Package ‘BMR’

July 23, 2015

Type Package

Title Bayesian Macroeconometrics in R

Version 0.5.1

Date 2015-07-22

Author Keith O'Hara <keith.ohara@nyu.edu>

Maintainer Keith O'Hara <keith.ohara@nyu.edu>

Description A package for estimating Bayesian macroeconometric models.

Depends Rcpp (>= 0.11.6), RcppArmadillo (>= 0.5.2.0.0), doParallel (>= 1.0.8), ggplot2 (>= 1.0.0)

LinkingTo Rcpp, RcppArmadillo

Imports foreach, parallel, grid

NeedsCompilation yes

Repository CRAN

R topics documented:

BMR *Bayesian Macroeconometrics in R*

Description

Bayesian Macroeconometrics in R ('BMR') is a collection of R and C++ routines for estimating Bayesian Vector Autoregressive (BVAR) and Dynamic Stochastic General Equilibrium (DSGE) models in the R statistical environment.

Details

Package:	BMR
Type:	Package
Version:	0.5.1
Date:	2015-07-22
License:	GPL (>= 2)
LazyLoad:	Yes
Depends:	Rcpp, RcppArmadillo, doParallel, ggplot2, grid

How to cite this package

```
@Manual{OHara2015,
 author       = {Keith OHara},
 title        = {{BMR}: Bayesian Macroeconometrics in R.},
 year         = {2015},
 note         = {R package version 0.5.1.}}
```

License

BMR is licensed under the GNU General Public License (GPL) version 2, or (at your option) any later version. The experimental nature of this software implies that it is intended for academic use only.

Author(s)

Keith O'Hara

BMRLSData *Data: Artificial DSGE Data*

Description

Artificial data generated from the log-linearised Lubik-Schorfheide (2007) model.

Usage

```
data(BMRLSData)
```

Format

A matrix called 'LSData' of size 200 x 5.

BMRMCData *Data: Artificial VAR Data*

Description

Artificial data generated from a bi-variate VAR(2) model. See section 6.1 of the vignette for details relating to model specification.

Usage

```
data(BMRMCData)
```

Format

A matrix called 'bvarMCdata' of size 100 x 2.

BMRVARData *Data: Monetary Policy VAR Data*

Description

This is an updated version of the Stock and Watson (2001) dataset, with inflation, unemployment, and the Federal Funds rate, from Q2 1954 to Q4 2011.

Usage

```
data(BMRVARData)
```

Format

A dataframe of size 231 x 4.

Source

Federal Reserve Economic Data

BVARM *BVAR with Minnesota Prior.*

Description

Estimate a Bayesian VAR with Minnesota prior.

Usage

```
BVARM(mydata,coefprior=NULL,p=4,constant=TRUE,
      irf.periods=20,keep=10000,burnin=1000,
      VType=1,decay="H",HP1=0.5,HP2=0.5,HP3=1,HP4=2)
```

Arguments

`mydata`	A matrix or data frame containing the data series used for estimation; this should be of size T x m.
`coefprior`	A numeric vector of length m, matrix of size (m p + 1_c) x m, or a value of 'NULL', that contains the prior mean-value of each coefficient. Providing a numeric vector of length m will set a zero prior on all coefficients except the own first-lags, which are set according to the elements in 'coefprior'. Setting this input to 'NULL' will give a random-walk-in-levels prior.
`p`	The number of lags to include of each variable. The default value is 4.
`constant`	A logical statement on whether to include a constant vector (intercept) in the model. The default is 'TRUE', and the alternative is 'FALSE'.
`irf.periods`	An integer value for the horizon of the impulse response calculations, which must be greater than zero. The default value is 20.
`keep`	The number of Gibbs sampling replications to keep from the sampling run.
`burnin`	The sample burn-in length for the Gibbs sampler.
`VType`	Whether to use a 'VType=1' or 'VType=2'. The default is 1.
`decay`	Whether to use harmonic or geometric decay for the VType=2 case.
`HP1, HP2, HP3, HP4`	These correspond to H_1, H_2, H_3, and H_4, respectively, from section 2.3 of the vignette.

Details

For technical details of the model, see the accompanying vignette.

Value

The function returns an object of class BVARM, which contains:

`Beta`	A matrix of size (m p + 1_c) x m containing the posterior mean of the coefficient matrix (beta).
`BDraws`	An array of size (m p + 1_c) x m x keep which contains the post burn-in draws of beta.
`BetaVPr`	An matrix of size (m p + 1_c) m x (m p + 1_c) m containing the prior covariance matrix of vec(beta).

`Sigma`	A matrix of size m x m containing the fixed residual covariance matrix (Sigma).
`IRFs`	A four-dimensional object of size irf.periods x m x keep x m containing the impulse response function calculations; the first m refers to responses to the last m shock. Note: IRF calculations are based on a one standard deviation shock to each variable.
`data`	The data used for estimation.
`constant`	A logical value, TRUE or FALSE, indicating whether the user chose to include a vector of constants in the model.

Author(s)

Keith O'Hara

References

Canova, Fabio, *Methods for Applied Macroeconomic Research,* Princeton, New Jersey: Princeton University Press, 2007.

Koop, Gary and Dimitris Korobilis, "Bayesian Multivariate Time Series Methods for Empirical Macroeconomics," *Mimeo*, 2010.

See Also

`forecast.BVARM`, `IRF.BVARM`, `plot.BVARM`.

Examples

```
## Not run:
data(BMRVARData)
prior<-c(0.9,0.95,0.95)
testbvarm <- BVARM(USMacroData[,2:4],prior,p=4,constant=T,irf.periods=20,
                   keep=10000,burnin=5000,VType=1,
                   HP1=0.5,HP2=0.5,HP3=10)
plot(testbvarm,save=F)
IRF(testbvarm,save=F)
forecast(testbvarm,shocks=TRUE,backdata=10,save=FALSE)

## End(Not run)
```

`BVARS` *BVAR with Steady-State Prior.*

Description

Estimate a Bayesian VAR with steady-state prior.

Usage

```
BVARS(mydata,psiprior=NULL,coefprior=NULL,p=4,
      irf.periods=20,keep=10000,burnin=1000,
      XiPsi=1,HP1=0.5,HP4=2,gamma=NULL)
```

Arguments

`mydata`	A matrix or data frame containing the data series used for estimation; this should be of size T x m.
`psiprior`	A numeric vector of length m that contains the prior mean of each series found in 'mydata'. The user MUST specify this prior, or the function will return an error.
`coefprior`	A numeric vector of length m, matrix of size (m p) x m, or a value of 'NULL', that contains the prior mean-value of each coefficient. Providing a numeric vector of length m will set a zero prior on all coefficients except the own first-lags, which are set according to the elements in 'coefprior'. Setting this input to 'NULL' will give a random-walk-in-levels prior.
`p`	The number of lags to include of each variable. The default value is 4.
`irf.periods`	An integer value for the horizon of the impulse response calculations, which must be greater than zero. The default value is 20.
`keep`	The number of Gibbs sampling replications to keep from the sampling run.
`burnin`	The sample burn-in length for the Gibbs sampler.
`XiPsi`	A numeric vector of length 1 or matrix of size m x m that defines the prior location matrix of Psi.
`HP1`	H_1 from section 3 of the vignette.
`HP4`	H_4 from section 3 of the vignette.
`gamma`	A numeric vector of length 1 corresponding to the prior degrees of freedom of the error covariance matrix. The minimum value is m+1, and this is the default value.

Details

For technical details of the model, see the accompanying vignette.

Value

The function returns an object of class BVARS, which contains:

`Beta`	A matrix of size (m p) x m containing the posterior mean of the coefficient matrix (beta).
`BDraws`	An array of size (m p) x m x keep which contains the post burn-in draws of beta.
`Psi`	A matrix of size 1 x m containing the posterior mean estimate of the unconditional mean matrix (Psi).
`PDraws`	An array of size 1 x m x keep which contains the post burn-in draws of Psi.
`Sigma`	A matrix of size m x m containing the posterior mean estimate of the residual covariance matrix (Sigma).
`SDraws`	An array of size m x m x keep which contains the post burn-in draws of Sigma.
`IRFs`	A four-dimensional object of size irf.periods x m x keep x m containing the impulse response function calculations; the first m refers to responses to the last m shock. Note: IRF calculations are based on a one standard deviation shock to each variable.
`data`	The data used for estimation.

Author(s)

Keith O'Hara

References

Villani, Mattias, "Steady-State Priors for Vector Autoregressions," *Journal of Applied Econometrics, 2009, 24* (4), 630–650.

See Also

forecast.BVARS, IRF.BVARS, plot.BVARS.

Examples

```
## Not run:
data(BMRVARData)
mycfp = c(0.9,0.95,0.95)
mypsi = c(3,6,5)
testbvars <- BVARS(USMacroData[,2:4],mypsi,mycfp,p=4,irf.periods=20,
                   keep=20000,burnin=5000,XiPsi=1,HP1=0.5,HP4=2,gamma=NULL)
plot(testbvars,save=F)
IRF(testbvars,save=F)
forecast(testbvars,shocks=T,backdata=10,save=F)

## End(Not run)
```

BVARTVP *BVAR with Time-Varying Coefficients.*

Description

Estimate a Bayesian VAR with time-varying coefficients.

Usage

```
BVARTVP(mydata,timelab=NULL,coefprior=NULL,tau=NULL,p=4,
        irf.periods=20,irf.points=NULL,
        keep=10000,burnin=5000,
        XiBeta=1,XiQ=0.01,gammaQ=NULL,
        XiSigma=1,gammaS=NULL)
```

Arguments

mydata	A matrix or data frame containing the data series used for estimation; this should be of size T x m.
timelab	This is a numeric vector of length T that provides labels for the observations.
coefprior	A numeric vector of length m, matrix of size (m p + 1) x m, or a value 'NULL', that contains the prior mean-value of each coefficient. Providing a numeric vector of length m will set a zero prior on all coefficients except the own first-lags, which are set according to the elements in 'coefprior'. Setting this input to 'NULL' will give a random-walk-in-levels prior. Note that, when tau is set to 'NULL', this input becomes the initial draw for the sampling algorithm, and starting with an explosive draw might be a bad idea.

`tau` ‘tau’ is the length of the training-sample prior. If this is set a value other than ‘NULL’, it will replace ‘coefprior’ above with the coefficients from a pre-sampling estimation run. Selecting this option also affects the ‘XiBeta’ choice below.

`p` The number of lags to include of each variable. The default value is 4.

`irf.periods` An integer value for the horizon of the impulse response calculations, which must be great than zero. The default value is 20.

`irf.points` A numeric vector of length (0,T]. If the user supplied a ‘timelab’ list above, then this vector should contain points corresponding to that list. The default of ‘NULL’ will mean that all IRFs, for T - tau, will be computed. The IRFs are stored in a 5 dimensional array of size irf.periods x m x m x length(irf.points) x keep.

If the number of variables, replications, and/or observations is quite large, then calculating all IRFs will take up a lot of memory. For example, with an IRF horizon of 20, 3 variables, 200 observations, training sample size of 50, and 50000 post-burn-in replications, we have 1,350,000,000 elements to store.

`keep` The number of Gibbs sampling replications to keep from the sampling run.

`burnin` The sample burn-in length for the Gibbs sampler.

`XiBeta` A numeric vector of length 1 or matrix of size (m p + 1) m x (m p + 1) m that contains the prior covariance of each coefficient for beta_0. The structure of Xi_beta corresponds to vec(beta).

Note that if tau != NULL, ‘XiBeta’ should be a numeric vector of length 1 (scalar) that scales the OLS estimate of the covariance matrix of beta.

`XiQ` A numeric vector of length 1 or matrix of size (m p + 1) m x (m p + 1) m that contains the location matrix of the inverse-Wishart prior on Q.

`gammaQ` A numeric vector of length 1 corresponding to the prior degrees of freedom of the Q matrix. The minimum value is (m p + 1) m + 1, and this is the default value, unless tau != NULL, in which case gamma_S = tau.

`XiSigma` A numeric vector of length 1 or matrix of size m x m that contains the location matrix of the inverse-Wishart prior on Sigma.

`gammaS` A numeric vector of length 1 corresponding to the prior degrees of freedom of the error covariance matrix. The minimum value is m+1, and this is the default value.

Details

For technical details of the model, see the accompanying vignette.

Value

The function returns an object of class BVARTVP, which contains:

`Beta` A matrix of size (m p + 1) m x (T - tau) containing the posterior mean of the coefficient matrix, beta, in vectorised form, for (tau + 1) : T.

`BDraws` An array of size (m p + 1) x m x keep x (T - tau) which contains the post burn-in draws of beta.

`Q` A matrix of size (m p + 1) m x (m p + 1) m containing the posterior mean estimates of the covariance matrix Q.

`QDraws`	An array of size (m p + 1) m x (m p + 1) m x keep which contains the post burn-in draws of Q.
`Sigma`	A matrix of size m x m containing the posterior mean estimates of the residual covariance matrix, Sigma.
`SDraws`	An array of size m x m x keep which contains the post burn-in draws of Sigma.
`IRFs`	Let ell = number of 'irf.points' the user selected. 'IRFs' is then a five-dimensional object of size irf.periods x m x m x ell x keep containing the impulse response function calculations; the first m refers to responses to the last m shock. Note: IRF calculations are based on a one standard deviation shock to each variable.
`data`	The data used for estimation.
`irf.points`	The points in the sample where the user elected to produce IRFs.
`tau`	The length of the training sample.

Author(s)

Keith O'Hara

References

Koop, Gary and Dimitris Korobilis, "Bayesian Multivariate Time Series Methods for Empirical Macroeconomics," *Mimeo*, 2010.

See Also

`IRF.BVARTVP`, `plot.BVARTVP`.

Examples

```
## Not run:
irf.points<-c(1979,1988,1997,2006)
yearlab<-seq(1955.00,2010.75,0.25)
USMacroData<-USMacroData[3:226,2:4]
bvartvptest <- BVARTVP(USMacroData,timelab=yearlab,
                coefprior=NULL,tau=80,p=4,
                irf.periods=20,irf.points=irf.points,
                keep=30000,burnin=40000,
                XiBeta=4,XiQ=0.005,gammaQ=NULL,
                XiSigma=1,gammaS=NULL)

## End(Not run)
```

`BVARW` *BVAR with normal-inverse-Wishart Prior.*

Description

Estimate a Bayesian VAR with normal-inverse-Wishart prior.

Usage

```
BVARW(mydata,cores=1,coefprior=NULL,p=4,constant=TRUE,
      irf.periods=20,keep=10000,burnin=1000,
      XiBeta=1,XiSigma=1,gamma=NULL)
```

Arguments

`mydata`	A matrix or data frame containing the data series used for estimation; this should be of size T x m.
`cores`	A positive integer value indicating the number of CPU cores that should be used for the sampling run. DO NOT ENTER MORE CORES THAN YOUR COMPUTER CAN SAFELY HANDLE! If in doubt, set cores = 1, which is the default.
`coefprior`	A numeric vector of length m, matrix of size (m p + 1_c) x m, or a value of 'NULL', that contains the prior mean-value of each coefficient. Providing a numeric vector of length m will set a zero prior on all coefficients except the own first-lags, which are set according to the elements in 'coefprior'. Setting this input to 'NULL' will give a random-walk-in-levels prior.
`p`	The number of lags to include of each variable. The default value is 4.
`constant`	A logical statement on whether to include a constant vector (intercept) in the model. The default is 'TRUE', and the alternative is 'FALSE'.
`irf.periods`	An integer value for the horizon of the impulse response calculations, which must be greater than zero. The default value is 20.
`keep`	The number of Gibbs sampling replications to keep from the sampling run.
`burnin`	The sample burn-in length for the Gibbs sampler.
`XiBeta`	A numeric vector of length 1 or matrix of size (m p + 1_c) m x (m p + 1_c) m comprising the prior covariance of each coefficient. The structure of Xi_beta corresponds to vec(beta).
`XiSigma`	A numeric vector of length 1 or matrix of size m x m that contains the location matrix of the inverse-Wishart prior.
`gamma`	A numeric vector of length 1 corresponding to the prior degrees of freedom of the error covariance matrix. The minimum value is m+1, and this is the default value.

Details

For technical details of the model, see the accompanying vignette.

Value

The function returns an object of class BVARW, which contains:

`Beta`	A matrix of size (m p + 1_c) x m containing the posterior mean of the coefficient matrix (beta).
`BDraws`	An array of size (m p + 1_c) x m x keep which contains the post burn-in draws of beta.
`Sigma`	A matrix of size m x m containing the posterior mean estimate of the residual covariance matrix (Sigma).
`SDraws`	An array of size m x m x keep which contains the post burn-in draws of Sigma.

`IRFs` A four-dimensional object of size irf.periods x m x keep x m containing the impulse response function calculations; the first m refers to responses to the last m shock.

Note: IRF calculations are based on a one standard deviation shock to each variable.

`data` The data used for estimation.

`constant` A logical value, TRUE or FALSE, indicating whether the user chose to include a vector of constants in the model.

Author(s)

Keith O'Hara

References

Koop, Gary and Dimitris Korobilis, "Bayesian Multivariate Time Series Methods for Empirical Macroeconomics," *Mimeo*, 2010.

See Also

`forecast.BVARW`, `IRF.BVARW`, `plot.BVARW`.

Examples

```
## Not run:
data(BMRVARData)
testbvarw <- BVARW(USMacroData[,2:4],cores=1,c(0.9,0.95,0.95),p=4,constant=T,
                    irf.periods=20,keep=10000,burnin=5000,
                    XiBeta=4,XiSigma=1,gamma=4)
plot(testbvarw,save=F)
IRF(testbvarw,save=F)
forecast(testbvarw,shocks=TRUE,backdata=10,save=FALSE)

## End(Not run)
```

`CVAR` *Classical VAR.*

Description

OLS estimation of a VAR model with bootstrapped IRFs.

Usage

```
CVAR(mydata,p=4,constant=TRUE,irf.periods=20,boot=10000)
```

Arguments

`mydata`	A matrix or data frame containing the data series used for estimation; this should be of size T x m.
`p`	The number of lags to include of each variable. The default value is 4.
`constant`	A logical statement on whether to include a constant vector (intercept) in the model. The default is 'TRUE', and the alternative is 'FALSE'.
`irf.periods`	An integer value for the horizon of the impulse response calculations, which must be greater than zero. The default value is 20.
`boot`	The number of replications to run for the bootstrapped IRFs. The default is 10,000.

Details

For technical details of the model, see the accompanying vignette.

Value

The function returns an object of class CVAR, which contains:

`Beta`	A matrix of size (m p + 1_c) x m containing the OLS estimate of the coefficient matrix (beta).
`BDraws`	An array of size (m p + 1_c) x m x keep which contains the bootstrapped beta draws.
`Sigma`	A matrix of size m x m containing the OLS estimate of the residual covariance matrix (Sigma).
`SDraws`	An array of size m x m x keep which contains bootstrapped Sigma draws.
`IRFs`	A four-dimensional object of size irf.periods x m x boot x m containing the impulse response function calculations; the first m refers to responses to the last m shock. Note: IRF calculations are based on a one standard deviation shock to each variable.
`data`	The data used for estimation.
`constant`	A logical value, TRUE or FALSE, indicating whether the user chose to include a vector of constants in the model.

Author(s)

Keith O'Hara

See Also

`forecast.CVAR`, `IRF.CVAR`.

Examples

```
## Not run:
data(BMRVARData)
testcvar <- CVAR(USMacroData[,2:4],p=4,constant=TRUE,irf.periods=20,boot=10000)
IRF(testcvar,save=F)
forecast(testcvar,backdata=10,save=FALSE)

## End(Not run)
```

DSGESim *Simulate from a DSGE Model.*

Description

Simulate a data series using a solved DSGE model.

Usage

```
DSGESim(obj,...)

## S3 method for class gensys
DSGESim(obj,shocks.cov,sim.periods,burnin=NULL,
        seedval=1122,hpfiltered=FALSE,lambda=1600,...)
## S3 method for class uhlig
DSGESim(obj,shocks.cov,sim.periods,burnin=NULL,
        seedval=1122,hpfiltered=FALSE,lambda=1600,...)
## S3 method for class SDSGE
DSGESim(obj,shocks.cov,sim.periods,burnin=NULL,
        seedval=1122,hpfiltered=FALSE,lambda=1600,...)
```

Arguments

`obj`	An object of class 'SDSGE', 'gensys', or 'uhlig'. The user should first solve a model using one of the solver functions ('SDSGE', 'gensys', or 'uhlig'), then pass the solution to 'DSGESim'.
`shocks.cov`	A matrix of size k x k that describes the covariance structure of the model shocks.
`sim.periods`	The number of simulation periods the function should return.
`burnin`	The length of sample burn-in. The default, 'burnin = NULL', will set burn-in to one-half of the number given in 'sim.periods'.
`seedval`	Seed the random number generator.
`hpfiltered`	Whether to pass the simulated series through a Hodrick-Prescott filter before retuning.
`lambda`	If 'hpfiltered = TRUE', this is the value of the smoothing parameter in the H-P filter.
`...`	Additional arguments (not used).

Details

For an example, see the accompanying vignette.

Value

The function will return a matrix of simulated observations from a solved DSGE model.

Author(s)

Keith O'Hara

See Also

gensys, uhlig, SDSGE

Examples

```
## Not run:
dsgetest <- gensys(Gamma0,Gamma1,C,Psi,Pi)
dsgetestsim <- DSGESim(dsgetest,Sigma,200,200)

## End(Not run)
```

DSGEVAR *DSGE-VAR Estimation.*

Description

Estimate a DSGE-VAR model.

Usage

```
DSGEVAR(dsgedata,chains=1,cores=1,lambda=Inf,p=2,
        constant=FALSE,ObserveMat,initialvals,partomats,
        priorform,priorpars,parbounds,parnames=NULL,
        optimMethod="Nelder-Mead",
        optimLower=NULL,optimUpper=NULL,
        optimControl=list(),
        IRFs=TRUE,irf.periods=20,scalepar=1,
        keep=50000,burnin=10000,
        tables=TRUE)
```

Arguments

dsgedata	A matrix or data frame of size T x j containing the data series used for estimation. Note: in order to identify the structural shocks, there must be the same number of observable series as there are shocks in the DSGE model.
chains	A positive integer value indicating the number of MCMC chains to run.
cores	A positive integer value indicating the number of CPU cores that should be used for estimation. This number should be less than or equal to the number of chains. DO NOT ENTER MORE CORES THAN YOUR COMPUTER CAN SAFELY HANDLE! If in doubt, set cores = 1, which is the default.
lambda	The proportion of DSGE dummy data to actual data. Acceptable values lie in the interval j x (p+1)/T to +infinity. (See the vignette for details.)
p	The number of lags to include of each variable. The default value is 2.
constant	A logical statement on whether to include a constant vector (intercept) in the model. The default is 'FALSE', and the alternative is 'TRUE'.
ObserveMat	The (m+n+k) x j observable matrix H that maps the state variables to the observable series in the measurement equation.
initialvals	Initial values to begin the optimization routine.

`partomats`	This is perhaps the most important function input. 'partomats' should be a function that maps the deep parameters of the DSGE model to the matrices of a solution method, and contain: a k x k matrix labelled 'shocks' containing the variances of the structural shocks; a j x 1 matrix labelled 'MeasCons' containing any constant terms in the measurement equation; and a j x j matrix labelled 'MeasErrs' containing the variances of the measurement errors.
`priorform`	The prior distribution of each parameter.
`priorpars`	The parameters of the prior densities. For example, if the user selects a Gaussian prior for a parameter, then the first entry will be the mean and the second its variance.
`parbounds`	The lower- and (where relevant) upper-bounds on the parameter values. 'NA' values are permitted.
`parnames`	A character vector containing labels for the parameters.
`optimMethod`	The optimization algorithm used to find the posterior mode. The user may select: the "Nelder-Mead" simplex method, which is the default; "BFGS", a quasi-Newton method; "CG" for a conjugate gradient method; "L-BFGS-B", a limited-memory BFGS algorithm with box constraints; or "SANN", a simulated-annealing algorithm. See `optim` for more details. If more than one method is entered, e.g., c(Nelder-Mead, CG), optimization will proceed in a sequential manner, updating the initial values with the result of the previous optimization routine.
`optimLower`	If optimMethod="L-BFGS-B", this is the lower bound for optimization.
`optimUpper`	If optimMethod="L-BFGS-B", this is the upper bound for optimization.
`optimControl`	A control list for optimization. See `optim` for more details.
`IRFs`	Whether to calculate impulse response functions.
`irf.periods`	If IRFs=TRUE, then use this option to set the IRF horizon.
`scalepar`	The scaling parameter, c, for the MCMC run.
`keep`	The number of replications to keep. If keep is set to zero, the function will end with a normal approximation at the posterior mode.
`burnin`	The number of sample burn-in points.
`tables`	Whether to print results of the posterior mode estimation and summary statistics of the MCMC run.

Details

For technical details, see the accompanying vignette.

Value

The function returns an object of class DSGEVAR, which contains:

`Parameters`	A matrix with 'keep x chains' number of rows that contains the estimated, post sample burn-in parameter draws.
`Beta`	An array of size (j x p) x m x (keep x chains) which contains the post burn-in draws of beta.

Sigma
: An array of size j x j x (keep x chains) which contains the post burn-in draws of Sigma.

DSGEIRFs
: A four-dimensional object of size irf.periods x (m + n + k) x n x (keep x chains) containing the impulse response function calculations for the DSGE model. The first m refers to responses to the last m shock.

DSGEVARIRFs
: A four-dimensional object of size irf.periods x j x n x (keep x chains) containing the impulse response function calculations for the VAR. The last m refers to the structural shock.

parMode
: Estimated posterior mode parameter values.

ModeHessian
: The Hessian computed at the posterior mode for the transformed parameters.

logMargLikelihood
: The log marginal likelihood from a Laplacian approximation at the posterior mode.

AcceptanceRate
: The acceptance rate of the chain(s).

RootRConvStats
: Gelman's sqrtR-between-chain convergence statistics for each parameter. A value close 1 would signal convergence.

ObserveMat
: The user-supplied H matrix from the Kalman filter recursion.

data
: The data used for estimation.

Author(s)

Keith O'Hara

See Also

forecast.DSGEVAR, IRF.DSGEVAR, optim, plot.DSGEVAR, states.DSGEVAR.

Examples

```
## Not run:
NKMVAR <- DSGEVAR(dsgedata,chains=1,cores=1,lambda=1,p=4,
                  FALSE,ObserveMat,initialvals,partomats,
                  priorform,priorpars,parbounds,parnames,
                  optimMethod=c("Nelder-Mead","CG"),
                  optimLower=NULL,optimUpper=NULL,
                  optimControl=list(maxit=20000,reltol=(10^(-12))),
                  IRFs=TRUE,irf.periods=5,
                  scalepar=0.28,keep=25000,burnin=25000)

## End(Not run)
```

EDSGE *DSGE Estimation.*

Description

Estimate a DSGE model.

Usage

```
EDSGE(dsgedata,chains=1,cores=1,
      ObserveMat,initialvals,partomats,
      priorform,priorpars,parbounds,parnames=NULL,
      optimMethod="Nelder-Mead",
      optimLower=NULL,optimUpper=NULL,
      optimControl=list(),
      DSGEIRFs=TRUE,irf.periods=20,
      scalepar=1,keep=50000,burnin=10000,
      tables=TRUE)
```

Arguments

`dsgedata` A matrix or data frame of size T x j containing the data series used for estimation.

`chains` A positive integer value indicating the number of MCMC chains to run.

`cores` A positive integer value indicating the number of CPU cores that should be used for estimation. This number should be less than or equal to the number of chains.

DO NOT ENTER MORE CORES THAN YOUR COMPUTER CAN SAFELY HANDLE! If in doubt, set cores = 1, which is the default.

`ObserveMat` The (m+n+k) x j observable matrix H that maps the state variables to the observable series in the measurement equation.

`initialvals` Initial values to begin the optimization routine.

`partomats` This is perhaps the most important function input.

'partomats' should be a function that maps the deep parameters of the DSGE model to the matrices of a solution method, and contain: a k x k matrix labelled 'shocks' containing the variances of the structural shocks; a j x 1 matrix labelled 'MeasCons' containing any constant terms in the measurement equation; and a j x j matrix labelled 'MeasErrs' containing the variances of the measurement errors.

`priorform` The prior distribution of each parameter.

`priorpars` The parameters of the relevant prior densities.

For example, if the user selects a Gaussian prior for a parameter, then the first entry will be the mean and the second its variance.

`parbounds` The lower- and (where relevant) upper-bounds on the parameter values.

`parnames` A character vector containing labels for the parameters.

`optimMethod` The optimization algorithm used to find the posterior mode. The user may select: the "Nelder-Mead" simplex method, which is the default; "BFGS", a quasi-Newton method; "CG" for a conjugate gradient method; "L-BFGS-B", a limited-memory BFGS algorithm with box constraints; or "SANN", a simulated-annealing algorithm.

See `optim` for more details.

If more than one method is entered, e.g., c(Nelder-Mead, CG), optimization will proceed in a sequential manner, updating the initial values with the result of the previous optimization routine.

`optimLower` If optimMethod="L-BFGS-B", this is the lower bound for optimization.

`optimUpper` If optimMethod="L-BFGS-B", this is the upper bound for optimization.

`optimControl` A control list for optimization. See `optim` for more details.

DSGEIRFs	Whether to calculate impulse response functions.
irf.periods	If DSGEIRFs=TRUE, then use this option to set the IRF horizon.
scalepar	The scaling parameter, c, for the MCMC run.
keep	The number of replications to keep. If keep is set to zero, the function will end with a normal approximation at the posterior mode.
burnin	The number of sample burn-in points.
tables	Whether to print results of the posterior mode estimation and summary statistics of the MCMC run.

Details

For technical details, see the accompanying vignette.

Value

The function returns an object of class EDSGE, which contains:

Parameters	A matrix with 'keep x chains' number of rows that contains the estimated, post sample burn-in parameter draws.
parMode	Estimated posterior mode parameter values.
ModeHessian	The Hessian computed at the posterior mode for the transformed parameters.
logMargLikelihood	The log marginal likelihood from a Laplacian approximation at the posterior mode.
IRFs	The IRFs (if any), based on the posterior parameter draws.
AcceptanceRate	The acceptance rate of the chain(s).
RootRConvStats	Gelman's sqrtR-between-chain convergence statistics for each parameter. A value close 1 would signal convergence.
ObserveMat	The user-supplied H matrix from the Kalman filter recursion.
data	The data used for estimation.

Author(s)

Keith O'Hara

See Also

forecast.EDSGE, IRF.EDSGE, optim, plot.EDSGE, states.EDSGE.

Examples

```
## Not run:
NKMest <- EDSGE(dsgedata,chains=1,cores=1,
                ObserveMat,initialvals,partomats,
                priorform,priorpars,parbounds,parnames,
                optimMethod=c("Nelder-Mead","CG"),
                optimLower=NULL,optimUpper=NULL,
                optimControl=list(maxit=10000),
                DSGEIRFs=TRUE,irf.periods=40,
                scalepar=0.27,keep=50000,burnin=75000)

## End(Not run)
```

`forecast` *Forecasting with VAR, DSGE, and DSGE-VAR Models.*

Description

Compute forecasts using VAR, DSGE, and DSGE-VAR models.

Usage

```
forecast(obj,...)

## S3 method for class BVARM
forecast(obj,periods=20,shocks=TRUE,plot=TRUE,
         percentiles=c(.05,.50,.95),useMean=FALSE,
         backdata=0,save=FALSE,height=13,width=11,...)
## S3 method for class BVARS
forecast(obj,periods=20,shocks=TRUE,plot=TRUE,
         percentiles=c(.05,.50,.95),useMean=FALSE,
         backdata=0,save=FALSE,height=13,width=11,...)
## S3 method for class BVARW
forecast(obj,periods=20,shocks=TRUE,plot=TRUE,
         percentiles=c(.05,.50,.95),useMean=FALSE,
         backdata=0,save=FALSE,height=13,width=11,...)
## S3 method for class DSGEVAR
forecast(obj,periods=20,shocks=TRUE,plot=TRUE,
         percentiles=c(.05,.50,.95),useMean=FALSE,
         backdata=0,save=FALSE,height=13,width=11,...)

## S3 method for class CVAR
forecast(obj,periods=20,plot=TRUE,confint=0.95,
         backdata=0,save=FALSE,height=13,width=11,...)

## S3 method for class EDSGE
forecast(obj,periods=20,plot=TRUE,
         percentiles=c(.05,.50,.95),useMean=FALSE,
         backdata=0,save=FALSE,height=13,width=11,...)
```

Arguments

For objects of class 'BVARM', 'BVARS', 'BVARW', or 'DSGEVAR',

`obj`	An object of the above class.
`periods`	The forecast horizon.
`shocks`	Whether to include uncertainty about future shocks when calculating the distribution of forecasts.
`plot`	Whether to plot the forecasts.
`percentiles`	The percentiles of the conditional posterior distribution of forecasts to use for plotting.
`useMean`	Whether the user would prefer to use the mean of the forecast distribution rather than the middle value in 'percentiles'.

`backdata`	How many 'real' data points to plot before plotting the forecast. A broken line will indicate whether the 'real' data ends and the forecast begins.
`save`	Whether to save the plots.
`height`	If save=TRUE, use this to set the height of the plot.
`width`	If save=TRUE, use this to set the width of the plot.
	For objects of class 'CVAR',
`confint`	The confidence interval to use.
`...`	Additional arguments (not used).

Details

This function will work with DSGE, DSGE-VAR, and VAR models estimated with BMR.

Value

The function returns a plot of the forecast with user-selected percentiles, as well as the values used to create the plot; see the vignette for more details on the values returned.

Author(s)

Keith O'Hara

Examples

```
## Not run:
data(BMRVARData)
testbvarw <- BVARW(USMacroData[1:203,2:4],c(0.9,0.95,0.95),p=4,constant=T,
                   irf.periods=20,keep=10000,burnin=5000,
                   XiBeta=4,XiSigma=1,gamma=4)
forecast(testbvarw,periods=10,shocks=T,plot=T,percentiles=c(.05,.50,.95),backdata=10,save=FALSE)

## End(Not run)
```

`gacf` *Autocorrelation Function.*

Description

Plot an autocorrelation function using ggplot2.

Usage

```
gacf(y,lags=10,ci=.95,plot=TRUE,barcolor="purple",
     names=FALSE,save=FALSE,height=12,width=12)
```

Arguments

`y`	A matrix or data frame of size T x m containing the relevant series.
`lags`	The number of lags to plot.
`ci`	A numeric value between 0 and 1 specifying the confidence interval to use; the default value is 0.95.
`plot`	Whether to plot the ACF.
`barcolor`	The color of the bars.
`names`	Whether to plot the names of the series.
`save`	Whether to save the plots. The default is 'FALSE'.
`height`	If save = TRUE, use this to set the height of the plot.
`width`	If save = TRUE, use this to set the width of the plot.

Details

Plot an autocorrelation function (ACF) using ggplot2.

Value

Returns a plot of the ACF.

Author(s)

Keith O'Hara

Examples

```
## Not run:
data(BMRVARData)
gacf(USMacroData[,2:4],lags=12,ci=0.95,plot=T,barcolor="purple",
      names=T,save=T,height=6,width=12)

## End(Not run)
```

`gensys` *Gensys Solver.*

Description

Solve a DSGE model using Sims' method.

Usage

```
gensys(Gamma0,Gamma1,C,Psi,Pi)
```

Arguments

`Gamma0`	Coefficients on present-time variables.
`Gamma1`	Coefficients on lagged variables.
`C`	Intercept terms.
`Psi`	Coefficients on any exogenous shocks.
`Pi`	One-step-ahead expectational errors.

Details

For the technical details of Sims' method, see the accompanying vignette.

Value

The function returns an object of class 'gensys', which contains:

`G1`	Autoregressive solution matrix.
`Cons`	Intercept terms.
`impact`	Coefficients on the exogenous shocks.
`eu`	A 2 x 1 vector indicating existence and uniqueness (respectively) of the solution. A value of 1 can be read as 'yes', while 0 is 'no'.
`Psi`	User-specified shock matrix.
`Pi`	User-specified expectational errors matrix.

Author(s)

Keith O'Hara

References

Sims, Christopher A., *Solving Linear Rational Expectations Models*, Computational Economics, 20(2), 2002.

See Also

`IRF.gensys`, `DSGESim`.

Examples

```
## Not run:
dsgetest <- gensys(Gamma0,Gamma1,C,Psi,Pi)

## End(Not run)
```

`gpacf` *Partial Autocorrelation Function.*

Description

Plotting a partial autocorrelation function using ggplot2.

Usage

```
gpacf(y,lags=10,ci=.95,plot=TRUE,barcolor="darkred",
      names=FALSE,save=FALSE,height=12,width=12)
```

Arguments

`y`	A matrix or data frame of size T x m containing the relevant series.
`lags`	The number of lags to plot.
`ci`	A numeric value betweein 0 and 1 specifying the confidence interval to use; the default value is 0.95.
`plot`	Whether to plot the PACF.
`barcolor`	The color of the bars.
`names`	Whether to plot the names of the series.
`save`	Whether to save the plots. The default is 'FALSE'.
`height`	If save = TRUE, use this to set the height of the plot.
`width`	If save = TRUE, use this to set the width of the plot.

Details

Plot a partial autocorrelation function (PACF) using ggplot2.

Value

Returns a plot of the PACF.

Author(s)

Keith O'Hara

Examples

```
## Not run:
data(BMRVARData)
gpacf(USMacroData[,2:4],lags=12,ci=0.95,plot=T,barcolor="darkred",
      names=T,save=T,height=6,width=12)

## End(Not run)
```

`gtsplot` *Time-series plot using ggplot2.*

Description

A simple wrapper function for plotting time-series data with ggplot2.

Usage

```
gtsplot(X,dates=NULL,rowdates=FALSE,dates.format,
        save=FALSE,height=13,width=11)
```

Arguments

X	A matrix or data frame of size T x m containing the relevant time-series data, where m is the number of series.
dates	A T x 1 date or character vector containing the relevant date stamps for the data.
rowdates	A TRUE or FALSE statement indicating whether the row names of the X matrix contain the date stamps for the data.
dates.format	If 'dates' is not set to NULL, then indicate what format the dates are in, such as Year-Month-Day.
save	Whether to save the plot(s).
height	The height of the saved plot(s).
width	The width of the saved plot(s).

Value

Returns a plot of the data.

Author(s)

Keith O'Hara

Examples

```
## Not run:
data(BMRVARData)
gtsplot(USMacroData[,2:4],dates=USMacroData[,1])

## End(Not run)
```

IRF *Plotting IRFs.*

Description

A function for plotting impulse response funtions using ggplot2.

Usage

```
  IRF(obj,...)

  ## S3 method for class BVARM
IRF(obj,percentiles=c(.05,.50,.95),save=TRUE,height=13,width=13,...)
  ## S3 method for class BVARS
IRF(obj,percentiles=c(.05,.50,.95),save=TRUE,height=13,width=13,...)
  ## S3 method for class BVARW
IRF(obj,percentiles=c(.05,.50,.95),save=TRUE,height=13,width=13,...)
  ## S3 method for class CVAR
IRF(obj,percentiles=c(.05,.50,.95),save=TRUE,height=13,width=13,...)

  ## S3 method for class BVARTVP
IRF(obj,whichirfs=NULL,percentiles=c(.05,.50,.95),
```

```
  save=FALSE,height=13,width=13,...)

  ## S3 method for class DSGEVAR
IRF(obj,varnames=NULL,percentiles=c(.05,.50,.95),comparison=TRUE,
    save=TRUE,height=13,width=13,...)
  ## S3 method for class EDSGE
IRF(obj,observableIRFs=FALSE,varnames=NULL,percentiles=c(.05,.50,.95),
    save=TRUE,height=13,width=13,...)

  ## S3 method for class gensys
IRF(obj,shocks,irf.periods=20,varnames=NULL,
    plot=TRUE,save=FALSE,height=13,width=13,...)
  ## S3 method for class uhlig
IRF(obj,shocks,irf.periods=20,varnames=NULL,
    plot=TRUE,save=FALSE,height=13,width=13,...)
  ## S3 method for class SDSGE
IRF(obj,shocks,irf.periods=20,varnames=NULL,
    plot=TRUE,save=FALSE,height=13,width=13,...)
```

Arguments

	For objects of class 'BVARM', 'BVARS', 'BVARW', or 'CVAR',
obj	An object of the above class.
percentiles	The percentiles of the distribution the user wants to use.
save	Whether to save the plots.
height	If save=TRUE, use this to set the height of the plot.
width	If save=TRUE, use this to set the width of the plot.
	For objects of class 'BVARTVP',
whichirfs	Which IRFs to plot. (The default is to plot all of the IRFs contained in the estimation object.)
	For objects of class 'DSGEVAR',
varnames	A character vector with the names of the relevant variables.
comparison	Whether to plot corresponding DSGE model IRFs.
	For objects of class 'EDSGE',
observableIRFs	Whether to plot the IRFs relating to the state variables, or the implied IRFs of the observable series.
	For objects of class 'gensys', 'uhlig', or 'SDSGE',
shocks	A numeric vector containing the standard deviations of the shocks.
irf.periods	The horizon of the IRFs.
plot	Whether to plot the IRFs.
...	Additional arguments (not used).

Details

This function will work with any estimated VAR model, estimated DSGE or DSGE-VAR model, or solved DSGE model.

Value

The function returns a plot of the IRFs with user-selected percentiles.

Author(s)

Keith O'Hara

Examples

```
## Not run:
data(BMRVARData)
testbvarw <- BVARW(USMacroData[,2:4],c(0.9,0.95,0.95),p=4,constant=T,
                   irf.periods=20,keep=10000,burnin=5000,
                   XiBeta=4,XiSigma=1,gamma=4)
IRF(testbvarw,percentiles=c(0.05,0.5,0.95),save=FALSE)

## End(Not run)
```

---

modecheck *Check the Posterior Mode.*

Description

Plot the log posterior around the posterior mode values.

Usage

```
modecheck(obj,...)

## S3 method for class DSGEVAR
modecheck(obj,gridsize=200,scalepar=NULL,plottransform=FALSE,parnames=NULL,
  save=FALSE,height=13,width=13,...)
## S3 method for class EDSGE
modecheck(obj,gridsize=200,scalepar=NULL,plottransform=FALSE,parnames=NULL,
  save=FALSE,height=13,width=13,...)
```

Arguments

obj	An object of class 'EDSGE' or 'DSGEVAR'.
gridsize	The number of grid points to use when calculating the log posterior around the mode values.
scalepar	A value to replace the scaling parameter from estimation ('c') when plotting the log posterior.
plottransform	Whether to plot the transformed values (i.e., such that the support of each parameter is unbounded), or to plot the untransformed values.
parnames	A vector of expressions containing the name of each parameter.

`save`	Whether to save the plot(s).
`height`	If save=TRUE, use this to set the height of the plot(s).
`width`	If save=TRUE, use this to set the width of the plot(s).
`...`	Additional arguments (not used).

Details

For an example, see the accompanying vignette.

Value

The function will plot the value of the log posterior around the posterior mode.

Author(s)

Keith O'Hara

Examples

```
## Not run:
modecheck(NKMest,200,1,FALSE,save=FALSE)

## End(Not run)
```

---

plot *Plot BMR Objects.*

Description

Plot the output of a BVAR, DSGEVAR, or EDSGE object.

Usage

```
  ## S3 method for class BVARM
plot(x,type=1,save=FALSE,height=13,width=13,...)
  ## S3 method for class BVARS
plot(x,type=1,plotSigma=TRUE,save=FALSE,height=13,width=13,...)
  ## S3 method for class BVARTVP
plot(x,percentiles=c(.05,.50,.95),save=FALSE,height=13,width=13,...)
  ## S3 method for class BVARW
plot(x,type=1,plotSigma=TRUE,save=FALSE,height=13,width=13,...)

  ## S3 method for class DSGEVAR
plot(x,parnames=NULL,BinDenom=40,MCMCplot=FALSE,save=FALSE,height=13,width=13,...)
  ## S3 method for class EDSGE
plot(x,parnames=NULL,BinDenom=40,MCMCplot=FALSE,save=FALSE,height=13,width=13,...)
```

Arguments

For objects of class ‘BVARM’,

An object of the above class.

`x` `type` An integer value indicating the plot style; type=1 will produce a histogram, while type=2 will use smoothed densities.

`save` Whether to save the plots.

`height` If save=TRUE, use this to set the height of the plot.

`width` If save=TRUE, use this to set the width of the plot.

For objects of class ‘BVARS’ or ‘BVARW’,

`plotSigma` Whether to plot the elements of the residual covariance matrix.

For objects of class ‘BVARTVP’,

`percentiles` Which percentiles of the posterior distribution the function should use.

For objects of class ‘DSGEVAR’ or ‘EDSGE’,

`parnames` A vector of expressions containing the name of each parameter.

`BinDenom` Bin width.

`MCMCplot` Whether to plot a trace of the MCMC run.

`...` Additional arguments (not used).

Details

For examples of the plotting features, see the accompanying vignette.

Value

Plots the marginal posterior distributions of the relevant coefficients/parameters.

Author(s)

Keith O’Hara

Examples

```
## Not run:
data(BMRVARData)
prior<-c(0.9,0.95,0.95)
testbvarm <- BVARM(USMacroData[,2:4],prior,p=4,constant=T,irf.periods=20,
                   keep=10000,burnin=5000,VType=1,
                   HP1=0.5,HP2=0.5,HP3=10)
plot(testbvarm,save=F)
IRF(testbvarm,save=F)
forecast(testbvarm,shocks=TRUE,backdata=10,save=FALSE)

## End(Not run)
```

`prior` *Parameterize the Prior Distributions.*

Description

Plot and print the moments of a specified prior distribution.

Usage

```
prior(priorform,priorpars,parname=NULL,moments=TRUE,NR=NULL,NC=NULL)
```

Arguments

`priorform`	This should be a valid prior form for the EDSGE or DSGEVAR functions, such as “Gamma” or “Beta”.
`priorpars`	The relevant parameters of the distribution.
`parname`	A title for the plot.
`moments`	Whether to print the mean, mode, and variance of the distribution.
`NR`	For use with multiple plots. See the vignette for an example.
`NC`	For use with multiple plots. See the vignette for an example.

Details

This function can be used when selecting appropriate prior distributions for the EDSGE and DSGEVAR functions.

Author(s)

Keith O’Hara

Examples

```
## Not run:
prior("Normal",c(0,1))
prior("Gamma",c(2,2))

## End(Not run)
```

`SDSGE` *Solve a DSGE Model.*

Description

Solve a DSGE model using one of the available solvers in BMR.

Usage

```
SDSGE(mats,type=NULL)
```

Arguments

`mats`	The relevant matrices.
`type`	Which solver to use: 1 (gensys) or 2 (uhlig). If type=NULL, the solver will attempt to detect which method to use based on the names of the objects contained in 'mats'.

Details

For technical details of the available solution methods, see the accompanying vignette.

Value

The function returns an object of class 'SDSGE', which contains the output of either 'gensys' or 'uhlig'.

Author(s)

Keith O'Hara

References

Sims, Christopher A., *Solving Linear Rational Expectations Models*, Computational Economics, 20(2), 2002.

Uhlig, Harald, *A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily* Computational Methods for the Study of Dynamic Economics, Oxford University Press, 1999.

See Also

`IRF.SDSGE`, `DSGESim`.

Examples

```
## Not run:
mats <- list()
mats$Gamma0 <- Gamma0; mats$Gamma1 <- Gamma1; mats$C <- C; mats$Psi <- Psi; mats$Pi <- Pi
dsgetest <- SDSGE(mats)

## End(Not run)
```

`states` *Plot State Variables.*

Description

Plot the filtered state variables of an estimated DSGE or DSGE-VAR model.

Usage

```
states(obj,...)

## S3 method for class DSGEVAR
states(obj,percentiles=c(.05,.50,.95),
       varnames=NULL,useMean=FALSE,
       save=FALSE,height=13,width=11,...)
## S3 method for class EDSGE
states(obj,percentiles=c(.05,.50,.95),
       varnames=NULL,useMean=FALSE,
       save=FALSE,height=13,width=11,...)
```

Arguments

	For objects of class 'EDSGE' or 'DSGEVAR',
`obj`	An object of class 'EDSGE' or 'DSGEVAR'.
`percentiles`	Which percentiles of the distribution to use.
`varnames`	Name labels for the states.
`useMean`	Whether the user would prefer to use the mean of the forecast distribution rather than the middle value in 'percentiles'.
`save`	Whether to save the plots.
`height`	If save=TRUE, use this to set the height of the plot.
`width`	If save=TRUE, use this to set the width of the plot.
`...`	Additional arguments (not used).

Details

This function will work with estimated DSGE and DSGEVAR models.

Value

The function returns a plot of the states with user-selected percentiles, as well as the values used to create the plot; see the vignette for more details on the values returned.

Author(s)

Keith O'Hara

Examples

```
## Not run:
NKMest <- EDSGE(dsgedata,chains=1,cores=1,
                ObserveMat,initialvals,partomats,
                priorform,priorpars,parbounds,parnames,
                optimMethod=c("Nelder-Mead","CG"),
                optimLower=NULL,optimUpper=NULL,
                optimControl=list(maxit=10000),
                DSGEIRFs=TRUE,irf.periods=40,
                scalepar=0.27,keep=50000,burnin=75000,
                tables=TRUE)
#
```

```
varnames=c("Output Gap","Output","Inflation","Natural Int",
           "Nominal Int","Labour Supply","Technology","MonetaryPolicy")
states(NKMest,percentiles=c(0.01,0.50,0.99),varnames=varnames)

## End(Not run)
```

---

statespace *DSGE State-Space Format.*

Description

State-space representation of a DSGE model.

Usage

```
statespace(obj)
```

Arguments

obj	An object of type 'gensys', 'uhlig', or 'SDSGE'.

Details

For technical details, see the accompanying vignette.

Value

The function constructs a state-space representation for the various DSGE solvers in BMR.

Author(s)

Keith O'Hara

See Also

IRF, DSGESim.

Examples

```
## Not run:
mats <- list()
mats$Gamma0 <- Gamma0; mats$Gamma1 <- Gamma1; mats$C <- C; mats$Psi <- Psi; mats$Pi <- Pi
dsgetest <- SDSGE(mats)
ssmats <- statespace(dsgetest)

## End(Not run)
```

stationarity	*Testing for Stationarity.*

Description

ADF test of a unit root and KPSS test of stationarity.

Usage

```
stationarity(y,KPSSp=4,ADFp=8,print=TRUE)
```

Arguments

y	A matrix or data frame containing the series to be used in testing, and should be of size T x m.
KPSSp	The number of lags to include for KPSS test.
ADFp	The maximum number of (first-differenced) lags to include in the Augmented Dickey-Fuller (ADF) tests. Lag order is determined by minimising the Bayesian information criterion.
print	A logical statement on whether the test results should be printed in the output screen. the default is 'TRUE'.

Details

Remember, the null hypothesis of the ADF test is NOT the same as the KPSS test; the former is a test with a null of a unit root, while the latter is a test with a null of stationarity.

Value

Returns a list with

KPSS	KPSS test statistics and critical values.
ADF	ADF test statistics and critical values.
ADFLag	Number of lags used for the ADF tests.

Author(s)

Keith O'Hara

References

Hamilton, James D., *Time Series Analysis,* Princeton, New Jersey: Princeton University Press, 1994.

Kwiatkowski, Denis, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin, "Testing the Null Hypothesis of Stationarity Against the Alternative of a Unit Root," *Journal of Econometrics,* 1992, *54* (1), 159-178.

Examples

```
## Not run:
stationarity(USMacroData[,2:4],4,8)

## End(Not run)
```

`uhlig` *Uhlig's Method.*

Description

Solve a DSGE model using Uhlig's method.

Usage

```
uhlig(A,B,C,D,F,G,H,J,K,L,M,N,whichEig=NULL)
```

Arguments

`A,B,C,D`	The 'uhlig' function requires the three blocks of matrices, with 12 matrices in total. The A, B, C, and D matrices form the deterministic block.
`F,G,H,J,K,L,M`	The F, G, and H matrices form the expectational block for the control variables. The J and K matrices are for the 'jump' variables, and L and M are for the exogenous shocks.
`N`	The N matrix defines the autoregressive structure of any exogenous shocks.
`whichEig`	The function will return the eigenvalues and (right) eigenvectors used to construct the solution matrices, with the eigenvalues sorted in order of smallest to largest (in absolute value). By default, BMR will select the first (smallest) m eigenvalues (out of a total of 2m eigenvalues). However, if you prefer to select the eigenvalues yourself, then enter a numeric vector of length m indicating which elements of the eigenvalue matrix you wish to use.

Details

For the technical details of Uhlig's method, see the accompanying vignette.

Value

The function returns an object of class 'uhlig', which contains:

`N`	The user-specified N matrix, defining the autoregressive structure of any exogenous shocks.
`P`	The P matrix from Uhlig's solution.
`Q`	The Q matrix from Uhlig's solution.
`R`	The R matrix from Uhlig's solution.
`S`	The S matrix from Uhlig's solution.
`EigenValues`	The sorted eigenvalues that form the solution to the P matrix. If a situation of plus/minus infinity in the real part of an eigenvalue (with a corresponding NaN-valued imaginary part) arises, the eigenvalue will be set to 1E+07 +0i.
`EigenVectors`	The eigenvectors corresponding to the sorted eigenvalues.

Author(s)

Keith O'Hara

References

Uhlig, Harald, *A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily* Computational Methods for the Study of Dynamic Economics, Oxford University Press, 1999.

See Also

`IRF.uhlig`, `DSGESim`.

Examples

```
## Not run:
dsgetest <- uhlig(A,B,C,D,F,G,H,J,K,L,M,N)

## End(Not run)
```

Index