoregressive (GMVAR) model, quantile residual tests, graphical diagnostics,
simulations, forecasting, and estimation of generalized impulse response function and generalized forecast error variance decomposition.
Leena Kalliovirta, Mika Meitz, Pentti Saikkonen (2016) <doi:10.1016/j.jeconom.2016.02.012>, Savi Virolainen (2020) <arXiv:2007.04713>.

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports Brobdingnag (>= 1.2-4), mvnfast (>= 0.2.5), parallel (>= 3.0.0), stats (>= 3.0.0), pbapply (>= 1.4-2), graphics (>= 3.0.0), grDevices (>= 3.0.0)

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Savi Virolainen [aut, cre]

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

Repository CRAN

Date/Publication 2021-09-17 12:40:02 UTC

R topics documented:

add_data *Add data to an 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_cond_moments = TRUE, calc_std_errors = FALSE)
```

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.
gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
calc_cond_moments	should conditional means and covariance matrices should be calculated? Default is TRUE if the model contains data and FALSE otherwise.
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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, iterate_more, update_numtols

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
  0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
  0.136, 0.674)
mod12 <- GMVAR(p=1, M=2, d=2, params=params12)
mod12
```

```
mod12_2 <- add_data(gdpdef, mod12)
mod12_2


# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(p=2, M=2, d=2, params=params22s, structural_pars=list(W=W_22))
mod22s

mod22s_2 <- add_data(gdpdef, mod22s)
mod22s_2
```

`alt_gmvar` *Construct a GMVAR model based on results from an arbitrary estimation round of* `fitGMVAR`

Description

`alt_gmvar` constructs a GMVAR model based on results from an arbitrary estimation round of `fitGMVAR`.

Usage

```
alt_gmvar(
  gmvar,
  which_round = 1,
  which_largest,
  calc_cond_moments = TRUE,
  calc_std_errors = TRUE
)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.
`which_round`	based on which estimation round should the model be constructed? An integer value in 1,...,`ncalls`.
`which_largest`	based on estimation round with which largest log-likelihood should the model be constructed? An integer value in 1,...,`ncalls`. For example, `which_largest=2` would take the second largest log-likelihood and construct the model based on the corresponding estimates. If used, then `which_round` is ignored.
`calc_cond_moments`	should conditional means and covariance matrices should be calculated? Default is `TRUE` if the model contains data and `FALSE` otherwise.
`calc_std_errors`	should approximate standard errors be calculated?

Details

It's sometimes useful to examine other estimates than the one with the highest log-likelihood. This function is wrapper around GMVAR that picks the correct estimates from an object returned by fitGMVAR.

Value

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

Remark that the first autocovariance/correlation matrix in $uncond_moments is for the lag zero, the second one for the lag one, etc.

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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, iterate_more, update_numtols

Examples

```
# GMVAR(1,2) model
fit12 <- fitGMVAR(gdpdef, p=1, M=2, ncalls=2, seeds=4:5)
fit12
fit12_2 <- alt_gmvar(fit12, which_largest=2)
fit12_2
```

---

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 a class 'gmvar' object. get_soc returns eigenvalues of the Hessian matrix, and get_foc is the same as get_gradient but named conveniently.

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)

get_soc(gmvar, h = 6e-06)

get_foc(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 derivatives.
...	other arguments passed to fn
gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.

Details

In particular, the functions get_foc and get_soc can be used to check whether the found estimates denote a (local) maximum point, a saddle point, or something else. Note that profile log-likelihood functions can be conveniently plotted with the function profile_logliks.

Value

Gradient functions return numerical approximation of the gradient and Hessian functions return numerical approximation of the Hessian. get_soc returns eigenvalues of the Hessian matrix.

Warning

No argument checks!

See Also

profile_logliks

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)


# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
get_gradient(mod12)
get_hessian(mod12)
get_soc(mod12)
```

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

Description

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

Usage

```
check_parameters(
  p,
  M,
  d,
  params,
  parametrization = c("intercept", "mean"),
  constraints = NULL,
  same_means = NULL,
  structural_pars = NULL,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

Arguments

`p`	a positive integer specifying the autoregressive order of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	the number of time series in the system.
`params`	a real valued vector specifying the parameter values. **For unconstrained models:** Should be size $((M(pd^2+d+d(d+1)/2+1)-1)x1)$ and have the 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 the form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

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

For same_means models: Should have the form θ= $(\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- μ= $(\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, ψ= $(\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- λ_m = $(\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

`parametrization`

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

`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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that

should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

`posdef_tol` numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

Examples

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

```
# GMVAR(1, 1), d=2 model:
params11 <- 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:
params22 <- 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=params22)

# 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=params22c, constraints=C_mat)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints
# (no error):
params22s <- c(1.03, 2.36, 1.79, 3, 1.25, 0.06, 0.04, 1.34, -0.29,
 -0.08, -0.05, -0.36, 1.2, 0.05, 0.05, 1.3, -0.3, -0.1, -0.05, -0.4,
  0.89, 0.72, -0.37, 2.16, 7.16, 1.3, 0.37)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
check_parameters(p=2, M=2, d=2, params=params22s,
 structural_pars=list(W=W_22))

## End(Not run)
```

`cond_moments` *Compute conditional moments of a GMVAR model*

Description

`loglikelihood` compute conditional regimewise means, conditional means, and conditional covariance matrices of a GMVAR model.

Usage

```
cond_moments(
  data,
  p,
  M,
  params,
  parametrization = c("intercept", "mean"),
  constraints = NULL,
```

```
  same_means = NULL,
  structural_pars = NULL,
  to_return = c("regime_cmeans", "total_cmeans", "total_ccovs"),
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

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 order of the model.
M	a positive integer specifying the number of mixture components.
params	a real valued vector specifying the parameter values.

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

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

For same_means models: Should have the form $\theta = (\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- $\mu = (\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, $\psi = (\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

`parametrization`

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

`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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`to_return` should the regimewise conditional means, total conditional means, or total conditional covariance matrices be returned?

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

`posdef_tol` numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller

than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

The first p values are used as the initial values, and by conditional we mean conditioning on the past. Formulas for the conditional means and covariance matrices are given in equations (3) and (4) of KMS (2016).

Value

If `to_return=="regime_cmeans"`**:** an `[T-p,d,M]` array containing the regimewise conditional means (the first p values are used as the initial values).

If `to_return=="total_cmeans"`**:** a `[T-p,d]` matrix containing the conditional means of the process (the first p values are used as the initial values).

If `to_return=="total_ccov"`**:** an `[d,d,T-p]` array containing the conditional covariance matrices of the process (the first p values are used as the initial values).

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

Other moment functions: `get_regime_autocovs()`, `get_regime_means()`, `uncond_moments()`

Examples

```
# GMVAR(2, 2), d=2 model;
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
cond_moments(data=gdpdef, p=2, M=2, params=params22, to_return="regime_cmeans")
cond_moments(data=gdpdef, p=2, M=2, params=params22, to_return="total_cmeans")
cond_moments(data=gdpdef, p=2, M=2, params=params22, to_return="total_ccovs")
```

cond_moment_plot *Conditional mean or variance plot for a GMVAR model*

Description

cond_moment_plot plots the one-step in-sample conditional means/variances of the model along with the individual time series contained in the model (e.g. the time series the model was fitted to). Also plots the regimewise conditional means/variances multiplied with mixing weights.

Usage

```
cond_moment_plot(
  gmvar,
  which_moment = c("mean", "variance"),
  grid = FALSE,
  ...
)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
which_moment	should conditional means or variances be plotted?
grid	add grid to the plots?
...	additional paramters passed to grid(...) plotting the grid if grid == TRUE.

Details

The conditional mean plot works best if the data contains positive values only. acf from the package stats and the plot method for class 'acf' objects is employed.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

profile_logliks, fitGMVAR, GMVAR, quantile_residual_tests, LR_test, Wald_test, diagnostic_plot

Examples

```
# GMVAR(2, 2), d=2 model;
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22)

cond_moment_plot(mod22, which_moment="mean")
cond_moment_plot(mod22, which_moment="variance")
cond_moment_plot(mod22, which_moment="mean", grid=TRUE, lty=3)
```

---

diagnostic_plot *Quantile residual diagnostic plot for a GMVAR model*

Description

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

Usage

```
diagnostic_plot(
  gmvar,
  type = c("all", "series", "ac", "ch", "norm"),
  maxlag = 12,
  wait_time = 4
)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
type	which type of diagnostic plot should be plotted? • "all" all below sequentially. • "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 histogram with theoretical standard normal density (dashed line) and standard normal QQ-plots.
maxlag	the maximum lag considered in types "ac" and "ch".
wait_time	if type == all how many seconds to wait before showing next figure?

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 employed.

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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

profile_logliks, fitGMVAR, GMVAR, quantile_residual_tests, LR_test, Wald_test, cond_moment_plot, acf, density, predict.gmvar

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319,
 0.005, 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185,
 -0.012, 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
diagnostic_plot(mod12, type="series")
diagnostic_plot(mod12, type="ac")

# GMVAR(2,2), d=2 model:
params22 <-  c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406,
 -0.005, 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218,
 0.02, -0.119, 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004,
  0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22)
diagnostic_plot(mod22, type="ch")
diagnostic_plot(mod22, 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))
params22c <- c(0.418, 0.153, 0.513, 0.057, 0.204, 0.028, -0.169,
 0.591, 0.241, 0.014, 0.091, 0.248, 1.068, -0.01, 0.111, 0.219,
 0.004, 0.027, 0.501)
mod22c <- GMVAR(gdpdef, p=2, M=2, params=params22c, constraints=C_mat)
diagnostic_plot(mod22c, wait_time=0.2)
diagnostic_plot(mod22c, type="ac", maxlag=12)
```

---

diag_Omegas *Simultaneously diagonalize two covariance matrices*

Description

diag_Omegas Simultaneously diagonalizes two covariance matrices using eigenvalue decomposition.

Usage

```
diag_Omegas(Omega1, Omega2)
```

Arguments

Omega1	a positive definite (dxd) covariance matrix $(d > 1)$
Omega2	another positive definite (dxd) covariance matrix

Details

See the return value and Muirhead (1982), Theorem A9.9 for details.

Value

Returns a length $d^2 + d$ vector where the first d^2 elements are $vec(W)$ with the columns of W being (specific) eigenvectors of the matrix $\Omega_2\Omega_1^{-1}$ and the rest d elements are the corresponding eigenvalues "lambdas". The result satisfies $WW' = Omega1$ and $Wdiag(lambdas)W' = Omega2$.

If Omega2 is not supplied, returns a vectorized symmetric (and pos. def.) square root matrix of Omega1.

Warning

No argument checks! Does not work with dimension $d = 1$!

References

- Muirhead R.J. 1982. Aspects of Multivariate Statistical Theory, *Wiley*.

Examples

```
d <- 2
W0 <- matrix(1:(d^2), nrow=2)
lambdas0 <- 1:d
(Omg1 <- W0%*%t(W0))
(Omg2 <- W0%*%diag(lambdas0)%*%t(W0))
res <- diag_Omegas(Omg1, Omg2)
W <- matrix(res[1:(d^2)], nrow=d, byrow=FALSE)
tcrossprod(W) # == Omg1
lambdas <- res[(d^2 + 1):(d^2 + d)]
W%*%diag(lambdas)%*%t(W) # == Omg2
```

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

Description

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

Usage

```
fitGMVAR(
  data,
  p,
  M,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  constraints = NULL,
  same_means = NULL,
  structural_pars = NULL,
  ncalls = 100,
  ncores = 2,
  maxit = 500,
  seeds = NULL,
  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 order 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
`parametrization`	`"intercept"` or `"mean"` determining whether the model is parametrized with intercept parameters $\phi_{m,0}$ or regime means μ_m, m=1,...,M.
`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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

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

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

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

`seeds` a length `ncalls` vector containing the random number generator seed for each call to the genetic algorithm, or `NULL` for not initializing the seed. Exists for creating reproducible results.

`print_res` should summaries of estimation results be printed?

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

Details

If you wish to estimate a structural model without overidentifying constraints that is identified statistically, specify your W matrix is `structural_pars` to be such that it contains the same sign constraints in a single row (e.g. a row of ones) and leave the other elements as `NA`. In this way, the genetic algorithm works the best. The ordering and signs of the columns of the W matrix can be changed afterwards with the functions `reorder_W_columns` and `swap_W_signs`.

Because of complexity and high 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 or saddle point instead. Therefore, a (sometimes large) 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. **With two regimes and couple hundred observations in a two-dimensional time series, 50 rounds is usually enough. Several hundred estimation rounds often suffices for reliably fitting two-regimes models to 3 or 4 dimensional time series. With more than two regimes and more than couple hundred observations, thousands of estimation rounds (or more) are often required to obtain reliable results.**

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`. Alternatively, one may use the found estimates as starting values for the genetic algorithm and and employ another round of estimation (see `?GAfit` how to set up an initial population with the dot parameters).

If the estimation algorithm fails to create an initial population for the genetic algorithm, it usually helps to scale the individual series so that the AR coefficients (of a VAR model) will be relative small, preferably less than one. Even if one is able to create an initial population, it should be preferred to scale the series so that most of the AR coefficients will not be very large, as the estimation algorithm works better with relatively small AR coefficients. If needed, another package can be used to fit linear VARs to the series to see which scaling of the series results in relatively small AR coefficients.

The code of the genetic algorithm is mostly based on the description by *Dorsey and Mayer (1995)* but it includes some extra features that were found useful for this particular estimation problem. For instance, the genetic algorithm uses a slightly modified version of the 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`.

Note that the structural models are even more difficult to estimate than the reduced form models due to the different parametrization of the covariance matrices, so larger number of estimation rounds should be considered. Also, be aware that if the lambda parameters are constrained in any other way than by restricting some of them to be identical, the parameter "lambda_scale" of the genetic algorithm (see `?GAfit`) needs to be carefully adjusted accordingly.

Finally, the function fails to calculate approximate standard errors and the parameter estimates are near the border of the parameter space, it might help to use smaller numerical tolerance for the stationarity and positive definiteness conditions. The numerical tolerance of an existing model can be changed with the function `update_numtols`.

Value

Returns an object of class `'gmvar'` defining the estimated (reduced form or structural) 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`). See `?GMVAR` for the form of the parameter vector, if needed.

Remark that the first autocovariance/correlation matrix in `$uncond_moments` is for the lag zero, the second one for the lag one, etc.

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.
- 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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

GMVAR, iterate_more, predict.gmvar, profile_logliks, simulateGMVAR, quantile_residual_tests, print_std_errors, swap_parametrization, get_gradient, GIRF, GFEVD, LR_test, Wald_test, gmvar_to_sgmvar, reorder_W_columns, swap_W_signs, cond_moment_plot, update_numtols

Examples

```
## These are long running examples that use parallel computing!
# Running all the below examples will take approximately 3-4 minutes.

# GMVAR(1,2) model: 10 estimation rounds with seeds set
# for reproducibility
fit12 <- fitGMVAR(gdpdef, p=1, M=2, ncalls=10, seeds=1:10)
fit12
plot(fit12)
summary(fit12)
print_std_errors(fit12)
profile_logliks(fit12)

# The rest of the examples only use a single estimation round with a given
# seed that produces the MLE to reduce running time of the examples. When
# estimating models for empirical applications, a large number of estimation
# rounds (ncalls = a large number) should be performed to ensure reliability
# of the estimates (see the section details).

# Structural GMVAR(1,2) model identified with sign
# constraints.
W_122 <- matrix(c(1, 1, -1, 1), nrow=2)
fit12s <- fitGMVAR(gdpdef, p=1, M=2, structural_pars=list(W=W_122),
```

```
  ncalls=1, seeds=1)
fit12s
# A statistically identified structural model can also be obtained with
# gmvar_to_sgmvar(fit12)

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for both regimes
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22c <- fitGMVAR(gdpdef, p=2, M=2, constraints=C_mat, ncalls=1, seeds=1)
fit22c

# GMVAR(2,2) model with autoregressive parameters restricted
# to be the same for both regimes and non-diagonal 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(gdpdef, p=2, M=2, constraints=C_mat2, ncalls=1,
  seeds=1)
fit22c2
```

GAfit *Genetic algorithm for preliminary estimation of a GMVAR model*

Description

GAfit estimates the specified GMVAR model using a 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,
  same_means = NULL,
  structural_pars = NULL,
  ngen = 200,
  popsize,
  smart_mu = min(100, ceiling(0.5 * ngen)),
  initpop = NULL,
  mu_scale,
  mu_scale2,
  omega_scale,
  W_scale,
```

```
  lambda_scale,
  ar_scale = 0.2,
  upper_ar_scale = 1,
  ar_scale2 = 1,
  regime_force_scale = 1,
  red_criteria = c(0.05, 0.01),
  pre_smart_mu_prob = 0,
  to_return = c("alt_ind", "best_ind"),
  minval,
  seed = 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 order 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
`parametrization`	`"intercept"` or `"mean"` determining whether the model is parametrized with intercept parameters $\phi_{m,0}$ or regime means μ_m, m=1,...,M.
`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 related 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.
`same_means`	Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**
`structural_pars`	If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements: • `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero. • `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where γ is the new $(rx1)$ parameter subject to which the model is

estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`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 unconstrained 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.

For structural GMVAR model: Should have the form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M)$ where

- $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line

with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

`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"`. However, input (in `initpop`) and output (return value) parameter vectors can 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)$resid` This argument is ignored if structural model is considered.

`W_scale` a size $(dx1)$ strictly positive vector partly specifying the scale and variability of the random covariance matrices in random mutations. The elements of the matrix W are drawn independently from such normal distributions that the expectation of the main **diagonal** elements of the first regime's error term covariance matrix $\Omega_1 = WW'$ is `W_scale`. The distribution of Ω_1 will be in some sense like a Wishart distribution but with the columns (elements) of W obeying the given constraints. The constraints are accounted for by setting the element to be always zero if it is subject to a zero constraint and for sign constraints the absolute value or negative the absolute value are taken, and then the variances of the elements of W are adjusted accordingly. This argument is ignored if reduced form model is considered.

`lambda_scale` a length $M-1$ vector specifying the **standard deviation** of the mean zero normal distribution from which the eigenvalue λ_{mi} parameters are drawn from in random mutations. As the eigenvalues should always be positive, the absolute value is taken. The elements of `lambda_scale` should be strictly positive real numbers with the $m-1$th element giving the degrees of freedom for the mth regime. The expected value of the main **diagonal** elements ij of the mth $(m > 1)$ error term covariance matrix will be `W_scale[i]*(d-n_i)^(-1)*sum(lambdas*ind_fun)` where the $(dx1)$ vector `lambdas` is drawn from the absolute value of the t-distribution, `n_i` is the number of zero constraints in the ith row of W and `ind_fun` is an indicator function that takes the value one iff the ijth element of W is not constrained to zero. Basically, larger lambdas (or smaller degrees of freedom) imply larger variance.

If the lambda parameters are **constrained** with the $(d(M-1)xr)$ constraint matrix C_lambda, then provide a length r vector specifying the standard deviation of the (absolute value of the) mean zero normal distribution each of the γ parameters are drawn from (the γ is a $(rx1)$ vector). The expected value of the main diagonal elements of the covariance matrices then depend on the constraints.

This argument is ignored if $M == 1$ or a reduced form model is considered. Default is `rep(3,times=M-1)` if lambdas are not constrained and `rep(3,times=r)` if lambdas are constrained.

As with omega_scale and W_scale, this argument should be adjusted carefully if specified by hand. **NOTE** that if lambdas are constrained in some other way than restricting some of them to be identical, this parameter should be adjusted accordingly in order to the estimation succeed!

`ar_scale` a positive real number adjusting how large AR parameter values are typically proposed in construction of the initial population: larger value implies larger coefficients (in absolute value). After construction of the initial population, a new scale is drawn from `(0,0.)` uniform distribution in each iteration.

`upper_ar_scale` the upper bound for `ar_scale` parameter (see above) in the random mutations. Setting this too high might lead to failure in proposing new parameters that are well enough inside the parameter space, and especially with large `p` one might want to try smaller upper bound (e.g., 0.5).

`ar_scale2` a positive real number adjusting how large AR parameter values are typically proposed in some random mutations (if AR constraints are employed, in all random mutations): larger value implies **smaller** coefficients (in absolute value). **Values larger than 1 can be used if the AR coefficients are expected to be very small. If set smaller than 1, be careful as it might lead to failure in the creation of stationary parameter candidates**

`regime_force_scale`

a non-negative real number specifying how much should natural selection favor individuals with less regimes that have almost all mixing weights (practically) at zero. Set to zero for no favoring or large number for heavy favoring. Without any favoring 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 probability weights for individuals with no wasted regimes compared to individuals with one wasted regime. Number `2` would give $3x$ larger probability weights etc.

`red_criteria` a length 2 numeric vector specifying the criteria that is used to determine whether a regime is redundant (or "wasted") or not. Any regime `m` which satisfies `sum(mixingWeights[,m] > red_criteria[1]) < red_criteria[2]*n_obs` will be considered "redundant". One should be careful when adjusting this argument (set `c(0,0)` to fully disable the 'redundant regime' features from the algorithm).

`pre_smart_mu_prob`

A number in $[0, 1]$ giving a probability of a "smart mutation" occuring randomly in each iteration before the iteration given by the argument `smart_mu`.

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

`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)) + d) -1)`.

`seed` a single value, interpreted as an integer, or NULL, that sets seed for the random number generator in the beginning of the function call. If calling `GAfit` from `fitGMVAR`, use the argument `seeds` instead of passing the argument `seed`.

Details

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

By "redundant" or "wasted" regimes we mean regimes that have the time varying mixing weights practically at zero for almost all t. A model including redundant regimes would have about the same log-likelihood value without the redundant regimes and there is no purpose to have redundant regimes in a model.

Value

Returns the 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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@export

---

gdpdef *U.S. real GDP percent change and GDP implicit price deflator percent change.*

Description

A dataset containing a quarterly U.S. time series with two components: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 - 2019Q4.

Usage

```
gdpdef
```

Format

A numeric matrix of class `'ts'` with 244 rows and 2 columns with one time series in each column:

First column (GDP): The quarterly percent change of real U.S. GDP, from 1959Q1 to 2019Q4, https://fred.stlouisfed.org/series/GDPC1.

Second column (GDPDEF): The quarterly percent change of U.S. GDP implicit price deflator, from 1959Q1 to 2019Q4, https://fred.stlouisfed.org/series/GDPDEF.

Source

The Federal Reserve Bank of St. Louis database and the Federal Reserve Bank of Atlanta's website

---

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`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.

Value

Returns a matrix with $d * p$ rows and M columns - one column for each regime. The mth column contains the absolute values (or modulus) of the eigenvalues of the "bold A" matrix containing the AR coefficients correspinding to regime m.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

Examples

```
# GMVAR(2, 2), d=2 model
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
  0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(p=2, M=2, d=2, params=params22)
get_boldA_eigens(mod22)
```

---

get_omega_eigens *Calculate the eigenvalues of the "Omega" error term covariance matrices*

Description

`get_omega_eigens` calculates the eigenvalues of the "Omega" error term covariance matrices for each mixture component.

Usage

```
get_omega_eigens(gmvar)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.

Value

Returns a matrix with d rows and M columns - one column for each regime. The mth column contains the eigenvalues of the "Omega" error term covariance matrix of the mth regime.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

Examples

```
# GMVAR(2, 2), d=2 model
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
  0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(p=2, M=2, d=2, params=params22)
get_omega_eigens(mod22)
```

`get_regime_autocovs` *Calculate regimewise autocovariance matrices*

Description

`get_regime_autocovs` calculates the first p regimewise autocovariance matrices $\Gamma_m(j)$ for the given GMVAR model.

Usage

```
get_regime_autocovs(gmvar)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.

Value

Returns an $(dxdxp + 1xM)$ array containing the first p regimewise autocovariance matrices. The subset `[,,j,m]` contains the j-1:th lag autocovariance matrix of the m:th regime.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

Other moment functions: `cond_moments()`, `get_regime_means()`, `uncond_moments()`

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
get_regime_autocovs(mod12)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
```

```
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s, structural_pars=list(W=W_22))
mod22s
get_regime_autocovs(mod22s)
```

`get_regime_means` *Calculate 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`	an object of class `'gmvar'` created with `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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

`uncond_moments`, `get_regime_autocovs`, `cond_moments`

Other moment functions: `cond_moments()`, `get_regime_autocovs()`, `uncond_moments()`

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
mod12
get_regime_means(mod12)
```

```
# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s, structural_pars=list(W=W_22))
mod22s
get_regime_means(mod22s)
```

GFEVD *Estimate generalized forecast error variance decomposition for a structural GMVAR model.*

Description

GFEVD estimates generalized generalized forecast error variance decomposition for a structural GMVAR model.

Usage

```
GFEVD(
  gmvar,
  shock_size = 1,
  N = 30,
  initval_type = c("data", "random", "fixed"),
  R1 = 250,
  R2 = 250,
  init_regimes = NULL,
  init_values = NULL,
  which_cumulative = numeric(0),
  include_mixweights = FALSE,
  ncores = 2,
  seeds = NULL
)

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

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

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
shock_size	What shocks size should be used for all shocks? By the definition of the SGMVAR model, the conditional covariance matrix of the structural shock is identity matrix.

`N` a positive integer specifying the horizon how far ahead should the GFEVD be calculated.

`initval_type` What type initial values are used for estimating the GIRFs that the GFEVD is based on?

`"data"`: Estimate the GIRF for all the possible length p histories in the data.

`"random"`: Estimate the GIRF for several random initial values generated from the stationary distribution of the process or from the stationary distribution of specific regime(s) chosen with the argument `init_regimes`. The number of initial values is set with the argument `R2`.

`"fixed"`: Estimate the GIRF for a fixed initial value only, which is specified with the argument `init_values`.

`R1` the number of repetitions used to estimate GIRF for each initial value.

`R2` the number of initial values to be drawn if `initval_type="random"`.

`init_regimes` a numeric vector of length at most M and elements in $1, ..., M$ specifying the regimes from which the initial values should be generated from. The initial values will be generated from a mixture distribution with the mixture components being the stationary distributions of the specific regimes and the (proportional) mixing weights given by the mixing weight parameters of those regimes. Note that if `init_regimes=1:M`, the initial values are generated from the stationary distribution of the process and if `init_regimes=m`, the initial value are generated from the stationary distribution of the mth regime. Ignored if `init_value` is specified.

`init_values` a matrix or a multivariate class `'ts'` object with d columns and at least p rows specifying an initial value for the GIRF. The last p rows are taken to be the initial value assuming that the **last** row is the most recent observation.

`which_cumulative`

a numeric vector with values in $1, ..., d$ (`d=ncol(data)`) specifying which the variables for which the impulse responses should be cumulative. Default is none.

`include_mixweights`

should the GFEVD be estimated for the mixing weights as well? Note that this is ignored if `M=1` and if `M=2` the GFEVD will be the same for both regime's mixing weights.

`ncores` the number CPU cores to be used in parallel computing. Only single core computing is supported if an initial value is specified (and the GIRF won't thus be estimated multiple times).

`seeds` a numeric vector containing the random number generator seed for estimation of each GIRF. Should have the length...

- ...`nrow(data) - p + 1` if `initval_type="data"`.
- ...`R2` if `initval_type="random"`.
- ...`1` if `initval_type="fixed."`.

Set to `NULL` for not initializing the seed. Exists for creating reproducible results.

`x` object of class `'gfevd'` generated by the function `GFEVD`.

`...` currently not used.

`digits` the number of decimals to print

`N_to_print` an integer specifying the horizon how far to print the estimates. The default is that all the values are printed.

Details

The model needs to be structural in order for this function to be applicable. A structural GMVAR model can be estimated by specifying the argument `structural_pars` in the function `fitGMVAR`.

The GFEVD is a forecast error variance decomposition calculated with the generalized impulse response function (GIRF). Lanne and Nyberg (2016) for details. Note, however, that the related GIRFs are calculated using the algorithm given in Virolainen (2020).

Value

Returns and object of class 'gfevd' containing the GFEVD for all the variables and if `include_mixweights=TRUE` also to the mixing weights. Note that the decomposition does not exist at horizon zero for mixing weights because the related GIRFs are always zero at impact.

Methods (by generic)

- `plot`: plot method
- `print`: print method

References

- Lanne M. and Nyberg H. 2016. Generalized Forecast Error Variance Decomposition for Linear and Nonlineae Multivariate Models. *Oxford Bulletin of Economics and Statistics*, **78**, 4, 595-603.
- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

`GIRF`, `fitGMVAR`, `GMVAR`, `gmvar_to_sgmvar`, `reorder_W_columns`, `swap_W_signs`, `simulateGMVAR`

Examples

```
# These are long-running examples that use parallel computing.
# It takes approximately 30 seconds to run all the below examples.

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s,
 structural_pars=list(W=W_22))
mod22s
# Alternatively, use:
#fit22s <- fitGMVAR(gdpdef, p=2, M=2, structural_pars=list(W=W_22),
#                   ncalls=20, seeds=1:20)
```

```
# To obtain an estimated version of the same model.

## NOTE: Use larger R1 is empirical applications! Small R1 is used
## Below only to fasten the execution time of the examples.

# Estimating the GFEVD using all possible histories in the data as the
# initial values:
gfevd1 <- GFEVD(mod22s, N=24, R1=20, initval_type="data")
gfevd1
plot(gfevd1)

# Estimate GFEVD with the initial values generated from the stationary
# distribution of the process:
gfevd2 <- GFEVD(mod22s, N=24, R1=20, R2=100, initval_type="random")
gfevd2
plot(gfevd2)

# Estimate GFEVD with fixed hand specified initial values. We use the
# unconditional mean of the process:
myvals <- rbind(mod22s$uncond_moments$uncond_mean,
                mod22s$uncond_moments$uncond_mean)
gfevd3 <- GFEVD(mod22s, N=36, R1=50, initval_type="fixed",
 init_values=myvals, include_mixweights=TRUE)
gfevd3
plot(gfevd3)
```

GIRF *Estimate generalized impulse response function for a structural GMVAR model.*

Description

GIRF estimates generalized impulse response function for a structural GMVAR model.

Usage

```
GIRF(
  gmvar,
  which_shocks,
  shock_size = 1,
  N = 30,
  R1 = 250,
  R2 = 250,
  init_regimes = 1:gmvar$model$M,
  init_values = NULL,
  which_cumulative = numeric(0),
  scale = NULL,
  ci = c(0.95, 0.8),
```

```
  include_mixweights = TRUE,
  ncores = 2,
  plot = TRUE,
  seeds = NULL,
  ...
)

## S3 method for class 'girf'
plot(x, add_grid = FALSE, margs, ...)

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

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
which_shocks	a numeric vector of length at most d (=ncol(data)) and elements in $1, ..., d$ specifying the structural shocks for which the GIRF should be estimated.
shock_size	a non-zero scalar value specifying the common size for all scalar components of the structural shock. Note that the conditional covariance matrix of the structural shock is an identity matrix and that the (generalized) impulse responses may not be symmetric to the sign and size of the shock.
N	a positive integer specifying the horizon how far ahead should the generalized impulse responses be calculated.
R1	the number of repetitions used to estimate GIRF for each initial value.
R2	the number of initial values to be drawn from a stationary distribution of the process or of a specific regime? The confidence bounds will be sample quantiles of the GIRFs based on different initial values. Ignored if the argument init_value is specified.
init_regimes	a numeric vector of length at most M and elements in $1, ..., M$ specifying the regimes from which the initial values should be generated from. The initial values will be generated from a mixture distribution with the mixture components being the stationary distributions of the specific regimes and the (proportional) mixing weights given by the mixing weight parameters of those regimes. Note that if init_regimes=1:M, the initial values are generated from the stationary distribution of the process and if init_regimes=m, the initial value are generated from the stationary distribution of the mth regime. Ignored if init_value is specified.
init_values	a matrix or a multivariate class 'ts' object with d columns and at least p rows specifying an initial value for the GIRF. The last p rows are taken to be the initial value assuming that the **last** row is the most recent observation.
which_cumulative	a numeric vector with values in $1, ..., d$ (d=ncol(data)) specifying which the variables for which the impulse responses should be cumulative. Default is none.
scale	should the GIRFs to some of the shocks be scaled so that they correspond to a specific magnitude of instantaneous movement of some specific variable? Provide a length three vector where the shock of interest is given in the first element

(an integer in $1, ..., d$), the variable of interest is given in the second element (an integer in $1, ..., d$), and the magnitude of its instantaneous movement (a non-zero real number) in the third element. If the GIRFs of multiple shocks should be scaled, provide a matrix which has one column for each of the shocks with the columns being the length three vectors described above.

`ci` a numeric vector with elements in $(0, 1)$ specifying the confidence levels of the confidence intervals.

`include_mixweights` should the generalized impulse response be calculated for the mixing weights as well? `TRUE` or `FALSE`.

`ncores` the number CPU cores to be used in parallel computing. Only single core computing is supported if an initial value is specified (and the GIRF won't thus be estimated multiple times).

`plot` `TRUE` if the results should be plotted, `FALSE` if not.

`seeds` a length `R2` vector containing the random number generator seed for estimation of each GIRF. A single number of an initial value is specified. or `NULL` for not initializing the seed. Exists for creating reproducible results.

`...` arguments passed to `grid` which plots grid to the figure.

`x` object of class `'girf'` generated by the function `GIRF`.

`add_grid` should grid be added to the plots?

`margs` numeric vector of length four that adjusts the `[bottom_marginal,left_marginal,top_marginal,righ` as the relative sizes of the marginals to the figures of the responses.

`digits` the number of decimals to print

`N_to_print` an integer specifying the horizon how far to print the estimates and confidence intervals. The default is that all the values are printed.

Details

The model needs to be structural in order for this function to be applicable. A structural GMVAR model can be estimated by specifying the argument `structural_pars` in the function `fitGMVAR`.

The confidence bounds reflect uncertainty about the initial state (but currently not about the parameter estimates) if initial values are not specified. If initial values are specified, there won't currently be confidence intervals. See the cited paper by Virolainen (2020) for details about the algorithm.

Note that if the argument `scale` is used, the scaled responses of the mixing weights might be more than one in absolute valie.

Value

Returns a class `'girf'` list with the GIRFs in the first element (`$girf_res`) and the used arguments the rest. The first element containing the GIRFs is a list with the mth element containing the point estimates for the GIRF in `$point_est` (the first element) and confidence intervals in `$conf_ints` (the second element). The first row is for the GIRF at impact $(n = 0)$, the second for $n = 1$, the third for $n = 2$, and so on.

Methods (by generic)

- `plot`: plot method
- `print`: print method

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

`GFEVD`, `fitGMVAR`, `GMVAR`, `gmvar_to_sgmvar`, `reorder_W_columns`, `swap_W_signs`, `simulateGMVAR`, `predict.gmvar`, `profile_logliks`, `quantile_residual_tests`, `LR_test`, `Wald_test`

Examples

```
# These are long-running examples that use parallel computing.
# It takes approximately 30 seconds to run all the below examples.

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s,
 structural_pars=list(W=W_22))
mod22s
# Alternatively, use:
#fit22s <- fitGMVAR(gdpdef, p=2, M=2, structural_pars=list(W=W_22),
#                   ncalls=20, seeds=1:20)
# To obtain an estimated version of the same model.

# Estimating the GIRFs of both structural shocks with initial values
# drawn from the stationary distribution of the process,
# 12 periods ahead, confidence levels 0.95 and 0.8:
girf1 <- GIRF(mod22s, N=12, R1=100, R2=100)
girf1
plot(girf1)

# Estimating the GIRF of the second shock only, 12 periods ahead
# and shock size 1, initial values drawn from the stationary distribution
# of the first regime, confidence level 0.9:
girf2 <- GIRF(mod22s, which_shocks=2, shock_size=1, N=12, init_regimes=1,
              ci=0.9, R1=100, R2=100)

# Estimating the GIRFs of both structural shocks, negative one standard
```

```
# error shock, N=20 periods ahead, estimation based on 200 Monte Carlo
# simulations, and fixed initial values given by the last p observations
# of the data:
girf3 <- GIRF(mod22s, shock_size=-1, N=20, R1=200,
              init_values=mod22s$data)
```

GMVAR *Create a class 'gmvar' object defining a reduced form or structural GMVAR model*

Description

GMVAR creates a class 'gmvar' object that defines a reduced form or structural GMVAR model

Usage

```
GMVAR(
  data,
  p,
  M,
  d,
  params,
  conditional = TRUE,
  parametrization = c("intercept", "mean"),
  constraints = NULL,
  same_means = NULL,
  structural_pars = NULL,
  calc_cond_moments,
  calc_std_errors = FALSE,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)

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

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

## S3 method for class 'gmvar'
summary(object, ..., digits = 2)

## 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 order 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 unconstrained models: Should be size $((M(pd^2+d+d(d+1)/2+1)-1)x1)$ and have the 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 the form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

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

For same_means models: Should have the form $\theta = (\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- $\mu = (\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, $\psi = (\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line

with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

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

`parametrization`

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

`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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`calc_cond_moments`

should conditional means and covariance matrices should be calculated? Default is `TRUE` if the model contains data and `FALSE` otherwise.

`calc_std_errors`

should approximate standard errors be calculated?

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

posdef_tol	numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.
object	object of class 'gmvar' generated by fitGMVAR or GMVAR.
...	currectly not used.
digits	number of digits to be printed.
x	object of class 'gmvar' generated by fitGMVAR or GMVAR.
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.

If the function fails to calculate approximative standard errors and the parameter values are near the border of the parameter space, it might help to use smaller numerical tolerance for the stationarity and positive definiteness conditions.

The first plot displays the time series together with estimated mixing weights. The second plot displays (Gaussian) kernel density estimates of the individual series together with the marginal stationary density implied by the model. The colored regimewise stationary densities are multiplied with the mixing weight parameter estimates.

Value

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

Remark that the first autocovariance/correlation matrix in $uncond_moments is for the lag zero, the second one for the lag one, etc.

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

About S3 methods

Only the print method is available if data is not provided. If data is provided, then in addition to the ones listed above, 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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, add_data, swap_parametrization, GIRF, gmvar_to_sgmvar, reorder_W_columns, swap_W_signs, update_numtols

Examples

```
# GMVAR(1, 2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
  0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
  0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
mod12

# GMVAR(1, 2), d=2 model without data
mod12_2 <- GMVAR(p=1, M=2, d=2, params=params12)
mod12_2

# GMVAR(2, 2), d=2 model with mean-parametrization:
params22 <- c(0.869, 0.549, 0.223, 0.059, -0.151, 0.395, 0.406,
 -0.005, 0.083, 0.299, 0.215, 0.002, 0.03, 0.576, 1.168, 0.218,
 0.02, -0.119, 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004,
 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22, parametrization="mean")
mod22

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s,
 structural_pars=list(W=W_22))
mod22s
```

---

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

Description

gmvarkit is a package for reduced form and structural Gaussian mixture vector autoregressive (GMVAR) model analysis. It provides functions for unconstrained and constrained maximum likelihood estimation of the model parameters, quantile residuals tests, graphical diagnostics, estimation of generalized impulse response function, estimation of generalized forecast error variance decomposition, simulation from GMVAR processes, forecasting, and more.

The readme file is a good place to start and the vignette might be useful too.

`gmvar_to_sgmvar` *Switch from two-regime reduced form GMVAR model to a structural GMVAR model.*

Description

`gmvar_to_sgmvar` constructs SGMVAR model based on a reduced form GMVAR model.

Usage

```
gmvar_to_sgmvar(gmvar, calc_std_errors = TRUE)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.
`calc_std_errors`	should approximate standard errors be calculated?

Details

The switch is made by simultaneously diagonalizing the two error term covariance matrices with a well known matrix decomposition (Muirhead, 1982, Theorem A9.9) and then normalizing the diagonal of the matrix W positive (which implies positive diagonal of the B-matrix). Models with more that two regimes are not supported because the matrix decomposition does not generally exists for more than two covariance matrices. If the model has only one regime (= regular SVAR model), a symmetric and pos. def. square root matrix of the error term covariance matrix is used.

The columns of W as well as the lambda parameters can be re-ordered (without changing the implied reduced form model) afterwards with the function `reorder_W_columns`. Also all signs in any column of W can be swapped (without changing the implied reduced form model) afterwards with the function `swap_W_signs`. These two functions work with models containing any number of regimes.

Value

Returns an object of class `'gmvar'` defining a structural GMVAR model based on a two-regime reduced form GMVAR model with the main diagonal of the B-matrix normalized to be positive.

References

- Muirhead R.J. 1982. Aspects of Multivariate Statistical Theory, *Wiley*.
- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, GIRF, reorder_W_columns, swap_W_signs

Examples

```
# Reduced form GMVAR(1,2) model
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319,
 0.005, 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185,
 -0.012, 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)

# Form a structural model based on the reduced form model:
mod12s <- gmvar_to_sgmvar(mod12)
mod12s
```

---

in_paramspace *Determine whether the parameter vector lies in the parameter space*

Description

in_paramspace checks whether the given parameter vector lies in the parameter space. Does NOT test the identification conditions!

Usage

```
in_paramspace(
  p,
  M,
  d,
  params,
  constraints = NULL,
  same_means = NULL,
  structural_pars = NULL,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

Arguments

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

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

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

For same_means models: Should have the form θ= $(\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- μ= $(\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, ψ= $(\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form θ= $(\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- λ_m = $(\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

`posdef_tol` numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

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.

- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

Examples

```
# GMVAR(1,1), d=2 model:
params11 <- 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=params11)

# GMVAR(2,2), d=2 model:
params22 <- 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=params22)

# 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))
params22c <- 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=params22c, constraints=C_mat)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(1.03, 2.36, 1.79, 3, 1.25, 0.06, 0.04, 1.34, -0.29,
 -0.08, -0.05, -0.36, 1.2, 0.05, 0.05, 1.3, -0.3, -0.1, -0.05, -0.4,
  0.89, 0.72, -0.37, 2.16, 7.16, 1.3, 0.37)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
in_paramspace(p=2, M=2, d=2, params=params22s,
  structural_pars=list(W=W_22))
```

---

in_paramspace_int *Determine whether the parameter vector lies in the parameter space*

Description

in_paramspace_int checks whether the parameter vector lies in the parameter space.

Usage

```
in_paramspace_int(
  p,
  M,
  d,
  params,
  all_boldA,
```

```
  alphas,
  all_Omega,
  W_constraints = NULL,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

Arguments

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

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

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

For same_means models: Should have the form $\theta = (\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- $\mu = (\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, $\psi = (\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

`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`.

`W_constraints` set `NULL` for reduced form models. For structural models, this should be the constraint matrix W from the list of structural parameters.

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

`posdef_tol` numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

The parameter vector in the argument `params` should be unconstrained and it is used for structural models only.

Value

Returns `TRUE` if the given parameter values are in the parameter space and `FALSE` otherwise. This function 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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

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

Description

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

Usage

```
iterate_more(
  gmvar,
  maxit = 100,
  calc_std_errors = TRUE,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.
`maxit`	the maximum number of iterations in the variable metric algorithm.
`calc_std_errors`	should approximate standard errors be calculated?
`stat_tol`	numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.
`posdef_tol`	numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

The 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`. `iterate_more` is essentially a wrapper around the function `optim` from the package `stats` and `GMVAR` from the package `gmvarkit`.

Value

Returns an object of class `'gmvar'` defining the estimated GMVAR model.

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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, optim, profile_logliks, update_numtols

Examples

```
## These are long running examples that use parallel computing!
## Running the below examples takes approximately 2 minutes

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

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

---

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

Description

`loglikelihood` computes log-likelihood of a GMVAR model using parameter vector instead of an object of class 'gmvar'. Exists for convenience if one wants to for example employ other estimation algorithms than the ones 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,
  same_means = NULL,
  structural_pars = NULL,
  minval = NA,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

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 order of the model.
M	a positive integer specifying the number of mixture components.
params	a real valued vector specifying the parameter values.

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

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

For same_means models: Should have the form $\theta = (\mu, \psi, \sigma_1, ..., \sigma_M, \alpha_1, ..., \alpha_{M-1})$, where

- $\mu = (\mu_1, ..., \mu_g)$ where μ_i is the mean parameter for group i and g is the number of groups.
- If AR constraints are employed, ψ is as for constrained models, and if AR constraints are not employed, $\psi = (\phi_1, ..., \phi_M)$.

For structural GMVAR model: Should have the form $\theta = (\phi_{1,0}, ..., \phi_{M,0}, \phi_1, ..., \phi_M, vec(W), \lambda_2, ..., \lambda_M$ where

- $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ contains the eigenvalues of the mth mixture component.

If AR parameters are constrained: Replace $\phi_1, ..., \phi_M$ with ψ $(qx1)$ that satisfies $(\phi_1, ..., \phi_M) = C\psi$, as above.

If same_means: Replace $(\phi_{1,0}, ..., \phi_{M,0})$ with $(\mu_1, ..., \mu_g)$, as above.

If W is constrained: Remove the zeros from $vec(W)$ and make sure the other entries satisfy the sign constraints.

If λ_{mi} are constrained: Replace $\lambda_2, ..., \lambda_M$ with γ $(rx1)$ that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where C_λ is a $(d(M-1)xr)$ constraint matrix.

Above, $\phi_{m,0}$ is the intercept parameter, $A_{m,i}$ denotes the ith coefficient matrix of the mth mixture component, Ω_m denotes the error term covariance matrix of the m:th mixture component, and α_m is the mixing weight parameter. The W and λ_{mi} are structural parameters replacing the error term covariance

matrices (see Virolainen, 2020). If $M = 1$, α_m and λ_{mi} are dropped. 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 columns of a given matrix from the principal diagonal downwards (including elements on the diagonal) into a vector. The notation is in line with the cited article by *Kalliovirta, Meitz and Saikkonen (2016)* introducing the GMVAR model.

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

`parametrization`

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

`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 related 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.

`same_means` Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda\gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

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

`stat_tol` numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that `1 -stat_tol` the model is classified as

non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

`posdef_tol` numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

`loglikelihood_int` takes use of the function `dmvn` from the package `mvnfast`.

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.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

`fitGMVAR`, `GMVAR`, `calc_gradient`

Examples

```
# GMVAR(2, 2), d=2 model;
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
loglikelihood(data=gdpdef, p=2, M=2, params=params22)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
loglikelihood(data=gdpdef, p=2, M=2, params=params22s, structural_pars=list(W=W_22))
```

`LR_test` *Perform likelihood ratio test for a GMVAR or SGMVAR model*

Description

`LR_test` performs a likelihood ratio test for a GMVAR or SGMVAR model

Usage

```
LR_test(gmvar1, gmvar2)
```

Arguments

`gmvar1`	an object of class `'gmvar'` generated by `fitGMVAR` or `GMVAR`, containing the **freely estimated** model.
`gmvar2`	an object of class `'gmvar'` generated by `fitGMVAR` or `GMVAR`, containing the **constrained** model.

Details

Performs a likelihood ratio test, testing the null hypothesis that the true parameter value lies in the constrained parameter space. Under the null, the test statistic is asymptotically χ^2-distributed with k degrees of freedom, k being the difference in the dimensions of the unconstrained and constrained parameter spaces.

Note that this function does **not** verify that the two models are actually nested.

Value

A list with class "htest" containing the following components:

`statistic`	the value of the likelihood ratio statistics.
`parameter`	the degrees of freedom of the likelihood ratio statistic.
`p.value`	the p-value of the test.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating the type of the test (likelihood ratio test).
`data.name`	a character string giving the names of the supplied models, gsmar1 and gsmar2.
`gmvar1`	the supplied argument gmvar1
`gmvar2`	the supplied argument gmvar2

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

Wald_test, fitGMVAR, GMVAR, diagnostic_plot, profile_logliks, quantile_residual_tests, cond_moment_plot

Examples

```
## These are long running examples that use parallel computing!
## The below examples take around 1 minute to run.

# Structural GMVAR(2, 2), d=2 model with recursive identification
W22 <- matrix(c(1, NA, 0, 1), nrow=2, byrow=FALSE)
fit22s <- fitGMVAR(gdpdef, p=2, M=2, structural_pars=list(W=W22),
                   ncalls=1, seeds=2)

# The same model but the AR coefficients restricted to be the same
# in both regimes:
C_mat <- rbind(diag(2*2^2), diag(2*2^2))
fit22sc <- fitGMVAR(gdpdef, p=2, M=2, constraints=C_mat,
                    structural_pars=list(W=W22), ncalls=1, seeds=1)

# Test the AR constraints with likelihood ratio test:
LR_test(fit22s, fit22sc)
```

---

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, mix_weights = TRUE, add_grid = TRUE)
```

Arguments

x	object of class 'gmvarpred' generated by predict.gmvar.
...	arguments passed to grid which plots grid to the figure.
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)*0.15).
mix_weights	TRUE if forecasts for mixing weights should be plotted, FALSE in not.
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.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

`plot.qrtest` *Quantile residual tests*

Description

`quantile_residual_tests` performs quantile residual tests described by *Kalliovirta and Saikkonen 2010*, testing 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, 3, 6, 12),
  lags_ch = lags_ac,
  nsimu = 1,
  print_res = TRUE,
  stat_tol,
  posdef_tol
)
```

Arguments

`x`	object of class `'qrtest'` generated by the function `quantile_residual_tests)`.
`...`	currently not used.
`digits`	the number of decimals to print
`gmvar`	an object of class `'gmvar'` created with `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 might improve the tests size properties but it increases the computation time.
print_res	should the test results be printed while computing the tests?
stat_tol	numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that 1 -stat_tol the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.
posdef_tol	numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

If the function fails to calculate the tests because of numerical problems and the parameter values are near the border of the parameter space, it might help to use smaller numerical tolerance for the stationarity and positeve definiteness conditions. The numerical tolerance of an existing model can be changed with the function update_numtols or you can set it directly with the arguments stat_tol and posdef_tol.

Value

Returns an object of class 'qrtest' which has its own print method. The returned object is a list containing the 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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, quantile_residuals, GIRF, diagnostic_plot, predict.gmvar, profile_logliks, LR_test, Wald_test, cond_moment_plot, update_numtols

Examples

```
# GMVAR(3,2) model
fit32 <- fitGMVAR(gdpdef, p=3, M=2, ncalls=1, seeds=2)
qrtests32 <- quantile_residual_tests(fit32)
qrtests32
plot(qrtests32)

# Structural GMVAR(1,2) model identified with sign
# constraints and build with hand-specified parameter values.
# Tests based on simulation procedure with nsimu=1000:
params12s <- c(0.55, 0.112, 0.619, 0.173, 0.344, 0.055, -0.009, 0.718,
 0.255, 0.017, -0.136, 0.858, 0.541, 0.057, -0.162, 0.162, 3.623,
 4.726, 0.674)
W_12 <- matrix(c(1, 1, -1, 1), nrow=2)
mod12s <- GMVAR(gdpdef, p=1, M=2, params=params12s,
                structural_pars=list(W=W_12))
qrtests12s <- quantile_residual_tests(mod12s, nsimu=1000)
qrtests12s
```

---

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

Description

Forecast GMVAR process defined as a class 'gmvar' object. The forecasts are computed by performing independent simulations and using the sample medians or means as point forecasts and empirical quantiles as prediction 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,
  pi = c(0.95, 0.8),
  pi_type = c("two-sided", "upper", "lower", "none"),
  pred_type = c("median", "mean", "cond_mean"),
  plot_res = TRUE,
```

```
  mix_weights = TRUE,
  nt
)
```

Arguments

object	an object of class 'gmvar', generated by function fitGMVAR or GMVAR.
...	additional arguments passed to grid (ignored if plot_res==FALSE) which plots grid to the figure.
n_ahead	how many steps ahead should be predicted?
n_simu	to how many independent simulations should the forecast be based on?
pi	a numeric vector specifying the confidence levels of the prediction intervals.
pi_type	should the prediction intervals be "two-sided", "upper", or "lower"?
pred_type	should the prediction be based on sample "median" or "mean"? Or should it be one-step-ahead forecast based on the exact conditional mean ("cond_mean")? Prediction intervals won't be calculated if the exact conditional mean is used.
plot_res	should the results be plotted?
mix_weights	TRUE if forecasts for mixing weights should be plotted, FALSE in not.
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)*0.15).

Value

Returns a class 'gmvarpred' object containing, among the specifications,...

$pred Point forecasts

$pred_int Prediction intervals, as [,,d].

$mix_pred Point forecasts for the mixing weights

mix_pred_int Individual prediction intervals for mixing weights, as [,,m], m=1,..,M.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

GIRF, GFEVD, simulateGMVAR

Examples

```
# GMVAR(2, 2), d=2 model
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
  0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, d=2, params=params22)
p1 <- predict(mod22, n_ahead=10, pred_type="median", n_simu=500)
p1
p2 <- predict(mod22, n_ahead=10, nt=20, lty=1, n_simu=500)
p2
p3 <- predict(mod22, n_ahead=10, pi=c(0.99, 0.90, 0.80, 0.70),
              nt=30, lty=0, n_simu=500)
p3

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s, parametrization="mean",
 structural_pars=list(W=W_22))
p1 <- predict(mod22s, n_ahead=10, n_simu=500)
```

---

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

Description

print.gmvarpred is a 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.
...	currently not used.
digits	the number of decimals to print

Examples

```
# GMVAR(2, 2), d=2 model;
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22)
```

```
pred22 <- predict(mod22, n_ahead=3, plot_res=FALSE)
print(pred22)
print(pred22, 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, ..., digits)
```

Arguments

`x`	object of class 'gmvarsum' generated by `summary.gmvar`.
`...`	currently not used.
`digits`	the number of digits to be printed.

Examples

```
# GMVAR(2, 2), d=2 model;
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22)
sumry22 <- summary(mod22)
print(sumry22)
```

`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	an object of class 'gmvar' created with 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. Note that if the model is intercept parametrized, there won't be standard errors for the unconditional means, and vice versa. Also, there is no standard error for the last mixing weight alpha_M because it is not parametrized.

Note that if linear constraints are imposed and they involve summations or multiplications, then the AR parameter standard errors are printed separately as they don't correspond one-to-one to the model parameter standard errors.

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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

profile_logliks, fitGMVAR, GMVAR, print.gmvar, swap_parametrization

Examples

```
# GMVAR(1,2) model
fit12 <- fitGMVAR(gdpdef, p=1, M=2, ncalls=1, seeds=1)
fit12
print_std_errors(fit12)
```

---

profile_logliks *Plot profile log-likehoods around the estimates*

Description

profile_logliks plots profile log-likelihoods around the estimates.

Usage

```
profile_logliks(
  gmvar,
  which_pars,
  scale = 0.02,
  nrows,
  ncols,
  precision = 200,
  stat_tol = 0.001,
  posdef_tol = 1e-08
)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
which_pars	the profile log-likelihood function of which parameters should be plotted? An integer vector specifying the positions of the parameters in the parameter vector. The parameter vector has the form... **For unconstrained 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. The default is that profile log-likelihood functions for all parameters are plotted.
scale	a numeric scalar specifying the interval plotted for each estimate: the estimate plus-minus abs(scale*estimate).
nrows	how many rows should be in the plot-matrix? The default is max(ceiling(log2(length(which_pars))-1),1).
ncols	how many columns should be in the plot-matrix? The default is ceiling(length(which_pars)/nrows). Note that nrows*ncols should not be smaller than the length of which_pars.
precision	at how many points should each profile log-likelihood be evaluated at?
stat_tol	numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that 1 -stat_tol the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.
posdef_tol	numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller

than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

When the number of parameters is large, it might be better to plot a smaller number of profile log-likelihood functions at a time using the argument `which_pars`.

The red vertical line points the estimate.

Value

Only plots to a graphical device and doesn't return anything.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

`get_soc`, `diagnostic_plot`, `fitGMVAR`, `GMVAR`, `GIRF`, `LR_test`, `Wald_test`, `cond_moment_plot`

Examples

```
# Running all the below examples takes approximately 2 minutes.

# GMVAR(1,2) model
fit12 <- fitGMVAR(gdpdef, p=1, M=2, ncalls=1, seeds=1)
fit12
profile_logliks(fit12)

# Structural GMVAR(1,2) model identified with sign
# constraints: model build based on inaccurate hand-given estimates.
W_122 <- matrix(c(1, 1, -1, 1), nrow=2)
params12s <- c(0.55, 0.11, 0.62, 0.17, 0.34, 0.05, -0.01, 0.72, 0.25,
 0.02, -0.14, 0.86, 0.54, 0.06, -0.16, 0.16, 3.62, 4.73, 0.67)
mod12s <- GMVAR(gdpdef, p=1, M=2, params=params12s,
                structural_pars=list(W=W_122))
profile_logliks(mod12s)
```

quantile_residuals *Calculate multivariate quantile residuals of a GMVAR model*

Description

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

Usage

```
quantile_residuals(gmvar)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.

Value

Returns $((n_obs - p)xd)$ matrix containing the multivariate quantile residuals, j:th column corresponds to 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 "unconditioned 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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, quantile_residual_tests, diagnostic_plot, predict.gmvar, profile_logliks

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005, 0.03,
 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012, 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
quantile_residuals(mod12)

# GMVAR(2,2), d=2 model with mean-parametrization:
params22 <- c(0.869, 0.549, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.576, 1.168, 0.218, 0.02, -0.119,
```

```
 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22, parametrization="mean")
quantile_residuals(mod22)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s, structural_pars=list(W=W_22))
quantile_residuals(mod22s)
```

`random_ind2` *Create somewhat random parameter vector of a 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,
  same_means = NULL,
  structural_pars = NULL,
  mu_scale,
  mu_scale2,
  omega_scale,
  ar_scale = 1,
  W_scale,
  lambda_scale
)
```

Arguments

`p`	a positive integer specifying the autoregressive order of the model.
`M`	a positive integer specifying the number of mixture components.
`d`	the number of time series in the system.
`same_means`	Restrict the mean parameters of some regimes to be the same? Provide a list of numeric vectors such that each numeric vector contains the regimes that should share the common mean parameters. For instance, if `M=3`, the argument `list(1,2:3)` restricts the mean parameters of the second and third regime to be the same but the first regime has freely estimated (unconditional) mean. Ignore or set to `NULL` if mean parameters should not be restricted to be the same

among any regimes. **This constraint is available only for mean parametrized models; that is, when** `parametrization="mean"`**.**

`structural_pars`

If `NULL` a reduced form model is considered. For structural model, should be a list containing the following elements:

- `W` - a (dxd) matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero.
- `C_lambda` - a $(d(M-1)xr)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either **positive** or **zero**. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained.

See Virolainen (2020) for the conditions required to identify the shocks and for the B-matrix as well (it is W times a time-varying diagonal matrix with positive diagonal entries).

`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"`. However, input (in `initpop`) and output (return value) parameter vectors can 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)$resid` This argument is ignored if structural model is considered.

`ar_scale` a positive real number adjusting how large AR parameter values are typically proposed in construction of the initial population: larger value implies larger coefficients (in absolute value). After construction of the initial population, a new scale is drawn from `(0,0.)` uniform distribution in each iteration.

`W_scale` a size $(dx1)$ strictly positive vector partly specifying the scale and variability of the random covariance matrices in random mutations. The elements of the matrix W are drawn independently from such normal distributions that the expectation of the main **diagonal** elements of the first regime's error term covariance matrix $\Omega_1 = WW'$ is `W_scale`. The distribution of Ω_1 will be in some sense like a Wishart distribution but with the columns (elements) of W obeying the given constraints. The constraints are accounted for by setting the element to be always zero if it is subject to a zero constraint and for sign constraints the absolute value or negative the absolute value are taken, and then the variances of the

elements of W are adjusted accordingly. This argument is ignored if reduced form model is considered.

`lambda_scale` a length $M-1$ vector specifying the **standard deviation** of the mean zero normal distribution from which the eigenvalue λ_{mi} parameters are drawn from in random mutations. As the eigenvalues should always be positive, the absolute value is taken. The elements of `lambda_scale` should be strictly positive real numbers with the $m-1$th element giving the degrees of freedom for the mth regime. The expected value of the main **diagonal** elements ij of the mth $(m > 1)$ error term covariance matrix will be `W_scale[i]*(d -n_i)^(-1)*sum(lambdas*ind_fun)` where the $(dx1)$ vector `lambdas` is drawn from the absolute value of the t-distribution, `n_i` is the number of zero constraints in the ith row of W and `ind_fun` is an indicator function that takes the value one iff the ijth element of W is not constrained to zero. Basically, larger lambdas (or smaller degrees of freedom) imply larger variance.

If the lambda parameters are **constrained** with the $(d(M-1)xr)$ constraint matrix C_lambda, then provide a length r vector specifying the standard deviation of the (absolute value of the) mean zero normal distribution each of the γ parameters are drawn from (the γ is a $(rx1)$ vector). The expected value of the main diagonal elements of the covariance matrices then depend on the constraints.

This argument is ignored if $M == 1$ or a reduced form model is considered. Default is `rep(3,times=M-1)` if lambdas are not constrained and `rep(3,times=r)` if lambdas are constrained.

As with omega_scale and W_scale, this argument should be adjusted carefully if specified by hand. **NOTE** that if lambdas are constrained in some other way than restricting some of them to be identical, this parameter should be adjusted accordingly in order to the estimation succeed!

Details

The coefficient matrices are generated using the algorithm proposed by Ansley and Kohn (1986) which forces stationarity. It's not clear in detail how `ar_scale` exactly affects the coefficient matrices but larger `ar_scale` seems to result in larger AR coefficients. 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.

Models with AR parameters constrained are not supported!

Value

Returns random mean-parametrized parameter vector that has the same form as the argument `params` in the other functions, for instance, in the function `loglikelihood`.

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.

- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

`redecompose_Omegas` *In the decomposition of the covariance matrices (Muirhead, 1982, Theorem A9.9), change the order of the covariance matrices.*

Description

`redecompose_Omegas` exchanges the order of the covariance matrices in the decomposition of Muirhead (1982, Theorem A9.9) and returns the new decomposition.

Usage

```
redecompose_Omegas(M, d, W, lambdas, perm = 1:M)
```

Arguments

`M`	a positive integer specifying the number of mixture components.
`d`	the number of time series in the system.
`W`	a length `d^2` vector containing the vectorized W matrix.
`lambdas`	a length `d*(M-1)` vector of the form $\lambda_2, ...,\lambda_M$ where $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$
`perm`	a vector of length `M` giving the new order of the covarince matrices (relative to the current order)

Details

We consider the following decomposition of positive definite covariannce matrices: $\Omega_1 = WW'$, $\Omega_m = W\Lambda_m W'$, $m = 2, .., M$ where $\Lambda_m = diag(\lambda_{m1}, ..., \lambda_{md})$ contains the strictly postive eigenvalues of $\Omega_m\Omega_1^{-1}$ and the column of the invertible W are the corresponding eigenvectors. Note that this decomposition does not necessarily exists for $M > 2$.

See Muirhead (1982), Theorem A9.9 for more details on the decomposition and the source code for more details on the reparametrization.

Value

Returns a $d^2 + (M-1) * dx1$ vector of the form `c(vec(new_W),new_lambdas)` where the lambdas parameters are in the regimewise order (first regime 2, then 3, etc) and the "new W" and "new lambdas" are constitute the new decomposition with the order of the covariance matrices given by the argument `perm`. Notice that if the first element of `perm` is one, the W matrix will be the same and the lambdas are just re-ordered.

Note that unparametrized zero elements ARE present in the returned W!

Warning

No argument checks! Does not work with dimension $d = 1$ or with only one mixture component $M = 1$.

References

- Muirhead R.J. 1982. Aspects of Multivariate Statistical Theory, *Wiley*.

Examples

```
d <- 2
M <- 2
Omega1 <- matrix(c(2, 0.5, 0.5, 2), nrow=d)
Omega2 <- matrix(c(1, -0.2, -0.2, 1), nrow=d)

# Decomposition with Omega1 as the first covariance matrix:
decomp1 <- diag_Omegas(Omega1, Omega2)
W <- matrix(decomp1[1:d^2], nrow=d, ncol=d)
lambdas <- decomp1[(d^2 + 1):length(decomp1)]
tcrossprod(W) # = Omega1
W%*%tcrossprod(diag(lambdas), W) # = Omega2

# Reorder the covariance matrices in the decomposition so that now
# the first covariance matrix is Omega2:
decomp2 <- redecompose_Omegas(M=M, d=d, W=as.vector(W), lambdas=lambdas,
                              perm=2:1)
new_W <- matrix(decomp2[1:d^2], nrow=d, ncol=d)
new_lambdas <- decomp2[(d^2 + 1):length(decomp2)]
tcrossprod(new_W) # = Omega2
new_W%*%tcrossprod(diag(new_lambdas), new_W) # = Omega1
```

---

reorder_W_columns *Reorder columns of the W-matrix and lambda parameters of a structural GMVAR model.*

Description

`reorder_W_columns` reorder columns of the W-matrix and lambda parameters of a structural GMVAR model.

Usage

```
reorder_W_columns(gmvar, perm)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.
`perm`	an integer vector of length d specifying the new order of the columns of W. Also lambda parameters of each regime will be reordered accordingly.

Details

The order of the columns of W can be changed without changing the implied reduced form model as long as the order of lambda parameters is also changed accordingly. Note that the constraints imposed on W (or the B-matrix) will also be modified accordingly.

This function does not support models with constraints imposed on the lambda parameters!

Also all signs in any column of W can be swapped (without changing the implied reduced form model) with the function `swap_W_signs` but this obviously also swaps the sign constraints in the corresponding columns of W.

Value

Returns an object of class `'gmvar'` defining a structural GMVAR model with the modified structural parameters and constraints.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

`fitGMVAR`, `GMVAR`, `GIRF`, `gmvar_to_sgmvar`, `swap_W_signs`

Examples

```
# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(p=2, M=2, d=2, params=params22s, structural_pars=list(W=W_22))
mod22s

# The same reduced form model, reordered W and lambda in the structual model:
mod22s_2 <- reorder_W_columns(mod22s, perm=2:1)
mod22s_2
```

`simulateGMVAR` *Simulate from GMVAR process*

Description

`simulateGMVAR` simulates observations from a GMVAR process.

Usage

```
simulateGMVAR(
  gmvar,
  nsimu,
  init_values = NULL,
  ntimes = 1,
  drop = TRUE,
  seed = NULL,
  girf_pars = NULL
)
```

Arguments

gmvar	an object of class 'gmvar' created with 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 drawn from the stationary distribution of the process.
ntimes	how many sets of simulations should be performed?
drop	if TRUE (default) then the components of the returned list are coerced to lower dimension if ntimes==1, i.e., $sample and $mixing_weights will be matrices, and $component will be vector.
seed	set seed for the random number generator?
girf_pars	This argument is used internally in the estimation of generalized impulse response functions (see ?GIRF). You should ignore it.

Details

The argument ntimes is intended for forecasting: a GMVAR process can be forecasted by simulating its possible future values. One can easily perform a large number simulations and calculate the sample quantiles from the simulated values to obtain prediction intervals (see the forecasting example).

Value

If drop==TRUE and ntimes==1 (default): $sample, $component, and $mixing_weights are matrices. Otherwise, returns a list with...

$sample a size (nsimu$xdx$ntimes) array containing the samples: the dimension [t,,] is the time index, the dimension [,d,] indicates the marginal time series, and the dimension [,,i] indicates the i:th set of simulations.

$component a size (nsimu$x$ntimes) matrix containing the information from which mixture component each value was generated from.

$mixing_weights a size (nsimu$xM x$ntimes) array containing the mixing weights corresponding to the sample: the dimension [t,,] is the time index, the dimension [,m,] indicates the regime, and the dimension [,,i] indicates the i:th set of simulations.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

fitGMVAR, GMVAR, diagnostic_plot, predict.gmvar, profile_logliks, quantile_residual_tests, GIRF, GFEVD

Examples

```
# GMVAR(1,2), d=2 process, initial values from the stationary
# distribution
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012, 0.136,
 0.674)
mod12 <- GMVAR(p=1, M=2, d=2, params=params12)
set.seed(1)
sim12 <- simulateGMVAR(mod12, nsimu=500)
plot.ts(sim12$sample)
ts.plot(sim12$mixing_weights, col=c("blue", "red"), lty=2)
plot(sim12$component, type="l")

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s,
 structural_pars=list(W=W_22))
sim22s <- simulateGMVAR(mod22s, nsimu=100)
plot.ts(sim22s$sample)

## FORECASTING EXAMPLE ##
# Forecast 5-steps-ahead, 500 sets of simulations with initial
# values from the data:
# GMVAR(2,2), d=2 model
params22 <- c(0.36, 0.121, 0.223, 0.059, -0.151, 0.395, 0.406, -0.005,
 0.083, 0.299, 0.215, 0.002, 0.03, 0.484, 0.072, 0.218, 0.02, -0.119,
  0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004, 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22)
sim22 <- simulateGMVAR(mod22, nsimu=5, ntimes=500)

# Point forecast + 95% prediction intervals:
apply(sim22$sample, MARGIN=1:2, FUN=quantile, probs=c(0.025, 0.5, 0.972))
```

```
# Similar forecast for the mixing weights:
apply(sim22$mixing_weights, MARGIN=1:2, FUN=quantile,
      probs=c(0.025, 0.5, 0.972))
```

`swap_parametrization` *Swap the parametrization of a GMVAR model*

Description

`swap_parametrization` swaps the parametrization of a GMVAR model to `"mean"` if the current parametrization is `"intercept"`, and vice versa.

Usage

```
swap_parametrization(gmvar)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.

Details

`swap_parametrization` is a 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`, the approximate standard errors are only available for parametrized parameters.

Value

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

Remark that the first autocovariance/correlation matrix in `$uncond_moments` is for the lag zero, the second one for the lag one, etc.

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*.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

`fitGMVAR`, `GMVAR`, `iterate_more`, `update_numtols`

Examples

```
# GMVAR(2, 2), d=2 model with mean-parametrization:
params22 <- c(0.869, 0.549, 0.223, 0.059, -0.151, 0.395, 0.406,
 -0.005, 0.083, 0.299, 0.215, 0.002, 0.03, 0.576, 1.168, 0.218,
 0.02, -0.119, 0.722, 0.093, 0.032, 0.044, 0.191, 1.101, -0.004,
 0.105, 0.58)
mod22 <- GMVAR(gdpdef, p=2, M=2, params=params22, parametrization="mean")
mod22 # mean parametrization

mod22_2 <- swap_parametrization(mod22)
mod22_2 # intercept parametrization


# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(p=2, M=2, d=2, params=params22s, structural_pars=list(W=W_22))
mod22s # intercept parametrization

mod22s_2 <- swap_parametrization(mod22s)
mod22s_2 # mean parametrization
```

`swap_W_signs` *Swap all signs in pointed columns a the W matrix of a structural GMVAR model.*

Description

`swap_W_signs` swaps all signs in pointed columns a the W matrix of a structural GMVAR model. Consequently, signs in the columns of the B-matrix are also swapped accordingly.

Usage

```
swap_W_signs(gmvar, which_to_swap)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.
`which_to_swap`	a numeric vector of length at most d and elemnts in $1, .., d$ specifying the columns of W whose sign should be swapped.

Details

All signs in any column of W can be swapped without changing the implied reduced form model. Consequently, also the signs in the columns of the B-matrix are swapped. Note that the sign constraints imposed on W (or the B-matrix) are also swapped in the corresponding columns accordingly.

Also the order of the columns of W can be changed (without changing the implied reduced form model) as long as the order of lambda parameters is also changed accordingly. This can be done with the function `reorder_W_columns`.

Value

Returns an object of class `'gmvar'` defining a structural GMVAR model with the modified structural parameters and constraints.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

`fitGMVAR`, `GMVAR`, `GIRF`, `reorder_W_columns`, `gmvar_to_sgmvar`

Examples

```
# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(p=2, M=2, d=2, params=params22s, structural_pars=list(W=W_22))
mod22s

# The same reduced form model, with signs in the second column of W swapped:
swap_W_signs(mod22s, which_to_swap=2)

# The same reduced form model, with signs in both column of W swapped:
swap_W_signs(mod22s, which_to_swap=1:2)
```

`uncond_moments` *Calculate the unconditional mean, variance, the first p autocovariances, and the first p autocorrelations of a GMVAR process*

Description

`uncond_moments` calculates the unconditional mean, variance, the first p autocovariances, and the first p autocorrelations of the given GMVAR process.

Usage

```
uncond_moments(gmvar)
```

Arguments

`gmvar`	an object of class `'gmvar'` created with `fitGMVAR` or `GMVAR`.

Details

The unconditional moments are based on the stationary distribution of the process.

Value

Returns a list with three components:

`$uncond_mean` a length d vector containing the unconditional mean of the process.

`$autocovs` an $(dxdxp + 1)$ array containing the lag 0,1,...,p autocovariances of the process. The subset `[,,j]` contains the lag `j-1` autocovariance matrix (lag zero for the variance).

`$autocors` the autocovariance matrices scaled to autocorrelation matrices.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Lütkepohl H. 2005. New Introduction to Multiple Time Series Analysis, *Springer*.
- McElroy T. 2017. Computation of vector ARMA autocovariances. *Statistics and Probability Letters*, **124**, 92-96.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

See Also

Other moment functions: `cond_moments()`, `get_regime_autocovs()`, `get_regime_means()`

Examples

```
# GMVAR(1,2), d=2 model:
params12 <- c(0.55, 0.112, 0.344, 0.055, -0.009, 0.718, 0.319, 0.005,
 0.03, 0.619, 0.173, 0.255, 0.017, -0.136, 0.858, 1.185, -0.012,
 0.136, 0.674)
mod12 <- GMVAR(gdpdef, p=1, M=2, params=params12)
uncond_moments(mod12)

# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
 0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
 0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(gdpdef, p=2, M=2, params=params22s, structural_pars=list(W=W_22))
mod22s
uncond_moments(mod22s)
```

---

update_numtols *Update the stationarity and positive definiteness numerical tolerances of an existing class 'gmvar' model.*

Description

update_numtols updates the stationarity and positive definiteness numerical tolerances of an existing class 'gmvar' model.

Usage

```
update_numtols(gmvar, stat_tol = 0.001, posdef_tol = 1e-08)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
stat_tol	numerical tolerance for stationarity of the AR parameters: if the "bold A" matrix of any regime has eigenvalues larger that 1 -stat_tol the model is classified as non-stationary. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.
posdef_tol	numerical tolerance for positive definiteness of the error term covariance matrices: if the error term covariance matrix of any regime has eigenvalues smaller than this, the model is classified as not satisfying positive definiteness assumption. Note that if the tolerance is too small, numerical evaluation of the log-likelihood might fail and cause error.

Details

All signs in any column of W can be swapped without changing the implied reduced form model. Consequently, also the signs in the columns of the B-matrix are swapped. Note that the sign constraints imposed on W (or the B-matrix) are also swapped in the corresponding columns accordingly.

Also the order of the columns of W can be changed (without changing the implied reduced form model) as long as the order of lambda parameters is also changed accordingly. This can be done with the function `reorder_W_columns`.

Value

Returns an object of class `'gmvar'` defining a structural GMVAR model with the modified structural parameters and constraints.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

`fitGMVAR`, `GMVAR`, `GIRF`, `reorder_W_columns`, `gmvar_to_sgmvar`

Examples

```
# Structural GMVAR(2, 2), d=2 model identified with sign-constraints:
params22s <- c(0.36, 0.121, 0.484, 0.072, 0.223, 0.059, -0.151, 0.395,
  0.406, -0.005, 0.083, 0.299, 0.218, 0.02, -0.119, 0.722, 0.093, 0.032,
  0.044, 0.191, 0.057, 0.172, -0.46, 0.016, 3.518, 5.154, 0.58)
W_22 <- matrix(c(1, 1, -1, 1), nrow=2, byrow=FALSE)
mod22s <- GMVAR(p=2, M=2, d=2, params=params22s, structural_pars=list(W=W_22))
mod22s

# Update numerical tolerances:
mod22s <- update_numtols(mod22s, stat_tol=1e-4, posdef_tol=1e-9)
mod22s # The same model
```

Wald_test *Perform Wald test for a GMVAR or SGMVAR model*

Description

`Wald_test` performs a Wald test for a GMVAR or SGMVAR model

Usage

```
Wald_test(gmvar, A, c, h = 6e-06)
```

Arguments

gmvar	an object of class 'gmvar' created with fitGMVAR or GMVAR.
A	a size (kxn_params) matrix with full row rank specifying part of the null hypothesis where n_params is the number of parameters in the (unconstrained) model. See details for more information.
c	a length k vector specifying part of the null hypothesis. See details for more information.
h	difference used to approximate the derivatives.

Details

Denoting the true parameter value by θ_0, we test the null hypothesis $A\theta_0 = c$. Under the null, the test statistic is asymptotically χ^2-distributed with k (=nrow(A)) degrees of freedom. The parameter θ_0 is assumed to have the same form as in the model supplied in the argument gmvar and it is presented in the documentation of the argument params in the function GMVAR (see ?GMVAR).

Finally, note that this function does **not** check whether the specified constraints are feasible (e.g. whether the implied constrained model would be stationary or have positive definite error term covariance matrices).

Value

A list with class "htest" containing the following components:

statistic	the value of the Wald statistics.
parameter	the degrees of freedom of the Wald statistic.
p.value	the p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating the type of the test (Wald test).
data.name	a character string giving the names of the supplied model, constraint matrix A, and vector c.
gmvar	the supplied argument gmvar.
A	the supplied argument A.
c	the supplied argument c.
h	the supplied argument h.

References

- Kalliovirta L., Meitz M. and Saikkonen P. 2016. Gaussian mixture vector autoregression. *Journal of Econometrics*, **192**, 485-498.
- Virolainen S. 2020. Structural Gaussian mixture vector autoregressive model. Unpublished working paper, available as arXiv:2007.04713.

@keywords internal

See Also

LR_test, fitGMVAR, GMVAR, diagnostic_plot, profile_logliks, quantile_residual_tests, cond_moment_plot

Examples

```
# Structural GMVAR(2, 2), d=2 model with recursive identification
W22 <- matrix(c(1, NA, 0, 1), nrow=2, byrow=FALSE)
fit22s <- fitGMVAR(gdpdef, p=2, M=2, structural_pars=list(W=W22),
                   ncalls=1, seeds=2)
fit22s

# Test whether the lambda parameters (of the second regime) are identical
# (due to the zero constraint, the model is identified under the null):
# fit22s has parameter vector of length 26 with the lambda parameters
# in elements 24 and 25.
A <- matrix(c(rep(0, times=23), 1, -1, 0), nrow=1, ncol=26)
c <- 0
Wald_test(fit22s, A=A, c=c)

# Test whether the off-diagonal elements of the first regime's first
# AR coefficient matrix (A_11) are both zero:
# fit22s has parameter vector of length 26 and the off-diagonal elements
# of the 1st regime's 1st AR coefficient matrix are in the elements 6 and 7.
A <- rbind(c(rep(0, times=5), 1, rep(0, times=20)),
           c(rep(0, times=6), 1, rep(0, times=19)))
c <- c(0, 0)
Wald_test(fit22s, A=A, c=c)
```

Index