Package ‘MSBVAR’

November 15, 2016

Version 0.9-3

Date 2011-11-14

Title Markov-Switching, Bayesian, Vector Autoregression Models

Maintainer Patrick Brandt <pbrandt@utdallas.edu>

Imports KernSmooth, xtable, coda, bit, mvtnorm, lattice

Description Provides methods for estimating frequentist and Bayesian Vector Autoregression (VAR) models and Markov-switching Bayesian VAR (MSBVAR). Functions for reduced form and structural VAR models are also available. Includes methods for the generating posterior inferences for these models, forecasts, impulse responses (using likelihood-based error bands), and forecast error decompositions. Also includes utility functions for plotting forecasts and impulse responses, and generating draws from Wishart and singular multivariate normal densities. Current version includes functionality to build and evaluate models with Markov switching.

License MIT + file LICENSE

License_is_FOSS yes

License_restricts_use no

SystemRequirements gcc (>= 4.0)

NeedsCompilation yes

Author Patrick Brandt [aut, cre], W. Ryan Davis [ctb]

Repository CRAN

Date/Publication 2016-11-15 00:14:13

R topics documented:

A02mcmc *Converts A0 objects to coda MCMC objects*

Description

Converts A0 objects from `gibbs.A0.BSVAR` into `mcmc` objects for analysis with `coda`

Usage

```
A02mcmc(x)
```

Arguments

x	$N2 \times$ number of free parameters in A(0) MCMC Gibbs sample object for the B-SVAR model A_0 from `gibbs.A0`. This matrix is a column major to row major version of A(0) that can be used to diagnose covergence and summarize the elements of A(0)

Details

Returns an object of the class `mcmc`, an N2 x number free parameters in A(0) matrix. This can then be fed into `coda` for further analysis of the posterior.

Value

Object with class `mcmc`

Author(s)

Patrick T. Brandt

See Also

`gibbs.A0`,`mcmc`

`BCFdata` *Subset of Data from Brandt, Colaresi, and Freeman (2008)*

Description

This data set in two matrices about the Israeli-Palestinian conflict. The first matrix is a set of endogenous variables that gives 1) monthly Goldstein scaled means that summarize the Israeli-Palestinian conflict from April 1996 - March 2005, 2) average Jewish peace index score [0 = no support to 100=full support] that measure Jewish public support of the peace process based on polls of Jewish respondents from the Tami Steinmetz Center for Peace Research. The conflict measures are dyadic or directed actions from one party towards the other. Positive values indicate an average of more cooperation and less conflict and negative values indicate an average with more conflict than cooperation. These are a subset of the Levant dataset from the Kansas / Penn State / Computational Event Data Project Levant dataset. The data are from AFP news sources and encoded into the World Event Interaction Survey (WEIS) coding system and Goldstein scalings using the Event Data Project TABARI program. Source data can be found on the site below.

Usage

```
data(BCFdata)
```

Format

Two matrices containing 108 observations. The first matrix "Y" is a multiple `ts` object of the endogenous series that measure the average conflict-cooperation level and the public opinion data. This matrix has three columns. Column one, "I2P", is average Goldstein scaled Israeli actions towards the Palestinians; column two, "P2I" is average Goldstein scaled Palestinian actions towards the Israelis; column three is the average Jewish peace index value for the month, "JPI".

The second matrix, "z2" is a set of nine control variables for shifts in the conflict, the Israeli prime ministerial regime, and election trends. The columns of this matrix are 1) a dummy variable for the period from the start of the second Intifada to the start of the Battle of Jenin (October 2000–April 2002, end of the second Intifada). 2) a dummy variable for the post-Battle of Jenin period (May 2002–March 2005), 3-5) dummy variables for the identities of the Israeli prime ministers in each month (one each for Netanyahu, Barak, and Sharon, with Rabin/Peres treated as the reference category). 6-9) a separate time counter that starts at the value 1 in the month after each Israeli election and increases until the time of the next constitutionally mandated election.

Source

Brandt, Patrick T., Michael P. Colaresi and John R. Freeman. 2008. “The Dynamics of Reciprocity, Accountability and Credibility.” *Journal of Conflict Resolution*. 52(3): 343-374.

Replication materials at http://jcr.sagepub.com/content/52/3/343

References

Goldstein, Joshua. S. 1992. "A Conflict-Cooperation Scale for WEIS Event Data" *Journal of Conflict Resolution*. 36:369-385.

Computational Event Data Project http://eventdata.parusanalytics.com/

`cf.forecasts` *Compare VAR forecasts to each other or real data*

Description

Computes the root mean sqaured error and mean absolute error for a series of forecasts or for forecasts and real data.

Usage

```
cf.forecasts(m1, m2)
```

Arguments

`m1`	Matrix of VAR forecasts produced by `forecast.VAR`.
`m2`	Matrix of VAR forecasts or a matrix of real data to compare to forecasts.

Details

Simple RMSE and MAE computation for the forecasts. The reported values are summed over the series and time points.

Value

An object with two elements:

`rmse`	Forecast RMSE
`mae`	Forecast MAE

Author(s)

Patrick T. Brandt

See Also

`forecast` for forecast computations

Examples

```
data(IsraelPalestineConflict)
Y.sample1 <- window(IsraelPalestineConflict, end=c(2002, 52))
Y.sample2 <- window(IsraelPalestineConflict, start=c(2003,1))

# Fit a BVAR model
fit.bvar <- szbvar(Y.sample1, p=6, lambda0=0.6, lambda1=0.1, lambda3=2,
                   lambda4=0.25, lambda5=0, mu5=0, mu6=0, prior=0)

# Forecast -- this gives back the sample PLUS the forecasts!

forecasts <- forecast(fit.bvar, nsteps=nrow(Y.sample2))

# Compare forecasts to real data
cf.forecasts(forecasts[(nrow(Y.sample1)+1):nrow(forecasts),], Y.sample2)
```

---

decay.spec *Lag decay specification check*

Description

Provides a quick way to visualize the lag decay specification in a BVAR model for given parameters by computing the variance of the prior VAR coefficients across various lags.

Usage

```
decay.spec(qm, p, lambda)
```

Arguments

qm	Periodicity parameter: either 4 or 12 for quarterly or monthly data.
p	Number of lags
lambda	Lag decay parameter [>0], which is `lambda3` in the Sims-Zha BVAR specification in `szbvar`

Details

Computes the relative decay in the prior variance of the VAR prior across the lags from 1 to p. Useful for visualizing the rate of decay or how tight the prior becomes at higher order lags.

Value

A time series of length `p` of the prior variances for each lag.

Author(s)

Patrick T. Brandt

References

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

See Also

szbvar

Examples

```
# Harmonic lag decay example
harmonic <- decay.spec(4, 6, 1)

# Quadratic lag decay example
quadratic <- decay.spec(4, 6, 2)

plot(cbind(harmonic,quadratic))
```

---

dfev — *Decompositions of Forecast Error Variance (DFEV) for VAR/BVAR/BSVAR models*

Description

Computes the m dimensional decomposition of forecast error variance (DFEV) for a VAR, BVAR, and BSVAR models. User can specify the decomposition of the contemporaneous innovations.

Usage

```
dfev(varobj, A0 = NULL, k)
```

Arguments

varobj	VAR/BVAR/BSVAR object created from fitting a VAR/BVAR/BSVAR model using `szbvar`, `szbsvar`, or `reduced.form.var`.
A0	Decomposition of the contemporaneous error covariance matrix. Default is to use the (lower) Cholesky decomposition of the residual error covariance matrix for VAR and BVAR models, or the inverse of A_0 in B-SVAR models.
k	Number of periods over which to compute the deccomposition.

Details

The decomposition of the forecast error variance (DFEV) provides a measure of the relationship among forecast errors or impact of shocks to a VAR/BVAR/BSVAR system. It is computed by finding the moving average representation (MAR) of the VAR/BVAR/BSVAR model and then tracing out the path of innovations through the system. For each of the M innovations in a VAR/BVAR/BSVAR, the amount of the variance in these forecast errors or innovations is computed and returned in a table. The table can be accessed via the `print.dfev` and `summary.dfev` functions.

Value

Returns a list with

`errors`	M x M x K of the percentage of the innovations in variable i explained by the other M variables.
`std.err`	M x k dimension matrix of the forecast standard errors.
`names`	Variable names

Note

The interpretation of the DFEV depends on the decomposition of the contemporaneous residuals. In the default case of a Cholesky decomposition, this means that the ordering of the variables in the decomposition determines the effect of each innovation on the subsequent DFEVs. For high correlated series, this will mean that the DFEV is not very robust to the ordering.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John T. Williams. Multiple Time Series Models. Thousand Oaks, CA; Sage Press.

See Also

See also `print.dfev` and `summary.dfev` for a nicely formatted tables and an output example

Examples

```
data(IsraelPalestineConflict)
varnames <- colnames(IsraelPalestineConflict)
fitted.BVAR <- szbvar(IsraelPalestineConflict, p=6, z=NULL,
                       lambda0=0.6, lambda1=0.1,
                       lambda3=2, lambda4=0.25, lambda5=0, mu5=0,
                       mu6=0, nu=3, qm=4, prior=0,
                       posterior.fit=FALSE)

A0 <- t(chol(fitted.BVAR$mean.S))
dat.dfev <- dfev(fitted.BVAR, A0, 24)
```

```
print(dat.dfev)
summary(dat.dfev)
```

`forc.ecdf` *Empirical CDF computations for posterior forecast samples*

Description

Computes (pointwise over time) empirical density (error bands) and mean forecasts for a Monte Carlo or Bayesian posterior sample of forecasts.

Usage

```
forc.ecdf(forecasts, probs = c(0.05, 0.95), start = c(0, 1), ...)
```

Arguments

`forecasts`	Posterior sample of VAR forecasts produced by `hc.forecast.VAR()` or `uc.forecast.VAR()`
`probs`	Error band width in percentiles, default is 90% error band.
`start`	Start value for the time series – as in the `ts()` for the forecast horizon
`...`	Other `ecdf()` parameters

Details

For each endogenous variable in the VAR and each point in the forecast horizon this function estimates the percentile based confidence interval. It then returns a time series matrix beginning at `start` of the mean forecast and the limits of the confidence region for each variable in the forecast sample.

Value

A multiple time series object is returned where the first column is the mean estimate followed by the upper and lower bounds of the confidence region.

Author(s)

Patrick T. Brandt

forecast *Generate forecasts for fitted VAR objects*

Description

Forecasting for VAR/BVAR/BSVAR/MSBVAR objects with structural (endogenous) and exogenous shocks.

Usage

```
forecast(varobj, nsteps, A0=t(chol(varobj$mean.S)),
         shocks=matrix(0, nrow=nsteps, ncol=dim(varobj$ar.coefs)[1]),
         exog.fut=matrix(0, nrow=nsteps, ncol=nrow(varobj$exog.coefs)),
         N1, N2)
```

Arguments

`varobj`	Fitted VAR model of the class VAR, BVAR, BSVAR, or MSBVAR produced by `reduced.form.var`, `szbvar`, `szbsvar` or `gibbs.msbvar`.
`nsteps`	Number of periods in the forecast horizon
`A0`	$m \times m$ matrix of the decomposition of the contemporaneous endogenous forecast innovations for BSVAR models.
`shocks`	Structural shocks to the VAR, BVAR, or BSVAR models. These must be scaled consistent with the structural identification in A_0.
`exog.fut`	nsteps x number of exogenous variables matrix of the future values of exogenous variable shocks. Only implemented for VAR, BVAR, and BSVAR models at present.
`N1`	integer, number of burnin draws for the MSBVAR forecasts.
`N2`	integer, number of final posterior draws for MSBVAR forecasts.

Details

VAR / BVAR / BSVAR models:

This function computes forecasts for the classical and Bayesian VAR models that are estimated in the MSBVAR package. Users can specify shocks to the system over the forecast horizon (both structural and exogenous shocks) for VAR, BVAR, and BSVAR models. The forecasting model is that described by Waggoner and Zha (1999) and can be used to construct unconditional forecasts based on the structural shocks and the contemporaneous decomposition of the innovation variance, A0.

MSBVAR:

Generates a set of `N2` draws from the posterior forecast density. Forecasts are constructed using data augmentation, so the forecasts account for both forecast and parameter uncertainty. The function for the MSBVAR model takes as arguments `varobj`, which is the posterior parameters from a call to `gibbs.msbvar`, and `N1` and `N2` to set the burnin and number of draws from the posterior. The posterior forecasts are based on the mixture over the h regimes for the specified model.

Value

For VAR, BVAR, and BSVAR models:

A matrix time series object, $((T + nsteps) \times m)$ of the original series and forecasts.

For MSBVAR models, a list of 4 elements:

forecasts	$N2 \times nsteps \times m$ array of the posterior forecasts.
ss.sample	bit compressed version of the MS state space. (can be summarized with plot.SS or mean.SS.)
k	number of forecast steps, nsteps
h	integer, number of MS regimes used in the forecasts.

Note

The forecasts can be plotted using the `plot.forecast()` command to select the appropriate sample-forecast horizon.

Author(s)

Patrick T. Brandt

References

Waggoner, Daniel F. and Tao Zha. 1999. "Conditional Forecasts in Dynamic Multivariate Models" Review of *Economics and Statistics*, 81(4):639-651.

See Also

reduced.form.var, szbvar and szbsvar for estimation methods that create the elements needed to forecast

Examples

```
data(IsraelPalestineConflict)
Y.sample1 <- window(IsraelPalestineConflict, end=c(2002, 52))
Y.sample2 <- window(IsraelPalestineConflict, start=c(2003,1))

# Fit a BVAR model
fit.bvar <- szbvar(Y.sample1, p=6, lambda0=0.6, lambda1=0.1, lambda3=2,
                   lambda4=0.25, lambda5=0, mu5=0, mu6=0, prior=0)

# Forecast -- this gives back the sample PLUS the forecasts!

forecasts <- forecast(fit.bvar, nsteps=nrow(Y.sample2))
forecasts.only <- forecasts[(nrow(Y.sample1)+1):nrow(forecasts),]

# Plot forecasts and actual data
i2p <- ts(cbind(Y.sample2[,1], forecasts.only[,1]),
          start=c(2003,1), freq=52)
```

```
p2i <- ts(cbind(Y.sample2[,2], forecasts.only[,2]),
          start=c(2003,1), freq=52)

par(mfrow=c(2,1))
plot(i2p, plot.type=c("single"))
plot(p2i, plot.type=c("single"))


## Not run:
# MSBVAR forecasts

# Fit model
m1 <- msbvar(Y.sample1, p=1, h=2, lambda0=0.8, lambda1=0.2,
             lambda3=1, lambda4=0.2, lambda5=0, mu5=0, mu6=0,
             qm=12, prior=0)

# Gibbs sampling
m1id <- gibbs.msbvar(m1, N1=1000, N2=10000, permute=FALSE, Sigma.idx=1)

# Forecast density estimation
msforc <- forecast(m1id, nsteps=nrow(Y.sample2), N1=1000, N2=10000)

# Summarize forecasts
apply(msforc$forecasts, c(2,3), mean)


## End(Not run)
```

gibbs.A0 *Gibbs sampler for posterior of Bayesian structural vector autoregression models*

Description

Samples from the structural contemporaneous parameter matrix A_0 of a Bayesian Structural Vector Autoregression (B-SVAR) model.

Usage

```
gibbs.A0(varobj, N1, N2, thin=1, normalization="DistanceMLA")
```

Arguments

varobj	A structural BVAR object created by szbsvar
N1	Number of burn-in iterations for the Gibbs sampler (should probably be greater than or equal to 1000).
N2	Number of iterations in the posterior sample.
thin	Thinning parameter for the Gibbs sampler.
normalization	Normalization rule as defined in normalize.svar. Default is "DistanceMLA" as recommended in Waggoner and Zha (2003b).

Details

Samples the posterior pdf of an A_0 matrix for a Bayesian structural VAR using the algorithm described in Waggoner and Zha (2003a). This function is meant to be called after `szbsvar`, so one should consult that function for further information. The function draws `N2 * thin` draws from the sampler and returns the `N2` draws that are the `thin`'th elements of the Gibbs sampler sequence.

The computations are done using compiled C++ code as of version 0.3.0. See the package source code for details about the implementation.

Value

A list of class `gibbs.A0` with five elements:

`A0.posterior` A list of three elements containing the results of the `N2` A_0 draws. The list contains a vector storing all of the draws, the location of the drawn elements in and the dimension of A_0. `A0.posterior$A0` is a vector of length equal to the number of parameters in A_0 times N2. `A0.posterior$struct` is a vector of length equal to the number of free parameters in A_0 that gives the index positions of the elements in A_0. `A0.posterior$m` is m, an integer, the number of equations in the system.

`W.posterior` A list of three elements that describes the vectorized W matrices that characterize the covariance of the restricted parameter space of each column of A_0. `W.posterior$W` is a vector of the elements of all the sampled W matrices. `W.posterior$W.index` is a cumulative index of the elements of W that defines how the W matrices for each iteration of the sampler are stored in the vector. `W.posterior$m` is m, an integer, the number of equations in the system.

`ident` `ident` matrix from the `varobj` of binary elements that defined the free and restricted parameters, as specified in `szbsvar`

`thin` `thin` value that was input into the function for thinning the Gibbs sampler.

`N2` `N2`, size of the posterior sample.

Note

You must have called / loaded an `szbsvar` object to use this Gibbs sampler.

Author(s)

Patrick T. Brandt

References

Waggoner, Daniel F. and Tao A. Zha. 2003a. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

Waggoner, Daniel F. and Tao A. Zha, 2003b. "Likelihood Preserving Normalization in Multiple Equation Models" *Journal of Econometrics*, 114: 329–347

See Also

szbsvar for estimation of the posterior moments of the B-SVAR model,

normalize.svar for a discussion of and references on A_0 normalization.

posterior.fit for computing the marginal log likelihood for the model after sampling the posterior,

and plot for a unique density plot of the A_0 elements.

Examples

```
# SZ, B-SVAR model for the Levant data
data(BCFdata)
m <- ncol(Y)
ident <- diag(m)
ident[1,] <- 1
ident[2,1] <- 1

# estimate the model's posterior moments
set.seed(123)
model <- szbsvar(Y, p=2, z=z2, lambda0=0.8, lambda1=0.1, lambda3=1,
                 lambda4=0.1, lambda5=0.05, mu5=0, mu6=5,
                 ident, qm=12)

# Set length of burn-in and size of posterior.  These are only an
# example.  Production runs should set these much higher.
N1 <- 1000
N2 <- 1000

A0.posterior.obj <- gibbs.A0(model, N1, N2, thin=1)

# Use coda to look at the posterior.
A0.free <- A02mcmc(A0.posterior.obj)

plot(A0.free)
```

---

gibbs.msbvar *Gibbs sampler for a Markov-switching Bayesian reduced form vector autoregression model*

Description

Draws a Bayesian posterior sample for a Markov-switching Bayesian reduced form vector autoregression model based on the setup from the msbvar function.

Usage

```
gibbs.msbvar(x, N1 = 1000, N2 = 1000, permute = TRUE,
             Beta.idx = NULL, Sigma.idx = NULL, Q.method="MH")
```

Arguments

`x`	MSBVAR setup and posterior mode estimate generated using the `msbvar` function.
`N1`	Number of burn-in iterations for the Gibbs sampler (should probably be greater than or equal to 1000)
`N2`	Number of iterations in the posterior sample.
`permute`	Logical (default = TRUE). Should random permutation sampling be used to explore the h! posterior modes?
`Beta.idx`	A two element vector indicating the MSBVAR ceofficient matrix that is to be ordered for non-permutation sampling, i.e., the ordering of the states. The states will be put into ascending order for the parameter selected. The two elements provide are for the two-dimensional array of the VAR coefficients. The first number gives the coefficient, the second the equation numbers. Coefficients are ordered by lag, then variable. So for an m equation VAR where we want the AR(1) coefficient on the second variable's equation, use `c(2,2)`. The intercept is the last value, or $mp + 1$. So the intercept for the first equation in a 4 variable model with two lags is `c(9,1)`.
`Sigma.idx`	Scalar integer giving the equation variance that is to be ordered for non-permutation sampling, i.e., the ordering of the states. The states will be put into ascending order for the variance parameter selected. So if you want to identify the results based on equation three, set `Sigma.idx=3`
`Q.method`	choice of the sampler step for the transition matrix, `Q.default=MH` uses a Metropolis-Hastings algorithm that assumes a stationary Markov process. The other option is `Gibbs` which uses a Gibbs sampler Dirichlet draw for a non-stationary Markov-switching process. See Fruwirth-Schnatter (2006: 318, 340-341 for details)

Details

This function implements a Gibbs sampler for the posterior of a MSBVAR model setup with `msbvar`. This is a reduced form MSBVAR model. The estimation is done in a mixture of native R code and Fortran. The sampling of the BVAR coefficients, the transition matrix, and the error covariances for each regime are done in native R code. The forward-filtering-backward-sampling of the Markov-switching process (The most computationally intensive part of the estimation) is handled in compiled Fortran code. As such, this model is reasonably fast for small samples / small numbers of regimes (say less than 5000 observations and 2-4 regimes). The reason for this mixed implementation is that it is easier to setup variants of the model (E.g., Some coefficients switching, others not; different sampling methods, etc. Details will come in future versions of the package.)

The random permuation of the states is done using a multinomial step: at each draw of the Gibbs sampler, the states are permuted using a multinomial draw. This generates a posterior sample where the states are unidentified. This makes sense, since the user may have little idea of how to select among the $h!$ posterior models of the reduced form MSBVAR model (see e.g., Fruhwirth-Schnatter (2006)). Once a posterior sample has been draw with random permuation, a clustering algorithm (see `plotregimeid`) can be used to identify the states, for example, by examining the intercepts or covariances across the regimes (see the example below for details).

Only the Beta.idx or Sigma.idx value is followed. If the first is given the second will be ignored. So variance ordering for identification can only be used when Beta.idx=NULL. See plotregimeid for plotting and summary methods for the permuted sampler.

The Gibbs sampler is estimated using six steps:

Drawing the state-space for the Markov process This step uses compiled code to draw the 0-1 matrix of the regimes. It uses the Baum-Hamilton-Lee-Kim (BHLK) filter and smoother to estimate the regime probabilities. Draws are based on the standard forward-filter-backward-sample algorithm.

Drawing the Markov transition matrix Q Conditional on the other parameters, this takes a draw from a Dirichlet posterior with the alpha.prior prior.

Regression step update Conditional on the state-space and the Markov-switching process data augmentation steps, estimate a set of h regressions, one for each regime.

Draw the error covariances, Σ_h Conditional on the other steps, compute and draw the error covariances from an inverse Wishart pdf.

Draw the regression coefficients For each set of classified observations' (based on the previous step) BVAR regression coefficients, take a draw from their multivariate normal posterior.

Permute the states If permute = TRUE, then permute the states and the respective coefficients.

The state-space for the MS process is a $T \times h$ matrix of zeros and ones. Since this matrix classifies the observations infor states for the N2 posterior draws, it does not make sense to store it in double precisions. We use the bit package to compress this matrix into a 2-bit integer representation for more efficient storage. Functions are provided (see below) for summarizing and plotting the resulting state-space of the MS process.

Value

A list summarizing the reduced form MSBVAR posterior:

Beta.sample $N2 \times h(m^2p + m)$ of the BVAR regression coefficients for each regime. The ordering is based on regime, equation, intercept (and in the future covariates). So the first p coefficients are the the first equation in the first regime, ordered by lag, not variable; the next is the intercept. This pattern repeats for the remaining coefficents across the regimes.

Sigma.sample $N2 \times h(\frac{m(m+1)}{2})$ matrix of the covariance parameters for the error covariances Σ_h. Since these matrices are symmetric p.d., we only store the upper (or lower) portion. The elements in the matrix are the first, second, etc. columns / rows of the lower / upper version of the matrix.

Q.sample $N2 \times h^2$

transition.sample
An array of N2 $h \times h$ transition matrices.

ss.sample List of class SS for the N2 estimates of the state-space matrices coded as bit objects for compression / efficiency.

pfit A list of the posterior fit statistics for the MSBVAR model.

init.model Initial model – a varobj from a BVAR like szbvar that sets up the data and priors. See szbvar for a description.

`alpha.prior`	Prior for the state-space transitions Q. This is set in the call to `msbvar` and inherited here.
`h`	integer, number of regimes fit in the model.
`p`	integer, lag length
`m`	integer, number of equations

Note

Users need to call this function twice (unless they have really good a priori identification information!) The first call will be using the random permutation sampler (so with `permute = TRUE`) and then some exploration of the clustering of the posterior. Then, once the posterior is identified (i.e., you have chosen one of the $h!$ posterior modes), the function is called with `permute = FALSE` and values specified for `Beta.idx` or `Sigma.idx`. See the example below for usage.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. 2009. "Empirical, Regime-Specific Models of International, Inter-group Conflict, and Politics"

Fruhwirth-Schnatter, Sylvia. 2001. "Markov Chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models". Journal of the American Statistical Association. 96(153):194–209.

Fruhwirth-Schnatter, Sylvia. 2006. Finite Mixture and Markov Switching Models. Springer Series in Statistics New York: Springer.

Sims, Christopher A. and Daniel F. Waggoner and Tao Zha. 2008. "Methods for inference in large multiple-equation Markov-switching models" Journal of Econometrics 146(2):255–274.

Krolzig, Hans-Martin. 1997. Markov-Switching Vector Autoregressions: Modeling, Statistical Inference, and Application to Business Cycle Analysis.

See Also

`msbvar` for initial mode finding, `plot.SS` for plotting regime probabilities, `mean.SS` for computing the mean regime probabilities, `plotregimeid` for identifying the regimes from a permuted sample.

Examples

```
## Not run:
# This example can be pasted into a script or copied into R to run.  It
# takes a few minutes, but illustrates how the code can be used

data(IsraelPalestineConflict)

# Find the mode of an msbvar model
# Initial guess is based on random draw, so set seed.
set.seed(123)
```

```
xm <- msbvar(IsraelPalestineConflict, p=3, h=2,
             lambda0=0.8, lambda1=0.15,
             lambda3=1, lambda4=1, lambda5=0, mu5=0,
             mu6=0, qm=12,
             alpha.prior=matrix(c(10,5,5,9), 2, 2))

# Plot out the initial mode
plot(ts(xm$fp))
print(xm$Q)

# Now sample the posterior
N1 <- 1000
N2 <- 2000

# First, so this with random permutation sampling
x1 <- gibbs.msbvar(xm, N1=N1, N2=N2, permute=TRUE)

# Identify the regimes using clustering in plotregimeid()
plotregimeid(x1, type="all")

# Now re-estimate based on desired regime identification seen in the
# plots.  Here we are using the intercept of the first equation, so
# Beta.idx=c(7,1).

x2 <- gibbs.msbvar(xm, N1=N1, N2=N2, permute=FALSE, Beta.idx=c(7,1))

# Plot regimes
plot.SS(x2)

# Summary of transition matrix
summary(x2$Q.sample)

# Plot of the variance elements
plot(x2$Sigma.sample)

## End(Not run)
```

---

granger.test *Bivariate Granger causality testing*

Description

Bivariate Granger causality testing for multiple time series.

Usage

```
granger.test(y, p)
```

Arguments

y	T x m time series or matrix.
p	Lag length to be used for computing the test

Details

Estimates all possible bivariate Granger causality tests for m variables. Bivariate Granger causality tests for two variables X and Y evaluate whether the past values of X are useful for predicting Y once Y's history has been modeled. The null hypothesis is that the past p values of X do not help in predicting the value of Y.

The test is implemented by regressing Y on p past values of Y and p past values of X. An F-test is then used to determine whether the coefficients of the past values of X are jointly zero.

This produces a matrix with m*(m-1) rows that are all of the possible bivariate Granger causal relations. The results include F-statistics and p-values for each test. Tests are estimated using single equation OLS models.

Value

A matrix with 2 columns. Column 1 are the F-statistic values. Column 2 are the p-values for the F-tests. Row labels specifying the Granger causality relationship tested will be included if variables in the input time series `y` include variable or `dimnames`.

Note

These are bivariate tests – not block exogeneity tests for a fitted VAR model. Note also that these tests are highly sensitive to lag length (p) and the presence of unit roots. Results in the matrix include row labels for nice printing with `xtable()`

Author(s)

Patrick T. Brandt

References

Granger, C.W.J. 1969. "Investigating Causal Relations by Econometric Models and Cross-Spectral Methods" Econometrica 37:424-438.

Sims, C.A. 1972. "Money, Income, and Causality" American Economic Review. 62:540-552.

See Also

`reduced.form.var` for frequentist VAR estimation, `szbvar` for Bayesian VAR estimation with Sims-Zha prior, `var.lag.specification` for VAR lag length testing.

Examples

```
data(IsraelPalestineConflict)
granger.test(IsraelPalestineConflict, p=6)
```

HamiltonGDP *Quarterly U.S. GDP Growth, 1952Q3-1984Q4*

Description

Hamilton's (1989) quarterly data on U.S. GDP growth.

Usage

```
data(HamiltonGDP)
```

Format

`ts` object of quarterly GDP growth named `gdp`. This `ts` classed object has 130 observations.

Source

Hamilton, James. 1989. "A new approach to the economic analysis of nonstationary time series and the business cycle." Econmetrica, 357–384.

References

Hamilton, James. 1989. "A new approach to the economic analysis of nonstationary time series and the business cycle." Econmetrica, 357–384.

hc.forecast *Forecast density estimation of hard condition forecasts for VAR models via MCMC*

Description

Implements a "hard condition" forecast density estimator for VAR/BVAR/B-SVAR models as described in Waggoner and Zha (1999). A "hard condition" forecast is one where the forecast path of one or more variables in a VAR is constrained to be an exact value. The forecast densities are estimated as the posterior sample for the VAR model using Markov Chain Monte Carlo with data augmentation to account for the uncertainty of the forecasts and the parameters. This function DOES account for parameter uncertainty in the MCMC algorithm.

Usage

```
hc.forecast(varobj, yconst, nsteps, burnin,
            gibbs, exog = NULL)
```

Arguments

`varobj`	VAR object produced for an unrestricted VAR or BVAR using `szbvar` or `reduced.form.var`
`yconst`	nsteps x m matrix of the constrained forecasts that matches the variables in the endogenous variables of the VAR object. Unconstrained forecasts should be set to NA or zero.
`nsteps`	Number of periods in the forecast horizon
`burnin`	Burnin cycles for the MCMC algorithm
`gibbs`	Number of cycles of the Gibbs sampler after the `burnin` that are returned in the output
`exog`	num.exog x nsteps matrix of the exogenous variable values for the forecast horizon. If left at the `NULL` default, they are set to zero.

Details

"Hard conditions" are restrictions of the future forecast path of a variable in a VAR. Once a variable has been constrained along the forecast path, the paths of the other variables in the VAR forecasts must be re-estimated to satisfy the forecast constraint, since the constrained variable has a forecast variance of zero (it is assumed known). Thus, an MCMC algorithm must be used to determine the posterior of the forecasts and a consistent set of VAR parameter estimates that satisfy the forecast constraints. This function accounts for the uncertainty of the VAR parameters by sampling from them in the computation of the VAR forecasts.

Value

A list with two components:

`forecast`	gibbs x nsteps x m array of the samples of the VAR forecasts
`orig.y`	T x m time series object of the original endogenous variables

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Waggoner, Daniel F. and Tao Zha. 1999. "Conditional Forecasts in Dynamic Multivariate Models" *Review of Economics and Statistics*, 81(4):639-651.

See Also

`plot.forecast` for plotting, `forecast` for unconditional forecasting of forecast means, `uc.forecast` for MCMC estimation of forecast densities for unconstrained or unconditional forecasts

Examples

```
## Not run:
## Uses the example from Brandt and Freeman 2006.  Will not run unless
## you have their data from the Politcal
## Analysis website!
library(MSBVAR)

# Read the data and set up as a time series
data <- read.dta("levant.weekly.79-03.dta")
attach(data)

# Set up KEDS data
KEDS.data <- ts(cbind(a2i,a2p,i2a,p2a,i2p,p2i),
                start=c(1979,15),
                freq=52,
                names=c("A2I","A2P","I2A","P2A","I2P","P2I"))

# Select the sample we want to use.
KEDS <- window(KEDS.data, end=c(1988,50))


#############################
# Estimate the BVAR models
#############################

# Fit a flat prior model
KEDS.BVAR.flat <- szbvar(KEDS, p=6, z=NULL, lambda0=1,
                         lambda1=1, lambda3=1, lambda4=1, lambda5=0,
                         mu5=0, mu6=0, nu=0, qm=4, prior=2,
                         posterior.fit=F)

# Reference prior model -- Normal-IW prior pdf
KEDS.BVAR.informed <- szbvar(KEDS, p=6, z=NULL, lambda0=0.6,
                             lambda1=0.1, lambda3=2, lambda4=0.5,
                             lambda5=0, mu5=0, mu6=0,
                             nu=ncol(KEDS)+1, qm=4, prior=0,
                             posterior.fit=F)

# Set up conditional forecast matrix conditions
nsteps <- 12
a2i.condition <- rep(mean(KEDS[,1]) + sqrt(var(KEDS[,1])) , nsteps)

yhat<-matrix(c(a2i.condition,rep(0, nsteps*5)), ncol=6)

# Set the random number seed so we can replicate the results.
set.seed(11023)

# Conditional forecasts
conditional.forcs.ref <- hc.forecast(KEDS.BVAR.informed, yhat, nsteps,
                            burnin=3000, gibbs=5000, exog=NULL)

conditional.forcs.flat <- hc.forecast(KEDS.BVAR.flat, yhat, nsteps,
```

```
                                burnin=3000, gibbs=5000, exog=NULL)

# Unconditional forecasts
unconditional.forcs.ref <-uc.forecast(KEDS.BVAR.informed, nsteps,
                                          burnin=3000, gibbs=5000)

unconditional.forcs.flat <- uc.forecast(KEDS.BVAR.flat, nsteps,
                                            burnin=3000, gibbs=5000)

# Set-up and plot the unconditional and conditional forecasts.  This
# code pulls for the forecasts for I2P and P2I and puts them into the
# appropriate array for the figures we want to generate.
uc.flat <- NULL
hc.flat <- NULL
uc.ref <- NULL
hc.ref <- NULL

uc.flat$forecast <- unconditional.forcs.flat$forecast[,,5:6]
hc.flat$forecast <- conditional.forcs.flat$forecast[,,5:6]
uc.ref$forecast <- unconditional.forcs.ref$forecast[,,5:6]
hc.ref$forecast <- conditional.forcs.ref$forecast[,,5:6]

par(mfrow=c(2,2), omi=c(0.25,0.5,0.25,0.25))
plot(uc.flat,hc.flat, probs=c(0.16, 0.84), varnames=c("I2P", "P2I"),
     compare.level=KEDS[nrow(KEDS),5:6], lwd=2)
plot(hc.ref,hc.flat, probs=c(0.16, 0.84), varnames=c("I2P", "P2I"),
     compare.level=KEDS[nrow(KEDS),5:6], lwd=2)


## End(Not run)
```

`initialize.msbvar` *Initializes the mode-finder for a Markov-switching Bayesian VAR model*

Description

Sets up the initial values for the mode optimization of an MSBVAR model with a Sims-Zha prior. This sets up the `initialize.opt` argument of the `msbvar` function. Users can inputs values outside of the defaults for the `Q` transition matrix and other arguments with this function. This function also serves as a model for alternative, user-defined initial values for the Gibbs sampler.

Usage

```
initialize.msbvar(y, p, z = NULL, lambda0, lambda1, lambda3, lambda4,
                    lambda5, mu5, mu6, nu, qm, prior, h, Q = NULL)
```

Arguments

y	$T \times m$ multiple time series object created with `ts()`.
p	Lag length, an integer
z	NOT IMPLEMENTED AT PRESENT: THIS SHOULD BE A $T \times k$ matrix of exogenous variables. Can be `z = NULL` if there are none (the default).
lambda0	$[0, 1]$, Overall tightness of the prior (discounting of prior scale).
lambda1	$[0, 1]$, Standard deviation or tightness of the prior around the AR(1) parameters.
lambda3	Lag decay (> 0, with 1=harmonic)
lambda4	Standard deviation or tightness around the intercept > 0
lambda5	Standard deviation or tightness around the exogneous variable coefficients > 0
mu5	Sum of coefficients prior weight ≥ 0. Larger values imply difference stationarity.
mu6	Dummy initial observations or drift prior ≥ 0. Larger values allow for common trends.
nu	Prior degrees of freedom, $m + 1$
qm	Frequency of the data for lag decay equivalence. Default is 4, and a value of 12 will match the lag decay of monthly to quarterly data. Other values have the same effect as "4"
prior	One of three values: 0 = Normal-Wishart prior, 1 = Normal-flat prior, 2 = flat-flat prior (i.e., akin to MLE)
h	Number of regimes / states, an integer
Q	h dimensional transition matrix for the MS process. $h \times h$ Markov transition matrix whose rows sum to 1 with the main weights on the diagonal elements. Default is `NULL` and the initial value is defined by `qtune`.

Details

This function sets the initial or starting values for the the optimization algorithm for the mode of the MSBVAR models in `msbvar`. This is an attempt to (1) allow for a robust, smart guess for starting the block-optimization algorithm and (2) allow for user inputs to `initialize.opt`.

The function does three things:

(1) Estimates an initial `szbvar` model as a baseline, non-regime switching model.

(2) Estimates a set of `h` VAR regressions based on a `kmeans` clustering of the time series with h clusters or centers. The VAR models fit to each of the h subsets of data are used to initialize the `msbvar` function.

(3) Sets an initial value for `Q` in the block optimization algorithm for the mode of the MLE / posterior for the MSBVAR model. If `Q=NULL`, for an $h \times h$ transition matrix `Q`, this initial value is set based on the results from the `kmeans` clustering of the data. If the user inputs a value of `Q`, this is used and error checked to make sure it has the correct format (i.e., rows sum to 1, etc.)

Value

A list with three elements (these are the inputs for the initialize.opt argument in msbvar)

init.model
: An object of the class SZBVAR, see szbvar for details

thetahat.start
: The starting values for the regression parameters for the block optimization algorithm in msbvar. This is an $m \times (mp + 1 + m) \times h$ array of the initial coefficients. For the ith element of the array, the m rows refer to the equations, the first column elements are the intercepts, the next $2 : (mp + 1)$ columns are the AR(p) coefficients, and the final $m \times m$ elements are the error covariance for the regime, for that array element.

Qhat.start
: Initial value of Q

Note

This function can be used to model other ways to set the initial conditions. The subsequent calls to the msbvar function only require an object that satisfies having the elements returned from this function — computed by this function or the user in some way.

Author(s)

Patrick T. Brandt

See Also

msbvar

Examples

```
##
```

---

irf *Impulse Response Function (IRF) Computation for a VAR*

Description

Computes the impulse response function (IRF) or moving average representation (MAR) for an m-dimensional set of VAR/BVAR/B-SVAR coefficients.

Usage

```
irf(varobj, nsteps, A0=NULL)
```

Arguments

- `varobj` — VAR, BVAR, or BSVAR objects for a fitted VAR, BVAR, or BSVAR model from `szbvar`, `szbsvar` or `reduced.form.var`
- `nsteps` — Number or steps, or the horizon over which to compute the IRFs (typically 1.5 to 2 times the lag length used in estimation
- `A0` — Decomposition contemporaneous error covariance of a VAR/BVAR/BSVAR, default is a Cholesky decomposition of the error covariance matrix for VAR and BVAR models, `A0 = chol(varobj$mean.S)`, and the inverse of A_0 for B-SVAR models, `A0 = solve(varobj$A0.mode)`

Details

This function should rarely be called by the user. It is a working function to compute the IRFs for a VAR model. Users will typically want to used one of the simulation functions that also compute error bands for the IRF, such as `mc.irf` which calls this function and simulates its multivariate posterior distribution.

Value

A list of the AR coefficients used in computing the IRF and the impulse response matrices:

- `B` — $m \times m \times nstep$ Autoregressive coefficient matrices in lag order. Note that all AR coefficient matrices for $nstep > p$ are zero.
- `mhat` — $m \times m \times \times nstep$ impulse response matrices. `mhat[,,i]` are the impulses for the i'th period for the m variables.

Note

The IRF depends on the ordering of the variables and the structure of the decomposition in A0.

Author(s)

Patrick T. Brandt

References

Sims, C.A. and Tao Zha. 1999. "Error Bands for Impulse Responses." Econometrica 67(5): 1113-1156.

Hamilton, James. 1994. Time Series Analysis. Chapter 11.

See Also

See also `dfev` for the related decompositions of the forecast error variance, `mc.irf` for Bayesian and frequentist computations of IRFs and their variances (which is what you probably really want).

Examples

```
data(IsraelPalestineConflict)
rf.var <- reduced.form.var(IsraelPalestineConflict, p=6)
plot(irf(rf.var, nsteps = 12))
```

IsraelPalestineConflict

Weekly Goldstein Scaled Israeli-Palestinian Conflict Data, 1979-2003

Description

This data set gives Goldstein scaled totals that summarize the Israeli-Palestinian conflict from April 1979 - December 2003. These are dyadic or directed actions from one party towards the other. Data are weekly starting April 15, 1979. Positive values indicate cooperation, negative values indicate aggression. These are a subset of the Levant dataset from the Kansas / Penn State Event Data System Levant dataset. The data are from Reuters and AFP news sources and encoded into the World Event Interaction Survey (WEIS) coding system and Goldstein scalings using the Penn State Event Data System TABARI program. Source data can be found on the KEDS site below.

Usage

```
data(IsraelPalestineConflict)
```

Format

A matrix containing 1278 observations. Column one, "i2p", is the Israeli actions towards the Palestinians and column two, "p2i" is the Palestinian actions towards the Israelis.

Source

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Goldstein, Joshua. S. 1992. "A Conflict-Cooperation Scale for WEIS Event Data" *Journal of Conflict Resolution*. 36:369-385.

Computational Event Data Project http://eventdata.parusanalytics.com

`ldwishart` *Log density for a Wishart variate*

Description

Computes log density for a Wishart random variable.

Usage

```
ldwishart(W, v, S)
```

Arguments

W	Wishart variate for which the log density is to be computed
v	degrees of freedom for the Wishart variate
S	scale factor for the Wishart variate (typically the inverse covariance if you are working with a multivariate random normal setup

Details

Computes the log density for a Wishart variate with mean S and degrees of freedom v. Special care has been taken to avoid underflow in the computation.

Value

A scalar, the value of the log density for the variate W with mean S and degrees of freedom v.

Note

This is modifed from the log density function in MCMCpack. It better handles underflows.

Author(s)

Patrick T. Brandt

See Also

rwishart

Examples

```
x <- matrix(rnorm(100), 50, 2)
XX <- crossprod(x)
ldwishart(solve(XX), 50, diag(2))
```

list.print — *Prints a list object for the VAR and BVAR models in MSBVAR*

Description

Provides a smartly formatted print method for the list objects created by MSBVAR objects. This will provide a table of estimates for the VAR and BVAR methods in this package.

Usage

```
list.print(x)
```

Arguments

`x`	Fitted model object from `szbvar` or `szbsvar`

Details

This is a way to view the coefficients from a B(S)-VAr model fit with this package.

Value

None. Results are send to STDOUT.

Author(s)

Patrick T. Brandt and Justin Appleby.

See Also

`szbvar`, `szbsvar`

mae — *Mean absolute error of VAR forecasts*

Description

Computes the mean absolute error of VAR forecasts

Usage

```
mae(m1, m2)
```

Arguments

`m1`	$nsteps \times m$ matrix of VAR forecasts
`m2`	$nsteps \times m$ matrix of VAR forecasts or true values

Details

Computes the mean absolute error (MAE) across a series of VAR forecasts.

Value

MAE value

Author(s)

Patrick T. Brandt

See Also

cf.forecasts, rmse

Examples

```
data(IsraelPalestineConflict)
Y.sample1 <- window(IsraelPalestineConflict, end=c(2002, 52))
Y.sample2 <- window(IsraelPalestineConflict, start=c(2003,1))

# Fit a BVAR model
fit.bvar <- szbvar(Y.sample1, p=6, lambda0=0.6, lambda1=0.1, lambda3=2,
                   lambda4=0.25, lambda5=0, mu5=0, mu6=0, prior=0)

# Forecast -- this gives back the sample PLUS the forecasts!

forecasts <- forecast(fit.bvar, nsteps=nrow(Y.sample2))

# Compare forecasts to real data
mae(forecasts[(nrow(Y.sample1)+1):nrow(forecasts),], Y.sample2)
```

---

mc.irf *Monte Carlo Integration / Simulation of Impulse Response Functions*

Description

Simulates a posterior of impulse response functions (IRF) by Monte Carlo integration. This can handle Bayesian and frequentist VARs and Bayesian (structural) VARs estimated with the `szbvar`, `szbsvar` or `reduced.form.var` functions. The decomposition of the contemporaneous innovations is handled by a Cholesky decomposition of the error covariance matrix in reduced form (B)VAR object, or for the contemporaneous structure in S-VAR models. Simulations of IRFs from the Bayesian model utilize the posterior estimates for that model.

Usage

```
mc.irf(varobj, nsteps, draws=1000, A0.posterior=NULL,
       sign.list=rep(1, ncol(varobj$Y)))
```

Arguments

`varobj`	VAR objects for a fitted VAR model from either `reduced.form.var`, `szbvar` or `szbsvar`.
`nsteps`	Number of periods over which to compute the impulse responses
`draws`	Number of draws for the simulation of the posterior distribution of the IRFs (if not a `szbsvar` object. For the MSBVAR model, this is the value of $N2$ from the MCMC sampling (default). You probably should use more than the default given here.
`A0.posterior`	Posterior sample objects generated by `gibbs.A0()` for B-SVAR models, based on the structural identification in `varobj$ident`.
`sign.list`	A list of signs (length = number of variables) for normalization given as either 1 or -1.

Details

VAR/BVAR:

Draws a set of posterior samples from the VAR coefficients and computes impulse responses for each sample. These samples can then be summarized to compute MCMC-based estimates of the responses using the error band methods described in Sims and Zha (1999).

B-SVAR: Generates a set of `N2` draws from the impulse responses for the Bayesian SVAR model in `varobj`. The function takes as its arguments the posterior moments of the B-SVAR model in `varobj`, the draws of the contemporaneous structural coefficients A_0 from `gibbs.A0`, and a list of signs for normalization. This function then computes a posterior sample of the impulse responses based on the Schur product of the sign list and the draws of A_0 and draws from the normal posterior pdf for the other coefficients in the model.

The computations are done using compiled C++ code as of version 0.3.0. See the package source code for details about the implementation.

MSBVAR:

Computes a set of regime specific impulse responses. There will be h of the $m\times$ responses, per the discussion above (shocks in columns, equations / responses in rows. The default is for these to be presented serially. Finally, a regime averaged set of responses, based on the ergodic probability of being in each regime is presented as the "long run" responses. At present this is experimental and open to changes.

Value

VAR/BVAR:

An `mc.irf.VAR` or `mc.irf.BVAR` class object object that is the array of impulse response samples for the Monte Carlo samples

`impulse`	$draws \times nsteps \times m^2$ array of the impulse responses

B-SVAR: `mc.irf.BSVAR` object which is an $(N2, nsteps, m^2)$ array of the impulse responses for the associated B-SVAR model in `varobj` and the posterior A_0.

MS-BVAR `mc.irf.MSBVAR` object which is a list of two arrays. The first array are $(N2, nsteps, m^2, h)$ array of the short-run, regime specific impulse shock-response combinations. The second array are the regime averaged, long run responses based on the ergodic regime probabilities. This second list item is an array of dimensions $(N2, nsteps, m^2)$.

Note

Users need to think carefully about the number of steps and the size of the posterior sample in A_0, since enough memory needs to be available to store and process the posterior of the impulse responses. The number of bytes consumed by the impulse responses will be approximately $m^2 \times nsteps \times N2 \times 16$ where `N2` is the number of draws of A_0 from the `gibbs.A0`. Be sure you have enough memory available to store the object you create!

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Sims, C.A. and Tao Zha. 1999. "Error Bands for Impulse Responses." *Econometrica* 67(5): 1113-1156.

Hamilton, James. 1994. Time Series Analysis. Chapter 11.

Waggoner, Daniel F. and Tao A. Zha. 2003. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

See Also

See also as `plot.mc.irf` for plotting methods and error band construction for the posterior of these impulse response functions, `szbsvar` for estimation of the posterior moments of the B-SVAR model, `gibbs.A0` for drawing posterior samples of A_0 for the B-SVAR model before the IRF computations, and `msbvar` and `gibbs.msbvar` for the specification and computation of the posterior for the MSBVAR models.

Examples

```
# Example 1
data(IsraelPalestineConflict)
varnames <- colnames(IsraelPalestineConflict)

fit.BVAR <- szbvar(Y=IsraelPalestineConflict, p=6, z=NULL,
                   lambda0=0.6, lambda1=0.1,
                   lambda3=2, lambda4=0.25, lambda5=0, mu5=0,
                   mu6=0, nu=3, qm=4,
                   prior=0, posterior.fit=FALSE)

# Draw from the posterior pdf of the impulse responses.
posterior.impulses <- mc.irf(fit.BVAR, nsteps=10, draws=5000)

# Plot the responses
```

```
plot(posterior.impulses, method=c("Sims-Zha2"), component=1,
                 probs=c(0.16,0.84), varnames=varnames)

# Example 2
ident <- diag(2)
varobj <- szbsvar(Y=IsraelPalestineConflict, p=6, z = NULL,
                  lambda=0.6, lambda1=0.1, lambda3=2, lambda4=0.25,
                  lambda5=0, mu5=0, mu6=0, ident, qm = 4)

A0.posterior <- gibbs.A0(varobj, N1=1000, N2=1000)

# Note you need to explcitly reference the sampled A0.posterior object
# in the following call for R to find it in the namespace!

impulse.sample <- mc.irf(varobj, nsteps=12, A0.posterior=A0.posterior)

plot(impulse.sample, varnames=colnames(IsraelPalestineConflict),
     probs=c(0.16,0.84))
```

---

mcmc.szbsvar *Gibbs sampler for coefficients of a B-SVAR model*

Description

Draws a posterior sample of the reduced form coefficients for a Bayesian SVAR model

Usage

```
mcmc.szbsvar(varobj, A0.posterior)
```

Arguments

varobj	A B-SVAR object created by szbsvar
A0.posterior	A posterior sample object generated by gibbs.A0

Details

This function draws the parameters from the Bayesian SVAR model described by Waggoner and Zha (2003). The details can be found in szbsvar. The draws are done for the SVAR model and then translated into the reduced form parameters.

Value

A list of the class "mcmc.bsvar.posterior" with the following components:

A0.posterior	$m \times m \times N2$ array of the posterior matrices A_0.
B.sample	$N2 \times ncoef$ matrix of the reduced form coefficients for the SVAR.

Author(s)

Patrick T. Brandt

References

Waggoner, Daniel F. and Tao A. Zha. 2003. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

See Also

szbsvar

Examples

```
## Not run:
varobj <- szbsvar(Y, p, z = NULL, lambda0, lambda1, lambda3, lambda4,
                  lambda5, mu5, mu6, ident, qm = 4)
posterior <- mcmc.szbsvar(varobj, N1, N2)

## End(Not run)
```

---

mean.SS *Summary measures and plots for MS-B(S)VAR state-spaces*

Description

Provides a summary and plotting methods for the SS class objects produced from sampling the posterior of an MSBVAR model. These functions provide the mean regime probabilities and a plotting method for them.

Usage

```
## S3 method for class 'SS'
mean(x, ...)
## S3 method for class 'SS'
sum(x, ...)
## S3 method for class 'SS'
plot(x, ylab="State Probabilities", ...)
```

Arguments

x	SS class object produced by sampling the posterior of a Markov-switching BVAR model in MSBVAR. These are produced by gibbs.msbvar.
ylab	y-axis label for the regime plot.
...	Other argument or graphics parameters for plot.

Details

The first two provide the sum and mean of the number of time periods in each state of Markov-process. The last produces a time series plot of the regime or state probabilities. These are computed from the Markov Chain Monte Carlo sample computed from `gibbs.msbvar`

Value

Mean and sum are $T \times h$ matrices for the first two summary functions. The plot function generates a plot in the current device. These are the posterior probability measures of the Markov process regimes across T periods.

Author(s)

Patrick T. Brandt

See Also

`gibbs.msbvar`, `msbvar`

`mountains` *Mountain plots for summarizing forecast densities*

Description

"Mountain plots" summarize the bivariate density of 2 variables for two competing forecasts of those variables.

Usage

```
mountains(fcasts1, fcasts2, varnames, pts, ...)
```

Arguments

`fcasts1`	$gibbs \times 2$ set of forecasts from model 1
`fcasts2`	$gibbs \times 2$ set of forecasts from model 2
`varnames`	`c("name1","name2")` object of the variable names
`pts`	`c(pt1,pt2)` which are reference points to be plotted.
`...`	Other graphics parameters.

Details

A "mountain plot" provide a 2×2 graph of plots that summarize the bivariate forecasts for two competing forecasts. This function presents four perspectives on the bivariate density or 'hills' for a set of forecasts. Starting from the bottom right plot and working counter-clockwise, the first plot is the bivariate density of the two competing forecasts. The next plot is a contour map that provide the topography of the densities. The third and fourth plots are projections of densities in each variable. The first forecast in the function is presented in black, the second in red. The densities are estimated from the Gibbs Monte Carlo sample of forecasts using the `bkde2D` bivariate kernel density estimator with an optimal plug-in bandwidth selected using `dpill`.

Value

None. Produces the mountain plot described above in the current graphics device.

Note

This function requires the bivariate kernel smoother in the package `bkde2D`

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

See Also

`bkde2D` for details of the density estimators

Examples

```
## Not run:
data(IsraelPalestineConflict)

# Fit a BVAR model
fit.BVAR <- szbvar(IsraelPalestineConflict, p=6, z=NULL, lambda0=0.6,
                   lambda1=0.1, lambda3=2, lambda4=0.5, lambda5=0,
                   mu5=0, mu6=0, nu=3, qm=4, prior=0,
                   posterior.fit=FALSE)

# Fit a flat prior / MLE model
fit.FREQ <- szbvar(IsraelPalestineConflict, p=6, z=NULL, lambda0=0.6,
                   lambda1=0.1, lambda3=2, lambda4=0.5, lambda5=0,
                   mu5=0, mu6=0, nu=3, qm=4, prior=2,
                   posterior.fit=FALSE)

# Generate unconditional forecasts for both models
forecast.BVAR <- uc.forecast.var(fit.BVAR, nsteps=2,
```

```
                                burnin=100, gibbs=1000)

forecast.FREQ <- uc.forecast.var(fit.FREQ, nsteps=2,
                                burnin=100, gibbs=1000)

# Plot the densities for the forecasts in period of the forecast horizon

mountains(forecast.BVAR$forecast[,2,1:2],
  forecast.FREQ$forecast[,2,1:2], varnames=c("I2P","P2I"), pts=c(0,0))

## End(Not run)
```

---

msbvar *Markov-switching Bayesian reduced form vector autoregression model setup and posterior mode estimation*

Description

Sets up and estimates the posterior mode of a reduced form Markov-switching Bayesian vector autoregression model with a Sims-Zha prior. This is the setup and input function for the Gibbs sampler for this model.

Usage

```
msbvar(Y, z=NULL, p, h, lambda0, lambda1, lambda3,
       lambda4, lambda5, mu5, mu6, qm,
       alpha.prior=100*diag(h) + matrix(2, h, h),
       prior=0, max.iter=40, initialize.opt=NULL)
```

Arguments

Y	A $T \times m$ multiple time series object created with `ts()`.
z	NOT IMPLEMENTED AT PRESENT: THIS SHOULD BE A $T \times k$ matrix of exogenous variables. Can be `z = NULL` if there are none (the default).
p	Lag length, aninteger
h	Number of regimes / states, an integer
lambda0	Value in $[0, 1]$, Overall tightness of the prior (discounting of prior scale).
lambda1	Value in $[0, 1]$, Standard deviation or tightness of the prior around the AR(1) parameters.
lambda3	Lag decay (> 0, with 1=harmonic)
lambda4	Standard deviation or tightness around the intercept > 0
lambda5	Standard deviation or tightness around the exogneous variable coefficients > 0
mu5	Sum of coefficients prior weight ≥ 0. Larger values imply difference stationarity.

`mu6` Dummy initial observations or drift prior ≥ 0. Larger values allow for common trends.

`qm` Frequency of the data for lag decay equivalence. Default is 4, and a value of 12 will match the lag decay of monthly to quarterly data. Other values have the same effect as "4"

`alpha.prior` Prior for the Dirichlet process for the MS process. Default is `100 * diag(h) + matrix(2, h, h)`, but the model will be sensitive to this.

`prior` One of three values: 0= Normal-Wishart prior, 1 = Normal-flat prior, 2 = flat-flat prior (i.e., akin to MLE). The conjugate prior is the first one, which is the default.

`max.iter` Maximum number of iterations for the block EM algorithm used to fit an initial guess of the model posterior. Default value is 40 iterations. Larger problems will need more iterations.

`initialize.opt`
Initial values for the block optimization algorithm. If `default=NULL` `initialize.msbvar` is called to provide values. User can specify values as long as they conform to the structure produced by `initialize.msbvar`.

Details

This function estimates the posterior mode of a reduced form Bayesian Markov-switching VAR model. The MSBVAR mode is estimated using block EM algorithm where the blocks are 1) the BVAR regression coefficients for each regime (separating optimands for intercepts, AR coefficients, and error covariances) and 2) the transition matrix. Starting values are randomly drawn, so a random number seed should be set prior to calling the function in order to make the results replicable.

The steps of the blockwise optimization follow the suggestions of Sims, Waggoner, and Zha (2008). The joint optimization problem is partitioned into the following separate blocks. For each block, a separate call to `optim` is made, holding all of the other blocks constant:

1. *Maximize over the intercepts*
2. *Maximize over the AR(p) coefficients*
3. *Maximize over the error covariances* Σ
4. *Maximize over the transition matrix* Q

These four blocks are iterated a total of `max.iter` times. Internal to each block, the state-space filtering algorithm for the regime classifications is computing using compiled Fortran code for speed. Despite the use of compiled code, this algorithm can take several minutes to compute.

The user should try multiple starting values and number of iterations to ensure convergence. The algorithm will improve with each step of the optimization, although sometimes this can be very incremental improvement.

The results for posterior sampling via `gibbs.msbvar` will be sensitive to the choice of `alpha.prior`. This is the prior for the independent Dirichlet process for the MS process. Note that the prior is roughly proportionate to the number of time periods spent in each regime, since the estimated MS probabilities map to the duration of the regime via $1/(1-p)$ where p is the probability of staying in the regime.

This function should NOT be used for inference, since it only finds the posterior mode of the model. This function is intended to generate starting values for the Gibbs sampling of the model. See gibbs.msbvar for further details of the Gibbs sampling.

Value

A list describing the posterior mode of the MSBVAR model and the inputs necessary for the subsequent Gibbs sampler.

init.model	An object of the class BVAR that describes the setup of the model. See szbvar for details.
hreg	A list containing the regime-specific moment matrices, VAR coefficients, and error covariances
Q	The $h \times h$ Markov transition matrix.
fp	The $T \times h$ matrix of the filtered regime probabilities. First column is the first regime, etc.
m	Integer, the number of endogenous variables in the system.
p	Integer, the lag length of the VAR.
h	Integer, the number of regimes in the MS process.
alpha.prior	The $h \times h$ matrix for the prior for the Dirichlet density for the MS process.

Note

Users should consult the reference papers and the (coming) package vignette to see how this function is used to setup an MSBVAR model. An example is currently in gibbs.msbvar.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. 2009. "Empirical, Regime-Specific Models of International, Inter-group Conflict, and Politics"

Fruhwirth-Schnatter, Sylvia. 2001. "Markov Chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models". Journal of the American Statistical Association. 96(153):194–209.

Fruhwirth-Schnatter, Sylvia. 2006. Finite Mixture and Markov Switching Models. Springer Series in Statistics New York: Springer.

Kim, Chang-Jin and Charles R. Nelson. 1999. State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications. Cambridge: MIT Press.

Sims, Christopher A. and Daniel F. Waggoner and Tao Zha. 2008. "Methods for inference in large multiple-equation Markov-switching models" Journal of Econometrics 146(2):255–274.

Sims, Christopher A. and Tao A. Zha. 1998. "Bayesian Methods for Dynamic Multivariate Models" International Economic Review 39(4):949-968.

Sims, Christopher A. and Tao A. Zha. 2006. "Were There Regime Switches in U.S. Monetary Policy?" American Economic Review. 96(1):54–81.

See Also

gibbs.msbvar for the MCMC sampler after using this function, szbvar for a non-switching, Bayesian VAR and more details.

Examples

```
## Not run:
# Simple replication of Hamilton (1989) as in
# Kim and Nelson (1999: 79, 220)

data(HamiltonGDP)
set.seed(214)

m2 <- msbvar(HamiltonGDP, p=1, h=2,
             lambda0=0.8, lambda1=0.15, lambda3=1, lambda4=0.25,
             lambda5=1, mu5=0, mu6=0, qm=12,
             alpha.prior=c(100, 30)*diag(2) +
             matrix(12, 2, 2), prior=0, max.iter=30,
             initialize.opt=NULL)

# Now plot the filtered probabilities of a recession
# Compare to Kim and Nelson (1999: 79, 220)

fp.rec <- ts(m2$fp[,1], start=tsp(HamiltonGDP)[1],
             freq=tsp(HamiltonGDP)[3])
plot(fp.rec)


## End(Not run)
```

---

msvar *Markov-switching vector autoregression (MSVAR) estimator*

Description

Estimates a Markov-swtiching vector autoregression (MSVAR) model with h regimes (states) by maximum likelihood. The Hamilton filtering algorithm is used to estimate the regimes. The numerical optimization to compute the MLE is based on the block-wise algorithm of Sims, Waggoner and Zha (2008).

Usage

```
msvar(Y, p, h, niterblkopt = 10)
```

Arguments

`Y`	$T \times m$ multiple time series object created with `ts()`.
`p`	Lag length, an integer
`h`	Number of regimes / states, an integer
`niterblkopt`	Number of iterations to allow for the block-wise optimization.

Details

This function computes ML estimates for an MSVAR(p,h) model where p is the number of lags and h is the number of regimes. The model is estimated using the block-wise algorithm of Sims, Waggoner, and Zha (2008). This ML optimization algorithm splits the parameter space of the MSVAR model into separate block components: (1) the transition matrix Q, (2) the intercepts, (3) the autoregressive coefficients, (4) the error covariances. The algorithm does 4 separate optimizations for each `niterblkopt` calls. Each component of the model is optimized separately over the `niterblkopt` values using separate calls to `optim`. Within each `optim` call, Fortran code is used to do the work of the filtering algorithm for the regimes in the model

Value

A list of class `MSVAR` and the appropriate inputs objects to feed the results into subsequent functions like `gibbs.msbvar` (though you should use `msbvar` and specify a prior!).

`init.model`	Description of 'comp1'
`hreg`	Description of 'comp2'
`Q`	$h \times h$ Markov transition matrix
`fp`	$T \times h$ Transition probability matrix
`m`	Integer, number of equations
`p`	Integer, number of lags
`h`	Integer, number of regimes
`llfval`	Vector of length `niterblkopt`
`DirectBFGSLastSuccess`	`optim` convergence code returned in the last optimization used in the last block-wise optimization

Note

Consult the `msbvar` function for more details on the model. This function is only included as a baseline or helper to the overall estimation goal of fitting MSBVAR models.

Author(s)

Patrick T. Brandt and Ryan Davis

References

Hamilton, James. 1989. "A new approach to the economic analysis of nonstationary time series and the business cycle." Econmetrica, 357–384.

Sims, Christopher A. and Daniel F. Waggoner and Tao Zha. 2008. "Methods for inference in large multiple-equation Markov-switching models" Journal of Econometrics 146(2):255–274.

See Also

`msbvar` for the Bayesian estimator, `szbvar` for the Bayesian, non-regime-switching version, `gibbs.msbvar` for posterior sampling.

Examples

```
## Not run:
# Simple replication of Hamilton (1989) as in
# Kim and Nelson (1999: 79, 220)

data(HamiltonGDP)
set.seed(1)

m2 <- msvar(HamiltonGDP, p=1, h=2, niterblkopt=20)

# Now plot the filtered probabilities of a recession
# Compare to Kim and Nelson (1999: 79, 220)

fp.rec <- ts(m2$fp[,1], start=tsp(gdp)[1], freq=tsp(gdp)[3])
plot(fp.rec)


## End(Not run)
```

---

normalize.svar *Likelihood normalization of SVAR models*

Description

Computes various sign normalizations of Bayesian structural VAR (B-SVAR) models.

Usage

```
normalize.svar(A0unnormalized, A0mode,
                method = c("DistanceMLA", "DistanceMLAhat",
                           "Euclidean", "PositiveDiagA",
                           "PositiveDiagAinv", "Unnormalized"),
                switch.count = 0)
```

Arguments

`A0unnormalized`	$m \times m$ unnormalized matrix value of A_0 in an B-SVAR
`A0mode`	$m \times m$ matrix of the A_0 to normalize around
`method`	string that selects the normalization method
`switch.count`	counter that counts the number of sign switches. Can be non-zero if you want to track the sign switches iteratively.

Details

The likelihood of VAR models are invariant to sign changes of the structural equation coefficients across equations. Thus a VAR with m equations has a likelihood with 2^m identical peaks, each a different set of signs (but with the same posterior peak). Normalization is used to choose among these peaks. The most common choice is to select the peak where the diagonal elements of A_0 are all positive, but will not be possible in all cases since no such normalization may exist. Thus, one should select a single peak and map all of the draws back to that peak.

The available normalization methods are 1) "DistanceMLA" : normalize around the ML peak of A0mode, 2) "DistanceMLAhat" : normalize around the ML peak of inv(A0mode) 3) "Euclidean" : normalize by minimizing the distance between the two matrices. 4) "PositiveDiagA" : normalize by making the diagonal positive 5) "PositiveDiagAinv" : normalize by making the diagonal of inv(A0) positive. 6) "Unnormalized" : no normalization is performed and the function returns A0 unnormalized.

Value

A list with two elements

`A0normalized`	$m \times m$ matrix, the normalized value of A_0 according to the selected normalization rule.
`switch.count`	Number of signs changed in the normalization

Note

This function is called in `gibbs.A0.BSVAR`, the Gibbs sampling of `szbsvar` models. In those functions, the A_0 produced by `szbsvar` is unnormalized. The Gibbs sampled draws are then normalized using the "DistanceMLA" method, which is consistent with the positive system shocks typically seen in the literature, if such a normalization exists. Note that Waggoner and Zha prefer the "DistanceMLA" method.

Author(s)

Patrick T. Brandt

References

Waggoner, Daniel F. and Tao A. Zha. 2003a. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

Waggoner, Daniel F. and Tao A. Zha. 2003b. "Likelihood preserving normalization in multiple equation models". *Journal of Econometrics*. 114: 329–347.

See Also

szbsvar, gibbs.A0

`null.space` *Find the null space of a matrix*

Description

Computes the null space of A for an arbitrary linear system of the form Ax = b.

Usage

```
null.space(x)
```

Arguments

x	$m \times n$ A matrix of a linear system $Ax = b$

Details

Computes the null space via singular value decomposition (SVD) of A by finding the columns of the SVD of A that correspond to the non-singular column vectors that span A.

Value

Returns an $m \times q$ matrix that is the null space, where q is the rank of A.

Author(s)

Patrick T. Brandt

See Also

svd

plot.forc.ecdf — *Plots VAR forecasts and their empirical error bands*

Description

Plots mean VAR forecasts and pointwise error bands

Usage

```
## S3 method for class 'forc.ecdf'
plot(x, probs = c(0.05, 0.95),
                xlab = "", ylab = "", ylim = NA, ...)
```

Arguments

`x`	N x nstep matrix of forecasts
`probs`	width of error band probabilities, default is 90% quantiles or `c(0.05,0.95)`
`xlab`	x-axis labels
`ylab`	y-axis labels
`ylim`	Bounds for y-axis in standard format `c(lower,upper)`
`...`	other plot parameters

Details

Plots the mean forecast and the pointwise empirical confidence region for a posterior sample of VAR forecasts.

Value

None.

Author(s)

Patrick T. Brandt

See Also

`plot.forecast`

Examples

```
## Not run:
data(IsraelPalestineConflict)

# Fit a BVAR model
fit.BVAR <- szbvar(IsraelPalestineConflict, p=6, z=NULL, lambda0=0.6,
                   lambda1=0.1, lambda3=2, lambda4=0.5, lambda5=0,
```

```
                       mu5=0, mu6=0, nu=3, qm=4, prior=0,
                       posterior.fit=FALSE)

# Generate unconditional forecasts for both models
forecast.BVAR <- uc.forecast(fit.BVAR, nsteps=12,
                             burnin=100, gibbs=1000)

# Plot the forecasts
par(mfrow=c(2,1))

plot(forecast.BVAR$forecast[,,1], probs=c(0.16,0.84),
               main="I2P Forecast")
abline(h=0)

plot(forecast.BVAR$forecast[,,2], probs=c(0.16,0.84),
               main="P2I Forecast")
abline(h=0)

## End(Not run)
```

`plot.forecast` *Plot function for forecasts*

Description

Generates simple plots of forecasts obtained from forecast.VAR / forecast.BVAR / forecast.B-SVAR

Usage

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

Arguments

`x`	Plots generated from forecast genrated through fitted VAR, BVAR, or B-SVAR model from `forecast`.
`...`	Other graphics parameters

Details

Generates a plot in the current graphics device for the m time series in the respective (B)VAR model.

Value

None. Generates a plot in the current graphics device.

Author(s)

Patrick T. Brandt

See Also

summary

Examples

```
## Not run:
plot(x)

## End(Not run)
```

---

plot.gibbs.A0 *Plot a parameter density summary for B-SVAR A(0) objects*

Description

Generates an $m \times m$ matrix of density plots for each free parameter in an szbsvar A_0 object produced by gibbs.A0, with associated highest posterior density (HPD) regions.

Usage

```
## S3 method for class 'gibbs.A0'
plot(x, hpd = 0.68, varnames=attr(x, "eqnames"), ...)
```

Arguments

x	An A_0 posterior object created by szbsvar.
hpd	Probability width of the highest posterior density region, default is 0.68 or approximately one standard deviation around the mode of the parameter
varnames	List of variable names for labeling the equations and variables. Default are the names of the variables for the input data to szbsvar as fed through gibbs.A0. For an SVAR, users often want to relabel these as economic sectors or groups of actors for the time series and this is the place this can be done.
...	optional graphics arguments

Details

This function plots an $m \times m$ matrix of densities for the posterior of the A_0 free parameters for a B-SVAR model. The plot is arranged such that the unrestricted parameters for each contemporaneous effect of each variable on an equation are in the row for that equation. So the first row shows densities for the contemporaneous effects of the column variables (as in an impulse response plot like plot.irf or plot.mc.irf). Elements of A_0 that were restricted to zero are left empty in the matrix of densities. The pattern of the densities will match the *tranpose* of the ident matrix passed to szbsvar.

Highest posterior density regions are plotted using Hyndman's 91996) density quantile algorithm. These HPDs are defined by a set of vertical bars over the HPD interval. The vertical line in each plot measures the value of the density at the boundaries of the HPD region. The HDR is superimposed at the bottom of each density.

Value

None. Main purposed is to plot density summaries and HPDs for each of the free parameters in an A_0 matrix.

Note

The plot will tend to be large, so be sure to adjust the size of your plotting device accordingly so things are visible.

Author(s)

Patrick T. Brandt

References

Hyndman, Rob J. 1996. "Computing and Graphic Highest Density Regions", The American Statistician, 50(2):120–126

HPD code is borrowed from Hyndman's hdrcde package, version 2.07.

See Also

`plot.mcmc`, `summary.mcmc`, and `A02mcmc`.

Examples

```
# SZ, B-SVAR model for the Levant data
data(BCFdata)
m <- ncol(Y)
ident <- diag(m)
ident[1,] <- 1
ident[2,1] <- 1

# estimate the model's posterior moments
set.seed(123)
model <- szbsvar(Y, p=2, z=z2, lambda0=0.8, lambda1=0.1, lambda3=1, lambda4=0.1,
                 lambda5=0.05, mu5=0, mu6=5, ident, qm=12)

# Set length of burn-in and size of posterior.  These are only an
# example.  Production runs should set these much higher.
N1 <- 1000
N2 <- 10000

A0.posterior.obj <- gibbs.A0(model, N1, N2, thin=1)

# Plot the matrix of the densities
dev.new()
plot.gibbs.A0(A0.posterior.obj, hpd=0.68, varnames=colnames(Y))
```

plot.irf *Plots impulse responses*

Description

Plots the m x m matrix of impulse responses produced by `irf`.

Usage

```
## S3 method for class 'irf'
plot(x, varnames = attr(x, "eqnames"), ...)
```

Arguments

`x`	Impulse response object produced by `irf`
`varnames`	Names of equations and shocks in the format `c("name1","name2",...)`. Default is to use the names of the input variables from the estimation method.
`...`	other plot arguments

Details

Generates a plot in ther current plotting device of the impulse responses in `irf`. See below for functions that allow one to add error bands and confidence regions to the impulse responses. Impulses or shocks are in the columns and the rows are the responses.

Value

None. Draws a graph in the current device.

Note

This function should NOT be used for Monte Carlo samples of IRFs. Use `plot.mc.irf` for this purpose.

Author(s)

Patrick T. Brandt

References

Hamilton, James. 1994. Time Series Analysis, Chapter 11.

Sims, C.A. 1980. "Macroeconomics and Reality" Econometrica.

See Also

`irf` to produce impulse responses from a VAR object, `mc.irf`, and `plot.mc.irf` for methods that allow frequentist and Bayesian error bands in the impulse responses

Examples

```
data(IsraelPalestineConflict)
rf.var <- reduced.form.var(IsraelPalestineConflict, p=6)
plot(irf(rf.var, nsteps = 12))
```

---

plot.mc.irf *Plotting posteriors of Monte Carlo simulated impulse responses*

Description

Provides a plotting method for the mc.irf Monte Carlo sample of impulse responses. Responses can be plotted with classical or Bayesian error bands, as suggested by Sims and Zha (1999).

Usage

```
## S3 method for class 'mc.irf'
plot(x, method=c("Sims-Zha2"), component=1,
                        probs=c(0.16,0.84), varnames = attr(x, "eqnames"),
                        regimelabels=NULL, ask=TRUE, ...)
```

Arguments

x	Output of the mc.irf function
method	Method to be used for the error band construction. Default method is to use the eigendecomposition method proposed by Sims and Zha. Defined methods are "Percentile" (error bands are based on percentiles specified in probs), "Normal Approximation" (Gaussian approximation for interval of width probs), "Sims-Zha1" (Gaussian approximation with linear eigendecomposition), "Sims-Zha2" (Percentiles with eigendecomposition for each impulse response function), "Sims-Zha3" (Percentiles with eigendecomposition of the full stacked impulse responses)
component	If using one of the eigendecomposition methods, the eigenvector component to be used for the error band construction. Default is the first or largest eigenvector component.
probs	is the width of the error bands. Default is c(0.16, 0.84) which is a 68% band that is approximately one standard deviation, as suggested by Sims and Zha.
varnames	List of variable names of length m for labeling the impulse responses. Default are the input variable names from the relevent estimation method.
regimelabels	For MSBVAR models from mc.irf, a character vector of length h for the regime-specific IRFs. Default of NULL leads to automatic generation of "Regime 1", "Regime 2", etc.
ask	Default = TRUE, ask before showing the next regime's IRFs for MSBVAR models?
...	Other graphics parameters.

Details

This function plots the output of a Monte Carlo simulation of (MS)(B)(BS)VAR impulse response functions produced by `mc.irf`. The function allows the user to choose among a variety of frequentist (normal appproximation and percentile) and Bayesian (eigendecomposition) methods for constructing error bands around a set of impulse responses. Impulses or shocks are in the columns and the rows are the responses.

Value

The primary reason for this function is to plot impulse responses and their error bands. Secondarily, it returns an invisible list of the impulses responses, their error bands, and summary measures of the fractions of the variance in the eigenvector methods that explain the total variation of each response.

`responses` Responses and their error bands

`eigenvector.fractions`
Fraction of the variation in each response that is explained by the chosen eigenvectors. `NULL` for non-eigenvector methods.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Sims, C.A. and Tao Zha. 1999. "Error Bands for Impulse Responses." *Econometrica*. 67(5): 1113-1156.

See Also

See Also `mc.irf` for the computation of Monte Carlo samples of impulse responses, `szbsvar` for estimation of the posterior moments of the B-SVAR model, `gibbs.A0` for Gibbs sampling the posterior of the A_0 for the model, and

Examples

```
## Not run:
data(IsraelPalestineConflict)
fit.BVAR <- szbvar(IsraelPalestineConflict, p=6, z=NULL, lambda0=0.6,
                   lambda1=0.1, lambda3=2, lambda4=0.5, lambda5=0,
                   mu5=0, mu6=0, nu=3, qm=4, prior=0,
                   posterior.fit=FALSE)

posterior.impulses <- mc.irf(fit.BVAR, nsteps=12, draws=1000)
plot(posterior.impulses, method = c("Percentile"))

## End(Not run)
```

plot.ms.irf *Color plot of MSBVAR impulse response functions*

Description

Provides an overplotted, color-coded version of the MSBVAR IRFs plot. This is an experimental function using color rather than the separate plots produced in `plot.mc.irf`

Usage

```
## S3 method for class 'ms.irf'
plot(x, method = "Sims-Zha2", component = 1,
          probs = c(0.16, 0.84), varnames = attr(x, "eqnames"), ...)
```

Arguments

x	Output of the `mc.irf` function for an MSBVAR model via `gibbs.msbvar`
method	Method to be used for the error band construction. Default method is to use the eigendecomposition method proposed by Sims and Zha. Defined methods are "Percentile" (error bands are based on percentiles specified in `probs`), "Normal Approximation" (Gaussian approximation for interval of width `probs`), "Sims-Zha1" (Gaussian approximation with linear eigendecomposition), "Sims-Zha2" (Percentiles with eigendecomposition for each impulse response function), "Sims-Zha3" (Percentiles with eigendecomposition of the full stacked impulse responses)
component	If using one of the eigendecomposition methods, the eigenvector `component` to be used for the error band construction. Default is the first or largest eigenvector component.
probs	is the width of the error bands. Default is `c(0.16, 0.84)` which is a 68% band that is approximately one standard deviation, as suggested by Sims and Zha.
varnames	List of variable names of length m for labeling the impulse responses. Default are the input variable names from the relevent estimation method.
...	Other graphics parameters.

Details

This function plots the output of a Monte Carlo simulation of MSBVAR impulse response functions produced by `mc.irf`. The function allows the user to choose among a variety of frequentist (normal appproximation and percentile) and Bayesian (eigendecomposition) methods for constructing error bands around a set of impulse responses. Impulses or shocks are in the columns and the rows are the responses. Here the plot colors the responses for each reqime, per the R default color pallette for colors `1:h`.

Value

The primary reason for this function is to plot impulse responses and their error bands. Secondarily, it returns an invisible list of the impulses responses, their error bands, and summary measures of the fractions of the variance in the eigenvector methods that explain the total variation of each response.

`responses` Responses and their error bands

`eigenvector.fractions`
Fraction of the variation in each response that is explained by the chosen eigenvectors. `NULL` for non-eigenvector methods.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Sims, C.A. and Tao Zha. 1999. "Error Bands for Impulse Responses." *Econometrica*. 67(5): 1113-1156.

See Also

plot.mc.irf

Examples

```
## Not run:
data(IsraelPalestineConflict)
m1 <- msbvar(IsraelPalestineConflict, p=1, h=2, lambda0=0.6,
             lambda1=0.1, lambda3=1, lambda4=0.5, lambda5=0,
             mu5=0, mu6=0, qm=12, alpha.prior=matrix(10, 2, 2),
             prior=0, max.iter=20)
m2p <- gibbs.msbvar(m1, N1=1000, N2=10000, permute=FALSE, Sigma.idx=1)

irf2 <- mc.irf(m2p, nsteps=12)
plot.ms.irf(irf2)


## End(Not run)
```

plotregimeid *Clustering and plotting function for msbvar permuted sample output*

Description

Identifies and plots regime-specific coefficients from the random permutation sampler for regime identification

Usage

```
plotregimeid(x,
             type = c("all", "intercepts", "AR1", "Sigma", "Q"),
             ask = TRUE, ...)
```

Arguments

`x`	Gibbs sampler output of class `MSBVAR` from the posterior of an MSBVAR model, a call to the `gibbs.msbvar` function.
`type`	Items to be clustered and plots to be produced to identify the posterior regimes / modes of the Gibbs sampler based on the randomly permuted draws. The `type` can be "intercepts" where the clustering of the posterior draws and the plots are based on the intercepts in each equation (so a change in equilibrium model), "AR1" where the clustering of the posterior draws are based on the coefficients in the VAR(1) matrices across the regimes, "Sigma" or the variances of the equations across the regimes, or "Q" based on the elements of the transition matrix, Q. The option "all" generates the plots and clustering for all of the above options and is the default.
`ask`	logical, `default=TRUE`. Ask about which plots to show, ala the syntax in `coda`. If `TRUE` then all relevant responses are displayed on the current graphical device with user input. Otherwise, all plots run by in the current device as generated.
`...`	Optional graphical and lattice parameters to be fed to the plots. There is no assurance that these will work. E-mail if you have inputs on this that do not work, but that you think should.

Details

The posterior of a Markov-switching (MS) model estimated by an unrestricted Gibbs sampler has $h!$ identical posterior modes. The modes are identical in the sense that they are merely relabelings of the regime labels. Since the analyst may not apriori know what defines or separates the regimes in the parameter space, this function allows one to explore the randomly permuted labelings that are generated by the `gibbs.msbvar` function.

This function takes the permuted output of `gibbs.msbvar` and shows colored pairs, scatter, densityplots, and traceplots for the posterior parameters. The coloring follow standard R color pallates. The determination of how the regimes are identified is based on a `kmeans` clustering of either the the parameters "intercepts", "AR1", "Sigma" (variances), or "Q" transition probabilities. This is the method suggested by Fruhwirth-Schanatter (2001, 2006). The utility here is that this function handles subsetting the data, setting up the clustering and plotting and labeling the results for the user.

Regime identification and labeling is necessary so that one can sample from a single mode of the posterior to get sensible regime classification plots from say `plot.SS` or regime probabilities from say `mean.SS`.

Value

None. A series of plots are produced in the current graphics device.

Note

This is the first version of this function. Future versions may use a slightly different syntax and only use one input argument.

Author(s)

Patrick T. Brandt

References

Fruhwirth-Schnatter, Sylvia. 2001. "Markov Chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models". Journal of the American Statistical Association. 96(153):194–209.

Fruhwirth-Schnatter, Sylvia. 2006. Finite Mixture and Markov Switching Models. Springer Series in Statistics New York: Springer.

See Also

`msbvar`, `plot.SS`, `mean.SS`, `gibbs.msbvar`

Examples

```
## Not run:
# This example can be pasted into a script or copied into R to run.  It
# takes a few minutes, but illustrates how the code can be used

data(IsraelPalestineConflict)

# Find the mode of an msbvar model
# Initial guess is based on random draw, so set seed.
set.seed(123)

xm <- msbvar(IsraelPalestineConflict, p=1, h=2,
             lambda0=0.8, lambda1=0.15,
             lambda3=2, lambda4=1, lambda5=0, mu5=0,
             mu6=0, qm=12,
             alpha.prior=matrix(c(100,40,30,50), 2, 2))

# Plot out the initial mode
plot(ts(xm$fp))
```

```
print(xm$Q)

# Now sample the posterior
N1 <- 100
N2 <- 500

# First, so this with random permutation sampling
x1 <- gibbs.msbvar(xm, N1=N1, N2=N2, permute=TRUE)

# Identify the regimes using clustering in plotregimeid()
plotregimeid(x1, type="all")

# Now re-estimate based on desired regime identification seen in the
# plots.  Here we are using the variance of the first equation, so
# Sigma.idx=1.

x2 <- gibbs.msbvar(xm, N1=N1, N2=N2, permute=FALSE, Sigma.idx=1)

# Plot the variances.  Note the strict hyperplane between the variances
# for the first equation versus the others.
plotregimeid(xm, x2, type="Sigma")

## End(Not run)
```

---

posterior.fit *Estimates the marginal likelihood or log posterior probability for BVAR, BSVAR, and MSBVAR models*

Description

Computes the marginal log likelihood other posterior fit measures for BVAR, BSVAR, and MSBVAR models fit with szbvar, szbsvar and, msbvar (and their posterior samplers).

Usage

```
posterior.fit(varobj, A0.posterior.obj=NULL, maxiterbs=500)
```

Arguments

varobj — Varies for BVAR, BSVAR, or MSBVAR models. For a BVAR model, varobj = output from a call to szbvar. For a BSVAR model, varobj = output from a call to szbsvar. For MSBVAR models, varobj = output from a call to gibbs.msbvar.

A0.posterior.obj — MCMC Gibbs object for the B-SVAR model A_0 from gibbs.A0

maxiterbs — Number of iterations for the bridge sampler for computing the marginal likelihood for MSBVAR models

Details

Estimates the marginal log likelihood, also known as a log marginal data density for the various models. For the BVAR models, this can be computed in closed form. For the BSVAR models the MCMC data augmentation method of Chib (1995) is employed. For the MSBVAR models, the importance sampler, reciprocal importance sampler, and bridge sampler methods of Fruwirth-Schnatter (2006) are used. Consult these references for details (or look at the source code).

The computations are done using compiled C++ and Fortran code as of version 0.3.0. See the package source code for details about the implementation.

Value

BVAR:

A list of the class "posterior.fit.VAR" that includes the following elements:

`data.marg.llf` Log marginal density, the probability of the data after integrating out the parameters in the model.

`data.marg.post` Predictive marginal posterior density

`coefficient.post` Contribution to the posterior fit from the pdf of the coefficients.

BSVAR:

A list of the class "posterior.fit.BSVAR" that includes the following elements:

`log.prior` Log prior probability

`log.llf` $T \times 1$ list of the log probabilities for each observation conditional on the parameters.

`log.posterior.Aplus` Log marginal probability of $A_1, \ldots, A_p$ conditional on the data and A_0

`log.marginal.data.density` Log data density or marginal log likelihood, the probability of the data after integrating out the parameters in the model.

`log.marginal.A0k` $m \times 1$ list of the log probabilities of each column (corresponding to the equations) of A_0 conditional on the other columns.

MSBVAR:

A list of the class "posterior.fit.MSBVAR" that includes the following elements:

Note

The log Bayes factor for two model can be computed using the log.marginal.data.density:

log BF = log.marginal.data.density.1 - log.marginal.data.density.2

Note that at present, the scale factors for the BVAR and B-SVAR models are different (one used the concentrated likelihood, the other does NOT). Thus, one cannot compare fit measures across the two functions. To compare a recursive B-SVAR to a non-recursive B-SVAR model, one should estimate the recursive model with `szbsvar` using the appropriate `ident` matrix and then call `posterior.fit` on the two B-SVAR models!

Author(s)

Patrick T. Brandt and W. Ryan Davis

References

Chib, Siddartha. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90(432): 1313–1321.

Waggoner, Daniel F. and Tao A. Zha. 2003. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

Fruhwirth-Schnatter, Sylvia. 2006. Finite Mixture and Markov Switching Models. Springer Series in Statistics New York: Springer., esp. Sections 5.4 and 5.5.

See Also

`szbvar`, `szbsvar`, `gibbs.A0`, `gibbs.msbvar`, and `print.posterior.fit` for a print method.

Examples

```
## Not run:
varobj <- szbsvar(Y, p, z = NULL, lambda0, lambda1, lambda3, lambda4,
                  lambda5, mu5, mu6, ident, qm = 4)
A0.posterior <- gibbs.A0(varobj, N1, N2)
fit <- posterior.fit(varobj, A0.posterior)
print(fit)

## End(Not run)
```

`print.dfev` *Printing DFEV tables*

Description

Prints decomposition of forecast error variance tables

Usage

```
## S3 method for class 'dfev'
print(x, latex = F, file = NULL, ...)

## S3 method for class 'dfev'
summary(object, latex = F, file = NULL, ...)
```

Arguments

`x`	DFEV object created by `dfev`
`object`	DFEV object created by `dfev`
`latex`	Logical. T = format results in LaTeX tables, default is F, text output
`file`	File for the results. If `NULL`, prints to standard output device.
`...`	Other print and summary arguments

Details

Prints DFEV results in a table using `xtable` functions.

Value

None.

Author(s)

Patrick T. Brandt

See Also

See `dfev` for an example.

`print.posterior.fit` *Print method for posterior fit measures*

Description

Prints objects of the classes "posterior.fit.VAR", "posterior.fit.BVAR", and "posterior.fit.BSVAR".

Usage

```
## S3 method for class 'posterior.fit'
print(x, ...)
```

Arguments

`x`	object produced by `posterior.fit`,
`...`	other print options

Details

Called for its side effect — printing the output of `posterior.fit`

Value

None

Author(s)

Patrick T. Brandt

See Also

szbvar, szbsvar, gibbs.A0, gibbs.msbvar, mc.irf, posterior.fit

Examples

```
## Not run:
varobj <- szbsvar(Y, p, z = NULL, lambda0, lambda1, lambda3, lambda4,
                  lambda5, mu5, mu6, ident, qm = 4)
A0.posterior <- gibbs.A0(varobj, N1, N2)
fit <- posterior.fit(varobj, A0.posterior)
print(fit)

## End(Not run)
```

---

rdirichlet *Random draws from and density for Dirichlet distribution*

Description

Generate draws from a random Dirichlet distribution or compute its density.

Usage

```
rdirichlet(n, alpha)
ddirichlet(x, alpha)
```

Arguments

n	Number of draws
alpha	Scale matrix, $h \times h$
x	value to compute density

Details

Draws n values for an $h \times h$ Dirichlet random variable or computes the density.

Value

x	An $n \times h$ matrix of the draws

Note

Based on code from Kevin Quinn in the MCMCpack package and the gregmisc package.

References

MCMCpack and gregmisc

Examples

```
rdirichlet(2, matrix(rep(1, 4), 2, 2))
```

`reduced.form.var` *Estimation of a reduced form VAR model*

Description

Estimates a reduced form VAR using equation-by-equation seemingly unrelated regression (SUR).

Usage

```
reduced.form.var(Y, p, z=NULL)
```

Arguments

`Y`	$T \times m$ multiple time series object created with `ts()`.
`p`	Lag length
`z`	$T \times k$ exogenous variables in a matrix of T rows. Can be `NULL` if there are none.

Details

This is a frequentist VAR estimator. This is a workhorse function — you will want to use other functions such as `irf`, `mc.irf` or `dfev` to report and interpret the results of this object.

Value

List of class "VAR" with elements,

`intercept`	Row vector of the m intercepts.
`ar.coefs`	$m \times m \times p$ array of the AR coefficients. The first $m \times m$ array is for lag 1, the p'th array for lag p.
`Bhat`	$(mp + k + 1) \times m$ matrix of the coefficients, where the columns correspond to the variables in the VAR. Intercepts follow the AR coefficients, etc.
`exog.coefs`	$k \times m$ matrix of exogenous coefficients, or `NA` if `z=NULL`
`vcv`	$m \times m$ matrix of the maximum likelihood estimate of the residual covariance
`mean.S`	$m \times m$matrix of the posterior residual covariance.

hstar	$mp \times mp$ right hand side variables crossproduct.
X	Right hand side variables for the estimation of BVAR
Y	Left hand side variables for the estimation of BVAR
y	Input data (Y)

Author(s)

Patrick T. Brandt

References

Sims, C.A. 1980. "Macroeconomics and Reality" *Econometrica* 48(1): 1-48.

See Also

See also `szbvar` for BVAR models with the Sims-Zha prior and `szbsvar` for Bayesian SVAR models with the Sims-Zha prior.

Examples

```
data(IsraelPalestineConflict)
rf.var <- reduced.form.var(IsraelPalestineConflict, p=6)
plot(irf(rf.var, nsteps=12))
```

---

regimeSummary — *Regime probability summaries and regime duration estimates based on MCMC output for MSBVAR models*

Description

Provides summary and quantile computations for regime probabilities and regime durations based on MSBVAR MCMC output

Usage

```
regimeSummary(x, quantiles = c(0.025, 0.25, 0.5, 0.75, 0.975))
```

Arguments

x	output from `gibbs.msbvar`, the MCMC sampler for the MSBVAR models
quantiles	quantiles one wants to compute, as is done in the coda package. Defaults are as given above.

Details

This function is mainly a wrapper for calls to the `coda` package to summarize the MCMC output for the transition matrix of an MSBVAR model estimated from `gibbs.msbvar`. It adds labels to the output so one know which regime is which in the output. In the summary of the transition matrix $Q's$ elements q_{ij} for the transition from regime i to regime j.

The ergodic regime probabilities are computed for draw k of the MSBVAR MCMC sampler as described in Kim and Nelson (1999):

eta <- solve(rbind(cbind(diag(h-1) - t(Q)[1:(h-1),1:(h-1)], t(Q)[1:(h-1),h]), rep(1,h)))

This is the gives $N2$ draws of the ergodic probabilities of being in regime k. These are summarized again using `coda` functions.

Finally, the ergodic regime probabilities can be used to estimate expected long run regime durations. For η_k the expected regime duration is $1/(1-\eta_k)$. This again is summarized over the $N2$ draws using `coda` functions.

Value

Invisible list with 3 elements:

`Q.summary`	Summary and quantiles of the `x$Q.sample` draws of the transition matrix
`lrQ`	Summary and quantiles of the long run or ergodic regime probabilities
`durations`	Summary and quantiles of the estimated regime durations

Author(s)

Patrick T. Brandt

References

Kim, Chang-Jin and Charles R. Nelson. 1999. State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications. Cambridge: MIT Press.

See Also

`gibbs.msbvar`, `plotregimeid`, `msbvar`

Examples

```
## Not run:
regimeSummary(x)

## End(Not run)
```

restmtx — *Utility function for generating the restriction matrix for hard condition forecasting*

Description

Generates the restriction matrix for a set of hard condition forecasts. See `hc.forecast` for details.

Usage

```
restmtx(nsteps, m)
```

Arguments

`nsteps`	Number of periods in the forecast horizon
`m`	Number of endogenous variables in the VAR.

Details

Builds the appropriately dimensioned and filled restriction matrix of zeros and ones for hard condition forecasting.

Value

A matrix of dimensions (nsteps x m*nsteps) that can be used to represent the restrictions in hard condition forecasting using `hc.forecast`

Author(s)

Patrick T. Brandt

References

Waggoner, Daniel F. and Tao Zha. 1999. "Conditional Forecasts in Dynamic Multivariate Models" *Review of Economics and Statistics*, 81(4):639-651.

See Also

`hc.forecast`

rmse — *Root mean squared error of a Monte Carlo / MCMC sample of forecasts*

Description

Computes the root mean squared error (RMSE) of a Monte Carlo sample of forecasts.

Usage

```
rmse(m1, m2)
```

Arguments

m1	Forecast sample for model 1
m2	Forecast sample for model 2

Details

User needs to subset the forecasts if necessary.

Value

Forecast RMSE.

Author(s)

Patrick T. Brandt

See Also

mae, forecast

Examples

```
data(IsraelPalestineConflict)
Y.sample1 <- window(IsraelPalestineConflict, end=c(2002, 52))
Y.sample2 <- window(IsraelPalestineConflict, start=c(2003,1))

# Fit a BVAR model
fit.bvar <- szbvar(Y.sample1, p=6, lambda0=0.6, lambda1=0.1, lambda3=2,
                   lambda4=0.25, lambda5=0, mu5=0, mu6=0, prior=0)

# Forecast -- this gives back the sample PLUS the forecasts!

forecasts <- forecast(fit.bvar, nsteps=nrow(Y.sample2))

# Compare forecasts to real data
rmse(forecasts[(nrow(Y.sample1)+1):nrow(forecasts),], Y.sample2)
```

rmultnorm *Multivariate Normal Random Number Generator*

Description

Generates multivariate normal random variates for give mean and covariance vectors. Can also handle generation of multivariate normal deviates with singular covariance distributions via singular value decomposition (SVD).

Usage

```
rmultnorm(n, mu, vmat, tol = 1e-10)
```

Arguments

n	Number of variates to draw.
mu	m column matrix of multivariate means
vmat	$m \times m$ covariance matrix
tol	Tolerance level used for SVD of the covariance. Default is 1e-10

Details

Generates n draws from a multivariate normal distribution with mean matrix `mu` and covariance matrix `vmat`.

Value

Matrix of the random draw that is conformable with the input `mu`.

Note

Based on code by Jeff Gill. This function is called in the hard condition forecasting in `hc.forecast` for simulating the structural innovations.

Author(s)

Patrick T. Brandt

See Also

`rnorm`

Examples

```
rmultnorm(1, matrix(c(1,2),2,1), vmat=matrix(c(1,1,0,1),2,2))
```

`rwishart` *Random deviates from a Wishart distribution*

Description

Draws random deviates from a Wishart pdf.

Usage

```
rwishart(N, df, Sigma)
```

Arguments

`N`	Number of random deviates to draw.
`df`	Degrees of freedom for Wishart distribution
`Sigma`	Mean of the Wishart from which to draw the deviates

Details

Draws N matrices of draws from a Wishart with mean Sigma. This is used to draw error covariances for the VAR and BVAR models which are distributed inverse Wisharts deviates.

Value

Returns an N dimensional array of dim(Sigma) square matrices for the Wishart random deviates. If N=1, it returns a single matrix.

Author(s)

Patrick T. Brandt

See Also

See also as `rmultnorm` for multivariate normal deviates, `rgamma` for the univariate analog to drawing Wishart deviates, and `ldwishart` for computing the log density for a Wishart variate.

Examples

```
x <- matrix(rnorm(100), 50, 2)
XX <- crossprod(x)
tmp <- rwishart(1, 50, XX)
```

`simulateMSAR` *Simulate (univariate) Markov-switching autoregressive (MSAR) data*

Description

Simulate (univariate) Markov-switching autoregressive (MSAR) data

Usage

```
simulateMSAR(bigt, Q, theta, st1, y1)
```

Arguments

`bigt`	Integer, number of observations to generate.
`Q`	h dimensional transition matrix for the MS process. $h \times h$ Markov transition matrix whose rows sum to 1 with the main weights on the diagonal elements.
`theta`	Matrix of the MSAR coeffients with h rows and $m \times p + 2$ columns. The first column is the constants, the next $m \times p + 1$ columns are the autoregressive coefficients (by lag – so the first $m \times 1$ are the AR(1) coefficients, etc.) and the last $m \times 1$ elements are the error variances (remember, this is univariate!)
`st1`	Starting regime, an integer less than or equal to h
`y1`	Starting value for simulated data in regime `st1`

Details

This function simulates a univariate MSAR model. The user needs to input the transition matrix Q and the autoregression coefficients via $theta$. The assumption in this model is that the error process is Gaussian.

Value

A list with two elements:

`Y`	The simulated univariate MSAR time series
`st`	A vector of integers identifying the regime of each observation in `Y`

Author(s)

Patrick T. Brandt and Ryan Davis

References

Kim, Chang-Jin and Charles R. Nelson. 1999. State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications. Cambridge: MIT Press.

See Also

simulateMSVAR for the multivariate version

Examples

```
## Example of call here
```

---

simulateMSVAR *Simulate a Markov-switching VAR (MSVAR) process*

Description

Simulate Markov-switching vector autoregression data

Usage

```
simulateMSVAR(bigt, m, p, var.beta0, var.betas, e.vcv, Q, seed = 214)
```

Arguments

bigt	Integer, number of observations to generate.
m	Integer, number of equations in the VAR process
p	Integer, lag length of the VAR(p) process.
var.beta0	Array of dimension $m \times 1 \times h$ of the VAR intercepts for each regime (h)
var.betas	Array of dimension $m \times mp \times h$ of the autoregressive coefficients. In each element of the array, rows correspond to equations, columns to lags. The first $m \times m$ columns are the AR(1) coefficients, etc.
e.vcv	Array of dimentsion $m \times m \times h$ of the error covariances. The $m \times m$ matrices are the error covariances for each regime.
Q	h dimensional transition matrix for the MS process. $h \times h$ Markov transition matrix whose rows sum to 1 with the main weights on the diagonal elements.
seed	Integer. Random number seed.

Details

This function simulates a multivariate Markov-switching model, MSVAR with m equations, p lags and h regimes. The assumption is that the error process is Gaussian.

Value

A list with two elements:

Y	The simulated MSVAR time series in a `ts` object of dimension $bigt \times m$.
st	A vector of integers identifying the regime of each observation in `Y`

Author(s)

Patrick T. Brandt and Ryan Davis

References

Kim, Chang-Jin and Charles R. Nelson. 1999. State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications. Cambridge: MIT Press.

See Also

simulateMSAR for the univariate version; msbvar

Examples

```
## Not run:
# Example: simulate an MS(h)-VAR(p) model with two equations.
# Have h = 2, m=2, and p=1, simplest case

# VAR simulation parameters
bigt <- 500  # number of observations
m <- 2  # number of endogenous variables
p <- 1  # lag length
h <- 2  # number of regimes

# setup transition matrix with two states

Q <- matrix(c(.98, .02,
              .05, .95), nrow=h, byrow=TRUE)

# theta stores paramater values
# 1st column is intercept
# 2:m*p are the AR coefficients
# (mp+2)'th columns are variance

# regime 1
var.beta0.st1  <- c(1,2)  # intercepts
var.betas.st1  <- matrix(c(.7, .1,
                           .1, .7), m, byrow=TRUE)
# regime 2
var.beta0.st2  <- c(0,0)  # intercepts
var.betas.st2  <- matrix(c(.2, .1,
                           .2, .1), m, byrow=TRUE)

# Build the array
var.beta0 <- array(NA, c(m,1,h))
var.betas <- array(NA, c(m,p*m,h))
var.beta0[,,1] <- var.beta0.st1
var.beta0[,,2] <- var.beta0.st2
var.betas[,,1] <- var.betas.st1
var.betas[,,2] <- var.betas.st2

# Variance-Covariance Matrix for
```

```
# MVN distributed disturbances
# regime 1
e.vcv.st1  <- matrix(c(.3, .1,
                       .1, .3), 2)
# regime 2
e.vcv.st2  <- matrix(c(.1, .05,
                       .05, .1), 2)
# combine
e.vcv <- array(NA, c(m, m, h))
e.vcv[,,1] <- e.vcv.st1
e.vcv[,,2] <- e.vcv.st2

# hold true values of parameters for easy comparison to estimates
theta.true.var <- array(NA, c(m, 1+m*p+m, h))
theta.true.var[,1,] <- var.beta0
theta.true.var[,2:(1+p*m),] <- var.betas
theta.true.var[,(1+m*p+1):ncol(theta.true.var),] <- e.vcv

simdata <- simulateMSVAR(bigt, m, p, var.beta0, var.betas, e.vcv, Q)

# Plot
plot(as.ts(simdata[[1]]))

# Fit a simple model

model <- msvar(Y=simdata[[1]], p=1, h=2, niterblkopt=50)

# Plot regime estimates and compare to true simulated values
par(mfrow=c(2,1))
plot(ts(model$fp))
plot(ts(simdata$st))

## End(Not run)
```

`SS.ffbs` *State-space forward-filter and backwards-sampler for a Markov-switching VAR model*

Description

This function estimates the h state probabilities for all of the observations for a Gaussian likelihood

Usage

```
SS.ffbs(e, bigt, m, p, h, sig2, Q)
```

Arguments

e	$bigt \times m \times h$ array of the residuals for an MSBVAR process
bigt	integer, number of observations in the model

m	integer, number of equations or variables in the MSBVAR model
p	integer, number of lags in the model
h	integer, number of regimes in the MSBVAR model
sig2	$m \times m \times h$ array of the covariances for each regime (can be the same for each of the h regimes)
Q	$h \times h$ first order Markov transition matrix; each row must sum to 1

Details

The estimation of an MSBVAR model requires and efficient classifier of the states for the observed filtered probabilities. This function provides a way to accomplish this and is one of the workhorses in the estimation in the msbvar and gibbs.msbvar function.

This function uses compiled Fortran code to draw the 0-1 matrix of the regimes. It uses the Baum-Hamilton-Lee-Kim (BHLK) filter and smoother to estimate the regime probabilities. Draws are based on the standard forward-filter-backward-sample algorithm.

Value

A $T \times h$ matrix of the sampled regimes. Each row corresponds to an identity matrix element giving the regime classification for the observation.

Note

This function assumes that the innovation in the MSBVAR model are multivariate normal. The resulting filter and sample follows that in the references listed above. This function is provided so users can build their own customized MSBVAR models. Users can write functions to generate the (B)VAR residuals for their own customized MSBVAR models than then provide the residuals and their covariances and transition matrix Q. This function can then be used to estimate / sample the regime probabilities. So if you need an MSBVAR model where only certain parameters change — rather than all of them as in the existing msbvar and gibbs.msbvar functions — you can build your own estimator using this function. This function takes care of the hard part of building an MSBVAR model.

Author(s)

Patrick T. Brandt

References

Kim, C.J. and C.R. Nelson. 1999. State-space models with regime switching. Cambridge, Mass: MIT Press.

Krolzig, Hans-Martin. 1997. Markov-Switching Vector Autoregressions: Modeling, Statistical Inference, and Application to Business Cycle Analysis.

Sims, Christopher A. and Daniel F. Waggoner and Tao Zha. 2008. "Methods for inference in large multiple-equation Markov-switching models" Journal of Econometrics 146(2):255–274.

See Also

msbvar, gibbs.msbvar

Examples

```
# Simple example to show how data are input to the filter-sampler.
# Assumes a simple bivariate regression model with switching means and
# variances.

TT <- 100
h <- 2
m <- 2
set.seed(123)
x1 <- rnorm(TT)
x2 <- rnorm(TT)
y1 <- 5 + 2*x1 + rnorm(TT)
y2 <- 1 + x2 + 5*rnorm(TT)

Y <- rbind(cbind(y1[1:(0.5*TT)],y2[1:(0.5*TT)]),
           cbind(y2[((0.5*TT)+1):TT],y1[((0.5*TT)+1):TT]))
X <- rbind(cbind(x1[1:(0.5*TT)],x2[1:(0.5*TT)]),
           cbind(x2[((0.5*TT)+1):TT],x1[((0.5*TT)+1):TT]))

u1 <- Y - tcrossprod(cbind(rep(1,TT), X), matrix(c(5,2,0,1,1,0), 2, 3))
u2 <- Y - tcrossprod(cbind(rep(1,TT), X), matrix(c(1,1,0,5,2,0), 2, 3))

u <- array(0, c(TT, m, h))
u[,,1] <- u1
u[,,2] <- u2

Sik <- array(0, c(m,m,h))
Sik[,,1] <- diag(c(1,25))
Sik[,,2] <- diag(c(25,1))

Q <- matrix(c(0.9,0.2,0.1,0.8), h, h)

# estimate the states 100 times
ss <- replicate(100, SS.ffbs(u, TT, m, p=1, h, Sik, Q), simplify=FALSE)

# Get the state estimates from the 100 simulations
ss.est <- matrix(unlist(ss), nrow=(h*TT + h^2))

ss.prob <- matrix(rowMeans(ss.est[1:(h*TT),]), ncol=h)
ss.transition <- matrix(rowMeans(ss.est[((h*TT)+1):((h*TT) + h^2),]),
                        h, h)
```

---

summary *Summary functions for VAR / BVAR / B-SVAR model objects*

Description

Prints a summary of the coefficient matrices for various VAR / BVAR / B-SVAR model objects to standard output.

Usage

```
summary(object, ...)
```

Arguments

`object`	Fitted VAR, BVAR, or B-SVAR model from either `reduced.form.var`, `szbvar`, or `szbsvar`
`...`	other arguments

Details

Prints (posterior) coefficient matrices for each lag and error covariance summaries as appropriate.

Value

None.

Author(s)

Patrick T. Brandt

See Also

`summary`

Examples

```
## Not run:
summary(x)

## End(Not run)
```

`summary.forecast` *Summary functions for forecasts obtained through VAR / BVAR / B-SVAR model objects*

Description

Prints a summary of the mean and quantile values for the forecasts generated through VAR / BVAR / B-SVAR model objects to standard output.

Usage

```
## S3 method for class 'forecast'
summary(object, probs = c(0.16,0.84), ...)
```

Arguments

`object`	Forecast object genrated through fitting a VAR, BVAR, or B-SVAR model from either `forecast.VAR`, `forecast.BVAR`, or `forecast.BSVAR`
`probs`	vector list of probability range for quantiles. default: c(0.16,0.84) or a 68% region (approximately one standard deviation on each side of the mean)
`...`	optional arguments (ignored, but included for S3 consistency)

Details

Prints a summary of the mean and quantile values for the forecasts.

Value

Returns the mean forecast and the specified posterior probability interval for the forecasts.

Author(s)

Patrick T. Brandt

See Also

`summary`

Examples

```
## Not run:
summary(x)

## End(Not run)
```

SZ.prior.evaluation *Sims-Zha Bayesian VAR Prior Specification Search*

Description

Estimates posterior and in-sample fit measures for a reduced form vector autoregression model with different specifications of the Sims-Zha hyperparameters values.

Usage

```
SZ.prior.evaluation(Y, p,
                    lambda0, lambda1, lambda3, lambda4, lambda5,
                    mu5, mu6, z = NULL, nu = ncol(Y) + 1, qm,
                    prior = 0, nsteps, y.future)
```

Arguments

`Y`	T x m matrix of endogenous variables for the VAR
`p`	Lag length
`lambda0`	List of values, e,g, `c(0.7, 0.8, 0.9)` in [0,1], Overall tightness of the prior (discounting of prior scale).
`lambda1`	List of values, e,g, `c(0.05, 0.1, 0.2)` in [0,1], Standard deviation or tightness of the prior around the AR(1) parameters.
`lambda3`	List of values, e,g, `c(0, 1, 2)` for Lag decay (>0, with 1=harmonic)
`lambda4`	List of values, e,g, `c(0.15, 0.2, 0.5)` for Standard deviation or tightness around the intercept [>0]
`lambda5`	Single value for the standard deviation or tightness around the exogneous variable coefficients [>0]
`mu5`	Single value for sum of coefficients prior weight [>=0]
`mu6`	Single value for dummy Initial observations or cointegration prior [>=0]
`z`	Exogenous variables
`nu`	Prior degrees of freedom = m+1
`qm`	Frequency of the data for lag decay equivalence. Default is 4, and a value of 12 will match the lag decay of monthly to quarterly data. Other values have the same effect as "4"
`prior`	One of three values: 0 = Normal-Wishart prior, 1 = Normal-flat prior, 2 = flat-flat prior (i.e., akin to MLE)
`nsteps`	Number of periods in the forecast horizon
`y.future`	Future values of the series, nsteps x m for computing the root mean squared error and mean absolute error for the fit

Details

This function fits a series of BVAR models for the combinations of `lambda0`, `lambda1`, `lambda3`, and `lambda4` provided. For each possible value of these parameters specified, a Sims-Zha prior BVAR model is fit, posterior fit measures are computed, and forecasts are generated over `nsteps`. These `nstep` forecasts are then compared to a new set of data in `y.future` and root mean sqaured error and mean absolute error measures are computed.

Value

A matrix of the results with columns corresponding to the values of "lambda0", "lambda1", "lambda3", "lambda4", "lambda5", "mu5", "mu6", "RMSE", "MAE", "MargLLF","MargPosterior".

Note

The matrix of the results can be usefully plotted using the `lattice` package. See the example below.

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

See Also

`szbvar`

Examples

```
Y <- EuStockMarkets
results <- SZ.prior.evaluation(window(Y, start=c(1998, 1),
                                       end=c(1998,149)),
                               p=3,
                               lambda0=c(1,0.9),
                               lambda1=c(0.1,0.2),
                               lambda3=c(0,1),
                               lambda4=c(0.1,0.25),
                               lambda5=0,
                               mu5=4,
                               mu6=4, z=NULL,
                               nu=ncol(Y)+1, qm=4,
                               prior=0,
                               nstep=20,
                               y.future=window(Y, start=c(1998,150)))

# Now plot the RMSE and marginal posterior of the data for each of the
# 6 period forecasts as a function of the prior parameters.  This can
```

```
# easily be done using a lattice graphic.

library(lattice)

attach(as.data.frame(results))
dev.new()
xyplot(RMSE ~ lambda0 | lambda1 + lambda3)
dev.new()
xyplot(logMDD ~ lambda0 | lambda1 + lambda3)
dev.new()
xyplot(LLF ~ lambda0 | lambda1 + lambda3)
```

---

szbsvar *Structural Sims-Zha Bayesian VAR model estimation*

Description

Estimates the posterior mode for a Bayesian Structural Vector Autoregression (B-SVAR) model using the prior specified by Sims and Zha (1998)

Usage

```
szbsvar(Y, p, z = NULL,
        lambda0, lambda1, lambda3, lambda4, lambda5,
        mu5, mu6, ident, qm = 4)
```

Arguments

Y	$T \times m$ multiple time series object created with `ts()` with no NAs.
p	integer lag length for the model
z	$T \times k$ matrix of exogenous variables (not including an intercept)
lambda0	$[0, 1]$, Overall tightness of the prior (discounting of prior scale).
lambda1	$[0, 1]$, Standard deviation or tightness of the prior around the AR(1) parameters.
lambda3	Lag decay (> 0, with 1=harmonic)
lambda4	Standard deviation or tightness around the intercept > 0
lambda5	Standard deviation or tightness around the exogneous variable coefficients > 0
mu5	Sum of coefficients prior weight ≥ 0. Larger values imply difference stationarity.
mu6	Dummy Initial observations or drift prior ≥ 0. Larger values allow for common trends.
ident	$m \times m$ matrix of binary indicators for the identification of the free and restricted contemporaneous parameters in A_0.
qm	Frequency of the data for lag decay equivalence. Default is 4, and a value of 12 will match the lag decay of monthly to quarterly data. Other values have the same effect as "4"

Details

This function estimates the posterior mode for the Bayesian structural VAR (B-SVAR) model described by Sims and Zha (1998) and Waggoner and Zha (2003). This B-SVAR model is based a specification of the dynamic simultaneous equation representation of the model. The prior is constructed for the structural parameters.

The basic SVAR model has the form of Waggoner and Zha (2003):

$$y_t' A_0 = \sum_{\ell=1}^{p} Y_{t-\ell}' A_\ell + z_t' D + \epsilon_t', t = 1, \ldots, T,$$

where A_i are $m \times m$ parameter matrices for the contemporaneous and lagged effects of the endogenous variables, D is an $h \times m$ parameter matrix for the exogenous variables (including an intercept), y_t is the $m \times 1$ matrix of the endogenous variables, z_t is a $h \times 1$ vector of exogenous variables (including an intercept) and ϵ_t is the $m \times 1$ matrix of structural shocks. NOTE that in this representation of the model, the columns of the A_ℓ matrices refer to the equations!

The structural shocks are normal with mean and variance equal to the following:

$$E[\epsilon_t | y_1, \ldots, y_{t-1}, z_1, \ldots z_{t-1}] = 0$$

$$E[\epsilon_t \epsilon_t' | y_1, \ldots, y_{t-1}, z_1, \ldots z_{t-1}] = I$$

The *reduced form representation* of the SVAR model can be found by post-multiplying through by A_0^{-1}:

$$y_t' A_0 A_0^{-1} = \sum_{\ell=1}^{p} Y_{t-\ell}' A_\ell A_0^{-1} + z_t' D A_0^{-1} + \epsilon_t' A_0^{-1}$$

$$y_t' = \sum_{\ell=1}^{p} Y_{t-\ell}' B_\ell + z_t' \Gamma + \epsilon_t' A_0^{-1}.$$

The reduced form error covariance matrix is found from the crossproduct of the reduced form innovations:

$$\Sigma = E[(\epsilon_t' A_0^{-1})(\epsilon_t' A_0^{-1})'] = [A_0 A_0']^{-1}.$$

.

Restrictions on the contemporaneous parameters in A_0 are expressed by the specification of the `ident` matrix that defines the shocks that "hit" each equation in the contemporaneous specification. If `ident` is defined as in the following table,

		Equations	
Variables	Eqn 1	Eqn 2	Eqn 3
Var. 1	1	0	0
Var. 2	1	1	0
Var. 3	0	1	1

then the corresponding A_0 is restricted to

	Equations		
Variables	Eqn 1	Eqn 2	Eqn 3
Var. 1	a_{11}	0	0
Var. 2	a_{12}	a_{22}	0
Var. 3	0	a_{23}	a_{33}

which is interpreted as shocks in variables 1 and 2 hit equation 1 (the first column); shocks in variables 2 and 3 hit the second equation (column 2); and, shocks in variable 3 hit the third equation (column 3).

As in Sims and Zha (1998) and Waggoner and Zha (2003), the prior for the model is formed for each of the equations. To illustrate the prior, the model is written in the more compact notation

$$y'_t A_0 = x'_t F + \epsilon'_t$$

where

$$x'_t = [y'_{t-1} \cdots y'_{t-p}, z'_t], F' = [A'_1 \cdots A'_p \, D']$$

are the matrices of the right hand side variables and the right hand side coefficients for the SVAR model.

The general form of this prior is then

$$a_i \sim N(0, \bar{S}_i) \quad \text{and} \quad f_i | a_i \sim N(\bar{P}_i a_i, \bar{H}_i)$$

where $\bar{S}_i$ is an $m \times m$ prior covariance of the contemporaneous parameters, and $\bar{H}_i$ is the $k \times k$ prior covariance of the parameters in $f_i|a_i$. The prior means of a_i are zero in the structural model, while the "random walk" component is in $\bar{P}_i a_i$.

The prior covariance matrix of the errors, $\bar{S}_i$, is initially estimated using a VAR(p) model via OLS, with an intercept and no demeaning of the data.

The Bayesian prior is constructed for the unrestricted VAR model and then mapped into the restricted prior parameter space, as discussed in Waggoner and Zha (2003a).

Value

A list of the class "BSVAR" that summarizes the posterior mode of the B-SVAR model

`XX` $X'X + H_0$ crossproduct moment matrix for the predetermined variables in the model plus the prior

`XY` $X'Y$ for the model, including the dummy observations for mu5 and mu6

`YY` $m \times m$ Crossproduct for the Y's in the model

`y` $T \times m$ input data in `dat` plus the m dummy observations for `dat`

`structural.innovations`
$T \times m$ structural innovations for the SVAR model

`Ui` $m \times q_i$ Null space matrices that map the columns of A_0 to the free parameters of the columns

`Hpinv.tilde` Prior covariance for the predetermined and exogenous regression in the B-SVAR

`H0inv.tilde` m dimensional list of the prior covariances for the free parameters of the i'th equation in the model's A_0 matrix

`Pi.tilde` list of $(m^2p + 1 + h) \times q_i$ matrices of the prior for the parameters for the predetermined variables in the model

`Hpinv.posterior` $(m^2p + 1 + h) \times m$ matrix of the posterior of the structural parameters for the predetermined variables

`P.posterior` list of $(m^2p + 1 + h) \times m$ matrices of the posterior of the paramters for the predetermined variables in the model

`H0inv.posterior` m dimensional list of the posterior covariances for the free parameters of the i'th equation in the model's A_0 matrix

`A0.mode` posterior mode of the A_0 matrix

`F.posterior` $(m^2p + 1 + h) \times m$ matrix of the posterior of the structural parameters for the predetermined variables

`B.posterior` $(m^2p + 1 + h) \times m$ matrix of the posterior of the reduced form parameters for the predetermined variables

`ar.coefs` $(m^2p) \times m$ matrix of the posterior of the reduced form autoregressive parameters

`intercept` m dimensional vector of the reduced form intercepts

`exog.coefs` $h \times m$ matrix of the reduced form exogenous variable coefficients

`prior` List of the prior parameter: `c(lambda0,lambda1,lambda3,lambda4,lambda5, mu5, mu6)`.

`df` Degrees of freedom for the model: T + number of dummy observations - lag length.

`n0` m dimensional list of the number of free parameters for the A_0 matrix for equation i.

`ident` $m \times m$ identification matrix `ident`.

Warning

If you do not understand the model described here, you probably want the models described in `szbvar` or `reduced.form.var`

Author(s)

Patrick T. Brandt

References

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

Waggoner, Daniel F. and Tao A. Zha. 2003a. "A Gibbs sampler for structural vector autoregressions" *Journal of Economic Dynamics \& Control*. 28:349–366.

Waggoner, Daniel F. and Tao A. Zha. 2003b. "Likelihood preserving normalization in multiple equation models". *Journal of Econometrics*. 114: 329–347.

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis". *Political Analysis* 14(1):1-36.

See Also

`szbvar` for reduced form Bayesian VAR models, `reduced.form.var` for non-Bayesian reduced form VAR models, `gibbs.A0` for drawing from the posterior of this model using a Gibbs sampler, `posterior.fit` for assessing the posterior fit of the model, and `mc.irf` for computing impulse responses for this model.

Examples

```
# SZ, B-SVAR model for the Levant data
data(BCFdata)
m <- ncol(Y)
ident <- diag(m)
ident[1,] <- 1
ident[2,1] <- 1

# estimate the model's posterior moments
model <- szbsvar(Y, p=2, z=z2, lambda0=0.8, lambda1=0.1,
                 lambda3=1, lambda4=0.1, lambda5=0.05,
                 mu5=0, mu6=5, ident, qm=12)
```

---

szbvar — *Reduced form Sims-Zha Bayesian VAR model estimation*

Description

Estimation of the Bayesian VAR model for just identified VARs described in Sims and Zha (1998)

Usage

```
szbvar(Y, p, z = NULL, lambda0, lambda1, lambda3, lambda4, lambda5,
       mu5, mu6, nu = ncol(Y)+1, qm = 4, prior = 0,
       posterior.fit = FALSE)
```

Arguments

`Y`	$T \times m$ multiple time series object created with `ts()`.
`p`	Lag length
`z`	$T \times k$ matrix of exogenous variables. Can be `z = NULL` if there are none.
`lambda0`	$[0, 1]$, Overall tightness of the prior (discounting of prior scale).

`lambda1`	$[0, 1]$, Standard deviation or tightness of the prior around the AR(1) parameters.
`lambda3`	Lag decay (> 0, with 1=harmonic)
`lambda4`	Standard deviation or tightness around the intercept > 0
`lambda5`	Standard deviation or tightness around the exogneous variable coefficients > 0
`mu5`	Sum of coefficients prior weight ≥ 0. Larger values imply difference stationarity.
`mu6`	Dummy initial observations or drift prior ≥ 0. Larger values allow for common trends.
`nu`	Prior degrees of freedom, $m + 1$
`qm`	Frequency of the data for lag decay equivalence. Default is 4, and a value of 12 will match the lag decay of monthly to quarterly data. Other values have the same effect as "4"
`prior`	One of three values: 0 = Normal-Wishart prior, 1 = Normal-flat prior, 2 = flat-flat prior (i.e., akin to MLE)
`posterior.fit`	logical, F = do not estimate log-posterior fit measures, T = estimate log-posterior fit measures.

Details

This function estimates the Bayesian VAR (BVAR) model described by Sims and Zha (1998). This BVAR model is based a specification of the dynamic simultaneous equation representation of the model. The prior is constructed for the structural parameters. The basic SVAR model used here is documented in `szbsvar`.

The prior covariance matrix of the errors, $\bar{S}_i$, is initially estimated using a VAR(p) model via OLS, with an intercept and no demeaning of the data.

Value

Returns a list of multiple elements. This is a workhorse function to get the estimates, so nothing is displayed to the screen. The elements of the list are intended as inputs for the various post-estimation functions (e.g., impulse response analyses, forecasting, decompositions of forecast error variance, etc.)

Returns a list of the class "BVAR" with the following elements:

`intercept`	$m \times 1$ row vector of the m intercepts
`ar.coefs`	$m \times m \times p$ array of the AR coefficients. The first $m \times m$ array is for lag 1, the p'th array for lag p.
`exog.coefs`	$k \times m$ matrix of the coefficients for any exogenous variables
`Bhat`	$(mp + k + 1) \times m$ matrix of the coefficients, where the columns correspond to the variables in the VAR
`vcv`	$m \times m$ matrix of the maximum likelihood estimate of the residual covariance
`vcv.Bh`	Posterior estimate of the parameter covariance that is conformable with Bhat.
`mean.S`	$m \times m$ matrix of the posterior residual covariance.
`St`	$m \times m$ matrix of the degrees of freedom times the posterior residual covariance.

hstar	$(mp + k + 1) \times (mp + k + 1)$ prior precision plus right hand side variables crossproduct.
hstarinv	$(mp + k + 1) \times (mp + k + 1)$ prior covariance crossproduct solve(hstar)
H0	$(mp + k + 1) \times (mp + k + 1)$ prior precision for the parameters
S0	$m \times m$ prior error covariance
residuals	$(T - p) \times m$ matrix of the residuals
X	$T \times (mp+1+k)$ matrix of right hand side variables for the estimation of BVAR
Y	$T \times m$ matrix of the left hand side variables for the estimation of BVAR
y	$T \times m$ input data in dat
z	$T \times k$ exogenous variables matrix
p	Lag length
num.exog	Number of exogenous variables
qm	Value of parameter to match quarterly to monthly lag decay (4 or 12)
prior.type	Numeric code for prior type: 0 = Normal-Wishart, 1 = Normal-Flat, 2 = Flat-Flat (approximate MLE)
prior	List of the prior parameter: c(lambda0,lambda1,lambda3,lambda4,lambda5, mu5, mu6, nu)
marg.llf	Value of the in-sample marginal log-likelihood for the data, if posterior.fit=T
marg.post	Value of the in-sample marginal log posterior of the data, if posterior.fit=T
coef.post	Value of the marginal log posterior estimate of the coefficients, if posterior.fit=T

Note

This is a work horse function. You will probably want to use other functions to summarize and report the BVAR results.

Author(s)

Patrick T. Brandt, based on code from Robertson and Tallman and Sims and Zha.

References

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

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis". *Political Analysis*.

See Also

reduced.form.var szbsvar

Examples

```
## Not run:
  data(IsraelPalestineConflict)
  varnames <- colnames(IsraelPalestineConflict)

  fit.BVAR <- szbvar(IsraelPalestineConflict, p=6, z=NULL,
                         lambda0=0.6, lambda1=0.1,
                          lambda3=2, lambda4=0.25, lambda5=0, mu5=0,
                          mu6=0, nu=3, qm=4,
                          prior=0, posterior.fit=FALSE)

  # Draw from the posterior pdf of the impulse responses.
  posterior.impulses <- mc.irf(fit.BVAR, nsteps=10, draws=5000)

  # Plot the responses
  plot(posterior.impulses, method=c("Sims-Zha2"), component=1,
       probs=c(0.16,0.84), varnames=varnames)

## End(Not run)
```

`uc.forecast` *Forecast density estimation unconditional forecasts for VAR/BVAR/BSVAR models via MCMC*

Description

Implements unconditional forecast density estimator for VAR/BVAR/BSVAR models described in Waggoner and Zha (1999). The unconditional forecasts place no restriction on the paths of the forecasts (cf. `hc.forecast`). The forecast densities are estimated as the posterior sample for the VAR/BVAR/BSVAR model using Markov Chain Monte Carlo with data augmentation to account for the uncertainty of the forecasts and the parameters. This function DOES account for parameter uncertainty in the MCMC algorithm.

Usage

```
uc.forecast(varobj, nsteps, burnin, gibbs, exog = NULL)
```

Arguments

`varobj`	VAR or BVAR object produced for a VAR or BVAR using `szbvar` or `reduced.form.var`
`nsteps`	Number of periods in the forecast horizon
`burnin`	Burnin cycles for the MCMC algorithm
`gibbs`	Number of cycles of the Gibbs sampler after the `burnin` that are returned in the output
`exog`	num.exog x nsteps matrix of the exogenous variable values for the forecast horizon. If left at the `NULL` default, they are set to zero.

Details

Produces a posterior sample of unconstrained VAR/BVAR/BSVAR forecasts via MCMC. This function accounts for the uncertainty of the VAR/BVAR/BSVAR parameters by sampling from them in the computation of the VAR/BVAR/BSVAR forecasts and then regenerating the forecasts. Data augmentation is used to account for the impact of the forecast uncertainty on the parameters.

Value

A list with three components:

`yforc`	Forecast sample
`orig.y`	Original endogenous variables time series
`hyperp`	values of the hyperparameters used in the BVAR estimation / MCMC

Author(s)

Patrick T. Brandt

References

Brandt, Patrick T. and John R. Freeman. 2006. "Advances in Bayesian Time Series Modeling and the Study of Politics: Theory Testing, Forecasting, and Policy Analysis" *Political Analysis* 14(1):1-36.

Waggoner, Daniel F. and Tao Zha. 1999. "Conditional Forecasts in Dynamic Multivariate Models" *Review of Economics and Statistics*, 81(4):639-651.

See Also

hc.forecast

Examples

```
## Not run:
## Uses the example from Brandt and Freeman 2006.  Will not run unless
## you have their data from the Politcal
## Analysis website!
library(MSBVAR)   # Brandt's package for Bayesian VAR models

# Read the data and set up as a time series
data <- read.dta("levant.weekly.79-03.dta")
attach(data)

# Set up KEDS data
KEDS.data <- ts(cbind(a2i,a2p,i2a,p2a,i2p,p2i),
                start=c(1979,15),
                freq=52,
                names=c("A2I","A2P","I2A","P2A","I2P","P2I"))

# Select the sample we want to use.
KEDS <- window(KEDS.data, end=c(1988,50))
```

```
###############################################################################
# Estimate the BVAR models
###############################################################################

# Fit a flat prior model
KEDS.BVAR.flat <- szbvar(KEDS, p=6, z=NULL, lambda0=1,
                         lambda1=1, lambda3=1, lambda4=1, lambda5=0,
                         mu5=0, mu6=0, nu=0, qm=4, prior=2,
                         posterior.fit=F)

# Reference prior model -- Normal-IW prior pdf
KEDS.BVAR.informed <- szbvar(KEDS, p=6, z=NULL, lambda0=0.6,
                             lambda1=0.1, lambda3=2, lambda4=0.5, lambda5=0,
                             mu5=0, mu6=0, nu=ncol(KEDS)+1, qm=4, prior=0,
                             posterior.fit=F)

# Set up conditional forecast matrix conditions
nsteps <- 12
a2i.condition <- rep(mean(KEDS[,1]) + sqrt(var(KEDS[,1])) , nsteps)

yhat<-matrix(c(a2i.condition,rep(0, nsteps*5)), ncol=6)

# Set the random number seed so we can replicate the results.
set.seed(11023)

# Conditional forecasts
conditional.forcs.ref <- hc.forecast(KEDS.BVAR.informed, yhat, nsteps,
                            burnin=3000, gibbs=5000, exog=NULL)

conditional.forcs.flat <- hc.forecast(KEDS.BVAR.flat, yhat, nsteps,
                             burnin=3000, gibbs=5000, exog=NULL)

# Unconditional forecasts
unconditional.forcs.ref <-uc.forecast(KEDS.BVAR.informed, nsteps,
                                          burnin=3000, gibbs=5000)

unconditional.forcs.flat <- uc.forecast(KEDS.BVAR.flat, nsteps,
                                            burnin=3000, gibbs=5000)

# Set-up and plot the unconditional and conditional forecasts.  This
# code pulls for the forecasts for I2P and P2I and puts them into the
# appropriate array for the figures we want to generate.
uc.flat <- NULL
hc.flat <- NULL
uc.ref <- NULL
hc.ref <- NULL

uc.flat$forecast <- unconditional.forcs.flat$forecast[,,5:6]
hc.flat$forecast <- conditional.forcs.flat$forecast[,,5:6]
uc.ref$forecast <- unconditional.forcs.ref$forecast[,,5:6]
hc.ref$forecast <- conditional.forcs.ref$forecast[,,5:6]
```

```
par(mfrow=c(2,2), omi=c(0.25,0.5,0.25,0.25))
plot(uc.flat,hc.flat, probs=c(0.16, 0.84), varnames=c("I2P", "P2I"),
     compare.level=KEDS[nrow(KEDS),5:6], lwd=2)
plot(hc.ref,hc.flat, probs=c(0.16, 0.84), varnames=c("I2P", "P2I"),
     compare.level=KEDS[nrow(KEDS),5:6], lwd=2)


## End(Not run)
```

`var.lag.specification` *Automated VAR lag specification testing*

Description

Estimates a series of test statistics and measures for VAR lag length selection.

Usage

```
var.lag.specification(y, lagmax = 20)
```

Arguments

`y`	T x m multiple time series
`lagmax`	Maximum lag order to be evaluated. Function will return lag length tests for all lag orders less than lagmax.

Details

Estimates a series of frequentist VAR models for 1 to lagmax and returns a sequence of χ^2 tests, AIC, BIC and Hannan-Quinn criterion values for each lag length.

Value

Results are printed to standard output (screen or file). In addition, a list of two matrices is returned:

`ldets`	Lag length, log-determinants, χ^2 tests, and p-values for each lag length, compared to the null of the next shorter lag length
`results`	Lag length, AIC, BIC, and HQ criteria for each lag length. Selection criteria should be minimized.

Note

Sizes of p-values are uncorrected for multiple testing. Use cautiously.

Author(s)

Patrick T. Brandt

References

Lutkepohl, Helmut 2004."Vector Autoregressive and Vector Error Correction Models", Chapter 3. In Applied Time Series Econometrics. Lutkepohl,, Helmut and Markus Kratzig eds. Cambridge: CUP.

See Also

See Also `reduced.form.var` for frequentist VAR estimation, `szbvar` for Bayesian VAR estimation, and `szbsvar` for Bayesian Structural VAR estimation.

Examples

```
data(IsraelPalestineConflict)
var.lag.specification(IsraelPalestineConflict, lagmax=12)
```

Index