Package 'gEcon.estimation'

January 18, 2023

Encoding UTF-8

Title DSGE models estimation

Description The package provides routines for estimating dynamic stochastic general equilibrium models (DSGE) written using gEcon package. The user may specify prior distribution for the model parameters and perform Bayesian estimation of the model or use classical, maximum likelihood estimation method. The package also includes implementation of historical shock decomposition, Kalman smoother, and functions for forecasting which may be helpful in model analysis and applications.

License_restricts_use yes

License file LICENCE

Copyright Chancellery of the Prime Minister of the Republic of Poland 2014-2015 Karol Podemski 2015-2023

Imports coda, doParallel, foreach, KFAS, Matrix, methods, numDeriv, parallel

Depends gEcon(>= 1.2.1), R(>= 4.2.2), FKF(>= 0.2.4)

Suggests GenSA

Version 0.1.1

Date 2023-01-18

Author Karol Podemski, Paweł Romański, Michał Marciniak (plotting functions), idea and design by Grzegorz Klima

Maintainer Karol Podemski <karol.podemski@gmail.com>

Collate gecon_prior.R gecon_estimation_results.R maximisation_kernel.R bayesian_estimation.R bfgsi.R csminit.R csminwel.R csminwelNew.R csolve.R derivVec.R distribution_densities.R distribution_coord.R distribution_moments.R distribution_parameters.R marginal_density.R forecast.R gecon.estimation.R plots.R likelihood.R lyapunov.R ml_estimation.R numgrad.R numHess.R plot_tools.R shock_decomposition.R smoother.R utils.R

NeedsCompilation no

R topics documented:

`gecon.estimation-package`

Dynamic stochastic general equilibrium (DSGE) models estimation

---

Description

Package for estimating dynamic stochastic general equilibrium models.

Details

This package allows for estimation of models developed in `gEcon` using either Bayesian methods (`bayesian_estimation`) or classical maximum likelihood approach `ml_estimation`). In addition, the package includes implementation of historical shock decomposition (`shock_decomposition`), Kalman smoother for unobservable variables (`smoother`), and forecasting routines (`forecast`) and (`forecast_posterior`) which may be helpful in model analysis and applications.

The likelihood function is constructed using Kalman filter and state-space representation of model based on the 1st order perturbation solution. This allows for estimation of the model parameters using either Bayesian or classical (maximum likelihood) methods. Bayesian approach involves combining the likelihood function with prior distributions of the model parameters to obtain posterior distributions of parameters. A point estimate for model parameters can be obtained as a statistic of posterior distribution (minimising specific Bayesian loss function, e.g. mean minimises quadratic loss function). The maximum likelihood method involves maximisation of the likelihood function with respect to model parameters (parameter vector maximising the likelihood function is the point estimate).

Mapping of the DSGE models into state-space representation requires brief description as it can give the user an insight into how to prepare model and data for estimation. The matrices used in the state-space representation are created based on the solution of the log-linearised model (model variables are expressed as percentage deviations from their steady-state values) and do not contain intercept or deterministic term (do not allow for deterministic trend nor seasonality). This implies that to obtain meaningful results from estimation process, the data used for estimation have to be filtered for trend, seasonally adjusted, and demeaned. This can be done either by HP-filtering or computing log differences and then demeaning the resulting series.

The following state-space representation is assumed by the package:

$$X_t = Hx_t + Ju_t,$$

$$x_t = F(\theta)x_{t-1} + G(\theta)e_t,$$

$$e_t \sim N(0, \Sigma),$$

where X_t denotes the vector of observable variables (of length m), x_t stands for the vector of model variables (of length $m + s$, where s is the number of state variables). The second equation is called transition equation for the unobservable variables (x_t, the model variables) while the first equation - observation equation - maps these variables into the observable data. In the current release of the package, the state-space representation cannot contain explicit measurement errors (the J has all entries equal to zero). The H matrix (with dimensions $m \times m + s$) contains mapping of x_t to the X_t vector of observables. In each row it has one entry equal to 1 with all entries set to zero.

The $F(\theta)$ and $G(\theta)$ matrices from transition equation are created from gEcon model's solution matrices (P, Q, R, S, for details see gEcon package Users' guide). The $F(\theta)$ matrix (with dimension $(m + s) \times (m + s)$) is created by stacking rows of R matrix from the solution of DSGE model corresponding to observable variables and the entire P matrix. The $G(\theta)$ matrix is created by stacking rows of S matrix from the solution of log-linearised DSGE model corresponding to observable variables and the entire Q matrix. Obviously, the $F(\theta)$ and $G(\theta)$ matrices are dependent on model parameters (θ). The shock vector (e_t) is assumed to be normally distributed with mean zero. The variance-covariance matrix (Σ) can be estimated or (if assumed or known) set in object of `gecon_model` class.

Author(s)

Karol Podemski, Paweł Romański, Michał Marciniak (plotting functions), idea and design by Grzegorz Klima

Examples

```
# ####################################################################
# 1. prepare gEcon model

# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# solve the model
dsge_model <- steady_state(dsge_model)
dsge_model <- solve_pert(dsge_model, loglin = TRUE)

# set the stochastic shocks distribution parameters
dsge_model <- set_shock_distr_par(dsge_model,
                                  distr_par = list("sd( epsilon_G )" = 0.01,
```

```
                                                    "sd( epsilon_Z )" = 0.01))
shock_info(model = dsge_model, all = TRUE)

# ###################################################################
# 2. simulate the model to obtain data for the estimation

# choose variables of interest
set.seed(250)
series_length <- 150
observables <- c("Y", "G")

# simulate random path
dsge_simulation <- random_path(model = dsge_model,
                               sim_length = series_length,
                               variables = observables)
model_data <- get_simulation_results(dsge_simulation)

# create data set to be used for estimation (ts object)
estimation_data <- ts(data = t(model_data)[, observables],
                      start = c(1973, 1),
                      frequency = 4, names = observables)

# remove mean from the data series
mean_var <- matrix(apply(estimation_data, 2, mean),
                   byrow = TRUE,
                   nrow = nrow(estimation_data),
                   ncol = ncol(estimation_data))
estimation_data <- estimation_data  - mean_var

# ###################################################################
# 3. declare prior distribution

dsge_prior <- gecon_prior(
    prior_list = list(
        list(par = "sd(epsilon_Z)", type = "inv_gamma",
             mean = 0.012, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.0012),
        list(par = "sd(epsilon_G)", type = "inv_gamma",
             mean = 0.008, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.006),
        list(par = "omega", type = "normal",
             mean = 1.45, sd = 0.1, lower_bound = 1,
             upper_bound  = 2, initial = 1.5),
        list(par = "phi_G", type = "beta",
             mean = 0.88, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.95),
        list(par = "phi_Z", type = "beta",
             mean = 0.92, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.95)),
    model = dsge_model)

plot_prior(dsge_prior)

# ###################################################################
# 4. estimate the model (Bayesian estimation)

estimation_result <- bayesian_estimation(data_set = estimation_data,
```

```
                                        optim_options_list = list(solver = "csminwel"),
                                        mcmc_options_list = list(chain_length = 1000,
                                                                 burn = 200,
                                                                 cores = 2, chains = 2,
                                                                 scale = rep(0.5, 5)),
                                        observables =  observables,
                                        model = dsge_model,
                                        prior = dsge_prior)


plot_posterior(estimation_result)
# retrieve estimates
#
# true model parameters were:
# sd(epsilon_Z) 0.01
# sd(epsilon_G) 0.01
# omega         1.45
# phi_G         0.9
# phi_Z         0.9
est_par <- get_estimated_par(estimation_result)
free_par <- est_par$free_par
shock_distr_par <- est_par$shock_distr_par
estimated_dsge_model <- set_free_par(dsge_model, free_par = free_par)
estimated_dsge_model <- set_shock_distr_par(estimated_dsge_model, distr_par = shock_distr_par)

estimated_dsge_model <- steady_state(estimated_dsge_model)
estimated_dsge_model <- solve_pert(estimated_dsge_model, loglin = TRUE)

# ####################################################################
# 5. historical shock decomposition and variable smoothing

# find historical shock decomposition
dsge_shock_decomp <- shock_decomposition(model = estimated_dsge_model,
                                        data_set = window(estimation_data,
                                                          start = c(2004, 1),
                                                          end = c(2010, 1),
                                                          frequency = 4),
                                        observables = observables,
                                        variables = observables)
plot_shock_decomposition(dsge_shock_decomp)

# use Kalman smoother to obtain smoothed variables' values
dsge_smoothed_variables <- smoother(model = estimated_dsge_model,
                                    data_set = estimation_data,
                                    observables = c("Y", "G"),
                                    variables = c("K", "I", "C"))

# print smoothed shocks' values
dsge_smoothed_variables$smoothed_shock
# print smoothed variables' values
dsge_smoothed_variables$smoothed_var
# print the MSE matrix
dsge_smoothed_variables$MSE

# ####################################################################
# 6. forecast using the model
```

```
# forecast using point estimates of parameters
fc_res <- forecast(model = estimated_dsge_model,
                   data_set = estimation_data,
                   observables = observables,
                   variables = c("Y", "G"),
                   horizon = 20)

# forecast using posterior distribution
fc_res_post <- forecast_posterior(est_results = estimation_result,
                                  data_set = estimation_data,
                                  observables = observables,
                                  variables = c("Y", "G"),
                                  horizon = 20)

# plot forecasts
plot_forecast(fc_res_post)
plot_forecast(fc_res)
```

`bayesian_estimation` *Bayesian estimation of DSGE models*

Description

The `bayesian_estimation` function performs Bayesian estimation of model parameters. It has the following properties:

- uses full-information likelihood implied by the model approximation rather than selected statistics (e.g. discrepancy between model and VAR implied IRFs),
- allows for incorporating additional information (e.g. from panel or microeconometric studies) about possible model parameter distribution by formulation of the prior distribution for parameters,
- allows for comparison of different model specifications by simple probabilistic formulas.

Bayesian approach to estimation is relatively simple from conceptual point of view, yet numerically complex. It is based on Bayes' formula:

$$P(\theta|Y) = \frac{P(Y|\theta)P(\theta)}{P(Y)},$$

where θ are model parameters, and Y is the data used for estimation. The left-hand side expression $P(\theta|Y)$ is called a posterior density. The $P(Y|\theta)$ expression is the likelihood function. The unconditional distribution of parameters is called prior distribution ($P(\theta)$) and allows for incorporation of the additional information into the estimation process. Probability of data $P(Y)$ is usually irrelevant for estimation so the expression for posterior density can be simplified to:

$$P(\theta|Y) \sim P(Y|\theta)P(\theta).$$

As there is no explicit expression for posterior density, it has to be simulated numerically. The `gEcon.estimation` package uses random walk Metropolis-Hastings algorithm to sample from posterior distribution (see Details section).

Usage

```
bayesian_estimation(data_set, observables, model, prior,
  optim_options_list = NULL, mcmc_options_list = NULL)
```

Arguments

`data_set` an object of `ts` or `mts` class containing data for the estimation of the model. The time series supplied have to be stationary, detrended, and deseasoned. The number of series supplied cannot be greater than the number of shocks in the model. Estimation based on data sets with missing observations is not supported in the current version.

`observables` a character with the length equal to the number of `data_set` columns. It specifies the names of observable variables (corresponding to model variables).

`model` an object of `gecon_model` class.

`prior` an object of `gecon_prior` class; declares the prior distribution of estimated parameters.

`optim_options_list`
a list of options for the posterior kernel maximisation routines:

- `solver` a character; the name of solver:
  - `csminwelNew` - `csminwelNew` solver (updated version of `csminwel`),
  - `csminwel` - `csminwel` solver (default),
  - `Nelder-Mead` - the Nelder-Mead routine implemented in the `optim` function,
  - `GenSA` - a simulated annealing solver from `GenSA` package.
- `solver_crit` a numeric, the convergence criterion for a solver. The default value is 1e-7. Not used by the `GenSA` solver.
- `solver_iter` a numeric, the maximum number of iterations for a solver. The default value is 1000.
- `init_hess_scale` a numeric, size of the diagonal elements of initial Hessian used by `csminwel` and `csminwelNew` procedures. Not used by `Nelder-Mead` and `GenSA` solvers. The default value is 1e-4.
- `temperature` a numeric, the initial temperature for `GenSA` solver. The default value is 5230.
- `visiting.param` a numeric, a visiting parameter for `GenSA` solver. The default value is 2.62.
- `acceptance.param` a numeric, an acceptance parameter for `GenSA` solver. The default value is -5.
- `max.call` a numeric, the maximum number of function calls if `GenSA` solver is used. The default value is 1e+06.
- `solver_hess` a logical value. If set to TRUE, the Hessian returned from the numerical solver is used for computation of variance of sampling distribution. By setting this value to TRUE, one can obtain only approximate Hessian. However, such an approximate Hessian may be more often positive definite (thanks to rank-one updates) than one computed explicitly. The `Nelder-Mead` and `GenSA` solvers do not return Hessian. The default value is FALSE.
- `hessian_d` a numeric, delta used to compute Hessian numerically. The default value is 0.001.
- `hessian_eps` a numeric, tolerance of the routine computing Hessian. The default value is 1e-7.

`mcmc_options_list`
a list containing options of the posterior density simulator:

- `chains` a numeric, the number of Markov chains. The default value is 2 (if the number of cores is not specified) or the number of cores (if this number is specified by the user).
- `cores` a numeric, the number of processor cores used for estimation. The default value is 2 (if the number of chains is not specified) or the smallest value from the number of CPU cores and the number of chains (if number of chains is specified).
- `burn` a numeric, the number of initial draws in each chain which will be discarded. The default value is 200.
- `chain_length` a numeric, the number of draws which will be left in each chain. The default value is 500.
- `scale` a vector of scale parameters for candidate draws. It should be tailored to ensure acceptance rate between 0.25 and 0.33. If acceptance rate is too high, this argument should rise. In the opposite case, it should be lowered. The default value is a vector of ones.
- `output_freq` a numeric, frequency of output. The Metropolis-Hastings algorithm run is divided into bundles. After each of the bundles is processed, short information about the progress is printed. Frequent output may influence computation time significantly. The default output frequency is 250.

Details

The `bayesian_estimation` function uses random walk Metropolis-Hastings procedure to draw from posterior distribution (precisely, from the log-transformed posterior distribution: $\log(P(\theta|Y)) \sim \log(P(Y|\theta)) + \log(P(\theta))$). In this procedure, candidate draws ($\theta^\star$) are sampled from multivariate normal distribution $N(\theta^{s-1}, \tilde{\Sigma})$, with mean dependent on previously accepted draw (θ^{s-1}). The chain will converge to sample from posterior distribution if the proper variance of sampling density ($\tilde{\Sigma}$) and good initial draw are used. `gEcon.estimation` package finds variance of sampling density and initial points by performing maximisation of the logarithm of the posterior kernel $\log P(Y|\theta) + \log P(\theta)$. The inverse of Hessian of the kernel at the mode is taken as sampling density variance, while mode of the posterior kernel ($\tilde{\theta}$) is used as initial draw (in case of one Markov Chain). When many chains are used, initial draws for each of chains are sampled from $N(\tilde{\theta}, \tilde{\Sigma})$.

The following steps are performed iteratively. The candidate draw $\theta^\star$ is taken from candidate density. Then jump from θ^{s-1} is accepted ($\theta^s = \theta^\star$) with probability:

$$\min\{1, \frac{P(\theta^\star|Y)P(\theta^\star)}{P(\theta^{(s-1)}|Y)P(\theta^{(s-1)})}\}$$

and rejected otherwise ($\theta^s = \theta^{s-1}$). As the result, chain of draws from the posterior distribution is obtained $\theta = (\theta^1, \theta^2, ..., \theta^N)$. Any function $g(\theta)$ may be then computed using that chain and treated as point estimate of parameter (usually mean or median is chosen). The simulated density is stored in objects of `gecon_estimation_results` class.

This procedure may be time consuming due to two computationally complex steps. The maximisation of the posterior kernel is generally difficult task due to many local optima of this function and dimension of the problem. Secondly, each draw from sampling density in the Metropolis-Hastings algorithm may require updating steady state and perturbation solution.

The `bayesian_estimation` function employs one of four numerical solvers in order to maximise the posterior kernel. For more information about `csminwel` and `csminwelNew` solvers, see http://sims.princeton.edu/yftp/optimize/. The simulated annealing solver from the `GenSA` package is robust against falling into local optima but its efficiency is highly dependent on parametrisation (initial temperature, visiting parameter).

Value

The function returns an object of gEcon_estimation_results class with the estimation results.

Examples

```
# ####################################################################
# 1. prepare gEcon model

# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# solve the model
dsge_model <- steady_state(dsge_model)
dsge_model <- solve_pert(dsge_model, loglin = TRUE)

# set the stochastic shocks distribution parameters
dsge_model <- set_shock_distr_par(dsge_model,
                                  distr_par = list("sd( epsilon_G )" = 0.01,
                                                   "sd( epsilon_Z )" = 0.01))
shock_info(model = dsge_model, all = TRUE)

# load data
data(example_estimation_data)

# ####################################################################
# 2. declare prior distribution

dsge_prior <- gecon_prior(
    prior_list = list(
        list(par = "sd(epsilon_Z)", type = "inv_gamma",
             mean = 0.012, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "sd(epsilon_G)", type = "inv_gamma",
             mean = 0.008, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "omega", type = "normal",
             mean = 1.45, sd = 0.1, lower_bound = 1,
             upper_bound  = 2, initial = 1.5),
        list(par = "phi_G", type = "beta",
             mean = 0.88, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.92),
        list(par = "phi_Z", type = "beta",
             mean = 0.92, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.96)),
    model = dsge_model)

plot_prior(dsge_prior)

# ####################################################################
# 3. estimate the model (Bayesian estimation)

estimation_result <- bayesian_estimation(data_set = example_estimation_data,
                                         optim_options_list = list(solver = "csminwelNew"),
                                            mcmc_options_list = list(chain_length = 1000,
```

```
                                                    burn = 200,
                                                    cores = 2, chains = 2,
                                                    scale = rep(0.5, 5)),
                                        observables =  c("Y", "G"),
                                        model = dsge_model,
                                        prior = dsge_prior)

# retrieve estimates
plot_posterior(estimation_result)

# true model parameters were:
# sd(epsilon_Z) 0.01
# sd(epsilon_G) 0.01
# omega         1.45
# phi_G         0.9
# phi_Z         0.9
est_par <- get_estimated_par(estimation_result)
free_par <- est_par$free_par
shock_distr_par <- est_par$shock_distr_par

# update the model
estimated_dsge_model <- set_free_par(dsge_model, free_par = free_par)
estimated_dsge_model <- set_shock_distr_par(estimated_dsge_model, distr_par = shock_distr_par)

# solve the updated model
estimated_dsge_model <- steady_state(estimated_dsge_model)
estimated_dsge_model <- solve_pert(estimated_dsge_model, loglin = TRUE)
```

`create_mcmc.list` *Create an object of* `mcmc.list` *class*

Description

The `create_mcmc.list` function retrieves Markov chains from an object of `gecon_estimation_results` class and transforms them into an object which may be analysed by the `coda-package` package.

Usage

```
create_mcmc.list(est_results)
```

Arguments

`est_results`	an object of `gecon_estimation_results` class.

Value

The function returns a `mcmc.list` object created based on the estimation results.

Examples

```
# load example estimation results
data(example_estimation_result)
summary(example_estimation_result)
```

```
# create an object for the analysis with the coda package
dsge_mcmc_list <- create_mcmc.list(example_estimation_result)
```

`estimated_model` *Example estimated DSGE model*

Description

An estimated DSGE model specified in the `dsge_model.gcn` file (the file can be found in the package directory). The data from `example_estimation_data` were used to estimate model.

Usage

```
data(estimated_model)
```

`example_estimation_data`
Example data generated from DSGE model

Description

An example data set generated from DSGE model specified in the `dsge_model.gcn` file (the file can be found in the package directory).

Usage

```
data(example_estimation_data)
```

`example_estimation_result`
Example `gecon_estimation_results` *object*

Description

An example `gecon_estimation_results` object, created by estimating DSGE model specified in the `dsge_model.gcn` file (the file can be found in the package directory) using `bayesian_estimation` function.

Usage

```
data(example_estimation_result)
```

forecast *DSGE model-based forecasting*

Description

The forecast function computes n-periods ahead forecasts and mean squared errors for the forecasts based on gecon_model objects.

Usage

```
forecast(model, data_set, observables, variables, horizon = 4)
```

Arguments

model	an object of gecon_model class.
data_set	an object of ts class of observable variables historical realisations.
observables	a character with the length equal to the number of data_set argument columns. Specifies names of observable variables supplied in the data_set argument.
variables	a character vector of the forecasted variables' names.
horizon	a positive numeric, the horizon of the forecast. The default value is 4.

Details

This function can be used to forecast based on either calibrated or estimated model. If model was estimated using either bayesian_estimation or ml_estimation function the estimated parameters should be retrieved from gecon_estimation_results object using the get_estimated_par function and assigned to the appropriate gecon_model object.

The forecasts are based on state-space system generated by the model. To obtain starting values of the state variables for the forecast and the initial MSE, Kalman smoother routine is run. The detailed description of forecasting based on state-space systems can be found in Hamilton, J. D. 1994. *Time series analysis*. Princeton, NJ: Princeton Univ. Press.

Value

The function returns a list with four objects:

- forec__ a ts object of the forecasts,
- forec_err__ a ts object of the forecasts' mean squared errors,
- data_set a ts object of the data supplied to the forecasting function,
- variables_tex a character vector of variables' LaTeX names,
- model_info a character vector of the length 3, information about the model for which the decomposition has been computed: the input file name, the input file path, and the date of creation.

See Also

For the models estimated using Bayesian methods, the forecast_posterior function allows to use full posterior distribution for forecasting.

Examples

```
# load estimated model and data
data(estimated_model)
data(example_estimation_data)

# forecast based on data
hp_forecast <- forecast(model = estimated_model,
                        data_set = example_estimation_data,
                        variables = c("Y"),
                        observables = c("Y", "G"),
                        horizon = 10)

plot_forecast(hp_forecast, conf_band = 0.5)
```

`forecast_posterior` *DSGE model-based forecasting using posterior distribution*

Description

The `forecast_posterior` function allows to create forecasts by sampling parameter values from the posterior distribution, solving model, and simulating it for the `n`-periods ahead. The initial values of state variables for the forecast are obtained by the application of Kalman smoother.

Usage

```
forecast_posterior(est_results, data_set, observables, variables, horizon = 4,
  posterior_sample = NULL, shock_uncertainty = TRUE)
```

Arguments

`est_results`	an object of `gecon_estimation_results` class returned by the `bayes_estimation` function.
`data_set`	an object of `ts` class of observable variables historical realisations.
`observables`	a character with the length equal to the number of `data_set` argument columns. Specifies names of observable variables supplied in the `data_set` argument.
`variables`	a character vector of the forecasted variables' names.
`horizon`	a positive numeric, the horizon of the forecast. The default value is 4.
`posterior_sample`	a positive numeric, the size of the posterior sample. By default, this number equals to 1/4 of the posterior sample size but not less than 250.
`shock_uncertainty`	a logical. If TRUE, uncertainty associated with the structural shocks is taken into account when computing the confidence intervals.

Details

If the function is called with the `shock_uncertainty` parameter set to TRUE, the uncertainty related to the structural innovations is taken into account. If the above mentioned parameter is set to FALSE, only the parameter uncertainty influences the confidence bands.

Value

The function returns a list with three elements:

- `forecasts` a list of lists. Each sublist is a matrix containing forecasts (subsequent periods in columns) generated for specific values of parameters (rows).
- `data_set` a `ts` object storing the data based on which the model has been estimated,
- `data_set_var` a character vector of data set variable names,
- `variables_tex` a character vector of variables' LaTeX names,
- `model_info` a character vector of length 3, containing information about the model for which the decomposition has been computed: the input file name, the input file path, and the date of creation.

Examples

```
# load example data
data(example_estimation_data)

# load estimation results
data(example_estimation_result)

# forecast taking into account shock and parameter uncertainty
post_forecast_shock <- forecast_posterior(est_results = example_estimation_result,
                                          data_set = example_estimation_data,
                                          shock_uncertainty = TRUE,
                                          variables = c("Y"),
                                          observables = c("Y", "G"),
                                          horizon = 10)

# forecast taking into account parameter uncertainty
post_forecast <- forecast_posterior(est_results = example_estimation_result,
                                    data_set = example_estimation_data,
                                    shock_uncertainty = FALSE,
                                    variables = c("Y"),
                                    observables = c("Y", "G"),
                                    horizon = 10)

# comparison of the forecasts
plot_forecast(post_forecast_shock)
plot_forecast(post_forecast)
```

---

gecon_estimation_results-class

DSGE model estimation results

Description

The `gecon_estimation_results` class stores the results of estimation.

Slots

`chains` a list of matrices. Each matrix represents Markov chain drawn from posterior distribution.

`acceptance_rate` a numeric value indicating acceptance rate for Metropolis-Hastings procedure.

`prior` an object of `gecon_prior` class. This object declares prior distribution for the estimated parameters.

`models` a list of objects of `gecon_model` class. Each of the list elements contains model solved during the last iteration of the random walk Metropolis-Hastings algorithm.

`est_parameters` a character vector of estimated parameter names.

`total_n` a numeric, the total number of draws from posterior distribution.

`opt_result` a numeric vector of parameter values returned from the maximisation routine.

`opt_err` a numeric vector of std. errors for parameter estimates returned from the maximisation routine.

`hessian_pos_def` a logical. If TRUE, the Hessian of optimised function computed at its maximum was positive definite.

`marg_density` a numeric, the Laplace approximation to the marginal density of posterior.

`estimates` a numeric vector of posterior means. Relevant for Bayesian estimation only.

`square_estimates` a numeric vector of expected squared values for each parameter (computed based on posterior sample). Relevant for Bayesian estimation only.

`stddev` a numeric vector of standard deviations for each estimated parameter.

`naive_se` a numeric vector of naive standard errors for each estimated parameter.

`method` a character indicating the method of estimation. Equals to either "Bayes" for Bayesian estimation or "ML" for maximum likelihood estimation.

Methods

show: Prints name of estimation method performed and estimation statistics.

print: Prints name of estimation method, estimation statistics, and the names of estimated parameters.

summary: Prints a summary of an object of `gecon_estimation_results` class consisting of name of estimation method, estimation statistics, and estimates of model parameters.

`gecon_prior` *Constructor for the* `gecon_prior` *class*

Description

The `gecon_prior` function is a constructor of `gecon_prior` class objects.

Usage

```
gecon_prior(prior_list, model)
```

Arguments

`prior_list`	a list of lists with the declaration of prior distribution for DSGE model parameters. Each sub-list should contain the `par` and `type` elements declaring name of parameter and type of the distribution, respectively. The distribution parameters may be supplied either explicitly (`a` and `b`) or by mean and standard deviation of the distribution (`mean` and `sd`). Additionally, bounds for distribution values (`lower_bound` and `upper_bound`) and initial values for computation of the posterior kernel distribution (`initial`) may be specified in each sub-list.
`model`	an object of `gecon_model` class for which the parameter prior distributions are specified.

Details

Prior distributions can be specified either for model free parameters or standard deviations/correlations of shocks. In the current release of the package the following distributions can be specified (L stands for `lower_bound` and U stands for `upper_bound`):

- beta distribution with the parameters $a > 0$, $b > 0$ and density function:

$$f(x) = \Gamma(a+b)/(\Gamma(a)\Gamma(b))(x-L)^{(a-1)}(1-(U-x))^{(b-1)}/(U-L)^{(a+b-1)}$$

  for $L < x < U$ and zero otherwise.

- gamma distribution with the parameters $a > 0$, $b > 0$ and density function:

$$f(x) = 1/(b^a\Gamma(a))(x-L)^{(a-1)}e^{-((x-L)/b)}$$

  for $x > L$ and zero otherwise,

- inverted gamma distribution (1st type - inverted square root of gamma distributed variable) with the parameters $a > 0$, $b > 0$ and density function:

$$f(x) = (\frac{2}{b})^{a/2}\Gamma(\frac{a}{2})2(x-L)^{(-a-1)}e^{-b/2(x-L)^2}$$

  for $x > L$ and zero otherwise,

- inverted gamma distribution (2nd type - inverted gamma distributed variable) with the parameters $a > 0$, $b > 0$:

$$f(x) = (\frac{2}{b})^{a/2}\Gamma(\frac{a}{2})(x-L)^{(-a/2-1)}e^{-b/2(x-L)}$$

  for $x > L$ and zero otherwise,

- normal distribution with the parameters $a \in R$, $b > 0$:

$$f(x) = 1/(\sqrt{2\pi}b)e^{-((x-a)^2/(2b^2))},$$

- uniform distribution with the parameters L, U ($L < U$):

$$f(x) = 1/(U-L)$$

  for $L < x < U$ and 0 otherwise.

Prior distributions are declared by passing a list of named lists. Each list should contain a name of a model parameter for which prior distribution is specified, type of distribution, and its mean and standard deviation or parameters (the constructor calculates parameters of distribution, if the distribution mean and standard deviation are supplied). Priors for standard deviations and correlations

of shocks can be declared using the following syntax: standard deviation: "sd(shock_name)", correlation: "cor(shock_name1, shock_name2)".

Additionaly, custom bounds for the distribution parameters, and initial values for computation of the posteror kernel mode may be supplied. The default bounds are equal to:

- lower_bound = 0, upper_bound = 1 for beta distribution,
- lower_bound = 0 for gamma distribution,
- lower_bound = 0 for inverted gamma distributions.

In the case of the uniform distribution l_bound and u_bound are treated both as bounds and parameters.

For each distribution, the parameter initial can be also provided specifying the initial values for parameter estimation.

Value

The function constructs an object of gecon_prior class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# declare prior distribution
dsge_prior <- gecon_prior(
    prior_list = list(
        list(par = "sd(epsilon_Z)", type = "inv_gamma",
             mean = 0.012, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "sd(epsilon_G)", type = "inv_gamma",
             mean = 0.008, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "omega", type = "normal",
             mean = 1.45, sd = 0.1, lower_bound = 1,
             upper_bound  = 2, initial = 1.5),
        list(par = "phi_G", type = "beta",
             mean = 0.88, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.92),
        list(par = "phi_Z", type = "beta",
             mean = 0.92, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.96)),
    model = dsge_model)

plot_prior(dsge_prior)
```

---

gecon_prior-class *S4 class to represent the prior distributions for model parameters*

Description

The gecon_prior class stores information about prior distribution for estimated parameters of DSGE model.

Details

For detailed description of how to specify prior distributions, see gecon_prior.

Slots

model_par a character vector of parameter names.

parameters_tex a character vector of all parameters' LaTeX names.

dist_type a numeric vector of distribution codes (1 - beta distribution, 2 - gamma distribution, 3 - inverted gamma distribution (of the 1st type), 4 - inverted gamma distribution (of the 2nd type), 5 - normal distribution, 6 - uniform distribution).

parameters a numeric matrix of the prior distribution parameter values. The first column corresponds to the a parameter while the second to the b parameter.

bounds a numeric matrix of the prior distribution bounds. The first column corresponds to the lower bound while the second to the upper bound.

moments a numeric matrix of the prior distribution's statistics. The first column corresponds to the mean while the second to the standard deviations.

initial a numeric vector of initial values of parameters for the estimation.

par_type a numeric vector of length equal to the number of estimated parameters with the following codes:

- 0 denotes free parameter,
- 1 denotes standard deviation of a shock,
- 2 denotes correlation of shocks.

model_info a character vector with information about the model the prior distribution of parameters refers to: the input file name, the input file path, and the date of creation.

shock_names_matrix a matrix of characters; the names of shocks for which standard deviations and correlations have been specified.

Methods

print: Prints number of parameters for which prior distributions have been specified.

show: Prints names of parameters for which prior distributions have been specified.

summary: Prints a summary of an object of gecon_prior class consisting of moments of prior distribution, values of parameters, and their initial values used for the estimation.

---

get_chain *Retrieving the specific MCMC chain from the* gecon_estimation_results *class objects*

Description

The get_chain function allows to retrieve specific chain from the objects of gecon_estimation_results class.

Usage

```
get_chain(est_results, number)
```

Arguments

est_results	an object of gecon_estimation results class.
number	a numeric. The chain number

Value

A matrix of the MCMC chain.

---

get_chains *Retrieving the MCMC chains from the* gecon_estimation_results *class objects*

Description

The get_chains function allows to retrieve chains from the objects of gecon_estimation_results class.

Usage

```
get_chains(est_results)
```

Arguments

est_results	an object of gecon_estimation results class.

Value

The function returns a list of MCMC chains.

---

get_estimated_par *Retrieve model parameters based on estimation results*

Description

The get_estimated_par function allows to retrieve estimated parameters' values from the object of gecon_estimation_results class. For models estimated using Bayesian methods, point estimates are computed as a mean of posterior distribution (default) or as a result of a function specified by user.

Usage

```
get_estimated_par(est_results, est_function = NULL)
```

Arguments

est_results	an object of gecon_estimation results class.
est_function	a list of the names of functions that are to be used to transform the posterior into point estimates.

Value

A list of parameter values. The `free_par` element contains values of estimated model parameters while the `shock_distr_par` element contains values of shock distibution parameters.

Examples

```
# load example estimation results
data(example_estimation_result)
summary(example_estimation_result)

# retrieve point estimates of parameter values
# (computed as a mean and as a median of the posterior distribution)
dsge_estimated_par_mean <- get_estimated_par(est_results = example_estimation_result)

dsge_estimated_par_median <- get_estimated_par(est_results = example_estimation_result,
                                               est_function = list("sd(epsilon_G)" = "median",
                                                                   "sd(epsilon_Z)" = "median",
                                                                         "phi_G" = "median",
                                                                         "phi_Z" = "median",
                                                                         "omega" = "median"))

cbind(dsge_estimated_par_mean$free_par, dsge_estimated_par_median$free_par)
cbind(dsge_estimated_par_mean$shock_distr_par, dsge_estimated_par_median$shock_distr_par)
```

`get_optimisation_statistics` *Likelihood / posterior kernel optimisation statistics*

Description

The `get_optimisation_statistics` function allows to retrieve the results of the optimisation routine. The statistics may refer to posterior kernel optimisation (Bayesian estimation) or likelihood function optimisation (maximum likelihood method).

Usage

```
get_optimisation_statistics(est_results, silent = FALSE)
```

Arguments

`est_results`	an object of `gecon_estimation_results` class.
`silent`	logical. The default value is FALSE. If set to TRUE, the console output is suppressed.

Value

The function returns a list with two elements:

- `mode` a vector of posterior kernel modes or maximum likelihood estimates,
- `std_error` a vector of standard errors for the estimated parameters.

Examples

```
# ##################################################################
# 1. prepare gEcon model

# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# solve the model
dsge_model <- steady_state(dsge_model)
dsge_model <- solve_pert(dsge_model, loglin = TRUE)

# set the stochastic shocks distribution parameters
dsge_model <- set_shock_distr_par(dsge_model,
                                  distr_par = list("sd( epsilon_G )" = 0.01,
                                                   "sd( epsilon_Z )" = 0.01))
shock_info(model = dsge_model, all = TRUE)

# load data
data(example_estimation_data)

# ##################################################################
# 2. maximum likelihood estimation

ml_estimation_result <- ml_estimation(data_set = example_estimation_data,
                                      observables = c("Y", "G"),
                                      model = dsge_model,
                                      est_par = c("sd(epsilon_G)", "sd(epsilon_Z)"),
                                      initial_vals = c("sd(epsilon_G)" = 0.007,
                                                       "sd(epsilon_Z)" =  0.007))

# retrieving estimation statistics
summary(ml_estimation_result)
get_optimisation_statistics(ml_estimation_result)
```

`get_posterior_statistics` *Retrieving the posterior distribution statistics*

Description

The `get_posterior_statistics` function allows to retrieve moments and position statistics from the posterior distribution.

Usage

```
get_posterior_statistics(est_results, quantile = c(0.05, 0.95),
  silent = FALSE)
```

Arguments

`est_results`	an object of `gecon_estimation_results` class.

quantile	a numeric vector of numbers in the [0,1] interval. By this argument, the relevant quantiles of posterior distribution can be requested.
silent	logical. The default value is FALSE. If set to TRUE, the console output is suppressed.

Value

The function returns a list with the following elements:

- means a numeric vector of posterior distribution means for each estimated parameter,
- medians a numeric vector of posterior distribution medians for each estimated parameter,
- std_dev a numeric vector of posterior distribution standard deviations for each estimated parameter,
- quantiles_computed a numeric matrix of posterior distribution quantiles for each estimated parameter.

Examples

```
# load example estimation results
data(example_estimation_result)

# retrieve statistics summarising the posterior distribution
get_posterior_statistics(example_estimation_result,
                         quantile = c(0.1, 0.4, 0.8, 0.9))
```

---

ml_estimation *Maximum likelihood estimation of DSGE models*

Description

The ml_estimation function allows to estimate DSGE model parameters using maximum likelihood method. The vector of parameters maximising the likelihood function is treated as point estimate of parameters. For details about likelihood construction method, see gecon.estimation-package.

Usage

```
ml_estimation(data_set, observables, model = NULL, est_parameters, l_bound,
  u_bound, initial_vals, optim_options_list = NULL)
```

Arguments

data_set	an object of ts or mts class containing data for the estimation of the model. The time series supplied have to be stationary, detrended, and deseasoned. The number of series supplied cannot be greater than number of shocks in the model. Estimation based on data sets with missing observations is not supported in the current version.
observables	a character with the length equal to the number of data_set columns. By this argument the names of observable variables can be specified (each of them must correspond to one of the model variables).
model	an object of gecon_model class.

`est_parameters` a character vector of estimated parameters' names.

`l_bound` a numeric of parameter range lower bounds.

`u_bound` a numeric of parameter range upper bounds.

`initial_vals` a named vector of initial values of estimated parameters.

`optim_options_list`
a list of options for the likelihood maximisation routines:

- `solver` a character; the name of solver:
  - `csminwelNew` - `csminwelNew` solver (updated version of `csminwel`),
  - `csminwel` - `csminwel` solver (default),
  - `Nelder-Mead` - the Nelder-Mead routine implemented in the `optim` function,
  - `GenSA` - a simulated annealing solver from `GenSA` package.
- `solver_crit` a numeric, the convergence criterion for a solver. The default value is 1e-7. Not used by the `GenSA` solver.
- `solver_iter` a numeric, the maximum number of iterations for a solver. The default value is 1000.
- `init_hess_scale` a numeric, size of the diagonal elements of initial Hessian used by `csminwel` and `csminwelNew` procedures. Not used by `Nelder-Mead` and `GenSA` solvers. The default value is 1e-4.
- `temperature` a numeric, the initial temperature for `GenSA` solver. The default value is 5230.
- `visiting.param` a numeric, a visiting parameter for `GenSA` solver. The default value is 2.62.
- `acceptance.param` a numeric, an acceptance parameter for `GenSA` solver. The default value is -5.
- `max.call` a numeric, the maximum number of function calls if `GenSA` solver is used. The default value is 1e+06.
- `solver_hess` a logical value. If set to TRUE, the Hessian returned from the numerical solver is used for computation of variance of sampling distribution. By setting this value to TRUE, one can obtain only approximate Hessian. However, such an approximate Hessian may be more often positive definite (thanks to rank-one updates) than one computed explicitly. The `Nelder-Mead` and `GenSA` solvers do not return Hessian. The default value is FALSE.
- `hessian_d` a numeric, delta used to compute Hessian numerically. The default value is 0.001.
- `hessian_eps` a numeric, tolerance of the routine computing Hessian. The default value is 1e-7.

Examples

```
# ###################################################################
# 1. prepare gEcon model

# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# solve the model
```

```
dsge_model <- steady_state(dsge_model)
dsge_model <- solve_pert(dsge_model, loglin = TRUE)

# set the stochastic shocks distribution parameters
dsge_model <- set_shock_distr_par(dsge_model,
                                  distr_par = list("sd( epsilon_G )" = 0.01,
                                                   "sd( epsilon_Z )" = 0.01))
shock_info(model = dsge_model, all = TRUE)

# load data
data(example_estimation_data)

# ######################################################################
# 2. maximum likelihood estimation

ml_estimation_result <- ml_estimation(data_set = example_estimation_data,
                                      observables = c("Y", "G"),
                                      model = dsge_model,
                                      est_par = c("sd(epsilon_G)", "sd(epsilon_Z)"),
                                      initial_vals = c("sd(epsilon_G)" = 0.007,
                                                       "sd(epsilon_Z)" =  0.007))

# retrieving estimation statistics
summary(ml_estimation_result)
get_optimisation_statistics(ml_estimation_result)
```

`plot_forecast` *Plot DSGE model forecast*

Description

The `plot_forecast` function plots the DSGE model-based forecasts.

Usage

```
plot_forecast(forec_list, conf_band = c(0.5, 0.8, 0.9), to_eps = FALSE)
```

Arguments

`forec_list`	a list returned from either `forecast` or `forecast_posterior` function.
`conf_band`	a numeric vector of values from the [0, 1] interval. The forecast confidence interval width. The default value is c(0.5, 0.8, 0.9), corresponding to 25%-75%, 10%-90%, 5%-95% confidence intervals.
`to_eps`	logical. If TRUE, plot(s) are saved as `.eps` file(s) in the model's subdirectory `/plots` and added to `.results.tex` file.

Examples

```
# load estimated model and data
data(estimated_model)
data(example_estimation_data)

# forecast based on data
```

```
hp_forecast <- forecast(model = estimated_model,
                        data_set = example_estimation_data,
                        variables = c("Y"),
                        observables = c("Y", "G"),
                        horizon = 10)

plot_forecast(hp_forecast, conf_band = c(0.5, 0.6, 0.9))
```

`plot_posterior` *Plot posterior distributions*

Description

The `plot_posterior` function plots posterior distributions for the estimated parameters. The prior distributions are plotted for comparison.

Usage

```
plot_posterior(est_results, kernel = "epanechnikov", bw_adjust = 1.5,
  to_eps = FALSE)
```

Arguments

`est_results`	an object of `gecon_estimation_results` class.
`kernel`	a character specifying the name of smoothing kernel to be used. One of the following kernels can be used: `"gaussian"`, `"rectangular"`, `"triangular"`, `"epanechnikov"`, `"biweight"`, `"cosine"` or `"optcosine"`. The default value is `"epanechnikov"`.
`bw_adjust`	a numeric. The adjustment parameter for bandwith. It allows the user to control degree of smoothness for non-parametric density estimation results.
`to_eps`	logical. If TRUE, plot(s) are saved as `.eps` file(s) in the model's subdirectory `/plots` and added to `.results.tex` file.

Examples

```
# load example estimation results
data(example_estimation_result)

# plot posterior
plot_posterior(example_estimation_result)
```

plot_prior *Plot prior distributions*

Description

The plot_prior function plots prior distributions.

Usage

```
plot_prior(prior, to_eps = FALSE)
```

Arguments

prior	an object of gecon_prior class.
to_eps	logical. If TRUE, plot(s) are saved as .eps file(s) in the model's subdirectory /plots and added to .results.tex file.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon.estimation"),
                           "dsge_model.gcn"), to = getwd())
dsge_model <- make_model("dsge_model.gcn")

# declare prior distribution
dsge_prior <- gecon_prior(
    prior_list = list(
        list(par = "sd(epsilon_Z)", type = "inv_gamma",
             mean = 0.012, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "sd(epsilon_G)", type = "inv_gamma",
             mean = 0.008, sd = 0.3, lower_bound = 0.0001,
             upper_bound  = 0.9, initial = 0.008),
        list(par = "omega", type = "normal",
             mean = 1.45, sd = 0.1, lower_bound = 1,
             upper_bound  = 2, initial = 1.5),
        list(par = "phi_G", type = "beta",
             mean = 0.88, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.92),
        list(par = "phi_Z", type = "beta",
             mean = 0.92, sd = 0.03, lower_bound = 0.5,
             upper_bound  = 0.999, initial = 0.96)),
    model = dsge_model)

plot_prior(dsge_prior)
```

plot_shock_decomposition *Plot shock decomposition*

Description

The plot_shock_decomposition function plots shock decomposition returned by the shock_decomposition function.

Usage

```
plot_shock_decomposition(shock_dec_list, to_eps = FALSE)
```

Arguments

shock_dec_list	a list created by the shock_decomposition function.
to_eps	logical. If TRUE, plot(s) are saved as .eps file(s) in the model's subdirectory /plots and added to .results.tex file.

Examples

```
# load data_set and estimation results
data(estimated_model)
data(example_estimation_data)

# perform a shock decomposition
dsge_shock_decomposition <- shock_decomposition(model = estimated_model,
                                                data_set =  example_estimation_data,
                                                observables = c("Y", "G"),
                                                variables = c("K", "I", "C"))
plot_shock_decomposition(dsge_shock_decomposition)
```

print,gecon_prior-method *These methods print information about objects of the* gecon_estimation_results *and* gecon_prior *classes.*

Description

These methods print information about objects of the gecon_estimation_results and gecon_prior classes.

Usage

```
## S4 method for signature 'gecon_prior'
print(x)

## S4 method for signature 'gecon_estimation_results'
print(x)
```

Arguments

x	an object for which information is to be printed.

Methods

`signature(x = "gecon_estimation_results")`: Prints estimation method name, estimation statistics, and the names of estimated parameters.

`signature(x = "gecon_prior")`: Prints names of parameters for which prior distributions have been specified.

---

shock_decomposition *Historical decomposition of the model variables*

Description

The `shock_decomposition` function computes individual contributions of structural shocks to variability of selected variables.

Usage

```
shock_decomposition(model, data_set, observables, variables = NULL)
```

Arguments

`model`	an object of `gecon_model` class. The solution of this model is used for constructing state-space representation and the smoother formulas.
`data_set`	an object of `ts` class containing observables from the time period for which the decomposition is to be performed.
`observables`	a character with the length equal to the number of `data_set` argument columns. By this argument one has to specify names of observable variables supplied in the `data_set` argument.
`variables`	a character vector of the variables' names for which the decomposition is to be performed. If set to NULL (default), only state variables are decomposed.

Details

The decomposition is calculated as follows:

- the Kalman smoother formulas are applied to parametrised model and data so as to obtain the smoothed estimates of stochastic innovations and initial values of the state variables,
- the model is simulated given path of smoothed shocks,
- the difference between smoothed values of variables and values simulated based on smoothed shocks is computed. It can be attributed to the initial values of observables (the impact of shocks former to the start date of supplied data set).

Value

The function returns a list consisting of two elements:

- shock_dec_list a list of ts objects. Each element of the list corresponds to one variable. The columns correspond to model shocks and impact of the initial values.
- variables_tex a character vector of variables' LaTeX names.
- model_info a character vector containing information about the model for which the decomposition has been computed: the input file name, the input file path, and the date of creation.

Examples

```
# load example data set and estimation results
data(estimated_model)
data(example_estimation_data)

# perform a shock decomposition
dsge_shock_decomposition <- shock_decomposition(model = estimated_model,
                                                data_set =  example_estimation_data,
                                                observables = c("Y", "G"),
                                                variables = c("K", "I", "C"))
plot_shock_decomposition(dsge_shock_decomposition)
```

---

show,gecon_prior-method

These methods print general information about objects of the gecon_estimation_results *and* gecon_prior *classes.*

Description

These methods print general information about objects of the gecon_estimation_results and gecon_prior classes.

Usage

```
## S4 method for signature 'gecon_prior'
show(object)

## S4 method for signature 'gecon_estimation_results'
show(object)
```

Arguments

object	an object for which information is to be printed.

Methods

signature(object = "gecon_estimation_results"): Prints estimation method name and estimation statistics.

signature(object = "gecon_prior"): Prints number of parameters for which prior distributions have been specified.

smoother *Kalman smoother for DSGE models*

Description

The smoother function uses the Kalman smoother formulas to obtain smoothed past values of variables and stochastic innovations.

Usage

```
smoother(model, data_set, observables, variables = NULL)
```

Arguments

model	an object of gecon_model class. The state-space representation for the smoother is created based on the model solution.
data_set	an object of ts class containing time series based on which the smoothing is to be performed.
observables	a character of the length equal to the number of data_set argument columns. By this argument one has to specify names of observable variables supplied in the data_set argument.
variables	a character vector of the model variables' names for which the smoothing is to be performed. If set to NULL (default), only state variables are smoothed.

Details

The detailed description of Kalman filter and smoother can be found in Hamilton, J. D. 1994. *Time series analysis*. Princeton, NJ: Princeton Univ. Press.

Value

The function returns a list with three elements:

- smoothed_var a ts object of smoothed values of the variables of interest,
- smoothed_shock a ts object of smoothed shock values,
- P a matrix of the mean squared errors.

Examples

```
# load data_set and estimation results
data(estimated_model)
data(example_estimation_data)

# perform smoothing
dsge_smoothed_variables <- smoother(model = estimated_model,
                                    data_set =  example_estimation_data,
                                    observables = c("Y", "G"),
                                    variables = c("K", "I", "C"))

# smoothed shock values
dsge_smoothed_variables$smoothed_shock
```

```
# smoothed variable values
dsge_smoothed_variables$smoothed_var
# MSE matrix
dsge_smoothed_variables$MSE
```

---

summary,gecon_prior-method

These methods summarise information about objects of gecon_estimation_results *and* gecon_prior *classes.*

Description

These methods summarise information about objects of gecon_estimation_results and gecon_prior classes.

Usage

```
## S4 method for signature 'gecon_prior'
summary(object)

## S4 method for signature 'gecon_estimation_results'
summary(object)
```

Arguments

object	an object for which the summary is to be printed.

Methods

signature(object = "gecon_estimation_results"): Prints a summary of an object of gecon_estimation_resul class consisting of estimation method name, estimation statistics, and estimates of model parameters.

signature(object = "gecon_prior"): Prints a summary of an object of gecon_prior class consisting of moments of prior distribution, values of parameters, and their initial values used for the estimation.

Index