Package ‘bsvars’

September 1, 2022

Type Package

Title Bayesian Estimation of Structural Vector Autoregressive Models

Version 1.0.0

Date 2022-08-29

Description Efficient algorithms for Bayesian estimation of Structural Vector Autoregressive (SVAR) models via Markov chain Monte Carlo methods. A wide range of SVAR models is considered, including homo- and heteroskedastic specifications and those with non-normal structural shocks. The heteroskedastic SVAR model setup is similar as in Woźniak & Droumaguet (2015) <doi:10.13140/RG.2.2.19492.55687> and Lütkepohl & Woźniak (2020) <doi:10.1016/j.jedc.2020.103862>. The sampler of the structural matrix follows Waggoner & Zha (2003) <doi:10.1016/S0165-1889(02)00168-9>, whereas that for autoregressive parameters follows Chan, Koop, Yu (2022) <`https://www.joshuachan.org/papers/OISV.pdf`>. The specification of Markov switching heteroskedasticity is inspired by Song & Woźniak (2021) <doi:10.1093/acrefore/9780190625979.013.174>, and that of Stochastic Volatility model by Kastner & Frühwirth-Schnatter (2014) <doi:10.1016/j.csda.2013.01.002>.

License GPL (>= 3)

Maintainer Tomasz Woźniak <`wozniak.tom@pm.me`>

Encoding UTF-8

Imports Rcpp (>= 1.0.7), RcppProgress (>= 0.1), RcppTN, GIGrvg, R6

Suggests tinytest

LinkingTo Rcpp, RcppProgress, RcppArmadillo, RcppTN

RoxygenNote 7.2.1

BugReports `https://github.com/donotdespair/bsvars/issues`

NeedsCompilation yes

Author Tomasz Woźniak [aut, cre] (<`https://orcid.org/0000-0003-2212-2378`>)

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2022-09-01 15:40:06 UTC

R topics documented:

`bsvars-package` *Bayesian Estimation of Structural Vector Autoregressive Models*

Description

Efficient and fast algorithms for Bayesian estimation of Structural Vector Autoregressive (SVAR) models via Markov chain Monte Carlo methods. A wide range of SVAR models is considered, including homo- and heteroskedastic specifications and those with non-normal structural shocks. The heteroskedastic SVAR model setup is similar as in Woźniak & Droumaguet (2015) <doi:10.13140/RG.2.2.19492.55687> and Lütkepohl & Woźniak (2020) <doi:10.1016/j.jedc.2020.103862>. The sampler of the structural matrix follows Waggoner & Zha (2003) ,doi:10.1016/S0165-1889(02)00168-9>, whereas that for autoregressive parameters follows Chan, Koop, Yu (2022) <https://www.joshuachan.org/papers/OISV.pdf>. The specification of Markov switching heteroskedasticity is inspired by Song & Woźniak (2021) <doi:10.1093/acrefore/9780190625979.013.174>, and that of Stochastic Volatility model by Kastner & Frühwirth-Schnatter (2014) <doi:10.1016/j.csda.2013.01.002>.

Details

All the SVAR models in this package are specified by two equations, including the reduced form equation:

$$Y = AX + E$$

where Y is an `NxT` matrix of dependent variables, X is a `KxT` matrix of explanatory variables, E is an `NxT` matrix of reduced form error terms, and A is an `NxK` matrix of autoregressive slope coefficients and parameters on deterministic terms in X.

The structural equation is given by:

$$BE = U$$

where U is an `NxT` matrix of structural form error terms, and B is an `NxN` matrix of contemporaneous relationships.

Finally, all of the models share the following assumptions regarding the structural shocks `U`, namely, joint conditional normality given the past observations collected in matrix `X`, and temporal and contemporaneous independence. The latter implies zero correlations and autocorrelations.

The various SVAR models estimated differ by the specification of structural shocks variances. The different models include:

- homoskedastic model with unit variances
- heteroskedastic model with stationary Markov switching in the variances
- heteroskedastic model with Stochastic Volatility process for variances
- non-normal model with a finite mixture of normal components and component-specific variances
- heteroskedastic model with sparse Markov switching in the variances where the number of heteroskedastic components is estimated
- non-normal model with a sparse mixture of normal components and component-specific variances where the number of heteroskedastic components is estimated

Note

This package is currently in active development. Your comments, suggestions and requests are warmly welcome!

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Woźniak, T., and Droumaguet, M., (2022) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs.

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_sv(10, specification)

# estimate the model
posterior      = estimate_bsvar_sv(50, burn_in$get_last_draw())

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB             # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised
```

estimate_bsvar — *Bayesian estimation of a homoskedastic Structural Vector Autoregression via Gibbs sampler*

Description

Estimates the homoskedastic SVAR using the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2021) for the autoregressive slope parameters A. Additionally, the parameter matrices A and B follow a Minnesota prior and generalised-normal prior distributions respectively with the matrix-specific overall shrinkage parameters estimated using a 3-level hierarchical prior distribution. See section **Details** for the model equations.

Usage

```
estimate_bsvar(S, specification, thin = 10, show_progress = TRUE)
```

Arguments

`S`	a positive integer, the number of posterior draws to be generated
`specification`	an object of class BSVAR generated using the `specify_bsvar$new()` function.
`thin`	a positive integer, specifying the frequency of MCMC output thinning
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Details

The homoskedastic SVAR model is given by the reduced form equation:

$$Y = AX + E$$

where Y is an `NxT` matrix of dependent variables, X is a `KxT` matrix of explanatory variables, E is an `NxT` matrix of reduced form error terms, and A is an `NxK` matrix of autoregressive slope coefficients and parameters on deterministic terms in X.

The structural equation is given by

$$BE = U$$

where U is an `NxT` matrix of structural form error terms, and B is an `NxN` matrix of contemporaneous relationships.

Finally, the structural shocks, `U`, are temporally and contemporaneously independent and jointly normally distributed with zero mean and unit variances.

Value

An object of class PosteriorBSVAR containing the Bayesian estimation output and containing two elements:

`posterior` a list with a collection of `S` draws from the posterior distribution generated via Gibbs sampler containing:

A an `NxKxS` array with the posterior draws for matrix A

B an `NxNxS` array with the posterior draws for matrix B

hyper a `5xS` matrix with the posterior draws for the hyper-parameters of the hierarchical prior distribution

`last_draw` an object of class BSVAR with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Sampling from the generalised-normal full conditional posterior distribution of matrix B is implemented using the Gibbs sampler by:

Waggoner, D.F., and Zha, T., (2003) A Gibbs sampler for structural vector autoregressions. *Journal of Economic Dynamics and Control*, **28**, 349–366, doi:10.1016/S01651889(02)001689.

Sampling from the multivariate normal full conditional posterior distribution of each of the A matrix row is implemented using the sampler by:

Chan, J.C.C., Koop, G, and Yu, X. (2021) Large Order-Invariant Bayesian VARs with Stochastic Volatility.

See Also

`specify_bsvar`, `specify_posterior_bsvar`, `normalise_posterior`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar(50, specification)

# estimate the model
posterior      = estimate_bsvar(100, burn_in$get_last_draw())
```

`estimate_bsvar_mix` *Bayesian estimation of a Structural Vector Autoregression with shocks following a finite mixture of normal components via Gibbs sampler*

Description

Estimates the SVAR with non-normal residuals following a finite `M` mixture of normal distributions proposed by Woźniak & Droumaguet (2022). Implements the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2021) for the autoregressive slope parameters A. Additionally, the parameter matrices A and B follow a Minnesota prior and generalised-normal prior distributions respectively with the matrix-specific overall shrinkage parameters estimated thanks to a 3-level hierarchical prior distribution. The finite mixture of normals model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2022). See section **Details** for the model equations.

Usage

```
estimate_bsvar_mix(S, specification, thin = 10, show_progress = TRUE)
```

Arguments

`S`	a positive integer, the number of posterior draws to be generated
`specification`	an object of class BSVAR-MIX generated using the `specify_bsvar_mix$new()` function.
`thin`	a positive integer, specifying the frequency of MCMC output thinning
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Details

The heteroskedastic SVAR model is given by the reduced form equation:

$$Y = AX + E$$

where Y is an `NxT` matrix of dependent variables, X is a `KxT` matrix of explanatory variables, E is an `NxT` matrix of reduced form error terms, and A is an `NxK` matrix of autoregressive slope coefficients and parameters on deterministic terms in X.

The structural equation is given by

$$BE = U$$

where U is an `NxT` matrix of structural form error terms, and B is an `NxN` matrix of contemporaneous relationships.

Finally, the structural shocks, U, are temporally and contemporaneously independent and finite-mixture of normals distributed with zero mean. The conditional variance of the `n`th shock at time `t` is given by:

$$Var_{t-1}[u_{n.t}] = s^2_{n.s_t}$$

where s_t is a the regime indicator of the regime-specific conditional variances of structural shocks $s^2_{n.s_t}$. In this model, the variances of each of the structural shocks sum to `M`.

The regime indicator s_t is either such that:

- the regime probabilities are non-zero which requires all regimes to have a positive number occurrences over the sample period, or
- sparse with potentially many regimes with zero occurrences over the sample period and in which the number of regimes is estimated.

These model selection also with this respect is made using function `specify_bsvar_mix`.

Value

An object of class PosteriorBSVAR-MIX containing the Bayesian estimation output and containing two elements:

`posterior` a list with a collection of `S` draws from the posterior distribution generated via Gibbs sampler containing:

A an `NxKxS` array with the posterior draws for matrix A

B an `NxNxS` array with the posterior draws for matrix B

hyper a `5xS` matrix with the posterior draws for the hyper-parameters of the hierarchical prior distribution

sigma2 an `NxMxS` array with the posterior draws for the structural shocks conditional variances

PR_TR an `MxMxS` array with the posterior draws for the transition matrix.

xi an `MxTxS` array with the posterior draws for the regime allocation matrix.

pi_0 an `MxS` matrix with the posterior draws for the ergodic probabilities

`last_draw` an object of class BSVAR-MIX with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_mix()`.

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

The model, prior distributions, and estimation algorithms were proposed by

Woźniak, T., and Droumaguet, M., (2022) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Some more analysis on heteroskedastic SVAR models was proposed by:

Lütkepohl, H., and Woźniak, T., (2020) Bayesian Inference for Structural Vector Autoregressions Identified by Markov-Switching Heteroskedasticity. *Journal of Economic Dynamics and Control* **113**, 103862, doi:10.1016/j.jedc.2020.103862.

Sampling from the generalised-normal full conditional posterior distribution of matrix B is implemented using the Gibbs sampler by:

Waggoner, D.F., and Zha, T., (2003) A Gibbs sampler for structural vector autoregressions. *Journal of Economic Dynamics and Control*, **28**, 349–366, doi:10.1016/S01651889(02)001689.

Sampling from the multivariate normal full conditional posterior distribution of each of the A matrix row is implemented using the sampler by:

Chan, J.C.C., Koop, G, and Yu, X. (2021) Large Order-Invariant Bayesian VARs with Stochastic Volatility.

The estimation of the mixture of normals heteroskedasticity closely follows procedures described by:

Song, Y., and Woźniak, T., (2021) Markov Switching. *Oxford Research Encyclopedia of Economics and Finance*, Oxford University Press, doi:10.1093/acrefore/9780190625979.013.174.

and

Frühwirth-Schnatter, S., (2006) Finite Mixture and Markov Switching Models. Springer Series in Statistics. New York: Springer, doi:10.1007/9780387357683.

The sparse model is inspired by:

Malsiner-Walli, G., Frühwirth-Schnatter, S., and Grün, B. (2016) Model-based clustering based on sparse finite Gaussian mixtures. *Statistics and Computing*, **26**(1–2), 303–324, doi:10.1007/s11222-01495002.

The forward-filtering backward-sampling is implemented following the proposal by:

Chib, S. (1996) Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, **75**(1), 79–97, doi:10.1016/03044076(95)017704.

See Also

`specify_bsvar_mix`, `specify_posterior_bsvar_mix`, `normalise_posterior`

Examples

```
# upload data
data(us_fiscal_lsuw)
```

```
# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_mix(50, specification)

# estimate the model
posterior      = estimate_bsvar_mix(100, burn_in$get_last_draw())
```

`estimate_bsvar_msh` *Bayesian estimation of a Structural Vector Autoregression with Markov-switching heteroskedasticity via Gibbs sampler*

Description

Estimates the SVAR with Markov-switching heteroskedasticity with `M` regimes (MS(M)) proposed by Woźniak & Droumaguet (2022). Implements the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2021) for the autoregressive slope parameters A. Additionally, the parameter matrices A and B follow a Minnesota prior and generalised-normal prior distributions respectively with the matrix-specific overall shrinkage parameters estimated thanks to a 3-level hierarchical prior distribution. The MS model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2022). See section **Details** for the model equations.

Usage

```
estimate_bsvar_msh(S, specification, thin = 10, show_progress = TRUE)
```

Arguments

`S`	a positive integer, the number of posterior draws to be generated
`specification`	an object of class BSVAR-MSH generated using the `specify_bsvar_msh$new()` function.
`thin`	a positive integer, specifying the frequency of MCMC output thinning
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Details

The heteroskedastic SVAR model is given by the reduced form equation:

$$Y = AX + E$$

where Y is an `NxT` matrix of dependent variables, X is a `KxT` matrix of explanatory variables, E is an `NxT` matrix of reduced form error terms, and A is an `NxK` matrix of autoregressive slope coefficients and parameters on deterministic terms in `X`.

The structural equation is given by

$$BE = U$$

where U is an `NxT` matrix of structural form error terms, and B is an `NxN` matrix of contemporaneous relationships.

Finally, the structural shocks, U, are temporally and contemporaneously independent and jointly normally distributed with zero mean. The conditional variance of the `n`th shock at time `t` is given by:

$$Var_{t-1}[u_{n.t}] = s^2_{n.s_t}$$

where s_t is a Markov process driving the time-variability of the regime-specific conditional variances of structural shocks $s^2_{n.s_t}$. In this model, the variances of each of the structural shocks sum to `M`.

The Markov process s_t is either:

- stationary, irreducible, and aperiodic which requires all regimes to have a positive number occurrences over the sample period, or
- sparse with potentially many regimes with zero occurrences over the sample period and in which the number of regimes is estimated.

These model selection also with this respect is made using function `specify_bsvar_msh`.

Value

An object of class PosteriorBSVAR-MSH containing the Bayesian estimation output and containing two elements:

`posterior` a list with a collection of `S` draws from the posterior distribution generated via Gibbs sampler containing:

A an `NxKxS` array with the posterior draws for matrix A

B an `NxNxS` array with the posterior draws for matrix B

hyper a `5xS` matrix with the posterior draws for the hyper-parameters of the hierarchical prior distribution

sigma2 an `NxMxS` array with the posterior draws for the structural shocks conditional variances

PR_TR an `MxMxS` array with the posterior draws for the transition matrix.

xi an `MxTxS` array with the posterior draws for the regime allocation matrix.

pi_0 an `MxS` matrix with the posterior draws for the initial state probabilities

`last_draw` an object of class BSVAR-MSH with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_msh()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

The model, prior distributions, and estimation algorithms were proposed by

Woźniak, T., and Droumaguet, M., (2022) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Some more analysis on heteroskedastic SVAR models was proposed by:

Lütkepohl, H., and Woźniak, T., (2020) Bayesian Inference for Structural Vector Autoregressions Identified by Markov-Switching Heteroskedasticity. *Journal of Economic Dynamics and Control* **113**, 103862, doi:10.1016/j.jedc.2020.103862.

Sampling from the generalised-normal full conditional posterior distribution of matrix B is implemented using the Gibbs sampler by:

Waggoner, D.F., and Zha, T., (2003) A Gibbs sampler for structural vector autoregressions. *Journal of Economic Dynamics and Control*, **28**, 349–366, doi:10.1016/S01651889(02)001689.

Sampling from the multivariate normal full conditional posterior distribution of each of the A matrix row is implemented using the sampler by:

Chan, J.C.C., Koop, G, and Yu, X. (2021) Large Order-Invariant Bayesian VARs with Stochastic Volatility.

The estimation of the Markov-switching heteroskedasticity closely follows procedures described by:

Song, Y., and Woźniak, T., (2021) Markov Switching. *Oxford Research Encyclopedia of Economics and Finance*, Oxford University Press, doi:10.1093/acrefore/9780190625979.013.174.

and

Frühwirth-Schnatter, S., (2006) Finite Mixture and Markov Switching Models. Springer Series in Statistics. New York: Springer, doi:10.1007/9780387357683.

The sparse model is inspired by:

Malsiner-Walli, G., Frühwirth-Schnatter, S., and Grün, B. (2016) Model-based clustering based on sparse finite Gaussian mixtures. *Statistics and Computing*, **26**(1–2), 303–324, doi:10.1007/s11222-01495002.

The forward-filtering backward-sampling is implemented following the proposal by:

Chib, S. (1996) Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, **75**(1), 79–97, doi:10.1016/03044076(95)017704.

See Also

`specify_bsvar_msh`, `specify_posterior_bsvar_msh`, `normalise_posterior`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)
```

```
# run the burn-in
burn_in        = estimate_bsvar_msh(50, specification)

# estimate the model
posterior      = estimate_bsvar_msh(100, burn_in$get_last_draw())
```

`estimate_bsvar_sv` *Bayesian estimation of a Structural Vector Autoregression with Stochastic Volatility heteroskedasticity via Gibbs sampler*

Description

Estimates the SVAR with Stochastic Volatility (SV) heteroskedasticity proposed by Lütkepohl, Shang, Uzeda, and Woźniak (2022). Implements the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2021) for the autoregressive slope parameters A. Additionally, the parameter matrices A and B follow a Minnesota prior and generalised-normal prior distributions respectively with the matrix-specific overall shrinkage parameters estimated thanks to a 3-level hierarchical prior distribution. The SV model is estimated in a non-centred parameterisation using a range of techniques including: simulation smoother, auxiliary mixture, ancillarity-sufficiency interweaving strategy, and generalised inverse Gaussian distribution summarised by Kastner & Frühwirth-Schnatter (2014). See section **Details** for the model equations.

Usage

```
estimate_bsvar_sv(S, specification, thin = 10, show_progress = TRUE)
```

Arguments

`S`	a positive integer, the number of posterior draws to be generated
`specification`	an object of class BSVAR-SV generated using the `specify_bsvar_sv$new()` function.
`thin`	a positive integer, specifying the frequency of MCMC output thinning
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Details

The heteroskedastic SVAR model is given by the reduced form equation:

$$Y = AX + E$$

where Y is an `NxT` matrix of dependent variables, X is a `KxT` matrix of explanatory variables, E is an `NxT` matrix of reduced form error terms, and A is an `NxK` matrix of autoregressive slope coefficients and parameters on deterministic terms in X.

The structural equation is given by

$$BE = U$$

where U is an `NxT` matrix of structural form error terms, and B is an `NxN` matrix of contemporaneous relationships.

Finally, the structural shocks, U, are temporally and contemporaneously independent and jointly normally distributed with zero mean. The conditional variance of the `n`th shock at time `t` is given by:

$$Var_{t-1}[u_{n.t}] = exp(w_n h_{n.t})$$

where w_n is the estimated conditional standard deviation of the log-conditional variance and the log-volatility process $h_{n.t}$ follows an autoregressive process:

$$h_{n.t} = g_n h_{n.t-1} + v_{n.t}$$

where $h_{n.0} = 0$, g_n is an autoregressive parameter and $v_{n.t}$ is a standard normal error term.

Value

An object of class PosteriorBSVAR-SV containing the Bayesian estimation output and containing two elements:

`posterior` a list with a collection of `S` draws from the posterior distribution generated via Gibbs sampler containing:

A an `NxKxS` array with the posterior draws for matrix A

B an `NxNxS` array with the posterior draws for matrix B

hyper a `5xS` matrix with the posterior draws for the hyper-parameters of the hierarchical prior distribution

h an `NxTxS` array with the posterior draws of the log-volatility processes

rho an `NxS` matrix with the posterior draws of SV autoregressive parameters

omega an `NxS` matrix with the posterior draws of SV process conditional standard deviations

S an `NxTxS` array with the posterior draws of the auxiliary mixture component indicators

sigma2_omega an `NxS` matrix with the posterior draws of the variances of the zero-mean normal prior for `omega`

s_ an `S`-vector with the posterior draws of the scale of the gamma prior of the hierarchical prior for `sigma2_omega`

`last_draw` an object of class BSVAR-SV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_sv()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

The model, prior distributions, and estimation algorithms were proposed by

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2022) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference.

Sampling from the generalised-normal full conditional posterior distribution of matrix B is implemented using the Gibbs sampler by:

Waggoner, D.F., and Zha, T., (2003) A Gibbs sampler for structural vector autoregressions. *Journal of Economic Dynamics and Control*, **28**, 349–366, doi:10.1016/S01651889(02)001689.

Sampling from the multivariate normal full conditional posterior distribution of each of the A matrix row is implemented using the sampler by:

Chan, J.C.C., Koop, G, and Yu, X. (2021) Large Order-Invariant Bayesian VARs with Stochastic Volatility.

Many of the techniques employed for the estimation of the Stochastic Volatility model are summarised by:

Kastner, G. and Frühwirth-Schnatter, S. (2014) Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Estimation of Stochastic Volatility Models. *Computational Statistics & Data Analysis*, **76**, 408–423, doi:10.1016/j.csda.2013.01.002.

See Also

specify_bsvar_sv, specify_posterior_bsvar_sv, normalise_posterior

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_sv(10, specification)

# estimate the model
posterior      = estimate_bsvar_sv(50, burn_in$get_last_draw())
```

normalise_posterior — *Waggoner & Zha (2003) row signs normalisation of the posterior draws for matrix B*

Description

Normalises the sign of rows of matrix B MCMC draws, provided as the first argument `posterior_B`, relative to matrix `B_hat`, provided as the second argument of the function. The implemented procedure proposed by Waggoner, Zha (2003) normalises the MCMC output in an optimal way leading to the unimodal posterior. Only normalised MCMC output is suitable for the computations of the posterior characteristics of the B matrix elements and their functions such as the impulse response functions and other economically interpretable values.

Usage

```
normalise_posterior(posterior, B_hat)
```

Arguments

posterior	posterior estimation outcome - an object of either of classes: PosteriorBSVAR, PosteriorBSVAR-MSH, PosteriorBSVAR-MIX, or PosteriorBSVAR-SV containing, amongst other draws, the `S` draws from the posterior distribution of the `NxN` matrix of contemporaneous relationships B. These draws are to be normalised with respect to:
B_hat	an `NxN` matrix specified by the user to have the desired row signs

Value

Nothing. The normalised elements overwrite the corresponding elements of the first argument `posterior_B` by reference.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

References

Waggoner, D.F., and Zha, T., (2003) Likelihood Preserving Normalization in Multiple Equation Models. *Journal of Econometrics*, **114**(2), 329–47, doi:10.1016/S03044076(03)000873.

See Also

`estimate_bsvar`, `estimate_bsvar_msh`, `estimate_bsvar_sv`, `estimate_bsvar_mix`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_sv(10, specification)

# estimate the model
posterior      = estimate_bsvar_sv(50, burn_in$get_last_draw())

# normalise the posterior
BB             = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat         = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised
```

`specify_bsvar` *R6 Class representing the specification of the homoskedastic BSVAR model*

Description

The class BSVAR presents complete specification for the homoskedastic bsvar model.

Public fields

`p` a non-negative integer specifying the autoregressive lag order of the model.

`identification` an object IdentificationBSVAR with the identifying restrictions.

`prior` an object PriorBSVAR with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVAR with the starting values.

Methods

Public methods:

- `specify_bsvar$new()`
- `specify_bsvar$get_data_matrices()`
- `specify_bsvar$get_identification()`
- `specify_bsvar$get_prior()`
- `specify_bsvar$get_starting_values()`
- `specify_bsvar$clone()`

Method `new()`**:** Create a new specification of the homoskedastic bsvar model BSVAR.

Usage:

```
specify_bsvar$new(data, p = 1L, B, stationary = rep(FALSE, ncol(data)))
```

Arguments:

`data` a `(T+p)xN` matrix with time series data.

`p` a positive integer providing model's autoregressive lag order.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

Returns: A new complete specification for the homoskedastic bsvar model BSVAR.

Method `get_data_matrices()`**:** Returns the data matrices as the DataMatricesBSVAR object.

Usage:

```
specify_bsvar$get_data_matrices()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_data_matrices()
```

Method `get_identification()`**:** Returns the identifying restrictions as the IdentificationBSVARs object.

Usage:

```
specify_bsvar$get_identification()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_identification()
```

Method `get_prior()`**:** Returns the prior specification as the PriorBSVAR object.

Usage:

```
specify_bsvar$get_prior()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_prior()
```

Method `get_starting_values()`**:** Returns the starting values as the StartingValuesBSVAR object.

Usage:

```
specify_bsvar$get_starting_values()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_starting_values()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
specify_bsvar$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

`estimate_bsvar`, `specify_posterior_bsvar`

Examples

```
data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)


## ------------------------------------------------
## Method `specify_bsvar$get_data_matrices`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_data_matrices()


## ------------------------------------------------
## Method `specify_bsvar$get_identification`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_identification()


## ------------------------------------------------
## Method `specify_bsvar$get_prior`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_prior()


## ------------------------------------------------
## Method `specify_bsvar$get_starting_values`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_starting_values()
```

`specify_bsvar_mix` *R6 Class representing the specification of the BSVAR model with a zero-mean mixture of normals model for structural shocks.*

Description

The class BSVAR-MIX presents complete specification for the BSVAR model with a zero-mean mixture of normals model for structural shocks.

Super class

`bsvars::BSVAR-MSH` -> `BSVAR-MIX`

Public fields

`p` a non-negative integer specifying the autoregressive lag order of the model.

`identification` an object IdentificationBSVARs with the identifying restrictions.

`prior` an object PriorBSVAR-MIX with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVAR-MIX with the starting values.

`finiteM` a logical value - if true a finite mixture model is estimated. Otherwise, a sparse mixture model is estimated in which `M=20` and the number of visited states is estimated.

Methods

Public methods:

- `specify_bsvar_mix$new()`
- `specify_bsvar_mix$clone()`

Method `new()`**:** Create a new specification of the BSVAR model with a zero-mean mixture of normals model for structural shocks, BSVAR-MIX.

Usage:

```
specify_bsvar_mix$new(
  data,
  p = 1L,
  M,
  B,
  stationary = rep(FALSE, ncol(data)),
  finiteM = TRUE
)
```

Arguments:

`data` a `(T+p)xN` matrix with time series data.

`p` a positive integer providing model's autoregressive lag order.

`M` an integer greater than 1 - the number of components of the mixture of normals.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

`finiteM` a logical value - if true a finite mixture model is estimated. Otherwise, a sparse mixture model is estimated in which `M=20` and the number of visited states is estimated.

Returns: A new complete specification for the bsvar model with a zero-mean mixture of normals model for structural shocks, BSVAR-MIX.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
specify_bsvar_mix$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

`estimate_bsvar_mix`, `specify_posterior_bsvar_mix`

Examples

```
data(us_fiscal_lsuw)
spec = specify_bsvar_mix$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
```

`specify_bsvar_msh` *R6 Class representing the specification of the BSVAR model with Markov Switching Heteroskedasticity.*

Description

The class BSVAR-MSH presents complete specification for the BSVAR model with Markov Switching Heteroskedasticity.

Public fields

`p` a non-negative integer specifying the autoregressive lag order of the model.

`identification` an object IdentificationBSVARs with the identifying restrictions.

`prior` an object PriorBSVAR-MSH with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVAR-MSH with the starting values.

`finiteM` a logical value - if true a stationary Markov switching model is estimated. Otherwise, a sparse Markov switching model is estimated in which `M=20` and the number of visited states is estimated.

Methods

Public methods:

- `specify_bsvar_msh$new()`
- `specify_bsvar_msh$get_data_matrices()`
- `specify_bsvar_msh$get_identification()`
- `specify_bsvar_msh$get_prior()`
- `specify_bsvar_msh$get_starting_values()`
- `specify_bsvar_msh$clone()`

Method `new()`: Create a new specification of the BSVAR model with Markov Switching Heteroskedasticity, BSVAR-MSH.

Usage:

```
specify_bsvar_msh$new(
  data,
  p = 1L,
  M,
  B,
  stationary = rep(FALSE, ncol(data)),
  finiteM = TRUE
)
```

Arguments:

`data` a `(T+p)xN` matrix with time series data.

`p` a positive integer providing model's autoregressive lag order.

`M` an integer greater than 1 - the number of Markov process' heteroskedastic regimes.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

`finiteM` a logical value - if true a stationary Markov switching model is estimated. Otherwise, a sparse Markov switching model is estimated in which `M=20` and the number of visited states is estimated.

Returns: A new complete specification for the bsvar model with Markov Switching Heteroskedasticity, BSVAR-MSH.

Method `get_data_matrices()`**:** Returns the data matrices as the DataMatricesBSVAR object.

Usage:

```
specify_bsvar_msh$get_data_matrices()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_data_matrices()
```

Method `get_identification()`**:** Returns the identifying restrictions as the IdentificationBSVARs object.

Usage:

```
specify_bsvar_msh$get_identification()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_identification()
```

Method `get_prior()`**:** Returns the prior specification as the PriorBSVAR-MSH object.

Usage:

```
specify_bsvar_msh$get_prior()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_prior()
```

Method `get_starting_values()`**:** Returns the starting values as the StartingValuesBSVAR-MSH object.

Usage:

```
specify_bsvar_msh$get_starting_values()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_starting_values()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_bsvar_msh$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

`estimate_bsvar_msh`, `specify_posterior_bsvar_msh`

Examples

```
data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)


## ------------------------------------------------
## Method `specify_bsvar_msh$get_data_matrices`
## ------------------------------------------------

data(us_fiscal_lsuw)
```

```
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_data_matrices()


## ------------------------------------------------
## Method `specify_bsvar_msh$get_identification`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_identification()


## ------------------------------------------------
## Method `specify_bsvar_msh$get_prior`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_prior()


## ------------------------------------------------
## Method `specify_bsvar_msh$get_starting_values`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_msh$new(
   data = us_fiscal_lsuw,
   p = 4,
   M = 2
)
spec$get_starting_values()
```

`specify_bsvar_sv` *R6 Class representing the specification of the BSVAR model with Stochastic Volatility heteroskedasticity.*

Description

The class BSVAR-SV presents complete specification for the BSVAR model with Stochastic Volatility heteroskedasticity.

Public fields

`p` a non-negative integer specifying the autoregressive lag order of the model.

`identification` an object IdentificationBSVARs with the identifying restrictions.

`prior` an object PriorBSVAR-SV with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVAR-SV with the starting values.

Methods

Public methods:

- `specify_bsvar_sv$new()`
- `specify_bsvar_sv$get_data_matrices()`
- `specify_bsvar_sv$get_identification()`
- `specify_bsvar_sv$get_prior()`
- `specify_bsvar_sv$get_starting_values()`
- `specify_bsvar_sv$clone()`

Method `new()`**:** Create a new specification of the BSVAR model with Stochastic Volatility heteroskedasticity, BSVAR-SV.

Usage:

```
specify_bsvar_sv$new(data, p = 1L, B, stationary = rep(FALSE, ncol(data)))
```

Arguments:

`data` a `(T+p)xN` matrix with time series data.

`p` a positive integer providing model's autoregressive lag order.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

Returns: A new complete specification for the bsvar model with Stochastic Volatility heteroskedasticity, BSVAR-SV.

Method `get_data_matrices()`**:** Returns the data matrices as the DataMatricesBSVAR object.

Usage:

```
specify_bsvar_sv$get_data_matrices()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_data_matrices()
```

Method `get_identification()`**:** Returns the identifying restrictions as the IdentificationBSVARs object.

Usage:

```
specify_bsvar_sv$get_identification()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_identification()
```

Method `get_prior()`**:** Returns the prior specification as the PriorBSVAR-SV object.

Usage:

```
specify_bsvar_sv$get_prior()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_prior()
```

Method `get_starting_values()`**:** Returns the starting values as the StartingValuesBSVAR-SV object.

Usage:

```
specify_bsvar_sv$get_starting_values()
```

Examples:

```
data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_starting_values()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_bsvar_sv$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

estimate_bsvar_sv, specify_posterior_bsvar_sv

Examples

```
data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)


## ------------------------------------------------
## Method `specify_bsvar_sv$get_data_matrices`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_data_matrices()


## ------------------------------------------------
## Method `specify_bsvar_sv$get_identification`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_identification()


## ------------------------------------------------
## Method `specify_bsvar_sv$get_prior`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_prior()
```

```
## ------------------------------------------------
## Method `specify_bsvar_sv$get_starting_values`
## ------------------------------------------------

data(us_fiscal_lsuw)
spec = specify_bsvar_sv$new(
   data = us_fiscal_lsuw,
   p = 4
)
spec$get_starting_values()
```

`specify_data_matrices` *R6 Class Representing DataMatricesBSVAR*

Description

The class DataMatricesBSVAR presents the data matrices of dependent variables, Y, and regressors, X, for the homoskedastic bsvar model.

Public fields

`Y` an `NxT` matrix of dependent variables, Y.

`X` an `KxT` matrix of regressors, X.

Methods

Public methods:

- `specify_data_matrices$new()`
- `specify_data_matrices$get_data_matrices()`
- `specify_data_matrices$clone()`

Method `new()`**:** Create new data matrices DataMatricesBSVAR.

Usage:

`specify_data_matrices$new(data, p = 1L)`

Arguments:

`data` a `(T+p)xN` matrix with time series data.

`p` a positive integer providing model's autoregressive lag order.

Returns: New data matrices DataMatricesBSVAR.

Method `get_data_matrices()`**:** Returns the data matrices DataMatricesBSVAR as a `list`.

Usage:

`specify_data_matrices$get_data_matrices()`

Examples:

```
data(us_fiscal_lsuw)
YX = specify_data_matrices$new(data = us_fiscal_lsuw, p = 4)
YX$get_data_matrices()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_data_matrices$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
data(us_fiscal_lsuw)
YX = specify_data_matrices$new(data = us_fiscal_lsuw, p = 4)
dim(YX$Y); dim(YX$X)


## ------------------------------------------------
## Method `specify_data_matrices$get_data_matrices`
## ------------------------------------------------

data(us_fiscal_lsuw)
YX = specify_data_matrices$new(data = us_fiscal_lsuw, p = 4)
YX$get_data_matrices()
```

`specify_identification_bsvars`

R6 Class Representing IdentificationBSVARs

Description

The class IdentificationBSVARs presents the identifying restrictions for the bsvar models.

Public fields

`VB` a list of `N` matrices determining the unrestricted elements of matrix B.

Methods

Public methods:

- `specify_identification_bsvars$new()`
- `specify_identification_bsvars$get_identification()`
- `specify_identification_bsvars$set_identification()`

- specify_identification_bsvars$clone()

Method `new()`: Create new identifying restrictions IdentificationBSVARs.

Usage:

```
specify_identification_bsvars$new(N, B)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

Returns: Identifying restrictions IdentificationBSVARs.

Method `get_identification()`: Returns the elements of the identification pattern IdentificationBSVARs as a `list`.

Usage:

```
specify_identification_bsvars$get_identification()
```

Examples:

```
B    = matrix(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE), 3, 3); B
spec = specify_identification_bsvars$new(N = 3, B = B)
spec$get_identification()
```

Method `set_identification()`: Set new starting values StartingValuesBSVAR.

Usage:

```
specify_identification_bsvars$set_identification(N, B)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`B` a logical `NxN` matrix containing value `TRUE` for the elements of the structural matrix B to be estimated and value `FALSE` for exclusion restrictions to be set to zero.

Examples:

```
spec = specify_identification_bsvars$new(N = 3) # specify a model with the default option
B    = matrix(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE), 3, 3); B
spec$set_identification(N = 3, B = B)  # modify an existing specification
spec$get_identification()              # check the outcome
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
specify_identification_bsvars$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
specify_identification_bsvars$new(N = 3) # recursive specification for a 3-variable system

B = matrix(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE), 3, 3); B
specify_identification_bsvars$new(N = 3, B = B) # an alternative identification pattern


## ------------------------------------------------
## Method `specify_identification_bsvars$get_identification`
## ------------------------------------------------

B    = matrix(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE), 3, 3); B
spec = specify_identification_bsvars$new(N = 3, B = B)
spec$get_identification()


## ------------------------------------------------
## Method `specify_identification_bsvars$set_identification`
## ------------------------------------------------

spec = specify_identification_bsvars$new(N = 3) # specify a model with the default option
B    = matrix(c(TRUE,TRUE,TRUE,FALSE,FALSE,TRUE,FALSE,TRUE,TRUE), 3, 3); B
spec$set_identification(N = 3, B = B)  # modify an existing specification
spec$get_identification()              # check the outcome
```

`specify_posterior_bsvar` *R6 Class Representing PosteriorBSVAR*

Description

The class PosteriorBSVAR contains posterior output and the specification including the last MCMC draw for the homoskedastic bsvar model. Note that due to the thinning of the MCMC output the starting value in element `last_draw` might not be equal to the last draw provided in element `posterior`.

Public fields

`last_draw` an object of class BSVAR with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar()`.

`posterior` a list containing Bayesian estimation output collected in elements an `NxNxS` array `B`, an `NxKxS` array `A`, and a `5xS` matrix `hyper`.

Methods

Public methods:

- `specify_posterior_bsvar$new()`
- `specify_posterior_bsvar$get_posterior()`

- specify_posterior_bsvar$get_last_draw()
- specify_posterior_bsvar$is_normalised()
- specify_posterior_bsvar$set_normalised()
- specify_posterior_bsvar$clone()

Method `new()`**:** Create a new posterior output PosteriorBSVAR.

Usage:

```
specify_posterior_bsvar$new(specification_bsvar, posterior_bsvar)
```

Arguments:

`specification_bsvar` an object of class BSVAR with the last draw of the current MCMC run as the starting value.

`posterior_bsvar` a list containing Bayesian estimation output collected in elements an `NxNxS` array `B`, an `NxKxS` array `A`, and a `5xS` matrix `hyper`.

Returns: A posterior output PosteriorBSVAR.

Method `get_posterior()`**:** Returns a list containing Bayesian estimation output collected in elements an `NxNxS` array `B`, an `NxKxS` array `A`, and a `5xS` matrix `hyper`.

Usage:

```
specify_posterior_bsvar$get_posterior()
```

Examples:

```
data(us_fiscal_lsuw)
specification  = specify_bsvar$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate_bsvar(50, specification)
estimate$get_posterior()
```

Method `get_last_draw()`**:** Returns an object of class BSVAR with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar()`.

Usage:

```
specify_posterior_bsvar$get_last_draw()
```

Examples:

```
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar(10, specification)

# get the last draw
last_draw      = burn_in$get_last_draw()
```

```
# estimate the model
posterior      = estimate_bsvar(10, last_draw)
```

Method `is_normalised()`**:** Returns `TRUE` if the posterior has been normalised using `normalise_posterior()` and `FALSE` otherwise.

Usage:

```
specify_posterior_bsvar$is_normalised()
```

Examples:

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB          = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat       = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `set_normalised()`**:** Sets the private indicator `normalised` to TRUE.

Usage:

```
specify_posterior_bsvar$set_normalised(value)
```

Arguments:

`value` (optional) a logical value to be passed to indicator `normalised`.

Examples:

```
# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)
```

```
# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB         = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`specify_posterior_bsvar$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

`estimate_bsvar`, `specify_bsvar`

Examples

```
# This is a function that is used within estimate_bsvar()
data(us_fiscal_lsuw)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)
estimate       = estimate_bsvar(50, specification)
class(estimate)


## ------------------------------------------------
## Method `specify_posterior_bsvar$get_posterior`
## ------------------------------------------------

data(us_fiscal_lsuw)
specification  = specify_bsvar$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate_bsvar(50, specification)
estimate$get_posterior()


## ------------------------------------------------
## Method `specify_posterior_bsvar$get_last_draw`
## ------------------------------------------------
```

```
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar(10, specification)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar(10, last_draw)


## ------------------------------------------------
## Method `specify_posterior_bsvar$is_normalised`
## ------------------------------------------------

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()


## ------------------------------------------------
## Method `specify_posterior_bsvar$set_normalised`
## ------------------------------------------------

# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)
```

```
# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

---

specify_posterior_bsvar_mix

R6 Class Representing PosteriorBSVAR-MIX

Description

The class PosteriorBSVAR-MIX contains posterior output and the specification including the last MCMC draw for the bsvar model with a zero-mean mixture of normals model for structural shocks. Note that due to the thinning of the MCMC output the starting value in element `last_draw` might not be equal to the last draw provided in element `posterior`.

Public fields

`last_draw` an object of class BSVAR-MIX with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_mix()`.

`posterior` a list containing Bayesian estimation output.

Methods

Public methods:

- `specify_posterior_bsvar_mix$new()`
- `specify_posterior_bsvar_mix$get_posterior()`
- `specify_posterior_bsvar_mix$get_last_draw()`
- `specify_posterior_bsvar_mix$is_normalised()`
- `specify_posterior_bsvar_mix$set_normalised()`
- `specify_posterior_bsvar_mix$clone()`

Method `new()`: Create a new posterior output PosteriorBSVAR-MIX.

Usage:

```
specify_posterior_bsvar_mix$new(specification_bsvar, posterior_bsvar)
```

Arguments:

`specification_bsvar` an object of class BSVAR-MIX with the last draw of the current MCMC run as the starting value.

`posterior_bsvar` a list containing Bayesian estimation output.

Returns: A posterior output PosteriorBSVAR-MIX.

Method `get_posterior()`**:** Returns a list containing Bayesian estimation output.

Usage:

```
specify_posterior_bsvar_mix$get_posterior()
```

Examples:

```
data(us_fiscal_lsuw)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, M = 2)
set.seed(123)
estimate       = estimate_bsvar_mix(10, specification, thin = 1)
estimate$get_posterior()
```

Method `get_last_draw()`**:** Returns an object of class BSVAR-MIX with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_mix()`.

Usage:

```
specify_posterior_bsvar_mix$get_last_draw()
```

Examples:

```
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)

# run the burn-in
set.seed(123)
burn_in        = estimate_bsvar_mix(10, specification, thin = 2)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_mix(10, last_draw, thin = 2)
```

Method `is_normalised()`**:** Returns `TRUE` if the posterior has been normalised using `normalise_posterior()` and `FALSE` otherwise.

Usage:

```
specify_posterior_bsvar_mix$is_normalised()
```

Examples:

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_mix(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB          = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat       = diag(sign(diag(BB))) %*% BB          # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `set_normalised()`**:** Sets the private indicator `normalised` to TRUE.

Usage:

```
specify_posterior_bsvar_mix$set_normalised(value)
```

Arguments:

`value` (optional) a logical value to be passed to indicator `normalised`.

Examples:

```
# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB          = posterior$last_draw$starting_values$B     # get the last draw of B
```

```
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_posterior_bsvar_mix$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

`estimate_bsvar_mix`, `specify_bsvar_mix`

Examples

```
# This is a function that is used within estimate_bsvar()
data(us_fiscal_lsuw)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)
estimate       = estimate_bsvar_mix(10, specification, thin = 1)
class(estimate)


## ------------------------------------------------
## Method `specify_posterior_bsvar_mix$get_posterior`
## ------------------------------------------------

data(us_fiscal_lsuw)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, M = 2)
set.seed(123)
estimate       = estimate_bsvar_mix(10, specification, thin = 1)
estimate$get_posterior()


## ------------------------------------------------
## Method `specify_posterior_bsvar_mix$get_last_draw`
## ------------------------------------------------

data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)

# run the burn-in
set.seed(123)
burn_in        = estimate_bsvar_mix(10, specification, thin = 2)
```

```
# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_mix(10, last_draw, thin = 2)


## ------------------------------------------------
## Method `specify_posterior_bsvar_mix$is_normalised`
## ------------------------------------------------

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 4, M = 2)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_mix(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()


## ------------------------------------------------
## Method `specify_posterior_bsvar_mix$set_normalised`
## ------------------------------------------------

# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()
```

```
# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB             # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

---

specify_posterior_bsvar_msh

R6 Class Representing PosteriorBSVAR-MSH

Description

The class PosteriorBSVAR-MSH contains posterior output and the specification including the last MCMC draw for the bsvar model with Markov Switching Heteroskedasticity. Note that due to the thinning of the MCMC output the starting value in element `last_draw` might not be equal to the last draw provided in element `posterior`.

Public fields

`last_draw` an object of class BSVAR-MSH with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_msh()`.

`posterior` a list containing Bayesian estimation output.

Methods

Public methods:

- `specify_posterior_bsvar_msh$new()`
- `specify_posterior_bsvar_msh$get_posterior()`
- `specify_posterior_bsvar_msh$get_last_draw()`
- `specify_posterior_bsvar_msh$is_normalised()`
- `specify_posterior_bsvar_msh$set_normalised()`
- `specify_posterior_bsvar_msh$clone()`

Method `new()`**:** Create a new posterior output PosteriorBSVAR-MSH.

Usage:

`specify_posterior_bsvar_msh$new(specification_bsvar, posterior_bsvar)`

Arguments:

`specification_bsvar` an object of class BSVAR-MSH with the last draw of the current MCMC run as the starting value.

`posterior_bsvar` a list containing Bayesian estimation output.

Returns: A posterior output PosteriorBSVAR-MSH.

Method `get_posterior()`**:** Returns a list containing Bayesian estimation output.

Usage:

```
specify_posterior_bsvar_msh$get_posterior()
```

Examples:

```
data(us_fiscal_lsuw)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, M = 2)
set.seed(123)
estimate       = estimate_bsvar_msh(10, specification, thin = 1)
estimate$get_posterior()
```

Method `get_last_draw()`**:** Returns an object of class BSVAR-MSH with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_msh()`.

Usage:

```
specify_posterior_bsvar_msh$get_last_draw()
```

Examples:

```
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)

# run the burn-in
set.seed(123)
burn_in        = estimate_bsvar_msh(10, specification, thin = 2)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_msh(10, last_draw, thin = 2)
```

Method `is_normalised()`**:** Returns `TRUE` if the posterior has been normalised using `normalise_posterior()` and `FALSE` otherwise.

Usage:

```
specify_posterior_bsvar_msh$is_normalised()
```

Examples:

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)

# estimate the model
set.seed(123)
```

```
posterior      = estimate_bsvar_msh(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB         = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `set_normalised()`**:** Sets the private indicator `normalised` to TRUE.

Usage:

`specify_posterior_bsvar_msh$set_normalised(value)`

Arguments:

`value` (optional) a logical value to be passed to indicator `normalised`.

Examples:

```
# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB         = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_posterior_bsvar_msh$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

estimate_bsvar_msh, specify_bsvar_msh

Examples

```
# This is a function that is used within estimate_bsvar()
data(us_fiscal_lsuw)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)
estimate       = estimate_bsvar_msh(10, specification, thin = 1)
class(estimate)


## ------------------------------------------------
## Method `specify_posterior_bsvar_msh$get_posterior`
## ------------------------------------------------

data(us_fiscal_lsuw)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, M = 2)
set.seed(123)
estimate       = estimate_bsvar_msh(10, specification, thin = 1)
estimate$get_posterior()


## ------------------------------------------------
## Method `specify_posterior_bsvar_msh$get_last_draw`
## ------------------------------------------------

data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)

# run the burn-in
set.seed(123)
burn_in        = estimate_bsvar_msh(10, specification, thin = 2)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_msh(10, last_draw, thin = 2)


## ------------------------------------------------
## Method `specify_posterior_bsvar_msh$is_normalised`
## ------------------------------------------------
```

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 4, M = 2)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_msh(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()


## ------------------------------------------------
## Method `specify_posterior_bsvar_msh$set_normalised`
## ------------------------------------------------

# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# estimate the model
posterior      = estimate_bsvar(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)          # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

`specify_posterior_bsvar_sv` *R6 Class Representing PosteriorBSVAR-SV*

Description

The class PosteriorBSVAR-SV contains posterior output and the specification including the last MCMC draw for the bsvar model with Stochastic Volatility heteroskedasticity. Note that due to the thinning of the MCMC output the starting value in element `last_draw` might not be equal to the last draw provided in element `posterior`.

Public fields

`last_draw` an object of class BSVAR-SV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_sv()`.

`posterior` a list containing Bayesian estimation output.

Methods

Public methods:

- `specify_posterior_bsvar_sv$new()`
- `specify_posterior_bsvar_sv$get_posterior()`
- `specify_posterior_bsvar_sv$get_last_draw()`
- `specify_posterior_bsvar_sv$is_normalised()`
- `specify_posterior_bsvar_sv$set_normalised()`
- `specify_posterior_bsvar_sv$clone()`

Method `new()`**:** Create a new posterior output PosteriorBSVAR-SV.

Usage:

```
specify_posterior_bsvar_sv$new(specification_bsvar, posterior_bsvar)
```

Arguments:

`specification_bsvar` an object of class BSVAR-SV with the last draw of the current MCMC run as the starting value.

`posterior_bsvar` a list containing Bayesian estimation output.

Returns: A posterior output PosteriorBSVAR-SV.

Method `get_posterior()`**:** Returns a list containing Bayesian estimation.

Usage:

```
specify_posterior_bsvar_sv$get_posterior()
```

Examples:

```
data(us_fiscal_lsuw)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate_bsvar_sv(10, specification)
estimate$get_posterior()
```

Method `get_last_draw()`**:** Returns an object of class BSVAR-SV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_sv()`.

Usage:

```
specify_posterior_bsvar_sv$get_last_draw()
```

Examples:

```
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_sv(10, specification)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_sv(10, last_draw)
```

Method `is_normalised()`**:** Returns `TRUE` if the posterior has been normalised using `normalise_posterior()` and `FALSE` otherwise.

Usage:

```
specify_posterior_bsvar_sv$is_normalised()
```

Examples:

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_sv(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()
```

```
# normalise the posterior
BB         = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `set_normalised()`**:** Sets the private indicator `normalised` to TRUE.

Usage:

```
specify_posterior_bsvar_sv$set_normalised(value)
```

Arguments:

`value` (optional) a logical value to be passed to indicator `normalised`.

Examples:

```
# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_sv(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB         = posterior$last_draw$starting_values$B     # get the last draw of B
B_hat      = diag(sign(diag(BB))) %*% BB           # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_posterior_bsvar_sv$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

estimate_bsvar_sv, specify_bsvar_sv

Examples

```
# This is a function that is used within estimate_bsvar()
data(us_fiscal_lsuw)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)
estimate       = estimate_bsvar_sv(10, specification)
class(estimate)


## ------------------------------------------------
## Method `specify_posterior_bsvar_sv$get_posterior`
## ------------------------------------------------

data(us_fiscal_lsuw)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate_bsvar_sv(10, specification)
estimate$get_posterior()


## ------------------------------------------------
## Method `specify_posterior_bsvar_sv$get_last_draw`
## ------------------------------------------------

data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)

# run the burn-in
burn_in        = estimate_bsvar_sv(10, specification)

# get the last draw
last_draw      = burn_in$get_last_draw()

# estimate the model
posterior      = estimate_bsvar_sv(10, last_draw)


## ------------------------------------------------
## Method `specify_posterior_bsvar_sv$is_normalised`
## ------------------------------------------------

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
```

```
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_sv(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()


## ------------------------------------------------
## Method `specify_posterior_bsvar_sv$set_normalised`
## ------------------------------------------------

# This is an internal function that is run while executing normalise_posterior()
# Observe its working by analysing the workflow:

# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)

# estimate the model
set.seed(123)
posterior      = estimate_bsvar_sv(10, specification, thin = 1)

# check normalisation status beforehand
posterior$is_normalised()

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag(sign(diag(BB))) %*% BB              # set positive diagonal elements
bsvars::normalise_posterior(posterior, B_hat)         # draws in posterior are normalised

# check normalisation status afterwards
posterior$is_normalised()
```

`specify_prior_bsvar` *R6 Class Representing PriorBSVAR*

Description

The class PriorBSVAR presents a prior specification for the homoskedastic bsvar model.

Public fields

`A` an `NxK` matrix, the mean of the normal prior distribution for the parameter matrix A.

`A_V_inv` a `KxK` precision matrix of the normal prior distribution for each of the row of the parameter matrix A. This precision matrix is equation invariant.

`B_V_inv` an `NxN` precision matrix of the generalised-normal prior distribution for the structural matrix B. This precision matrix is equation invariant.

`B_nu` a positive integer greater of equal than `N`, a shape parameter of the generalised-normal prior distribution for the structural matrix B.

`hyper_nu` a positive scalar, the shape parameter of the inverted-gamma 2 prior distribution for the two overall shrinkage parameters for matrices B and A.

`hyper_a` a positive scalar, the shape parameter of the gamma prior for the two overall shrinkage parameters.

`hyper_V` a positive scalar, the shape parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`hyper_S` a positive scalar, the scale parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

Methods

Public methods:

- `specify_prior_bsvar$new()`
- `specify_prior_bsvar$get_prior()`
- `specify_prior_bsvar$clone()`

Method `new()`**:** Create a new prior specification PriorBSVAR.

Usage:

```
specify_prior_bsvar$new(N, p, stationary = rep(FALSE, N))
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

Returns: A new prior specification PriorBSVAR.

Examples:

```
# a prior for 3-variable example with one lag and stationary data
prior = specify_prior_bsvar$new(N = 3, p = 1, stationary = rep(TRUE, 3))
prior$A # show autoregressive prior mean
```

Method `get_prior()`**:** Returns the elements of the prior specification PriorBSVAR as a `list`.

Usage:

```
specify_prior_bsvar$get_prior()
```

Examples:

```
# a prior for 3-variable example with four lags
prior = specify_prior_bsvar$new(N = 3, p = 4)
prior$get_prior() # show the prior as list
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_prior_bsvar$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
prior = specify_prior_bsvar$new(N = 3, p = 1)  # a prior for 3-variable example with one lag
prior$A                                         # show autoregressive prior mean


## ------------------------------------------------
## Method `specify_prior_bsvar$new`
## ------------------------------------------------

# a prior for 3-variable example with one lag and stationary data
prior = specify_prior_bsvar$new(N = 3, p = 1, stationary = rep(TRUE, 3))
prior$A # show autoregressive prior mean


## ------------------------------------------------
## Method `specify_prior_bsvar$get_prior`
## ------------------------------------------------

# a prior for 3-variable example with four lags
prior = specify_prior_bsvar$new(N = 3, p = 4)
prior$get_prior() # show the prior as list
```

`specify_prior_bsvar_mix` *R6 Class Representing PriorBSVAR-MIX*

Description

The class PriorBSVAR-MIX presents a prior specification for the bsvar model with a zero-mean mixture of normals model for structural shocks.

Super classes

`bsvars::PriorBSVAR` -> `bsvars::PriorBSVAR-MSH` -> `PriorBSVAR-MIX`

Public fields

`A` an `NxK` matrix, the mean of the normal prior distribution for the parameter matrix A.

`A_V_inv` a `KxK` precision matrix of the normal prior distribution for each of the row of the parameter matrix A. This precision matrix is equation invariant.

`B_V_inv` an `NxN` precision matrix of the generalised-normal prior distribution for the structural matrix B. This precision matrix is equation invariant.

`B_nu` a positive integer greater of equal than `N`, a shape parameter of the generalised-normal prior distribution for the structural matrix B.

`hyper_nu` a positive scalar, the shape parameter of the inverted-gamma 2 prior distribution for the two overall shrinkage parameters for matrices B and A.

`hyper_a` a positive scalar, the shape parameter of the gamma prior for the two overall shrinkage parameters.

`hyper_V` a positive scalar, the shape parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`hyper_S` a positive scalar, the scale parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`sigma_nu` a positive scalar, the shape parameter of the inverted-gamma 2 for mixture component-dependent variances of the structural shocks, $\sigma^2_{n.s_t}$.

`sigma_s` a positive scalar, the scale parameter of the inverted-gamma 2 for mixture component-dependent variances of the structural shocks, $\sigma^2_{n.s_t}$.

`PR_TR` an `MxM` matrix, the matrix of hyper-parameters of the row-specific Dirichlet prior distribution for the state probabilities the Markov process s_t. Its rows must be identical.

Methods

Public methods:

- `specify_prior_bsvar_mix$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_prior_bsvar_mix$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
prior = specify_prior_bsvar_mix$new(N = 3, p = 1, M = 2)  # specify the prior
prior$A                                          # show autoregressive prior mean
```

`specify_prior_bsvar_msh` *R6 Class Representing PriorBSVAR-MSH*

Description

The class PriorBSVAR-MSH presents a prior specification for the bsvar model with Markov Switching Heteroskedasticity.

Super class

`bsvars::PriorBSVAR` -> `PriorBSVAR-MSH`

Public fields

`A` an `NxK` matrix, the mean of the normal prior distribution for the parameter matrix A.

`A_V_inv` a `KxK` precision matrix of the normal prior distribution for each of the row of the parameter matrix A. This precision matrix is equation invariant.

`B_V_inv` an `NxN` precision matrix of the generalised-normal prior distribution for the structural matrix B. This precision matrix is equation invariant.

`B_nu` a positive integer greater of equal than `N`, a shape parameter of the generalised-normal prior distribution for the structural matrix B.

`hyper_nu` a positive scalar, the shape parameter of the inverted-gamma 2 prior distribution for the two overall shrinkage parameters for matrices B and A.

`hyper_a` a positive scalar, the shape parameter of the gamma prior for the two overall shrinkage parameters.

`hyper_V` a positive scalar, the shape parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`hyper_S` a positive scalar, the scale parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`sigma_nu` a positive scalar, the shape parameter of the inverted-gamma 2 for MS state-dependent variances of the structural shocks, $\sigma^2_{n.s_t}$.

`sigma_s` a positive scalar, the scale parameter of the inverted-gamma 2 for MS state-dependent variances of the structural shocks, $\sigma^2_{n.s_t}$.

`PR_TR` an `MxM` matrix, the matrix of hyper-parameters of the row-specific Dirichlet prior distribution for transition probabilities matrix P of the Markov process s_t.

Methods

Public methods:

- `specify_prior_bsvar_msh$new()`
- `specify_prior_bsvar_msh$get_prior()`
- `specify_prior_bsvar_msh$clone()`

Method `new()`**:** Create a new prior specification PriorBSVAR-MSH.

Usage:

```
specify_prior_bsvar_msh$new(N, p, M, stationary = rep(FALSE, N))
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`M` an integer greater than 1 - the number of Markov process' heteroskedastic regimes.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

Returns: A new prior specification PriorBSVAR-MSH.

Method `get_prior()`**:** Returns the elements of the prior specification PriorBSVAR-MSH as a `list`.

Usage:

```
specify_prior_bsvar_msh$get_prior()
```

Examples:

```
# a prior for 3-variable example with four lags and two regimes
prior = specify_prior_bsvar_msh$new(N = 3, p = 4, M = 2)
prior$get_prior() # show the prior as list
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_prior_bsvar_msh$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
prior = specify_prior_bsvar_msh$new(N = 3, p = 1, M = 2)  # specify the prior
prior$A                                        # show autoregressive prior mean


## ------------------------------------------------
## Method `specify_prior_bsvar_msh$get_prior`
## ------------------------------------------------

# a prior for 3-variable example with four lags and two regimes
prior = specify_prior_bsvar_msh$new(N = 3, p = 4, M = 2)
prior$get_prior() # show the prior as list
```

`specify_prior_bsvar_sv` *R6 Class Representing PriorBSVAR-SV*

Description

The class PriorBSVAR-SV presents a prior specification for the bsvar model with Stochastic Volatility heteroskedasticity.

Super class

`bsvars::PriorBSVAR` -> `PriorBSVAR-SV`

Public fields

`A` an `NxK` matrix, the mean of the normal prior distribution for the parameter matrix A.

`A_V_inv` a `KxK` precision matrix of the normal prior distribution for each of the row of the parameter matrix A. This precision matrix is equation invariant.

`B_V_inv` an `NxN` precision matrix of the generalised-normal prior distribution for the structural matrix B. This precision matrix is equation invariant.

`B_nu` a positive integer greater of equal than `N`, a shape parameter of the generalised-normal prior distribution for the structural matrix B.

`hyper_nu` a positive scalar, the shape parameter of the inverted-gamma 2 prior distribution for the two overall shrinkage parameters for matrices B and A.

`hyper_a` a positive scalar, the shape parameter of the gamma prior for the two overall shrinkage parameters.

`hyper_V` a positive scalar, the shape parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`hyper_S` a positive scalar, the scale parameter of the inverted-gamma 2 for the level 3 hierarchy of shrinkage parameters.

`sv_a_` a positive scalar, the shape parameter of the gamma prior in the hierarchical prior for σ^2_ω.

`sv_s_` a positive scalar, the scale parameter of the gamma prior in the hierarchical prior for σ^2_ω.

Methods

Public methods:

- `specify_prior_bsvar_sv$new()`
- `specify_prior_bsvar_sv$get_prior()`
- `specify_prior_bsvar_sv$clone()`

Method `new()`**:** Create a new prior specification PriorBSVAR-SV.

Usage:

```
specify_prior_bsvar_sv$new(N, p, stationary = rep(FALSE, N))
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`stationary` an `N` logical vector - its element set to `FALSE` sets the prior mean for the autoregressive parameters of the `N`th equation to the white noise process, otherwise to random walk.

Returns: A new prior specification PriorBSVAR-SV.

Method `get_prior()`**:** Returns the elements of the prior specification PriorBSVAR-SV as a `list`.

Usage:

```
specify_prior_bsvar_sv$get_prior()
```

Examples:

```
# a prior for 3-variable example with four lags
prior = specify_prior_bsvar_sv$new(N = 3, p = 4)
prior$get_prior() # show the prior as list
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_prior_bsvar_sv$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
prior = specify_prior_bsvar_sv$new(N = 3, p = 1) # a prior for 3-variable example with one lag
prior$A                                          # show autoregressive prior mean


## ------------------------------------------------
## Method `specify_prior_bsvar_sv$get_prior`
## ------------------------------------------------

# a prior for 3-variable example with four lags
prior = specify_prior_bsvar_sv$new(N = 3, p = 4)
prior$get_prior() # show the prior as list
```

`specify_starting_values_bsvar`
R6 Class Representing StartingValuesBSVAR

Description

The class StartingValuesBSVAR presents starting values for the homoskedastic bsvar model.

Public fields

`A` an `NxK` matrix of starting values for the parameter A.

`B` an `NxN` matrix of starting values for the parameter B.

`hyper` a 5-vector of starting values for the shrinkage hyper-parameters of the hierarchical prior distribution.

Methods

Public methods:

- `specify_starting_values_bsvar$new()`
- `specify_starting_values_bsvar$get_starting_values()`
- `specify_starting_values_bsvar$set_starting_values()`
- `specify_starting_values_bsvar$clone()`

Method `new()`**:** Create new starting values StartingValuesBSVAR.

Usage:

```
specify_starting_values_bsvar$new(N, p)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

Returns: Starting values StartingValuesBSVAR.

Examples:

```
# starting values for a homoskedastic bsvar with 4 lags for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 4)
```

Method `get_starting_values()`**:** Returns the elements of the starting values StartingValues-BSVAR as a `list`.

Usage:

```
specify_starting_values_bsvar$get_starting_values()
```

Examples:

```
# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 1)
sv$get_starting_values()   # show starting values as list
```

Method `set_starting_values()`**:** Returns the elements of the starting values StartingValues-BSVAR as a `list`.

Usage:

```
specify_starting_values_bsvar$set_starting_values(last_draw)
```

Arguments:

`last_draw` a list containing the last draw of elements `B` - an `NxN` matrix, `A` - an `NxK` matrix, and `hyper` - a vector of 5 positive real numbers.

Returns: An object of class StartingValuesBSVAR including the last draw of the current MCMC as the starting value to be passed to the continuation of the MCMC estimation using `bsvar()`.

Examples:

```
# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 1)

# Modify the starting values by:
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_starting_values_bsvar$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# starting values for a homoskedastic bsvar for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 1)


## ------------------------------------------------
## Method `specify_starting_values_bsvar$new`
## ------------------------------------------------

# starting values for a homoskedastic bsvar with 4 lags for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 4)


## ------------------------------------------------
## Method `specify_starting_values_bsvar$get_starting_values`
## ------------------------------------------------

# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 1)
sv$get_starting_values()   # show starting values as list


## ------------------------------------------------
## Method `specify_starting_values_bsvar$set_starting_values`
## ------------------------------------------------

# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar$new(N = 3, p = 1)

# Modify the starting values by:
```

```
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

---

specify_starting_values_bsvar_mix

R6 Class Representing StartingValuesBSVAR-MIX

Description

The class StartingValuesBSVAR-MIX presents starting values for the bsvar model with a zero-mean mixture of normals model for structural shocks.

Super classes

`bsvars::StartingValuesBSVAR` -> `bsvars::StartingValuesBSVAR-MSH` -> `StartingValuesBSVAR-MIX`

Public fields

`A` an `NxK` matrix of starting values for the parameter A.

`B` an `NxN` matrix of starting values for the parameter B.

`hyper` a 5-vector of starting values for the shrinkage hyper-parameters of the hierarchical prior distribution.

`sigma2` an `NxM` matrix of starting values for the MS state-specific variances of the structural shocks. Its elements sum to value `M` over the rows.

`PR_TR` an `MxM` matrix of starting values for the probability matrix of the Markov process. Its rows must be identical and the elements of each row sum to 1 over the rows.

`xi` an `MxT` matrix of starting values for the Markov process indicator. Its columns are a chosen column of an identity matrix of order `M`.

`pi_0` an `M`-vector of starting values for mixture components state probabilities. Its elements sum to 1.

Methods

Public methods:

- `specify_starting_values_bsvar_mix$new()`
- `specify_starting_values_bsvar_mix$clone()`

Method `new()`: Create new starting values StartingValuesBSVAR-MIX.

Usage:

```
specify_starting_values_bsvar_mix$new(N, p, M, T, finiteM = TRUE)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`M` an integer greater than 1 - the number of components of the mixture of normals.

`T` a positive integer - the the time series dimension of the dependent variable matrix Y.

`finiteM` a logical value - if true a finite mixture model is estimated. Otherwise, a sparse mixture model is estimated in which `M=20` and the number of visited states is estimated.

Returns: Starting values StartingValuesBSVAR-MIX.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_starting_values_bsvar_mix$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# starting values for a bsvar model for a 3-variable system
sv = specify_starting_values_bsvar_mix$new(N = 3, p = 1, M = 2, T = 100)
```

`specify_starting_values_bsvar_msh`
R6 Class Representing StartingValuesBSVAR-MSH

Description

The class StartingValuesBSVAR-MSH presents starting values for the bsvar model with Markov Switching Heteroskedasticity.

Super class

`bsvars::StartingValuesBSVAR` -> `StartingValuesBSVAR-MSH`

Public fields

`A` an `NxK` matrix of starting values for the parameter A.

`B` an `NxN` matrix of starting values for the parameter B.

`hyper` a `5`-vector of starting values for the shrinkage hyper-parameters of the hierarchical prior distribution.

`sigma2` an `NxM` matrix of starting values for the MS state-specific variances of the structural shocks. Its elements sum to value `M` over the rows.

`PR_TR` an `MxM` matrix of starting values for the transition probability matrix of the Markov process. Its elements sum to 1 over the rows.

`xi` an `MxT` matrix of starting values for the Markov process indicator. Its columns are a chosen column of an identity matrix of order `M`.

`pi_0` an `M`-vector of starting values for state probability at time `t=0`. Its elements sum to 1.

Methods

Public methods:

- `specify_starting_values_bsvar_msh$new()`
- `specify_starting_values_bsvar_msh$get_starting_values()`
- `specify_starting_values_bsvar_msh$set_starting_values()`
- `specify_starting_values_bsvar_msh$clone()`

Method `new()`**:** Create new starting values StartingValuesBSVAR-MS.

Usage:

```
specify_starting_values_bsvar_msh$new(N, p, M, T, finiteM = TRUE)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`M` an integer greater than 1 - the number of Markov process' heteroskedastic regimes.

`T` a positive integer - the the time series dimension of the dependent variable matrix Y.

`finiteM` a logical value - if true a stationary Markov switching model is estimated. Otherwise, a sparse Markov switching model is estimated in which `M=20` and the number of visited states is estimated.

Returns: Starting values StartingValuesBSVAR-MS.

Method `get_starting_values()`**:** Returns the elements of the starting values StartingValuesBSVAR-MS as a `list`.

Usage:

```
specify_starting_values_bsvar_msh$get_starting_values()
```

Examples:

```
# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_msh$new(N = 3, p = 1, M = 2, T = 100)
sv$get_starting_values()   # show starting values as list
```

Method `set_starting_values()`**:** Returns the elements of the starting values StartingValuesBSVAR-MSH as a `list`.

Usage:

```
specify_starting_values_bsvar_msh$set_starting_values(last_draw)
```

Arguments:

`last_draw` a list containing the last draw.

Returns: An object of class StartingValuesBSVAR-MS including the last draw of the current MCMC as the starting value to be passed to the continuation of the MCMC estimation using `bsvar_msh()`.

Examples:

```
# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_msh$new(N = 3, p = 1, M = 2, T = 100)

# Modify the starting values by:
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
specify_starting_values_bsvar_msh$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# starting values for a bsvar model for a 3-variable system
sv = specify_starting_values_bsvar_msh$new(N = 3, p = 1, M = 2, T = 100)


## ------------------------------------------------
## Method `specify_starting_values_bsvar_msh$get_starting_values`
## ------------------------------------------------

# starting values for a homoskedastic bsvar with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_msh$new(N = 3, p = 1, M = 2, T = 100)
sv$get_starting_values()   # show starting values as list


## ------------------------------------------------
## Method `specify_starting_values_bsvar_msh$set_starting_values`
## ------------------------------------------------

# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_msh$new(N = 3, p = 1, M = 2, T = 100)

# Modify the starting values by:
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

---

specify_starting_values_bsvar_sv

R6 Class Representing StartingValuesBSVAR-SV

Description

The class StartingValuesBSVAR-SV presents starting values for the bsvar model with Stochastic Volatility heteroskedasticity.

Super class

`bsvars::StartingValuesBSVAR` -> `StartingValuesBSVAR-SV`

Public fields

`A` an `NxK` matrix of starting values for the parameter A.

`B` an `NxN` matrix of starting values for the parameter B.

`hyper` a `5`-vector of starting values for the shrinkage hyper-parameters of the hierarchical prior distribution.

`h` an `NxT` matrix with the starting values of the log-volatility processes.

`rho` an `N`-vector with values of SV autoregressive parameters.

`omega` an `N`-vector with values of SV process conditional standard deviations.

`S` an `NxT` integer matrix with the auxiliary mixture component indicators.

`sigma2_omega` an `N`-vector with variances of the zero-mean normal prior for ω_n.

`s_` a positive scalar with the scale of the gamma prior of the hierarchical prior for σ^2_ω.

Methods

Public methods:

- `specify_starting_values_bsvar_sv$new()`
- `specify_starting_values_bsvar_sv$get_starting_values()`
- `specify_starting_values_bsvar_sv$set_starting_values()`
- `specify_starting_values_bsvar_sv$clone()`

Method `new()`**:** Create new starting values StartingValuesBSVAR-SV.

Usage:

```
specify_starting_values_bsvar_sv$new(N, p, T)
```

Arguments:

`N` a positive integer - the number of dependent variables in the model.

`p` a positive integer - the autoregressive lag order of the SVAR model.

`T` a positive integer - the the time series dimension of the dependent variable matrix Y.

Returns: Starting values StartingValuesBSVAR-SV.

Method `get_starting_values()`**:** Returns the elements of the starting values StartingValuesBSVAR-SV as a `list`.

Usage:

```
specify_starting_values_bsvar_sv$get_starting_values()
```

Examples:

```
# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_sv$new(N = 3, p = 1, T = 100)
sv$get_starting_values()   # show starting values as list
```

Method `set_starting_values()`**:** Returns the elements of the starting values StartingValues-BSVAR_SV as a `list`.

Usage:

`specify_starting_values_bsvar_sv$set_starting_values(last_draw)`

Arguments:

`last_draw` a list containing the last draw of the current MCMC run.

Returns: An object of class StartingValuesBSVAR including the last draw of the current MCMC as the starting value to be passed to the continuation of the MCMC estimation using `bsvar()`.

Examples:

```
# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_sv$new(N = 3, p = 1, T = 100)

# Modify the starting values by:
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`specify_starting_values_bsvar_sv$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Examples

```
# starting values for a bsvar model for a 3-variable system
sv = specify_starting_values_bsvar_sv$new(N = 3, p = 1, T = 100)


## ------------------------------------------------
## Method `specify_starting_values_bsvar_sv$get_starting_values`
## ------------------------------------------------

# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_sv$new(N = 3, p = 1, T = 100)
sv$get_starting_values()   # show starting values as list


## ------------------------------------------------
## Method `specify_starting_values_bsvar_sv$set_starting_values`
## ------------------------------------------------

# starting values for a bsvar model with 1 lag for a 3-variable system
sv = specify_starting_values_bsvar_sv$new(N = 3, p = 1, T = 100)

# Modify the starting values by:
sv_list = sv$get_starting_values()   # getting them as list
sv_list$A <- matrix(rnorm(12), 3, 4) # modifying the entry
sv$set_starting_values(sv_list)      # providing to the class object
```

us_fiscal_lsuw *A 3-variable US fiscal system for the period 1950 Q1 – 2021 Q4*

Description

A system used to identify the US fiscal policy shocks.

Usage

```
data(us_fiscal_lsuw)
```

Format

A matrix and a `ts` object with time series of 288 observations on 3 variables:

ttr quarterly US total tax revenue expressed in log, real, per person terms

gs quarterly US total government spending expressed in log, real, per person terms

gdp quarterly US gross domestic product expressed in log, real, per person terms

The series are as described by Mertens & Ravn (2014) in footnote 3 and main body on page S3 of the paper. Differences with respect to Mertens & Ravn's data :

- The sample period is from quarter 1 of 1950 to quarter 4 of 2021,
- The population variable is not from Francis & Ramey (2009) but from the FRED (with the same definition),
- The original monthly population data is transformed to quarterly by taking monthly averages.

Source

U.S. Bureau of Economic Analysis, National Income and Product Accounts, https://www.bea.gov/products/national-income-and-product-accounts

FRED Economic Database, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/

References

Francis, N., and Ramey, V.A. (2009) Measures of per capita Hours and Their Implications for the Technology-hours Debate. *Journal of Money, Credit and Banking*, 41(6), 1071-1097, DOI: doi:10.1111/j.15384616.2009.00247.x.

Mertens, K., and Ravn, M.O. (2014) A Reconciliation of SVAR and Narrative Estimates of Tax Multipliers, *Journal of Monetary Economics*, 68(S), S1–S19. DOI: doi:10.1016/j.jmoneco.2013.04.004.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2022) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference.

Examples

```
data(us_fiscal_lsuw)   # upload the data
plot(us_fiscal_lsuw)   # plot the data
```

Index