Package ‘bvhar’

October 11, 2024

Type Package

Title Bayesian Vector Heterogeneous Autoregressive Modeling

Version 2.1.2

Description Tools to model and forecast multivariate time series including Bayesian Vector heterogeneous autoregressive (VHAR) model by Kim & Baek (2023) (<doi:10.1080/00949655.2023.2281644>). 'bvhar' can model Vector Autoregressive (VAR), VHAR, Bayesian VAR (BVAR), and Bayesian VHAR (BVHAR) models.

License GPL (>= 3)

URL https://ygeunkim.github.io/package/bvhar/, https://github.com/ygeunkim/bvhar

BugReports https://github.com/ygeunkim/bvhar/issues

Suggests covr, GIGrvg, knitr, parallel, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports lifecycle, magrittr, Rcpp, ggplot2, tidyr, tibble, dplyr, foreach, purrr, stats, optimParallel, posterior, bayesplot

LinkingTo BH (>= 1.84.0-0), Rcpp, RcppEigen(>= 0.3.4.0.0)

VignetteBuilder knitr

Depends R (>= 3.6.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Young Geun Kim [aut, cre, cph] (<https://orcid.org/0000-0001-8651-1167>), Changryong Baek [ctb]

Maintainer Young Geun Kim <ygeunkimstat@gmail.com>

Repository CRAN

Date/Publication 2024-10-11 15:20:02 UTC

Contents

autoplot.bvhardynsp . . . 4
autoplot.bvharirf . . . 4
autoplot.bvharsp . . . 5
autoplot.normaliw . . . 6
autoplot.predbvhar . . . 6
autoplot.summary.bvharsp . . . 7
autoplot.summary.normaliw . . . 8
bound_bvhar . . . 9
bvar_flat . . . 10
bvar_horseshoe . . . 12
bvar_minnesota . . . 14
bvar_ssvs . . . 17
bvar_sv . . . 20
bvhar_horseshoe . . . 22
bvhar_minnesota . . . 24
bvhar_ssvs . . . 27
bvhar_sv . . . 30
choose_bayes . . . 32
choose_bvar . . . 33
choose_ssvs . . . 35
choose_var . . . 37
coef . . . 37
compute_dic . . . 38
compute_logml . . . 39
confusion . . . 40
conf_fdr . . . 41
conf_fnr . . . 42
conf_fscore . . . 43
conf_prec . . . 44
conf_recall . . . 45
divide_ts . . . 46
dynamic_spillover . . . 46
etf_vix . . . 48
fitted . . . 49
forecast_expand . . . 50
forecast_roll . . . 52
FPE . . . 54
fromse . . . 55
geom_eval . . . 56
gg_loss . . . 56
HQ . . . 57
init_ssvs . . . 59
irf.varlse . . . 60
is.stable . . . 62
mae . . . 63
mape . . . 64

mase . . . 65
mrae . . . 66
mse . . . 67
predict . . . 68
print.summary.bvharsp . . . 73
relmae . . . 74
relspne . . . 75
residuals . . . 76
rmafe . . . 77
rmape . . . 78
rmase . . . 79
rmsfe . . . 80
set_bvar . . . 81
set_dl . . . 84
set_horseshoe . . . 85
set_intercept . . . 86
set_lambda . . . 86
set_ldlt . . . 88
set_ng . . . 89
set_ssvs . . . 90
sim_gig . . . 93
sim_horseshoe_var . . . 94
sim_iw . . . 95
sim_matgaussian . . . 95
sim_mncoef . . . 96
sim_mniw . . . 97
sim_mnormal . . . 98
sim_mnvhar_coef . . . 99
sim_mvt . . . 100
sim_ssvs_var . . . 101
sim_var . . . 103
sim_vhar . . . 104
spillover . . . 105
spne . . . 106
stableroot . . . 107
summary.normaliw . . . 108
summary.varlse . . . 110
summary.vharlse . . . 111
VARtoVMA . . . 113
var_bayes . . . 114
var_lm . . . 116
VHARtoVMA . . . 120
vhar_bayes . . . 120
vhar_lm . . . 123

Index **126**

autoplot.bvhardynsp *Dynamic Spillover Indices Plot*

Description

Draws dynamic directional spillover plot.

Usage

```
## S3 method for class 'bvhardynsp'
autoplot(
  object,
  type = c("tot", "to", "from", "net"),
  hcol = "grey",
  hsize = 1.5,
  row_facet = NULL,
  col_facet = NULL,
  ...
)
```

Arguments

object	A bvhardynsp object
type	Index to draw
hcol	color of horizontal line = 0 (By default, grey)
hsize	size of horizontal line = 0 (By default, 1.5)
row_facet	nrow of ggplot2::facet_wrap()
col_facet	ncol of ggplot2::facet_wrap()
...	Additional

autoplot.bvharirf *Plot Impulse Responses*

Description

Draw impulse responses of response ~ impulse in the facet.

Usage

```
## S3 method for class 'bvharirf'
autoplot(object, ...)
```

Arguments

`object`	A bvharirf object
`...`	Other arguments passed on the `ggplot2::geom_path()`.

Value

A ggplot object

See Also

`irf()`

---

autoplot.bvharsp — *Plot the Result of BVAR and BVHAR MCMC*

Description

Draw BVAR and BVHAR MCMC plots.

Usage

```
## S3 method for class 'bvharsp'
autoplot(
  object,
  type = c("coef", "trace", "dens", "area"),
  pars = character(),
  regex_pars = character(),
  ...
)
```

Arguments

`object`	A bvharsp object
`type`	The type of the plot. Posterior coefficient (`coef`), Trace plot (`trace`), kernel density plot (`dens`), and interval estimates plot (`area`).
`pars`	Parameter names to draw.
`regex_pars`	Regular expression parameter names to draw.
`...`	Other options for each `bayesplot::mcmc_trace()`, `bayesplot::mcmc_dens()`, and `bayesplot::mcmc_areas()`.

Value

A ggplot object

`autoplot.normaliw` *Residual Plot for Minnesota Prior VAR Model*

Description

This function draws residual plot for covariance matrix of Minnesota prior VAR model.

Usage

```
## S3 method for class 'normaliw'
autoplot(object, hcol = "grey", hsize = 1.5, ...)
```

Arguments

`object`	A `normaliw` object
`hcol`	color of horizontal line = 0 (By default, grey)
`hsize`	size of horizontal line = 0 (By default, 1.5)
`...`	additional options for geom_point

Value

A ggplot object

`autoplot.predbvhar` *Plot Forecast Result*

Description

Plots the forecasting result with forecast regions.

Usage

```
## S3 method for class 'predbvhar'
autoplot(
  object,
  type = c("grid", "wrap"),
  ci_alpha = 0.7,
  alpha_scale = 0.3,
  x_cut = 1,
  viridis = FALSE,
  viridis_option = "D",
  NROW = NULL,
  NCOL = NULL,
  ...
)
```

```
## S3 method for class 'predbvhar'
autolayer(object, ci_fill = "grey70", ci_alpha = 0.5, alpha_scale = 0.3, ...)
```

Arguments

`object`	A `predbvhar` object
`type`	Divide variables using `ggplot2::facet_grid()` ("grid": default) or `ggplot2::facet_wrap()` ("wrap")
`ci_alpha`	Transparency of CI
`alpha_scale`	Scale of transparency parameter (`alpha`) between the two layers. `alpha` of CI ribbon = `alpha_scale` * `alpha` of path (By default, .5)
`x_cut`	plot x axes from `x_cut` for visibility
`viridis`	If `TRUE`, scale CI and forecast line using `ggplot2::scale_fill_viridis_d()` and ggplot2::scale_colour_viridis_d, respectively.
`viridis_option`	Option for viridis string. See `option` of ggplot2::scale_colour_viridis_d. Choose one of `c("A", "B", "C", "D", "E")`. By default, `D`.
`NROW`	`nrow` of `ggplot2::facet_wrap()`
`NCOL`	`ncol` of `ggplot2::facet_wrap()`
`...`	additional option for `ggplot2::geom_path()`
`ci_fill`	color of CI

Value

A ggplot object

A ggplot layer

`autoplot.summary.bvharsp` *Plot the Heatmap of SSVS Coefficients*

Description

Draw heatmap for SSVS prior coefficients.

Usage

```
## S3 method for class 'summary.bvharsp'
autoplot(object, point = FALSE, ...)
```

Arguments

`object`	A `summary.bvharsp` object
`point`	Use point for sparsity representation
`...`	Other arguments passed on the `ggplot2::geom_tile()`.

Value

A ggplot object

---

autoplot.summary.normaliw *Density Plot for Minnesota Prior VAR Model*

Description

This function draws density plot for coefficient matrices of Minnesota prior VAR model.

Usage

```
## S3 method for class 'summary.normaliw'
autoplot(
  object,
  type = c("trace", "dens", "area"),
  pars = character(),
  regex_pars = character(),
  ...
)
```

Arguments

object	A summary.normaliw object
type	The type of the plot. Trace plot (trace), kernel density plot (dens), and interval estimates plot (area).
pars	Parameter names to draw.
regex_pars	Regular expression parameter names to draw.
...	Other options for each bayesplot::mcmc_trace(), bayesplot::mcmc_dens(), and bayesplot::mcmc_areas().

Value

A ggplot object

bound_bvhar *Setting Empirical Bayes Optimization Bounds*

Description

[Experimental] This function sets lower and upper bounds for `set_bvar()`, `set_bvhar()`, or `set_weight_bvhar()`.

Usage

```
bound_bvhar(
  init_spec = set_bvhar(),
  lower_spec = set_bvhar(),
  upper_spec = set_bvhar()
)

## S3 method for class 'boundbvharemp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.boundbvharemp(x)

## S3 method for class 'boundbvharemp'
knit_print(x, ...)
```

Arguments

`init_spec`	Initial Bayes model specification
`lower_spec`	Lower bound Bayes model specification
`upper_spec`	Upper bound Bayes model specification
`x`	`boundbvharemp` object
`digits`	digit option to print
`...`	not used

Value

`boundbvharemp` class

bvar_flat *Fitting Bayesian VAR(p) of Flat Prior*

Description

This function fits BVAR(p) with flat prior.

Usage

```
bvar_flat(
  y,
  p,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvar_flat(),
  include_mean = TRUE,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarflat'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

## S3 method for class 'bvarflat'
AIC(object, ...)

## S3 method for class 'bvarflat'
BIC(object, ...)

is.bvarflat(x)

## S3 method for class 'bvarflat'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
p	VAR lag
num_chains	Number of MCMC chains
num_iter	MCMC iteration number
num_burn	Number of burn-in (warm-up). Half of the iteration is the default choice.

thinning	Thinning every thinning-th iteration
bayes_spec	A BVAR model specification by set_bvar_flat().
include_mean	Add constant term (Default: TRUE) or not (FALSE)
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvarflat object
digits	digit option to print
...	not used
object	A bvarflat object

Details

Ghosh et al. (2018) gives flat prior for residual matrix in BVAR.

Under this setting, there are many models such as hierarchical or non-hierarchical. This function chooses the most simple non-hierarchical matrix normal prior in Section 3.1.

$$A \mid \Sigma_e \sim MN(0, U^{-1}, \Sigma_e)$$

where U: precision matrix (MN: matrix normal).

$$p(\Sigma_e) \propto 1$$

Value

bvar_flat() returns an object bvarflat class. It is a list with the following components:

coefficients Posterior Mean matrix of Matrix Normal distribution

fitted.values Fitted values

residuals Residuals

mn_prec Posterior precision matrix of Matrix Normal distribution

iw_scale Posterior scale matrix of posterior inverse-wishart distribution

iw_shape Posterior shape of inverse-wishart distribution

df Numer of Coefficients: mp + 1 or mp

p Lag of VAR

m Dimension of the time series

obs Sample size used when training = totobs - p

totobs Total number of the observation

process Process string in the bayes_spec: BVAR_Flat

spec Model specification (bvharspec)

type include constant term (const) or not (none)

call Matched call

prior_mean Prior mean matrix of Matrix Normal distribution: zero matrix

prior_precision Prior precision matrix of Matrix Normal distribution: U^{-1}

y0 Y_0

design X_0

y Raw input (matrix)

References

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

See Also

- `set_bvar_flat()` to specify the hyperparameters of BVAR flat prior.
- `coef.bvarflat()`, `residuals.bvarflat()`, and `fitted.bvarflat()`
- `predict.bvarflat()` to forecast the BVHAR process

`bvar_horseshoe` *Fitting Bayesian VAR(p) of Horseshoe Prior*

Description

[Deprecated] This function fits BVAR(p) with horseshoe prior.

Usage

```
bvar_horseshoe(
  y,
  p,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_horseshoe(),
  include_mean = TRUE,
  minnesota = FALSE,
  algo = c("block", "gibbs"),
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarhs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvarhs'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
p	VAR lag
num_chains	Number of MCMC chains
num_iter	MCMC iteration number
num_burn	Number of burn-in (warm-up). Half of the iteration is the default choice.
thinning	Thinning every thinning-th iteration
bayes_spec	Horseshoe initialization specification by set_horseshoe().
include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Minnesota type
algo	Ordinary gibbs sampling (gibbs) or blocked gibbs (Default: block).
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	**[Experimental]** Number of threads
x	bvarhs object
digits	digit option to print
...	not used

Value

bvar_horseshoe returns an object named bvarhs class. It is a list with the following components:

coefficients Posterior mean of VAR coefficients.

covmat Posterior mean of covariance matrix

psi_posterior Posterior mean of precision matrix Ψ

pip Posterior inclusion probabilities.

param posterior::draws_df with every variable: alpha, lambda, tau, omega, and eta

param_names Name of every parameter.

df Numer of Coefficients: mp + 1 or mp

p Lag of VAR

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VAR_Horseshoe

type include constant term (const) or not (none)

algo Usual Gibbs sampling (gibbs) or fast sampling (fast)

spec Horseshoe specification defined by set_horseshoe()

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

group Indicators for group.

num_group Number of groups.

y0 Y_0

design X_0

y Raw input

References

Carvalho, C. M., Polson, N. G., & Scott, J. G. (2010). *The horseshoe estimator for sparse signals*. Biometrika, 97(2), 465-480.

Makalic, E., & Schmidt, D. F. (2016). *A Simple Sampler for the Horseshoe Estimator*. IEEE Signal Processing Letters, 23(1), 179-182.

`bvar_minnesota` *Fitting Bayesian VAR(p) of Minnesota Prior*

Description

This function fits BVAR(p) with Minnesota prior.

Usage

```
bvar_minnesota(
  y,
  p = 1,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvar(),
  scale_variance = 0.05,
  include_mean = TRUE,
  parallel = list(),
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarmn'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvarhm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

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

## S3 method for class 'bvarmn'
AIC(object, ...)

## S3 method for class 'bvarmn'
BIC(object, ...)

is.bvarmn(x)

## S3 method for class 'bvarmn'
knit_print(x, ...)

## S3 method for class 'bvarhm'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`p`	VAR lag (Default: 1)
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A BVAR model specification by `set_bvar()`.
`scale_variance`	Proposal distribution scaling constant to adjust an acceptance rate
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`parallel`	List the same argument of `optimParallel::optimParallel()`. By default, this is empty, and the function does not execute parallel computation.
`verbose`	Print the progress bar in the console. By default, `FALSE`.
`num_thread`	Number of threads
`x`	`bvarhm` object
`digits`	digit option to print
`...`	not used
`object`	A `bvarmn` object

Details

Minnesota prior gives prior to parameters A (VAR matrices) and Σ_e (residual covariance).

$$A \mid \Sigma_e \sim MN(A_0, \Omega_0, \Sigma_e)$$
$$\Sigma_e \sim IW(S_0, \alpha_0)$$

(MN: matrix normal, IW: inverse-wishart)

Value

bvar_minnesota() returns an object bvarmn class. It is a list with the following components:

coefficients Posterior Mean

fitted.values Fitted values

residuals Residuals

mn_mean Posterior mean matrix of Matrix Normal distribution

mn_prec Posterior precision matrix of Matrix Normal distribution

iw_scale Posterior scale matrix of posterior inverse-Wishart distribution

iw_shape Posterior shape of inverse-Wishart distribution ($alpha_0$ - obs + 2). α_0: nrow(Dummy observation) - k

df Numer of Coefficients: mp + 1 or mp

m Dimension of the time series

obs Sample size used when training = totobs - p

prior_mean Prior mean matrix of Matrix Normal distribution: A_0

prior_precision Prior precision matrix of Matrix Normal distribution: Ω_0^{-1}

prior_scale Prior scale matrix of inverse-Wishart distribution: S_0

prior_shape Prior shape of inverse-Wishart distribution: α_0

y0 Y_0

design X_0

p Lag of VAR

totobs Total number of the observation

type include constant term (const) or not (none)

y Raw input (matrix)

call Matched call

process Process string in the bayes_spec: BVAR_Minnesota

spec Model specification (bvharspec)

It is also normaliw and bvharmod class.

References

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). *Prior Selection for Vector Autoregressions*. Review of Economics and Statistics, 97(2).

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

KADIYALA, K.R. and KARLSSON, S. (1997), *NUMERICAL METHODS FOR ESTIMATION AND INFERENCE IN BAYESIAN VAR-MODELS*. J. Appl. Econ., 12: 99-132.

Karlsson, S. (2013). *Chapter 15 Forecasting with Bayesian Vector Autoregression*. Handbook of Economic Forecasting, 2, 791-897.

Sims, C. A., & Zha, T. (1998). *Bayesian Methods for Dynamic Multivariate Models*. International Economic Review, 39(4), 949-968.

See Also

- `set_bvar()` to specify the hyperparameters of Minnesota prior.
- `summary.normaliw()` to summarize BVAR model

Examples

```
# Perform the function using etf_vix dataset
fit <- bvar_minnesota(y = etf_vix[,1:3], p = 2)
class(fit)

# Extract coef, fitted values, and residuals
coef(fit)
head(residuals(fit))
head(fitted(fit))
```

`bvar_ssvs` *Fitting Bayesian VAR(p) of SSVS Prior*

Description

[Deprecated] This function fits BVAR(p) with stochastic search variable selection (SSVS) prior.

Usage

```
bvar_ssvs(
  y,
  p,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
 bayes_spec = choose_ssvs(y = y, ord = p, type = "VAR", param = c(0.1, 10), include_mean
   = include_mean, gamma_param = c(0.01, 0.01), mean_non = 0, sd_non = 0.1),
  init_spec = init_ssvs(type = "auto"),
  include_mean = TRUE,
  minnesota = FALSE,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarssvs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvarssvs'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`p`	VAR lag
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A SSVS model specification by `set_ssvs()`. By default, use a default semiautomatic approach `choose_ssvs()`.
`init_spec`	SSVS initialization specification by `init_ssvs()`. By default, use OLS for coefficient and cholesky factor while 1 for dummies.
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`minnesota`	Apply cross-variable shrinkage structure (Minnesota-way). By default, `FALSE`.
`verbose`	Print the progress bar in the console. By default, `FALSE`.
`num_thread`	**[Experimental]** Number of threads
`x`	`bvarssvs` object
`digits`	digit option to print
`...`	not used

Details

SSVS prior gives prior to parameters $\alpha = vec(A)$ (VAR coefficient) and $\Sigma_e^{-1} = \Psi \Psi^T$ (residual covariance).

$$\alpha_j \mid \gamma_j \sim (1 - \gamma_j) N(0, \kappa_{0j}^2) + \gamma_j N(0, \kappa_{1j}^2)$$
$$\gamma_j \sim Bernoulli(q_j)$$

and for upper triangular matrix Ψ,

$$\psi_{jj}^2 \sim Gamma(shape = a_j, rate = b_j)$$
$$\psi_{ij} \mid w_{ij} \sim (1 - w_{ij}) N(0, \kappa_{0,ij}^2) + w_{ij} N(0, \kappa_{1,ij}^2)$$
$$w_{ij} \sim Bernoulli(q_{ij})$$

Value

`bvar_ssvs` returns an object named `bvarssvs` class. It is a list with the following components:

coefficients Posterior mean of VAR coefficients.

chol_posterior Posterior mean of cholesky factor matrix

covmat Posterior mean of covariance matrix

omega_posterior Posterior mean of omega

pip Posterior inclusion probability

param posterior::draws_df with every variable: alpha, eta, psi, omega, and gamma

param_names Name of every parameter.

df Numer of Coefficients: mp + 1 or mp

p Lag of VAR

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VAR_SSVS

type include constant term (const) or not (none)

spec SSVS specification defined by set_ssvs()

init Initial specification defined by init_ssvs()

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

group Indicators for group.

num_group Number of groups.

y0 Y_0

design X_0

y Raw input

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

bvar_sv *Fitting Bayesian VAR-SV*

Description

[Deprecated] This function fits VAR-SV. It can have Minnesota, SSVS, and Horseshoe prior.

Usage

```
bvar_sv(
  y,
  p,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvar(),
  sv_spec = set_sv(),
  intercept = set_intercept(),
  include_mean = TRUE,
  minnesota = TRUE,
  save_init = FALSE,
  convergence = NULL,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarsv'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvarsv'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
p	VAR lag
num_chains	Number of MCMC chains
num_iter	MCMC iteration number
num_burn	Number of burn-in (warm-up). Half of the iteration is the default choice.
thinning	Thinning every thinning-th iteration
bayes_spec	A BVAR model specification by set_bvar(), set_ssvs(), or set_horseshoe().
sv_spec	**[Experimental]** SV specification by set_sv().
intercept	**[Experimental]** Prior for the constant term by set_intercept().

include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Apply cross-variable shrinkage structure (Minnesota-way). By default, TRUE.
save_init	Save every record starting from the initial values (TRUE). By default, exclude the initial values in the record (FALSE), even when num_burn = 0 and thinning = 1. If num_burn > 0 or thinning != 1, this option is ignored.
convergence	Convergence threshold for rhat < convergence. By default, NULL which means no warning.
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvarsv object
digits	digit option to print
...	not used

Details

Cholesky stochastic volatility modeling for VAR based on

$$\Sigma_t^{-1} = L^T D_t^{-1} L$$

, and implements corrected triangular algorithm for Gibbs sampler.

Value

bvar_sv() returns an object named bvarsv class.

coefficients Posterior mean of coefficients.

chol_posterior Posterior mean of contemporaneous effects.

param Every set of MCMC trace.

param_names Name of every parameter.

group Indicators for group.

num_group Number of groups.

df Numer of Coefficients: 3m + 1 or 3m

p VAR lag

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VHAR_SSVS_SV, VHAR_Horseshoe_SV, or VHAR_minnesota-part_SV

type include constant term (const) or not (none)

spec Coefficients prior specification

sv log volatility prior specification

intercept Intercept prior specification

init Initial values

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

y0 Y_0

design X_0

y Raw input

If it is SSVS or Horseshoe:

pip Posterior inclusion probabilities.

References

Carriero, A., Chan, J., Clark, T. E., & Marcellino, M. (2022). *Corrigendum to "Large Bayesian vector autoregressions with stochastic volatility and non-conjugate priors" [J. Econometrics 212 (1)(2019) 137-154]*. Journal of Econometrics, 227(2), 506-512.

Chan, J., Koop, G., Poirier, D., & Tobias, J. (2019). *Bayesian Econometric Methods (2nd ed., Econometric Exercises)*. Cambridge: Cambridge University Press.

Cogley, T., & Sargent, T. J. (2005). *Drifts and volatilities: monetary policies and outcomes in the post WWII US*. Review of Economic Dynamics, 8(2), 262-302.

Gruber, L., & Kastner, G. (2022). *Forecasting macroeconomic data with Bayesian VARs: Sparse or dense? It depends!* arXiv.

`bvhar_horseshoe` *Fitting Bayesian VHAR of Horseshoe Prior*

Description

[Deprecated] This function fits VHAR with horseshoe prior.

Usage

```
bvhar_horseshoe(
  y,
  har = c(5, 22),
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_horseshoe(),
  include_mean = TRUE,
  minnesota = c("no", "short", "longrun"),
  algo = c("block", "gibbs"),
```

```
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvharhs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharhs'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
har	Numeric vector for weekly and monthly order. By default, `c(5, 22)`.
num_chains	Number of MCMC chains
num_iter	MCMC iteration number
num_burn	Number of burn-in (warm-up). Half of the iteration is the default choice.
thinning	Thinning every thinning-th iteration
bayes_spec	Horseshoe initialization specification by `set_horseshoe()`.
include_mean	Add constant term (Default: `TRUE`) or not (`FALSE`)
minnesota	Minnesota type
algo	Ordinary gibbs sampling (`gibbs`) or blocked gibbs (Default: `block`).
verbose	Print the progress bar in the console. By default, `FALSE`.
num_thread	**[Experimental]** Number of threads
x	`bvharhs` object
digits	digit option to print
...	not used

Value

`bvhar_horseshoe` returns an object named `bvarhs` class. It is a list with the following components:

coefficients Posterior mean of VHAR coefficients.

covmat Posterior mean of covariance matrix

psi_posterior Posterior mean of precision matrix Ψ

param posterior::draws_df with every variable: alpha, lambda, tau, omega, and eta

param_names Name of every parameter.

df Numer of Coefficients: `3m + 1` or `3m`

p 3 (The number of terms. It contains this element for usage in other functions.)

week Order for weekly term

month Order for monthly term

m Dimension of the data

obs Sample size used when training = `totobs` - `p`

totobs Total number of the observation

call Matched call

process Description of the model, e.g. `VHAR_Horseshoe`

type include constant term (`const`) or not (`none`)

algo Usual Gibbs sampling (`gibbs`) or fast sampling (`fast`)

spec Horseshoe specification defined by `set_horseshoe()`

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

group Indicators for group.

num_group Number of groups.

HARtrans VHAR linear transformation matrix

y0 Y_0

design X_0

y Raw input

References

Kim, Y. G., and Baek, C. (n.d.). Working paper.

`bvhar_minnesota` *Fitting Bayesian VHAR of Minnesota Prior*

Description

This function fits BVHAR with Minnesota prior.

Usage

```
bvhar_minnesota(
  y,
  har = c(5, 22),
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvhar(),
  scale_variance = 0.05,
  include_mean = TRUE,
  parallel = list(),
```

```
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvharmn'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharhm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

## S3 method for class 'bvharmn'
AIC(object, ...)

## S3 method for class 'bvharmn'
BIC(object, ...)

is.bvharmn(x)

## S3 method for class 'bvharmn'
knit_print(x, ...)

## S3 method for class 'bvharhm'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
har	Numeric vector for weekly and monthly order. By default, c(5, 22).
num_chains	Number of MCMC chains
num_iter	MCMC iteration number
num_burn	Number of burn-in (warm-up). Half of the iteration is the default choice.
thinning	Thinning every thinning-th iteration
bayes_spec	A BVHAR model specification by set_bvhar() (default) or set_weight_bvhar().
scale_variance	Proposal distribution scaling constant to adjust an acceptance rate
include_mean	Add constant term (Default: TRUE) or not (FALSE)
parallel	List the same argument of optimParallel::optimParallel(). By default, this is empty, and the function does not execute parallel computation.
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvharhm object
digits	digit option to print
...	not used
object	A bvharmn object

Details

Apply Minnesota prior to Vector HAR: Φ (VHAR matrices) and Σ_e (residual covariance).

$$\Phi \mid \Sigma_e \sim MN(M_0, \Omega_0, \Sigma_e)$$
$$\Sigma_e \sim IW(\Psi_0, \nu_0)$$

(MN: matrix normal, IW: inverse-wishart)

There are two types of Minnesota priors for BVHAR:

- VAR-type Minnesota prior specified by `set_bvhar()`, so-called BVHAR-S model.
- VHAR-type Minnesota prior specified by `set_weight_bvhar()`, so-called BVHAR-L model.

Value

`bvhar_minnesota()` returns an object `bvharmn` class. It is a list with the following components:

coefficients Posterior Mean

fitted.values Fitted values

residuals Residuals

mn_mean Posterior mean matrix of Matrix Normal distribution

mn_prec Posterior precision matrix of Matrix Normal distribution

iw_scale Posterior scale matrix of posterior inverse-wishart distribution

iw_shape Posterior shape of inverse-Wishart distribution (ν_0 - obs + 2). ν_0: nrow(Dummy observation) - k

df Numer of Coefficients: 3m + 1 or 3m

m Dimension of the time series

obs Sample size used when training = `totobs` - 22

prior_mean Prior mean matrix of Matrix Normal distribution: M_0

prior_precision Prior precision matrix of Matrix Normal distribution: Ω_0^{-1}

prior_scale Prior scale matrix of inverse-Wishart distribution: Ψ_0

prior_shape Prior shape of inverse-Wishart distribution: ν_0

y0 Y_0

design X_0

p 3, this element exists to run the other functions

week Order for weekly term

month Order for monthly term

totobs Total number of the observation

type include constant term (`const`) or not (`none`)

HARtrans VHAR linear transformation matrix: C_{HAR}

y Raw input (`matrix`)

call Matched call

process Process string in the bayes_spec: `BVHAR_MN_VAR` (BVHAR-S) or `BVHAR_MN_VHAR` (BVHAR-L)

spec Model specification (bvharspec)

It is also `normaliw` and `bvharmod` class.

References

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

See Also

- `set_bvhar()` to specify the hyperparameters of BVHAR-S
- `set_weight_bvhar()` to specify the hyperparameters of BVHAR-L
- `summary.normaliw()` to summarize BVHAR model

Examples

```
# Perform the function using etf_vix dataset
fit <- bvhar_minnesota(y = etf_vix[,1:3])
class(fit)

# Extract coef, fitted values, and residuals
coef(fit)
head(residuals(fit))
head(fitted(fit))
```

`bvhar_ssvs` *Fitting Bayesian VHAR of SSVS Prior*

Description

[Deprecated] This function fits BVAR(p) with stochastic search variable selection (SSVS) prior.

Usage

```
bvhar_ssvs(
  y,
  har = c(5, 22),
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = choose_ssvs(y = y, ord = har, type = "VHAR", param = c(0.1, 10),
   include_mean = include_mean, gamma_param = c(0.01, 0.01), mean_non = 0, sd_non = 0.1),
  init_spec = init_ssvs(type = "auto"),
  include_mean = TRUE,
```

```
  minnesota = c("no", "short", "longrun"),
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvharssvs'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharssvs'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`har`	Numeric vector for weekly and monthly order. By default, `c(5, 22)`.
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of warm-up (burn-in). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A SSVS model specification by `set_ssvs()`. By default, use a default semiautomatic approach `choose_ssvs()`.
`init_spec`	SSVS initialization specification by `init_ssvs()`. By default, use OLS for coefficient and cholesky factor while 1 for dummies.
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`minnesota`	Apply cross-variable shrinkage structure (Minnesota-way). Two type: `short` type and `longrun` type. By default, `no`.
`verbose`	Print the progress bar in the console. By default, `FALSE`.
`num_thread`	**[Experimental]** Number of threads
`x`	`bvharssvs` object
`digits`	digit option to print
`...`	not used

Details

SSVS prior gives prior to parameters $\alpha = vec(A)$ (VAR coefficient) and $\Sigma_e^{-1} = \Psi \Psi^T$ (residual covariance).

$$\alpha_j \mid \gamma_j \sim (1 - \gamma_j) N(0, \kappa_{0j}^2) + \gamma_j N(0, \kappa_{1j}^2)$$

$$\gamma_j \sim Bernoulli(q_j)$$

and for upper triangular matrix Ψ,

$$\psi_{jj}^2 \sim Gamma(shape = a_j, rate = b_j)$$

$$\psi_{ij} \mid w_{ij} \sim (1 - w_{ij}) N(0, \kappa_{0,ij}^2) + w_{ij} N(0, \kappa_{1,ij}^2)$$
$$w_{ij} \sim Bernoulli(q_{ij})$$

Gibbs sampler is used for the estimation.

Value

`bvhar_ssvs` returns an object named `bvharssvs` class. It is a list with the following components:

coefficients Posterior mean of VAR coefficients.

chol_posterior Posterior mean of cholesky factor matrix

covmat Posterior mean of covariance matrix

omega_posterior Posterior mean of omega

pip Posterior inclusion probability

param posterior::draws_df with every variable: alpha, eta, psi, omega, and gamma

param_names Name of every parameter.

df Numer of Coefficients: `3m + 1` or `3m`

p 3 (The number of terms. It contains this element for usage in other functions.)

week Order for weekly term

month Order for monthly term

m Dimension of the data

obs Sample size used when training = `totobs` - `p`

totobs Total number of the observation

call Matched call

process Description of the model, e.g. `VHAR_SSVS`

type include constant term (`const`) or not (`none`)

spec SSVS specification defined by `set_ssvs()`

init Initial specification defined by `init_ssvs()`

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

group Indicators for group.

num_group Number of groups.

HARtrans VHAR linear transformation matrix

y0 Y_0

design X_0

y Raw input

References

Kim, Y. G., and Baek, C. (n.d.). Working paper.

bvhar_sv *Fitting Bayesian VHAR-SV*

Description

[Deprecated] This function fits VHAR-SV. It can have Minnesota, SSVS, and Horseshoe prior. This function is deprecated. Use `vhar_bayes()` with `cov_spec = set_sv()` option.

Usage

```
bvhar_sv(
  y,
  har = c(5, 22),
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvhar(),
  sv_spec = set_sv(),
  intercept = set_intercept(),
  include_mean = TRUE,
  minnesota = c("longrun", "short", "no"),
  save_init = FALSE,
  convergence = NULL,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvharsv'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharsv'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`har`	Numeric vector for weekly and monthly order. By default, `c(5, 22)`.
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A BVHAR model specification by `set_bvhar()` (default) `set_weight_bvhar()`, `set_ssvs()`, or `set_horseshoe()`.
`sv_spec`	**[Experimental]** SV specification by `set_sv()`.

intercept	**[Experimental]** Prior for the constant term by set_intercept().
include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Apply cross-variable shrinkage structure (Minnesota-way). Two type: short type and longrun (default) type. You can also set no.
save_init	Save every record starting from the initial values (TRUE). By default, exclude the initial values in the record (FALSE), even when num_burn = 0 and thinning = 1. If num_burn > 0 or thinning != 1, this option is ignored.
convergence	Convergence threshold for rhat < convergence. By default, NULL which means no warning.
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvarsv object
digits	digit option to print
...	not used

Details

Cholesky stochastic volatility modeling for VHAR based on

$$\Sigma_t = L^T D_t^{-1} L$$

Value

bvhar_sv() returns an object named bvharsv class. It is a list with the following components:

coefficients Posterior mean of coefficients.

chol_posterior Posterior mean of contemporaneous effects.

param Every set of MCMC trace.

param_names Name of every parameter.

group Indicators for group.

num_group Number of groups.

df Numer of Coefficients: 3m + 1 or 3m

p 3 (The number of terms. It contains this element for usage in other functions.)

week Order for weekly term

month Order for monthly term

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VHAR_SSVS_SV, VHAR_Horseshoe_SV, or VHAR_minnesota-part_SV

type include constant term (const) or not (none)

spec Coefficients prior specification

sv log volatility prior specification

init Initial values

intercept Intercept prior specification

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

HARtrans VHAR linear transformation matrix

y0 Y_0

design X_0

y Raw input

If it is SSVS or Horseshoe:

pip Posterior inclusion probabilities.

References

Kim, Y. G., and Baek, C. (n.d.). Working paper.

`choose_bayes` *Finding the Set of Hyperparameters of Bayesian Model*

Description

[Experimental] This function chooses the set of hyperparameters of Bayesian model using `stats::optim()` function.

Usage

```
choose_bayes(
  bayes_bound = bound_bvhar(),
  ...,
  eps = 1e-04,
  y,
  order = c(5, 22),
  include_mean = TRUE,
  parallel = list()
)
```

Arguments

bayes_bound	Empirical Bayes optimization bound specification defined by bound_bvhar().
...	Additional arguments for stats::optim().
eps	Hyperparameter eps is fixed. By default, 1e-04.
y	Time series data
order	Order for BVAR or BVHAR. p of bvar_minnesota() or har of bvhar_minnesota(). By default, c(5, 22) for har.
include_mean	Add constant term (Default: TRUE) or not (FALSE)
parallel	List the same argument of optimParallel::optimParallel(). By default, this is empty, and the function does not execute parallel computation.

Value

bvharemp class is a list that has

... Many components of stats::optim() or optimParallel::optimParallel()

spec Corresponding bvharspec

fit Chosen Bayesian model

ml Marginal likelihood of the final model

References

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). *Prior Selection for Vector Autoregressions*. Review of Economics and Statistics, 97(2).

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

See Also

- bound_bvhar() to define L-BFGS-B optimization bounds.
- Individual functions: choose_bvar()

---

choose_bvar *Finding the Set of Hyperparameters of Individual Bayesian Model*

Description

Instead of these functions, you can use choose_bayes().

Usage

```
choose_bvar(
  bayes_spec = set_bvar(),
  lower = 0.01,
  upper = 10,
  ...,
  eps = 1e-04,
  y,
  p,
  include_mean = TRUE,
  parallel = list()
)

choose_bvhar(
  bayes_spec = set_bvhar(),
  lower = 0.01,
  upper = 10,
  ...,
  eps = 1e-04,
  y,
  har = c(5, 22),
  include_mean = TRUE,
  parallel = list()
)

## S3 method for class 'bvharemp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.bvharemp(x)

## S3 method for class 'bvharemp'
knit_print(x, ...)
```

Arguments

bayes_spec	Initial Bayes model specification.
lower	**[Experimental]** Lower bound. By default, `.01`.
upper	**[Experimental]** Upper bound. By default, `10`.
...	not used
eps	Hyperparameter eps is fixed. By default, `1e-04`.
y	Time series data
p	BVAR lag
include_mean	Add constant term (Default: `TRUE`) or not (`FALSE`)
parallel	List the same argument of `optimParallel::optimParallel()`. By default, this is empty, and the function does not execute parallel computation.
har	Numeric vector for weekly and monthly order. By default, `c(5, 22)`.

`x`	`bvharemp` object
`digits`	digit option to print

Details

Empirical Bayes method maximizes marginal likelihood and selects the set of hyperparameters. These functions implement `L-BFGS-B` method of `stats::optim()` to find the maximum of marginal likelihood.

If you want to set `lower` and `upper` option more carefully, deal with them like as in `stats::optim()` in order of `set_bvar()`, `set_bvhar()`, or `set_weight_bvhar()`'s argument (except `eps`). In other words, just arrange them in a vector.

Value

`bvharemp` class is a list that has

- `stats::optim()` or `optimParallel::optimParallel()`
- chosen `bvharspec` set
- Bayesian model fit result with chosen specification

  ... Many components of `stats::optim()` or `optimParallel::optimParallel()`

  spec Corresponding `bvharspec`

  fit Chosen Bayesian model

  ml Marginal likelihood of the final model

References

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). *A limited memory algorithm for bound constrained optimization*. SIAM Journal on scientific computing, 16(5), 1190-1208.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). *Prior Selection for Vector Autoregressions*. Review of Economics and Statistics, 97(2).

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

`choose_ssvs` *Choose the Hyperparameters Set of SSVS-VAR using a Default Semi-automatic Approach*

Description

[Deprecated] This function chooses (τ_{0i}, τ_{1i}) and $(\kappa_{0i}, \kappa_{1i})$ using a default semiautomatic approach.

Usage

```
choose_ssvs(
  y,
  ord,
  type = c("VAR", "VHAR"),
  param = c(0.1, 10),
  include_mean = TRUE,
  gamma_param = c(0.01, 0.01),
  mean_non = 0,
  sd_non = 0.1
)
```

Arguments

`y`	Time series data of which columns indicate the variables.
`ord`	Order for VAR or VHAR.
`type`	Model type (Default: `VAR` or `VHAR`).
`param`	Preselected constants $c_0 << c_1$. By default, `0.1` and `10` (See Details).
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`).
`gamma_param`	Parameters (shape, rate) for Gamma distribution. This is for the output.
`mean_non`	Prior mean of unrestricted coefficients. This is for the output.
`sd_non`	Standard deviance of unrestricted coefficients. This is for the output.

Details

Instead of using subjective values of (τ_{0i}, τ_{1i}), we can use

$$\tau_{ki} = c_k V A \hat{R(OLS)}$$

It must be $c_0 << c_1$.

In case of $(\omega_{0ij}, \omega_{1ij})$,

$$\omega_{kij} = c_k = V A \hat{R(OLS)}$$

similarly.

Value

`ssvsinput` object

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

choose_var — *Choose the Best VAR based on Information Criteria*

Description

This function computes AIC, FPE, BIC, and HQ up to p = `lag_max` of VAR model.

Usage

```
choose_var(y, lag_max = 5, include_mean = TRUE, parallel = FALSE)
```

Arguments

`y`	Time series data of which columns indicate the variables
`lag_max`	Maximum Var lag to explore (default = 5)
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`parallel`	Parallel computation using `foreach::foreach()`? By default, `FALSE`.

Value

Minimum order and information criteria values

coef — *Coefficient Matrix of Multivariate Time Series Models*

Description

By defining `stats::coef()` for each model, this function returns coefficient matrix estimates.

Usage

```
## S3 method for class 'varlse'
coef(object, ...)

## S3 method for class 'vharlse'
coef(object, ...)

## S3 method for class 'bvarmn'
coef(object, ...)

## S3 method for class 'bvarflat'
coef(object, ...)

## S3 method for class 'bvharmn'
coef(object, ...)
```

```
## S3 method for class 'bvharsp'
coef(object, ...)

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

Arguments

`object`	Model object
`...`	not used

Value

matrix object with appropriate dimension.

`compute_dic` *Deviance Information Criterion of Multivariate Time Series Model*

Description

Compute DIC of BVAR and BVHAR.

Usage

```
compute_dic(object, ...)

## S3 method for class 'bvarmn'
compute_dic(object, n_iter = 100L, ...)
```

Arguments

`object`	Model fit
`...`	not used
`n_iter`	Number to sample

Details

Deviance information criteria (DIC) is

$$-2 \log p(y \mid \hat{\theta}_{bayes}) + 2 p_{DIC}$$

where p_{DIC} is the effective number of parameters defined by

$$p_{DIC} = 2 (\log p(y \mid \hat{\theta}_{bayes}) - E_{post} \log p(y \mid \theta))$$

Random sampling from posterior distribution gives its computation, $\theta_i \sim \theta \mid y, i = 1, \ldots, M$

$$p_{DIC}^{computed} = 2(\log p(y \mid \hat{\theta}_{bayes}) - \frac{1}{M} \sum_i \log p(y \mid \theta_i))$$

Value

DIC value.

References

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Spiegelhalter, D.J., Best, N.G., Carlin, B.P. and Van Der Linde, A. (2002). *Bayesian measures of model complexity and fit*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64: 583-639.

`compute_logml` *Extracting Log of Marginal Likelihood*

Description

Compute log of marginal likelihood of Bayesian Fit

Usage

```
compute_logml(object, ...)

## S3 method for class 'bvarmn'
compute_logml(object, ...)

## S3 method for class 'bvharmn'
compute_logml(object, ...)
```

Arguments

`object`	Model fit
`...`	not used

Details

Closed form of Marginal Likelihood of BVAR can be derived by

$$p(Y_0) = \pi^{-mn/2} \frac{\Gamma_m((\alpha_0 + n)/2)}{\Gamma_m(\alpha_0/2)} \det(\Omega_0)^{-m/2} \det(S_0)^{\alpha_0/2} \det(\hat{V})^{-m/2} \det(\hat{\Sigma}_e)^{-(\alpha_0+n)/2}$$

Closed form of Marginal Likelihood of BVHAR can be derived by

$$p(Y_0) = \pi^{-ms_0/2} \frac{\Gamma_m((d_0 + n)/2)}{\Gamma_m(d_0/2)} \det(P_0)^{-m/2} \det(U_0)^{d_0/2} \det(\hat{V}_{HAR})^{-m/2} \det(\hat{\Sigma}_e)^{-(d_0+n)/2}$$

Value

log likelihood of Minnesota prior model.

References

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). *Prior Selection for Vector Autoregressions*. Review of Economics and Statistics, 97(2).

`confusion` *Evaluate the Sparsity Estimation Based on Confusion Matrix*

Description

This function computes FDR (false discovery rate) and FNR (false negative rate) for sparse element of the true coefficients given threshold.

Usage

```
confusion(x, y, ...)

## S3 method for class 'summary.bvharsp'
confusion(x, y, truth_thr = 0, ...)
```

Arguments

`x`	`summary.bvharsp` object.
`y`	True inclusion variable.
`...`	not used
`truth_thr`	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

When using this function, the true coefficient matrix Φ should be sparse.

In this confusion matrix, positive (0) means sparsity. FP is false positive, and TP is true positive. FN is false negative, and FN is false negative.

Value

Confusion table as following.

True-estimate	Positive (0)	Negative (1)
Positive (0)	TP	FN
Negative (1)	FP	TN

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

`conf_fdr` *Evaluate the Sparsity Estimation Based on FDR*

Description

This function computes false discovery rate (FDR) for sparse element of the true coefficients given threshold.

Usage

```
conf_fdr(x, y, ...)

## S3 method for class 'summary.bvharsp'
conf_fdr(x, y, truth_thr = 0, ...)
```

Arguments

`x`	`summary.bvharsp` object.
`y`	True inclusion variable.
`...`	not used
`truth_thr`	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

When using this function, the true coefficient matrix Φ should be sparse. False discovery rate (FDR) is computed by

$$FDR = \frac{FP}{TP + FP}$$

where TP is true positive, and FP is false positive.

Value

FDR value in confusion table

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

See Also

`confusion()`

conf_fnr *Evaluate the Sparsity Estimation Based on FNR*

Description

This function computes false negative rate (FNR) for sparse element of the true coefficients given threshold.

Usage

```
conf_fnr(x, y, ...)

## S3 method for class 'summary.bvharsp'
conf_fnr(x, y, truth_thr = 0, ...)
```

Arguments

x	`summary.bvharsp` object.
y	True inclusion variable.
...	not used
truth_thr	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

False negative rate (FNR) is computed by

$$FNR = \frac{FN}{TP + FN}$$

where TP is true positive, and FN is false negative.

Value

FNR value in confusion table

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

See Also

`confusion()`

conf_fscore *Evaluate the Sparsity Estimation Based on F1 Score*

Description

This function computes F1 score for sparse element of the true coefficients given threshold.

Usage

```
conf_fscore(x, y, ...)

## S3 method for class 'summary.bvharsp'
conf_fscore(x, y, truth_thr = 0, ...)
```

Arguments

x	`summary.bvharsp` object.
y	True inclusion variable.
...	not used
truth_thr	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

The F1 score is computed by

$$F_1 = \frac{2 precision \times recall}{precision + recall}$$

Value

F1 score in confusion table

See Also

`confusion()`

conf_prec — *Evaluate the Sparsity Estimation Based on Precision*

Description

This function computes precision for sparse element of the true coefficients given threshold.

Usage

```
conf_prec(x, y, ...)

## S3 method for class 'summary.bvharsp'
conf_prec(x, y, truth_thr = 0, ...)
```

Arguments

`x`	`summary.bvharsp` object.
`y`	True inclusion variable.
`...`	not used
`truth_thr`	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

If the element of the estimate $\hat{\Phi}$ is smaller than some threshold, it is treated to be zero. Then the precision is computed by

$$precision = \frac{TP}{TP + FP}$$

where TP is true positive, and FP is false positive.

Value

Precision value in confusion table

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

See Also

`confusion()`

`conf_recall` *Evaluate the Sparsity Estimation Based on Recall*

Description

This function computes recall for sparse element of the true coefficients given threshold.

Usage

```
conf_recall(x, y, ...)

## S3 method for class 'summary.bvharsp'
conf_recall(x, y, truth_thr = 0L, ...)
```

Arguments

`x`	`summary.bvharsp` object.
`y`	True inclusion variable.
`...`	not used
`truth_thr`	Threshold value when using non-sparse true coefficient matrix. By default, `0` for sparse matrix.

Details

Precision is computed by

$$recall = \frac{TP}{TP + FN}$$

where TP is true positive, and FN is false negative.

Value

Recall value in confusion table

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

See Also

`confusion()`

divide_ts *Split a Time Series Dataset into Train-Test Set*

Description

Split a given time series dataset into train and test set for evaluation.

Usage

```
divide_ts(y, n_ahead)
```

Arguments

y	Time series data of which columns indicate the variables
n_ahead	step to evaluate

Value

List of two datasets, train and test.

dynamic_spillover *Dynamic Spillover*

Description

This function gives connectedness table with h-step ahead normalized spillover index (a.k.a. variance shares).

Usage

```
dynamic_spillover(object, n_ahead = 10L, ...)

## S3 method for class 'bvhardynsp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvhardynsp'
knit_print(x, ...)

## S3 method for class 'olsmod'
dynamic_spillover(object, n_ahead = 10L, window, num_thread = 1, ...)

## S3 method for class 'normaliw'
dynamic_spillover(
  object,
  n_ahead = 10L,
```

```
  window,
  num_iter = 1000L,
  num_burn = floor(num_iter/2),
  thinning = 1,
  num_thread = 1,
  ...
)

## S3 method for class 'ldltmod'
dynamic_spillover(
  object,
  n_ahead = 10L,
  window,
  sparse = FALSE,
  num_thread = 1,
  ...
)

## S3 method for class 'svmod'
dynamic_spillover(object, n_ahead = 10L, sparse = FALSE, num_thread = 1, ...)
```

Arguments

`object`	Model object
`n_ahead`	step to forecast. By default, 10.
`...`	not used
`x`	`bvhardynsp` object
`digits`	digit option to print
`window`	Window size
`num_thread`	**[Experimental]** Number of threads
`num_iter`	Number to sample MNIW distribution
`num_burn`	Number of burn-in
`thinning`	Thinning every thinning-th iteration
`sparse`	**[Experimental]** Apply restriction. By default, `FALSE`.

References

Diebold, F. X., & Yilmaz, K. (2012). *Better to give than to receive: Predictive directional measurement of volatility spillovers*. International Journal of forecasting, 28(1), 57-66.

`etf_vix` *CBOE ETF Volatility Index Dataset*

Description

Chicago Board Options Exchange (CBOE) Exchange Traded Funds (ETFs) volatility index from FRED.

Usage

```
etf_vix
```

Format

A data frame of 1006 row and 9 columns:

From 2012-01-09 to 2015-06-27, 33 missing observations were interpolated by `stats::approx()` with `linear`.

GVZCLS Gold ETF volatility index

VXFXICLS China ETF volatility index

OVXCLS Crude Oil ETF volatility index

VXEEMCLS Emerging Markets ETF volatility index

EVZCLS EuroCurrency ETF volatility index

VXSLVCLS Silver ETF volatility index

VXGDXCLS Gold Miners ETF volatility index

VXXLECLS Energy Sector ETF volatility index

VXEWZCLS Brazil ETF volatility index

Details

Copyright, 2016, Chicago Board Options Exchange, Inc.

Note that, in this data frame, dates column is removed. This dataset interpolated 36 missing observations (nontrading dates) using `imputeTS::na_interpolation()`.

Source

Source: `https://www.cboe.com`

Release: `https://www.cboe.com/us/options/market_statistics/daily/`

References

Chicago Board Options Exchange, CBOE Gold ETF Volatility Index (GVZCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/GVZCLS, July 31, 2021.

Chicago Board Options Exchange, CBOE China ETF Volatility Index (VXFXICLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXFXICLS, August 1, 2021.

Chicago Board Options Exchange, CBOE Crude Oil ETF Volatility Index (OVXCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/OVXCLS, August 1, 2021.

Chicago Board Options Exchange, CBOE Emerging Markets ETF Volatility Index (VXEEMCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXEEMCLS, August 1, 2021.

Chicago Board Options Exchange, CBOE EuroCurrency ETF Volatility Index (EVZCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/EVZCLS, August 2, 2021.

Chicago Board Options Exchange, CBOE Silver ETF Volatility Index (VXSLVCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXSLVCLS, August 1, 2021.

Chicago Board Options Exchange, CBOE Gold Miners ETF Volatility Index (VXGDXCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXGDXCLS, August 1, 2021.

Chicago Board Options Exchange, CBOE Energy Sector ETF Volatility Index (VXXLECLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXXLECLS, August 1, 2021.

Chicago Board Options Exchange, CBOE Brazil ETF Volatility Index (VXEWZCLS), retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/VXEWZCLS, August 2, 2021.

---

fitted — *Fitted Matrix from Multivariate Time Series Models*

Description

By defining `stats::fitted()` for each model, this function returns fitted matrix.

Usage

```
## S3 method for class 'varlse'
fitted(object, ...)

## S3 method for class 'vharlse'
fitted(object, ...)
```

```
## S3 method for class 'bvarmn'
fitted(object, ...)

## S3 method for class 'bvarflat'
fitted(object, ...)

## S3 method for class 'bvharmn'
fitted(object, ...)
```

Arguments

object	Model object
...	not used

Value

matrix object.

---

forecast_expand — *Out-of-sample Forecasting based on Expanding Window*

Description

This function conducts expanding window forecasting.

Usage

```
forecast_expand(object, n_ahead, y_test, num_thread = 1, ...)

## S3 method for class 'olsmod'
forecast_expand(object, n_ahead, y_test, num_thread = 1, ...)

## S3 method for class 'normaliw'
forecast_expand(object, n_ahead, y_test, num_thread = 1, use_fit = TRUE, ...)

## S3 method for class 'ldltmod'
forecast_expand(
  object,
  n_ahead,
  y_test,
  num_thread = 1,
  level = 0.05,
  sparse = FALSE,
  lpl = FALSE,
  use_fit = TRUE,
  ...
)
```

```
## S3 method for class 'svmod'
forecast_expand(
  object,
  n_ahead,
  y_test,
  num_thread = 1,
  level = 0.05,
  use_sv = TRUE,
  sparse = FALSE,
  lpl = FALSE,
  use_fit = TRUE,
  ...
)
```

Arguments

object	Model object
n_ahead	Step to forecast in rolling window scheme
y_test	Test data to be compared. Use divide_ts() if you don't have separate evaluation dataset.
num_thread	**[Experimental]** Number of threads
...	Additional arguments.
use_fit	**[Experimental]** Use object result for the first window. By default, TRUE.
level	Specify alpha of confidence interval level 100(1 - alpha) percentage. By default, .05.
sparse	**[Experimental]** Apply restriction. By default, FALSE.
lpl	**[Experimental]** Compute log-predictive likelihood (LPL). By default, FALSE.
use_sv	Use SV term

Details

Expanding windows forecasting fixes the starting period. It moves the window ahead and forecast h-ahead in y_test set.

Value

predbvhar_expand class

References

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTEXTS. https://otexts.com/fpp3/

forecast_roll *Out-of-sample Forecasting based on Rolling Window*

Description

This function conducts rolling window forecasting.

Usage

```
forecast_roll(object, n_ahead, y_test, num_thread = 1, ...)

## S3 method for class 'bvharcv'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.bvharcv(x)

## S3 method for class 'bvharcv'
knit_print(x, ...)

## S3 method for class 'olsmod'
forecast_roll(object, n_ahead, y_test, num_thread = 1, ...)

## S3 method for class 'normaliw'
forecast_roll(object, n_ahead, y_test, num_thread = 1, use_fit = TRUE, ...)

## S3 method for class 'ldltmod'
forecast_roll(
  object,
  n_ahead,
  y_test,
  num_thread = 1,
  level = 0.05,
  sparse = FALSE,
  lpl = FALSE,
  use_fit = TRUE,
  ...
)

## S3 method for class 'svmod'
forecast_roll(
  object,
  n_ahead,
  y_test,
  num_thread = 1,
  level = 0.05,
  use_sv = TRUE,
  sparse = FALSE,
```

```
  lpl = FALSE,
  use_fit = TRUE,
  ...
)
```

Arguments

object	Model object
n_ahead	Step to forecast in rolling window scheme
y_test	Test data to be compared. Use divide_ts() if you don't have separate evaluation dataset.
num_thread	**[Experimental]** Number of threads
...	not used
x	bvharcv object
digits	digit option to print
use_fit	**[Experimental]** Use object result for the first window. By default, TRUE.
level	Specify alpha of confidence interval level 100(1 - alpha) percentage. By default, .05.
sparse	**[Experimental]** Apply restriction. By default, FALSE.
lpl	**[Experimental]** Compute log-predictive likelihood (LPL). By default, FALSE.
use_sv	Use SV term

Details

Rolling windows forecasting fixes window size. It moves the window ahead and forecast h-ahead in y_test set.

Value

predbvhar_roll class

References

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTEXTS.

`FPE` *Final Prediction Error Criterion*

Description

Compute FPE of VAR(p) and VHAR

Usage

```
FPE(object, ...)

## S3 method for class 'varlse'
FPE(object, ...)

## S3 method for class 'vharlse'
FPE(object, ...)
```

Arguments

`object`	Model fit
`...`	not used

Details

Let $\tilde{\Sigma}_e$ be the MLE and let $\hat{\Sigma}_e$ be the unbiased estimator (`covmat`) for Σ_e. Note that

$$\tilde{\Sigma}_e = \frac{n - k}{T} \hat{\Sigma}_e$$

Then

$$FPE(p) = (\frac{n + k}{n - k})^m \det \tilde{\Sigma}_e$$

Value

FPE value.

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

fromse *Evaluate the Estimation Based on Frobenius Norm*

Description

This function computes estimation error given estimated model and true coefficient.

Usage

```
fromse(x, y, ...)

## S3 method for class 'bvharsp'
fromse(x, y, ...)
```

Arguments

`x`	Estimated model.
`y`	Coefficient matrix to be compared.
`...`	not used

Details

Consider the Frobenius Norm $\lVert\cdot\rVert_F$. let $\hat{\Phi}$ be nrow x k the estimates, and let Φ be the true coefficients matrix. Then the function computes estimation error by

$$MSE = 100 \frac{\lVert\hat{\Phi} - \Phi\rVert_F}{nrow \times k}$$

Value

Frobenius norm value

References

Bai, R., & Ghosh, M. (2018). High-dimensional multivariate posterior consistency under global-local shrinkage priors. Journal of Multivariate Analysis, 167, 157-170.

geom_eval *Adding Test Data Layer*

Description

This function adds a layer of test dataset.

Usage

```
geom_eval(data, colour = "red", ...)
```

Arguments

data	Test data to draw, which has the same format with the train data.
colour	Color of the line (By default, red).
...	Other arguments passed on the ggplot2::geom_path().

Value

A ggplot layer

gg_loss *Compare Lists of Models*

Description

Draw plot of test error for given models

Usage

```
gg_loss(
  mod_list,
  y,
  type = c("mse", "mae", "mape", "mase"),
  mean_line = FALSE,
  line_param = list(),
  mean_param = list(),
  viridis = FALSE,
  viridis_option = "D",
  NROW = NULL,
  NCOL = NULL,
  ...
)
```

Arguments

`mod_list`	Lists of forecast results (`predbvhar` objects)
`y`	Test data to be compared. should be the same format with the train data and predict$forecast.
`type`	Loss function to be used (`mse`: MSE, `mae`: MAE, `mape`: MAPE, `mase`: MASE)
`mean_line`	Whether to draw average loss. By default, `FALSE`.
`line_param`	Parameter lists for `ggplot2::geom_path()`.
`mean_param`	Parameter lists for average loss with `ggplot2::geom_hline()`.
`viridis`	If `TRUE`, scale CI and forecast line using `ggplot2::scale_fill_viridis_d()` and ggplot2::scale_colour_viridis_d, respectively.
`viridis_option`	Option for viridis string. See `option` of ggplot2::scale_colour_viridis_d. Choose one of `c("A", "B", "C", "D", "E")`. By default, `D`.
`NROW`	`nrow` of `ggplot2::facet_wrap()`
`NCOL`	`ncol` of `ggplot2::facet_wrap()`
`...`	Additional options for `geom_loss` (`inherit.aes` and `show.legend`)

Value

A ggplot object

See Also

- `mse()` to compute MSE for given forecast result
- `mae()` to compute MAE for given forecast result
- `mape()` to compute MAPE for given forecast result
- `mase()` to compute MASE for given forecast result

HQ *Hannan-Quinn Criterion*

Description

Compute HQ of VAR(p), VHAR, BVAR(p), and BVHAR

Usage

```
HQ(object, ...)

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

## S3 method for class 'varlse'
HQ(object, ...)
```

```
## S3 method for class 'vharlse'
HQ(object, ...)

## S3 method for class 'bvarmn'
HQ(object, ...)

## S3 method for class 'bvarflat'
HQ(object, ...)

## S3 method for class 'bvharmn'
HQ(object, ...)
```

Arguments

`object`	A `logLik` object or Model fit
`...`	not used

Details

The formula is

$$HQ = -2 \log p(y \mid \hat{\theta}) + k \log \log(T)$$

which can be computed by `AIC(object, ..., k = 2 * log(log(nobs(object))))` with `stats::AIC()`. Let $\tilde{\Sigma}_e$ be the MLE and let $\hat{\Sigma}_e$ be the unbiased estimator (`covmat`) for Σ_e. Note that

$$\tilde{\Sigma}_e = \frac{n-k}{T} \hat{\Sigma}_e$$

Then

$$HQ(p) = \log \det \Sigma_e + \frac{2 \log \log n}{n} (\text{number of freely estimated parameters})$$

where the number of freely estimated parameters is pm^2.

Value

HQ value.

References

Hannan, E.J. and Quinn, B.G. (1979). *The Determination of the Order of an Autoregression*. Journal of the Royal Statistical Society: Series B (Methodological), 41: 190-195.

Hannan, E.J. and Quinn, B.G. (1979). *The Determination of the Order of an Autoregression*. Journal of the Royal Statistical Society: Series B (Methodological), 41: 190-195.

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

Quinn, B.G. (1980). *Order Determination for a Multivariate Autoregression*. Journal of the Royal Statistical Society: Series B (Methodological), 42: 182-185.

init_ssvs — *Initial Parameters of Stochastic Search Variable Selection (SSVS) Model*

Description

[Deprecated] Set initial parameters before starting Gibbs sampler for SSVS.

Usage

```
init_ssvs(
  init_coef,
  init_coef_dummy,
  init_chol,
  init_chol_dummy,
  type = c("user", "auto")
)

## S3 method for class 'ssvsinit'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.ssvsinit(x)

## S3 method for class 'ssvsinit'
knit_print(x, ...)
```

Arguments

`init_coef`	Initial coefficient matrix. Initialize with an array or list for multiple chains.
`init_coef_dummy`	Initial indicator matrix (1-0) corresponding to each component of coefficient. Initialize with an array or list for multiple chains.
`init_chol`	Initial cholesky factor (upper triangular). Initialize with an array or list for multiple chains.
`init_chol_dummy`	Initial indicator matrix (1-0) corresponding to each component of cholesky factor. Initialize with an array or list for multiple chains.
`type`	**[Experimental]** Type to choose initial values. One of `user` (User-given) and `auto` (OLS for coefficients and 1 for dummy).
`x`	`ssvsinit`
`digits`	digit option to print
`...`	not used

Details

Set SSVS initialization for the VAR model.

- `init_coef`: (kp + 1) x m A coefficient matrix.
- `init_coef_dummy`: kp x m Γ dummy matrix to restrict the coefficients.
- `init_chol`: k x k Ψ upper triangular cholesky factor, which $\Psi \Psi^\intercal = \Sigma_e^{-1}$.
- `init_chol_dummy`: k x k Ω upper triangular dummy matrix to restrict the cholesky factor.

Denote that `init_chol` and `init_chol_dummy` should be upper_triangular or the function gives error.

For parallel chain initialization, assign three-dimensional array or three-length list.

Value

`ssvsinit` object

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

`irf.varlse` *Impulse Response Analysis*

Description

Computes responses to impulses or orthogonal impulses

Usage

```
## S3 method for class 'varlse'
irf(object, lag_max = 10, orthogonal = TRUE, impulse_var, response_var, ...)

## S3 method for class 'vharlse'
irf(object, lag_max = 10, orthogonal = TRUE, impulse_var, response_var, ...)

## S3 method for class 'bvharirf'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

irf(object, lag_max, orthogonal, impulse_var, response_var, ...)

is.bvharirf(x)
```

```
## S3 method for class 'bvharirf'
knit_print(x, ...)
```

Arguments

`object`	Model object
`lag_max`	Maximum lag to investigate the impulse responses (By default, `10`)
`orthogonal`	Orthogonal impulses (`TRUE`) or just impulses (`FALSE`)
`impulse_var`	Impulse variables character vector. If not specified, use every variable.
`response_var`	Response variables character vector. If not specified, use every variable.
`...`	not used
`x`	`bvharirf` object
`digits`	digit option to print

Value

`bvharirf` class

Responses to forecast errors

If `orthogonal = FALSE`, the function gives W_j VMA representation of the process such that

$$Y_t = \sum_{j=0}^{\infty} W_j \epsilon_{t-j}$$

Responses to orthogonal impulses

If `orthogonal = TRUE`, it gives orthogonalized VMA representation

$$\Theta$$

. Based on variance decomposition (Cholesky decomposition)

$$\Sigma = PP^T$$

where P is lower triangular matrix, impulse response analysis if performed under MA representation

$$y_t = \sum_{i=0}^{\infty} \Theta_i v_{t-i}$$

Here,

$$\Theta_i = W_i P$$

and $v_t = P^{-1} \epsilon_t$ are orthogonal.

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

See Also

VARtoVMA()

VHARtoVMA()

---

is.stable *Stability of the process*

Description

Check the stability condition of coefficient matrix.

Usage

```
is.stable(x, ...)

## S3 method for class 'varlse'
is.stable(x, ...)

## S3 method for class 'vharlse'
is.stable(x, ...)

## S3 method for class 'bvarmn'
is.stable(x, ...)

## S3 method for class 'bvarflat'
is.stable(x, ...)

## S3 method for class 'bvharmn'
is.stable(x, ...)
```

Arguments

x	Model fit
...	not used

Details

VAR(p) is stable if

$$\det(I_m - Az) \neq 0$$

for $|z| \le 1$.

Value

logical class

logical class

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

---

mae *Evaluate the Model Based on MAE (Mean Absolute Error)*

Description

This function computes MAE given prediction result versus evaluation set.

Usage

```
mae(x, y, ...)

## S3 method for class 'predbvhar'
mae(x, y, ...)

## S3 method for class 'bvharcv'
mae(x, y, ...)
```

Arguments

`x`	Forecasting object
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. MAE is defined by

$$MSE = mean(|e_t|)$$

Some researchers prefer MAE to MSE because it is less sensitive to outliers.

Value

MAE vector corressponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

mape *Evaluate the Model Based on MAPE (Mean Absolute Percentage Error)*

Description

This function computes MAPE given prediction result versus evaluation set.

Usage

```
mape(x, y, ...)

## S3 method for class 'predbvhar'
mape(x, y, ...)

## S3 method for class 'bvharcv'
mape(x, y, ...)
```

Arguments

x	Forecasting object
y	Test data to be compared. should be the same format with the train data.
...	not used

Details

Let $e_t = y_t - \hat{y}_t$. Percentage error is defined by $p_t = 100e_t / Y_t$ (100 can be omitted since comparison is the focus).

$$MAPE = mean(|p_t|)$$

Value

MAPE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

mase *Evaluate the Model Based on MASE (Mean Absolute Scaled Error)*

Description

This function computes MASE given prediction result versus evaluation set.

Usage

```
mase(x, y, ...)

## S3 method for class 'predbvhar'
mase(x, y, ...)

## S3 method for class 'bvharcv'
mase(x, y, ...)
```

Arguments

`x`	Forecasting object
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. Scaled error is defined by

$$q_t = \frac{e_t}{\sum_{i=2}^{n} |Y_i - Y_{i-1}| / (n-1)}$$

so that the error can be free of the data scale. Then

$$MASE = mean(|q_t|)$$

Here, Y_i are the points in the sample, i.e. errors are scaled by the in-sample mean absolute error ($mean(|e_t|)$) from the naive random walk forecasting.

Value

MASE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

mrae — *Evaluate the Model Based on MRAE (Mean Relative Absolute Error)*

Description

This function computes MRAE given prediction result versus evaluation set.

Usage

```
mrae(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
mrae(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
mrae(x, pred_bench, y, ...)
```

Arguments

`x`	Forecasting object to use
`pred_bench`	The same forecasting object from benchmark model
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. MRAE implements benchmark model as scaling method. Relative error is defined by

$$r_t = \frac{e_t}{e_t^*}$$

where e_t^* is the error from the benchmark method. Then

$$MRAE = mean(|r_t|)$$

Value

MRAE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

`mse` *Evaluate the Model Based on MSE (Mean Square Error)*

Description

This function computes MSE given prediction result versus evaluation set.

Usage

```
mse(x, y, ...)

## S3 method for class 'predbvhar'
mse(x, y, ...)

## S3 method for class 'bvharcv'
mse(x, y, ...)
```

Arguments

`x`	Forecasting object
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. Then

$$MSE = mean(e_t^2)$$

MSE is the most used accuracy measure.

Value

MSE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

predict *Forecasting Multivariate Time Series*

Description

Forecasts multivariate time series using given model.

Usage

```
## S3 method for class 'varlse'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'vharlse'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'bvarmn'
predict(object, n_ahead, n_iter = 100L, level = 0.05, num_thread = 1, ...)

## S3 method for class 'bvharmn'
predict(object, n_ahead, n_iter = 100L, level = 0.05, num_thread = 1, ...)

## S3 method for class 'bvarflat'
predict(object, n_ahead, n_iter = 100L, level = 0.05, num_thread = 1, ...)

## S3 method for class 'bvarssvs'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'bvharssvs'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'bvarhs'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'bvharhs'
predict(object, n_ahead, level = 0.05, ...)

## S3 method for class 'bvarldlt'
predict(
  object,
  n_ahead,
  level = 0.05,
  num_thread = 1,
  sparse = FALSE,
  warn = FALSE,
  ...
)
```

```
## S3 method for class 'bvharldlt'
predict(
  object,
  n_ahead,
  level = 0.05,
  num_thread = 1,
  sparse = FALSE,
  warn = FALSE,
  ...
)

## S3 method for class 'bvarsv'
predict(
  object,
  n_ahead,
  level = 0.05,
  num_thread = 1,
  use_sv = TRUE,
  sparse = FALSE,
  warn = FALSE,
  ...
)

## S3 method for class 'bvharsv'
predict(
  object,
  n_ahead,
  level = 0.05,
  num_thread = 1,
  use_sv = TRUE,
  sparse = FALSE,
  warn = FALSE,
  ...
)

## S3 method for class 'predbvhar'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.predbvhar(x)

## S3 method for class 'predbvhar'
knit_print(x, ...)
```

Arguments

object	Model object
n_ahead	step to forecast
level	Specify alpha of confidence interval level 100(1 - alpha) percentage. By default,

	.05.
`...`	not used
`n_iter`	Number to sample residual matrix from inverse-wishart distribution. By default, 100.
`num_thread`	Number of threads
`sparse`	**[Experimental]** Apply restriction. By default, `FALSE`. Give CI level (e.g. `.05`) instead of `TRUE` to use credible interval across MCMC for restriction.
`warn`	Give warning for stability of each coefficients record. By default, `FALSE`.
`use_sv`	Use SV term
`x`	`predbvhar` object
`digits`	digit option to print

Value

`predbvhar` class with the following components:

process object$process

forecast forecast matrix

se standard error matrix

lower lower confidence interval

upper upper confidence interval

lower_joint lower CI adjusted (Bonferroni)

upper_joint upper CI adjusted (Bonferroni)

y object$y

n-step ahead forecasting VAR(p)

See pp35 of Lütkepohl (2007). Consider h-step ahead forecasting (e.g. n + 1, ... n + h).
Let $y_{(n)}^T = (y_n^T, ..., y_{n-p+1}^T, 1)$. Then one-step ahead (point) forecasting:

$$\hat{y}_{n+1}^T = y_{(n)}^T \hat{B}$$

Recursively, let $\hat{y}_{(n+1)}^T = (\hat{y}_{n+1}^T, y_n^T, ..., y_{n-p+2}^T, 1)$. Then two-step ahead (point) forecasting:

$$\hat{y}_{n+2}^T = \hat{y}_{(n+1)}^T \hat{B}$$

Similarly, h-step ahead (point) forecasting:

$$\hat{y}_{n+h}^T = \hat{y}_{(n+h-1)}^T \hat{B}$$

How about confident region? Confidence interval at h-period is

$$y_{k,t}(h) \pm z_(\alpha/2)\sigma_k(h)$$

Joint forecast region of $100(1-\alpha)\%$ can be computed by

$$\{(y_{k,1}, y_{k,h}) \mid y_{k,n}(i) - z_{(\alpha/2h)}\sigma_n(i) \le y_{n,i} \le y_{k,n}(i) + z_{(\alpha/2h)}\sigma_k(i), i = 1, \ldots, h\}$$

See the pp41 of Lütkepohl (2007).

To compute covariance matrix, it needs VMA representation:

$$Y_t(h) = c + \sum_{i=h}^{\infty} W_i \epsilon_{t+h-i} = c + \sum_{i=0}^{\infty} W_{h+i} \epsilon_{t-i}$$

Then

$$\Sigma_y(h) = MSE[y_t(h)] = \sum_{i=0}^{h-1} W_i \Sigma_\epsilon W_i^T = \Sigma_y(h-1) + W_{h-1} \Sigma_\epsilon W_{h-1}^T$$

n-step ahead forecasting VHAR

Let T_{HAR} is VHAR linear transformation matrix. Since VHAR is the linearly transformed VAR(22), let $y_{(n)}^T = (y_n^T, y_{n-1}^T, ..., y_{n-21}^T, 1)$.

Then one-step ahead (point) forecasting:

$$\hat{y}_{n+1}^T = y_{(n)}^T T_{HAR} \hat{\Phi}$$

Recursively, let $\hat{y}_{(n+1)}^T = (\hat{y}_{n+1}^T, y_n^T, ..., y_{n-20}^T, 1)$. Then two-step ahead (point) forecasting:

$$\hat{y}_{n+2}^T = \hat{y}_{(n+1)}^T T_{HAR} \hat{\Phi}$$

and h-step ahead (point) forecasting:

$$\hat{y}_{n+h}^T = \hat{y}_{(n+h-1)}^T T_{HAR} \hat{\Phi}$$

n-step ahead forecasting BVAR(p) with minnesota prior

Point forecasts are computed by posterior mean of the parameters. See Section 3 of Bańbura et al. (2010).

Let $\hat{B}$ be the posterior MN mean and let $\hat{V}$ be the posterior MN precision.

Then predictive posterior for each step

$$y_{n+1} \mid \Sigma_e, y \sim N(vec(y_{(n)}^T A), \Sigma_e \otimes (1 + y_{(n)}^T \hat{V}^{-1} y_{(n)}))$$

$$y_{n+2} \mid \Sigma_e, y \sim N(vec(\hat{y}_{(n+1)}^T A), \Sigma_e \otimes (1 + \hat{y}_{(n+1)}^T \hat{V}^{-1} \hat{y}_{(n+1)}))$$

and recursively,

$$y_{n+h} \mid \Sigma_e, y \sim N(vec(\hat{y}_{(n+h-1)}^T A), \Sigma_e \otimes (1 + \hat{y}_{(n+h-1)}^T \hat{V}^{-1} \hat{y}_{(n+h-1)}))$$

n-step ahead forecasting BVHAR

Let $\hat{\Phi}$ be the posterior MN mean and let $\hat{\Psi}$ be the posterior MN precision.

Then predictive posterior for each step

$$y_{n+1} \mid \Sigma_e, y \sim N(vec(y_{(n)}^T \tilde{T}^T \Phi), \Sigma_e \otimes (1 + y_{(n)}^T \tilde{T} \hat{\Psi}^{-1} \tilde{T} y_{(n)}))$$

$$y_{n+2} \mid \Sigma_e, y \sim N(vec(y_{(n+1)}^T \tilde{T}^T \Phi), \Sigma_e \otimes (1 + y_{(n+1)}^T \tilde{T} \hat{\Psi}^{-1} \tilde{T} y_{(n+1)}))$$

and recursively,

$$y_{n+h} \mid \Sigma_e, y \sim N(vec(y_{(n+h-1)}^T \tilde{T}^T \Phi), \Sigma_e \otimes (1 + y_{(n+h-1)}^T \tilde{T} \hat{\Psi}^{-1} \tilde{T} y_{(n+h-1)}))$$

n-step ahead forecasting VAR(p) with SSVS and Horseshoe

The process of the computing point estimate is the same. However, predictive interval is achieved from each Gibbs sampler sample.

$$y_{n+1} \mid A, \Sigma_e, y \sim N(vec(y_{(n)}^T A), \Sigma_e)$$

$$y_{n+h} \mid A, \Sigma_e, y \sim N(vec(\hat{y}_{(n+h-1)}^T A), \Sigma_e)$$

n-step ahead forecasting VHAR with SSVS and Horseshoe

The process of the computing point estimate is the same. However, predictive interval is achieved from each Gibbs sampler sample.

$$y_{n+1} \mid \Sigma_e, y \sim N(vec(y_{(n)}^T \tilde{T}^T \Phi), \Sigma_e \otimes (1 + y_{(n)}^T \tilde{T} \hat{\Psi}^{-1} \tilde{T} y_{(n)}))$$

$$y_{n+h} \mid \Sigma_e, y \sim N(vec(y_{(n+h-1)}^T \tilde{T}^T \Phi), \Sigma_e \otimes (1 + y_{(n+h-1)}^T \tilde{T} \hat{\Psi}^{-1} \tilde{T} y_{(n+h-1)}))$$

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

Corsi, F. (2008). *A Simple Approximate Long-Memory Model of Realized Volatility*. Journal of Financial Econometrics, 7(2), 174-196.

Baek, C. and Park, M. (2021). *Sparse vector heterogeneous autoregressive modeling for realized volatility*. J. Korean Stat. Soc. 50, 495-510.

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Karlsson, S. (2013). *Chapter 15 Forecasting with Bayesian Vector Autoregression*. Handbook of Economic Forecasting, 2, 791-897.

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Korobilis, D. (2013). *VAR FORECASTING USING BAYESIAN VARIABLE SELECTION*. Journal of Applied Econometrics, 28(2).

Korobilis, D. (2013). *VAR FORECASTING USING BAYESIAN VARIABLE SELECTION*. Journal of Applied Econometrics, 28(2).

Huber, F., Koop, G., & Onorante, L. (2021). *Inducing Sparsity and Shrinkage in Time-Varying Parameter Models*. Journal of Business & Economic Statistics, 39(3), 669-683.

`print.summary.bvharsp` *Summarizing BVAR and BVHAR with Shrinkage Priors*

Description

Conduct variable selection.

Usage

```
## S3 method for class 'summary.bvharsp'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.bvharsp'
knit_print(x, ...)

## S3 method for class 'ssvsmod'
summary(object, method = c("pip", "ci"), threshold = 0.5, level = 0.05, ...)

## S3 method for class 'hsmod'
summary(object, method = c("pip", "ci"), threshold = 0.5, level = 0.05, ...)

## S3 method for class 'ngmod'
summary(object, level = 0.05, ...)
```

Arguments

`x`	`summary.bvharsp` object
`digits`	digit option to print
`...`	not used
`object`	Model fit
`method`	Use PIP (`pip`) or credible interval (`ci`).
`threshold`	Threshold for posterior inclusion probability
`level`	Specify alpha of credible interval level 100(1 - alpha) percentage. By default, `.05`.

Value

summary.ssvsmod object

hsmod object

ngmod object

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

O'Hara, R. B., & Sillanpää, M. J. (2009). *A review of Bayesian variable selection methods: what, how and which*. Bayesian Analysis, 4(1), 85-117.

---

relmae *Evaluate the Model Based on RelMAE (Relative MAE)*

Description

This function computes RelMAE given prediction result versus evaluation set.

Usage

```
relmae(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
relmae(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
relmae(x, pred_bench, y, ...)
```

Arguments

x	Forecasting object to use
pred_bench	The same forecasting object from benchmark model
y	Test data to be compared. should be the same format with the train data.
...	not used

Details

Let $e_t = y_t - \hat{y}_t$. RelMAE implements MAE of benchmark model as relative measures. Let MAE_b be the MAE of the benchmark model. Then

$$RelMAE = \frac{MAE}{MAE_b}$$

where MAE is the MAE of our model.

Value

RelMAE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

---

relspne — *Evaluate the Estimation Based on Relative Spectral Norm Error*

Description

This function computes relative estimation error given estimated model and true coefficient.

Usage

```
relspne(x, y, ...)

## S3 method for class 'bvharsp'
relspne(x, y, ...)
```

Arguments

`x`	Estimated model.
`y`	Coefficient matrix to be compared.
`...`	not used

Details

Let $\lVert \cdot \rVert_2$ be the spectral norm of a matrix, let $\hat{\Phi}$ be the estimates, and let Φ be the true coefficients matrix. Then the function computes relative estimation error by

$$\frac{\lVert \hat{\Phi} - \Phi \rVert_2}{\lVert \Phi \rVert_2}$$

Value

Spectral norm value

References

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

`residuals` *Residual Matrix from Multivariate Time Series Models*

Description

By defining `stats::residuals()` for each model, this function returns residual.

Usage

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

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

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

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

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

Arguments

`object`	Model object
`...`	not used

Value

matrix object.

rmafe *Evaluate the Model Based on RMAFE*

Description

This function computes RMAFE (Mean Absolute Forecast Error Relative to the Benchmark)

Usage

```
rmafe(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
rmafe(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
rmafe(x, pred_bench, y, ...)
```

Arguments

`x`	Forecasting object to use
`pred_bench`	The same forecasting object from benchmark model
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. RMAFE is the ratio of L1 norm of e_t from forecasting object and from benchmark model.

$$RMAFE = \frac{sum(\lVert e_t \rVert)}{sum(\lVert e_t^{(b)} \rVert)}$$

where $e_t^{(b)}$ is the error from the benchmark model.

Value

RMAFE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

`rmape` *Evaluate the Model Based on RMAPE (Relative MAPE)*

Description

This function computes RMAPE given prediction result versus evaluation set.

Usage

```
rmape(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
rmape(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
rmape(x, pred_bench, y, ...)
```

Arguments

`x`	Forecasting object to use
`pred_bench`	The same forecasting object from benchmark model
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

RMAPE is the ratio of MAPE of given model and the benchmark one. Let $MAPE_b$ be the MAPE of the benchmark model. Then

$$RMAPE = \frac{mean(MAPE)}{mean(MAPE_b)}$$

where $MAPE$ is the MAPE of our model.

Value

RMAPE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

`rmase` *Evaluate the Model Based on RMASE (Relative MASE)*

Description

This function computes RMASE given prediction result versus evaluation set.

Usage

```
rmase(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
rmase(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
rmase(x, pred_bench, y, ...)
```

Arguments

`x`	Forecasting object to use
`pred_bench`	The same forecasting object from benchmark model
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

RMASE is the ratio of MAPE of given model and the benchmark one. Let $MASE_b$ be the MAPE of the benchmark model. Then

$$RMASE = \frac{mean(MASE)}{mean(MASE_b)}$$

where $MASE$ is the MASE of our model.

Value

RMASE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

`rmsfe` *Evaluate the Model Based on RMSFE*

Description

This function computes RMSFE (Mean Squared Forecast Error Relative to the Benchmark)

Usage

```
rmsfe(x, pred_bench, y, ...)

## S3 method for class 'predbvhar'
rmsfe(x, pred_bench, y, ...)

## S3 method for class 'bvharcv'
rmsfe(x, pred_bench, y, ...)
```

Arguments

`x`	Forecasting object to use
`pred_bench`	The same forecasting object from benchmark model
`y`	Test data to be compared. should be the same format with the train data.
`...`	not used

Details

Let $e_t = y_t - \hat{y}_t$. RMSFE is the ratio of L2 norm of e_t from forecasting object and from benchmark model.

$$RMSFE = \frac{sum(\lVert e_t \rVert)}{sum(\lVert e_t^{(b)} \rVert)}$$

where $e_t^{(b)}$ is the error from the benchmark model.

Value

RMSFE vector corresponding to each variable.

References

Hyndman, R. J., & Koehler, A. B. (2006). *Another look at measures of forecast accuracy*. International Journal of Forecasting, 22(4), 679-688.

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

set_bvar *Hyperparameters for Bayesian Models*

Description

Set hyperparameters of Bayesian VAR and VHAR models.

Usage

```
set_bvar(sigma, lambda = 0.1, delta, eps = 1e-04)

set_bvar_flat(U)

set_bvhar(sigma, lambda = 0.1, delta, eps = 1e-04)

set_weight_bvhar(sigma, lambda = 0.1, eps = 1e-04, daily, weekly, monthly)

## S3 method for class 'bvharspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.bvharspec(x)

## S3 method for class 'bvharspec'
knit_print(x, ...)
```

Arguments

sigma	Standard error vector for each variable (Default: sd)
lambda	Tightness of the prior around a random walk or white noise (Default: .1)
delta	Persistence (Default: Litterman sets 1 = random walk prior, White noise prior = 0)
eps	Very small number (Default: 1e-04)
U	Positive definite matrix. By default, identity matrix of dimension ncol(X0)
daily	Same as delta in VHAR type (Default: 1 as Litterman)
weekly	Fill the second part in the first block (Default: 1)
monthly	Fill the third part in the first block (Default: 1)
x	bvharspec object
digits	digit option to print
...	not used

Details

- Missing arguments will be set to be default values in each model function mentioned above.
- set_bvar() sets hyperparameters for bvar_minnesota().
- Each delta (vector), lambda (length of 1), sigma (vector), eps (vector) corresponds to δ_j, λ, δ_j, ϵ.

δ_i are related to the belief to random walk.

- If $\delta_i = 1$ for all i, random walk prior
- If $\delta_i = 0$ for all i, white noise prior

λ controls the overall tightness of the prior around these two prior beliefs.

- If $\lambda = 0$, the posterior is equivalent to prior and the data do not influence the estimates.
- If $\lambda = \infty$, the posterior mean becomes OLS estimates (VAR).

σ_i^2/σ_j^2 in Minnesota moments explain the data scales.

- set_bvar_flat sets hyperparameters for bvar_flat().
- set_bvhar() sets hyperparameters for bvhar_minnesota() with VAR-type Minnesota prior, i.e. BVHAR-S model.
- set_weight_bvhar() sets hyperparameters for bvhar_minnesota() with VHAR-type Minnesota prior, i.e. BVHAR-L model.

Value

Every function returns bvharspec class. It is the list of which the components are the same as the arguments provided. If the argument is not specified, NULL is assigned here. The default values mentioned above will be considered in each fitting function.

process Model name: BVAR, BVHAR

prior Prior name: Minnesota (Minnesota prior for BVAR), Hierarchical (Hierarchical prior for BVAR), MN_VAR (BVHAR-S), MN_VHAR (BVHAR-L), Flat (Flat prior for BVAR)

sigma Vector value (or bvharpriorspec class) assigned for sigma

lambda Value (or bvharpriorspec class) assigned for lambda

delta Vector value assigned for delta

eps Value assigned for epsilon

set_weight_bvhar() has different component with delta due to its different construction.

daily Vector value assigned for daily weight

weekly Vector value assigned for weekly weight

monthly Vector value assigned for monthly weight

Note

By using set_psi() and set_lambda() each, hierarchical modeling is available.

References

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

See Also

- lambda hyperprior specification `set_lambda()`
- sigma hyperprior specification `set_psi()`

Examples

```
# Minnesota BVAR specification------------------------
bvar_spec <- set_bvar(
  sigma = c(.03, .02, .01), # Sigma = diag(.03^2, .02^2, .01^2)
  lambda = .2, # lambda = .2
  delta = rep(.1, 3), # delta1 = .1, delta2 = .1, delta3 = .1
  eps = 1e-04 # eps = 1e-04
)
class(bvar_spec)
str(bvar_spec)
# Flat BVAR specification-------------------------
# 3-dim
# p = 5 with constant term
# U = 500 * I(mp + 1)
bvar_flat_spec <- set_bvar_flat(U = 500 * diag(16))
class(bvar_flat_spec)
str(bvar_flat_spec)
# BVHAR-S specification-----------------------
bvhar_var_spec <- set_bvhar(
  sigma = c(.03, .02, .01), # Sigma = diag(.03^2, .02^2, .01^2)
  lambda = .2, # lambda = .2
  delta = rep(.1, 3), # delta1 = .1, delta2 = .1, delta3 = .1
  eps = 1e-04 # eps = 1e-04
)
class(bvhar_var_spec)
str(bvhar_var_spec)
# BVHAR-L specification---------------------------
bvhar_vhar_spec <- set_weight_bvhar(
  sigma = c(.03, .02, .01), # Sigma = diag(.03^2, .02^2, .01^2)
  lambda = .2, # lambda = .2
  eps = 1e-04, # eps = 1e-04
```

```
  daily = rep(.2, 3), # daily1 = .2, daily2 = .2, daily3 = .2
  weekly = rep(.1, 3), # weekly1 = .1, weekly2 = .1, weekly3 = .1
  monthly = rep(.05, 3) # monthly1 = .05, monthly2 = .05, monthly3 = .05
)
class(bvhar_vhar_spec)
str(bvhar_vhar_spec)
```

---

set_dl *Dirichlet-Laplace Hyperparameter for Coefficients and Contemporaneous Coefficients*

Description

[Experimental] Set DL hyperparameters for VAR or VHAR coefficient and contemporaneous coefficient.

Usage

```
set_dl(dir_grid = 100L, shape = 0.01, rate = 0.01)

## S3 method for class 'dlspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.dlspec(x)
```

Arguments

dir_grid	Griddy gibbs grid size for Dirichlet hyperparameter
shape	Gamma shape
rate	Gamma rate
x	dlspec
digits	digit option to print
...	not used

Value

dlspec object

References

Bhattacharya, A., Pati, D., Pillai, N. S., & Dunson, D. B. (2015). *Dirichlet-Laplace Priors for Optimal Shrinkage*. Journal of the American Statistical Association, 110(512), 1479-1490.

Korobilis, D., & Shimizu, K. (2022). *Bayesian Approaches to Shrinkage and Sparse Estimation*. Foundations and Trends® in Econometrics, 11(4), 230-354.

set_horseshoe — *Horseshoe Prior Specification*

Description

Set initial hyperparameters and parameter before starting Gibbs sampler for Horseshoe prior.

Usage

```
set_horseshoe(local_sparsity = 1, group_sparsity = 1, global_sparsity = 1)

## S3 method for class 'horseshoespec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.horseshoespec(x)

## S3 method for class 'horseshoespec'
knit_print(x, ...)
```

Arguments

`local_sparsity`	Initial local shrinkage hyperparameters
`group_sparsity`	Initial group shrinkage hyperparameters
`global_sparsity`	Initial global shrinkage hyperparameter
`x`	`horseshoespec`
`digits`	digit option to print
`...`	not used

Details

Set horseshoe prior initialization for VAR family.

- `local_sparsity`: Initial local shrinkage
- `group_sparsity`: Initial group shrinkage
- `global_sparsity`: Initial global shrinkage

In this package, horseshoe prior model is estimated by Gibbs sampling, initial means initial values for that gibbs sampler.

References

Carvalho, C. M., Polson, N. G., & Scott, J. G. (2010). The horseshoe estimator for sparse signals. Biometrika, 97(2), 465-480.

Makalic, E., & Schmidt, D. F. (2016). *A Simple Sampler for the Horseshoe Estimator*. IEEE Signal Processing Letters, 23(1), 179-182.

set_intercept *Prior for Constant Term*

Description

Set Normal prior hyperparameters for constant term

Usage

```
set_intercept(mean = 0, sd = 0.1)

## S3 method for class 'interceptspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.interceptspec(x)

## S3 method for class 'interceptspec'
knit_print(x, ...)
```

Arguments

mean	Normal mean of constant term
sd	Normal standard deviance for constant term
x	`interceptspec` object
digits	digit option to print
...	not used

set_lambda *Hyperpriors for Bayesian Models*

Description

Set hyperpriors of Bayesian VAR and VHAR models.

Usage

```
set_lambda(mode = 0.2, sd = 0.4, param = NULL, lower = 1e-05, upper = 3)

set_psi(shape = 4e-04, scale = 4e-04, lower = 1e-05, upper = 3)

## S3 method for class 'bvharpriorspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.bvharpriorspec(x)

## S3 method for class 'bvharpriorspec'
knit_print(x, ...)
```

Arguments

`mode`	Mode of Gamma distribution. By default, `.2`.
`sd`	Standard deviation of Gamma distribution. By default, `.4`.
`param`	Shape and rate of Gamma distribution, in the form of `c(shape, rate)`. If specified, ignore `mode` and `sd`.
`lower`	**[Experimental]** Lower bound for `stats::optim()`. By default, `1e-5`.
`upper`	**[Experimental]** Upper bound for `stats::optim()`. By default, `3`.
`shape`	Shape of Inverse Gamma distribution. By default, `(.02)^2`.
`scale`	Scale of Inverse Gamma distribution. By default, `(.02)^2`.
`x`	`bvharpriorspec` object
`digits`	digit option to print
`...`	not used

Details

In addition to Normal-IW priors `set_bvar()`, `set_bvhar()`, and `set_weight_bvhar()`, these functions give hierarchical structure to the model.

- `set_lambda()` specifies hyperprior for λ (`lambda`), which is Gamma distribution.
- `set_psi()` specifies hyperprior for $\psi/(\nu_0 - k - 1) = \sigma^2$ (`sigma`), which is Inverse gamma distribution.

The following set of `(mode, sd)` are recommended by Sims and Zha (1998) for `set_lambda()`.

- `(mode = .2, sd = .4)`: default
- `(mode = 1, sd = 1)`

Giannone et al. (2015) suggested data-based selection for `set_psi()`. It chooses (0.02)^2 based on its empirical data set.

Value

`bvharpriorspec` object

References

Giannone, D., Lenza, M., & Primiceri, G. E. (2015). *Prior Selection for Vector Autoregressions*. Review of Economics and Statistics, 97(2).

Examples

```
# Hirearchical BVAR specification------------------------
set_bvar(
  sigma = set_psi(shape = 4e-4, scale = 4e-4),
  lambda = set_lambda(mode = .2, sd = .4),
  delta = rep(1, 3),
  eps = 1e-04 # eps = 1e-04
)
```

set_ldlt *Covariance Matrix Prior Specification*

Description

[Experimental] Set prior for covariance matrix.

Usage

```
set_ldlt(ig_shape = 3, ig_scl = 0.01)

set_sv(ig_shape = 3, ig_scl = 0.01, initial_mean = 1, initial_prec = 0.1)

## S3 method for class 'covspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.covspec(x)

is.svspec(x)

is.ldltspec(x)
```

Arguments

ig_shape	Inverse-Gamma shape of Cholesky diagonal vector. For SV (set_sv()), this is for state variance.
ig_scl	Inverse-Gamma scale of Cholesky diagonal vector. For SV (set_sv()), this is for state variance.
initial_mean	Prior mean of initial state.
initial_prec	Prior precision of initial state.
x	covspec
digits	digit option to print
...	not used

Details

set_ldlt() specifies LDLT of precision matrix,

$$\Sigma^{-1} = L^T D^{-1} L$$

set_sv() specifices time varying precision matrix under stochastic volatility framework based on

$$\Sigma_t^{-1} = L^T D_t^{-1} L$$

References

Carriero, A., Chan, J., Clark, T. E., & Marcellino, M. (2022). *Corrigendum to "Large Bayesian vector autoregressions with stochastic volatility and non-conjugate priors" [J. Econometrics 212 (1)(2019) 137-154]*. Journal of Econometrics, 227(2), 506-512.

Chan, J., Koop, G., Poirier, D., & Tobias, J. (2019). *Bayesian Econometric Methods (2nd ed., Econometric Exercises)*. Cambridge: Cambridge University Press.

---

set_ng — *Normal-Gamma Hyperparameter for Coefficients and Contemporaneous Coefficients*

Description

[Experimental] Set NG hyperparameters for VAR or VHAR coefficient and contemporaneous coefficient.

Usage

```
set_ng(
  shape_sd = 0.01,
  group_shape = 0.01,
  group_scale = 0.01,
  global_shape = 0.01,
  global_scale = 0.01,
  contem_global_shape = 0.01,
  contem_global_scale = 0.01
)

## S3 method for class 'ngspec'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.ngspec(x)
```

Arguments

`shape_sd`	Standard deviation used in MH of Gamma shape
`group_shape`	Inverse gamma prior shape for coefficient group shrinkage
`group_scale`	Inverse gamma prior scale for coefficient group shrinkage
`global_shape`	Inverse gamma prior shape for coefficient global shrinkage
`global_scale`	Inverse gamma prior scale for coefficient global shrinkage
`contem_global_shape`	Inverse gamma prior shape for contemporaneous coefficient global shrinkage
`contem_global_scale`	Inverse gamma prior scale for contemporaneous coefficient global shrinkage

x	ngspec
digits	digit option to print
...	not used

Value

ngspec object

References

Chan, J. C. C. (2021). *Minnesota-type adaptive hierarchical priors for large Bayesian VARs*. International Journal of Forecasting, 37(3), 1212-1226.

Huber, F., & Feldkircher, M. (2019). *Adaptive Shrinkage in Bayesian Vector Autoregressive Models*. Journal of Business & Economic Statistics, 37(1), 27-39.

Korobilis, D., & Shimizu, K. (2022). *Bayesian Approaches to Shrinkage and Sparse Estimation*. Foundations and Trends® in Econometrics, 11(4), 230-354.

---

set_ssvs *Stochastic Search Variable Selection (SSVS) Hyperparameter for Coefficients Matrix and Cholesky Factor*

Description

Set SSVS hyperparameters for VAR or VHAR coefficient matrix and Cholesky factor.

Usage

```
set_ssvs(
  coef_spike = 0.1,
  coef_slab = 5,
  coef_spike_scl = 0.01,
  coef_slab_shape = 0.01,
  coef_slab_scl = 0.01,
  coef_mixture = 0.5,
  coef_s1 = c(1, 1),
  coef_s2 = c(1, 1),
  mean_non = 0,
  sd_non = 0.1,
  shape = 0.01,
  rate = 0.01,
  chol_spike = 0.1,
  chol_slab = 5,
  chol_spike_scl = 0.01,
  chol_slab_shape = 0.01,
  chol_slab_scl = 0.01,
  chol_mixture = 0.5,
```

```
  chol_s1 = 1,
  chol_s2 = 1
)

## S3 method for class 'ssvsinput'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

is.ssvsinput(x)

## S3 method for class 'ssvsinput'
knit_print(x, ...)
```

Arguments

coef_spike	**[Deprecated]** Standard deviance for Spike normal distribution. Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
coef_slab	**[Deprecated]** Standard deviance for Slab normal distribution. Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
coef_spike_scl	Scaling factor (between 0 and 1) for spike sd which is Spike sd = c * slab sd
coef_slab_shape	Inverse gamma shape for slab sd
coef_slab_scl	Inverse gamma scale for slab sd
coef_mixture	**[Deprecated]** Bernoulli parameter for sparsity proportion. Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
coef_s1	First shape of coefficients prior beta distribution
coef_s2	Second shape of coefficients prior beta distribution
mean_non	**[Deprecated]** Prior mean of unrestricted coefficients Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
sd_non	**[Deprecated]** Standard deviance for unrestricted coefficients Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
shape	Gamma shape parameters for precision matrix (See Details).
rate	Gamma rate parameters for precision matrix (See Details).
chol_spike	Standard deviance for Spike normal distribution, in the cholesky factor. Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
chol_slab	Standard deviance for Slab normal distribution, in the cholesky factor. Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.
chol_spike_scl	Scaling factor (between 0 and 1) for spike sd which is Spike sd = c * slab sd in the cholesky factor
chol_slab_shape	Inverse gamma shape for slab sd in the cholesky factor
chol_slab_scl	Inverse gamma scale for slab sd in the cholesky factor
chol_mixture	**[Deprecated]** Bernoulli parameter for sparsity proportion, in the cholesky factor (See Details). Will be deleted when bvar_ssvs() and bvhar_ssvs() are removed in the package.

`chol_s1`	First shape of cholesky factor prior beta distribution
`chol_s2`	Second shape of cholesky factor prior beta distribution
`x`	`ssvsinput`
`digits`	digit option to print
`...`	not used

Details

Let α be the vectorized coefficient, $\alpha = vec(A)$. Spike-slab prior is given using two normal distributions.

$$\alpha_j \mid \gamma_j \sim (1 - \gamma_j) N(0, \tau_{0j}^2) + \gamma_j N(0, \tau_{1j}^2)$$

As spike-slab prior itself suggests, set τ_{0j} small (point mass at zero: spike distribution) and set τ_{1j} large (symmetric by zero: slab distribution).

γ_j is the proportion of the nonzero coefficients and it follows

$$\gamma_j \sim Bernoulli(p_j)$$

- `coef_spike`: τ_{0j}
- `coef_slab`: τ_{1j}
- `coef_mixture`: p_j
- $j = 1, \ldots, mk$: vectorized format corresponding to coefficient matrix
- If one value is provided, model function will read it by replicated value.
- `coef_non`: vectorized constant term is given prior Normal distribution with variance cI. Here, `coef_non` is $\sqrt{c}$.

Next for precision matrix Σ_e^{-1}, SSVS applies Cholesky decomposition.

$$\Sigma_e^{-1} = \Psi \Psi^T$$

where $\Psi = \{\psi_{ij}\}$ is upper triangular.

Diagonal components follow the gamma distribution.

$$\psi_{jj}^2 \sim Gamma(shape = a_j, rate = b_j)$$

For each row of off-diagonal (upper-triangular) components, we apply spike-slab prior again.

$$\psi_{ij} \mid w_{ij} \sim (1 - w_{ij}) N(0, \kappa_{0,ij}^2) + w_{ij} N(0, \kappa_{1,ij}^2)$$

$$w_{ij} \sim Bernoulli(q_{ij})$$

- `shape`: a_j
- `rate`: b_j
- `chol_spike`: $\kappa_{0,ij}$
- `chol_slab`: $\kappa_{1,ij}$
- `chol_mixture`: q_{ij}
- $j = 1, \ldots, mk$: vectorized format corresponding to coefficient matrix
- $i = 1, \ldots, j-1$ and $j = 2, \ldots, m$: $\eta = (\psi_{12}, \psi_{13}, \psi_{23}, \psi_{14}, \ldots, \psi_{34}, \ldots, \psi_{1m}, \ldots, \psi_{m-1,m})^T$
- `chol_` arguments can be one value for replication, vector, or upper triangular matrix.

Value

`ssvsinput` object

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Ishwaran, H., & Rao, J. S. (2005). *Spike and slab variable selection: Frequentist and Bayesian strategies*. The Annals of Statistics, 33(2).

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

`sim_gig` *Generate Generalized Inverse Gaussian Distribution*

Description

This function samples $GIG(\lambda, \psi, \chi)$ random variates.

Usage

```
sim_gig(num_sim, lambda, psi, chi)
```

Arguments

`num_sim`	Number to generate
`lambda`	Index of modified Bessel function of third kind.
`psi`	Second parameter of GIG. Should be positive.
`chi`	Third parameter of GIG. Should be positive.

Details

The density of $GIG(\lambda, \psi, \chi)$ considered here is as follows.

$$f(x) = \frac{(\psi/\chi)^(\lambda/2)}{2K_\lambda(\sqrt{\psi\chi})} x^{\lambda-1} \exp(-\frac{1}{2}(\frac{\chi}{x} + \psi x))$$

where $x > 0$.

References

Hörmann, W., Leydold, J. *Generating generalized inverse Gaussian random variates*. Stat Comput 24, 547-557 (2014).

Leydold, J, Hörmann, W.. *GIGrvg: Random Variate Generator for the GIG Distribution*. R package version 0.8 (2023).

sim_horseshoe_var *Generate Horseshoe Parameters*

Description

[Deprecated] This function generates parameters of VAR with Horseshoe prior.

Usage

```
sim_horseshoe_var(
  p,
  dim_data = NULL,
  include_mean = TRUE,
  minnesota = FALSE,
  method = c("eigen", "chol")
)

sim_horseshoe_vhar(
  har = c(5, 22),
  dim_data = NULL,
  include_mean = TRUE,
  minnesota = c("no", "short", "longrun"),
  method = c("eigen", "chol")
)
```

Arguments

p	VAR lag
dim_data	Specify the dimension of the data if hyperparameters of bayes_spec have constant values.
include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Only use off-diagonal terms of each coefficient matrices for restriction. In sim_horseshoe_var() function, use TRUE or FALSE (default). In sim_horseshoe_vhar() function, no (default), short type, or longrun type.
method	Method to compute $\Sigma^{1/2}$.
har	Numeric vector for weekly and monthly order. By default, c(5, 22).

`sim_iw` *Generate Inverse-Wishart Random Matrix*

Description

This function samples one matrix IW matrix.

Usage

```
sim_iw(mat_scale, shape)
```

Arguments

`mat_scale`	Scale matrix
`shape`	Shape

Details

Consider $\Sigma \sim IW(\Psi, \nu)$.

1. Upper triangular Bartlett decomposition: k x k matrix $Q = [q_{ij}]$ upper triangular with
   (a) $q_{ii}^2 \chi_{\nu-i+1}^2$
   (b) $q_{ij} \sim N(0, 1)$ with i < j (upper triangular)
2. Lower triangular Cholesky decomposition: $\Psi = LL^T$
3. $A = L(Q^{-1})^T$
4. $\Sigma = AA^T \sim IW(\Psi, \nu)$

Value

One k x k matrix following IW distribution

`sim_matgaussian` *Generate Matrix Normal Random Matrix*

Description

This function samples one matrix gaussian matrix.

Usage

```
sim_matgaussian(mat_mean, mat_scale_u, mat_scale_v, u_prec)
```

Arguments

`mat_mean`	Mean matrix
`mat_scale_u`	First scale matrix
`mat_scale_v`	Second scale matrix
`u_prec`	If `TRUE`, use `mat_scale_u` as its inverse.

Details

Consider n x k matrix $Y_1, \ldots, Y_n \sim MN(M, U, V)$ where M is n x k, U is n x n, and V is k x k.

1. Lower triangular Cholesky decomposition: $U = PP^T$ and $V = LL^T$
2. Standard normal generation: s x m matrix $Z_i = [z_{ij} \sim N(0, 1)]$ in row-wise direction.
3. $Y_i = M + PZ_iL^T$

This function only generates one matrix, i.e. Y_1.

Value

One n x k matrix following MN distribution.

`sim_mncoef` *Generate Minnesota BVAR Parameters*

Description

This function generates parameters of BVAR with Minnesota prior.

Usage

```
sim_mncoef(p, bayes_spec = set_bvar(), full = TRUE)
```

Arguments

`p`	VAR lag
`bayes_spec`	A BVAR model specification by `set_bvar()`.
`full`	Generate variance matrix from IW (default: `TRUE`) or not (`FALSE`)?

Details

Implementing dummy observation constructions, Bańbura et al. (2010) sets Normal-IW prior.

$$A \mid \Sigma_e \sim MN(A_0, \Omega_0, \Sigma_e)$$

$$\Sigma_e \sim IW(S_0, \alpha_0)$$

If `full = FALSE`, the result of Σ_e is the same as input (`diag(sigma)`).

Value

List with the following component.

coefficients BVAR coefficient (MN)

covmat BVAR variance (IW or diagonal matrix of sigma of bayes_spec)

References

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

Karlsson, S. (2013). *Chapter 15 Forecasting with Bayesian Vector Autoregression*. Handbook of Economic Forecasting, 2, 791-897.

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

See Also

- set_bvar() to specify the hyperparameters of Minnesota prior.

Examples

```
# Generate (A, Sigma)
# BVAR(p = 2)
# sigma: 1, 1, 1
# lambda: .1
# delta: .1, .1, .1
# epsilon: 1e-04
set.seed(1)
sim_mncoef(
  p = 2,
  bayes_spec = set_bvar(
    sigma = rep(1, 3),
    lambda = .1,
    delta = rep(.1, 3),
    eps = 1e-04
  ),
  full = TRUE
)
```

---

sim_mniw *Generate Normal-IW Random Family*

Description

This function samples normal inverse-wishart matrices.

Usage

```
sim_mniw(num_sim, mat_mean, mat_scale_u, mat_scale, shape, u_prec = FALSE)
```

Arguments

num_sim	Number to generate
mat_mean	Mean matrix of MN
mat_scale_u	First scale matrix of MN
mat_scale	Scale matrix of IW
shape	Shape of IW
u_prec	If TRUE, use mat_scale_u as its inverse. By default, FALSE.

Details

Consider $(Y_i, \Sigma_i) \sim MIW(M, U, \Psi, \nu)$.

1. Generate upper triangular factor of $\Sigma_i = C_i C_i^T$ in the upper triangular Bartlett decomposition.
2. Standard normal generation: n x k matrix $Z_i = [z_{ij} \sim N(0, 1)]$ in row-wise direction.
3. Lower triangular Cholesky decomposition: $U = PP^T$
4. $A_i = M + PZ_i C_i^T$

---

sim_mnormal *Generate Multivariate Normal Random Vector*

Description

This function samples n x muti-dimensional normal random matrix.

Usage

```
sim_mnormal(
  num_sim,
  mu = rep(0, 5),
  sig = diag(5),
  method = c("eigen", "chol")
)
```

Arguments

num_sim	Number to generate process
mu	Mean vector
sig	Variance matrix
method	Method to compute $\Sigma^{1/2}$. Choose between eigen (spectral decomposition) and chol (cholesky decomposition). By default, eigen.

Details

Consider $x_1, \ldots, x_n \sim N_m(\mu, \Sigma)$.

1. Lower triangular Cholesky decomposition: $\Sigma = LL^T$
2. Standard normal generation: $Z_{i1}, Z_{in} \stackrel{iid}{\sim} N(0, 1)$
3. $Z_i = (Z_{i1}, \ldots, Z_{in})^T$
4. $X_i = LZ_i + \mu$

Value

T x k matrix

`sim_mnvhar_coef` *Generate Minnesota BVAR Parameters*

Description

This function generates parameters of BVAR with Minnesota prior.

Usage

```
sim_mnvhar_coef(bayes_spec = set_bvhar(), full = TRUE)
```

Arguments

`bayes_spec`	A BVHAR model specification by `set_bvhar()` (default) or `set_weight_bvhar()`.
`full`	Generate variance matrix from IW (default: `TRUE`) or not (`FALSE`)?

Details

Normal-IW family for vector HAR model:

$$\Phi \mid \Sigma_e \sim MN(M_0, \Omega_0, \Sigma_e)$$

$$\Sigma_e \sim IW(\Psi_0, \nu_0)$$

Value

List with the following component.

coefficients BVHAR coefficient (MN)

covmat BVHAR variance (IW or diagonal matrix of `sigma` of `bayes_spec`)

References

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

See Also

- set_bvhar() to specify the hyperparameters of VAR-type Minnesota prior.
- set_weight_bvhar() to specify the hyperparameters of HAR-type Minnesota prior.

Examples

```
# Generate (Phi, Sigma)
# BVHAR-S
# sigma: 1, 1, 1
# lambda: .1
# delta: .1, .1, .1
# epsilon: 1e-04
set.seed(1)
sim_mnvhar_coef(
  bayes_spec = set_bvhar(
    sigma = rep(1, 3),
    lambda = .1,
    delta = rep(.1, 3),
    eps = 1e-04
  ),
  full = TRUE
)
```

---

sim_mvt *Generate Multivariate t Random Vector*

Description

This function samples n x multi-dimensional t-random matrix.

Usage

```
sim_mvt(num_sim, df, mu, sig, method = c("eigen", "chol"))
```

Arguments

num_sim	Number to generate process.
df	Degrees of freedom.
mu	Location vector
sig	Scale matrix.
method	Method to compute $\Sigma^{1/2}$. Choose between eigen (spectral decomposition) and chol (cholesky decomposition). By default, eigen.

Value

T x k matrix

sim_ssvs_var   *Generate SSVS Parameters*

Description

[Deprecated] This function generates parameters of VAR with SSVS prior.

Usage

```
sim_ssvs_var(
  bayes_spec,
  p,
  dim_data = NULL,
  include_mean = TRUE,
  minnesota = FALSE,
  mn_prob = 1,
  method = c("eigen", "chol")
)

sim_ssvs_vhar(
  bayes_spec,
  har = c(5, 22),
  dim_data = NULL,
  include_mean = TRUE,
  minnesota = c("no", "short", "longrun"),
  mn_prob = 1,
  method = c("eigen", "chol")
)
```

Arguments

bayes_spec	A SSVS model specification by set_ssvs().
p	VAR lag
dim_data	Specify the dimension of the data if hyperparameters of bayes_spec have constant values.
include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Only use off-diagonal terms of each coefficient matrices for restriction. In sim_ssvs_var() function, use TRUE or FALSE (default). In sim_ssvs_vhar() function, no (default), short type, or longrun type.
mn_prob	Probability for own-lags.
method	Method to compute $\Sigma^{1/2}$.
har	Numeric vector for weekly and monthly order. By default, c(5, 22).

Value

List including coefficients.

VAR(p) with SSVS prior

Let α be the vectorized coefficient of VAR(p).

$$(\alpha \mid \gamma)$$

$$(\gamma_i)$$

$$(\eta_j \mid \omega_j)$$

$$(\omega_{ij})$$

$$(\psi_{ii}^2)$$

VHAR with SSVS prior

Let ϕ be the vectorized coefficient of VHAR.

$$(\phi \mid \gamma)$$

$$(\gamma_i)$$

$$(\eta_j \mid \omega_j)$$

$$(\omega_{ij})$$

$$(\psi_{ii}^2)$$

References

George, E. I., & McCulloch, R. E. (1993). *Variable Selection via Gibbs Sampling*. Journal of the American Statistical Association, 88(423), 881-889.

George, E. I., Sun, D., & Ni, S. (2008). *Bayesian stochastic search for VAR model restrictions*. Journal of Econometrics, 142(1), 553-580.

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

Koop, G., & Korobilis, D. (2009). *Bayesian Multivariate Time Series Methods for Empirical Macroeconomics*. Foundations and Trends® in Econometrics, 3(4), 267-358.

sim_var — Generate Multivariate Time Series Process Following VAR(p)

Description

This function generates multivariate time series dataset that follows VAR(p).

Usage

```
sim_var(
  num_sim,
  num_burn,
  var_coef,
  var_lag,
  sig_error = diag(ncol(var_coef)),
  init = matrix(0L, nrow = var_lag, ncol = ncol(var_coef)),
  method = c("eigen", "chol"),
  process = c("gaussian", "student"),
  t_param = 5
)
```

Arguments

`num_sim`	Number to generated process
`num_burn`	Number of burn-in
`var_coef`	VAR coefficient. The format should be the same as the output of `coef()` from `var_lm()`
`var_lag`	Lag of VAR
`sig_error`	Variance matrix of the error term. By default, `diag(dim)`.
`init`	Initial y1, ..., yp matrix to simulate VAR model. Try `matrix(0L, nrow = var_lag, ncol = dim)`.
`method`	Method to compute $\Sigma^{1/2}$. Choose between `eigen` (spectral decomposition) and `chol` (cholesky decomposition). By default, `eigen`.
`process`	Process to generate error term. `gaussian`: Normal distribution (default) or `student`: Multivariate t-distribution.
`t_param`	**[Experimental]** argument for MVT, e.g. DF: 5.

Details

1. Generate $\epsilon_1, \epsilon_n \sim N(0, \Sigma)$
2. For i = 1, ... n,
$$y_{p+i} = (y_{p+i-1}^T, \ldots, y_i^T, 1)^T B + \epsilon_i$$
3. Then the output is $(y_{p+1}, \ldots, y_{n+p})^T$

Initial values might be set to be zero vector or $(I_m - A_1 - \cdots - A_p)^{-1} c$.

Value

T x k matrix

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

`sim_vhar` *Generate Multivariate Time Series Process Following VAR(p)*

Description

This function generates multivariate time series dataset that follows VAR(p).

Usage

```
sim_vhar(
  num_sim,
  num_burn,
  vhar_coef,
  week = 5L,
  month = 22L,
  sig_error = diag(ncol(vhar_coef)),
  init = matrix(0L, nrow = month, ncol = ncol(vhar_coef)),
  method = c("eigen", "chol"),
  process = c("gaussian", "student"),
  t_param = 5
)
```

Arguments

`num_sim`	Number to generated process
`num_burn`	Number of burn-in
`vhar_coef`	VAR coefficient. The format should be the same as the output of `coef()` from `var_lm()`
`week`	Weekly order of VHAR. By default, 5.
`month`	Weekly order of VHAR. By default, 22.
`sig_error`	Variance matrix of the error term. By default, `diag(dim)`.
`init`	Initial y1, ..., yp matrix to simulate VAR model. Try `matrix(0L, nrow = month, ncol = dim)`.
`method`	Method to compute $\Sigma^{1/2}$. Choose between `eigen` (spectral decomposition) and `chol` (cholesky decomposition). By default, `eigen`.
`process`	Process to generate error term. `gaussian`: Normal distribution (default) or `student`: Multivariate t-distribution.
`t_param`	**[Experimental]** argument for MVT, e.g. DF: 5.

Details

Let M be the month order, e.g. $M = 22$.

1. Generate $\epsilon_1, \epsilon_n \sim N(0, \Sigma)$
2. For i = 1, ... n,
$$y_{M+i} = (y_{M+i-1}^T, \ldots, y_i^T, 1)^T C_{HAR}^T \Phi + \epsilon_i$$
3. Then the output is $(y_{M+1}, \ldots, y_{n+M})^T$
4. For i = 1, ... n,
$$y_{p+i} = (y_{p+i-1}^T, \ldots, y_i^T, 1)^T B + \epsilon_i$$
5. Then the output is $(y_{p+1}, \ldots, y_{n+p})^T$

Initial values might be set to be zero vector or $(I_m - A_1 - \cdots - A_p)^{-1} c$.

Value

T x k matrix

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

---

spillover *h-step ahead Normalized Spillover*

Description

This function gives connectedness table with h-step ahead normalized spillover index (a.k.a. variance shares).

Usage

```
spillover(object, n_ahead = 10L, ...)

## S3 method for class 'bvharspillover'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharspillover'
knit_print(x, ...)

## S3 method for class 'olsmod'
spillover(object, n_ahead = 10L, ...)

## S3 method for class 'normaliw'
spillover(
  object,
  n_ahead = 10L,
```

```
  num_iter = 5000L,
  num_burn = floor(num_iter/2),
  thinning = 1L,
  ...
)

## S3 method for class 'bvarldlt'
spillover(object, n_ahead = 10L, sparse = FALSE, ...)

## S3 method for class 'bvharldlt'
spillover(object, n_ahead = 10L, sparse = FALSE, ...)
```

Arguments

`object`	Model object
`n_ahead`	step to forecast. By default, 10.
`...`	not used
`x`	`bvharspillover` object
`digits`	digit option to print
`num_iter`	Number to sample MNIW distribution
`num_burn`	Number of burn-in
`thinning`	Thinning every thinning-th iteration
`sparse`	**[Experimental]** Apply restriction. By default, `FALSE`.

References

Diebold, F. X., & Yilmaz, K. (2012). *Better to give than to receive: Predictive directional measurement of volatility spillovers*. International Journal of forecasting, 28(1), 57-66.

`spne` — *Evaluate the Estimation Based on Spectral Norm Error*

Description

This function computes estimation error given estimated model and true coefficient.

Usage

```
spne(x, y, ...)

## S3 method for class 'bvharsp'
spne(x, y, ...)
```

Arguments

`x`	Estimated model.
`y`	Coefficient matrix to be compared.
`...`	not used

Details

Let $\|\cdot\|_2$ be the spectral norm of a matrix, let $\hat{\Phi}$ be the estimates, and let Φ be the true coefficients matrix. Then the function computes estimation error by

$$\|\hat{\Phi} - \Phi\|_2$$

Value

Spectral norm value

References

Ghosh, S., Khare, K., & Michailidis, G. (2018). *High-Dimensional Posterior Consistency in Bayesian Vector Autoregressive Models*. Journal of the American Statistical Association, 114(526).

`stableroot` *Roots of characteristic polynomial*

Description

Compute the character polynomial of coefficient matrix.

Usage

```
stableroot(x, ...)

## S3 method for class 'varlse'
stableroot(x, ...)

## S3 method for class 'vharlse'
stableroot(x, ...)

## S3 method for class 'bvarmn'
stableroot(x, ...)

## S3 method for class 'bvarflat'
stableroot(x, ...)

## S3 method for class 'bvharmn'
stableroot(x, ...)
```

Arguments

x	Model fit
...	not used

Details

To know whether the process is stable or not, make characteristic polynomial.

$$\det(I_m - Az) = 0$$

where A is VAR(1) coefficient matrix representation.

Value

Numeric vector.

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

---

summary.normaliw — *Summarizing Bayesian Multivariate Time Series Model*

Description

summary method for normaliw class.

Usage

```
## S3 method for class 'normaliw'
summary(
  object,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  verbose = FALSE,
  num_thread = 1,
  ...
)

## S3 method for class 'summary.normaliw'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.normaliw'
knit_print(x, ...)
```

Arguments

`object`	A `normaliw` object
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`verbose`	Print the progress bar in the console. By default, `FALSE`.
`num_thread`	Number of threads
`...`	not used
`x`	`summary.normaliw` object
`digits`	digit option to print

Details

From Minnesota prior, set of coefficient matrices and residual covariance matrix have matrix Normal Inverse-Wishart distribution.

BVAR:

$$(A, \Sigma_e) \sim MNIW(\hat{A}, \hat{V}^{-1}, \hat{\Sigma}_e, \alpha_0 + n)$$

where $\hat{V} = X_*^T X_*$ is the posterior precision of MN.

BVHAR:

$$(\Phi, \Sigma_e) \sim MNIW(\hat{\Phi}, \hat{V}_H^{-1}, \hat{\Sigma}_e, \nu + n)$$

where $\hat{V}_H = X_+^T X_+$ is the posterior precision of MN.

Value

`summary.normaliw` class has the following components:

names Variable names

totobs Total number of the observation

obs Sample size used when training = `totobs` - `p`

p Lag of VAR

m Dimension of the data

call Matched call

spec Model specification (`bvharspec`)

mn_mean MN Mean of posterior distribution (MN-IW)

mn_prec MN Precision of posterior distribution (MN-IW)

iw_scale IW scale of posterior distribution (MN-IW)

iw_shape IW df of posterior distribution (MN-IW)

iter Number of MCMC iterations

burn Number of MCMC burn-in

thin MCMC thinning

alpha_record (BVAR) and phi_record (BVHAR) MCMC record of coefficients vector

psi_record MCMC record of upper cholesky factor

omega_record MCMC record of diagonal of cholesky factor

eta_record MCMC record of upper part of cholesky factor

param MCMC record of every parameter

coefficients Posterior mean of coefficients

covmat Posterior mean of covariance

References

Litterman, R. B. (1986). *Forecasting with Bayesian Vector Autoregressions: Five Years of Experience*. Journal of Business & Economic Statistics, 4(1), 25.

Bańbura, M., Giannone, D., & Reichlin, L. (2010). *Large Bayesian vector auto regressions*. Journal of Applied Econometrics, 25(1).

`summary.varlse` *Summarizing Vector Autoregressive Model*

Description

`summary` method for `varlse` class.

Usage

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

## S3 method for class 'summary.varlse'
print(x, digits = max(3L, getOption("digits") - 3L), signif_code = TRUE, ...)

## S3 method for class 'summary.varlse'
knit_print(x, ...)
```

Arguments

`object`	A `varlse` object
`...`	not used
`x`	`summary.varlse` object
`digits`	digit option to print
`signif_code`	Check significant rows (Default: `TRUE`)

Value

summary.varlse class additionally computes the following

names	Variable names
totobs	Total number of the observation
obs	Sample size used when training = totobs - p
p	Lag of VAR
coefficients	Coefficient Matrix
call	Matched call
process	Process: VAR
covmat	Covariance matrix of the residuals
corrmat	Correlation matrix of the residuals
roots	Roots of characteristic polynomials
is_stable	Whether the process is stable or not based on roots
log_lik	log-likelihood
ic	Information criteria vector

- AIC - AIC
- BIC - BIC
- HQ - HQ
- FPE - FPE

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

---

summary.vharlse *Summarizing Vector HAR Model*

Description

summary method for vharlse class.

Usage

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

## S3 method for class 'summary.vharlse'
print(x, digits = max(3L, getOption("digits") - 3L), signif_code = TRUE, ...)

## S3 method for class 'summary.vharlse'
knit_print(x, ...)
```

Arguments

`object`	A `vharlse` object
`...`	not used
`x`	`summary.vharlse` object
`digits`	digit option to print
`signif_code`	Check significant rows (Default: `TRUE`)

Value

`summary.vharlse` class additionally computes the following

`names`	Variable names
`totobs`	Total number of the observation
`obs`	Sample size used when training = `totobs` - `p`
`p`	3
`week`	Order for weekly term
`month`	Order for monthly term
`coefficients`	Coefficient Matrix
`call`	Matched call
`process`	Process: VAR
`covmat`	Covariance matrix of the residuals
`corrmat`	Correlation matrix of the residuals
`roots`	Roots of characteristic polynomials
`is_stable`	Whether the process is stable or not based on `roots`
`log_lik`	log-likelihood
`ic`	Information criteria vector

- `AIC` - AIC
- `BIC` - BIC
- `HQ` - HQ
- `FPE` - FPE

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

Corsi, F. (2008). *A Simple Approximate Long-Memory Model of Realized Volatility*. Journal of Financial Econometrics, 7(2), 174-196.

Baek, C. and Park, M. (2021). *Sparse vector heterogeneous autoregressive modeling for realized volatility*. J. Korean Stat. Soc. 50, 495-510.

VARtoVMA *Convert VAR to VMA(infinite)*

Description

Convert VAR process to infinite vector MA process

Usage

```
VARtoVMA(object, lag_max)
```

Arguments

`object`	A `varlse` object
`lag_max`	Maximum lag for VMA

Details

Let VAR(p) be stable.

$$Y_t = c + \sum_{j=0} W_j Z_{t-j}$$

For VAR coefficient $B_1, B_2, \ldots, B_p$,

$$I = (W_0 + W_1 L + W_2 L^2 + \cdots +)(I - B_1 L - B_2 L^2 - \cdots - B_p L^p)$$

Recursively,

$$W_0 = I$$

$$W_1 = W_0 B_1 (W_1^T = B_1^T W_0^T)$$

$$W_2 = W_1 B_1 + W_0 B_2 (W_2^T = B_1^T W_1^T + B_2^T W_0^T)$$

$$W_j = \sum_{j=1}^{k} W_{k-j} B_j (W_j^T = \sum_{j=1}^{k} B_j^T W_{k-j}^T)$$

Value

VMA coefficient of k(lag-max + 1) x k dimension

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

var_bayes	*Fitting Bayesian VAR with Coefficient and Covariance Prior*

Description

[Maturing] This function fits BVAR. Covariance term can be homoskedastic or heteroskedastic (stochastic volatility). It can have Minnesota, SSVS, and Horseshoe prior.

Usage

```
var_bayes(
  y,
  p,
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvar(),
  cov_spec = set_ldlt(),
  intercept = set_intercept(),
  include_mean = TRUE,
  minnesota = TRUE,
  save_init = FALSE,
  convergence = NULL,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvarldlt'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvarldlt'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`p`	VAR lag
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A BVAR model specification by `set_bvar()`, `set_ssvs()`, or `set_horseshoe()`.
`cov_spec`	**[Experimental]** SV specification by `set_sv()`.
`intercept`	**[Experimental]** Prior for the constant term by `set_intercept()`.

include_mean	Add constant term (Default: TRUE) or not (FALSE)
minnesota	Apply cross-variable shrinkage structure (Minnesota-way). By default, TRUE.
save_init	Save every record starting from the initial values (TRUE). By default, exclude the initial values in the record (FALSE), even when num_burn = 0 and thinning = 1. If num_burn > 0 or thinning != 1, this option is ignored.
convergence	Convergence threshold for rhat < convergence. By default, NULL which means no warning.
verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvarldlt object
digits	digit option to print
...	not used

Details

Cholesky stochastic volatility modeling for VAR based on

$$\Sigma_t^{-1} = L^T D_t^{-1} L$$

, and implements corrected triangular algorithm for Gibbs sampler.

Value

var_bayes() returns an object named bvarsv class.

coefficients Posterior mean of coefficients.

chol_posterior Posterior mean of contemporaneous effects.

param Every set of MCMC trace.

param_names Name of every parameter.

group Indicators for group.

num_group Number of groups.

df Numer of Coefficients: 3m + 1 or 3m

p VAR lag

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VHAR_SSVS_SV, VHAR_Horseshoe_SV, or VHAR_minnesota-part_SV

type include constant term (const) or not (none)

spec Coefficients prior specification

sv log volatility prior specification

intercept Intercept prior specification

init Initial values

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

y0 Y_0

design X_0

y Raw input

If it is SSVS or Horseshoe:

pip Posterior inclusion probabilities.

References

Carriero, A., Chan, J., Clark, T. E., & Marcellino, M. (2022). *Corrigendum to "Large Bayesian vector autoregressions with stochastic volatility and non-conjugate priors" [J. Econometrics 212 (1)(2019) 137-154]*. Journal of Econometrics, 227(2), 506-512.

Chan, J., Koop, G., Poirier, D., & Tobias, J. (2019). *Bayesian Econometric Methods (2nd ed., Econometric Exercises)*. Cambridge: Cambridge University Press.

Cogley, T., & Sargent, T. J. (2005). *Drifts and volatilities: monetary policies and outcomes in the post WWII US*. Review of Economic Dynamics, 8(2), 262-302.

Gruber, L., & Kastner, G. (2022). *Forecasting macroeconomic data with Bayesian VARs: Sparse or dense? It depends!* arXiv.

Huber, F., Koop, G., & Onorante, L. (2021). *Inducing Sparsity and Shrinkage in Time-Varying Parameter Models*. Journal of Business & Economic Statistics, 39(3), 669-683.

Korobilis, D., & Shimizu, K. (2022). *Bayesian Approaches to Shrinkage and Sparse Estimation*. Foundations and Trends® in Econometrics, 11(4), 230-354.

Ray, P., & Bhattacharya, A. (2018). *Signal Adaptive Variable Selector for the Horseshoe Prior*. arXiv.

`var_lm` *Fitting Vector Autoregressive Model of Order p Model*

Description

This function fits VAR(p) using OLS method.

Usage

```
var_lm(y, p = 1, include_mean = TRUE, method = c("nor", "chol", "qr"))

## S3 method for class 'varlse'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

## S3 method for class 'varlse'
AIC(object, ...)

## S3 method for class 'varlse'
BIC(object, ...)

is.varlse(x)

is.bvharmod(x)

## S3 method for class 'varlse'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`p`	Lag of VAR (Default: 1)
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`method`	Method to solve linear equation system. (`nor`: normal equation (default), `chol`: Cholesky, and `qr`: HouseholderQR)
`x`	A `varlse` object
`digits`	digit option to print
`...`	not used
`object`	A `varlse` object

Details

This package specifies VAR(p) model as

$$Y_t = A_1 Y_{t-1} + \cdots + A_p Y_{t-p} + c + \epsilon_t$$

If `include_type = TRUE`, there is constant term. The function estimates every coefficient matrix.

Consider the response matrix Y_0. Let T be the total number of sample, let m be the dimension of the time series, let p be the order of the model, and let $n = T - p$. Likelihood of VAR(p) has

$$Y_0 \mid B, \Sigma_e \sim MN(X_0 B, I_s, \Sigma_e)$$

where X_0 is the design matrix, and MN is matrix normal distribution.

Then log-likelihood of vector autoregressive model family is specified by

$$\log p(Y_0 \mid B, \Sigma_e) = -\frac{nm}{2} \log 2\pi - \frac{n}{2} \log \det \Sigma_e - \frac{1}{2} tr((Y_0 - X_0 B)\Sigma_e^{-1}(Y_0 - X_0 B)^T)$$

In addition, recall that the OLS estimator for the matrix coefficient matrix is the same as MLE under the Gaussian assumption. MLE for Σ_e has different denominator, n.

$$\hat{B} = \hat{B}^{LS} = \hat{B}^{ML} = (X_0^T X_0)^{-1} X_0^T Y_0$$

$$\hat{\Sigma}_e = \frac{1}{s-k}(Y_0 - X_0\hat{B})^T(Y_0 - X_0\hat{B})$$

$$\tilde{\Sigma}_e = \frac{1}{s}(Y_0 - X_0\hat{B})^T(Y_0 - X_0\hat{B}) = \frac{s-k}{s}\hat{\Sigma}_e$$

Let $\tilde{\Sigma}_e$ be the MLE and let $\hat{\Sigma}_e$ be the unbiased estimator (`covmat`) for Σ_e. Note that

$$\tilde{\Sigma}_e = \frac{n-k}{n}\hat{\Sigma}_e$$

Then

$$AIC(p) = \log \det \Sigma_e + \frac{2}{n}(\text{number of freely estimated parameters})$$

where the number of freely estimated parameters is mk, i.e. pm^2 or $pm^2 + m$.

Let $\tilde{\Sigma}_e$ be the MLE and let $\hat{\Sigma}_e$ be the unbiased estimator (`covmat`) for Σ_e. Note that

$$\tilde{\Sigma}_e = \frac{n-k}{T}\hat{\Sigma}_e$$

Then

$$BIC(p) = \log \det \Sigma_e + \frac{\log n}{n}(\text{number of freely estimated parameters})$$

where the number of freely estimated parameters is pm^2.

Value

`var_lm()` returns an object named `varlse` class. It is a list with the following components:

coefficients Coefficient Matrix

fitted.values Fitted response values

residuals Residuals

covmat LS estimate for covariance matrix

df Numer of Coefficients

p Lag of VAR

m Dimension of the data

obs Sample size used when training = `totobs` - `p`

totobs Total number of the observation

call Matched call

process Process: VAR

type include constant term (`const`) or not (`none`)

design Design matrix

y Raw input

y0 Multivariate response matrix

method Solving method

call Matched call

It is also a `bvharmod` class.

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

Akaike, H. (1969). *Fitting autoregressive models for prediction*. Ann Inst Stat Math 21, 243-247.

Akaike, H. (1971). *Autoregressive model fitting for control*. Ann Inst Stat Math 23, 163-180.

Akaike H. (1974). *A new look at the statistical model identification*. IEEE Transactions on Automatic Control, vol. 19, no. 6, pp. 716-723.

Akaike H. (1998). *Information Theory and an Extension of the Maximum Likelihood Principle*. In: Parzen E., Tanabe K., Kitagawa G. (eds) Selected Papers of Hirotugu Akaike. Springer Series in Statistics (Perspectives in Statistics). Springer, New York, NY.

Gideon Schwarz. (1978). *Estimating the Dimension of a Model*. Ann. Statist. 6 (2) 461 - 464.

See Also

- `summary.varlse()` to summarize VAR model

Examples

```
# Perform the function using etf_vix dataset
fit <- var_lm(y = etf_vix, p = 2)
class(fit)
str(fit)

# Extract coef, fitted values, and residuals
coef(fit)
head(residuals(fit))
head(fitted(fit))
```

VHARtoVMA *Convert VHAR to VMA(infinite)*

Description

Convert VHAR process to infinite vector MA process

Usage

```
VHARtoVMA(object, lag_max)
```

Arguments

`object`	A `vharlse` object
`lag_max`	Maximum lag for VMA

Details

Let VAR(p) be stable and let VAR(p) be $Y_0 = X_0 B + Z$

VHAR is VAR(22) with

$$Y_0 = X_1 B + Z = ((X_0 \tilde{T}^T))\Phi + Z$$

Observe that

$$B = \tilde{T}^T \Phi$$

Value

VMA coefficient of k(lag-max + 1) x k dimension

References

Lütkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*. Springer Publishing.

vhar_bayes *Fitting Bayesian VHAR with Coefficient and Covariance Prior*

Description

[Maturing] This function fits BVHAR. Covariance term can be homoskedastic or heteroskedastic (stochastic volatility). It can have Minnesota, SSVS, and Horseshoe prior.

Usage

```
vhar_bayes(
  y,
  har = c(5, 22),
  num_chains = 1,
  num_iter = 1000,
  num_burn = floor(num_iter/2),
  thinning = 1,
  bayes_spec = set_bvhar(),
  cov_spec = set_ldlt(),
  intercept = set_intercept(),
  include_mean = TRUE,
  minnesota = c("longrun", "short", "no"),
  save_init = FALSE,
  convergence = NULL,
  verbose = FALSE,
  num_thread = 1
)

## S3 method for class 'bvharldlt'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'bvharldlt'
knit_print(x, ...)
```

Arguments

`y`	Time series data of which columns indicate the variables
`har`	Numeric vector for weekly and monthly order. By default, `c(5, 22)`.
`num_chains`	Number of MCMC chains
`num_iter`	MCMC iteration number
`num_burn`	Number of burn-in (warm-up). Half of the iteration is the default choice.
`thinning`	Thinning every thinning-th iteration
`bayes_spec`	A BVHAR model specification by `set_bvhar()` (default) `set_weight_bvhar()`, `set_ssvs()`, or `set_horseshoe()`.
`cov_spec`	**[Experimental]** SV specification by `set_sv()`.
`intercept`	**[Experimental]** Prior for the constant term by `set_intercept()`.
`include_mean`	Add constant term (Default: `TRUE`) or not (`FALSE`)
`minnesota`	Apply cross-variable shrinkage structure (Minnesota-way). Two type: `short` type and `longrun` (default) type. You can also set `no`.
`save_init`	Save every record starting from the initial values (`TRUE`). By default, exclude the initial values in the record (`FALSE`), even when `num_burn = 0` and `thinning = 1`. If `num_burn > 0` or `thinning != 1`, this option is ignored.
`convergence`	Convergence threshold for rhat < convergence. By default, `NULL` which means no warning.

verbose	Print the progress bar in the console. By default, FALSE.
num_thread	Number of threads
x	bvharldlt object
digits	digit option to print
...	not used

Details

Cholesky stochastic volatility modeling for VHAR based on

$$\Sigma_t^{-1} = L^T D_t^{-1} L$$

Value

vhar_bayes() returns an object named bvharsv class. It is a list with the following components:

coefficients Posterior mean of coefficients.

chol_posterior Posterior mean of contemporaneous effects.

param Every set of MCMC trace.

param_names Name of every parameter.

group Indicators for group.

num_group Number of groups.

df Numer of Coefficients: 3m + 1 or 3m

p 3 (The number of terms. It contains this element for usage in other functions.)

week Order for weekly term

month Order for monthly term

m Dimension of the data

obs Sample size used when training = totobs - p

totobs Total number of the observation

call Matched call

process Description of the model, e.g. VHAR_SSVS_SV, VHAR_Horseshoe_SV, or VHAR_minnesota-part_SV

type include constant term (const) or not (none)

spec Coefficients prior specification

sv log volatility prior specification

init Initial values

intercept Intercept prior specification

chain The numer of chains

iter Total iterations

burn Burn-in

thin Thinning

HARtrans VHAR linear transformation matrix

y0 Y_0

design X_0

y Raw input

If it is SSVS or Horseshoe:

pip Posterior inclusion probabilities.

References

Kim, Y. G., and Baek, C. (2024). *Bayesian vector heterogeneous autoregressive modeling*. Journal of Statistical Computation and Simulation, 94(6), 1139-1157.

Kim, Y. G., and Baek, C. (n.d.). Working paper.

`vhar_lm` *Fitting Vector Heterogeneous Autoregressive Model*

Description

This function fits VHAR using OLS method.

Usage

```
vhar_lm(
  y,
  har = c(5, 22),
  include_mean = TRUE,
  method = c("nor", "chol", "qr")
)

## S3 method for class 'vharlse'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

## S3 method for class 'vharlse'
AIC(object, ...)

## S3 method for class 'vharlse'
BIC(object, ...)

is.vharlse(x)

## S3 method for class 'vharlse'
knit_print(x, ...)
```

Arguments

y	Time series data of which columns indicate the variables
har	Numeric vector for weekly and monthly order. By default, c(5, 22).
include_mean	Add constant term (Default: TRUE) or not (FALSE)
method	Method to solve linear equation system. (nor: normal equation (default), chol: Cholesky, and qr: HouseholderQR)
x	A vharlse object
digits	digit option to print
...	not used
object	A vharlse object

Details

For VHAR model

$$Y_t = \Phi^{(d)} Y_{t-1} + \Phi^{(w)} Y_{t-1}^{(w)} + \Phi^{(m)} Y_{t-1}^{(m)} + \epsilon_t$$

the function gives basic values.

Value

vhar_lm() returns an object named vharlse class. It is a list with the following components:

coefficients Coefficient Matrix
fitted.values Fitted response values
residuals Residuals
covmat LS estimate for covariance matrix
df Numer of Coefficients
m Dimension of the data
obs Sample size used when training = totobs - month
y0 Multivariate response matrix
p 3 (The number of terms. vharlse contains this element for usage in other functions.)
week Order for weekly term
month Order for monthly term
totobs Total number of the observation
process Process: VHAR
type include constant term (const) or not (none)
HARtrans VHAR linear transformation matrix
design Design matrix of VAR(month)
y Raw input
method Solving method
call Matched call

It is also a bvharmod class.

References

Baek, C. and Park, M. (2021). *Sparse vector heterogeneous autoregressive modeling for realized volatility*. J. Korean Stat. Soc. 50, 495-510.

Bubák, V., Kočenda, E., & Žikeš, F. (2011). *Volatility transmission in emerging European foreign exchange markets*. Journal of Banking & Finance, 35(11), 2829-2841.

Corsi, F. (2008). *A Simple Approximate Long-Memory Model of Realized Volatility*. Journal of Financial Econometrics, 7(2), 174-196.

See Also

- `coef.vharlse()`, `residuals.vharlse()`, and `fitted.vharlse()`
- `summary.vharlse()` to summarize VHAR model

Examples

```
# Perform the function using etf_vix dataset
fit <- vhar_lm(y = etf_vix)
class(fit)
str(fit)

# Extract coef, fitted values, and residuals
coef(fit)
head(residuals(fit))
head(fitted(fit))
```

Index

∗ **datasets**
etf_vix, 48

AIC.bvarflat *(bvar_flat)*, 10
AIC.bvarmn *(bvar_minnesota)*, 14
AIC.bvharmn *(bvhar_minnesota)*, 24
AIC.varlse *(var_lm)*, 116
AIC.vharlse *(vhar_lm)*, 123
autolayer.predbvhar *(autoplot.predbvhar)*, 6
autoplot.bvhardynsp, 4
autoplot.bvharirf, 4
autoplot.bvharsp, 5
autoplot.normaliw, 6
autoplot.predbvhar, 6
autoplot.summary.bvharsp, 7
autoplot.summary.normaliw, 8

bayesplot::mcmc_areas(), *5*, *8*
bayesplot::mcmc_dens(), *5*, *8*
bayesplot::mcmc_trace(), *5*, *8*
BIC.bvarflat *(bvar_flat)*, 10
BIC.bvarmn *(bvar_minnesota)*, 14
BIC.bvharmn *(bvhar_minnesota)*, 24
BIC.varlse *(var_lm)*, 116
BIC.vharlse *(vhar_lm)*, 123
bound_bvhar, 9
bound_bvhar(), *33*
bvar_flat, 10
bvar_flat(), *82*
bvar_horseshoe, 12
bvar_minnesota, 14
bvar_minnesota(), *33*, *82*
bvar_ssvs, 17
bvar_ssvs(), *91*
bvar_sv, 20
bvhar_horseshoe, 22
bvhar_minnesota, 24
bvhar_minnesota(), *33*, *82*
bvhar_ssvs, 27
bvhar_ssvs(), *91*
bvhar_sv, 30

choose_bayes, 32
choose_bayes(), *33*
choose_bvar, 33
choose_bvar(), *33*
choose_bvhar *(choose_bvar)*, 33
choose_ssvs, 35
choose_ssvs(), *18*, *28*
choose_var, 37
class, *9*, *11*, *13*, *16*, *18*, *21*, *23*, *26*, *29*, *31*, *33*, *35*, *51*, *53*, *61*, *70*, *82*, *109*, *111*, *112*, *115*, *118*, *122*, *124*
coef, 37
coef(), *103*, *104*
coef.bvarflat(), *12*
coef.vharlse(), *125*
compute_dic, 38
compute_logml, 39
conf_fdr, 41
conf_fnr, 42
conf_fscore, 43
conf_prec, 44
conf_recall, 45
confusion, 40
confusion(), *41–45*

divide_ts, 46
divide_ts(), *51*, *53*
dynamic_spillover, 46

etf_vix, 48

fitted, 49
fitted.bvarflat(), *12*
fitted.vharlse(), *125*
foreach::foreach(), *37*
forecast_expand, 50
forecast_roll, 52

FPE, 54
fromse, 55

geom_eval, 56
gg_loss, 56
ggplot2::facet_grid(), 7
ggplot2::facet_wrap(), 4, 7, 57
ggplot2::geom_hline(), 57
ggplot2::geom_path(), 5, 7, 56, 57
ggplot2::geom_tile(), 7
ggplot2::scale_colour_viridis_d, 7, 57
ggplot2::scale_fill_viridis_d(), 7, 57

HQ, 57

init_ssvs, 59
init_ssvs(), 18, 19, 28, 29
irf (irf.varlse), 60
irf(), 5
irf.varlse, 60
is.boundbvharemp (bound_bvhar), 9
is.bvarflat (bvar_flat), 10
is.bvarmn (bvar_minnesota), 14
is.bvharcv (forecast_roll), 52
is.bvharemp (choose_bvar), 33
is.bvharirf (irf.varlse), 60
is.bvharmn (bvhar_minnesota), 24
is.bvharmod (var_lm), 116
is.bvharpriorspec (set_lambda), 86
is.bvharspec (set_bvar), 81
is.covspec (set_ldlt), 88
is.dlspec (set_dl), 84
is.horseshoespec (set_horseshoe), 85
is.interceptspec (set_intercept), 86
is.ldltspec (set_ldlt), 88
is.ngspec (set_ng), 89
is.predbvhar (predict), 68
is.ssvsinit (init_ssvs), 59
is.ssvsinput (set_ssvs), 90
is.stable, 62
is.svspec (set_ldlt), 88
is.varlse (var_lm), 116
is.vharlse (vhar_lm), 123

knit_print.boundbvharemp (bound_bvhar), 9
knit_print.bvarflat (bvar_flat), 10
knit_print.bvarhm (bvar_minnesota), 14
knit_print.bvarhs (bvar_horseshoe), 12
knit_print.bvarldlt (var_bayes), 114
knit_print.bvarmn (bvar_minnesota), 14
knit_print.bvarssvs (bvar_ssvs), 17
knit_print.bvarsv (bvar_sv), 20
knit_print.bvharcv (forecast_roll), 52
knit_print.bvhardynsp (dynamic_spillover), 46
knit_print.bvharemp (choose_bvar), 33
knit_print.bvharhm (bvhar_minnesota), 24
knit_print.bvharhs (bvhar_horseshoe), 22
knit_print.bvharirf (irf.varlse), 60
knit_print.bvharldlt (vhar_bayes), 120
knit_print.bvharmn (bvhar_minnesota), 24
knit_print.bvharpriorspec (set_lambda), 86
knit_print.bvharspec (set_bvar), 81
knit_print.bvharspillover (spillover), 105
knit_print.bvharssvs (bvhar_ssvs), 27
knit_print.bvharsv (bvhar_sv), 30
knit_print.horseshoespec (set_horseshoe), 85
knit_print.interceptspec (set_intercept), 86
knit_print.predbvhar (predict), 68
knit_print.ssvsinit (init_ssvs), 59
knit_print.ssvsinput (set_ssvs), 90
knit_print.summary.bvharsp (print.summary.bvharsp), 73
knit_print.summary.normaliw (summary.normaliw), 108
knit_print.summary.varlse (summary.varlse), 110
knit_print.summary.vharlse (summary.vharlse), 111
knit_print.varlse (var_lm), 116
knit_print.vharlse (vhar_lm), 123

logLik.bvarflat (bvar_flat), 10
logLik.bvarmn (bvar_minnesota), 14
logLik.bvharmn (bvhar_minnesota), 24
logLik.varlse (var_lm), 116
logLik.vharlse (vhar_lm), 123

mae, 63
mae(), 57
mape, 64
mape(), 57
mase, 65

mase(), *57*
matrix, *38*, *50*, *76*
mrae, 66
mse, 67
mse(), *57*

optimParallel::optimParallel(), *15*, *25*, *33–35*

posterior::draws_df, *13*, *19*, *23*, *29*
predict, 68
predict.bvarflat(), *12*
print.boundbvharemp *(*bound_bvhar*)*, 9
print.bvarflat *(*bvar_flat*)*, 10
print.bvarhm *(*bvar_minnesota*)*, 14
print.bvarhs *(*bvar_horseshoe*)*, 12
print.bvarldlt *(*var_bayes*)*, 114
print.bvarmn *(*bvar_minnesota*)*, 14
print.bvarssvs *(*bvar_ssvs*)*, 17
print.bvarsv *(*bvar_sv*)*, 20
print.bvharcv *(*forecast_roll*)*, 52
print.bvhardynsp *(*dynamic_spillover*)*, 46
print.bvharemp *(*choose_bvar*)*, 33
print.bvharhm *(*bvhar_minnesota*)*, 24
print.bvharhs *(*bvhar_horseshoe*)*, 22
print.bvharirf *(*irf.varlse*)*, 60
print.bvharldlt *(*vhar_bayes*)*, 120
print.bvharmn *(*bvhar_minnesota*)*, 24
print.bvharpriorspec *(*set_lambda*)*, 86
print.bvharspec *(*set_bvar*)*, 81
print.bvharspillover *(*spillover*)*, 105
print.bvharssvs *(*bvhar_ssvs*)*, 27
print.bvharsv *(*bvhar_sv*)*, 30
print.covspec *(*set_ldlt*)*, 88
print.dlspec *(*set_dl*)*, 84
print.horseshoespec *(*set_horseshoe*)*, 85
print.interceptspec *(*set_intercept*)*, 86
print.ngspec *(*set_ng*)*, 89
print.predbvhar *(*predict*)*, 68
print.ssvsinit *(*init_ssvs*)*, 59
print.ssvsinput *(*set_ssvs*)*, 90
print.summary.bvharsp, 73
print.summary.normaliw *(*summary.normaliw*)*, 108
print.summary.varlse *(*summary.varlse*)*, 110
print.summary.vharlse *(*summary.vharlse*)*, 111
print.varlse *(*var_lm*)*, 116
print.vharlse *(*vhar_lm*)*, 123

relmae, 74
relspne, 75
residuals, 76
residuals.bvarflat(), *12*
residuals.vharlse(), *125*
rmafe, 77
rmape, 78
rmase, 79
rmsfe, 80

set_bvar, 81
set_bvar(), *9*, *15*, *17*, *20*, *35*, *87*, *96, 97*, *114*
set_bvar_flat *(*set_bvar*)*, 81
set_bvar_flat(), *11, 12*
set_bvhar *(*set_bvar*)*, 81
set_bvhar(), *9*, *25–27*, *30*, *35*, *87*, *99, 100*, *121*
set_dl, 84
set_horseshoe, 85
set_horseshoe(), *13*, *20*, *23, 24*, *30*, *114*, *121*
set_intercept, 86
set_intercept(), *20*, *31*, *114*, *121*
set_lambda, 86
set_lambda(), *82, 83*
set_ldlt, 88
set_ldlt(), *88*
set_ng, 89
set_psi *(*set_lambda*)*, 86
set_psi(), *82, 83*
set_ssvs, 90
set_ssvs(), *18–20*, *28–30*, *101*, *114*, *121*
set_sv *(*set_ldlt*)*, 88
set_sv(), *20*, *30*, *88*, *114*, *121*
set_weight_bvhar *(*set_bvar*)*, 81
set_weight_bvhar(), *9*, *25–27*, *30*, *35*, *87*, *99, 100*, *121*
sim_gig, 93
sim_horseshoe_var, 94
sim_horseshoe_vhar *(*sim_horseshoe_var*)*, 94
sim_iw, 95
sim_matgaussian, 95
sim_mncoef, 96
sim_mniw, 97
sim_mnormal, 98
sim_mnvhar_coef, 99
sim_mvt, 100

sim_ssvs_var, 101
sim_ssvs_vhar (sim_ssvs_var), 101
sim_var, 103
sim_vhar, 104
spillover, 105
spne, 106
stableroot, 107
stats::AIC(), *58*
stats::approx(), *48*
stats::coef(), *37*
stats::fitted(), *49*
stats::optim(), *32, 33*, *35*, *87*
stats::residuals(), *76*
summary.bvharsp (print.summary.bvharsp), 73
summary.hsmod (print.summary.bvharsp), 73
summary.ngmod (print.summary.bvharsp), 73
summary.normaliw, 108
summary.normaliw(), *17*, *27*
summary.ssvsmod (print.summary.bvharsp), 73
summary.varlse, 110
summary.varlse(), *119*
summary.vharlse, 111
summary.vharlse(), *125*

var_bayes, 114
var_lm, 116
var_lm(), *103, 104*
VARtoVMA, 113
VARtoVMA(), *62*
vhar_bayes, 120
vhar_bayes(), *30*
vhar_lm, 123
VHARtoVMA, 120
VHARtoVMA(), *62*