# Package ‘BVAR’

September 27, 2020

**Type** Package

**Title** Hierarchical Bayesian Vector Autoregression

**Version** 1.0.1

**Date** 2020-09-26

**Author** Nikolas Kuschnig [aut, cre] (<https://orcid.org/0000-0002-6642-2543>),
Lukas Vashold [aut] (<https://orcid.org/0000-0002-3562-3414>),
Michael McCracken [dtc],
Serena Ng [dtc]

**Maintainer** Nikolas Kuschnig <nikolas.kuschnig@wu.ac.at>

**Description** Estimation of hierarchical Bayesian vector autoregressive models.
Implements hierarchical prior selection for conjugate priors in the fashion
of Giannone, Lenza & Primiceri (2015) <doi:10.1162/REST_a_00483>. Functions
to compute and identify impulse responses, calculate forecasts,
forecast error variance decompositions and scenarios are available.
Several methods to print, plot and summarise results facilitate analysis.

**URL** https://github.com/nk027/bvar

**BugReports** https://github.com/nk027/bvar/issues

**Depends** R (>= 3.3.0)

**Imports** mvtnorm, stats, graphics, utils, grDevices

**Suggests** coda, vars, tinytest

**License** GPL-3 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-09-27 04:40:05 UTC

# R topics documented:



---

`BVAR-package` *BVAR: Hierarchical Bayesian vector autoregression*

---

### Description

Estimation of hierarchical Bayesian vector autoregressive models. Implements hierarchical prior selection for conjugate priors in the fashion of Giannone, Lenza & Primiceri (2015) <doi:10.1162/REST_a_00483>. Functions to compute and identify impulse responses, calculate forecasts, forecast error variance decompositions and scenarios are available. Several methods to print, plot and summarise results facilitate analysis.

### References

Giannone, D. and Lenza, M. and Primiceri, G. E. (2015) Prior Selection for Vector Autoregressions. *The Review of Economics and Statistics*, **97:2**, 436-451, https://doi.org/10.1162/REST_a_00483.

Kuschnig, N. and Vashold, L. (2021) BVAR: Bayesian Vector Autoregressions with Hierarchical Prior Selection in R. *Journal of Statistical Software*, **forthcoming**.

bvar *Hierarchical Bayesian vector autoregression*

## Description

Used to estimate hierarchical Bayesian Vector Autoregression (VAR) models in the fashion of Giannone, Lenza and Primiceri (2015). Priors are adjusted and added via `bv_priors`. The Metropolis-Hastings step can be modified with `bv_mh`.

## Usage

```
bvar(
  data,
  lags,
  n_draw = 10000L,
  n_burn = 5000L,
  n_thin = 1L,
  priors = bv_priors(),
  mh = bv_mh(),
  fcast = NULL,
  irf = NULL,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Numeric matrix or dataframe. Note that observations are expected to be ordered from earliest to latest, and variables in the columns. |
| `lags` | Integer scalar. Lag order of the model. |
| `n_draw, n_burn` | Integer scalar. The number of iterations to (a) cycle through and (b) burn at the start. |
| `n_thin` | Integer scalar. Every *n_thin*'th iteration is stored. For a given memory requirement thinning reduces autocorrelation, while increasing effective sample size. |
| `priors` | Object from `bv_priors` with prior settings. Used to adjust the Minnesota prior, add custom dummy priors, and choose hyperparameters for hierarchical estimation. |
| `mh` | Object from `bv_mh` with settings for the Metropolis-Hastings step. Used to tune automatic adjustment of the acceptance rate within the burn-in period, or manually adjust the proposal variance. |
| `fcast` | Object from `bv_fcast` with forecast settings. Options include the horizon and settings for conditional forecasts i.e. scenario analysis. May also be calculated ex-post using `predict.bvar`. |
| `irf` | Object from `bv_irf` with settings for the calculation of impulse responses and forecast error variance decompositions. Options include the horizon and different identification schemes. May also be calculated ex-post using `irf.bvar`. |

| | |
|---|---|
| `verbose` | Logical scalar. Whether to print intermediate results and progress. |
| `...` | Not used. |

## Details

The model can be expressed as:

$$y_t = a_0 + A_1 y_{t-1} + ... + A_p y_{t-p} + \epsilon_t$$

See Kuschnig and Vashold (2019) and Giannone, Lenza and Primiceri (2015) for further information. Methods for a `bvar` object and its derivatives can be used to:

- predict and analyse scenarios;
- evaluate shocks and the variance of forecast errors;
- visualise forecasts and impulse responses, parameters and residuals;
- retrieve coefficents and the variance-covariance matrix;
- calculate fitted and residual values;

Note that these methods generally work by calculating quantiles from the posterior draws. The full posterior may be retrieved directly from the objects. The function `str` can be very helpful for this.

## Value

Returns a list of class `bvar` with the following elements:

- `beta` - Numeric array with draws from the posterior of the VAR coefficients. Also see `coef.bvar`.
- `sigma` - Numeric array with draws from the posterior of the variance-covariance matrix. Also see `vcov.bvar`.
- `hyper` - Numeric matrix with draws from the posterior of the hierarchically treated hyperparameters.
- `ml` - Numeric vector with the marginal likelihood (with respect to the hyperparameters), that determines acceptance probability.
- `optim` - List with outputs of `optim`, which is used to find starting values for the hyperparameters.
- `prior` - Prior settings from `bv_priors`.
- `call` - Call to the function. See `match.call`.
- `meta` - List with meta information. Includes the number of variables, accepted draws, number of iterations, and data.
- `variables` - Character vector with the column names of *data*. If missing, variables are named iteratively.
- `explanatories` - Character vector with names of explanatory variables. Formatting is akin to: `"FEDFUNDS-lag1"`.
- `fcast` - Forecasts from `predict.bvar`.
- `irf` - Impulse responses from `irf.bvar`.

### Author(s)

Nikolas Kuschnig, Lukas Vashold

### References

Giannone, D. and Lenza, M. and Primiceri, G. E. (2015) Prior Selection for Vector Autoregressions. *The Review of Economics and Statistics*, **97:2**, 436-451, https://doi.org/10.1162/REST_a_00483.

Kuschnig, N. and Vashold, L. (2021) BVAR: Bayesian Vector Autoregressions with Hierarchical Prior Selection in R. *Journal of Statistical Software*, **forthcoming**.

### See Also

bv_priors; bv_mh; bv_fcast; bv_irf; predict.bvar; irf.bvar; plot.bvar;

### Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Calculate and store forecasts and impulse responses
predict(x) <- predict(x, horizon = 8)
irf(x) <- irf(x, horizon = 8, fevd = FALSE)

## Not run:
# Check convergence of the hyperparameters with a trace and density plot
plot(x)
# Plot forecasts and impulse responses
plot(predict(x))
plot(irf(x))
# Check coefficient values and variance-covariance matrix
summary(x)

## End(Not run)
```

---

## bv_dummy *Dummy prior settings*

---

### Description

Allows the creation of dummy observation priors for bv_priors. See the Details section for information on common dummy priors.

### Usage

```
bv_dummy(mode = 1, sd = 1, min = 0.0001, max = 5, fun)

bv_soc(mode = 1, sd = 1, min = 0.0001, max = 50)

bv_sur(mode = 1, sd = 1, min = 0.0001, max = 50)
```

### Arguments

| | |
|---|---|
| `mode` | Numeric scalar. Mode / standard deviation of the parameter. Note that the *mode* of *psi* is set automatically by default, and would need to be provided as vector. |
| `sd` | Numeric scalar. Mode / standard deviation of the parameter. Note that the *mode* of *psi* is set automatically by default, and would need to be provided as vector. |
| `min` | Numeric scalar. Minimum / maximum allowed value. Note that for *psi* these are set automatically or need to provided as vectors. |
| `max` | Numeric scalar. Minimum / maximum allowed value. Note that for *psi* these are set automatically or need to provided as vectors. |
| `fun` | Function taking *Y*, *lags* and the prior's parameter *par* to generate and return a named list with elements *X* and *Y* (numeric matrices). |

### Details

Dummy priors are often used to "reduce the importance of the deterministic component implied by VARs estimated conditioning on the initial observations" (Giannone, Lenza and Primiceri, 2015, p. 440). One such prior is the sum-of-coefficients (SOC) prior, which imposes the notion that a no-change forecast is optimal at the beginning of a time series. Its key parameter $\mu$ controls the tightness - i.e. for low values the model is pulled towards a form with as many unit roots as variables and no cointegration. Another such prior is the single-unit-root (SUR) prior, that allows for cointegration relationships in the data. It pushes variables either towards their unconditional mean or towards the presence of at least one unit root. These priors are implemented via Theil mixed estimation, i.e. by adding dummy-observations on top of the data matrix. They are available via the functions `bv_soc` and `bv_sur`.

### Value

Returns a named list of class `bv_dummy` for `bv_priors`.

### Functions

- `bv_soc`: Sum-of-coefficients dummy prior
- `bv_sur`: Single-unit-root dummy prior

### References

Giannone, D. and Lenza, M. and Primiceri, G. E. (2015) Prior Selection for Vector Autoregressions. *The Review of Economics and Statistics*, **97:2**, 436-451, https://doi.org/10.1162/REST_a_00483.

## See Also

bv_priors; bv_minnesota

## Examples

```
# Create a sum-of-coefficients prior
add_soc <- function(Y, lags, par) {
  soc <- if(lags == 1) {diag(Y[1, ]) / par} else {
    diag(colMeans(Y[1:lags, ])) / par
  }
  Y_soc <- soc
  X_soc <- cbind(rep(0, ncol(Y)), matrix(rep(soc, lags), nrow = ncol(Y)))

  return(list("Y" = Y_soc, "X" = X_soc))
}
soc <- bv_dummy(mode = 1, sd = 1, min = 0.0001, max = 50, fun = add_soc)

# Create a single-unit-root prior
add_sur <- function(Y, lags, par) {
  sur <- if(lags == 1) {Y[1, ] / par} else {
    colMeans(Y[1:lags, ]) / par
  }
  Y_sur <- sur
  X_sur <- c(1 / par, rep(sur, lags))

  return(list("Y" = Y_sur, "X" = X_sur))
}

sur <- bv_dummy(mode = 1, sd = 1, min = 0.0001, max = 50, fun = add_sur)

# Add the new custom dummy priors
bv_priors(hyper = "auto", soc = soc, sur = sur)
```

---

# bv_fcast — *Forecast settings*

---

## Description

Provide forecast settings to predict.bvar. Allows adjusting the horizon of forecasts, and for setting up conditional forecasts. See the Details section for further information.

## Usage

```
bv_fcast(horizon = 12, cond_path = NULL, cond_vars = NULL)
```

## Arguments

- `horizon` — Integer scalar. Horizon for which to compute forecasts.
- `cond_path` — Optional numeric vector or matrix used for conditional forecasts. Supply variable path(s) on which forecasts are conditioned on. Unrestricted future realisations should be filled with `NA`. Note that not all variables can be restricted at the same time.
- `cond_vars` — Optional character or numeric vector. Used to subset *cond_path* to specific variable(s) via name or position. Not needed when *cond_path* is constructed for all variables.

## Details

Conditional forecasts are calculated using the algorithm by Waggoner and Zha (1999). They are set up by imposing a path on selected variables.

## Value

Returns a named list of class `bv_fcast` with options for `bvar` or `predict.bvar`.

## References

Waggoner, D. F., & Zha, T. (1999). Conditional Forecasts in Dynamic Multivariate Models. *Review of Economics and Statistics*, **81:4**, 639-651, https://doi.org/10.1162/003465399558508.

## See Also

`predict.bvar`; `plot.bvar_fcast`

## Examples

```
# Set forecast-horizon to 20 time periods for unconditional forecasts
bv_fcast(horizon = 20)

# Define a path for the second variable (in the initial six periods).
bv_fcast(cond_path = c(1, 1, 1, 1, 1, 1), cond_var = 2)

# Constrain the paths of the first and third variables.
paths <- matrix(NA, nrow = 10, ncol = 2)
paths[1:5, 1] <- 1
paths[1:10, 2] <- 2
bv_fcast(cond_path = paths, cond_var = c(1, 3))
```

bv_irf *Impulse response settings and identification*

## Description

Provides settings for the computation of impulse responses to `bvar`, `irf.bvar` or `fevd.bvar`. Allows setting the horizon for which impulse responses should be computed, whether or not forecast error variance decompositions (FEVDs) should be included as well as if and what kind of identification should be used. See the Details section for further information on identification. Identification can be achieved via Cholesky decomposition, sign restrictions (Rubio-Ramirez, Waggoner and Zha, 2010), and zero and sign restrictions (Arias, Rubio-Ramirez and Waggoner, 2018).

## Usage

```
bv_irf(
  horizon = 12,
  fevd = FALSE,
  identification = TRUE,
  sign_restr = NULL,
  sign_lim = 1000
)
```

## Arguments

| | |
|---|---|
| `horizon` | Integer scalar. The horizon for which impulse responses (and FEVDs) should be computed. Note that the first period corresponds to impacts i.e. contemporaneous effects. |
| `fevd` | Logical scalar. Whether or not forecast error variance decompositions should be calculated. |
| `identification` | Logical scalar. Whether or not the shocks used for calculating impulses should be identified. Defaults to `TRUE`, i.e. identification via Cholesky decomposition of the VCOV-matrix unless *sign_restr* is provided. |
| `sign_restr` | Elements inform about expected impacts of certain shocks. Can be either $1$, $-1$ or $0$ depending on whether a positive, a negative or no contemporaneous effect of a certain shock is expected. Elements set to $NA$ indicate that there are no particular expectations for the contemporaneous effects. The default value is `NULL`. Note that in order to be fully identified at least $M*(M-1)/2$ restrictions have to be set and a maximum of $M-j$ zero restrictions can be imposed on the $j$'th column. |
| `sign_lim` | Integer scalar. Maximum number of tries to find suitable matrices to for fitting sign or zero and sign restrictions. |

### Details

Identification can be performed via Cholesky decomposition, sign restrictions, or zero and sign restrictions. The algorithm for generating suitable sign restrictions follows Rubio-Ramirez, Waggoner and Zha (2010), while the one for zero and sign restrictions follows Arias, Rubio-Ramirez and Waggoner (2018). Note the possiblity of finding no suitable zero/sign restrictions.

### Value

Returns a named list of class `bv_irf` with options for `bvar`, `irf.bvar` or `fevd.bvar`.

### References

Rubio-Ramirez, J. F. and Waggoner, D. F. and Zha, T. (2010) Structural Vector Autoregressions: Theory of Identification and Algorithms for Inference. *The Review of Economic Studies*, **77**, 665-696, https://doi.org/10.1111/j.1467-937X.2009.00578.x. Arias, J.E. and Rubio-Ramirez, J. F. and Waggoner, D. F. (2018) Inference Based on Structural Vector Autoregressions Identifiied with Sign and Zero Restrictions: Theory and Applications. *Econometrica*, **86**, 2, 685-720, https://doi.org/10.3982/ECTA14468.

### See Also

`irf.bvar`; `plot.bvar_irf`

### Examples

```
# Set impulse responses to a horizon of 20 time periods and enable FEVD
# (Identification is performed via Cholesky decomposition)
bv_irf(horizon = 20, fevd = TRUE)

# Set up structural impulse responses using sign restrictions
signs <- matrix(c(1, NA, NA, -1, 1, -1, -1, 1, 1), nrow = 3)
bv_irf(sign_restr = signs)

# Set up structural impulse responses using zero and sign restrictions
zero_signs <- matrix(c(1, 0, NA, -1, 1, 0, -1, 1, 1), nrow = 3)
bv_irf(sign_restr = zero_signs)

# Prepare to estimate unidentified impulse responses
bv_irf(identification = FALSE)
```

---

`bv_metropolis` *Metropolis-Hastings settings*

---

### Description

Function to provide settings for the Metropolis-Hastings step in `bvar`. Options include scaling the inverse Hessian that is used to draw parameter proposals and automatic scaling to achieve certain acceptance rates.

## Usage

```
bv_metropolis(
  scale_hess = 0.01,
  adjust_acc = FALSE,
  adjust_burn = 0.75,
  acc_lower = 0.25,
  acc_upper = 0.45,
  acc_change = 0.01
)

bv_mh(
  scale_hess = 0.01,
  adjust_acc = FALSE,
  adjust_burn = 0.75,
  acc_lower = 0.25,
  acc_upper = 0.45,
  acc_change = 0.01
)
```

## Arguments

| | |
|---|---|
| `scale_hess` | Numeric scalar or vector. Scaling parameter, determining the range of hyperparameter draws. Should be calibrated so a reasonable acceptance rate is reached. If provided as vector the length must equal the number of hyperparameters (one per variable for `psi`). |
| `adjust_acc` | Logical scalar. Whether or not to further scale the variability of parameter draws during the burn-in phase. |
| `adjust_burn` | Numeric scalar. How much of the burn-in phase should be used to scale parameter variability. See Details. |
| `acc_lower, acc_upper` | Numeric scalar. Lower (upper) bound of the target acceptance rate. Required if *adjust_acc* is set to `TRUE`. |
| `acc_change` | Numeric scalar. Percent change applied to the Hessian matrix for tuning acceptance rate. Required if *adjust_acc* is set to `TRUE`. |

## Details

Note that adjustment of the acceptance rate by scaling the parameter draw variability can only be done during the burn-in phase, as otherwise the resulting draws do not feature the desirable properties of a Markov chain. After the parameter draws have been scaled, some additional draws should be burnt.

## Value

Returns a named list of class `bv_metropolis` with options for `bvar`.

## Examples

```
# Increase the scaling parameter
bv_mh(scale_hess = 1)

# Turn on automatic scaling of the acceptance rate to [20%, 40%]
bv_mh(adjust_acc = TRUE, acc_lower = 0.2, acc_upper = 0.4)

# Increase the rate of automatic scaling
bv_mh(adjust_acc = TRUE, acc_lower = 0.2, acc_upper = 0.4, acc_change = 0.1)

# Use only 50% of the burn-in phase to adjust scaling
bv_mh(adjust_acc = TRUE, adjust_burn = 0.5)
```

---

bv_minnesota *Minnesota prior settings*

---

## Description

Provide settings for the Minnesota prior to bv_priors. See the Details section for further information.

## Usage

```
bv_minnesota(
  lambda = bv_lambda(),
  alpha = bv_alpha(),
  psi = bv_psi(),
  var = 10000000,
  b = 1
)

bv_mn(
  lambda = bv_lambda(),
  alpha = bv_alpha(),
  psi = bv_psi(),
  var = 10000000,
  b = 1
)

bv_lambda(mode = 0.2, sd = 0.4, min = 0.0001, max = 5)

bv_alpha(mode = 2, sd = 0.25, min = 1, max = 3)

bv_psi(scale = 0.004, shape = 0.004, mode = "auto", min = "auto", max = "auto")
```

### Arguments

| | |
|---|---|
| `lambda` | List constructed via `bv_lambda`. Arguments are *mode*, *sd*, *min* and *max*. May also be provided as a numeric vector of length 4. |
| `alpha` | List constructed via `bv_alpha`. Arguments are *mode*, *sd*, *min* and *max*. High values for *mode* may affect invertibility of the augmented data matrix. May also be provided as a numeric vector of length 4. |
| `psi` | List with elements *scale*, *shape* of the prior as well as *mode* and optionally *min* and *max*. The length of these needs to match the number of variables (i.e. columns) in the data. By default *mode* is set automatically to the square-root of the innovations variance after fitting an $AR(p)$ model to the data. If `arima` fails due to a non-stationary time series the order of integration is incremented by 1. By default *min* / *max* are set to *mode* divided / multiplied by 100. |
| `var` | Numeric scalar with the prior variance on the model's constant. |
| `b` | Numeric scalar, vector or matrix with the prior mean. A scalar is applied to all variables, with a default value of 1. Consider setting it to 0 for growth rates. A vector needs to match the number of variables (i.e. columns) in the data, with a prior mean per variable. If provided, a matrix needs to have a column per variable ($M$), and $M * p + 1$ rows, where $p$ is the number of lags applied. |
| `mode, sd` | Numeric scalar. Mode / standard deviation of the parameter. Note that the *mode* of *psi* is set automatically by default, and would need to be provided as vector. |
| `min, max` | Numeric scalar. Minimum / maximum allowed value. Note that for *psi* these are set automatically or need to provided as vectors. |
| `scale, shape` | Numeric scalar. Scale and shape parameters of a Gamma distribution. |

### Details

Essentially this prior imposes the hypothesis, that individual variables all follow random walk processes. This parsimonious specification typically performs well in forecasts of macroeconomic time series and is often used as a benchmark for evaluating accuracy (Kilian and Lütkepohl, 2017). The key parameter is $\lambda$ (*lambda*), which controls the tightness of the prior. The parameter $\alpha$ (*alpha*) governs variance decay with increasing lag order, while $\psi$ (*psi*) controls the prior's standard deviation on lags of variables other than the dependent. The Minnesota prior is often refined with additional priors, trying to minimise the importance of conditioning on initial observations. See `bv_dummy` for more information on such priors.

### Value

Returns a list of class `bv_minnesota` with options for `bvar`.

### Functions

- `bv_lambda`: Tightness of the Minnesota prior
- `bv_alpha`: Variance decay with increasing lag order
- `bv_psi`: Prior standard deviation on other lags

### References

Kilian, L. and Lütkepohl, H. (2017). *Structural Vector Autoregressive Analysis*. Cambridge University Press, https://doi.org/10.1017/9781108164818

### See Also

bv_priors; bv_dummy

### Examples

```
# Adjust alpha and the Minnesota prior variance.
bv_mn(alpha = bv_alpha(mode = 0.5, sd = 1, min = 1e-12, max = 10), var = 1e6)
# Optionally use a vector as shorthand
bv_mn(alpha = c(0.5, 1, 1e-12, 10), var = 1e6)

# Only adjust lambda's standard deviation
bv_mn(lambda = bv_lambda(sd = 2))

# Provide prior modes for psi (for a VAR with three variables)
bv_mn(psi = bv_psi(mode = c(0.7, 0.3, 0.9)))
```

---

## bv_priors *Prior settings*

---

### Description

Function to provide priors and their parameters to bvar. Used for adjusting the parameters treated as hyperparameters, the Minnesota prior and adding various dummy priors through the ellipsis parameter. Note that treating $\psi$ (*psi*) as a hyperparameter in a model with many variables may lead to very low acceptance rates and thus hinder convergence.

### Usage

```
bv_priors(hyper = "auto", mn = bv_mn(), ...)
```

### Arguments

| | |
|---|---|
| hyper | Character vector. Used to specify the parameters to be treated as hyperparameters. May also be set to "auto" or "full" for an automatic / full subset. Other allowed values are the Minnesota prior's parameters "lambda", "alpha" and "psi" as well as the names of additional dummy priors included via .... |
| mn | List of class "bv_minnesota". Options for the Minnesota prior, set via bv_mn. |
| ... | Optional lists of class bv_dummy with options for dummy priors. **Must be assigned a name in the function call**. Created with bv_dummy. |

### Value

Returns a named list of class bv_priors with options for bvar.

### See Also

bv_mn; bv_dummy

### Examples

```
# Extend the hyperparameters to the full Minnesota prior
bv_priors(hyper = c("lambda", "alpha", "psi"))
# Alternatively
# bv_priors("full")

# Add a dummy prior via `bv_dummy()`

# Re-create the single-unit-root prior
add_sur <- function(Y, lags, par) {
  sur <- if(lags == 1) {Y[1, ] / par} else {
    colMeans(Y[1:lags, ]) / par
  }
  Y_sur <- sur
  X_sur <- c(1 / par, rep(sur, lags))

  return(list("Y" = Y_sur, "X" = X_sur))
}
sur <- bv_dummy(mode = 1, sd = 1, min = 0.0001, max = 50, fun = add_sur)

# Add the new prior
bv_priors(hyper = "auto", sur = sur)
```

## coda — *Methods for* **coda** *Markov chain Monte Carlo objects*

### Description

Methods to convert parameter and/or coefficient draws from bvar to **coda**'s mcmc (or mcmc.list) format for further processing.

### Usage

```
as.mcmc.bvar(
  x,
  vars = NULL,
  vars_response = NULL,
  vars_impulse = NULL,
  chains = list(),
  ...
)

as.mcmc.bvar_chains(
  x,
```

```
  vars = NULL,
  vars_response = NULL,
  vars_impulse = NULL,
  chains = list(),
  ...
)
```

## Arguments

| | |
|---|---|
| x | A bvar object, obtained from bvar. |
| vars | Character vector used to select variables. Elements are matched to hyperparameters or coefficients. Coefficients may be matched based on the dependent variable (by providing the name or position) or the explanatory variables (by providing the name and the desired lag). See the example section for a demonstration. Defaults to NULL, i.e. all hyperparameters. |
| vars_response, vars_impulse | Optional character or integer vectors used to select coefficents. Dependent variables are specified with *vars_response*, explanatory ones with *vars_impulse*. See the example section for a demonstration. |
| chains | List with additional bvar objects. If provided, an object of class mcmc.list is returned. |
| ... | Other parameters for as.mcmc. |

## Value

Returns a **coda** mcmc (or mcmc.list) object.

## See Also

bvar; mcmc; mcmc.list

## Examples

```
library("coda")

# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate two BVARs using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 750L, n_burn = 250L, verbose = FALSE)
y <- bvar(data, lags = 1, n_draw = 750L, n_burn = 250L, verbose = FALSE)

# Convert the hyperparameter lambda
as.mcmc(x, vars = c("lambda"))

# Convert coefficients for the first dependent, use chains in method
as.mcmc(structure(list(x, y), class = "bvar_chains"), vars = "CPIAUCSL")
```

```
# Convert the coefs of variable three's first lag, use in the generic
as.mcmc(x, vars = "FEDFUNDS-lag1", chains = y)

# Convert hyperparameters and constant coefficient values for variable 1
as.mcmc(x, vars = c("lambda", "CPI", "constant"))

# Specify coefficent values to convert in alternative way
as.mcmc(x, vars_impulse = c("FED", "CPI"), vars_response = "UNRATE")
```

---

`coef.bvar` *Coefficient and VCOV methods for Bayesian VARs*

---

## Description

Retrieves coefficient / variance-covariance values from Bayesian VAR models generated with `bvar`. Note that coefficients are available for every stored draw and one may retrieve (a) credible intervals via the *conf_bands* argument, or (2) means via the *type* argument.

## Usage

```
## S3 method for class 'bvar'
coef(
  object,
  type = c("quantile", "mean"),
  conf_bands = 0.5,
  companion = FALSE,
  ...
)

## S3 method for class 'bvar'
vcov(object, type = c("quantile", "mean"), conf_bands = 0.5, ...)
```

## Arguments

| | |
|---|---|
| `object` | A bvar object, obtained from `bvar`. |
| `type` | Character scalar. Whether to return quantile or mean values. Note that *conf_bands* is ignored for mean values. |
| `conf_bands` | Numeric vector of confidence bands to apply. E.g. for bands at 5%, 10%, 90% and 95% set this to `c(0.05,0.1)`. Note that the median, i.e. 0.5 is always included. |
| `companion` | Logical scalar. Whether to retrieve the companion matrix of coefficients. See `companion.bvar`. |
| `...` | Not used. |

### Value

Returns a numeric array of class bvar_coefs or bvar_vcovs at the specified values.

### See Also

bvar; companion.bvar

### Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Get coefficent values at the 10%, 50% and 90% quantiles
coef(x, conf_bands = 0.10)

# Only get the median of the variance-covariance matrix
vcov(x, conf_bands = 0.5)
```

---

## companion *Retrieve companion matrix from a Bayesian VAR*

---

### Description

Calculates the companion matrix for Bayesian VARs generated via bvar.

### Usage

```
companion(object, ...)

## S3 method for class 'bvar'
companion(object, type = c("quantile", "mean"), conf_bands = 0.5, ...)
```

### Arguments

| | |
|---|---|
| object | A bvar object, obtained from bvar. |
| ... | Not used. |
| type | Character scalar. Whether to return quantile or mean values. Note that *conf_bands* is ignored for mean values. |
| conf_bands | Numeric vector of confidence bands to apply. E.g. for bands at 5%, 10%, 90% and 95% set this to c(0.05,0.1). Note that the median, i.e. 0.5 is always included. |

## Value

Returns a numeric array/matrix of class bvar_comp with the VAR's coefficents in companion form at the specified values.

## See Also

bvar; coef.bvar

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Get companion matrices for confidence bands at 10%, 50% and 90%
companion(x, conf_bands = 0.10)
```

---

# density.bvar *Density methods for Bayesian VARs*

---

## Description

Calculates densities of hyperparameters or coefficient draws from Bayesian VAR models generated via bvar. Wraps standard density outputs into a list.

## Usage

```
## S3 method for class 'bvar'
density(x, vars = NULL, vars_response = NULL, vars_impulse = NULL, ...)

## S3 method for class 'bvar_density'
plot(x, mar = c(2, 2, 2, 0.5), mfrow = c(length(x), 1), ...)

independent_index(var, n_vars, lag)
```

## Arguments

| | |
|---|---|
| x | A bvar object, obtained from bvar. |
| vars | Character vector used to select variables. Elements are matched to hyperparameters or coefficients. Coefficients may be matched based on the dependent variable (by providing the name or position) or the explanatory variables (by providing the name and the desired lag). See the example section for a demonstration. Defaults to NULL, i.e. all hyperparameters. |

| | |
|---|---|
| vars_response | Optional character or integer vectors used to select coefficents. Dependent variables are specified with *vars_response*, explanatory ones with *vars_impulse*. See the example section for a demonstration. |
| vars_impulse | Optional character or integer vectors used to select coefficents. Dependent variables are specified with *vars_response*, explanatory ones with *vars_impulse*. See the example section for a demonstration. |
| ... | Fed to density or par. |
| mar | Numeric vector. Margins for par. |
| mfrow | Numeric vector. Rows for par. |
| var, n_vars, lag | Integer scalars. Retrieve the position of lag *lag* of variable *var* given *n_vars* total variables. |

## Value

Returns a list with outputs of density.

## See Also

bvar; density

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Get densities of the hyperparameters
density(x)

# Plot them
plot(density(x))

# Only get the densities associated with dependent variable 1
density(x, vars_response = "CPI")

# Check out the constant's densities
plot(density(x, vars_impulse = 1))

# Get the densities of variable three's first lag
density(x, vars = "FEDFUNDS-lag1")

# Get densities of lambda and the coefficients of dependent variable 2
density(x, vars = c("lambda", "UNRATE"))
```

`fitted.bvar` *Fitted and residual methods for Bayesian VARs*

## Description

Calculates fitted or residual values for Bayesian VAR models generated with `bvar`.

## Usage

```
## S3 method for class 'bvar'
fitted(object, type = c("quantile", "mean"), conf_bands = 0.5, ...)

## S3 method for class 'bvar'
residuals(object, type = c("quantile", "mean"), conf_bands = 0.5, ...)

## S3 method for class 'bvar_resid'
plot(x, vars = NULL, mar = c(2, 2, 2, 0.5), ...)
```

## Arguments

| | |
|---|---|
| `object` | A `bvar` object, obtained from `bvar`. |
| `type` | Character scalar. Whether to return quantile or mean values. Note that *conf_bands* is ignored for mean values. |
| `conf_bands` | Numeric vector of confidence bands to apply. E.g. for bands at 5%, 10%, 90% and 95% set this to `c(0.05,0.1)`. Note that the median, i.e. 0.5 is always included. |
| `...` | Not used. |
| `x` | Object of class `bvar_fitted` / `bvar_resid`. |
| `vars` | Character vector used to select variables. Elements are matched to hyperparameters or coefficients. Coefficients may be matched based on the dependent variable (by providing the name or position) or the explanatory variables (by providing the name and the desired lag). See the example section for a demonstration. Defaults to `NULL`, i.e. all hyperparameters. |
| `mar` | Numeric vector. Margins for `par`. |

## Value

Returns a numeric array of class `bvar_fitted` or `bvar_resid` at the specified values.

## See Also

`bvar`

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Get fitted values and adjust confidence bands to 10%, 50% and 90%
fitted(x, conf_bands = 0.10)

# Get the residuals of variable 1
resid(x, vars = 1)

## Not run:
# Get residuals and plot them
plot(residuals(x))

## End(Not run)
```

---

`fred_qd` *FRED-MD and FRED-QD: Databases for Macroeconomic Research*

---

## Description

FRED-MD and FRED-QD are large macroeconomic databases, containing monthly and quarterly time series that are frequently used in the literature. They are intended to facilitate the reproduction of empirical work and simplify data related tasks. Included datasets are provided as is - transformation codes are available in `system.file("fred_trans.rds",package = "BVAR")`. These can be applied automatically with `fred_transform`.

## Usage

```
fred_qd

fred_md
```

## Format

A `data.frame` object with dates as rownames.

An object of class `data.frame` with 734 rows and 121 columns.

### Details

The versions of FRED-MD and FRED-QD that are provided here are licensed under a modified ODC-BY 1.0 license that can be found in the provided *LICENSE* file. The provided versions are subset to variables that are either in public domain or for which we were given permission to use. For further details see McCracken and Ng (2016) or https://research.stlouisfed.org/econ/mccracken/fred-databases/. We would like to thank Michael McCracken and Serena Ng, Adrienne Brennecke and the Federal Reserve Bank of St. Louis for creating, updating and making available the datasets and many of the contained time series. We also thank all other owners of included time series that permitted their use.

### Source

https://research.stlouisfed.org/econ/mccracken/fred-databases/

### References

McCracken, M. W. and Ng, S. (2016) FRED-MD: A Monthly Database for Macroeconomic Research. *Journal of Business & Economic Statistics*, **34:4**, 574-589, https://doi.org/10.1080/07350015.2015.1086655. McCracken, M. W., & Ng, S. (2020). FRED-QD: A Quarterly Database for Macroeconomic Research **w26872**. National Bureau of Economic Research.

### See Also

fred_transform

---

## fred_transform — *FRED transformation and subset helper*

### Description

Apply transformations given by FRED-MD or FRED-QD and generate rectangular subsets. See fred_qd for information on data and the Details section for information on the transformations. Call without arguments to retrieve available codes / all FRED suggestions.

### Usage

```
fred_transform(
  data,
  type = c("fred_qd", "fred_md"),
  codes,
  na.rm = TRUE,
  lag = 1L,
  scale = 100
)

fred_code(vars, type = c("fred_qd", "fred_md"), table = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | A `data.frame` with FRED-QD or FRED-MD time series. The column names are used to find the correct transformation. |
| `type` | Character scalar. Whether *data* stems from the FRED-QD or the FRED-MD database. |
| `codes` | Integer vector. Transformation code(s) to apply to *data*. Overrides automatic lookup of transformation codes. |
| `na.rm` | Logical scalar. Whether to subset to rows without any `NA` values. A warning is thrown if rows are non-sequential. |
| `lag` | Integer scalar. Number of lags to apply when taking differences. See `diff`. |
| `scale` | Numeric scalar. Scaling to apply to log differences. |
| `vars` | Character vector. Names of the variables to look for. |
| `table` | Logical scalar. Whether to return a table of matching transformation codes instead of just the codes. |

### Details

FRED-QD and FRED-MD include a transformation code for every variable. All codes are provided in `system.file("fred_trans.csv",package = "BVAR")`. The transformation codes are as follows:

1. `1` - no transformation;
2. `2` - first differences - $\Delta x_t$;
3. `3` - second differences - $\Delta^2 x_t$;
4. `4` - log transformation - $\log x_t$;
5. `5` - log differences - $\Delta \log x_t$;
6. `6` - log second differences - $\Delta^2 \log x_t$;
7. `7` - percent change differences - $\Delta x_t / x_{t-1} - 1$;

Note that the transformation codes of FRED-MD and FRED-QD may differ for the same series.

### Value

`fred_transform` returns a `data.frame` object with applied transformations. `fred_code` returns transformation codes, or a `data.frame` of matching transformation codes.

### See Also

`fred_qd`

## Examples

```
# Transform a subset of FRED-QD
fred_transform(fred_qd[, c("GDPC1", "INDPRO", "FEDFUNDS")])

# Get info on transformation codes for unemployment variables
fred_code("UNRATE", table = TRUE)

# Get the transformation code for GDPC1
fred_code("GDPC1", type = "fred_qd")

# Transform all of FRED-MD
## Not run:
fred_transform(fred_md, type = "fred_md")

## End(Not run)
```

---

irf.bvar *Impulse response and forecast error methods for Bayesian VARs*

---

## Description

Retrieves / calculates impulse response functions (IRFs) and/or forecast error variance decompositions (FEVDs) for Bayesian VARs generated via bvar. If the object is already present and no settings are supplied it is simply retrieved, otherwise it will be calculated ex-post. Note that FEVDs require the presence / calculation of IRFs. To store the results you may want to assign the output using the setter function (`irf(x) <-irf(x)`). May also be used to update confidence bands.

## Usage

```
## S3 method for class 'bvar'
irf(x, ..., conf_bands, n_thin = 1L)

## S3 method for class 'bvar'
fevd(x, ..., conf_bands, n_thin = 1L)

irf(x, ...)

irf(x) <- value

fevd(x, ...)

## S3 method for class 'bvar_irf'
summary(object, vars_impulse = NULL, vars_response = NULL, ...)
```

## Arguments

| | |
|---|---|
| `x, object` | A bvar object, obtained from bvar. Summary and print methods take in a `bvar_irf` / `bvar_fevd` object. |

| | |
|---|---|
| `...` | A `bv_irf` object or arguments to be fed into `bv_irf`. Contains settings for the IRFs / FEVDs. |
| `conf_bands` | Numeric vector of confidence bands to apply. E.g. for bands at 5%, 10%, 90% and 95% set this to `c(0.05,0.1)`. Note that the median, i.e. 0.5 is always included. |
| `n_thin` | Integer scalar. Every *n_thin*'th draw in *x* is used to calculate, others are dropped. |
| `value` | A `bvar_irf` object to assign. |
| `vars_impulse, vars_response` | Optional numeric or character vector. Used to subset the summary method's outputs to certain variables by position or name (must be available). Defaults to `NULL`, i.e. all variables. |

## Value

Returns a list of class `bvar_irf` including IRFs and optionally FEVDs at desired confidence bands. The `fevd` method only returns a the nested `bvar_fevd` object. The summary method returns a numeric array of impulse responses at the specified confidence bands.

## See Also

`plot.bvar_irf`; `bv_irf`

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Calculate and store structural IRFs (via Cholesky decomposition)
irf(x) <- irf(x, identification = TRUE)

# Update the confidence bands of the IRFs
irf(x, conf_bands = c(0.01, 0.05, 0.1))

# Compute and store with a longer horizon, no identification and thinning
irf(x) <- irf(x, bv_irf(horizon = 24L, identification = FALSE), n_thin = 10L)

# Recalculate with sign restrictions provided via the ellipsis
irf(x, sign_restr = matrix(c(1, NA, NA, -1, 1, -1, -1, 1, 1), nrow = 3))

# Recalculate with zero and sign restrictions provided via the ellipsis
irf(x, sign_restr = matrix(c(1, 0, 1, NA, 1, 1, -1, -1, 1), nrow = 3))

# Calculate the forecast error variance decomposition
fevd(x)
```

```
# Get a summary of the saved impulse response function
summary(x)

# Limit the summary to responses of variable #2
summary(x, vars_response = 2L)
```

---

logLik.bvar *Log-Likelihood method for Bayesian VARs*

---

## Description

Calculates the log-likelihood of Bayesian VAR models generated with bvar.

## Usage

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

## Arguments

| | |
|---|---|
| object | A bvar object, obtained from bvar. |
| ... | Not used. |

## Value

Returns an object of class logLik.

## See Also

bvar

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Calculate the log-likelihood
logLik(x)
```

## par_bvar *Parallel hierarchical Bayesian vector autoregression*

### Description

Wrapper for bvar to simplify parallel computation via parLapply. Make sure to properly start and stop the provided cluster.

### Usage

```
par_bvar(
  cl,
  n_runs = length(cl),
  data,
  lags,
  n_draw = 10000L,
  n_burn = 5000L,
  n_thin = 1L,
  priors = bv_priors(),
  mh = bv_mh(),
  fcast = NULL,
  irf = NULL
)
```

### Arguments

| | |
|---|---|
| cl | A cluster object obtained from makeCluster. |
| n_runs | The number of parallel runs to calculate. Defaults to the length of *cl*, i.e. the number of registered nodes. |
| data | Numeric matrix or dataframe. Note that observations are expected to be ordered from earliest to latest, and variables in the columns. |
| lags | Integer scalar. Lag order of the model. |
| n_draw | Integer scalar. The number of iterations to (a) cycle through and (b) burn at the start. |
| n_burn | Integer scalar. The number of iterations to (a) cycle through and (b) burn at the start. |
| n_thin | Integer scalar. Every *n_thin*'th iteration is stored. For a given memory requirement thinning reduces autocorrelation, while increasing effective sample size. |
| priors | Object from bv_priors with prior settings. Used to adjust the Minnesota prior, add custom dummy priors, and choose hyperparameters for hierarchical estimation. |
| mh | Object from bv_mh with settings for the Metropolis-Hastings step. Used to tune automatic adjustment of the acceptance rate within the burn-in period, or manually adjust the proposal variance. |

fcast Object from bv_fcast with forecast settings. Options include the horizon and settings for conditional forecasts i.e. scenario analysis. May also be calculated ex-post using predict.bvar.

irf Object from bv_irf with settings for the calculation of impulse responses and forecast error variance decompositions. Options include the horizon and different identification schemes. May also be calculated ex-post using irf.bvar.

## Value

Returns a list of class bvar_chain with bvar objects.

## See Also

bvar; parLapply

## Examples

```
library("parallel")

cl <- makeCluster(2L)

# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# A singular run using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Two parallel runs
y <- par_bvar(cl, n_runs = 2,
  data = data, lags = 1, n_draw = 1000L, n_burn = 200L)

stopCluster(cl)

# Plot lambda for all of the runs
## Not run:
plot(x, type = "full", vars = "lambda", chains = y)

# Convert the hyperparameter lambda to a coda mcmc.list object
coda::as.mcmc(y, vars = "lambda")

## End(Not run)
```

---

plot.bvar *Plotting method for Bayesian VARs*

---

## Description

Method to plot trace and densities of coefficient, hyperparameter and marginal likelihood draws obtained from bvar. Several types of plot are available via the argument *type*, including traces, densities, plots of forecasts and impulse responses.

## Usage

```
## S3 method for class 'bvar'
plot(
  x,
  type = c("full", "trace", "density", "irf", "fcast"),
  vars = NULL,
  vars_response = NULL,
  vars_impulse = NULL,
  chains = list(),
  mar = c(2, 2, 2, 0.5),
  ...
)
```

## Arguments

`x` A bvar object, obtained from bvar.

`type` A string with the type of plot desired. The default option "full" plots both densities and traces.

`vars` Character vector used to select variables. Elements are matched to hyperparameters or coefficients. Coefficients may be matched based on the dependent variable (by providing the name or position) or the explanatory variables (by providing the name and the desired lag). See the example section for a demonstration. Defaults to NULL, i.e. all hyperparameters.

`vars_response, vars_impulse` Optional character or integer vectors used to select coefficents. Dependent variables are specified with *vars_response*, explanatory ones with *vars_impulse*. See the example section for a demonstration.

`chains` List of bvar objects. Contents are then added to trace and density plots to help assessing covergence.

`mar` Numeric vector. Margins for par.

`...` Other graphical parameters for par.

## Value

Returns *x* invisibly.

## See Also

bvar; plot.bvar_fcast; plot.bvar_irf.

### Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Plot full traces and densities
plot(x)

# Only plot the marginal likelihood's trace
plot(x, "trace", "ml")

# Access IRF and forecast plotting functions
plot(x, type = "irf", vars_response = 2)
plot(x, type = "fcast", vars = 2)
```

---

## plot.bvar_fcast *Plotting method for Bayesian VAR predictions*

---

### Description

Plotting method for forecasts obtained from predict.bvar. Forecasts of all or a subset of the available variables can be plotted.

### Usage

```
## S3 method for class 'bvar_fcast'
plot(
  x,
  vars = NULL,
  col = "#737373",
  t_back = 1,
  area = FALSE,
  fill = "#808080",
  variables = NULL,
  orientation = c("vertical", "horizontal"),
  mar = c(2, 2, 2, 0.5),
  ...
)
```

### Arguments

| | |
|---|---|
| x | A bvar_fcast object, obtained from predict.bvar. |
| vars | Optional numeric or character vector. Used to subset the plot to certain variables by position or name (must be available). Defaults to NULL, i.e. all variables. |
| col | Character vector. Colour(s) of the lines delineating credible intervals. Single values will be recycled if necessary. Recycled HEX color codes are varied in transparency if not provided (e.g. "#737373FF"). Lines can be bypassed by setting this to "transparent". |
| t_back | Integer scalar. Number of observed datapoints to plot ahead of the forecast. |
| area | Logical scalar. Whether to fill the credible intervals using polygon. |
| fill | Character vector. Colour(s) to fill the credible intervals with. See *col* for more information. |
| variables | Optional character vector. Names of all variables in the object. Used to subset and title. Taken from x$variables if available. |
| orientation | String indicating the orientation of the plots. Defaults to "v" (i.e. vertical); may be set to "h" (i.e. horizontal). |
| mar | Numeric vector. Margins for par. |
| ... | Other graphical parameters for par. |

### Value

Returns *x* invisibly.

### See Also

bvar; predict.bvar

### Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Store predictions ex-post
predict(x) <- predict(x)

# Plot forecasts for all available variables
plot(predict(x))

# Subset to variables in positions 1 and 3 via their name
plot(predict(x), vars = c("CPI", "FED"))
```

```
# Subset via position, increase the plotted forecast horizon and past data
plot(predict(x, horizon = 20), vars = c(1, 3), t_back = 10)

# Adjust confidence bands and the plot's orientation
plot(predict(x, conf_bands = 0.25), orientation = "h")

# Draw areas inbetween the confidence bands and skip drawing lines
plot(predict(x), col = "transparent", area = TRUE)

# Plot a conditional forecast (with a constrained second variable).
plot(predict(x, cond_path = c(1, 1, 1, 1, 1, 1), cond_var = 2))
```

---

`plot.bvar_irf` *Plotting method for Bayesian VAR impulse responses*

---

## Description

Plotting method for impulse responses obtained from `irf.bvar`. Impulse responses of all or a subset of the available variables can be plotted.

## Usage

```
## S3 method for class 'bvar_irf'
plot(
  x,
  vars_response = NULL,
  vars_impulse = NULL,
  col = "#737373",
  area = FALSE,
  fill = "#808080",
  variables = NULL,
  mar = c(2, 2, 2, 0.5),
  ...
)
```

## Arguments

`x` A bvar_irf object, obtained from `irf.bvar`.

`vars_impulse, vars_response`
Optional numeric or character vector. Used to subset the plot's impulses / responses to certain variables by position or name (must be available). Defaults to `NULL`, i.e. all variables.

`col` Character vector. Colour(s) of the lines delineating credible intervals. Single values will be recycled if necessary. Recycled HEX color codes are varied in transparency if not provided (e.g. "#737373FF"). Lines can be bypassed by setting this to `"transparent"`.

| | |
|---|---|
| `area` | Logical scalar. Whether to fill the credible intervals using `polygon`. |
| `fill` | Character vector. Colour(s) to fill the credible intervals with. See *col* for more information. |
| `variables` | Optional character vector. Names of all variables in the object. Used to subset and title. Taken from `x$variables` if available. |
| `mar` | Numeric vector. Margins for `par`. |
| `...` | Other graphical parameters for `par`. |

## Value

Returns *x* invisibly.

## See Also

`bvar`; `irf.bvar`

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Store IRFs ex-post
irf(x) <- irf(x)

# Plot impulse responses for all available variables
plot(irf(x))

# Subset to impulse variables in positions 2 and 3 via their name
plot(irf(x), vars_impulse = c(2, 3))

# Subset via position and increase the plotted IRF horizon
plot(irf(x, horizon = 20), vars_impulse = c("UNRATE", "FED"))

# Adjust confidence bands and subset to one response variables
plot(irf(x, conf_bands = 0.25), vars_response = "CPI")

# Draw areas inbetween the confidence bands and skip drawing lines
plot(irf(x), col = "transparent", area = TRUE)

# Subset to a specific impulse and response
plot(irf(x), vars_response = "CPI", vars_impulse = "FED")
```

## predict.bvar *Predict method for Bayesian VARs*

### Description

Retrieves / calculates forecasts for Bayesian VARs generated via bvar. If a forecast is already present and no settings are supplied it is simply retrieved, otherwise it will be calculated. To store the results you may want to assign the output using the setter function (`predict(x) <-predict(x)`). May also be used to update confidence bands.

### Usage

```
## S3 method for class 'bvar'
predict(object, ..., conf_bands, n_thin = 1L, newdata)

predict(object) <- value

## S3 method for class 'bvar_fcast'
summary(object, vars = NULL, ...)
```

### Arguments

| | |
|---|---|
| `object` | A bvar object, obtained from bvar. Summary and print methods take in a `bvar_fcast` object. |
| `...` | A `bv_fcast` object or parameters to be fed into `bv_fcast`. Contains settings for the forecast. |
| `conf_bands` | Numeric vector of confidence bands to apply. E.g. for bands at 5%, 10%, 90% and 95% set this to `c(0.05,0.1)`. Note that the median, i.e. 0.5 is always included. |
| `n_thin` | Integer scalar. Every *n_thin*'th draw in *object* is used to predict, others are dropped. |
| `newdata` | Optional numeric matrix or dataframe. Used to base the prediction on. |
| `value` | A `bvar_fcast` object to assign. |
| `vars` | Optional numeric or character vector. Used to subset the summary to certain variables by position or name (must be available). Defaults to `NULL`, i.e. all variables. |

### Value

Returns a list of class `bvar_fcast` including forecasts at desired confidence bands. The summary method returns a numeric array of forecast paths at the specified confidence bands.

### See Also

`plot.bvar_fcast`; `bv_fcast`

## Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

# Calculate a forecast with an increased horizon
y <- predict(x, horizon = 20)

# Add some confidence bands and store the forecast
predict(x) <- predict(x, conf_bands = c(0.05, 0.16))

# Recalculate with different settings and increased thinning
predict(x, bv_fcast(24L), n_thin = 10L)

# Simulate some new data to predict on
predict(x, newdata = matrix(rnorm(300), ncol = 3))

# Calculate a conditional forecast (with a constrained second variable).
predict(x, cond_path = c(1, 1, 1, 1, 1, 1), cond_var = 2)

# Get a summary of the stored forecast
summary(x)

# Only get the summary for variable #2
summary(x, vars = 2L)
```

---

`summary.bvar` *Summary method for Bayesian VARs*

---

## Description

Retrieves several outputs of interest, including the median coefficient matrix, the median variance-covariance matrix, and the log-likelihood. Separate summary methods exist for impulse responses and forecasts.

## Usage

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

## Arguments

| | |
|---|---|
| `object` | A bvar object, obtained from `bvar`. |
| `...` | Not used. |

### Value

Returns a list of class bvar_summary with elements that can can be accessed individually:

- bvar - the bvar object provided.
- coef - coefficient values from coef.bvar.
- vcov - VCOV values from vcov.bvar.
- logLik - the log-likelihood from logLik.

### See Also

bvar; predict.bvar; irf.bvar

### Examples

```
# Access a subset of the fred_qd dataset
data <- fred_qd[, c("CPIAUCSL", "UNRATE", "FEDFUNDS")]
# Transform it to be stationary
data <- fred_transform(data, codes = c(5, 5, 1), lag = 4)

# Estimate a BVAR using one lag, default settings and very few draws
x <- bvar(data, lags = 1, n_draw = 1000L, n_burn = 200L, verbose = FALSE)

summary(x)
```

# Index