Package: bvarsv (via r-universe)

October 28, 2026

Type Package

Title Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters

Version 1.1

Date 2015-10-29

Author Fabian Krueger

Maintainer Fabian Krueger <Fabian.Krueger83@gmail.com>

Description R/C++ implementation of the model proposed by Primiceri (``Time Varying Structural Vector Autoregressions and Monetary Policy", Review of Economic Studies, 2005), with functionality for computing posterior predictive distributions and impulse responses.

License GPL (>= 2)

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp, RcppArmadillo

URL https://sites.google.com/site/fk83research/code

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2015-11-25 14:40:22 UTC

RemoteUrl https://github.com/cran/bvarsv

RemoteRef HEAD

RemoteSha b5c8c14214abe5113e8d11da08ef8268043c349e

Contents

bvarsv-package 2
bvar.sv.tvp 3
Example data sets 5
helpers 6
impulse.responses 8
sim.var1.sv.tvp 10

Index 12

`bvarsv-package` *Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters*

Description

R/C++ implementation of the Primiceri (2005) model, which allows for both stochastic volatility and time-varying regression parameters. The package contains functions for computing posterior predictive distributions and impulse responses from the model, based on an input data set.

Details

Package:	bvarsv
Type:	Package
Version:	1.0
Date:	2014-08-14
License:	GPL (>= 2)
URL:	https://sites.google.com/site/fk83research/code

Author(s)

Fabian Krueger <Fabian.Krueger83@gmail.com>, based on Matlab code by Dimitris Korobilis (see Koop and Korobilis, 2010).

References

The code incorporates the recent corrigendum by Del Negro and Primiceri (2015), which points to an error in the original MCMC algorithm of Primiceri (2005).

Del Negro, M. and Primicerio, G.E. (2015). 'Time Varying Structural Vector Autoregressions and Monetary Policy: A Corrigendum', Review of Economic Studies 82, 1342-1345.

Koop, G. and D. Korobilis (2010): 'Bayesian Multivariate Time Series Methods for Empirical Macroeconomics', Foundations and Trends in Econometrics 3, 267-358. Accompanying Matlab code available at https://sites.google.com/site/dimitriskorobilis/matlab.

Primiceri, G.E. (2005): 'Time Varying Structural Vector Autoregressions and Monetary Policy', Review of Economic Studies 72, 821-852.

Examples

```
## Not run:

# Load US macro data
data(usmacro)

# Estimate trivariate model using Primiceri's prior choices (default settings)
set.seed(5813)
bv <- bvar.sv.tvp(usmacro)


## End(Not run)
```

`bvar.sv.tvp` *Bayesian Analysis of a Vector Autoregressive Model with Stochastic Volatility and Time-Varying Parameters*

Description

Bayesian estimation of the flexible VAR model by Primiceri (2005) which allows for both stochastic volatility and time drift in the model parameters.

Usage

```
bvar.sv.tvp(Y, p = 1, tau = 40, nf = 10, pdrift = TRUE, nrep = 50000,
nburn = 5000, thinfac = 10, itprint = 10000, save.parameters = TRUE,
k_B = 4, k_A = 4, k_sig = 1, k_Q = 0.01, k_S = 0.1, k_W = 0.01,
pQ = NULL, pW = NULL, pS = NULL)
```

Arguments

`Y`	Matrix of data, where rows represent time and columns are different variables. `Y` must have at least two columns.
`p`	Lag length, greater or equal than 1 (the default)
`tau`	Length of the training sample used for determining prior parameters via least squares (LS). That is, data in `Y[1:tau, ]` are used for estimating prior parameters via LS; formal Bayesian analysis is then performed for data in `Y[(tau+1):nrow(Y), ]`.
`nf`	Number of future time periods for which forecasts are computed (integer, 1 or greater, defaults to 10).
`pdrift`	Dummy, indicates whether or not to account for parameter drift when simulating forecasts (defaults to TRUE).
`nrep`	Number of MCMC draws excluding burn-in (defaults to 50000)
`nburn`	Number of MCMC draws used to initialize the sampler (defaults to 5000). These draws do not enter the computation of posterior moments, forecasts etc.

`thinfac` Thinning factor for MCMC output. Defaults to 10, which means that the forecast sequences (`fc.mdraws`, `fc.vdraws`, `fc.ydraws`, see below) contain only every tenth draw of the original sequence. Set `thinfac` to one to obtain the full MCMC sequence.

`itprint` Print every `itprint`-th iteration. Defaults to 10000. Set to very large value to omit printing altogether.

`save.parameters`
If set to `TRUE`, parameter draws are saved in lists (these can be very large). Defaults to `TRUE`.

`k_B`, `k_A`, `k_sig`, `k_Q`, `k_W`, `k_S`, `pQ`, `pW`, `pS`
Quantities which enter the prior distributions, see the links below for details. Defaults to the exact values used in the original article by Primiceri.

Value

`Beta.postmean` Posterior means of coefficients. This is an array of dimension $[M, Mp + 1, T]$, where T denotes the number of time periods (= number of rows of `Y`), and M denotes the number of system variables (= number of columns of `Y`). The submatrix $[, , t]$ represents the coefficient matrix at time t. The intercept vector is stacked in the first column; the `p` coefficient matrices of dimension $[M, M]$ are placed next to it.

`H.postmean` Posterior means of error term covariance matrices. This is an array of dimension $[M, M, T]$. The submatrix $[, , t]$ represents the covariance matrix at time t.

`Q.postmean`, `S.postmean`, `W.postmean`
Posterior means of various covariance matrices.

`fc.mdraws` Draws for the forecast mean vector at various horizons (three-dimensional array, where the first dimension corresponds to system variables, the second to forecast horizons, and the third to MCMC draws). *Note:* The third dimension will be equal to `nrep/thinfac`, apart from possible rounding issues.

`fc.vdraws` Draws for the forecast covariance matrix. Design similar to `fc.mdraws`, except that the first array dimension contains the lower-diagonal elements of the forecast covariance matrix.

`fc.ydraws` Simulated future observations. Design analogous to `fc.mdraws`.

`Beta.draws`, `H.draws`
Matrices of parameter draws, can be used for computing impulse responses later on (see impulse.responses), and accessed via the helper function `parameter.draws`. These outputs are generated only if `save.parameters` has been set to `TRUE`.

Author(s)

Fabian Krueger, based on Matlab code by Dimitris Korobilis (see Koop and Korobilis, 2010). *Incorporates the corrigendum by Del Negro and Primiceri (2015), which points to an error in the original MCMC algorithm of Primiceri (2005).*

References

Del Negro, M. and Primicerio, G.E. (2015). 'Time Varying Structural Vector Autoregressions and Monetary Policy: A Corrigendum', Review of Economic Studies 82, 1342-1345.

Koop, G. and D. Korobilis (2010): 'Bayesian Multivariate Time Series Methods for Empirical Macroeconomics', Foundations and Trends in Econometrics 3, 267-358. Accompanying Matlab code available at https://sites.google.com/site/dimitriskorobilis/matlab.

Primiceri, G.E. (2005): 'Time Varying Structural Vector Autoregressions and Monetary Policy', Review of Economic Studies 72, 821-852.

See Also

The helper functions `predictive.density` and `predictive.draws` provide simple access to the forecast distribution produced by `bvar.sv.tvp`. Impulse responses can be computed using impulse.responses. For detailed examples and explanations, see the accompanying pdf file hosted at https://sites.google.com/site/fk83research/code.

Examples

```
## Not run:

# Load US macro data
data(usmacro)

# Estimate trivariate BVAR using default settings
set.seed(5813)
bv <- bvar.sv.tvp(usmacro)


## End(Not run)
```

`Example data sets` *US Macroeconomic Time Series*

Description

Inflation rate, unemployment rate and treasury bill interest rate for the US, as used by Primiceri (2005). Whereas `usmacro` covers the time period studied by Primiceri (1953:Q1 to 2001:Q3), `usmacro.update` updates the data until 2015:Q2.

Format

Multiple time series (`mts`) object, series names: 'inf', 'une', and 'tbi'.

Source

Inflation data provided by Federal Reserve Bank of Philadelphia (2015): 'Real-Time Data Research Center', https://www.phil.frb.org/research-and-data/real-time-center/real-time-data/data-files/p Accessed: 2015-10-29. The inflation rate is the year-over-year log growth rate of the GDP price index. We use the 2001:Q4 vintage of the price index for `usmacro`, and the 2015:Q3 vintage for `usmacro.update`.

Unemployment and Treasury Bill: Federal Reserve Bank of St. Louis (2015): 'Federal Reserve Economic Data', http://research.stlouisfed.org/fred2/. Accessed: 2015-10-29. The two series have the identifiers 'UNRATE' and 'TB3MS'. For each quarter, we compute simple averages over three monthly observations.

Disclaimer: Please note that the providers of the original data cannot take responsibility for the data posted here, nor can they answer any questions about them. Users should consult their respective websites for the official and most recent version of the data.

References

Primiceri, G.E. (2005): 'Time Varying Structural Vector Autoregressions and Monetary Policy', Review of Economic Studies 72, 821-852.

Examples

```
## Not run:

# Load and plot data
data(usmacro)
plot(usmacro)


## End(Not run)
```

`helpers` *Helper Functions to Access BVAR Forecast Distributions and Parameter Draws*

Description

Functions to extract a univariate posterior predictive distribution from a model fit generated by `bvar.sv.tvp`.

Usage

```
predictive.density(fit, v = 1, h = 1, cdf = FALSE)
predictive.draws(fit, v = 1, h = 1)
parameter.draws(fit, type = "lag1", row = 1, col = 1)
```

Arguments

fit	List, model fit generated by bvar.sv.tvp
v	Index for variable of interest. *Must be in line with the specification of* fit.
h	Index for forecast horizon of interest. *Must be in line with the specification of* fit.
cdf	Set to TRUE to return cumulative distribution function, set to FALSE to return probability density function
type	Character string, used to specify output for function parameter.draws. Setting to "intercept" returns parameter draws for the intercept vector. Setting to one of "lag1", ..., "lagX", (where X is the lag order used in fit) returns parameter draws from the autoregressive coefficient matrices. Setting to "vcv" returns draws for the elements of the residual variance-covariance matrix.
row, col	Row and column index for the parameter for which parameter.draws should return posterior draws. That is, the function returns the row, col element of the matrix specified by type. Note that col is irrelevant if type = "intercept" has been chosen.

Value

predictive.density returns a function f(z), which yields the value(s) of the predictive density at point(s) z. This function exploits conditional normality of the model, given the posterior draws of the parameters.

predictive.draws returns a list containing vectors of MCMC draws, more specifically:

y	Draws from the predictand itself
m	Mean of the normal distribution for the predictand in each draw
v	Variance of the normal distribution for the predictand in each draw

Both outputs should be closely in line with each other (apart from a small amount of sampling noise), see the link below for details.

parameter.draws returns posterior draws for a single (scalar) parameter of the model fitted by bvar.sv.tvp. The output is a matrix, with rows representing MCMC draws, and columns representing time.

Author(s)

Fabian Krueger

See Also

For examples and background, see the accompanying pdf file hosted at https://sites.google.com/site/fk83research/code.

Examples

```
## Not run:

# Load US macro data
data(usmacro)

# Estimate trivariate BVAR using default settings
set.seed(5813)
bv <- bvar.sv.tvp(usmacro)

# Construct predictive density function for the second variable (inflation), one period ahead
f <- predictive.density(bv, v = 2, h = 1)

# Plot the density for a grid of values
grid <- seq(-2, 5, by = 0.05)
plot(x = grid, y = f(grid), type = "l")

# Cross-check: Extract MCMC sample for the same variable and horizon
smp <- predictive.draws(bv, v = 2, h = 1)

# Add density estimate to plot
lines(density(smp), col = "green")


## End(Not run)
```

`impulse.responses` *Compute Impulse Response Function from a Fitted Model*

Description

Computes impulse response functions (IRFs) from a model fit produced by bvar.sv.tvp. The IRF describes how a variable responds to a shock in another variable, in the periods following the shock. To enable simple handling, this function computes IRFs for only one pair of variables that must be specified in advance (see `impulse.variable` and `response.variable` below).

Usage

```
impulse.responses(fit, impulse.variable = 1, response.variable = 2,
                  t = NULL, nhor = 20, scenario = 2, draw.plot = TRUE)
```

Arguments

`fit`	Model fit produced by bvar.sv.tvp, with the option `save.parameters` set to `TRUE`.
`impulse.variable`	Variable which experiences the shock.
`response.variable`	Variable which (possibly) responds to the shock.

t	Time point from which parameter matrices are to be taken. Defaults to most recent time point.
nhor	Maximal time between impulse and response (defaults to 20).
scenario	If 1, there is no orthogonalizaton, and the shock size corresponds to one unit of the impulse variable. If scenario is either 2 (the default) or 3, the error term variance-covariance matrix is orthogonalized via Cholesky decomposition. For scenario = 2, the Cholesky decomposition of the error term VCV matrix at time point t is used. scenario = 3 is the variant used in Del Negro and Primiceri (2015). Here, the diagonal elements are set to their averages over time, whereas the off-diagonal elements are specific to time t. See the notes below for further information.
draw.plot	If TRUE (the default): Produces a plot showing the 5, 25, 50, 75 and 95 percent quantiles of the simulated impulse responses.

Value

List of two elements:

contemporaneous	Contemporaneous impulse responses (vector of simulation draws).
irf	Matrix of simulated impulse responses, where rows represent simulation draws, and columns represent the number of time periods after the shock (1 in first column, nhor in last column).

Note

If scenario is set to either 2 or 3, the Cholesky transform (transpose of chol) is used to produce the orthogonal impulse responses. See Hamilton (1994), Section 11.4, and particularly Equation [11.4.22]. As discussed by Hamilton, the ordering of the system variables matters, and should be considered carefully. The magnitude of the shock (impulse) corresponds to one standard deviation of the error term.

If scenario = 1, the function simply outputs the matrices of the model's moving average representation, see Equation [11.4.1] in Hamilton (1994). The scenario considered here may be unrealistic, in that an isolated shock may be unlikely. The magnitude of the shock (impulse) corresponds to one unit of the error term.

Further supporting information is available at https://sites.google.com/site/FK83research/code.

Author(s)

Fabian Krueger

References

Hamilton, J.D. (1994): Time Series Analysis, Princeton University Press.

Del Negro, M. and Primicerio, G.E. (2015). 'Time Varying Structural Vector Autoregressions and Monetary Policy: A Corrigendum', Review of Economic Studies 82, 1342-1345. Supplementary material available at http://restud.oxfordjournals.org/content/82/4/1342/suppl/DC1 (accessed: 2015-11-17).

Examples

```
## Not run:

data(usmacro)
set.seed(5813)
# Run BVAR; save parameters
fit <- bvar.sv.tvp(usmacro, save.parameters = TRUE)
# Impulse responses
impulse.responses(fit)


## End(Not run)
```

`sim.var1.sv.tvp` *Simulate from a VAR(1) with Stochastic Volatility and Time-Varying Parameters*

Description

Simulate from a VAR(1) with Stochastic Volatility and Time-Varying Parameters

Usage

```
sim.var1.sv.tvp(B0 = NULL, A0 = NULL, Sig0 = NULL, Q = NULL,
S = NULL, W = NULL, t = 500, init = 1000)
```

Arguments

`B0`	Initial values of mean parameters: Matrix of dimension $[M, M+1]$, where the first column holds the intercept vector and the other columns hold the matrix of first-order autoregressive coefficients. By default (NULL), `B0` corresponds to $M = 2$ uncorrelated zero-mean processes with moderate persistence (first-order autocorrelation of 0.6).
`A0`	Initial values for (transformed) error correlation parameters: Vector of length $0.5 * M * (M-1)$. Defaults to a vector of zeros.
`Sig0`	Initial values for log error term volatility parameters: Vector of length M. Defaults to a vector of zeros.
`Q`, `S`, `W`	Covariance matrices for the innovation terms in the time-varying parameters (B, A, Sig). The matrices are symmetric, with dimensions equal to the number of elements in B, A and Sig, respectively. Default to diagonal matrices with very small terms (`1e-10`) on the main diagonal. This corresponds to essentially no time variation in the parameters and error term matrix elements.
`t`	Number of time periods to simulate.
`init`	Number of draws to initialize simulation (to decrease the impact of starting values).

Value

data	Simulated data, with rows corresponding to time and columns corresponding to the M system variables.
Beta	Array of dimension $[M, M+1, t]$. Submatrix $[, , l]$ holds the parameter matrix for time period l.
H	Array of dimension $[M, M, t]$. Submatrix $[, , l]$ holds the error term covariance matrix for period l.

Note

The choice of 'reasonable' values for the elements of Q, S and W requires some care. If the elements of these matrices are too large, parameter variation can easily become excessive. Too large elements of Q can lead the parameter matrix B into regions which correspond to explosive processes. Too large elements in S and (especially) W may lead to excessive error term variances.

Author(s)

Fabian Krueger

References

Primiceri, G.E. (2005): 'Time Varying Structural Vector Autoregressions and Monetary Policy', Review of Economic Studies 72, 821-852.

See Also

bvar.sv.tvp can be used to fit a model on data generated by sim.var1.sv.tvp. This can be a useful way to analyze the performance of the estimation methods.

Examples

```
## Not run:

# Generate data from a model with moderate time variation in the parameters
# and error term variances
set.seed(5813)
sim <- sim.var1.sv.tvp(Q = 1e-5*diag(6), S = 1e-5*diag(1), W = 1e-5*diag(2))
# Plot both series
matplot(sim$data, type = "l")
# Plot AR(1) parameters of both equations
matplot(cbind(sim$Beta[1,2,], sim$Beta[2,3,]), type = "l")


## End(Not run)
```

Index

∗ **datasets**
    Example data sets, 5
∗ **forecasting methods**
    bvar.sv.tvp, 3
    sim.var1.sv.tvp, 10
∗ **helpers**
    helpers, 6
∗ **impulse response analysis**
    impulse.responses, 8
∗ **package**
    bvarsv-package, 2

bvar.sv.tvp, 3, *5–8*, *11*
bvarsv *(*bvarsv-package*)*, 2
bvarsv-package, 2

chol, *9*

Example data sets, 5

helpers, 6

impulse.responses, *4, 5*, 8

parameter.draws, *4*, *7*
parameter.draws *(*helpers*)*, 6
predictive.density, *5*, *7*
predictive.density *(*helpers*)*, 6
predictive.draws, *5*, *7*
predictive.draws *(*helpers*)*, 6

sim.var1.sv.tvp, 10, *11*

usmacro *(*Example data sets*)*, 5