Package: bsvars (via r-universe)

June 29, 2024

Type Package

Title Bayesian Estimation of Structural Vector Autoregressive Models

Description Provides fast and efficient procedures for Bayesian analysis of Structural Vector Autoregressions. This package estimates a wide range of models, including homo-, heteroskedastic, and non-normal specifications. Structural models can be identified by adjustable exclusion restrictions, time-varying volatility, or non-normality. They all include a flexible three-level equation-specific local-global hierarchical prior distribution for the estimated level of shrinkage for autoregressive and structural parameters. Additionally, the package facilitates predictive and structural analyses such as impulse responses, forecast error variance and historical decompositions, forecasting, verification of heteroskedasticity, non-normality, and hypotheses on autoregressive parameters, as well as analyses of structural shocks, volatilities, and fitted values. Beautiful plots, informative summary functions, and extensive documentation complement all this. The implemented techniques align closely with those presented in Lütkepohl, Shang, Uzeda, & Woźniak (2024) <doi:10.48550/arXiv.2404.11057>, Lütkepohl & Woźniak (2020) <doi:10.1016/j.jedc.2020.103862>, and Song & Woźniak (2021) <doi:10.1093/acrefore/9780190625979.013.174>.

Version 3.0.1

Date 2024-06-26

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

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

Suggests tinytest

LinkingTo Rcpp, RcppProgress, RcppArmadillo, RcppTN

License GPL (>= 3)

URL https://bsvars.github.io/bsvars/

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

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation yes

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

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-06-28 06:40:02 UTC

Contents

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

Description

Provides fast and efficient procedures for Bayesian analysis of Structural Vector Autoregressions. This package estimates a wide range of models, including homo-, heteroskedastic and non-normal specifications. Structural models can be identified by adjustable exclusion restrictions, time-varying volatility, or non-normality. They all include a flexible three-level equation-specific local-global hierarchical prior distribution for the estimated level of shrinkage for autoregressive and structural parameters. Additionally, the package facilitates predictive and structural analyses such as impulse responses, forecast error variance and historical decompositions, forecasting, verification of heteroskedasticity and hypotheses on autoregressive parameters, and analyses of structural shocks, volatilities, and fitted values. Beautiful plots, informative summary functions, and extensive documentation complement all this. The implemented techniques align closely with those presented in Lütkepohl, Shang, Uzeda, & Woźniak (2024) <doi:10.48550/arXiv.2404.11057>, Lütkepohl & Woźniak (2020) <doi:10.1016/j.jedc.2020.103862>, Song & Woźniak (2021) <doi:10.1093/acrefore/9780190625979.013.17
and Woźniak & Droumaguet (2015) <doi:10.13140/RG.2.2.19492.55687>.

Details

Models. 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 non-centred Stochastic Volatility process for variances
- heteroskedastic model with centred 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

Prior distributions. All the models feature a Minnesota prior for autoregressive parameters in matrix A and a generalised-normal distribution for the structural matrix B. Both of these distributions feature a 3-level equation-specific local-global hierarchical prior that make the shrinkage estimation flexible improving the model fit and its forecasting performance.

Estimation algorithm. The models are estimated using frontier numerical methods making the Gibbs sampler fast and efficient. The sampler of the structural matrix follows Waggoner & Zha (2003), whereas that for autoregressive parameters follows Chan, Koop, Yu (2022). The specification of Markov switching heteroskedasticity is inspired by Song & Woźniak (2021), and that of Stochastic Volatility model by Kastner & Frühwirth-Schnatter (2014). The estimation algorithms for particular models are scrutinised in Lütkepohl, Shang, Uzeda, & Woźniak (2024) and Woźniak & Droumaguet (2024) and some other inferential and identification problems are considered in Lütkepohl & Woźniak (2020).

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

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

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

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs.

See Also

Useful links:

- https://bsvars.github.io/bsvars/
- Report bugs at https://github.com/bsvars/bsvars/issues

Examples

```
# upload data
data(us_fiscal_lsuw)    # upload dependent variables
data(us_fiscal_ex)      # upload exogenous variables

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

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses 2 years ahead
```

```
irf           = compute_impulse_responses(posterior, horizon = 8)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 4, exogenous = us_fiscal_ex) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevds
```

compute_conditional_sd

Computes posterior draws of structural shock conditional standard deviations

Description

Each of the draws from the posterior estimation of models is transformed into a draw from the posterior distribution of the structural shock conditional standard deviations.

Usage

```
compute_conditional_sd(posterior)
```

Arguments

posterior	posterior estimation outcome obtained by running the estimate function. The interpretation depends on the normalisation of the shocks using function normalise_posterior(). Verify if the default settings are appropriate.

Value

An object of class PosteriorSigma, that is, an NxTxS array with attribute PosteriorSigma containing S draws of the structural shock conditional standard deviations.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

estimate, normalise_posterior, summary

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks' conditional standard deviations
sigma          = compute_conditional_sd(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_conditional_sd() -> csd
```

---

compute_conditional_sd.PosteriorBSVAR

Computes posterior draws of structural shock conditional standard deviations

---

Description

Each of the draws from the posterior estimation of models is transformed into a draw from the posterior distribution of the structural shock conditional standard deviations.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_conditional_sd(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.

Value

An object of class `PosteriorSigma`, that is, an `NxTxS` array with attribute `PosteriorSigma` containing `S` draws of the structural shock conditional standard deviations.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks' conditional standard deviations
sigma          = compute_conditional_sd(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_conditional_sd() -> csd
```

`compute_conditional_sd.PosteriorBSVARMIX`
Computes posterior draws of structural shock conditional standard deviations

---

Description

Each of the draws from the posterior estimation of models is transformed into a draw from the posterior distribution of the structural shock conditional standard deviations.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_conditional_sd(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.

Value

An object of class `PosteriorSigma`, that is, an `NxTxS` array with attribute `PosteriorSigma` containing `S` draws of the structural shock conditional standard deviations.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks' conditional standard deviations
csd     = compute_conditional_sd(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_conditional_sd() -> csd
```

`compute_conditional_sd.PosteriorBSVARMSH`
Computes posterior draws of structural shock conditional standard deviations

---

Description

Each of the draws from the posterior estimation of models is transformed into a draw from the posterior distribution of the structural shock conditional standard deviations.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_conditional_sd(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.

Value

An object of class `PosteriorSigma`, that is, an `NxTxS` array with attribute `PosteriorSigma` containing `S` draws of the structural shock conditional standard deviations.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks' conditional standard deviations
```

```
csd     = compute_conditional_sd(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_conditional_sd() -> csd
```

---

compute_conditional_sd.PosteriorBSVARSV

Computes posterior draws of structural shock conditional standard deviations

---

Description

Each of the draws from the posterior estimation of models is transformed into a draw from the posterior distribution of the structural shock conditional standard deviations.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_conditional_sd(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.

Value

An object of class `PosteriorSigma`, that is, an `NxTxS` array with attribute `PosteriorSigma` containing `S` draws of the structural shock conditional standard deviations.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks' conditional standard deviations
csd     = compute_conditional_sd(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_conditional_sd() -> csd
```

---

compute_fitted_values   *Computes posterior draws from data predictive density*

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the data predictive density.

Usage

```
compute_fitted_values(posterior)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function.

Value

An object of class `PosteriorFitted`, that is, an `NxTxS` array with attribute `PosteriorFitted` containing `S` draws from the data predictive density.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

estimate, summary

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute draws from in-sample predictive density
fitted         = compute_fitted_values(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_fitted_values() -> fitted
```

compute_fitted_values.PosteriorBSVAR

Computes posterior draws from data predictive density

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the data predictive density.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_fitted_values(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.

Value

An object of class `PosteriorFitted`, that is, an `NxTxS` array with attribute `PosteriorFitted` containing `S` draws from the data predictive density.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute draws from in-sample predictive density
fitted         = compute_fitted_values(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_fitted_values() -> fitted
```

compute_fitted_values.PosteriorBSVARMIX

Computes posterior draws from data predictive density

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the data predictive density.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_fitted_values(posterior)
```

Arguments

posterior	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.

Value

An object of class `PosteriorFitted`, that is, an `NxTxS` array with attribute `PosteriorFitted` containing `S` draws from the data predictive density.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute draws from in-sample predictive density
csd     = compute_fitted_values(posterior)
```

```
# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_fitted_values() -> csd
```

`compute_fitted_values.PosteriorBSVARMSH` *Computes posterior draws from data predictive density*

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the data predictive density.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_fitted_values(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.

Value

An object of class `PosteriorFitted`, that is, an `NxTxS` array with attribute `PosteriorFitted` containing `S` draws from the data predictive density.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute draws from in-sample predictive density
csd     = compute_fitted_values(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_fitted_values() -> csd
```

`compute_fitted_values.PosteriorBSVARSV`
Computes posterior draws from data predictive density

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the data predictive density.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_fitted_values(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.

Value

An object of class `PosteriorFitted`, that is, an `NxTxS` array with attribute `PosteriorFitted` containing `S` draws from the data predictive density.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

estimate, summary

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute draws from in-sample predictive density
csd     = compute_fitted_values(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_fitted_values() -> csd
```

---

compute_historical_decompositions

Computes posterior draws of historical decompositions

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the historical decompositions. IMPORTANT! The historical decompositions are interpreted correctly for covariance stationary data. Application to unit-root non-stationary data might result in non-interpretable outcomes.

Usage

```
compute_historical_decompositions(posterior, show_progress = TRUE)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function. The interpretation depends on the normalisation of the shocks using function `normalise_posterior()`. Verify if the default settings are appropriate.
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Value

An object of class `PosteriorHD`, that is, an `NxNxTxS` array with attribute `PosteriorHD` containing `S` draws of the historical decompositions.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(diff(us_fiscal_lsuw), p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute historical decompositions
hd            = compute_historical_decompositions(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
diff(us_fiscal_lsuw) |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_historical_decompositions() -> hd
```

compute_historical_decompositions.PosteriorBSVAR

Computes posterior draws of historical decompositions

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the historical decompositions. IMPORTANT! The historical decompositions are interpreted correctly for covariance stationary data. Application to unit-root non-stationary data might result in non-interpretable outcomes.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_historical_decompositions(posterior, show_progress = TRUE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Value

An object of class `PosteriorHD`, that is, an `NxNxTxS` array with attribute `PosteriorHD` containing `S` draws of the historical decompositions.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(diff(us_fiscal_lsuw), p = 1)
```

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute historical decompositions
hd            = compute_historical_decompositions(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
diff(us_fiscal_lsuw) |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_historical_decompositions() -> hd
```

---

compute_historical_decompositions.PosteriorBSVARMIX

Computes posterior draws of historical decompositions

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the historical decompositions. IMPORTANT! The historical decompositions are interpreted correctly for covariance stationary data. Application to unit-root non-stationary data might result in non-interpretable outcomes.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_historical_decompositions(posterior, show_progress = TRUE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Value

An object of class `PosteriorHD`, that is, an `NxNxTxS` array with attribute `PosteriorHD` containing `S` draws of the historical decompositions.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute historical decompositions
hd             = compute_historical_decompositions(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_historical_decompositions() -> hds
```

`compute_historical_decompositions.PosteriorBSVARMSH`
Computes posterior draws of historical decompositions

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the historical decompositions. IMPORTANT! The historical decompositions are interpreted correctly for covariance stationary data. Application to unit-root non-stationary data might result in non-interpretable outcomes.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_historical_decompositions(posterior, show_progress = TRUE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Value

An object of class `PosteriorHD`, that is, an `NxNxTxS` array with attribute `PosteriorHD` containing `S` draws of the historical decompositions.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute historical decompositions
hd             = compute_historical_decompositions(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_historical_decompositions() -> hds
```

`compute_historical_decompositions.PosteriorBSVARSV` *Computes posterior draws of historical decompositions*

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the historical decompositions. IMPORTANT! The historical decompositions are interpreted correctly for covariance stationary data. Application to unit-root non-stationary data might result in non-interpretable outcomes.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_historical_decompositions(posterior, show_progress = TRUE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.
`show_progress`	a logical value, if `TRUE` the estimation progress bar is visible

Value

An object of class `PosteriorHD`, that is, an `NxNxTxS` array with attribute `PosteriorHD` containing `S` draws of the historical decompositions.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

# run the burn-in
burn_in        = estimate(specification, 5)

# estimate the model
posterior      = estimate(burn_in, 5)

# compute historical decompositions
hd             = compute_historical_decompositions(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 5) |>
  compute_historical_decompositions() -> hds
```

---

compute_impulse_responses

Computes posterior draws of impulse responses

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the impulse responses.

Usage

```
compute_impulse_responses(posterior, horizon, standardise = FALSE)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function. The interpretation depends on the normalisation of the shocks using function `normalise_posterior()`. Verify if the default settings are appropriate.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.
`standardise`	a logical value. If `TRUE`, the impulse responses are standardised so that the variables' own shocks at horizon 0 are equal to 1. Otherwise, the parameter estimates determine this magnitude.

Value

An object of class PosteriorIR, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorIR containing `S` draws of the impulse responses.

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses 2 years ahead
irf           = compute_impulse_responses(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_impulse_responses(horizon = 8) -> ir
```

compute_impulse_responses.PosteriorBSVAR
Computes posterior draws of impulse responses

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the impulse responses.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_impulse_responses(posterior, horizon, standardise = FALSE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.
`standardise`	a logical value. If `TRUE`, the impulse responses are standardised so that the variables' own shocks at horizon 0 are equal to 1. Otherwise, the parameter estimates determine this magnitude.

Value

An object of class PosteriorIR, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorIR containing `S` draws of the impulse responses.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses 2 years ahead
irf           = compute_impulse_responses(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_impulse_responses(horizon = 8) -> ir
```

---

compute_impulse_responses.PosteriorBSVARMIX

Computes posterior draws of impulse responses

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the impulse responses.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_impulse_responses(posterior, horizon, standardise = FALSE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.
`standardise`	a logical value. If `TRUE`, the impulse responses are standardised so that the variables' own shocks at horizon 0 are equal to 1. Otherwise, the parameter estimates determine this magnitude.

Value

An object of class PosteriorIR, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorIR containing `S` draws of the impulse responses.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses
irfs            = compute_impulse_responses(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_impulse_responses(horizon = 4) -> irfs
```

`compute_impulse_responses.PosteriorBSVARMSH` *Computes posterior draws of impulse responses*

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the impulse responses.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_impulse_responses(posterior, horizon, standardise = FALSE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.
`standardise`	a logical value. If `TRUE`, the impulse responses are standardised so that the variables' own shocks at horizon 0 are equal to 1. Otherwise, the parameter estimates determine this magnitude.

Value

An object of class PosteriorIR, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorIR containing `S` draws of the impulse responses.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses
irfs            = compute_impulse_responses(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_impulse_responses(horizon = 4) -> irfs
```

---

compute_impulse_responses.PosteriorBSVARSV

Computes posterior draws of impulse responses

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the impulse responses.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_impulse_responses(posterior, horizon, standardise = FALSE)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.
`standardise`	a logical value. If `TRUE`, the impulse responses are standardised so that the variables' own shocks at horizon 0 are equal to 1. Otherwise, the parameter estimates determine this magnitude.

Value

An object of class PosteriorIR, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorIR containing `S` draws of the impulse responses.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute impulse responses
irfs            = compute_impulse_responses(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_impulse_responses(horizon = 4) -> irfs
```

compute_regime_probabilities

Computes posterior draws of regime probabilities

Description

Each of the draws from the posterior estimation of a model is transformed into a draw from the posterior distribution of the regime probabilities. These represent either the realisations of the regime indicators, when `type = "realized"`, filtered probabilities, when `type = "filtered"`, forecasted regime probabilities, when `type = "forecasted"`, or the smoothed probabilities, when `type = "smoothed"`, .

Usage

```
compute_regime_probabilities(
  posterior,
  type = c("realized", "filtered", "forecasted", "smoothed")
)
```

Arguments

`posterior`	posterior estimation outcome of regime-dependent heteroskedastic models - an object of either of the classes: PosteriorBSVARMSH, or PosteriorBSVARMIX obtained by running the `estimate` function.
`type`	one of the values `"realized"`, `"filtered"`, `"forecasted"`, or `"smoothed"` denoting the type of probabilities to be computed.

Value

An object of class PosteriorRegimePr, that is, an `MxTxS` array with attribute PosteriorRegimePr containing `S` draws of the regime probabilities.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute the posterior draws of realized regime indicators
regimes        = compute_regime_probabilities(posterior)

# compute the posterior draws of filtered probabilities
filtered       = compute_regime_probabilities(posterior, "filtered")

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) -> posterior
regimes        = compute_regime_probabilities(posterior)
filtered       = compute_regime_probabilities(posterior, "filtered")
```

---

compute_regime_probabilities.PosteriorBSVARMIX

Computes posterior draws of regime probabilities

Description

Each of the draws from the posterior estimation of a model is transformed into a draw from the posterior distribution of the regime probabilities. These represent either the realisations of the regime indicators, when `type = "realized"`, filtered probabilities, when `type = "filtered"`, forecasted regime probabilities, when `type = "forecasted"`, or the smoothed probabilities, when `type = "smoothed"`,.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_regime_probabilities(
  posterior,
  type = c("realized", "filtered", "forecasted", "smoothed")
)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.
`type`	one of the values `"realized"`, `"filtered"`, `"forecasted"`, or `"smoothed"` denoting the type of probabilities to be computed.

Value

An object of class PosteriorRegimePr, that is, an `MxTxS` array with attribute PosteriorRegimePr containing `S` draws of the regime probabilities.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute the posterior draws of realized regime indicators
regimes        = compute_regime_probabilities(posterior)

# compute the posterior draws of filtered probabilities
filtered       = compute_regime_probabilities(posterior, "filtered")

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) -> posterior
regimes        = compute_regime_probabilities(posterior)
filtered       = compute_regime_probabilities(posterior, "filtered")
```

`compute_regime_probabilities.PosteriorBSVARMSH`
Computes posterior draws of regime probabilities

Description

Each of the draws from the posterior estimation of a model is transformed into a draw from the posterior distribution of the regime probabilities. These represent either the realisations of the regime indicators, when `type = "realized"`, filtered probabilities, when `type = "filtered"`, forecasted regime probabilities, when `type = "forecasted"`, or the smoothed probabilities, when `type = "smoothed"`, .

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_regime_probabilities(
  posterior,
  type = c("realized", "filtered", "forecasted", "smoothed")
)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.
`type`	one of the values `"realized"`, `"filtered"`, `"forecasted"`, or `"smoothed"` denoting the type of probabilities to be computed.

Value

An object of class PosteriorRegimePr, that is, an `MxTxS` array with attribute PosteriorRegimePr containing `S` draws of the regime probabilities.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

See Also

`estimate`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute the posterior draws of realized regime indicators
regimes        = compute_regime_probabilities(posterior)

# compute the posterior draws of filtered probabilities
filtered       = compute_regime_probabilities(posterior, "filtered")

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) -> posterior
regimes        = compute_regime_probabilities(posterior)
filtered       = compute_regime_probabilities(posterior, "filtered")
```

`compute_structural_shocks` *Computes posterior draws of structural shocks*

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the structural shocks.

Usage

```
compute_structural_shocks(posterior)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function. The interpretation depends on the normalisation of the shocks using function `normalise_posterior()`. Verify if the default settings are appropriate.

Value

An object of class PosteriorShocks, that is, an `NxTxS` array with attribute PosteriorShocks containing `S` draws of the structural shocks.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks
shocks         = compute_structural_shocks(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_structural_shocks() -> ss
```

`compute_structural_shocks.PosteriorBSVAR`

Computes posterior draws of structural shocks

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the structural shocks.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_structural_shocks(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.

Value

An object of class PosteriorShocks, that is, an `NxTxS` array with attribute PosteriorShocks containing `S` draws of the structural shocks.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks
shocks         = compute_structural_shocks(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_structural_shocks() -> ss
```

compute_structural_shocks.PosteriorBSVARMIX

Computes posterior draws of structural shocks

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the structural shocks.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_structural_shocks(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.

Value

An object of class PosteriorShocks, that is, an `NxTxS` array with attribute PosteriorShocks containing `S` draws of the structural shocks.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks
shocks         = compute_structural_shocks(posterior)
```

```
# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_structural_shocks() -> ss
```

compute_structural_shocks.PosteriorBSVARMSH

Computes posterior draws of structural shocks

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the structural shocks.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_structural_shocks(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.

Value

An object of class PosteriorShocks, that is, an `NxTxS` array with attribute PosteriorShocks containing `S` draws of the structural shocks.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks
shocks         = compute_structural_shocks(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_structural_shocks() -> ss
```

---

compute_structural_shocks.PosteriorBSVARSV

Computes posterior draws of structural shocks

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the structural shocks.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_structural_shocks(posterior)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.

Value

An object of class PosteriorShocks, that is, an `NxTxS` array with attribute PosteriorShocks containing `S` draws of the structural shocks.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

estimate, normalise_posterior, summary

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute structural shocks
shocks         = compute_structural_shocks(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_structural_shocks() -> ss
```

---

compute_variance_decompositions

Computes posterior draws of the forecast error variance decomposition

---

Description

Each of the draws from the posterior estimation of models from packages **bsvars** or **bsvarSIGNs** is transformed into a draw from the posterior distribution of the forecast error variance decomposition.

Usage

```
compute_variance_decompositions(posterior, horizon)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function. The interpretation depends on the normalisation of the shocks using function `normalise_posterior()`. Verify if the default settings are appropriate.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.

Value

An object of class PosteriorFEVD, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorFEVD containing `S` draws of the forecast error variance decomposition.

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`compute_impulse_responses`, `estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevd
```

compute_variance_decompositions.PosteriorBSVAR

Computes posterior draws of the forecast error variance decomposition

Description

Each of the draws from the posterior estimation of the model is transformed into a draw from the posterior distribution of the forecast error variance decomposition.

Usage

```
## S3 method for class 'PosteriorBSVAR'
compute_variance_decompositions(posterior, horizon)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.

Value

An object of class PosteriorFEVD, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorFEVD containing `S` draws of the forecast error variance decomposition.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`compute_impulse_responses`, `estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevd
```

---

compute_variance_decompositions.PosteriorBSVARMIX

Computes posterior draws of the forecast error variance decomposition

---

Description

Each of the draws from the posterior estimation of the model is transformed into a draw from the posterior distribution of the forecast error variance decomposition. In this mixture model the forecast error variance decompositions are computed for the forecasts with the origin at the last observation in sample data and using the conditional variance forecasts.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
compute_variance_decompositions(posterior, horizon)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.

Value

An object of class PosteriorFEVD, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorFEVD containing `S` draws of the forecast error variance decomposition.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`compute_impulse_responses`, `estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevd
```

`compute_variance_decompositions.PosteriorBSVARMSH` *Computes posterior draws of the forecast error variance decomposition*

Description

Each of the draws from the posterior estimation of the model is transformed into a draw from the posterior distribution of the forecast error variance decomposition. In this heteroskedastic model the forecast error variance decompositions are computed for the forecasts with the origin at the last observation in sample data and using the conditional variance forecasts.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
compute_variance_decompositions(posterior, horizon)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.

Value

An object of class PosteriorFEVD, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorFEVD containing `S` draws of the forecast error variance decomposition.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`compute_impulse_responses`, `estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevd
```

---

compute_variance_decompositions.PosteriorBSVARSV

Computes posterior draws of the forecast error variance decomposition

---

Description

Each of the draws from the posterior estimation of the model is transformed into a draw from the posterior distribution of the forecast error variance decomposition. In this heteroskedastic model the forecast error variance decompositions are computed for the forecasts with the origin at the last observation in sample data and using the conditional variance forecasts.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
compute_variance_decompositions(posterior, horizon)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.
`horizon`	a positive integer number denoting the forecast horizon for the impulse responses computations.

Value

An object of class PosteriorFEVD, that is, an `NxNx(horizon+1)xS` array with attribute PosteriorFEVD containing `S` draws of the forecast error variance decomposition.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Kilian, L., & Lütkepohl, H. (2017). Structural VAR Tools, Chapter 4, In: Structural vector autoregressive analysis. Cambridge University Press.

See Also

`compute_impulse_responses`, `estimate`, `normalise_posterior`, `summary`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# compute forecast error variance decomposition 2 years ahead
fevd           = compute_variance_decompositions(posterior, horizon = 8)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  compute_variance_decompositions(horizon = 8) -> fevd
```

`estimate` *Bayesian estimation of Structural Vector Autoregressions via Gibbs sampler*

Description

Estimates homo- or heteroskedastic SVAR models for packages **bsvars** and **bsvarSIGNs**. The packages apply the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2024) 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 3-level equation-specific local-global hierarchical prior for the shrinkage parameters. A variety of models for conditional variances are possible including versions of Stochastic Volatility and Markov-switching heteroskedasticity. Non-normal specifications include finite and sparse normal mixture model for the structural shocks. The estimation algorithms for particular models are scrutinised in Lütkepohl, Shang, Uzeda, & Woźniak (2024) and Woźniak & Droumaguet (2024) and some other inferential and identification problems are considered in Lütkepohl & Woźniak (2020) and Song & Woźniak (2021). Models from package **bsvars** implement identification via exclusion restrictions, heteroskedasticity and non-normality. Models from package **bsvarSIGNs** implement identification via sign and narrative restrictions. See section **Details** and package **bsvarSIGNs** documentation for more information.

Usage

```
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object generated using one of the `specify_bsvar*` functions or an object generated using the function `estimate`. The latter type of input facilitates the continuation of the MCMC sampling starting from the last draw of the previous run.
`S`	a positive integer, the number of posterior draws to be generated
`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.

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

The various SVAR models estimated differ by the specification of structural shocks variances. Their specification depends on the `specify_bsvar*` function used. 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

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 many arrays and vectors whose selection depends on the model specification. `last_draw` an object generated by one of the `specify_bsvar*` functions with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

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

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs.

See Also

`specify_bsvar`, `specify_bsvar_msh`, `specify_bsvar_mix`, `specify_bsvar_sv`, `normalise_posterior`

Examples

```
# simple 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)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) -> posterior
```

`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 (2024) 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 hierarchical prior distribution as in Lütkepohl, Shang, Uzeda, and Woźniak (2024). See section **Details** for the model equations.

Usage

```
## S3 method for class 'BSVAR'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class BSVAR generated using the `specify_bsvar$new()` function.
`S`	a positive integer, the number of posterior draws to be generated
`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 `estimate()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

See Also

`specify_bsvar`, `specify_posterior_bsvar`, `normalise_posterior`

Examples

```
# simple 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)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

`estimate.BSVARMIX` *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 (2024) 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 hierarchical prior distribution. The finite mixture of normals model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2024), Lütkepohl & Woźniak (2020), and Song & Woźniak (2021). See section **Details** for the model equations.

Usage

```
## S3 method for class 'BSVARMIX'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class BSVARMIX generated using the `specify_bsvar_mix$new()` function.
`S`	a positive integer, the number of posterior draws to be generated
`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 PosteriorBSVARMIX 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

sigma an `NxTxS` array with the posterior draws for the structural shocks conditional standard deviations' series over the sample period

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

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

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.

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

See Also

`specify_bsvar_mix`, `specify_posterior_bsvar_mix`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

`estimate.BSVARMSH` *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 (2024) 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 hierarchical prior distribution. The MS model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2024), Lütkepohl & Woźniak (2020), and Song & Woźniak (2021). See section **Details** for the model equations.

Usage

```
## S3 method for class 'BSVARMSH'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class BSVARMSH generated using the `specify_bsvar_msh$new()` function.
`S`	a positive integer, the number of posterior draws to be generated
`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 PosteriorBSVARMSH 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

sigma an `NxTxS` array with the posterior draws for the structural shocks conditional standard deviations' series over the sample period

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

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

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.

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

See Also

`specify_bsvar_msh`, `specify_posterior_bsvar_msh`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

# run the burn-in
burn_in        = estimate(specification, 5)
```

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

`estimate.BSVARSV` *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 (2024). Implements the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2024) 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 hierarchical prior distribution. The SV model is estimated 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

```
## S3 method for class 'BSVARSV'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class BSVARSV generated using the `specify_bsvar_sv$new()` function.
`S`	a positive integer, the number of posterior draws to be generated
`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.

Two alternative specifications of the conditional variance of the `n`th shock at time `t` can be estimated: non-centred Stochastic Volatility by Lütkepohl, Shang, Uzeda, and Woźniak (2022) or centred Stochastic Volatility by Chan, Koop, & Yu (2021).

The non-centred Stochastic Volatility by Lütkepohl, Shang, Uzeda, and Woźniak (2022) is selected by setting argument `centred_sv` of function `specify_bsvar_sv$new()` to value `FALSE`. It has the conditional variances 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.

The centred Stochastic Volatility by Chan, Koop, & Yu (2021) is selected by setting argument `centred_sv` of function `specify_bsvar_sv$new()` to value `TRUE`. Its conditional variances are given by:

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

where the log-conditional variances $h_{n.t}$ follow 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 zero-mean normal error term with variance $s^2_{v.n}$.

Value

An object of class PosteriorBSVARSV 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 BSVARSV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

See Also

specify_bsvar_sv, specify_posterior_bsvar_sv, normalise_posterior

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# estimate the model
posterior      = estimate(burn_in, 20, 2)

# workflow with the pipe |>
```

```
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

---

estimate.PosteriorBSVAR

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 (2024) 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 hierarchical prior distribution as in Lütkepohl, Shang, Uzeda, and Woźniak (2024). See section **Details** for the model equations.

Usage

```
## S3 method for class 'PosteriorBSVAR'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class PosteriorBSVAR generated using the `estimate.BSVAR()` function. This setup facilitates the continuation of the MCMC sampling starting from the last draw of the previous run.
`S`	a positive integer, the number of posterior draws to be generated
`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 `estimate()`.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

See Also

`specify_bsvar`, `specify_posterior_bsvar`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
```

```
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)
set.seed(123)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

`estimate.PosteriorBSVARMIX`

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 (2024) 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 hierarchical prior distribution. The finite mixture of normals model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2024), Lütkepohl & Woźniak (2020), and Song & Woźniak (2021). See section **Details** for the model equations.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class PosteriorBSVARMIX generated using the `estimate.BSVAR()` function. This setup facilitates the continuation of the MCMC sampling starting from the last draw of the previous run.
`S`	a positive integer, the number of posterior draws to be generated
`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 PosteriorBSVARMIX 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

sigma an `NxTxS` array with the posterior draws for the structural shocks conditional standard deviations' series over the sample period

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

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

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.

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

See Also

`specify_bsvar_mix`, `specify_posterior_bsvar_mix`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 2)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

estimate.PosteriorBSVARMSH

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 (2024) 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 hierarchical prior distribution. The MS model is estimated using the prior distributions and algorithms proposed by Woźniak & Droumaguet (2024), Lütkepohl & Woźniak (2020), and Song & Woźniak (2021). See section **Details** for the model equations.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class PosteriorBSVARMSH generated using the `estimate.BSVAR()` function. This setup facilitates the continuation of the MCMC sampling starting from the last draw of the previous run.
`S`	a positive integer, the number of posterior draws to be generated
`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 PosteriorBSVARMSH 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

sigma an `NxTxS` array with the posterior draws for the structural shocks conditional standard deviations' series over the sample period

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

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

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.

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.

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

See Also

`specify_bsvar_msh`, `specify_posterior_bsvar_msh`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

estimate.PosteriorBSVARSV

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 (2024). Implements the Gibbs sampler proposed by Waggoner & Zha (2003) for the structural matrix B and the equation-by-equation sampler by Chan, Koop, & Yu (2024) 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 hierarchical prior distribution. The SV model is estimated 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

```
## S3 method for class 'PosteriorBSVARSV'
estimate(specification, S, thin = 1, show_progress = TRUE)
```

Arguments

`specification`	an object of class PosteriorBSVARSV generated using the `estimate.BSVAR()` function. This setup facilitates the continuation of the MCMC sampling starting from the last draw of the previous run.
`S`	a positive integer, the number of posterior draws to be generated
`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.

Two alternative specifications of the conditional variance of the `nth` shock at time `t` can be estimated: non-centred Stochastic Volatility by Lütkepohl, Shang, Uzeda, and Woźniak (2022) or centred Stochastic Volatility by Chan, Koop, & Yu (2021).

The non-centred Stochastic Volatility by Lütkepohl, Shang, Uzeda, and Woźniak (2022) is selected by setting argument `centred_sv` of function `specify_bsvar_sv$new()` to value `FALSE`. It has the conditional variances 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.

The centred Stochastic Volatility by Chan, Koop, & Yu (2021) is selected by setting argument `centred_sv` of function `specify_bsvar_sv$new()` to value `TRUE`. Its conditional variances are given by:

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

where the log-conditional variances $h_{n.t}$ follow 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 zero-mean normal error term with variance $s^2_{v.n}$.

Value

An object of class PosteriorBSVARSV 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 BSVARSV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

References

Chan, J.C.C., Koop, G, and Yu, X. (2024) Large Order-Invariant Bayesian VARs with Stochastic Volatility. *Journal of Business & Economic Statistics*, **42**, doi:10.1080/07350015.2023.2252039.

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

See Also

`specify_bsvar_sv`, `specify_posterior_bsvar_sv`, `normalise_posterior`

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# estimate the model
posterior      = estimate(burn_in, 20, 2)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 2) |>
  compute_impulse_responses(horizon = 4) -> irf
```

`forecast` *Forecasting using Structural Vector Autoregression*

Description

Samples from the joint predictive density of all of the dependent variables for models from packages **bsvars**, **bsvarSIGNs** or **bvarPANELs** at forecast horizons from 1 to `horizon` specified as an argument of the function.

Usage

```
forecast(posterior, horizon = 1, exogenous_forecast, conditional_forecast)
```

Arguments

`posterior`	posterior estimation outcome obtained by running the `estimate` function.
`horizon`	a positive integer, specifying the forecasting horizon.
`exogenous_forecast`	forecasted values of the exogenous variables.
`conditional_forecast`	forecasted values for selected variables.

Value

A list of class `Forecasts` containing the draws from the predictive density and for heteroskedastic models the draws from the predictive density of structural shocks conditional standard deviations and data. The output elements include:

forecasts an `NxTxS` array with the draws from predictive density

forecasts_sigma provided only for heteroskedastic models, an `NxTxS` array with the draws from the predictive density of structural shocks conditional standard deviations

Y an NxT matrix with the data on dependent variables

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

# run the burn-in
```

```
burn_in        = estimate(specification, 10)

# estimate the model
posterior      = estimate(burn_in, 20)

# sample from predictive density 1 year ahead
predictive     = forecast(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 4) -> predictive
```

forecast.PosteriorBSVAR

Forecasting using Structural Vector Autoregression

Description

Samples from the joint predictive density of all of the dependent variables for models from packages **bsvars**, **bsvarSIGNs** or **bvarPANELs** at forecast horizons from 1 to `horizon` specified as an argument of the function.

Usage

```
## S3 method for class 'PosteriorBSVAR'
forecast(
  posterior,
  horizon = 1,
  exogenous_forecast = NULL,
  conditional_forecast = NULL
)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVAR` obtained by running the `estimate` function.
`horizon`	a positive integer, specifying the forecasting horizon.
`exogenous_forecast`	a matrix of dimension `horizon` x `d` containing forecasted values of the exogenous variables.

`conditional_forecast`
: a `horizon` x `N` matrix with forecasted values for selected variables. It should only contain `numeric` or `NA` values. The entries with `NA` values correspond to the values that are forecasted conditionally on the realisations provided as `numeric` values.

Value

A list of class `Forecasts` containing the draws from the predictive density and data. The output list includes element:

forecasts an `NxTxS` array with the draws from predictive density

Y an NxT matrix with the data on dependent variables

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# sample from predictive density 1 year ahead
predictive     = forecast(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 4) -> predictive

# conditional forecasting 2 quarters ahead conditioning on
#  provided future values for the Gross Domestic Product
############################################################
cf        = matrix(NA , 2, 3)
cf[,3]   = tail(us_fiscal_lsuw, 1)[3]  # conditional forecasts equal to the last gdp observation
predictive    = forecast(posterior, 2, conditional_forecast = cf)
```

```
# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 2, conditional_forecast = cf) -> predictive
```

---

forecast.PosteriorBSVARMIX

Forecasting using Structural Vector Autoregression

Description

Samples from the joint predictive density of all of the dependent variables for models from packages **bsvars**, **bsvarSIGNs** or **bvarPANELs** at forecast horizons from 1 to `horizon` specified as an argument of the function.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
forecast(
  posterior,
  horizon = 1,
  exogenous_forecast = NULL,
  conditional_forecast = NULL
)
```

Arguments

`posterior` posterior estimation outcome - an object of class `PosteriorBSVARMIX` obtained by running the `estimate` function.

`horizon` a positive integer, specifying the forecasting horizon.

`exogenous_forecast`
a matrix of dimension `horizon` x `d` containing forecasted values of the exogenous variables.

`conditional_forecast`
a `horizon` x `N` matrix with forecasted values for selected variables. It should only contain `numeric` or `NA` values. The entries with `NA` values correspond to the values that are forecasted conditionally on the realisations provided as `numeric` values.

Value

A list of class `Forecasts` containing the draws from the predictive density and for heteroskedastic models the draws from the predictive density of structural shocks conditional standard deviations and data. The output elements include:

forecasts an `NxTxS` array with the draws from predictive density

forecasts_sigma provided only for heteroskedastic models, an `NxTxS` array with the draws from the predictive density of structural shocks conditional standard deviations

Y an NxT matrix with the data on dependent variables

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# sample from predictive density 1 year ahead
predictive     = forecast(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 4) -> predictive

# conditional forecasting 2 quarters ahead conditioning on
#  provided future values for the Gross Domestic Product
############################################################
cf        = matrix(NA , 2, 3)
cf[,3]   = tail(us_fiscal_lsuw, 1)[3]  # conditional forecasts equal to the last gdp observation
predictive    = forecast(posterior, 2, conditional_forecast = cf)

# workflow with the pipe |>
############################################################
set.seed(123)
```

```
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 2, conditional_forecast = cf) -> predictive
```

---

forecast.PosteriorBSVARMSH *Forecasting using Structural Vector Autoregression*

Description

Samples from the joint predictive density of all of the dependent variables for models from packages **bsvars**, **bsvarSIGNs** or **bvarPANELs** at forecast horizons from 1 to `horizon` specified as an argument of the function.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
forecast(
  posterior,
  horizon = 1,
  exogenous_forecast = NULL,
  conditional_forecast = NULL
)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARMSH` obtained by running the `estimate` function.
`horizon`	a positive integer, specifying the forecasting horizon.
`exogenous_forecast`	a matrix of dimension `horizon` x `d` containing forecasted values of the exogenous variables.
`conditional_forecast`	a `horizon` x `N` matrix with forecasted values for selected variables. It should only contain `numeric` or `NA` values. The entries with `NA` values correspond to the values that are forecasted conditionally on the realisations provided as `numeric` values.

Value

A list of class `Forecasts` containing the draws from the predictive density and for heteroskedastic models the draws from the predictive density of structural shocks conditional standard deviations and data. The output elements include:

forecasts an `NxTxS` array with the draws from predictive density

forecasts_sigma provided only for heteroskedastic models, an NxTxS array with the draws from the predictive density of structural shocks conditional standard deviations

Y an NxT matrix with the data on dependent variables

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)

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

# estimate the model
posterior      = estimate(burn_in, 20)

# sample from predictive density 1 year ahead
predictive     = forecast(posterior, 4)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 4) -> predictive

# conditional forecasting 2 quarters ahead conditioning on
#  provided future values for the Gross Domestic Product
############################################################
cf        = matrix(NA , 2, 3)
cf[,3]   = tail(us_fiscal_lsuw, 1)[3]  # conditional forecasts equal to the last gdp observation
predictive    = forecast(posterior, 2, conditional_forecast = cf)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10) |>
  estimate(S = 20) |>
  forecast(horizon = 2, conditional_forecast = cf) -> predictive
```

`forecast.PosteriorBSVARSV`
Forecasting using Structural Vector Autoregression

Description

Samples from the joint predictive density of all of the dependent variables for models from packages **bsvars**, **bsvarSIGNs** or **bvarPANELs** at forecast horizons from 1 to `horizon` specified as an argument of the function.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
forecast(
  posterior,
  horizon = 1,
  exogenous_forecast = NULL,
  conditional_forecast = NULL
)
```

Arguments

`posterior`	posterior estimation outcome - an object of class `PosteriorBSVARSV` obtained by running the `estimate` function.
`horizon`	a positive integer, specifying the forecasting horizon.
`exogenous_forecast`	a matrix of dimension `horizon` x `d` containing forecasted values of the exogenous variables.
`conditional_forecast`	a `horizon` x `N` matrix with forecasted values for selected variables. It should only contain `numeric` or `NA` values. The entries with `NA` values correspond to the values that are forecasted conditionally on the realisations provided as `numeric` values.

Value

A list of class `Forecasts` containing the draws from the predictive density and for heteroskedastic models the draws from the predictive density of structural shocks conditional standard deviations and data. The output elements include:

forecasts an `NxTxS` array with the draws from predictive density

forecasts_sigma provided only for heteroskedastic models, an `NxTxS` array with the draws from the predictive density of structural shocks conditional standard deviations

Y an NxT matrix with the data on dependent variables

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)

# run the burn-in
burn_in        = estimate(specification, 5)

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

# sample from predictive density 1 year ahead
predictive     = forecast(posterior, 2)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 10, thin = 2) |>
  forecast(horizon = 2) -> predictive

# conditional forecasting 2 quarters ahead conditioning on
#  provided future values for the Gross Domestic Product
############################################################
cf        = matrix(NA , 2, 3)
cf[,3]   = tail(us_fiscal_lsuw, 1)[3]  # conditional forecasts equal to the last gdp observation
predictive    = forecast(posterior, 2, conditional_forecast = cf)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 10) |>
  forecast(horizon = 2, conditional_forecast = cf) -> predictive
```

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, PosteriorBSVARMSH, PosteriorBSVARMIX, or PosteriorBSVARSV 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`

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(specification, 10)
```

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

# normalise the posterior
BB            = posterior$last_draw$starting_values$B      # get the last draw of B
B_hat        = diag((-1) * sign(diag(BB))) %*% BB        # set negative diagonal elements
normalise_posterior(posterior, B_hat)                 # draws in posterior are normalised
```

`plot.Forecasts` *Plots fitted values of dependent variables*

Description

Plots of fitted values of dependent variables including their median and percentiles.

Usage

```
## S3 method for class 'Forecasts'
plot(
  x,
  probability = 0.9,
  data_in_plot = 1,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 2.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class Forecasts obtained using the `forecast()` function containing posterior draws of fitted values of dependent variables.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`data_in_plot`	a fraction value in the range (0, 1) determining how many of the last observations in the data should be plotted with the forecasts.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

forecast

Examples

```
data(us_fiscal_lsuw)                               # upload data
set.seed(123)                                      # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)       # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)   # estimate the model

# compute forecasts
fore            = forecast(posterior, horizon = 4)
plot(fore)                                         # plot forecasts

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  forecast(horizon = 4) |>
  plot()
```

---

plot.PosteriorFEVD *Plots forecast error variance decompositions*

Description

Plots of the posterior means of the forecast error variance decompositions.

Usage

```
## S3 method for class 'PosteriorFEVD'
plot(
  x,
  cols,
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 4.6),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorFEVD obtained using the `compute_variance_decompositions()` function containing posterior draws of forecast error variance decompositions.
`cols`	an `N`-vector with colours of the plot
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_variance_decompositions`

Examples

```
data(us_fiscal_lsuw)                                  # upload data
set.seed(123)                                         # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)          # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)      # estimate the model

# compute forecast error variance decompositions
fevd           = compute_variance_decompositions(posterior, horizon = 4)
plot(fevd)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_variance_decompositions(horizon = 4) |>
  plot()
```

`plot.PosteriorFitted` *Plots fitted values of dependent variables*

Description

Plots of fitted values of dependent variables including their median and percentiles.

Usage

```
## S3 method for class 'PosteriorFitted'
plot(
  x,
  probability = 0.9,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 2.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorFitted obtained using the `compute_fitted_values()` function containing posterior draws of fitted values of dependent variables.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_fitted_values`

Examples

```
data(us_fiscal_lsuw)                                  # upload data
set.seed(123)                                         # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)          # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)      # estimate the model

# compute fitted values
fitted         = compute_fitted_values(posterior)
plot(fitted)                                          # plot fitted values

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_fitted_values() |>
  plot()
```

plot.PosteriorHD *Plots historical decompositions*

Description

Plots of the posterior means of the historical decompositions.

Usage

```
## S3 method for class 'PosteriorHD'
plot(
  x,
  cols,
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 4.6),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorHD obtained using the `compute_historical_decompositions()` function containing posterior draws of historical decompositions.
`cols`	an `N`-vector with colours of the plot
`main`	an alternative main title for the plot

`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default oma argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_historical_decompositions`

Examples

```
data(us_fiscal_lsuw)                                  # upload data
set.seed(123)                                         # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)          # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)      # estimate the model

# compute historical decompositions
fevd           = compute_historical_decompositions(posterior)
plot(fevd)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_historical_decompositions() |>
  plot()
```

`plot.PosteriorIR` *Plots impulse responses*

Description

Plots of of all variables to all shocks including their median and percentiles.

Usage

```
## S3 method for class 'PosteriorIR'
plot(
  x,
  probability = 0.9,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.1, 0, 1.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorIR obtained using the `compute_impulse_responses()` function containing posterior draws of impulse responses.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_impulse_responses`

Examples

```
data(us_fiscal_lsuw)                                  # upload data
set.seed(123)                                         # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)          # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)      # estimate the model

# compute impulse responses
fitted         = compute_impulse_responses(posterior, horizon = 4)
plot(fitted)                                          # plot

# workflow with the pipe |>
```

```
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_impulse_responses(horizon = 4) |>
  plot()
```

---

plot.PosteriorRegimePr

Plots estimated regime probabilities

Description

Plots of estimated regime probabilities of Markov-switching heteroskedasticity or allocations of normal-mixture components including their median and percentiles.

Usage

```
## S3 method for class 'PosteriorRegimePr'
plot(
  x,
  probability = 0.9,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 2.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorRegimePr obtained using the `compute_regime_probabilities()` function containing posterior draws of regime probabilities.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

compute_regime_probabilities

Examples

```
data(us_fiscal_lsuw)                                # upload data
set.seed(123)                                       # set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw)# specify model
burn_in        = estimate(specification, 10)        # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)    # estimate the model

# compute regime probabilities
rp             = compute_regime_probabilities(posterior)
plot(rp)                                            # plot

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_regime_probabilities() |>
  plot()
```

---

plot.PosteriorShocks *Plots structural shocks*

Description

Plots of structural shocks including their median and percentiles.

Usage

```
## S3 method for class 'PosteriorShocks'
plot(
  x,
  probability = 0.9,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 2.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorShocks obtained using the `compute_structural_shocks()` function containing posterior draws of structural shocks.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

See Also

`compute_structural_shocks`

Examples

```
data(us_fiscal_lsuw)                                # upload data
set.seed(123)                                       # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)    # specify model
burn_in        = estimate(specification, 10)          # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)      # estimate the model

# compute structural shocks
shocks         = compute_structural_shocks(posterior)
plot(shocks)                                        # plot

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_structural_shocks() |>
  plot()
```

plot.PosteriorSigma *Plots structural shocks' conditional standard deviations*

Description

Plots of structural shocks' conditional standard deviations including their median and percentiles.

Usage

```
## S3 method for class 'PosteriorSigma'
plot(
  x,
  probability = 0.9,
  col = "#ff69b4",
  main,
  xlab,
  mar.multi = c(1, 4.6, 0, 2.1),
  oma.multi = c(6, 0, 5, 0),
  ...
)
```

Arguments

`x`	an object of class PosteriorSigma obtained using the `compute_conditional_sd()` function containing posterior draws of conditional standard deviations of structural shocks.
`probability`	a parameter determining the interval to be plotted. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot line and the ribbon
`main`	an alternative main title for the plot
`xlab`	an alternative x-axis label for the plot
`mar.multi`	the default `mar` argument setting in `graphics::par`. Modify with care!
`oma.multi`	the default `oma` argument setting in `graphics::par`. Modify with care!
`...`	additional arguments affecting the summary produced.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_conditional_sd`

Examples

```
data(us_fiscal_lsuw)                                  # upload data
set.seed(123)                                         # set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw) # specify model
burn_in        = estimate(specification, 5)           # run the burn-in
posterior      = estimate(burn_in, 5)                 # estimate the model

# compute structural shocks' conditional standard deviations
sigma          = compute_conditional_sd(posterior)
plot(sigma)                                            # plot conditional sds

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 5) |>
  estimate(S = 5) |>
  compute_conditional_sd() |>
  plot()
```

`plot_ribbon` *Plots the median and an interval between two specified percentiles for a sequence of* `K` *random variables*

Description

Plots the median and an interval between two specified percentiles for a sequence of `K` random variables based on the `S` posterior draws provided for each of them.

Usage

```
plot_ribbon(
  draws,
  probability = 0.9,
  col = "#ff69b4",
  ylim,
  ylab,
  xlab,
  start_at = 0,
  add = FALSE,
  ...
)
```

Arguments

`draws`	a `K x S` matrix with `S` posterior draws of `K` random variables, or a `K x S x N` array with `N` such matrices
`probability`	a number from interval `(0,1)` denoting the probability content of the plotted interval. The interval stretches from the `0.5 * (1 - probability)` to `1 - 0.5 * (1 - probability)` percentile of the posterior distribution.
`col`	a colour of the plot
`ylim`	the range of the `y` axis
`ylab`	the label of the `y` axis
`xlab`	the label of the `x` axis
`start_at`	an integer to denote the beginning of the `x` axis range
`add`	a logical value. If `TRUE` the current ribbon plot is added to an existing plot
`...`	other graphical parameters to be passed to `base::plot`

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

Examples

```
data(us_fiscal_lsuw)                                            # upload data
set.seed(123)                                                   # set seed
specification  = specify_bsvar$new(us_fiscal_lsuw)              # specify model

burn_in        = estimate(specification, 10)                    # run the burn-in
posterior      = estimate(burn_in, 20, thin = 1)                # estimate the model
irf            = compute_impulse_responses(posterior, horizon = 4) # impulse responses
plot_ribbon(irf[1,1,,])
```

`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,
  exogenous = NULL,
  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.

`exogenous` a `(T+p)xd` matrix of exogenous variables.

`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`, `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 BSVARMIX presents complete specification for the BSVAR model with a zero-mean mixture of normals model for structural shocks.

Super class

`bsvars::BSVARMSH` -> `BSVARMIX`

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 PriorBSVARMIX with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVARMIX 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, BSVARMIX.

Usage:

```
specify_bsvar_mix$new(
  data,
  p = 1L,
  M = 2L,
  B,
  exogenous = NULL,
  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.

`exogenous` a `(T+p)xd` matrix of exogenous variables.

`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, BSVARMIX.

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`, `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 BSVARMSH 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 PriorBSVARMSH with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVARMSH 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, BSVARMSH.

Usage:

```
specify_bsvar_msh$new(
  data,
  p = 1L,
  M = 2L,
  B,
  exogenous = NULL,
  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.

`exogenous` a `(T+p)xd` matrix of exogenous variables.

`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, BSVARMSH.

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 PriorBSVARMSH 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 StartingValuesBSVARMSH 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, 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 BSVARSV 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 PriorBSVARSV with the prior specification.

`data_matrices` an object DataMatricesBSVAR with the data matrices.

`starting_values` an object StartingValuesBSVARSV with the starting values.

`centred_sv` a logical value - if true a centred parameterisation of the Stochastic Volatility process is estimated. Otherwise, its non-centred parameterisation is estimated. See Lütkepohl, Shang, Uzeda, Woźniak (2022) for more info.

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, BSVARSV.

Usage:

```
specify_bsvar_sv$new(
  data,
  p = 1L,
  B,
  exogenous = NULL,
  centred_sv = FALSE,
  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.

`exogenous` a `(T+p)xd` matrix of exogenous variables.

`centred_sv` a logical value. If `FALSE` a non-centred Stochastic Volatility processes for conditional variances are estimated. Otherwise, a centred process is estimated.

`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, BSVARSV.

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 PriorBSVARSV 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 StartingValuesBSVARSV 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, 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, exogenous = NULL)
```

Arguments:

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

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

`exogenous` a `(T+p)xd` matrix of exogenous variables. This matrix should not include a constant term.

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 `estimate()`.

`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$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate(specification, 50)
estimate$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 `estimate()`.

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(specification, 10)

# estimate the model
posterior      = estimate(burn_in, 10)
```

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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB      # set negative diagonal elements
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(specification, 10, 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
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`, `specify_bsvar`

Examples

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


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

data(us_fiscal_lsuw)
specification  = specify_bsvar$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate(specification, 50)
estimate$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(specification, 10)

# estimate the model
posterior      = estimate(burn_in, 10)


## ------------------------------------------------
## 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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB        # set negative diagonal elements
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(specification, 10, 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
normalise_posterior(posterior, B_hat)                 # draws in posterior are normalised

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

---

specify_posterior_bsvar_mix

R6 Class Representing PosteriorBSVARMIX

Description

The class PosteriorBSVARMIX 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 BSVARMIX with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

`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 PosteriorBSVARMIX.

Usage:

`specify_posterior_bsvar_mix$new(specification_bsvar, posterior_bsvar)`

Arguments:

`specification_bsvar` an object of class BSVARMIX 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 PosteriorBSVARMIX.

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

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

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(specification, 10, thin = 2)

# estimate the model
posterior      = estimate(burn_in, 10, 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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB      # set negative diagonal elements
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_mix$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)

# estimate the model
posterior      = estimate(specification, 10, 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
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, specify_bsvar_mix

Examples

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


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

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

# estimate the model
posterior      = estimate(burn_in, 10, 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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB        # set negative diagonal elements
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_mix$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)

# estimate the model
posterior      = estimate(specification, 10, 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
normalise_posterior(posterior, B_hat)                 # draws in posterior are normalised

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

specify_posterior_bsvar_msh *R6 Class Representing PosteriorBSVARMSH*

Description

The class PosteriorBSVARMSH 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 BSVARMSH with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

`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 PosteriorBSVARMSH.

Usage:

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

Arguments:

`specification_bsvar` an object of class BSVARMSH 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 PosteriorBSVARMSH.

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

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

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(specification, 10, thin = 2)

# estimate the model
posterior      = estimate(burn_in, 10, 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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB      # set negative diagonal elements
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_msh$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)

# estimate the model
posterior      = estimate(specification, 10, 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
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`, `specify_bsvar_msh`

Examples

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

```
class(estimate)


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

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

# estimate the model
posterior      = estimate(burn_in, 10, 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(specification, 10, 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((-1) * sign(diag(BB))) %*% BB        # set negative diagonal elements
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_msh$new(us_fiscal_lsuw, p = 4, M = 2)
set.seed(123)

# estimate the model
posterior      = estimate(specification, 10, 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
normalise_posterior(posterior, B_hat)                 # draws in posterior are normalised

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

specify_posterior_bsvar_sv

R6 Class Representing PosteriorBSVARSV

Description

The class PosteriorBSVARSV 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 BSVARSV with the last draw of the current MCMC run as the starting value to be passed to the continuation of the MCMC estimation using `estimate()`.

`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 PosteriorBSVARSV.

Usage:

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

Arguments:

`specification_bsvar` an object of class BSVARSV 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 PosteriorBSVARSV.

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$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate(specification, 5, thin = 1)
estimate$get_posterior()
```

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

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(specification, 5, thin = 1)
```

```
# estimate the model
posterior      = estimate(burn_in, 5, thin = 1)
```

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(specification, 5, 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((-1) * sign(diag(BB))) %*% BB      # set negative diagonal elements
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(specification, 5, 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
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`, `specify_bsvar_sv`

Examples

```
# This is a function that is used within estimate()
data(us_fiscal_lsuw)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 4)
set.seed(123)
estimate       = estimate(specification, 5, thin = 1)
class(estimate)


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

data(us_fiscal_lsuw)
specification  = specify_bsvar_sv$new(us_fiscal_lsuw)
set.seed(123)
estimate       = estimate(specification, 5, thin = 1)
estimate$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(specification, 5, thin = 1)

# estimate the model
posterior      = estimate(burn_in, 5, thin = 1)


## ------------------------------------------------
## 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(specification, 5, 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((-1) * sign(diag(BB))) %*% BB        # set negative diagonal elements
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(specification, 5, 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
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_B` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix B.

`hyper_a_B` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix B.

`hyper_s_BB` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_BB` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_A` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix A.

`hyper_a_A` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix A.

`hyper_s_AA` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

`hyper_nu_AA` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

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, d = 0, 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.

`d` a positive integer - the number of `exogenous` variables in the 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 PriorBSVARMIX

Description

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

Super classes

`bsvars::PriorBSVAR` -> `bsvars::PriorBSVARMSH` -> `PriorBSVARMIX`

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_B` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix B.

`hyper_a_B` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix B.

`hyper_s_BB` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

hyper_nu_BB a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

hyper_nu_A a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix A.

hyper_a_A a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix A.

hyper_s_AA a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

hyper_nu_AA a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

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 PriorBSVARMSH

Description

The class PriorBSVARMSH presents a prior specification for the bsvar model with Markov Switching Heteroskedasticity.

Super class

`bsvars::PriorBSVAR` -> `PriorBSVARMSH`

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_B` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix B.

`hyper_a_B` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix B.

`hyper_s_BB` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_BB` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_A` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix A.

`hyper_a_A` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix A.

`hyper_s_AA` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

`hyper_nu_AA` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

`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 PriorBSVARMSH.

Usage:

```
specify_prior_bsvar_msh$new(N, p, d = 0, 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.

`d` a positive integer - the number of `exogenous` variables in the 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 PriorBSVARMSH.

Method `get_prior()`**:** Returns the elements of the prior specification PriorBSVARMSH 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 PriorBSVARSV*

Description

The class PriorBSVARSV presents a prior specification for the bsvar model with Stochastic Volatility heteroskedasticity.

Super class

`bsvars::PriorBSVAR` -> `PriorBSVARSV`

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_B` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix B.

`hyper_a_B` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix B.

`hyper_s_BB` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_BB` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix B.

`hyper_nu_A` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the overall shrinkage parameter for matrix A.

`hyper_a_A` a positive scalar, the shape parameter of the gamma prior for the second-level hierarchy for the overall shrinkage parameter for matrix A.

`hyper_s_AA` a positive scalar, the scale parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

`hyper_nu_AA` a positive scalar, the shape parameter of the inverted-gamma 2 prior for the third-level of hierarchy for overall shrinkage parameter for matrix A.

`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 PriorBSVARSV.

Usage:

```
specify_prior_bsvar_sv$new(N, p, d = 0, 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.

`d` a positive integer - the number of `exogenous` variables in the 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 PriorBSVARSV.

Method `get_prior()`**:** Returns the elements of the prior specification PriorBSVARSV 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 `(2*N+1)x2` matrix 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, d = 0)
```

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.

`d` a positive integer - the number of `exogenous` variables in the 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 `estimate()`.

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 StartingValuesBSVARMIX*

Description

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

Super classes

`bsvars::StartingValuesBSVAR` -> `bsvars::StartingValuesBSVARMSH` -> `StartingValuesBSVARMIX`

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 `(2*N+1)x2` matrix 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 StartingValuesBSVARMIX.

Usage:

```
specify_starting_values_bsvar_mix$new(N, p, M, T, d = 0, 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.

`d` a positive integer - the number of `exogenous` variables in the model.

`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 StartingValuesBSVARMIX.

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 StartingValuesBSVARMSH*

Description

The class StartingValuesBSVARMSH presents starting values for the bsvar model with Markov Switching Heteroskedasticity.

Super class

`bsvars::StartingValuesBSVAR` -> `StartingValuesBSVARMSH`

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 `(2*N+1)x2` matrix 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, d = 0, 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.

`d` a positive integer - the number of `exogenous` variables in the model.

`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 StartingValues-BSVARMSH 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 `estimate()`.

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 StartingValuesBSVARSV*

Description

The class StartingValuesBSVARSV presents starting values for the bsvar model with Stochastic Volatility heteroskedasticity.

Super class

`bsvars::StartingValuesBSVAR` -> `StartingValuesBSVARSV`

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 `(2*N+1)x2` matrix 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.

`sigma2v` an `N`-vector with values of SV process conditional variances.

`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 StartingValuesBSVARSV.

Usage:

```
specify_starting_values_bsvar_sv$new(N, p, T, d = 0)
```

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.

`d` a positive integer - the number of `exogenous` variables in the model.

Returns: Starting values StartingValuesBSVARSV.

Method `get_starting_values()`**:** Returns the elements of the starting values StartingValues-BSVARSV 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 `estimate()`.

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
```

`summary.Forecasts` *Provides posterior summary of Forecasts*

Description

Provides posterior summary of the forecasts including their mean, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'Forecasts'
summary(object, ...)
```

Arguments

`object`	an object of class Forecasts obtained using the `forecast()` function containing draws the predictive density.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the forecasts for each of the variables and forecast horizons.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

forecast

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# forecast
fore           = forecast(posterior, horizon = 2)
fore_summary   = summary(fore)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  forecast(horizon = 2) |>
  summary() -> fore_summary
```

summary.PosteriorBSVAR

Provides posterior summary of homoskedastic Structural VAR estimation

Description

Provides posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper parameters.

Usage

```
## S3 method for class 'PosteriorBSVAR'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorBSVAR obtained using the `estimate()` function applied to homoskedastic Bayesian Structural VAR model specification set by function `specify_bsvar$new()` containing draws from the posterior distribution of the parameters.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper-parameters.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `specify_bsvar`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)
summary(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  summary()
```

summary.PosteriorBSVARMIX
Provides posterior summary of non-normal Structural VAR estimation

Description

Provides posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper parameters.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorBSVARMIX obtained using the `estimate()` function applied to non-normal Bayesian Structural VAR model specification set by function `specify_bsvar_mix$new()` containing draws from the posterior distribution of the parameters.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper-parameters.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `specify_bsvar_mix`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_mix$new(us_fiscal_lsuw)

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

# estimate the model
```

```
posterior      = estimate(burn_in, 20, thin = 1)
summary(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  summary()
```

`summary.PosteriorBSVARMSH` *Provides posterior summary of heteroskedastic Structural VAR estimation*

Description

Provides posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper parameters.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorBSVARMSH obtained using the `estimate()` function applied to heteroskedastic Bayesian Structural VAR model specification set by function `specify_bsvar_msh$new()` containing draws from the posterior distribution of the parameters.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper-parameters.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`estimate`, `specify_bsvar_msh`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw)

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)
summary(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  summary()
```

---

summary.PosteriorBSVARSV

Provides posterior summary of heteroskedastic Structural VAR estimation

---

Description

Provides posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper parameters.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorBSVARSV obtained using the `estimate()` function applied to heteroskedastic Bayesian Structural VAR model specification set by function `specify_bsvar_sv$new()` containing draws from the posterior distribution of the parameters.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the parameters: the structural matrix B, autoregressive parameters A, and hyper-parameters.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

estimate, specify_bsvar_sv

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)
summary(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  summary()
```

---

summary.PosteriorFEVD *Provides posterior summary of forecast error variance decompositions*

Description

Provides posterior means of the forecast error variance decompositions of each variable at all horizons.

Usage

```
## S3 method for class 'PosteriorFEVD'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorFEVD obtained using the `compute_variance_decompositions()` function containing draws from the posterior distribution of the forecast error variance decompositions.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean of the forecast error variance decompositions of each variable at all horizons.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_variance_decompositions`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute forecast error variance decompositions
fevd           = compute_variance_decompositions(posterior, horizon = 4)
fevd_summary   = summary(fevd)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_variance_decompositions(horizon = 4) |>
  summary() -> fevd_summary
```

summary.PosteriorFitted

Provides posterior summary of variables' fitted values

Description

Provides posterior summary of the fitted values including their mean, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'PosteriorFitted'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorFitted obtained using the `compute_fitted_values()` function containing draws the predictive density of the sample data.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the fitted values for each of the shocks and periods.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

compute_fitted_values

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute fitted values
```

```
fitted         = compute_fitted_values(posterior)
fitted_summary = summary(fitted)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_fitted_values() |>
  summary() -> fitted_summary
```

summary.PosteriorHD *Provides posterior summary of historical decompositions*

Description

Provides posterior means of the historical decompositions variable by variable.

Usage

```
## S3 method for class 'PosteriorHD'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorHD obtained using the `compute_historical_decompositions()` function containing posterior draws of historical decompositions.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior means of historical decompositions for each of the variables.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

compute_historical_decompositions

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute historical decompositions
hds            = compute_historical_decompositions(posterior)
hds_summary    = summary(hds)

# workflow with the pipe |>
############################################################
set.seed(123)
diff(us_fiscal_lsuw) |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_historical_decompositions() |>
  summary() -> hds_summary
```

`summary.PosteriorIR` *Provides posterior summary of impulse responses*

Description

Provides posterior summary of the impulse responses of each variable to each of the shocks at all horizons. Includes their posterior means, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'PosteriorIR'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorIR obtained using the `compute_impulse_responses()` function containing draws from the posterior distribution of the impulse responses.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the impulse responses of each variable to each of the shocks at all horizons.

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

See Also

compute_impulse_responses

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute impulse responses
irf            = compute_impulse_responses(posterior, horizon = 4)
irf_summary    = summary(irf)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_impulse_responses(horizon = 4) |>
  summary() -> irf_summary
```

summary.PosteriorRegimePr

Provides posterior summary of regime probabilities

Description

Provides posterior summary of regime probabilities including their mean, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'PosteriorRegimePr'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorRegimePr obtained using the `compute_regime_probabilities()` function containing posterior draws of regime allocations.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean and standard deviations of the regime probabilities.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_regime_probabilities`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
set.seed(123)
specification  = specify_bsvar_msh$new(us_fiscal_lsuw)

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute regime probabilities
rp             = compute_regime_probabilities(posterior)
rp_summary     = summary(rp)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_regime_probabilities() |>
  summary() -> rp_summary
```

summary.PosteriorShocks

Provides posterior summary of structural shocks

Description

Provides posterior summary of the structural shocks including their mean, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'PosteriorShocks'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorShocks obtained using the `compute_structural_shocks()` function containing draws the posterior distribution of the structural shocks.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the structural shocks for each of the equations and periods.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_structural_shocks`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

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

# estimate the model
posterior      = estimate(burn_in, 20, thin = 1)

# compute structural shocks
```

```
shocks         = compute_structural_shocks(posterior)
shocks_summary = summary(shocks)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new() |>
  estimate(S = 10) |>
  estimate(S = 20, thin = 1) |>
  compute_structural_shocks() |>
  summary() -> shocks_summary
```

---

summary.PosteriorSigma

Provides posterior summary of structural shocks' conditional standard deviations

---

Description

Provides posterior summary of structural shocks' conditional standard deviations including their mean, standard deviations, as well as 5 and 95 percentiles.

Usage

```
## S3 method for class 'PosteriorSigma'
summary(object, ...)
```

Arguments

`object`	an object of class PosteriorSigma obtained using the `compute_conditional_sd()` function containing posterior draws of conditional standard deviations of structural shocks.
`...`	additional arguments affecting the summary produced.

Value

A list reporting the posterior mean, standard deviations, as well as 5 and 95 percentiles of the structural shocks' conditional standard deviations for each of the shocks and periods.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`compute_conditional_sd`

Examples

```
# upload data
data(us_fiscal_lsuw)

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

# run the burn-in
burn_in        = estimate(specification, 5)

# estimate the model
posterior      = estimate(burn_in, 5)

# compute structural shocks' conditional standard deviations
sigma          = compute_conditional_sd(posterior)
sigma_summary  = summary(sigma)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new() |>
  estimate(S = 5) |>
  estimate(S = 5) |>
  compute_conditional_sd() |>
  summary() -> sigma_summary
```

summary.SDDRautoregression

Provides summary of verifying hypotheses about autoregressive parameters

Description

Provides summary of the Savage-Dickey density ratios for verification of hypotheses about autoregressive parameters.

Usage

```
## S3 method for class 'SDDRautoregression'
summary(object, ...)
```

Arguments

`object`	an object of class `SDDRautoregression` obtained using the `verify_autoregression()` function.
`...`	additional arguments affecting the summary produced.

Value

A table reporting the logarithm of Bayes factors of the restriction against no restriction posterior odds in `"log(SDDR)"`, its numerical standard error `"NSE"`, and the implied posterior probability of the restriction holding or not hypothesis, `"Pr[H0|data]"` and `"Pr[H1|data]"` respectively.

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

See Also

`verify_autoregression`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0        # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)
summary(sddr)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) |>
  summary() -> sddr_summary
```

`summary.SDDRvolatility`

Provides summary of verifying homoskedasticity

Description

Provides summary of the Savage-Dickey density ratios for verification of structural shocks homoskedasticity.

Usage

```
## S3 method for class 'SDDRvolatility'
summary(object, ...)
```

Arguments

`object`	an object of class `SDDRvolatility` obtained using the `verify_volatility()` function.
`...`	additional arguments affecting the summary produced.

Value

A table reporting the logarithm of Bayes factors of homoskedastic to heteroskedastic posterior odds `"log(SDDR)"` for each structural shock, their numerical standard errors `"NSE"`, and the implied posterior probability of the homoskedasticity and heteroskedasticity hypothesis, `"Pr[homoskedasticity|data]"` and `"Pr[heteroskedasticity|data]"` respectively.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

See Also

`verify_volatility`

Examples

```
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)
summary(sddr)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() |>
  summary() -> sddr_summary
```

`us_fiscal_ex` *A 3-variable system of exogenous variables for the US fiscal model for the period 1948 Q1 – 2024 Q1*

Description

Exogenous variables used to identify the US fiscal policy shocks. Last data update was implemented on 2024-06-15.

Usage

```
data(us_fiscal_ex)
```

Format

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

t a time trend

t^2 a quadratic trend

1975Q2 a dummy variable taking the value of 1 for quarter 2 1975 and zero elsewhere

The series are as described by Mertens & Ravn (2014). The data was used by Lütkepohl, Shang, Uzeda, Woźniak (2024).

References

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

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.

Examples

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

`us_fiscal_lsuw` *A 3-variable US fiscal system for the period 1948 Q1 – 2024 Q1*

Description

A system used to identify the US fiscal policy shocks. Last data update was implemented on 2024-06-15.

Usage

```
data(us_fiscal_lsuw)
```

Format

A matrix and a `ts` object with time series of over three hundred 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 1948 to the last available observation,
- 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/`

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. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

Examples

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

`verify_autoregression` *Verifies hypotheses involving autoregressive parameters*

Description

Computes the logarithm of Bayes factor for the joint hypothesis, H_0, possibly for many autoregressive parameters represented by argument `hypothesis` via Savage-Dickey Density Ration (SDDR). The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against hypothesis. The estimation of both elements of the difference requires numerical integration.

Usage

```
verify_autoregression(posterior, hypothesis)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome
`hypothesis`	an `NxK` matrix of the same dimension as the autoregressive matrix A with numeric values for the parameters to be verified, in which case the values represent the joint hypothesis, and missing value `NA` for these parameters that are not tested

Value

An object of class `SDDRautoregression` that is a list of three components:

`logSDDR` a scalar with values of the logarithm of the Bayes factors for the autoregressive hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

log_denominator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the denominator

se_components a `30`-vector containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0       # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) -> sddr
```

`verify_autoregression.PosteriorBSVAR`
Verifies hypotheses involving autoregressive parameters

Description

Computes the logarithm of Bayes factor for the joint hypothesis, H_0, possibly for many autoregressive parameters represented by argument `hypothesis` via Savage-Dickey Density Ration (SDDR). The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against hypothesis. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVAR'
verify_autoregression(posterior, hypothesis)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome
`hypothesis`	an `NxK` matrix of the same dimension as the autoregressive matrix A with numeric values for the parameters to be verified, in which case the values represent the joint hypothesis, and missing value `NA` for these parameters that are not tested

Value

An object of class `SDDRautoregression` that is a list of three components:

`logSDDR` a scalar with values of the logarithm of the Bayes factors for the autoregressive hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

log_denominator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the denominator

se_components a `30`-vector containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0        # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) -> sddr
```

---

verify_autoregression.PosteriorBSVARMIX
Verifies hypotheses involving autoregressive parameters

Description

Computes the logarithm of Bayes factor for the joint hypothesis, H_0, possibly for many autoregressive parameters represented by argument `hypothesis` via Savage-Dickey Density Ration (SDDR). The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against hypothesis. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
verify_autoregression(posterior, hypothesis)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome
`hypothesis`	an `NxK` matrix of the same dimension as the autoregressive matrix A with numeric values for the parameters to be verified, in which case the values represent the joint hypothesis, and missing value `NA` for these parameters that are not tested

Value

An object of class `SDDRautoregression` that is a list of three components:

`logSDDR` a scalar with values of the logarithm of the Bayes factors for the autoregressive hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

log_denominator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the denominator

se_components a `30`-vector containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
```

```
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0        # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) -> sddr
```

`verify_autoregression.PosteriorBSVARMSH`
Verifies hypotheses involving autoregressive parameters

Description

Computes the logarithm of Bayes factor for the joint hypothesis, H_0, possibly for many autoregressive parameters represented by argument `hypothesis` via Savage-Dickey Density Ration (SDDR). The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against hypothesis. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
verify_autoregression(posterior, hypothesis)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome
`hypothesis`	an `NxK` matrix of the same dimension as the autoregressive matrix A with numeric values for the parameters to be verified, in which case the values represent the joint hypothesis, and missing value `NA` for these parameters that are not tested

Value

An object of class `SDDRautoregression` that is a list of three components:

`logSDDR` a scalar with values of the logarithm of the Bayes factors for the autoregressive hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

log_denominator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the denominator

se_components a `30`-vector containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak `<wozniak.tom@pm.me>`

References

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0        # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)

# workflow with the pipe |>
############################################################
```

```
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) -> sddr
```

---

verify_autoregression.PosteriorBSVARSV

Verifies hypotheses involving autoregressive parameters

Description

Computes the logarithm of Bayes factor for the joint hypothesis, H_0, possibly for many autoregressive parameters represented by argument `hypothesis` via Savage-Dickey Density Ration (SDDR). The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against hypothesis. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
verify_autoregression(posterior, hypothesis)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome
`hypothesis`	an `NxK` matrix of the same dimension as the autoregressive matrix A with numeric values for the parameters to be verified, in which case the values represent the joint hypothesis, and missing value `NA` for these parameters that are not tested

Value

An object of class `SDDRautoregression` that is a list of three components:

`logSDDR` a scalar with values of the logarithm of the Bayes factors for the autoregressive hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

log_denominator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the denominator

se_components a `30`-vector containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

Woźniak, T., and Droumaguet, M., (2024) Bayesian Assessment of Identifying Restrictions for Heteroskedastic Structural VARs

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify autoregression
H0             = matrix(NA, ncol(us_fiscal_lsuw), ncol(us_fiscal_lsuw) + 1)
H0[1,3]        = 0        # a hypothesis of no Granger causality from gdp to ttr
sddr           = verify_autoregression(posterior, H0)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_autoregression(hypothesis = H0) -> sddr
```

`verify_volatility` *Verifies heteroskedasticity of structural shocks equation by equation*

Description

Computes the logarithm of Bayes factor for the homoskedasticity hypothesis for each of the structural shocks via Savage-Dickey Density Ration (SDDR). The hypothesis of homoskedasticity, H_0, is represented by model-specific restrictions. Consult help files for individual classes of models for details. The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(H_0|data) - logp(H_0)$$

Therefore, a negative value of the difference is the evidence against homoskedasticity of the structural shock. The estimation of both elements of the difference requires numerical integration.

Usage

```
verify_volatility(posterior)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome

Value

An object of class `SDDRvolatility` that is a list of three components:

`logSDDR` an `N`-vector with values of the logarithm of the Bayes factors for the homoskedasticity hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

se_components an `Nx30` matrix containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() -> sddr
```

---

verify_volatility.PosteriorBSVAR
Verifies heteroskedasticity of structural shocks equation by equation

---

Description

Displays information that the model is homoskedastic.

Usage

```
## S3 method for class 'PosteriorBSVAR'
verify_volatility(posterior)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome

Value

Nothing. Just displays a message: The model is homoskedastic.

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() -> sddr
```

`verify_volatility.PosteriorBSVARMIX`
Verifies heteroskedasticity of structural shocks equation by equation

---

Description

Computes the logarithm of Bayes factor for the homoskedasticity hypothesis for each of the structural shocks via Savage-Dickey Density Ration (SDDR). The hypothesis of homoskedasticity is represented by restriction:

$$H_0 : \sigma^2_{n.1} = ... = \sigma^2_{n.M} = 1$$

The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$log p(\omega_n = 0|data) - log p(\omega_n = 0)$$

Therefore, a negative value of the difference is the evidence against homoskedasticity of the structural shock. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARMIX'
verify_volatility(posterior)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome

Value

An object of class `SDDRvolatility` that is a list of three components:

`logSDDR` an `N`-vector with values of the logarithm of the Bayes factors for the homoskedasticity hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

se_components an `Nx30` matrix containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <wozniak.tom@pm.me>

References

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

See Also

specify_bsvar_mix, estimate

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_mix$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_mix$new(p = 1, M = 2) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() -> sddr
```

---

verify_volatility.PosteriorBSVARMSH

Verifies heteroskedasticity of structural shocks equation by equation

---

Description

Computes the logarithm of Bayes factor for the homoskedasticity hypothesis for each of the structural shocks via Savage-Dickey Density Ration (SDDR). The hypothesis of homoskedasticity is represented by restriction:

$$H_0 : \sigma^2_{n.1} = ... = \sigma^2_{n.M} = 1$$

The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(\omega_n = 0|data) - logp(\omega_n = 0)$$

Therefore, a negative value of the difference is the evidence against homoskedasticity of the structural shock. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARMSH'
verify_volatility(posterior)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome

Value

An object of class `SDDRvolatility` that is a list of three components:

`logSDDR` an `N`-vector with values of the logarithm of the Bayes factors for the homoskedasticity hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

se_components an `Nx30` matrix containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

See Also

specify_bsvar_msh, estimate

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_msh$new(us_fiscal_lsuw, p = 1, M = 2)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_msh$new(p = 1, M = 2) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() -> sddr
```

verify_volatility.PosteriorBSVARSV

Verifies heteroskedasticity of structural shocks equation by equation

Description

Computes the logarithm of Bayes factor for the homoskedasticity hypothesis for each of the structural shocks via Savage-Dickey Density Ration (SDDR). The hypothesis of homoskedasticity is represented by restriction:

$$H_0 : \omega_n = 0$$

The logarithm of Bayes factor for this hypothesis can be computed using the SDDR as the difference of logarithms of the marginal posterior distribution ordinate at the restriction less the marginal prior distribution ordinate at the same point:

$$logp(\omega_n = 0|data) - logp(\omega_n = 0)$$

Therefore, a negative value of the difference is the evidence against homoskedasticity of the structural shock. The estimation of both elements of the difference requires numerical integration.

Usage

```
## S3 method for class 'PosteriorBSVARSV'
verify_volatility(posterior)
```

Arguments

`posterior`	the `posterior` element of the list from the estimation outcome

Value

An object of class `SDDRvolatility` that is a list of three components:

`logSDDR` an `N`-vector with values of the logarithm of the Bayes factors for the homoskedasticity hypothesis for each of the shocks

`log_SDDR_se` an `N`-vector with estimation standard errors of the logarithm of the Bayes factors reported in output element `logSDDR` that are computed based on 30 random sub-samples of the log-ordinates of the marginal posterior and prior distributions.

`components` a list of three components for the computation of the Bayes factor

log_denominator an `N`-vector with values of the logarithm of the Bayes factor denominators

log_numerator an `N`-vector with values of the logarithm of the Bayes factor numerators

log_numerator_s an `NxS` matrix of the log-full conditional posterior density ordinates computed to estimate the numerator

se_components an `Nx30` matrix containing the log-Bayes factors on the basis of which the standard errors are computed

Author(s)

Tomasz Woźniak <`wozniak.tom@pm.me`>

References

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.

Lütkepohl, H., Shang, F., Uzeda, L., and Woźniak, T. (2024) Partial Identification of Heteroskedastic Structural VARs: Theory and Bayesian Inference. *University of Melbourne Working Paper*, 1–57, doi:10.48550/arXiv.2404.11057.

See Also

specify_bsvar_sv, estimate

Examples

```
# simple workflow
############################################################
# upload data
data(us_fiscal_lsuw)

# specify the model and set seed
specification  = specify_bsvar_sv$new(us_fiscal_lsuw, p = 1)
set.seed(123)

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

# verify heteroskedasticity
sddr           = verify_volatility(posterior)

# workflow with the pipe |>
############################################################
set.seed(123)
us_fiscal_lsuw |>
  specify_bsvar_sv$new(p = 1) |>
  estimate(S = 10, thin = 1) |>
  verify_volatility() -> sddr
```

Index