Package 'bvartools'

July 23, 2020

Title Bayesian Inference of Vector Autoregressive Models

Version 0.0.3

Date 2020-07-22

Description
Assists in the set-up of algorithms for Bayesian inference of vector autoregressive (VAR) models. Functions for posterior simulation, forecasting, impulse response analysis and forecast error variance decomposition are largely based on the introductory texts of Koop and Korobilis (2010) <doi:10.1561/0800000013> and Luetkepohl (2007, ISBN: 9783540262398).

License GPL (>= 2)

Depends R (>= 3.3.0)

Imports coda, grDevices, graphics, Matrix, Rcpp (>= 0.12.14), stats

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.1.1

URL https://github.com/franzmohr/bvartools

BugReports https://github.com/franzmohr/bvartools/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

Collate 'RcppExports.R' 'bvar.R' 'bvartools.R' 'bvec.R'
'bvec_to_bvar.R' 'data.R' 'fevd.R' 'gen_var.R' 'gen_vec.R'
'inclusion_prior.R' 'irf.R' 'minnesota_prior.R'
'plot.bvarfevd.R' 'plot.bvarirf.R' 'plot.bvarprd.R'
'predict.bvar.R' 'summary.bvar.R' 'print.summary.bvar.R'
'summary.bvec.R' 'print.summary.bvec.R' 'ssvs_prior.R' 'thin.R'
'zzz.R'

NeedsCompilation yes

Author Franz X. Mohr [aut, cre]

Maintainer Franz X. Mohr <bvartools@outlook.com>

Repository CRAN

Date/Publication 2020-07-23 10:10:03 UTC

R topics documented:

bvar — *Bayesian Vector Autoregression Objects*

Description

bvar is used to create objects of class "bvar".

Forecasting a Bayesian VAR object of class "bvar" with credible bands.

Usage

```
bvar(
  data = NULL,
  exogen = NULL,
  y = NULL,
  x = NULL,
  A0 = NULL,
  A = NULL,
  B = NULL,
  C = NULL,
  Sigma = NULL
)

## S3 method for class 'bvar'
predict(object, ..., n.ahead = 10, new_x = NULL, new_D = NULL, ci = 0.95)
```

Arguments

`data`	the original time-series object of endogenous variables.
`exogen`	the original time-series object of unmodelled variables.
`y`	a $K \times T$ matrix of endogenous variables, usually, a result of a call to `gen_var`.
`x`	a $(pK + (1 + s)M + N) \times T$ matrix of regressor variables, usually, a result of a call to `gen_var`.
`A0`	a $K^2 \times S$ matrix of MCMC coefficient draws of structural parameters.
`A`	a $pK^2 \times S$ matrix of MCMC coefficient draws of lagged endogenous variables.
`B`	a $((1 + s)MK) \times S$ matrix of MCMC coefficient draws of unmodelled, non-deterministic variables.
`C`	an $KN \times S$ matrix of MCMC coefficient draws of deterministic terms.
`Sigma`	a $K^2 \times S$ matrix of variance-covariance MCMC draws.
`object`	an object of class `"bvar"`, usually, a result of a call to `bvar` or `bvec_to_bvar`.
`...`	additional arguments.
`n.ahead`	number of steps ahead at which to predict.
`new_x`	a matrix of new non-deterministic, exogenous variables. Must have `n.ahead` rows.
`new_D`	a matrix of new deterministic variables. Must have `n.ahead` rows.
`ci`	a numeric between 0 and 1 specifying the probability mass covered by the credible intervals. Defaults to 0.95.

Details

For the VARX model

$$A_0 y_t = \sum_{i=1}^{p} A_i y_{t-i} + \sum_{i=0}^{s} B_i x_{t-i} + C d_t + u_t$$

the function collects the S draws of a Gibbs sampler (after the burn-in phase) in a standardised object, where y_t is a K-dimensional vector of endogenous variables, A_0 is a $K \times K$ matrix of structural coefficients. A_i is a $K \times K$ coefficient matrix of lagged endogenous variabels. x_t is an M-dimensional vector of unmodelled, non-deterministic variables and B_i its corresponding coefficient matrix. d_t is an N-dimensional vector of deterministic terms and C its corresponding coefficient matrix. u_t is an error term with $u_t \sim N(0, \Sigma_u)$.

The draws of the different coefficient matrices provided in `A0`, `A`, `B`, `C` and `Sigma` have to correspond to the same MCMC iteration.

For the VAR model

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + \sum_{i=0}^{s} B_i x_{t-i} + CD_t + A_0^{-1} u_t,$$

with $u_t \sim N(0, \Sigma)$ the function produces `n.ahead` forecasts.

Value

An object of class "bvar" containing the following components, if specified:

`data`	the original time-series object of endogenous variables.
`exogen`	the original time-series object of unmodelled variables.
`y`	a $K \times T$ matrix of endogenous variables.
`x`	a $(pK + (1 + s)M + N) \times T$ matrix of regressor variables.
`A0`	an $S \times K^2$ "mcmc" object of coefficient draws of structural parameters.
`A`	an $S \times pK^2$ "mcmc" object of coefficient draws of lagged endogenous variables.
`B`	an $S \times ((1 + s)MK)$ "mcmc" object of coefficient draws of unmodelled, non-deterministic variables.
`C`	an $S \times NK$ "mcmc" object of coefficient draws of deterministic terms.
`Sigma`	an $S \times K^2$ "mcmc" object of variance-covariance draws.
`specifications`	a list containing information on the model specification.

A time-series object of class `"bvarprd"`.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Examples

```
data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients
```

```
# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)
data("e1")
e1 <- diff(log(e1))
data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin
```

```
t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)

# Generate forecasts
bvar_pred <- predict(bvar_est, n.ahead = 10, new_D = rep(1, 10))

# Plot forecasts
plot(bvar_pred)
```

bvartools *bvartools: Bayesian Inference of Vector Autoregressive Models*

Description

A collection of R and C++ functions, which assist in the set-up of algorithms for Bayesian inference of vector autoregressive (VAR) models.

Details

The package `bvartools` implements some common functions used for Bayesian inference for mulitvariate time series models. It should give researchers maximum freedom in setting up MCMC algorithms in R and keep calculation time limited at the same time. This is achieved by implementing posterior simulation functions in C++. Its main features are

- The `bvar` and `bvec` functions collect the output of a Gibbs sampler in standardised objects, which can be used for further analyses.
- Further functions such as `predict`, `irf`, `fevd` for forecasting, impulse response analysis and forecast error variance decomposition, respectively.
- Computationally intensive functions - such as for posterior simulation - are written in C++ using the `RcppArmadillo` package of Eddelbuettel and Sanderson (2014).

Author(s)

Franz X. Mohr

References

Chan, J., Koop, G., Poirier, D. J., & Tobias, J. L. (2020). *Bayesian Econometric Methods* (2nd ed.). Cambridge: University Press.

Durbin, J., & Koopman, S. J. (2002). A simple and efficient simulation smoother for state space time series analysis. *Biometrika, 89*(3), 603–615.

Eddelbuettel, D., & Sanderson C. (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis, 71*, 1054–1063. `http://dx.doi.org/10.1016/j.csda.2013.02.005`

George, E. I., Sun, D., & Ni, S. (2008). Bayesian stochastic search for VAR model restrictions. *Journal of Econometrics, 142*(1), 553–580. `https://doi.org/10.1016/j.jeconom.2007.08.017`

Koop, G, & Korobilis, D. (2010), Bayesian multivariate time series Methods for empirical macroeconomics, *Foundations and Trends in Econometrics, 3*(4), 267–358. `http://dx.doi.org/10.1561/0800000013`

Koop, G., León-González, R., & Strachan R. W. (2010). Efficient posterior simulation for cointegrated models with priors on the cointegration space. *Econometric Reviews, 29*(2), 224–242. `https://doi.org/10.1080/07474930903382208`

Korobilis, D. (2013). VAR forecasting using Bayesian variable selection. *Journal of Applied Econometrics, 28*(2), 204–230. `https://doi.org/10.1002/jae.1271`

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Sanderson, C., & Curtin, R. (2016). Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software, 1*(2), 26. `http://dx.doi.org/10.21105/joss.00026`

bvec *Bayesian Vector Error Correction Objects*

Description

'bvec' is used to create objects of class "bvec".

Usage

```
bvec(
  data = NULL,
  exogen = NULL,
  y = NULL,
  w = NULL,
  x = NULL,
  alpha = NULL,
  beta = NULL,
  Pi = NULL,
  Pi_x = NULL,
  Pi_d = NULL,
  A0 = NULL,
  Gamma = NULL,
  Upsilon = NULL,
  C = NULL,
  Sigma = NULL
)
```

Arguments

`data`	the original time-series object of endogenous variables.
`exogen`	the original time-series object of unmodelled variables.
`y`	a $K \times T$ matrix of differenced endogenous variables, usually, a result of a call to `gen_vec`.
`w`	a $(K + M + N^R) \times T$ matrix of variables in the cointegration term, usually, a result of a call to `gen_vec`.
`x`	a $(K(p-1) + Ms + N^{UR}) \times T$ matrix of differenced regressors of y and x, and unrestricted deterministic terms, usually, a result of a call to `gen_vec`.
`alpha`	a $Kr \times S$ matrix of MCMC coefficient draws of the loading matrix α.
`beta`	a $((K + M + N^R)r) \times S$ matrix of MCMC coefficient draws of cointegration matrix β.
`Pi`	a $K^2 \times S$ matrix of MCMC coefficient draws of endogenous varaibles in the cointegration matrix.
`Pi_x`	a $KM \times S$ matrix of MCMC coefficient draws of unmodelled, non-deterministic variables in the cointegration matrix.

`Pi_d`	a $KN^R \times S$ matrix of MCMC coefficient draws of restricted deterministic terms.
`A0`	a $K^2 \times S$ matrix of MCMC coefficient draws of structural parameters.
`Gamma`	a $(p-1)K^2 \times S$ matrix of MCMC coefficient draws of differenced lagged endogenous variables.
`Upsilon`	an $sMK \times S$ matrix of MCMC coefficient draws of differenced unmodelled variables.
`C`	an $KN^{UR} \times S$ matrix of MCMC coefficient draws of unrestricted deterministic terms.
`Sigma`	a $K^2 \times S$ matrix of variance-covariance MCMC draws.

Details

For the VECX model

$$\Delta y_t = \Pi^+ \left(y \right)_{t-1} x_{t-1} d^R_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta y_{t-i} + \sum_{i=0}^{s-1} \Upsilon_i \Delta x_{t-i} + C^{UR} d^{UR}_t + A_0^{-1} u_t$$

the function collects the S draws of a Gibbs sampler (after the burn-in phase) in a standardised object, where Δy_t is a K-dimensional vector of differenced endogenous variables and A_0 is a $K \times K$ matrix of structural coefficients. $\Pi^+ = \left[\Pi, \Pi^x, \Pi^d \right]$ is the coefficient matrix of the error correction term, where y_{t-1}, x_{t-1} and d^R_{t-1} are the first lags of endogenous, exogenous variables in levels and restricted deterministic terms, respectively. Π, Π^x, and Π^d are the corresponding coefficient matrices, respectively. Γ_i is a coefficient matrix of lagged differenced endogenous variabels. Δx_t is an M-dimensional vector of unmodelled, non-deterministic variables and Υ_i its corresponding coefficient matrix. d_t is an N^{UR}-dimensional vector of unrestricted deterministics and C^{UR} the corresponding coefficient matrix. u_t is an error term with $u_t \sim N(0, \Sigma_u)$.

The draws of the different coefficient matrices provided in `alpha`, `beta`, `Pi`, `Pi_x`, `Pi_d`, `A0`, `Gamma`, `Ypsilon`, `C` and `Sigma` have to correspond to the same MCMC iteration.

Value

An object of class "gvec" containing the following components, if specified:

`data`	the original time-series object of endogenous variables.
`exogen`	the original time-series object of unmodelled variables.
`y`	a $K \times T$ matrix of differenced endogenous variables.
`w`	a $(K + M + N^R) \times T$ matrix of variables in the cointegration term.
`x`	a $((p-1)K + sM + N^{UR}) \times T$ matrix of differenced regressor variables and unrestricted deterministic terms.
`A0`	an $S \times K^2$ "mcmc" object of coefficient draws of structural parameters.
`alpha`	an $S \times Kr$ "mcmc" object of coefficient draws of loading parameters.
`beta`	an $S \times ((K + M + N^R)r)$ "mcmc" object of coefficient draws of cointegration parameters.
`Pi`	an $S \times K^2$ "mcmc" object of coefficient draws of endogenous variables in the cointegration matrix.

`Pi_x`	an $S \times KM$ "mcmc" object of coefficient draws of unmodelled, non-deterministic variables in the cointegration matrix.
`Pi_d`	an $S \times KN^R$ "mcmc" object of coefficient draws of unrestricted deterministic variables in the cointegration matrix.
`Gamma`	an $S \times (p-1)K^2$ "mcmc" object of coefficient draws of differenced lagged endogenous variables.
`Upsilon`	an $S \times sMK$ "mcmc" object of coefficient draws of differenced unmodelled variables.
`C`	an $S \times KN^{UR}$ "mcmc" object of coefficient draws of deterministic terms.
`Sigma`	an $S \times K^2$ "mcmc" object of variance-covariance draws.
`specifications`	a list containing information on the model specification.

Examples

```
data("e6")
data <- gen_vec(e6, p = 4, const = "unrestricted", season = "unrestricted")

y <- data$Y
w <- data$W
x <- data$X

# Reset random number generator for reproducibility
set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

r <- 1 # Set rank

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
k_w <- nrow(w) # Number of regressors in error correction term
k_x <- nrow(x) # Number of differenced regressors and unrestrictec deterministic terms

k_alpha <- k * r # Number of elements in alpha
k_beta <- k_w * r # Number of elements in beta
k_gamma <- k * k_x

# Set uninformative priors
a_mu_prior <- matrix(0, k_x * k) # Vector of prior parameter means
a_v_i_prior <- diag(0, k_x * k) # Inverse of the prior covariance matrix

v_i <- 0
p_tau_i <- diag(1, k_w)

u_sigma_df_prior <- r # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
```

```
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
beta <- matrix(c(1, -4), k_w, r)

u_sigma_i <- diag(.0001, k)
u_sigma <- solve(u_sigma_i)

g_i <- u_sigma_i

# Data containers
draws_alpha <- matrix(NA, k_alpha, store)
draws_beta <- matrix(NA, k_beta, store)
draws_pi <- matrix(NA, k * k_w, store)
draws_gamma <- matrix(NA, k_gamma, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  temp <- post_coint_kls(y = y, beta = beta, w = w, x = x, sigma_i = u_sigma_i,
                         v_i = v_i, p_tau_i = p_tau_i, g_i = g_i,
                         gamma_mu_prior = a_mu_prior,
                         gamma_v_i_prior = a_v_i_prior)
  alpha <- temp$alpha
  beta <- temp$beta
  Pi <- temp$Pi
  gamma <- temp$Gamma

  # Draw variance-covariance matrix
  u <- y - Pi %*% w - matrix(gamma, k) %*% x
  u_sigma_scale_post <- solve(tcrossprod(u) +
     v_i * alpha %*% tcrossprod(crossprod(beta, p_tau_i) %*% beta, alpha))
  u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
  u_sigma <- solve(u_sigma_i)

  # Update g_i
  g_i <- u_sigma_i

  # Store draws
  if (draw > burnin) {
    draws_alpha[, draw - burnin] <- alpha
    draws_beta[, draw - burnin] <- beta
    draws_pi[, draw - burnin] <- Pi
    draws_gamma[, draw - burnin] <- gamma
    draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Number of non-deterministic coefficients
k_nondet <- (k_x - 4) * k

# Generate bvec object
```

```
bvec_est <- bvec(y = y, w = w, x = x,
                 Pi = draws_pi,
                 Gamma = draws_gamma[1:k_nondet,],
                 C = draws_gamma[(k_nondet + 1):nrow(draws_gamma),],
                 Sigma = draws_sigma)
```

---

bvec_to_bvar *Transform a VECM to VAR in levels*

Description

An object of class "bvec" is transformed to a VAR in level representation.

Usage

```
bvec_to_bvar(object)
```

Arguments

`object`	an object of class "bvec".

Value

An object of class "bvar".

Examples

```
data("e6")
data <- gen_vec(e6, p = 4, const = "unrestricted", season = "unrestricted")

y <- data$Y
w <- data$W
x <- data$X

# Reset random number generator for reproducibility
set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

r <- 1 # Set rank

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
k_w <- nrow(w) # Number of regressors in error correction term
k_x <- nrow(x) # Number of differenced regressors and unrestrictec deterministic terms
```

```
k_alpha <- k * r # Number of elements in alpha
k_beta <- k_w * r # Number of elements in beta
k_gamma <- k * k_x

# Set uninformative priors
a_mu_prior <- matrix(0, k_x * k) # Vector of prior parameter means
a_v_i_prior <- diag(0, k_x * k) # Inverse of the prior covariance matrix

v_i <- 0
p_tau_i <- diag(1, k_w)

u_sigma_df_prior <- r # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
beta <- matrix(c(1, -4), k_w, r)

u_sigma_i <- diag(.0001, k)
u_sigma <- solve(u_sigma_i)

g_i <- u_sigma_i

# Data containers
draws_alpha <- matrix(NA, k_alpha, store)
draws_beta <- matrix(NA, k_beta, store)
draws_pi <- matrix(NA, k * k_w, store)
draws_gamma <- matrix(NA, k_gamma, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  temp <- post_coint_kls(y = y, beta = beta, w = w, x = x, sigma_i = u_sigma_i,
                         v_i = v_i, p_tau_i = p_tau_i, g_i = g_i,
                         gamma_mu_prior = a_mu_prior,
                         gamma_v_i_prior = a_v_i_prior)
  alpha <- temp$alpha
  beta <- temp$beta
  Pi <- temp$Pi
  gamma <- temp$Gamma

  # Draw variance-covariance matrix
  u <- y - Pi %*% w - matrix(gamma, k) %*% x
  u_sigma_scale_post <- solve(tcrossprod(u) +
     v_i * alpha %*% tcrossprod(crossprod(beta, p_tau_i) %*% beta, alpha))
  u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
  u_sigma <- solve(u_sigma_i)

  # Update g_i
  g_i <- u_sigma_i
  # Store draws
  if (draw > burnin) {
    draws_alpha[, draw - burnin] <- alpha
    draws_beta[, draw - burnin] <- beta
    draws_pi[, draw - burnin] <- Pi
    draws_gamma[, draw - burnin] <- gamma
    draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Number of non-deterministic coefficients
k_nondet <- (k_x - 4) * k

# Generate bvec object
bvec_est <- bvec(y = y, w = w, x = x,
                 Pi = draws_pi,
                 Gamma = draws_gamma[1:k_nondet,],
                 C = draws_gamma[(k_nondet + 1):nrow(draws_gamma),],
                 Sigma = draws_sigma)

# Thin posterior draws
bvec_est <- thin(bvec_est, thin = 5)

# Transfrom VEC output to VAR output
bvar_form <- bvec_to_bvar(bvec_est)
```

bvs *Bayesian Variable Selection*

Description

bvs employs Bayesian variable selection as proposed by Korobilis (2013) to produce a vector of inclusion parameters for the coefficient matrix of a VAR model.

Usage

```
bvs(y, z, a, lambda, sigma_i, prob_prior, include = NULL)
```

Arguments

y	a $K \times T$ matrix of the endogenous variables.
z	a $KT \times M$ matrix of explanatory variables.
a	an M-dimensional vector of parameter draws. If time varying parameters are used, an $M \times T$ coefficient matrix can be provided.
lambda	an $M \times M$ inclusion matrix that should be updated.
sigma_i	the inverse variance-covariance matrix. If the variance-covariance matrix is time varying, a $KT \times K$ matrix can be provided.

`prob_prior`	an M-dimensional vector of prior inclusion probabilities.
`include`	an integer vector specifying the positions of variables, which should be included in the BVS algorithm. If `NULL` (default), BVS will be applied to all variables.

Details

The function employs Bayesian variable selection as proposed by Korobilis (2013) to produce a vector of inclusion parameters, which are the diagonal elements of the inclusion matrix Λ for the VAR model

$$y_t = Z_t \Lambda a_t + u_t,$$

where $u_t \sim N(0, \Sigma_t)$. y_t is a K-dimensional vector of endogenous variables and $Z_t = x_t' \otimes I_K$ is a $K \times M$ matrix of regressors with x_t as a vector of regressors.

Value

A matrix of inclusion parameters on its diagonal.

References

Korobilis, D. (2013). VAR forecasting using Bayesian variable selection. *Journal of Applied Econometrics, 28*(2), 204–230. https://doi.org/10.1002/jae.1271

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))
temp <- gen_var(data, p = 2, deterministic = "const")
y <- temp$Y
x <- temp$Z
z <- kronecker(t(x), diag(1, nrow(y)))
t <- ncol(y)
m <- nrow(y) * nrow(x)

# Priors
a_mu_prior <- matrix(0, m)
a_v_i_prior <- diag(0.1, m)

# Prior for inclusion parameter
prob_prior <- matrix(0.5, m)

# Initial value of Sigma
sigma <- tcrossprod(y) / t
sigma_i <- solve(sigma)

lambda <- diag(1, m)

z_bvs <- z %*% lambda

a <- post_normal_sur(y = y, z = z_bvs, sigma_i = sigma_i,
                     a_prior = a_mu_prior, v_i_prior = a_v_i_prior)
```

```
lambda <- bvs(y = y, z = z, a = a, lambda = lambda,
              sigma_i = sigma_i, prob_prior = prob_prior)
```

e1 *West German economic time series data*

Description

The data set contains quarterly, seasonally adjusted time series for West German fixed investment, disposable income, and consumption expenditures in billions of DM from 1960Q1 to 1982Q4. It was produced from file E1 of the data sets associated with Lütkepohl (2007). Raw data are available at `http://www.jmulti.de/download/datasets/e1.dat` and were originally obtained from Deutsche Bundesbank.

Usage

```
data("e1")
```

Format

A named time-series object with 92 rows and 3 variables:

invest fixed investment.

income disposable income.

cons consumption expenditures.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

e6 *German interest and inflation rate data*

Description

The data set contains quarterly, seasonally unadjusted time series for German long-term interest and inflation rates from 1972Q2 to 1998Q4. It was produced from file E6 of the data sets associated with Lütkepohl (2007). Raw data are available at `http://www.jmulti.de/download/datasets/e6.dat` and were originally obtained from Deutsche Bundesbank and Deutsches Institut für Wirtschaftsforschung.

Usage

```
data("e6")
```

Format

A named time-series object with 107 rows and 2 variables:

R nominal long-term interest rate (Umlaufsrendite).

Dp Δ log of GDP deflator.

Details

The data cover West Germany until 1990Q2 and all of Germany aferwards. The values refer to the last month of a quarter.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

`fevd` *Forecast Error Variance Decomposition*

Description

Produces the forecast error variance decomposition of a Bayesian VAR model.

A plot function for objects of class "bvarfevd".

Usage

```
fevd(object, response = NULL, n.ahead = 5, type = "oir", normalise_gir = FALSE)

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

Arguments

`object`	an object of class `"bvar"`, usually, a result of a call to `bvar` or `bvec_to_bvar`.
`response`	name of the response variable.
`n.ahead`	number of steps ahead.
`type`	type of the impulse responses used to calculate forecast error variable decompositions. Possible choices are orthogonalised `oir` (default) and generalised `gir` impulse responses.
`normalise_gir`	logical. Should the GIR-based FEVD be normalised?
`x`	an object of class "bvarfevd", usually, a result of a call to `fevd`.
`...`	further graphical parameters.

Details

The function produces forecast error variance decompositions (FEVD) for the VAR model

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + A_0^{-1} u_t,$$

with $u_t \sim N(0, \Sigma)$.

If the FEVD is based on the orthogonalised impulse resonse (OIR), the FEVD will be calculated as

$$\omega^{OIR}_{jk,h} = \frac{\sum_{i=0}^{h-1} (e'_j \Phi_i P e_k)^2}{\sum_{i=0}^{h-1} (e'_j \Phi_i \Sigma \Phi'_i e_j)},$$

where Φ_i is the forecast error impulse response for the ith period, P is the lower triangular Choleski decomposition of the variance-covariance matrix Σ, e_j is a selection vector for the response variable and e_k a selection vector for the impulse variable.

If type = "sir", the structural FEVD will be calculated as

$$\omega^{SIR}_{jk,h} = \frac{\sum_{i=0}^{h-1} (e'_j \Phi_i A_0^{-1} \Sigma^{\frac{1}{2}} e_k)^2}{\sum_{i=0}^{h-1} (e'_j \Phi_i A_0^{-1} \Sigma A_0^{-1\prime} \Phi'_i e_j)},$$

where σ_{jj} is the diagonal element of the jth variable of the variance covariance matrix.

If type = "gir", the generalised FEVD will be calculated as

$$\omega^{GIR}_{jk,h} = \frac{\sigma_{jj}^{-1} \sum_{i=0}^{h-1} (e'_j \Phi_i \Sigma e_k)^2}{\sum_{i=0}^{h-1} (e'_j \Phi_i \Sigma \Phi'_i e_j)},$$

where σ_{jj} is the diagonal element of the jth variable of the variance covariance matrix.

If type = "sgir", the structural generalised FEVD will be calculated as

$$\omega^{SGIR}_{jk,h} = \frac{\sigma_{jj}^{-1} \sum_{i=0}^{h-1} (e'_j \Phi_i A_0^{-1} \Sigma e_k)^2}{\sum_{i=0}^{h-1} (e'_j \Phi_i A_0^{-1} \Sigma A_0^{-1\prime} \Phi'_i e_j)}$$

.

Since GIR-based FEVDs do not add up to unity, they can be normalised by setting normalise_gir = TRUE.

Value

A time-series object of class "bvarfevd".

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Pesaran, H. H., & Shin, Y. (1998). Generalized impulse response analysis in linear multivariate models. *Economics Letters, 58*, 17-29.

Examples

```
data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)

# Generate forecast error variance decomposition (FEVD)
bvar_fevd <- fevd(bvar_est, response = "cons")

# Plot FEVD
plot(bvar_fevd, main = "FEVD of consumption")
data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)
```

```
# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)

# Generate forecast error variance decomposition (FEVD)
bvar_fevd <- fevd(bvar_est, response = "cons")

# Plot FEVD
plot(bvar_fevd, main = "FEVD of consumption")
```

gen_var *Vector Autoregressive Model Input*

Description

gen_var produces the input for the estimation of a vector autoregressive (VAR) model.

Usage

```
gen_var(
  data,
  p = 2,
  exogen = NULL,
  s = 2,
  deterministic = "const",
  seasonal = FALSE
)
```

Arguments

data	a time-series object of endogenous variables.
p	an integer of the lag order (default is p = 2).
exogen	an optional time-series object of external regressors.
s	an optional integer of the lag order of the exogenous variables (default is s = 2).

`deterministic`	a character specifying which deterministic terms should be included. Available values are `"none"`, `"const"` (default), `"trend"`, and `"both"`.
`seasonal`	logical. If `TRUE`, seasonal dummy variables are generated. The amount of dummies depends on the frequency of the time-series object provided in `data`.

Details

The function produces the variable matrices of a vector autoregressive (VAR) model, which can also include exogenous variables:

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + \sum_{i=0}^{s} B_i x_{t-i} + C D_t + u_t,$$

where y_t is a K-dimensional vector of endogenous variables, A_i is a $K \times K$ coefficient matrix of endogenous variables, x_t is an M-dimensional vector of exogenous regressors and B_i its corresponding $K \times M$ coefficient matrix. D_t is an N-dimensional vector of deterministic terms and C its corresponding $K \times N$ coefficient matrix. p is the lag order of endogenous variables, s is the lag order of exogenous variables, and u_t is an error term.

In matrix notation the above model can be written as

$$Y = PZ + U,$$

where Y is a $K \times T$ matrix of endogenous variables, Z is a $Kp + M(1+s) + N \times T$ matrix of regressor variables, and U is a $K \times T$ matrix of errors. The function `gen_var` generates the matrices Y and Z.

Value

A list containing the following elements:

`Y`	a matrix of endogenous variables.
`Z`	a matrix of regressor variables.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Examples

```
data("e1")
e1 <- diff(log(e1))
data <- gen_var(e1, p = 2, deterministic = "const")
```

gen_vec *Vector Error Correction Model Input*

Description

gen_vec produces the input for the estimation of a vector error correction (VEC) model.

Usage

```
gen_vec(
  data,
  p = 2,
  exogen = NULL,
  s = 2,
  const = NULL,
  trend = NULL,
  seasonal = NULL
)
```

Arguments

data	a time-series object of endogenous variables.
p	an integer of the lag order of the series (levels) in the VAR.
exogen	an optional time-series object of external regressors.
s	an optional integer of the lag order of the exogenous variables of the series (levels) in the VAR.
const	a character specifying whether a constant term enters the error correction term ("restricted") or the non-cointegration term as an "unrestricted" variable. If NULL (default) no constant term will be added.
trend	a character specifying whether a trend term enters the error correction term ("restricted") or the non-cointegration term as an "unrestricted" variable. If NULL (default) no constant term will be added.
seasonal	a character specifying whether seasonal dummies should be included in the error correction term ("restricted") or in the non-cointegreation term as "unrestricted" variables. If NULL (default) no seasonal terms will be added. The amount of dummy variables depends on the frequency of the time-series object provided in data.

Details

The function produces the variable matrices of a vector error correction (VEC) model, which can also include exogenous variables:

$$\Delta y_t = \Pi w_t + \sum_{i=1}^{p-1} \Gamma_i \Delta y_{t-i} + \sum_{i=0}^{s-1} \Upsilon_i \Delta x_{t-i} + C^{UR} d_t^{UR} + u_t,$$

where Δy_t is a $K \times 1$ vector of differenced endogenous variables, w_t is a $(K + M + N^R) \times 1$ vector of cointegration variables, Π is a $K \times (K + M + N^R)$ matrix of cointegration parameters, Γ_i is a $K \times K$ coefficient matrix of endogenous variables, Δx_t is a $M \times 1$ vector of differenced exogenous regressors, Υ_i is a $K \times M$ coefficient matrix of exogenous regressors, d_t^{UR} is a $N \times 1$ vector of deterministic terms, and C^{UR} is a $K \times N^{UR}$ coefficient matrix of deterministic terms that do not enter the cointegration term. p is the lag order of endogenous variables and s is the lag order of exogenous variables of the corresponding VAR model. u_t is a $K \times 1$ error term.

In matrix notation the above model can be re-written as

$$Y = \Pi W + \Gamma X + U,$$

where Y is a $K \times T$ matrix of differenced endogenous variables, W is a $(K + M + N^R) \times T$ matrix of variables in the cointegration term, X is a $(K(p-1) + Ms + N^{UR}) \times T$ matrix of differenced regressor variables and unrestricted deterministic terms. U is a $K \times T$ matrix of errors.

Value

A list containing the following elements:

`Y`	a matrix of differenced dependent variables.
`W`	a matrix of variables in the cointegration term.
`X`	a matrix of non-cointegration regressors.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Examples

```
data("e6")
data <- gen_vec(e6, p = 4, const = "unrestricted", season = "unrestricted")
```

`inclusion_prior` *Prior Inclusion Probabilities*

Description

Prior inclusion probabilities as required for stochastic search variable selection (SSVS) à la George et al. (2008) and Bayesian variable selection (BVS) à la Korobilis (2013).

Usage

```
inclusion_prior(
  object,
  prob = 0.5,
  exclude_deterministics = TRUE,
  minnesota_like = FALSE,
  kappa = c(0.8, 0.5, 0.5, 0.8)
)
```

Arguments

`object`	an object of class `"bvarmodel"`, usually, a result of a call to `gen_var` or `gen_vec`.
`prob`	a numeric specifying the prior inclusion probability of all model parameters.
`exclude_deterministics`	logical. If `TRUE` (default), the vector of the positions of included variables does not include the positions of deterministic terms.
`minnesota_like`	logical. If `TRUE`, the prior inclusion probabilities of the parameters are calculated in a similar way as the Minnesota prior. See 'Details'.
`kappa`	a numeric vector of four elements containing the prior inclusion probabilities of coefficients that correspond to own lags of endogenous variables, to endogenous variables, which do not correspond to own lags, to exogenous variables and deterministic terms, respectively. Only used if `minnesota_like = TRUE`. See 'Details'.

Details

If `minnesota_like = TRUE`, prior inclusion probabilities $\underline{\pi}_1$ are calculated as

$$\begin{array}{ll} \frac{\kappa_1}{r} & \text{for own lags of endogenous variables,} \\ \frac{\kappa_2}{r} & \text{for other endogenous variables,} \\ \frac{\kappa_3}{1+r} & \text{for exogenous variables,} \\ \kappa_4 & \text{for deterministic variables,} \end{array}$$

for lag r with κ_1, κ_2, κ_3, κ_4 as the first, second, third and forth element in `kappa`, respectively.

For vector error correction models the function generates prior inclusion probabilities for differenced variables and unrestricted deterministc terms as described above. For variables in the error correction term prior inclusion probabilites are calculated as

$$\begin{array}{ll} \kappa_1 & \text{fow own levels of endogenous variables,} \\ \kappa_2 & \text{for levels of other endogenous variables,} \\ \kappa_3 & \text{for levels of exogenous variables,} \\ \kappa_4 & \text{for deterministic variables.} \end{array}$$

Value

A list containing a matrix of prior inclusion probabilities and an integer vector specifying the positions of variables, which should be included in the variable selction algorithm.

References

George, E. I., Sun, D., & Ni, S. (2008). Bayesian stochastic search for VAR model restrictions. *Journal of Econometrics, 142*(1), 553–580. https://doi.org/10.1016/j.jeconom.2007.08.017

Korobilis, D. (2013). VAR forecasting using Bayesian variable selection. *Journal of Applied Econometrics, 28*(2), 204–230. https://doi.org/10.1002/jae.1271

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))

# Generate model input
object <- gen_var(data)

# Obtain inclusion prior
pi_prior <- inclusion_prior(object)
```

`irf` *Impulse Response Function*

Description

Computes the impulse response coefficients of an object of class `"bvar"` for `n.ahead` steps.

A plot function for objects of class "bvarirf".

Usage

```
irf(
  object,
  impulse = NULL,
  response = NULL,
  n.ahead = 5,
  ci = 0.95,
  type = "feir",
  cumulative = FALSE,
  keep_draws = FALSE
)

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

Arguments

`object`	an object of class `"bvar"`, usually, a result of a call to `bvar` or `bvec_to_bvar`.
`impulse`	name of the impulse variable.
`response`	name of the response variable.
`n.ahead`	number of steps ahead.
`ci`	a numeric between 0 and 1 specifying the probability mass covered by the credible intervals. Defaults to 0.95.

`type`	type of the impulse resoponse. Possible choices are forecast error `"feir"` (default), orthogonalised `"oir"`, structural `"sir"`, generalised `"gir"`, and structural generalised `"sgir"` impulse responses.
`cumulative`	logical specifying whether a cumulative IRF should be calculated.
`keep_draws`	logical specifying whether the function should return all draws of the posterior impulse response function. Defaults to `FALSE` so that the median and the credible intervals of the posterior draws are returned.
`x`	an object of class "bvarirf", usually, a result of a call to `irf`.
`...`	further graphical parameters.

Details

The function produces different types of impulse responses for the VAR model

$$y_t = \sum_{i=1}^{p} A_i y_{t-i} + A_0^{-1} u_t,$$

with $u_t \sim N(0, \Sigma)$.

Forecast error impulse responses Φ_i are obtained by recursions

$$\Phi_i = \sum_{j=1}^{i} \Phi_{i-j} A_j, i = 1, 2, ..., h$$

with $\Phi_0 = I_K$.

Orthogonalised impulse responses Θ^o_i are calculated as $\Theta^o_i = \Phi_i P$, where P is the lower triangular Choleski decomposition of Σ. A_0 is assumed to be an identity matrix.

Structural impulse responses Θ^s_i are calculated as $\Theta^s_i = \Phi_i A_0^{-1}$.

(Structural) Generalised impulse responses for variable j, i.e. $\Theta^g_j i$ are calculated as $\Theta^g_{ji} = \sigma_{jj}^{-1/2} \Phi_i A_0^{-1} \Sigma e_j$, where σ_{jj} is the variance of the j^{th} diagonal element of Σ and e_i is a selection vector containing one in its j^{th} element and zero otherwise. If the `"bvar"` object does not contain draws of A_0, it is assumed to be an identity matrix.

Value

A time-series object of class `"bvarirf"` and if `keep_draws = TRUE` a simple matrix.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Pesaran, H. H., Shin, Y. (1998). Generalized impulse response analysis in linear multivariate models. *Economics Letters, 58*, 17-29.

Examples

```
data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)

# Generate impulse response
IR <- irf(bvar_est, impulse = "income", response = "cons", n.ahead = 8)

# Plot
plot(IR, main = "Forecast Error Impulse Response", xlab = "Period", ylab = "Response")


data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
```

```
    a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

  # Draw variance-covariance matrix
  u <- y - matrix(a, k) %*% x # Obtain residuals
  u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
  u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
  u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

  # Store draws
  if (draw > burnin) {
    draws_a[, draw - burnin] <- a
    draws_sigma[, draw - burnin] <- u_sigma
    }
  }

  # Generate bvar object
  bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                   C = draws_a[19:21, ], Sigma = draws_sigma)

  # Generate impulse response
  IR <- irf(bvar_est, impulse = "income", response = "cons", n.ahead = 8)

  # Plot
  plot(IR, main = "Forecast Error Impulse Response", xlab = "Period", ylab = "Response")
```

kalman_dk *Durbin and Koopman Simulation Smoother*

Description

An implementation of the Kalman filter and backward smoothing algorithm proposed by Durbin and Koopman (2002).

Usage

```
kalman_dk(y, z, sigma_u, sigma_v, B, a_init, P_init)
```

Arguments

y	a $K \times T$ matrix of endogenous variables.
z	a $KT \times M$ matrix of explanatory variables.
sigma_u	the constant $K \times K$ error variance-covariance matrix. For time varying variance-covariance matrices a $KT \times K$ can be specified.
sigma_v	the constant $M \times M$ coefficient variance-covariance matrix. For time varying variance-covariance matrices a $MT \times M$ can be specified.
B	an $M \times M$ autocorrelation matrix of the transition equation.
a_init	an M-dimensional vector of initial states.
P_init	an $M \times M$ variance-covariance matrix of the initial states.

Details

The function uses algorithm 2 from Durbin and Koopman (2002) to produce a draw of the state vector a_t for $t = 1, ..., T$ for a state space model with measurement equation

$$y_t = Z_t a_t + u_t$$

and transition equation

$$a_{t+1} = B_t a_t + v_t,$$

where $u_t \sim N(0, \Sigma_{u,t})$ and $v_t \sim N(0, \Sigma_{v,t})$. y_t is a K-dimensional vector of endogenous variables and $Z_t = z_t' \otimes I_K$ is a $K \times M$ matrix of regressors with z_t as a vector of regressors.

The algorithm takes into account Jarociński (2015), where a possible missunderstanding in the implementation of the algorithm of Durbin and Koopman (2002) is pointed out. Following that note the function sets the mean of the initial state to zero in the first step of the algorithm.

Value

A $M \times T + 1$ matrix of state vector draws.

References

Durbin, J., & Koopman, S. J. (2002). A simple and efficient simulation smoother for state space time series analysis. *Biometrika, 89*(3), 603–615.

Jarociński, M. (2015). A note on implementing the Durbin and Koopman simulation smoother. *Computational Statistics and Data Analysis, 91*, 1–3. https://doi.org/10.1016/j.csda.2015.05.001

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))
temp <- gen_var(data, p = 2, deterministic = "const")
y <- temp$Y
x <- temp$Z
k <- nrow(y)
z <- kronecker(t(x), diag(1, k))
t <- ncol(y)
m <- k * nrow(x)

# Priors
a_mu_prior <- matrix(0, m)
a_v_i_prior <- diag(0.1, m)

a_Q <- diag(.0001, m)

# Initial value of Sigma
sigma <- tcrossprod(y) / t
sigma_i <- solve(sigma)
```

```
# Initial values for Kalman filter
y_init <- y * 0
a_filter <- matrix(0, m, t + 1)

# Initialise the Kalman filter
for (i in 1:t) {
  y_init[, i] <- y[, i] - z[(i - 1) * k + 1:k,] %*% a_filter[, i]
}
a_init <- post_normal_sur(y = y_init, z = z, sigma_i = sigma_i,
                          a_prior = a_mu_prior, v_i_prior = a_v_i_prior)
y_filter <- y - matrix(a_init, k) %*% x

# Kalman filter and backward smoother
a_filter <- kalman_dk(y = y_filter, z = z, sigma_u = sigma,
                      sigma_v = a_Q, B = diag(1, m),
                      a_init = matrix(0, m), P_init = a_Q)

a <- a_filter + matrix(a_init, m, t + 1)
```

`loglik_normal` *Calculates the log-likelihood of a multivariate normal distribution.*

Description

Calculates the log-likelihood of a multivariate normal distribution.

Usage

```
loglik_normal(u, sigma)
```

Arguments

u	a $K \times T$ matrix of residuals.
sigma	a $K \times K$ or $KT \times K$ variance-covariance matrix.

Details

The log-likelihood is calculated for each vector in period t as

$$-\frac{K}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_t| - \frac{1}{2}u_t'\Sigma_t^{-1}u_t$$

, where $u_t = y_t - \mu_t$.

Examples

```
data("e1")
e1 <- diff(log(e1))

# Generate VAR model
data <- gen_var(e1, p = 2, deterministic = "const")
y <- data$Y
x <- data$Z

# LS estimate
ols <- tcrossprod(y, x) %*% solve(tcrossprod(x))

# Residuals
u <- y - ols %*% x # Residuals

# Covariance matrix
sigma <- tcrossprod(u) / ncol(u)

# Log-likelihood
loglik_normal(u = u, sigma = sigma)
```

`minnesota_prior` *Minnesota Prior*

Description

Calculates the Minnesota prior for a VAR model.

Usage

```
minnesota_prior(
  object,
  kappa0 = 2,
  kappa1 = 0.5,
  kappa2 = NULL,
  kappa3 = 5,
  max_var = NULL,
  coint_var = FALSE,
  sigma = "AR"
)
```

Arguments

`object`	an object of class `"bvarmodel"`, usually, a result of a call to `gen_var` or `gen_vec`.
`kappa0`	a numeric specifying the prior variance of coefficients that correspond to own lags of endogenous variables.

`kappa1`	a numeric specifying the size of the prior variance of endogenous variables, which do not correspond to own lags, relative to argument `kappa0`.
`kappa2`	a numeric specifying the size of the prior variance of non-deterministic exogenous variables relative to argument `kappa0`. Default is `NULL`, which indicates that the formula for the calculation of the prior variance of deterministic terms is used for all exogenous variables.
`kappa3`	a numeric specifying the size of the prior variance of deterministic terms relative to argument `kappa0`.
`max_var`	a positive numeric specifying the maximum prior variance that is allowed for coefficients of non-deterministic variables. If `NULL` (default), the prior variances are not limited.
`coint_var`	a logical specifying whether the model is a cointegrated VAR model, for which the prior means of first own lags should be set to one.
`sigma`	either `"AR"` (default) or `"VAR"` indicating that the variances of the endogenous variables σ^2 are calculated based on a univariate AR regression or a least squares estimate of the VAR form, respectively. In both cases all deterministic variables are used in the regressions, if they appear in the model.

Details

The function calculates the Minnesota prior of a VAR model. For the endogenous variable i the prior variance of the lth lag of regressor j is obtained as

$$\frac{\kappa_0}{l^2} \text{ for own lags of endogenous variables,}$$

$$\frac{\kappa_0 \kappa_1}{l^2} \frac{\sigma_i^2}{\sigma_j^2} \text{ for endogenous variables other than own lags,}$$

$$\frac{\kappa_0 \kappa_2}{l^2} \frac{\sigma_i^2}{\sigma_j^2} \text{ for exogenous variables,}$$

$$\kappa_0 \kappa_3 \sigma_i^2 \text{ for deterministic terms,}$$

where σ_i is the residual standard deviation of variable i of an unrestricted LS estimate. For exogenous variables σ_i is the sample standard deviation.

For VEC models the function only provides priors for the non-cointegration part of the model. The residual standard errors σ_i are based on an unrestricted LS regression of the endogenous variables on the error correction term and the non-cointegration regressors.

Value

A list containing a matrix of prior means and the precision matrix of the cofficients and the inverse variance-covariance matrix of the error term.

References

Chan, J., Koop, G., Poirier, D. J., & Tobias, J. L. (2020). *Bayesian Econometric Methods* (2nd ed.). Cambridge: University Press.

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))

# Generate model input
object <- gen_var(data)

# Obtain Minnesota prior
prior <- minnesota_prior(object)
```

`plot.bvarprd` *Plotting Forecasts of BVAR Models*

Description

A plot function for objects of class "bvarprd".

Usage

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

Arguments

`x`	an object of class "bvarprd", usually, a result of a call to `predict.bvar`.
`...`	further graphical parameters.

Examples

```
data("e1")
e1 <- diff(log(e1))

data <- gen_var(e1, p = 2, deterministic = "const")

y <- data$Y[, 1:73]
x <- data$Z[, 1:73]

set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

t <- ncol(y) # Number of observations
```

```
k <- nrow(y) # Number of endogenous variables
m <- k * nrow(x) # Number of estimated coefficients

# Set (uninformative) priors
a_mu_prior <- matrix(0, m) # Vector of prior parameter means
a_v_i_prior <- diag(0, m) # Inverse of the prior covariance matrix

u_sigma_df_prior <- 0 # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
u_sigma_i <- diag(.00001, k)
u_sigma <- solve(u_sigma_i)

# Data containers for posterior draws
draws_a <- matrix(NA, m, store)
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  a <- post_normal(y, x, u_sigma_i, a_mu_prior, a_v_i_prior)

# Draw variance-covariance matrix
u <- y - matrix(a, k) %*% x # Obtain residuals
u_sigma_scale_post <- solve(u_sigma_scale_prior + tcrossprod(u))
u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
u_sigma <- solve(u_sigma_i) # Invert Sigma_i to obtain Sigma

# Store draws
if (draw > burnin) {
  draws_a[, draw - burnin] <- a
  draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Generate bvar object
bvar_est <- bvar(y = y, x = x, A = draws_a[1:18,],
                 C = draws_a[19:21, ], Sigma = draws_sigma)

# Generate forecasts
bvar_pred <- predict(bvar_est, n.ahead = 10, new_D = rep(1, 10))

# Plot forecasts
plot(bvar_pred)
```

post_coint_kls *Posterior Draw for Cointegration Models*

Description

Produces a draw of coefficients for cointegration models with a prior on the cointegration space as proposed in Koop et al. (2010) and a draw of non-cointegration coefficients from a normal density.

Usage

```
post_coint_kls(
  y,
  beta,
  w,
  sigma_i,
  v_i,
  p_tau_i,
  g_i,
  x = NULL,
  gamma_mu_prior = NULL,
  gamma_v_i_prior = NULL
)
```

Arguments

`y`	a $K \times T$ matrix of differenced endogenous variables.
`beta`	a $M \times r$ cointegration matrix β.
`w`	a $M \times T$ matrix of variables in the cointegration term.
`sigma_i`	an inverse of the $K \times K$ variance-covariance matrix.
`v_i`	a numeric between 0 and 1 specifying the shrinkage of the cointegration space prior.
`p_tau_i`	an inverted $M \times M$ matrix specifying the central location of the cointegration space prior of $sp(\beta)$.
`g_i`	a $K \times K$ matrix.
`x`	a $N \times T$ matrix of differenced regressors and unrestricted deterministic terms.
`gamma_mu_prior`	a $KN \times 1$ prior mean vector of non-cointegration coefficients.
`gamma_v_i_prior`	an inverted $KN \times KN$ prior covariance matrix of non-cointegration coefficients.

Details

The function produces posterior draws of the coefficient matrices α, β and Γ for the model

$$y_t = \alpha \beta' w_{t-1} + \Gamma z_t + u_t,$$

where y_t is a K-dimensional vector of differenced endogenous variables. w_t is an $M \times 1$ vector of variables in the cointegration term, which include lagged values of endogenous and exogenous variables in levels and restricted deterministic terms. z_t is an N-dimensional vector of differenced endogenous and exogenous explanatory variabes as well as unrestricted deterministic terms. The error term is $u_t \sim \Sigma$.

Draws of the loading matrix α are obtained using the prior on the cointegration space as proposed in Koop et al. (2010). The posterior covariance matrix is

$$\overline{V}_\alpha = \left[\left(v^{-1}(\beta' P_\tau^{-1} \beta) \otimes G_{-1}\right) + \left(ZZ' \otimes \Sigma^{-1}\right)\right]^{-1}$$

and the posterior mean by

$$\overline{\alpha} = \overline{V}_\alpha + vec(\Sigma^{-1} Y Z'),$$

where Y is a $K \times T$ matrix of differenced endogenous variables and $Z = \beta' W$ with W as an $M \times T$ matrix of variables in the cointegration term.

For a given prior mean vector $\underline{\Gamma}$ and prior covariance matrix $\underline{V_\Gamma}$ the posterior covariance matrix of non-cointegration coefficients in Γ is obtained by

$$\overline{V}_\Gamma = \left[\underline{V}_\Gamma^{-1} + \left(XX' \otimes \Sigma^{-1}\right)\right]^{-1}$$

and the posterior mean by

$$\overline{\Gamma} = \overline{V}_\Gamma \left[\underline{V}_\Gamma^{-1} \underline{\Gamma} + vec(\Sigma^{-1} Y X')\right],$$

where X is an $M \times T$ matrix of explanatory variables, which do not enter the cointegration term.

Draws of the cointegration matrix β are obtained using the prior on the cointegration space as proposed in Koop et al. (2010). The posterior covariance matrix of the unrestricted cointegration matrix B is

$$\overline{V}_B = \left[\left(A' G^{-1} A \otimes v^{-1} P_\tau^{-1}\right) + \left(A' \Sigma^{-1} A \otimes WW'\right)\right]^{-1}$$

and the posterior mean by

$$\overline{B} = \overline{V}_B + vec(W Y_B^{-1} \Sigma^{-1} A),$$

where $Y_B = Y - \Gamma X$ and $A = \alpha(\alpha'\alpha)^{-\frac{1}{2}}$.

The final draws of α and β are calculated using $\beta = B(B'B)^{-\frac{1}{2}}$ and $\alpha = A(B'B)^{\frac{1}{2}}$.

Value

A named list containing the following elements:

`alpha`	a draw of the $K \times r$ loading matrix.
`beta`	a draw of the $M \times r$ cointegration matrix.
`Pi`	a draw of the $K \times M$ cointegration matrix $\Pi = \alpha\beta'$.
`Gamma`	a draw of the $K \times N$ coefficient matrix for non-cointegration parameters.

References

Koop, G., León-González, R., & Strachan R. W. (2010). Efficient posterior simulation for cointegrated models with priors on the cointegration space. *Econometric Reviews, 29*(2), 224-242. https://doi.org/10.1080/07474930903382208

Examples

```
# Prepare data
data("e6")
temp <- gen_vec(e6, p = 1)
y <- temp$Y
ect <- temp$W

k <- nrow(y)
t <- ncol(y)

# Initial value of Sigma
sigma <- tcrossprod(y) / t
sigma_i <- solve(sigma)

# Initial values of beta
beta <- matrix(c(1, -4), k)

# Draw parameters
coint <- post_coint_kls(y = y, beta = beta, w = ect, sigma_i = sigma_i,
                        v_i = 0, p_tau_i = diag(1, k), g_i = sigma_i)
```

`post_coint_kls_sur` *Posterior Draw for Cointegration Models*

Description

Produces a draw of coefficients for cointegration models in SUR form with a prior on the cointegration space as proposed in Koop et al. (2010) and a draw of non-cointegration coefficients from a normal density.

Usage

```
post_coint_kls_sur(
  y,
  beta,
  w,
  sigma_i,
  v_i,
  p_tau_i,
  g_i,
  x = NULL,
  gamma_mu_prior = NULL,
  gamma_v_i_prior = NULL,
  svd = FALSE
)
```

Arguments

`y`	a $K \times T$ matrix of differenced endogenous variables.
`beta`	a $M \times r$ cointegration matrix β, where $\beta'\beta = I$.
`w`	a $M \times T$ matrix of variables in the cointegration term.
`sigma_i`	the inverse of the constant $K \times K$ error variance-covariance matrix. For time varying variance-covariance matrics a $KT \times K$ can be provided.
`v_i`	a numeric between 0 and 1 specifying the shrinkage of the cointegration space prior.
`p_tau_i`	an inverted $M \times M$ matrix specifying the central location of the cointegration space prior of $sp(\beta)$.
`g_i`	a $K \times K$ or $KT \times K$ matrix. If the matrix is $KT \times K$, the function will automatically produce a $K \times K$ matrix containing the means of the time varying $K \times K$ covariance matrix.
`x`	a $KT \times NK$ matrix of differenced regressors and unrestricted deterministic terms.
`gamma_mu_prior`	a $KN \times 1$ prior mean vector of non-cointegration coefficients.
`gamma_v_i_prior`	an inverted $KN \times KN$ prior covariance matrix of non-cointegration coefficients.
`svd`	logical. If `TRUE` the singular value decomposition is used to determine the root of the posterior covariance matrix. Default is `FALSE` which means that the eigenvalue decomposition is used.

Details

The function produces posterior draws of the coefficient matrices α, β and Γ for the model

$$y_t = \alpha\beta' w_{t-1} + \Gamma z_t + u_t,$$

where y_t is a K-dimensional vector of differenced endogenous variables. w_t is an $M \times 1$ vector of variables in the cointegration term, which include lagged values of endogenous and exogenous variables in levels and restricted deterministic terms. z_t is an N-dimensional vector of differenced endogenous and exogenous explanatory variabes as well as unrestricted deterministic terms. The error term is $u_t \sim \Sigma$.

Draws of the loading matrix α are obtained using the prior on the cointegration space as proposed in Koop et al. (2010). The posterior covariance matrix is

$$\overline{V}_{\alpha} = \left[\left(v^{-1}(\beta' P_{\tau}^{-1}\beta) \otimes G_{-1}\right) + \left(ZZ' \otimes \Sigma^{-1}\right)\right]^{-1}$$

and the posterior mean by

$$\overline{\alpha} = \overline{V}_{\alpha} + vec(\Sigma^{-1} Y Z'),$$

where Y is a $K \times T$ matrix of differenced endogenous variables and $Z = \beta' W$ with W as an $M \times T$ matrix of variables in the cointegration term.

For a given prior mean vector $\underline{\Gamma}$ and prior covariance matrix $\underline{V_{\Gamma}}$ the posterior covariance matrix of non-cointegration coefficients in Γ is obtained by

$$\overline{V}_{\Gamma} = \left[\underline{V}_{\Gamma}^{-1} + \left(XX' \otimes \Sigma^{-1}\right)\right]^{-1}$$

and the posterior mean by

$$\overline{\Gamma} = \overline{V}_\Gamma \left[\underline{V}_\Gamma^{-1} \underline{\Gamma} + vec(\Sigma^{-1} Y X') \right],$$

where X is an $M \times T$ matrix of explanatory variables, which do not enter the cointegration term.

Draws of the cointegration matrix β are obtained using the prior on the cointegration space as proposed in Koop et al. (2010). The posterior covariance matrix of the unrestricted cointegration matrix B is

$$\overline{V}_B = \left[\left(A' G^{-1} A \otimes v^{-1} P_\tau^{-1} \right) + \left(A' \Sigma^{-1} A \otimes W W' \right) \right]^{-1}$$

and the posterior mean by

$$\overline{B} = \overline{V}_B + vec(W Y_B^{-1} \Sigma^{-1} A),$$

where $Y_B = Y - \Gamma X$ and $A = \alpha(\alpha'\alpha)^{-\frac{1}{2}}$.

The final draws of α and β are calculated using $\beta = B(B'B)^{-\frac{1}{2}}$ and $\alpha = A(B'B)^{\frac{1}{2}}$.

Value

A named list containing the following elements:

alpha	a draw of the $K \times r$ loading matrix.
beta	a draw of the $M \times r$ cointegration matrix.
Pi	a draw of the $K \times M$ cointegration matrix $\Pi = \alpha\beta'$.
Gamma	a draw of the $K \times N$ coefficient matrix for non-cointegration parameters.

References

Koop, G., León-González, R., & Strachan R. W. (2010). Efficient posterior simulation for cointegrated models with priors on the cointegration space. *Econometric Reviews, 29*(2), 224-242. https://doi.org/10.1080/07474930903382208

Examples

```
data("e6")
temp <- gen_vec(e6, p = 1)
y <- temp$Y
ect <- temp$W

k <- nrow(y)
t <- ncol(y)
m <- nrow(ect)

# Initial value of Sigma
sigma <- tcrossprod(y) / t
sigma_i <- solve(sigma)

# Initial values of beta
beta <- matrix(c(1, -4), k)

# Draw parameters
coint <- post_coint_kls_sur(y = y, beta = beta, w = ect,
                            sigma_i = sigma_i, v_i = 0, p_tau_i = diag(1, m),
                                g_i = sigma_i)
```

`post_normal` *Posterior Draw from a Normal Distribution*

Description

Produces a draw of coefficients from a normal posterior density.

Usage

```
post_normal(y, x, sigma_i, a_prior, v_i_prior)
```

Arguments

`y`	a $K \times T$ matrix of endogenous variables.
`x`	an $M \times T$ matrix of explanatory variables.
`sigma_i`	the inverse of the $K \times K$ variance-covariance matrix.
`a_prior`	a $KM \times 1$ numeric vector of prior means.
`v_i_prior`	the inverse of the $KM \times KM$ prior covariance matrix.

Details

The function produces a vectorised posterior draw a of the $K \times M$ coefficient matrix A for the model

$$y_t = Ax_t + u_t,$$

where y_t is a K-dimensional vector of endogenous variables, x_t is an M-dimensional vector of explanatory variabes and the error term is $u_t \sim \Sigma$.

For a given prior mean vector $\underline{a}$ and prior covariance matrix $\underline{V}$ the posterior covariance matrix is obtained by

$$\overline{V} = \left[\underline{V}^{-1} + \left(XX' \otimes \Sigma^{-1}\right)\right]^{-1}$$

and the posterior mean by

$$\overline{a} = \overline{V}\left[\underline{V}^{-1}\underline{a} + vec(\Sigma^{-1}YX')\right],$$

where Y is a $K \times T$ matrix of the endogenous variables and X is an $M \times T$ matrix of the explanatory variables.

Value

A vector.

References

Lütkepohl, H. (2007). *New introduction to multiple time series analysis* (2nd ed.). Berlin: Springer.

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))
temp <- gen_var(data, p = 2, deterministic = "const")
y <- temp$Y
x <- temp$Z
k <- nrow(y)
t <- ncol(y)
m <- k * nrow(x)

# Priors
a_mu_prior <- matrix(0, m)
a_v_i_prior <- diag(0.1, m)

# Initial value of inverse Sigma
sigma_i <- solve(tcrossprod(y) / t)

# Draw parameters
a <- post_normal(y = y, x = x, sigma_i = sigma_i,
                 a_prior = a_mu_prior, v_i_prior = a_v_i_prior)
```

`post_normal_sur` *Posterior Draw from a Normal Distribution*

Description

Produces a draw of coefficients from a normal posterior density for a model with seemingly unrelated regresssions (SUR).

Usage

```
post_normal_sur(y, z, sigma_i, a_prior, v_i_prior, svd = FALSE)
```

Arguments

`y`	a $K \times T$ matrix of endogenous variables.
`z`	a $KT \times M$ matrix of explanatory variables.
`sigma_i`	the inverse of the constant $K \times K$ error variance-covariance matrix. For time varying variance-covariance matrics a $KT \times K$ can be provided.
`a_prior`	a $Mx1$ numeric vector of prior means.
`v_i_prior`	the inverse of the MxM prior covariance matrix.
`svd`	logical. If `TRUE` the singular value decomposition is used to determine the root of the posterior covariance matrix. Default is `FALSE` which means that the eigenvalue decomposition is used.

Details

The function produces a posterior draw of the coefficient vector a for the model

$$y_t = Z_t a + u_t,$$

where $u_t \sim N(0, \Sigma_t)$. y_t is a K-dimensional vector of endogenous variables and $Z_t = z_t' \otimes I_K$ is a $K \times KM$ matrix of regressors with z_t as a vector of regressors.

For a given prior mean vector $\underline{a}$ and prior covariance matrix $\underline{V}$ the posterior covariance matrix is obtained by

$$\overline{V} = \left[\underline{V}^{-1} + \sum_{t=1}^{T} Z_t' \Sigma_t^{-1} Z_t \right]^{-1}$$

and the posterior mean by

$$\overline{a} = \overline{V} \left[\underline{V}^{-1} \underline{a} + \sum_{t=1}^{T} Z_t' \Sigma_t^{-1} y_t \right].$$

Value

A vector.

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))
temp <- gen_var(data, p = 2, deterministic = "const")
y <- temp$Y
x <- temp$Z
k <- nrow(y)
z <- kronecker(t(x), diag(1, k))
t <- ncol(y)
m <- k * nrow(x)

# Priors
a_mu_prior <- matrix(0, m)
a_v_i_prior <- diag(0.1, m)

# Initial value of inverse Sigma
sigma_i <- solve(tcrossprod(y) / t)

# Draw parameters
a <- post_normal_sur(y = y, z = z, sigma_i = sigma_i,
                     a_prior = a_mu_prior, v_i_prior = a_v_i_prior)
```

ssvs *Stochastic Search Variable Selection*

Description

`ssvs` employs stochastic search variable selection as proposed by George et al. (2008) to produce a draw of the precision matrix of the coefficients in a VAR model.

Usage

```
ssvs(a, tau0, tau1, prob_prior, include = NULL)
```

Arguments

`a`	an M-dimensional vector of coefficient draws.
`tau0`	an M-dimensional vector of prior standard deviations for restricted coefficients in vector `a`.
`tau1`	an M-dimensional vector of prior standard deviations for unrestricted coefficients in vector `a`.
`prob_prior`	an M-dimensional vector of prior inclusion probabilites for the coefficients in vector `a`.
`include`	an integer vector specifying the positions of coefficients in vector `a`, which should be included in the SSVS algorithm. If `NULL` (default), SSVS will be applied to all coefficients.

Details

The function employs stochastic search variable selection (SSVS) as proposed by George et al. (2008) to produce a draw of the diagonal inverse prior covariance matrix $\underline{V}^{-1}$ and the corresponding vector of inclusion parameters λ of the vectorised coefficient matrix $a = vec(A)$ for the VAR model

$$y_t = Ax_t + u_t,$$

where y_t is a K-dimensional vector of endogenous variables, x_t is a vector of explanatory variabes and the error term is $u_t \sim \Sigma$.

Value

A named list containing two components:

`v_i`	an $M \times M$ inverse prior covariance matrix.
`lambda`	an M-dimensional vector of inclusion parameters.

References

George, E. I., Sun, D., & Ni, S. (2008). Bayesian stochastic search for VAR model restrictions. *Journal of Econometrics, 142*(1), 553–580. https://doi.org/10.1016/j.jeconom.2007.08.017

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))
temp <- gen_var(data, p = 2, deterministic = "const")
y <- temp$Y
x <- temp$Z
k <- nrow(y)
tt <- ncol(y)
m <- k * nrow(x)

# OLS estimates for semiautomatic approach
ols <- tcrossprod(y, x) %*% solve(tcrossprod(x))
# OLS error covariance matrix
sigma_ols <- tcrossprod(y - ols %*% x) / (tt - nrow(x))
# Covariance matrix
cov_ols <- kronecker(solve(tcrossprod(x)), sigma_ols)
# Standard errors
se_ols <- matrix(sqrt(diag(cov_ols)))

# SSVS priors
tau0 <- se_ols * 0.1 # If excluded
tau1 <- se_ols * 10 # If included

# Prior for inclusion parameter
prob_prior <- matrix(0.5, m)

# Priors
a_mu_prior <- matrix(0, m)
a_v_i_prior <- diag(c(tau1^2), m)

# Initial value of Sigma
sigma_i <- solve(sigma_ols)

# Draw parameters
a <- post_normal(y = y, x = x, sigma_i = sigma_i,
                 a_prior = a_mu_prior, v_i_prior = a_v_i_prior)

# Run SSVS
lambda <- ssvs(a = a, tau0 = tau0, tau1 = tau1,
               prob_prior = prob_prior)
```

`ssvs_prior` *Stochastic Search Variable Selection Prior*

Description

Calculates the priors for a Bayesian VAR model, which employs stochastic search variable selection (SSVS).

Usage

```
ssvs_prior(object, tau = c(0.05, 10), semiautomatic = NULL)
```

Arguments

`object`	an object of class `"bvarmodel"`, usually, a result of a call to `gen_var` or `gen_vec`.
`tau`	a numeric vector of two elements containing the prior standard errors of restricted variables (τ_0) as its first element and unrestricted variables (τ_1) as its second. Default is `c(0.05,10)`.
`semiautomatic`	an optional numeric vector of two elements containing the factors by which the standard errors associated with an unconstrained least squares estimate of the VAR model are multiplied to obtain the prior standard errors of restricted (τ_0) and unrestricted (τ_1) variables. This is the semiautomatic approach described in George et al. (2008).

Value

A list containing the vectors of prior standard deviations for restricted and unrestricted variables, respectively.

References

George, E. I., Sun, D., & Ni, S. (2008). Bayesian stochastic search for VAR model restrictions. *Journal of Econometrics, 142*(1), 553–580. `https://doi.org/10.1016/j.jeconom.2007.08.017`

Examples

```
# Prepare data
data("e1")
data <- diff(log(e1))

# Generate model input
object <- gen_var(data)

# Obtain SSVS prior
prior <- ssvs_prior(object, semiautomatic = c(.1, 10))
```

`summary.bvar` *Summarising Bayesian VAR Coefficients*

Description

summary method for class `"bvar"`.

Usage

```
## S3 method for class 'bvar'
summary(object, ci = 0.95, ...)

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

Arguments

`object`	an object of class "`bvar`", usually, a result of a call to `bvar` or `bvec_to_bvar`.
`ci`	a numeric between 0 and 1 specifying the probability of the credible band. Defaults to 0.95.
`...`	further arguments passed to or from other methods.
`x`	an object of class "`summary.bvar`", usually, a result of a call to `summary.bvar`.
`digits`	the number of significant digits to use when printing.

Value

`summary.bvar` returns a list of class "`summary.bvar`", which contains the following components:

`coefficients`	A list of various summary statistics of the posterior draws of the VAR coefficients.
`sigma`	A list of various summary statistics of the posterior draws of the variance-covariance matrix.
`specifications`	a list containing information on the model specification.

`summary.bvec` *Summarising Bayesian VEC Coefficients*

Description

summary method for class "`bvec`".

Usage

```
## S3 method for class 'bvec'
summary(object, ci = 0.95, ...)

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

Arguments

`object`	an object of class `"bvec"`, usually, a result of a call to `bvec`.
`ci`	a numeric between 0 and 1 specifying the probability of the credible band. Defaults to 0.95.
`...`	further arguments passed to or from other methods.
`x`	an object of class `"summary.bvec"`, usually, a result of a call to `summary.bvec`.
`digits`	the number of significant digits to use when printing.

Value

`summary.bvec` returns a list of class `"summary.bvec"`, which contains the following components:

`coefficients`	A list of various summary statistics of the posterior draws of the VAR coefficients.
`sigma`	A list of various summary statistics of the posterior draws of the variance-covariance matrix.
`specifications`	a list containing information on the model specification.

`thin` *Thinning Posterior Draws*

Description

Thins the MCMC posterior draws in an object of class `"bvar"` or `"bvec"`.

Usage

```
thin(object, thin = 5)
```

Arguments

`object`	an object of class `"bvar"` or `"bvec"`.
`thin`	an integer specifying the thinning interval between successive values of posterior draws.

Value

An object of class `"bvar"` or `"bvec"`.

Examples

```
data("e6")
data <- gen_vec(e6, p = 4, const = "unrestricted", season = "unrestricted")

y <- data$Y
w <- data$W
x <- data$X

# Reset random number generator for reproducibility
set.seed(1234567)

iter <- 500 # Number of iterations of the Gibbs sampler
# Chosen number of iterations should be much higher, e.g. 30000.

burnin <- 100 # Number of burn-in draws
store <- iter - burnin

r <- 1 # Set rank

t <- ncol(y) # Number of observations
k <- nrow(y) # Number of endogenous variables
k_w <- nrow(w) # Number of regressors in error correction term
k_x <- nrow(x) # Number of differenced regressors and unrestrictec deterministic terms

k_alpha <- k * r # Number of elements in alpha
k_beta <- k_w * r # Number of elements in beta
k_gamma <- k * k_x

# Set uninformative priors
a_mu_prior <- matrix(0, k_x * k) # Vector of prior parameter means
a_v_i_prior <- diag(0, k_x * k) # Inverse of the prior covariance matrix

v_i <- 0
p_tau_i <- diag(1, k_w)

u_sigma_df_prior <- r # Prior degrees of freedom
u_sigma_scale_prior <- diag(0, k) # Prior covariance matrix
u_sigma_df_post <- t + u_sigma_df_prior # Posterior degrees of freedom

# Initial values
beta <- matrix(c(1, -4), k_w, r)

u_sigma_i <- diag(.0001, k)
u_sigma <- solve(u_sigma_i)

g_i <- u_sigma_i

# Data containers
draws_alpha <- matrix(NA, k_alpha, store)
draws_beta <- matrix(NA, k_beta, store)
draws_pi <- matrix(NA, k * k_w, store)
draws_gamma <- matrix(NA, k_gamma, store)
```

```
draws_sigma <- matrix(NA, k^2, store)

# Start Gibbs sampler
for (draw in 1:iter) {
  # Draw conditional mean parameters
  temp <- post_coint_kls(y = y, beta = beta, w = w, x = x, sigma_i = u_sigma_i,
                         v_i = v_i, p_tau_i = p_tau_i, g_i = g_i,
                         gamma_mu_prior = a_mu_prior,
                         gamma_v_i_prior = a_v_i_prior)
  alpha <- temp$alpha
  beta <- temp$beta
  Pi <- temp$Pi
  gamma <- temp$Gamma

  # Draw variance-covariance matrix
  u <- y - Pi %*% w - matrix(gamma, k) %*% x
  u_sigma_scale_post <- solve(tcrossprod(u) +
     v_i * alpha %*% tcrossprod(crossprod(beta, p_tau_i) %*% beta, alpha))
  u_sigma_i <- matrix(rWishart(1, u_sigma_df_post, u_sigma_scale_post)[,, 1], k)
  u_sigma <- solve(u_sigma_i)

  # Update g_i
  g_i <- u_sigma_i

  # Store draws
  if (draw > burnin) {
    draws_alpha[, draw - burnin] <- alpha
    draws_beta[, draw - burnin] <- beta
    draws_pi[, draw - burnin] <- Pi
    draws_gamma[, draw - burnin] <- gamma
    draws_sigma[, draw - burnin] <- u_sigma
  }
}

# Number of non-deterministic coefficients
k_nondet <- (k_x - 4) * k

# Generate bvec object
bvec_est <- bvec(y = y, w = w, x = x,
                 Pi = draws_pi,
                 Gamma = draws_gamma[1:k_nondet,],
                 C = draws_gamma[(k_nondet + 1):nrow(draws_gamma),],
                 Sigma = draws_sigma)

# Thin posterior draws
bvec_est <- thin(bvec_est, thin = 4)
```

us_macrodata *US macroeconomic data*

Description

The data set contains quarterly time series for the US CPI inflation rate, unemployment rate, and Fed Funds rate from 1959Q2 to 2007Q4. It was produced from file "US_macrodata.csv" of the data sets associated with Chan, Koop, Poirier and Tobias (2019). Raw data are available at `https://web.ics.purdue.edu/~jltobias/second_edition/Chapter20/code_for_exercise_1/US_macrodata.csv`.

Usage

```
data("us_macrodata")
```

Format

A named time-series object with 195 rows and 3 variables:

Dp CPI inflation rate.

u unemployment rate.

r Fed Funds rate.

References

Chan, J., Koop, G., Poirier, D. J., & Tobias J. L. (2019). *Bayesian econometric methods* (2nd ed.). Cambridge: Cambridge University Press.

Index