Package 'gEcon'

February 28, 2015

Type Package

Title General Equilibrium Economic Modelling Language and Solution Framework (gEcon)

Version 0.9.0

Date 2015-02-28

Author Grzegorz Klima, Karol Podemski, Kaja Retkiewicz-Wijtiwiak

Copyright Chancellery of the Prime Minister of the Republic of Poland

Maintainer Grzegorz Klima <gklima@users.sourceforge.net>

Description Package for developing and solving dynamic general equilibrium models

Depends R(>= 3.0.0)

Imports methods, MASS, Matrix, nleqslv, Rcpp(>= 0.11.0)

License_restricts_use yes

License file LICENCE

R topics documented:

`gecon-package` *General Equilibrium Economic Modelling Language and Solution Framework (gEcon)*

Description

Package for developing and solving dynamic (stochastic) and static general equilibrium models.

Details

gEcon is a framework for developing and solving large scale dynamic (stochastic) & static general equilibrium models. It consists of model description language and an interface with a set of solvers in R. It was developed at the Department for Strategic Analyses at the Chancellery of the Prime Minister of the Republic of Poland as a part of a project aiming at construction of large scale DSGE & CGE models of the Polish economy.

Publicly available toolboxes used in RBC/DSGE modelling require users to derive the first order conditions (FOCs) and linearisation equations by pen & paper (e.g. Uhlig's tool-kit) or at least require manual derivation of the FOCs (e.g. Dynare). Derivation of FOCs is also required by GAMS and GEMPACK - probably the two most popular frameworks used in CGE modelling. Owing to the development of an algorithm for automatic derivation of first order conditions and implementation of a comprehensive symbolic library, gEcon allows users to describe their models in terms of optimisation problems of agents. To authors' best knowledge there is no other publicly available framework for writing and solving DSGE & CGE models in this natural way. Writing models in terms of optimisation problems instead of the FOCs is far more natural to an economist, takes off the burden of tedious differentiation, and reduces the risk of making a mistake. gEcon allows users to focus on economic aspects of the model and makes it possible to design large-scale (100+ variables) models. To this end, gEcon provides template mechanism (similar to those found in CGE modelling packages), which allows to declare similar agents (differentiated by parameters only) in a single block. Additionally, gEcon can automatically produce a draft of LaTeX documentation for a model.

The model description language is simple and intuitive. Given optimisation problems, constraints and identities, computer derives the FOCs, steady-state equations, and linearisation matrices automatically. Numerical solvers can be then employed to determine the steady state and approximate equilibrium laws of motion around it.

Author(s)

Grzegorz Klima <gklima@users.sourceforge.net>
Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Maintainer: Grzegorz Klima <gklima@users.sourceforge.net>

References

Cf. gEcon manual distributed with the package.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
```

```
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
rbc <- compute_moments(rbc, ref_var = 'Y')
get_moments(rbc,
            moments = TRUE,
            correlations = TRUE,
            autocorrelations = TRUE,
            var_dec = TRUE)

# compute and plot the IRFs
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
summary(rbc)
```

---

check_bk *Blanchard Kahn conditions and eigenvalues*

Description

The check_bk function checks Blanchard Kahn conditions and prints information about eigenvalues.

Usage

```
check_bk(model)
```

Arguments

model	an object of the gecon_model class.

Details

Eigenvalues are computed when gEcon attempts to solve the perturbation (solver uses the Lapack zgges function to compute eigenvalues). The solve_pert function must be called before the eigenvalues can be retrieved.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

Blanchard, O., Kahn C. M. (1980), "The Solution of Linear Difference Models under Rational Expectations", *Econometrica*

Examples

```
# copy the example to the current working directory, process it and solve for the steady state
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())
rbc <- make_model('rbc.gcn')
rbc <- steady_state(rbc)

# solve in log-linearised form
rbc <- solve_pert(rbc)

# check eigenvalues
check_bk(rbc)
```

`compute_irf` *Compute impulse response functions (IRFs)*

Description

The `compute_irf` function computes the impulse response functions for the specified set of variables and shocks and returns an object of the `gecon_simulation` class.

Usage

```
compute_irf(model, shock_list = NULL, var_list = NULL,
            path_length = 40, cholesky = TRUE)
```

Arguments

`model`	an object of the `gecon_model` class.
`shock_list`	a list of shocks for which the IRFs are to be computed. If this argument is missing, the IRFs are computed for all the shocks in the model. By default, the impulse response functions are created for all the shocks in the model.
`var_list`	a list of variables, for which the impact of shocks has to be computed. By default, the impulse response functions are created for the state variables only.
`path_length`	the number of periods for which the IRFs are to be computed.
`cholesky`	logical. If this option is set to FALSE, the function computes the IRFs based on uncorrelated shocks, otherwise the variance-covariance matrix is orthogonalized by using the Cholesky decomposition and the IRFs are computed by using this matrix.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

The function returns an object of gecon_simulation class. Generic functions such as print and summary provide information about the impulse response functions. The plot_simulation function allows to visualize the IRFs.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

For details, see gecon_simulation-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- compute_irf(rbc, cholesky = TRUE,
                           var_list = c('K_s', 'C', 'Z', 'I', 'Y'))
summary(irf_rbc)
plot_simulation(irf_rbc)
```

---

compute_moments *Computation of correlations*

Description

This function computes the statistics of the model by using spectral and simulation methods.

Usage

```
compute_moments(model, ngrid = (64 * 16), filter = TRUE,
             sim = FALSE, nrun = 1e+05, lambda = 1600,
             ref_var = NULL, n_leadlags = 6)
```

Arguments

model	an object of gecon_model class.
ngrid	density of grid used by the Fourier transform. It has to be a multiplicity of 8.
filter	logical. If TRUE, the statistics are computed for the HP-filtered series, otherwise non-filtered series are used for the statistics computation.
sim	logical. If TRUE, simulation methods are used for correlations computations, otherwise spectral methods are used.
nrun	the number of MC simulation runs.
lambda	the lambda parameter for the HP filter.
ref_var	the name or the number of a variable in relation to which correlations are computed. When not specified, the first variable in variables list is treated as the reference value.
n_leadlags	the number of leads/lags for computing correlation tables.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

An object of gecon_model class representing a model. Generic functions such as print and summary allow to show model elements. The get_moments function returns various statistics of the model (both absolute and relative).

Note

The grid has to be large enough (at least 64 * 8) for spectral methods to converge to simulation means.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

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

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load model
rbc <- make_model('rbc.gcn')

# solve the model
```

```
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
rbc <- compute_moments(rbc, ref_var = 'Y')
get_moments(rbc,
            moments = TRUE,
            correlations=TRUE,
            autocorrelations=TRUE,
            var_dec=TRUE)
```

---

gecon-solution_status *Model solution status*

Description

Functions allowing to check the solution status of gecon_model objects.

Usage

```
ss_solved(model)

re_solved(model)
```

Arguments

model	an object of the gecon_model class.

Value

ss_solved: TRUE, if the steady state (equilibrium) of the model has been found. FALSE otherwise.

re_solved: TRUE, if the perturbation has been solved. FALSE otherwise.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# before the model is solved both return FALSE
ss_solved(rbc)
re_solved(rbc)

# find the steady state
```

```
rbc <- steady_state(rbc)
# solve the model in log-linearised form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)

# after the model has been solved both return TRUE
ss_solved(rbc)
re_solved(rbc)
```

---

gecon_model *Constructor of "gecon_model" class*

Description

The gecon_model function is a constructor of gecon_model class objects.

Usage

```
gecon_model(model_info, index_sets, variables, shocks, parameters, parameters_free,
            parameters_free_val, equations, calibr_equations,
            var_eq_map, shock_eq_map, var_ceq_map, cpar_eq_map,
            cpar_ceq_map, fpar_eq_map, fpar_ceq_map, ss_function,
            calibr_function, ss_calibr_function_jac, pert)
```

Arguments

model_info	a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.
index_sets	a list containing information about index sets. The names of the list components correspond to the set names. Each component contains character vector of the names of the relevant set elements.
variables	a character vector of all the variable names.
shocks	a character vector of all the shock names.
parameters	a character vector of all the parameter names.
parameters_free	a character vector of all the free parameter names.
parameters_free_val	a numeric vector of the values of all the free parameters.
equations	a character vector of model equations.
calibr_equations	a character vector of model calibrating equations.
var_eq_map	a sparse matrix (a Matrix class object) representing the mapping of variables to equations.
shock_eq_map	a sparse matrix (a Matrix class object) representing the mapping of shocks to equations.

`var_ceq_map` a sparse matrix (a `Matrix` class object) representing the mapping of variables to calibrating equations.

`cpar_eq_map` a sparse matrix (a `Matrix` class object) representing the mapping of calibrated parameters to equations.

`cpar_ceq_map` a sparse matrix (a `Matrix` class object) representing the mapping of calibrated parameters to calibrating equations.

`fpar_eq_map` a sparse matrix (a `Matrix` class object) representing the mapping of free parameters to equations.

`fpar_ceq_map` a sparse matrix (a `Matrix` class object) representing the mapping of free parameters to calibrating equations.

`ss_function` a function returning residuals from the steady-state (equilibrium for the static models) equations.

`calibr_function` a function used for the calibration of parameters.

`ss_calibr_function_jac` a function returning a Jacobian of functions returning residuals from the steady-state (equilibrium for the static models) equations.

`pert` a function returning a list with the matrices representing canonical form of the model.

Value

An object of the `gecon_model` class.

Note

The `gecon_model` constructor is used in R files created by `gEcon`.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

---

gecon_model-class *Class* "gecon_model"

Description

The class for storing models.

Objects from the Class

Objects can be created by calling `new("gecon_model", ...)` or preferably using `gecon_model` constructor.

Slots

`model_info`: a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.

`index_sets`: a list containing information about index sets. The names of the list components correspond to the set names. Each component contains character vector of the names of the relevant set elements.

`parameters`: a character vector of all parameter names.

`parameters_free`: a character vector of free parameter names.

`map_free_into_params`: the mapping of free parameters to parameters indices.

`parameters_free_mod_flag`: a logical vector of flags denoting if particular free parameter has been modified with respect to the .gcn file calibration

`parameters_calibr`: a character vector of the names of calibrated parameters.

`parameters_calibr_mod_flag`: a logical vector of the flags denoting if any non-default value has been given for the calibrated parameter.

`map_calibr_into_params`: a numeric vector of the mapping of free parameters to parameters indices.

`variables`: a character vector of the names of variables.

`shocks`: a character vector of the names of shocks.

`equations`: a character vector of the names of equations.

`calibr_equations`: a character vector of the names of calibrating equations.

`var_eq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of variables to equations.

`shock_eq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of shocks to equations.

`var_ceq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of variables to calibrating equations.

`cpar_eq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of calibrated parameters to equations.

`cpar_ceq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of calibrated parameters to calibrating equations.

`fpar_eq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of free parameters to equations.

`fpar_ceq_map`: a sparse matrix (a `Matrix` class object) representing the mapping of free parameters to calibrating equations.

`init_calib_pars_supplied`: logical. It informs if calibrated parameters values have been supplied.

`init_vals_supplied`: logical. It informs if values of variables have been supplied.

`loglin`: logical. It informs if the model has to be log-linearised. The default value is TRUE.

`loglin_var`: logical. Flags are set to TRUE for log-linearised variables.

`re_solved`: logical. It is set to TRUE if the model has been solved. The default value is FALSE.

`corr_computed`: logical. If TRUE, indicates that the correlations and other statistics of variables have been computed. The default value is FALSE.

`is_stochastic`: logical. If TRUE, the model has stochastic shocks.

`is_dynamic`: logical. If TRUE, the model has any lead or lagged variables.

`is_calibrated`: logical. If TRUE, the model takes into account calibrating equations when solving for the steady state for a dynamic model (equilibrium in case of static model).

`parameters_free_init_val`: a vector of free parameters values which have been declared in .gcn file.

`parameters_free_val`: a vector of current free parameter values.

`parameters_calibr_val`: a vector of current values of calibrated parameters.

`params`: a vector of the model parameters.

`steady`: a vector of the steady-state values of variables for dynamic models or equilibrium for static models.

`ss_function`: a function returning the steady state (equilibrium for static models) equations residuals.

`ss_function_jac`: a function computing the Jacobian of steady-state function (equilibrium for static models).

`ss_calibr_function_jac`: a function computing the Jacobian of both steady-state (equilibrium) and calibrating functions.

`calibr_function`: calibrating functions

`init_residual_vector`: a numeric vector of residuals of the steady-state (equilibrium) function computed for intial values and calibrating parameters.

`residual_vector`: a numeric vector of residuals of the steady-state (equilibrium) function computed for the values of variables and calibrating parameters after the nonlinear solver has exited.

`solver_status`: a character string describing the steady-state (equilibrium) solver status.

`ss_solved`: logical. If TRUE, steady state (equilibrium for static models) has been found.

`pert`: functions returning the perturbation of first order (returning a list of matrices).

`eig_vals`: a matrix of system eigenvalues.

`solution`: a list with elements P, Q, R, S storing solution of the model.

`state_var_indices`: a numeric vector containing the indices of state variables.

`solver_exit_info`: a character string containing information about perturbation solver exit information.

`solution_resid`: residuals of checking equations, verifying if the model has been solved.

`active_shocks`: a logical vector of the length equal to the number of shocks. If entry is set to FALSE, the shock is not taken into account while performing stochastic simulations of the model.

`cov_mat`: a variance-covariance matrix of model shocks.

`shock_mat_flag`: logical. Set to TRUE when the user specifies non-default entries in variance-covariance matrix for shocks.

`corr_mat`: a matrix of the model variables correlations.

`autocorr_mat`: a matrix of the model variables autocorrelations.

`corr_variable_mat`: a matrix of correlations of variables with the reference variable lead and lagged values.

`var_position`: a numeric value indicating position of reference variable for the computation of statistics.

`var_dec`: a matrix of variance decomposition of shocks.

`sdev`: a vector of standard deviations of variables.

Methods

print `signature(x = "gecon_model")`: prints information about the model solution status.

show `signature(object = "gecon_model")`: prints short information about the model solution status.

summary `signature(object = "gecon_model")`: prints detailed results of the model.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')
print(rbc)
class(rbc)
```

---

gecon_par_info-class *Class* "gecon_par_info"

Description

The class storing information about parameters chosen by the user.

Objects from the Class

Objects of this class are created when calling the `par_info` function.

Slots

`model_info`: a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.

`model_variable_name`: a character string denoting the name of a variable storing the model for which the information about parameters has been created.

`par_names`: a character vector of parameter names.

`gcn_values`: a numeric vector of the values of free parameters specified in the .gcn file.

`current_values`: a numeric vector of the most recent values of the parameters.

`calibr_flag`: a logical vector of the length equal to the number of the parameters. The TRUE entries denote that a corresponding parameter is a calibrated parameter.

`incid_mat`: a `Matrix` object representing the mapping of parameters to equations and calibrating equations.

Methods

print `signature(x = "gecon_par_info")`: Prints all the available information (short listing, values, type, incidence) about the parameters retrieved from the model when creating a `gecon_par_info-class` object.

show `signature(object = "gecon_par_info")`: Prints information about parameters' types, values, and the incidence matrix.

summary `signature(object = "gecon_par_info")`: Prints all the available information (short listing, values, type, incidence) about the parameters, retrieved from the model when creating a `gecon_par_info-class` object.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`par_info` to create a `gecon_par_info` object. The analogous classes storing the information about shocks and variables are `gecon_shock_info-class` and `gecon_var_info-class`.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# compute the steady state and calibrate alpha
rbc <- steady_state(rbc)
rbc_par_info <- par_info(rbc, all_parameters = TRUE)
print(rbc_par_info)
```

```
summary(rbc_par_info)
show(rbc_par_info)
```

---

gecon_shock_info-class *Class* "gecon_shock_info"

Description

The class storing information about shocks chosen by the user.

Objects from the Class

Objects of this class are created when calling the shock_info function.

Slots

model_info: a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.

model_variable_name: a character string denoting the name of a variable storing the model for which simulations have been performed.

shock_names: a character vector of the shock names.

shock_matrix: a matrix object containing columns of the variance-covariance matrix corresponding to given shocks.

shock_matrix_flag: logical. Set to TRUE when the user specifies non-default entries in a variance-covariance matrix of shocks.

incid_mat: a Matrix object representing the mapping of shocks to equations.

Methods

print signature(x = "gecon_shock_info"): Prints all the available information (the incidence matrix, the variance-covariance matrix) about the shocks.

show signature(object = "gecon_shock_info"): Prints the incidence matrix and the variance-covariance matrix of shocks specified when creating a gecon_shock_info object.

summary signature(object = "gecon_shock_info"): Prints all the available information (the incidence matrix, the variance-covariance matrix) about the shocks.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

shock_info to create a gecon_shock_info object. The analogous classes storing the information about variables and parameters are gecon_var_info-class and gecon_par_info-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
rbc <- compute_moments(rbc,  ref_var = 'Y')

# display info about the shocks
rbc_shock_info <- shock_info(rbc, all_shocks = TRUE)
print(rbc_shock_info)
summary(rbc_shock_info)
show(rbc_shock_info)
```

---

gecon_simulation *Constructor for "gecon_simulation" class object*

Description

This function creates an object of gecon_simulation class.

Usage

```
gecon_simulation(sim, shock_list, var_list, sim_type, time_n, model_info, model_variable_name)
```

Arguments

sim	the array of simulation results (three dimensional when the impulse response functions have been computed for more than one shock).
shock_list	a list of shocks for which the simulations have been computed.
var_list	a list of variables used.
sim_type	a type of simulation.
time_n	the number of periods for which the simulation has been performed.
model_info	a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.
model_variable_name	a character string denoting the name of the model for which the simulation has been performed.

Value

An object of the gecon_simulation class.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

Generic functions such as print and summary can provide information about the simulations. The plot_simulation function allows to visualize the impact on variables.

---

gecon_simulation-class

Class "gecon_simulation"

Description

The class storing simulation results.

Objects from the Class

Objects can be created by calling the new("gecon_simulation", ...) form or (preferably) using gecon_simulation constructor.

Slots

sim: a three-dimensional array with impulse response functions (the dimensions are variables, time, shocks) or two-dimensional array when storing the results of user-specified path of shocks or random path of shocks.

shock_list: a vector of shocks for which simulations have been computed.

var_list: a vector of names of simulated variables.

sim_type: a type of simulation.

time_n: the number of simulation periods.

model_info: a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.

model_variable_name: a character string denoting the name of a variable storing the model for which the simulations have been performed.

Methods

print `signature(x = "gecon_simulation")`: prints diagnostic information about the simulation performed.

show `signature(object = "gecon_simulation")`: prints short information about the simulation.

summary `signature(object = "gecon_simulation")`: prints and returns the simulation results in the form of list.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`get_simulation_results` to retrieve the simulated series from `sim` slot.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- compute_irf(rbc)
summary(irf_rbc)
class(irf_rbc)
```

---

gecon_var_info-class *Class* "gecon_var_info"

Description

The class storing information about variables chosen by the user.

Objects from the Class

Objects of this class are created when calling the `var_info` function.

Slots

`model_info`: a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.

`model_variable_name`: a character string, the name of a variable storing the model for which the simulations have been performed.

`var_names`: a character vector of the variable names.

`is_stochastic`: logical. If TRUE, the model, based on which the info was generated, has stochastic shocks.

`is_dynamic`: logical. If TRUE, the model, based on which the info was generated, has any lead or lagged variables.

`ss_solved`: logical. If TRUE, the steady state (equilibrium for static models) for the model has been found.

`re_solved`: logical. It is set to TRUE if the model, based on which the info was generated, has been solved. The default value is FALSE.

`corr_computed`: logical. If TRUE, it indicates that the correlations and other statistics of variables have been computed. The default value is FALSE.

`ss_val`: a vector of the steady-state values of variables (dynamic models) or equilibrium (static models). If the steady state has not been computed, this slot contains initial values of variables.

`state`: a logical vector of the length equal to the number of the variables. The TRUE entries denote that a corresponding variable is a state variable.

`state_var_impact`: the rows of the matrices P and R of state space representation corresponding to the chosen variables.

`shock_impact`: the rows of the matrices Q and S of state space representation corresponding to the chosen variables.

`std_dev_val`: a numeric vector of standard deviations of chosen variables.

`loglin_flag`: a logical vector of the length equal to the number of the variables. The TRUE entries denote that a corresponding variable has been loglinearised before solving the model.

`cr`: a matrix containing the correlations of the chosen variables with all the model variables.

`incid_mat`: a `Matrix` object representing the mapping of variables to equations and calibrating equations.

Methods

print `signature(x = "gecon_var_info")`: Prints all the available information (short listing, values, statistics, incidence) about the variables, retrieved from the model when creating a `gecon_var_info-class` object.

show `signature(object = "gecon_var_info")`: Prints information about the variables incidence and the results already obtained for the variables.

summary `signature(object = "gecon_var_info")`: Prints all the available information (short listing, values, statistics, incidence) about the variables, retrieved from the model when creating a `gecon_var_info-class` object.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

var_info to create a gecon_var_info object. The analogous classes storing the information about shocks and parameters are gecon_shock_info-class and gecon_par_info-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# compute the steady state
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc_var_info <- var_info(rbc, all_variables = TRUE)
print(rbc_var_info)
summary(rbc_var_info)
show(rbc_var_info)
```

---

get_cov_mat *Accessing a variance-covariance matrix of model shocks.*

Description

The get_cov_mat function returns a variance-covariance matrix of model shocks.

Usage

```
get_cov_mat(model)
```

Arguments

model	an object of gecon_model class.

Value

The function returns a variance-covariance matrix of model shocks.

See Also

For details, see gecon_model-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'home_production.gcn'), to = getwd())

# make and load the model
home_production <- make_model('home_production.gcn')
shock_info(home_production, all_shocks = TRUE)

# set the shock distribution parameters
home_production <- set_shock_distr_par(home_production,
                                       distr_par = list('sd(epsilon_h)' = 0.7,
                                                        'var(epsilon_m)' = 0.49,
                                                        'cor(epsilon_m,
                                                             epsilon_h)' = 2/3))

# retrieve and show the variance-covariance matrix of model shocks
cov_mat <- get_cov_mat(home_production)
cov_mat
```

`get_index_sets` *List of index sets*

Description

The `get_index_sets` function retrieves a list with all the index sets specified in the `.gcn` file.

Usage

```
get_index_sets(model)
```

Arguments

`model`	an object of `gecon_model` class.

Details

Cf. gEcon manual, chapter "Templates".

Value

The function returns a list of index sets. Each component of the list corresponds to one set and contains all the set elements' names as a character vector.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'pure_exchange_t.gcn'), to = getwd())

# make and load the model
pure_exchange_t <- make_model('pure_exchange_t.gcn')

# retrieve the index sets
pure_exchange_ind_sets <- get_index_sets(pure_exchange_t)
```

`get_model_info` *Accessing information about the name and the creation date of the model*

Description

The `get_model_info` function returns a character vector with information about the model.

Usage

```
get_model_info(model)
```

Arguments

`model`	an object of `gecon_model` class.

Value

The function returns a character vector of length 3, containing information about the model: the input file name, the input file path, and the date of creation.

See Also

For details, see `gecon_model-class`.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# retrieve and show model information
model_info <- get_model_info(rbc)
model_info
```

get_moments *Statistics of the model*

Description

The get_moments function prints and returns the statistics of the model (absolute and relative to the reference variable).

Usage

```
get_moments(model, var_names, relative_to = FALSE, moments = TRUE,
            correlations = TRUE, autocorrelations = TRUE, var_dec = TRUE,
            to_tex = FALSE)
```

Arguments

model	an object of the gecon_model class.
var_names	the names of the variables of interest.
relative_to	logical. If TRUE, the function returns moments relative to one of the variables in accordance with relevant options chosen in the compute_moments (then only 'moments' and 'correalations' are active options). The default value is FALSE.
moments	logical. If TRUE, the moments of variables: steady state values, standard deviations and variances are returned with the information about which variables have been log-linearised. If 'relative_to' is set to TRUE then the moments and steady-state values relative to the reference variable are returned.
correlations	logical. If TRUE, a correlation matrix is returned. If relative_to is set to TRUE, then the corraletions of variables with lagged and leading values of a chosen variable are returned. The default value is TRUE.
autocorrelations	logical. If TRUE then the autocorrelations of variables are returned. If the relative_to is set to TRUE, this option is inactive. The default value is TRUE.
var_dec	logical. If TRUE then the variance decomposition (of shocks) is returned. If the relative_to is set to TRUE, the option is inactive. The default value is TRUE.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.

Value

The function returns a list of absolute or relative moments of variables depending on the value of the relative_to argument.

When the relative_to is set to FALSE, the list may consist of the following elements:

- moments - means, standard deviations, and variances of variables,
- correlation_matrix - a matrix of correlation of variables,
- autocorrelations - a matrix of correlation of variables with their own lagged values (autocorrelations),

- `variance_decomposition` - the variance decomposition, describing the amount of variable variability that can be ascribed to each of shocks.

When the `relative_to` is set to TRUE, the list may consist of two elements:

- `relative_moments` - means, standard deviations, and variance of variables with respect to reference variable specified in the `compute_moments` function,
- `correlations_variable` - a matrix of correlation of variables with lead and lagged values of a reference variable (usually GDP).

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

the `compute_moments` function to see how the statistics are computed.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')

# compute and retrieve correlations
rbc <- compute_moments(rbc,  ref_var = 'Y')
get_moments(model = rbc, relative_to = FALSE, moments = TRUE, correlations = TRUE,
            autocorrelations = TRUE, var_dec = TRUE)
get_moments(model = rbc, relative_to = TRUE, moments = TRUE, correlations = TRUE)
```

`get_par_names` *Accessing parameter names used by* `gecon_model` *class objects*

Description

The `get_par_names` function allows to retrieve the names of parameters from `gecon_model` class objects.

Usage

```
get_par_names(model, free_par = TRUE, calibr_par = TRUE)
```

Arguments

model	an object of gecon_model class.
free_par	logical. If TRUE (default), free parameters are added to the vector of parameter names.
calibr_par	logical. If TRUE (default), calibrated parameters are added to the vector of parameter names.

Value

The function returns a character vector of parameter names, stored by the chosen object of gecon_model class.

See Also

For details, see gecon_model-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load model
rbc <- make_model('rbc.gcn')

# get names of all parameters
get_par_names(rbc)

# get names of the free parameters
get_par_names(rbc, calibr_par = FALSE)

# get names of the calibrated parameters
get_par_names(rbc, free_par = FALSE)
```

---

get_par_names_by_index

Parameters corresponding to given indices

Description

The get_par_names_by_index function retrieves the names of parameters with given indices.

Usage

```
get_par_names_by_index(model, index_names)
```

Arguments

model	an object of gecon_model class.
index_names	a character vector of the chosen indices.

Details

Cf. gEcon manual, chapter "Templates".

Value

The function returns a character vector of relevant parameter names.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'pure_exchange_t.gcn'), to = getwd())

# make and load the model
pure_exchange_t <- make_model('pure_exchange_t.gcn')

# model calibration
pure_exchange_t <- set_free_par(pure_exchange_t, free_par= c("alpha__A__1" = 0.3, "alpha__A__2" = 0.4,
                                                  "alpha__A__3" = 0.3, "alpha__B__1" = 0.3,
                                                  "alpha__B__2" = 0.4, "alpha__B__3" = 0.3,
                                                "e_calibr__A__1" = 3, "e_calibr__B__1" = 1,
                                                "e_calibr__A__2" = 2, "e_calibr__B__2" = 1,
                                                "e_calibr__A__3" = 1, "e_calibr__B__3" = 3))

# get all parameters associated with agent A
par_names_A <- get_par_names_by_index(pure_exchange_t, index_names = "A")
par_info(pure_exchange_t, par_names_A)

# get all parameters associated with agent B
par_names_B <- get_par_names_by_index(pure_exchange_t, index_names = "B")
par_info(pure_exchange_t, par_names_B)
```

---

get_par_values *Parameters of the model*

Description

The get_par_values function prints and returns the values of parameters.

Usage

```
get_par_values(model, par_names, to_tex)
```

Arguments

model	an object of the gecon_model class.
par_names	a list of requested parameters names.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.

Value

This function returns both free and calibrated parameter values.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

For details, see gecon_model-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# find the steady state
rbc <- steady_state(rbc)

# get parameter values
get_par_values(rbc)
```

---

get_pert_solution *Recursive laws of motion for the model variables*

Description

The get_pert_solution function prints and returns the recursive laws of motion for the model's variables.

Usage

```
# getting recursive laws of motion
get_pert_solution(model, to_tex = FALSE, silent = FALSE)
```

Arguments

model	an object of the gecon_model class.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.
silent	logical. If TRUE, console output is suppressed. The default value is FALSE.

Value

A list with P, Q, R, S elements. P and Q matrices denote the impact of lagged state variables and current values of shocks variables on current values of state variables. R and S matrices denote the impact of lagged state variables and current values of shocks variables on current values of non-state variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

solve_pert for the description of solution procedure and description of output.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# find the steady state
rbc <- steady_state(rbc)

# solve the model in log-linearised form and get the results
rbc <- solve_pert(rbc)
get_pert_solution(rbc)
```

---

get_residuals *Retriving residuals*

Description

The get_residuals function allows to check the residuals of the steady-state equations (equations characterising equilibrium in case of static models) and identify equations with the highest errors. This information may help to assign better initial values to variables when the solver cannot find the steady state (equilibrium).

Usage

```
get_residuals(model, highest = 5)
```

Arguments

model	an object of the gecon_model class.
highest	the number of equations with the highest error to be printed.

Value

This function returns a list with the initial and final elements. The initial residuals are residuals computed using the initial values. The final residuals are residuals computed after the solver has exited. The function prints the indices of equations with the highest initial and final errors. The equations can be investigated by using the list_eq function.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
                           'home_prod_templ.gcn'), to = getwd())

# make and load the model
home_prod_templ <- make_model('home_prod_templ.gcn')

# for the purpose of the example, we set the initial values extremely far from the solution
home_prod_templ <- initval_var(home_prod_templ, c(N = 0.02,
                                                  N__H = 0.01,
                                                  N__M = 0.01))
home_prod_templ <- steady_state(home_prod_templ)
get_residuals(home_prod_templ)

# after setting more reasonable values the steady state is found
home_prod_templ <- initval_var(home_prod_templ, c(N = 0.5,
                                                  N__H = 0.25,
                                                  N__M = 0.25))
home_prod_templ <- steady_state(home_prod_templ)
get_residuals(home_prod_templ)
```

get_shock_names *Accessing shock names used by* gecon_model *class objects*

Description

The get_shock_names function allows to retrieve the names of shocks from gecon_model class objects.

Usage

```
get_shock_names(model)
```

Arguments

model	an object of gecon_model class.

Value

The function returns a character vector of shock names, stored by the chosen object of gecon_model class.

See Also

For details, see gecon_model-class.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# get shock names
get_shock_names(rbc)
```

get_shock_names_by_index *Shocks corresponding to given indices*

Description

The get_shock_names_by_index function retrieves the names of shocks with given indices.

Usage

```
get_shock_names_by_index(model, index_names)
```

Arguments

model	an object of gecon_model class.
index_names	a character vector of the chosen indices.

Details

Cf. gEcon manual, chapter "Templates".

Value

The function returns a character vector of relevant shock names.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'home_prod_templ.gcn'), to = getwd())

# make and load the model
home_prod_templ <- make_model('home_prod_templ.gcn')

# get shocks affecting home production technology
hp_shocks <- get_shock_names_by_index(home_prod_templ, 'H')

# print information obout the selected shocks
shock_info(home_prod_templ, hp_shocks)
```

---

get_simulation_results

Retrieve series of simulated variables

Description

The get_simulation_results function retrieves the series of simulated variables from an object of the gecon_simulation class.

Usage

```
get_simulation_results(sim_obj)
```

Arguments

sim_obj	An object of the gecon_simulation-class class.

Value

The results are returned as one element list when the simulation has been invoked by the `random_path` or `simulate_model` functions or a list of more elements corresponding to the number of shocks when the simulation has been performed with the `compute_irf` function.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

For details, see `gecon_simulation-class`.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- compute_irf(rbc, cholesky = TRUE,
                       var_list = c('K_s', 'C', 'Z', 'I', 'Y'))
get_simulation_results(irf_rbc)
```

---

get_ss_values *Return the steady-state (equilibrium) values*

Description

The `get_ss_values` function returns (and prints) the steady state of the model for dynamic models (equilibrium for static models).

Usage

```
get_ss_values(model, var_names = NULL, to_tex = FALSE, silent = FALSE)
```

Arguments

model	an object of the gecon_model class.
var_names	the names or the indices of the variables, whose steady-state values (equilibrium values) are to be returned. The default option is a vector containing all the variable names.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.
silent	logical. If TRUE, console output is suppressed. The default value is FALSE.

Value

A numeric vector of the steady-state (equilibrium for static models) values.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# find and print the steady state values
rbc <- steady_state(rbc)
get_ss_values(rbc)
get_ss_values(rbc, var_names=c('K_s', 'L_s'))
```

---

get_var_names *Accessing variable names used by* gecon_model *class objects*

Description

The get_var_names function allows to retrieve the names of variables from gecon_model class objects.

Usage

```
get_var_names(model)
```

Arguments

model	an object of gecon_model class.

Value

The function returns a character vector of variable names, stored by the chosen object of `gecon_model` class.

See Also

For details, see `gecon_model-class`.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# get variable names
get_var_names(rbc)
```

`get_var_names_by_index`

Variables corresponding to given indices

Description

The `get_var_names_by_index` function retrieves the names of variables with given indices.

Usage

```
get_var_names_by_index(model, index_names)
```

Arguments

`model`	an object of `gecon_model` class.
`index_names`	a character vector of the chosen indices.

Details

Cf. gEcon manual, chapter "Templates".

Value

The function returns a character vector of relevant variable names.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'pure_exchange_t.gcn'), to = getwd())

# make and load the model
pure_exchange_t <- make_model('pure_exchange_t.gcn')

# model calibration
pure_exchange_t <- set_free_par(pure_exchange_t,
                                free_par = c("alpha__A__1" = 0.3, "alpha__A__2" = 0.4,
                                             "alpha__A__3" = 0.3, "alpha__B__1" = 0.3,
                                             "alpha__B__2" = 0.4, "alpha__B__3" = 0.3,
                                             "e_calibr__A__1" = 3, "e_calibr__B__1" = 1,
                                             "e_calibr__A__2" = 2, "e_calibr__B__2" = 1,
                                            "e_calibr__A__3" = 1, "e_calibr__B__3" = 3))

# find the equilibrium
pure_exchange_t <- steady_state(pure_exchange_t)

# get all variable names associated with agent A
var_names_A <- get_var_names_by_index(pure_exchange_t, index_names = "A")


# get all variable names associated with agent B
var_names_B <- get_var_names_by_index(pure_exchange_t, index_names = "B")

# compare equilibrium allocations
get_ss_values(pure_exchange_t, var_names_A)
get_ss_values(pure_exchange_t, var_names_B)
```

---

initval_calibr_par *Setting initial values of calibrated parameters*

Description

The initval_calibr_par function enables setting the initial values of calibrated parameters for the nonlinear solver searching for the steady state of dynamic models (equilibrium for static models) and the values of calibrated parameters. If not set by this function, the default values of parameters are assumed to be 0.5.

Usage

```
initval_calibr_par(model, calibr_par)
```

Arguments

model	an object of the gecon_model class.
calibr_par	a named list or a vector of parameters.

Details

The values of parameters passed to the gecon_model are treated as initial values for the steady-state solver when the user specifies calibrating equations in a .gcn file and requests that steady_state function shall use it. If the calibration is omitted, the initial values of calibrated parameters are treated as their final values, so one has to specify the right set of calibrated parameters values when decides to omit the calibrating equations.

Value

An object of the gecon_model class representing the model. Generic functions such as print and summary allow to show the model's elements. The get_par_values function return parameter values.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# set initial values for the calibrated parameters
rbc <- initval_calibr_par(rbc, calibr_par=list(alpha = 0.4))

# find the steady state and values of the calibrated parameters
rbc <- steady_state(rbc)
get_par_values(rbc, c('alpha'))
```

---

initval_var *Setting initial values of variables.*

Description

The initval_var function sets the initial values of the model's variables to values specified by the user. The initial values close to solution will help the nonlinear equations solver to find the solution.

Usage

```
initval_var(model, init_var)
```

Arguments

model	an object of the gecon_model class.
init_var	a named list or vector of the initial values of variables.

Value

An object of the gecon_model class representing the model. Generic functions such as print and summary allow to show model elements. The get_ss_values function returns the steady-state (equilibrium) values of the model variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# set initial values of variables
rbc <- initval_var(rbc, list(K_s = 10, C = 2, I = 0.5, Y = 2.5))
rbc <- initval_var(rbc, c(pi=0))
```

---

list_calibr_eq *Find calibrating equations*

Description

The list_calibr_eq function returns calibrating equations with given indices.

Usage

```
list_calibr_eq(model, no_eq = NULL)
```

Arguments

model	an object of the gecon model class.
no_eq	a numeric variable, specifies the indices of requested equations.

Value

A character vector of requested equations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# list calibrating equations
list_calibr_eq(rbc, c(1))
```

---

list_eq *Find model equations*

Description

The `list_eq` function returns equations with the specified indices.

Usage

```
list_eq(model, no_eq = NULL)
```

Arguments

`model`	an object of the `gecon_model` class.
`no_eq`	a numeric variable, specifies the indices of requested equations.

Value

A character vector of requested equations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# get the 3rd and the 5th model equation
list_eq(rbc, c(3, 5))
```

`load_model` *Load model from .R file*

Description

The `load_model` function loads the already generated .R file with the model and creates an object of the `gecon_model` class.

Usage

```
load_model(model_file)
```

Arguments

`model_file`	the name of the .R file containing the model's functions and variables. It can be a name of file or a name of file ending with a .model.R extension.

Details

The .R file with the model specification has to be created first. It can be done by using the `make_model` command and the `gcn` file model specification or manually.

Value

An object of the `gecon_model` class representing the model. Generic functions such as `print` and `summary` allow to show the model's elements.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

the `make_model` function in order to create an R file with the model elements based on the model specification.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')
print(rbc)
# load the already generated model
rbc2 <- load_model('rbc.model.R')
print(rbc2)
```

`make_model` *Make model from .gcn file*

Description

This function calls the dynamic library, parses the .gcn model file, generates an .R file, and loads it into a `gecon_model` class object.

Usage

```
make_model(model_file)
```

Arguments

`model_file`	the name of the .gcn file containing model formulation. It must be ended with a `.gcn` extension.

Details

Cf. gEcon manual, chapters "Model description language" and "Derivation of First Order Conditions".

Value

An object of the `gecon_model` class representing the model. Generic functions such as `print` and `summary` allow to show the model elements.

Note

When the function is called, an R file with the same name as the .gcn file is created in the the .gcn file directory. Additional files such as a Latex documentation files or a logfile may be created when relevant options are set in the .gcn file.

Author(s)

Grzegorz Klima <gklima@users.sourceforge.net>
Karol Podemski <karol.podemski@gmail.com>
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

load_model function to load already created .R file with model.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')
print(rbc)
```

---

par_info *Information about parameters*

Description

The par_info function allows to create an object of class gecon_par_info, containing information about given parameters of the chosen model. It allows to check type and value of a given set of parameters, and the incidence matrix.

Usage

```
par_info(model, par_names = NULL, all_parameters = FALSE)
```

Arguments

model	an object of the gecon_model class.
par_names	the names of the parameters of interest.
all_parameters	the logical value. If set to TRUE, the par_names argument is overwritten with a vector of all parameters appearing in the model. The default value is FALSE.

Details

If the function result is not assigned to any variable, the information about the requested parameters is printed in the console.

Value

An object of gecon_par_info class.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

shock_info for information about the shocks and var_info for information about the variables.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# find the steady state and the values of the calibrated parameters
rbc <- steady_state(rbc)

# get information about all parameters
par_info(rbc, all_parameters = TRUE)
```

---

plot_simulation *Plot a gecon_simulation object*

Description

The plot_simulation function plots the simulations or saves them as .eps files in the model's subdirectory /plots.

Usage

```
plot_simulation(sim_obj, to_tex = NULL, to_eps = NULL)
```

Arguments

sim_obj	an object of the gecon_simulation class.
to_tex	logical. If TRUE, the plots are added to a .tex file.
to_eps	logical. if TRUE, plot(s) are saved as .eps file(s) in the model's subdirectory /plots.

Value

If the number of variables of interest is greater then five, more then one plots for each impulse are created (max. 5 variables on each plot). Separate plots are created for all the impulses, if the compute_irf function has been used for generating simulations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and plot the IRFs
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
```

---

print-methods *Print methods for classes in the gEcon package*

Description

Prints information about objects of the gecon_simulation, gecon_model, gecon_var_info, gecon_shock_info, and gecon_par_info classes.

Methods

signature(x = "gecon_simulation") Prints the name of the model for which the simulations have been generated, information about the time span, shock, and variables used.

signature(x = "gecon_model") Shows the type of the model, the date of creation, the solving status, and more detailed information about the number of variables and parameters then the show generic.

signature(x = "gecon_var_info") Prints all the available information (a short listing, the incidence matrix, the statistics) about the variables, retrieved from the model when creating a gecon_var_info-class object.

signature(x = "gecon_shock_info") Prints all the available information (a short listing, the incidence matrix , the variance-covariance matrix) about the shocks, retrieved from the model when creating a gecon_shock_info-class object.

signature(x = "gecon_par_info") Prints all the available information (a short listing, the values, the type, and the incidence) about the parameters, retrieved from the model when creating a gecon_par_info-class object.

`random_path` *Simulation of the model using a random path of shocks*

Description

This function draws random shocks from distribution with user specified covariance matrix and then simulates the behaviour of the system.

Usage

```
random_path(model, shock_list = NULL, var_list = NULL, path_length = 100)
```

Arguments

`model`	an object of the `gecon_model` class.
`shock_list`	a list of shock names that should be taken into account. If not specified, the system of all the shocks is simulated.
`var_list`	a list of variables on which the impact of shocks is to be computed. By default, the impact of random path is evaluated for the state variables only.
`path_length`	the length of stochastic path, default value = 100.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

An object of the `gecon_simulation` class with simulated paths of variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

the `simulate_model` function enables user to specify her own path of shocks and simulate the impact. The function returns an object of the `gecon_simulation` class. Generic functions such as `print` and `summary` provide information about the simulations. The `plot_simulation` function allows to visualize the impact on variables.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# perform simulation and plot the results
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- random_path(rbc, path_length = 100,
                           var_list = c('K_s', 'C', 'Z', 'I', 'Y'))
plot_simulation(irf_rbc)
```

`set_free_par` *Setting free parameters of model*

Description

The `set_free_par` function specifies parameters of a `gecon_model` class object.

Usage

```
set_free_par(model, free_par, reset = FALSE)
```

Arguments

`model`	an object of class `gecon_model`.
`free_par`	a named list or a vector of parameters.
`reset`	logical value. If TRUE, the function allows to reset free parameters to values specified in the .gcn file.

Value

An object of the `gecon_model` class representing the model. If the `reset` option has been set to TRUE, the model's parameters will be set back to values from the .gcn file. Generic functions such as `print` and `summary` allow to show model elements. The `get_par_values` function returns parameter values currently in use.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# set different free parameters values from those in the .gcn file
rbc <- set_free_par(rbc, free_par = list(beta = 0.98, delta = 0.01))
rbc <- steady_state(rbc, options = list(method = 'Broyden',
                                        global = 'gline'))
get_ss_values(rbc)

# reset values to .gcn file values
rbc <- set_free_par(rbc, reset = TRUE)
rbc <- steady_state(rbc)
get_ss_values(rbc)
```

`set_shock_cov_mat` *Setting a variance-covariance matrix of stochastic shocks.*

Description

The `set_shock_cov_mat` function allows to set a variance-covariance matrix for the model shocks.

Usage

```
set_shock_cov_mat(model, shock_matrix, shock_order = NULL)
```

Arguments

`model`	an object of the `gecon_model` class.
`shock_matrix`	a symmetric, positive definite matrix with the dimensions (n * n), where n is the number of shocks in the model.
`shock_order`	a character vector specifying the order of shocks in the shock_matrix. If not specified, it is assumed that the order is in accordance with the internal order of the model. The default order can be displayed by using the `shock_info` function with the `all_shocks` argument set to TRUE.

Details

The rows and columns of shock matrix must agree with the order of shocks stored in a `gecon_model-class` object if the `shock_order` argument is not supplied. This order can be checked by using the `shock_info` function and the generic function `print`.

Value

An object of the gecon_model class, which is representing the model. Generic functions such as print and summary allow to show the model elements. The shock_info function returns names of shocks, information about which equations they appear in and the current variance-covariance matrix.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')
shock_info(rbc, all_shocks = TRUE)

# set the shock covariance matrix
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
shock_info(rbc, all_shocks = TRUE)
```

---

set_shock_distr_par *Setting distribution parameters of model shocks*

Description

The set_shock_distr_par function assigns distribution parameters (standard deviations, correlations of shocks etc) to shocks in an object of gecon_model class.

Usage

```
set_shock_distr_par(model, distr_par = NULL)
```

Arguments

model	an object of gecon_model class.
distr_par	a list or vector of distribution parameters.

Details

By default, gEcon uses an identity matrix as the variance-covariance matrix for shocks. Valid parameter names should match any of the following patterns:

"sd(SHOCK_NAME)"

"var(SHOCK_NAME)"

"cov(SHOCK_NAME_1, SHOCK_NAME_2)"

"cor(SHOCK_NAME_1, SHOCK_NAME_2)"

There are two issues which the user should be careful about while using the set_shock_distr_par function. First, in contrast to other parameters, shock distribution parameters require quotation marks to be assigned properly. If quotation marks are omitted, R parser treats elements of the distr_par list or vector as functions and attempts to evaluate them, producing errors. Second, parameters passed to the distr_par argument should not be specified twice.

Value

An object of the gecon_model class representing the model.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'home_production.gcn'), to = getwd())

# make and load the model
home_production <- make_model('home_production.gcn')
shock_info(home_production, all_shocks = TRUE)

# set the shock distribution parameters
home_production <- set_shock_distr_par(home_production,
                                       distr_par = list('sd(epsilon_h)' = 0.7,
                                                        'var(epsilon_m)' = 0.49,
                                                        'cor(epsilon_m,
                                                             epsilon_h)' = 2/3))
# get information about shocks in the model
shock_info(home_production, all_shocks = TRUE)
```

shock_info *Information about shocks*

Description

The shock_info function allows to create an object of the gecon_shock_info class, which contains the information about the model's shocks (occurence in equations and the variance-covariance matrix).

Usage

```
shock_info(model, shock_names = NULL, all_shocks = FALSE)
```

Arguments

model	an object of the gecon_model class.
shock_names	the names of shocks of interest.
all_shocks	the logical value. If set to TRUE, the shock_names argument is overwritten with the vector of all shocks appearing in the model. The default value is FALSE.

Value

An object of the gecon_shock_info-class class.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load model
rbc <- make_model('rbc.gcn')

# set the shock covariance matrix
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')

# get information about shocks
shock_info(rbc, all_shocks = TRUE)
```

show-methods *Show methods for classes in the gEcon package*

Description

This method shows objects of the gecon_simulation, gecon_model, gecon_var_info, gecon_shock_info, and gecon_par_info classes.

Methods

signature(object = "gecon_simulation") Shows the name of the model for which the simulations have been created, information about time span, shocks, and variables used.

signature(object = "gecon_model") Shows the type of the model, the date of creation, the solving status, and the information about number of variables and parameters.

signature(object = "gecon_var_info") Prints information about variables' incidence and the results that have been already obtained for the variables.

signature(object = "gecon_shock_info") Prints the incidence matrix and the variance-covariance matrix of shocks, retrieved from the model when creating gecon_shock_info object.

signature(object = "gecon_par_info") Prints information about parameters' type, value, and the incidence matrix.

simulate_model *Simulation of the model*

Description

The simulate_model function simulates model based on realisations of shock values given by the user. In particular it enables to compute the impact of negative shocks.

Usage

```
simulate_model(model, shock_list = NULL, var_list = NULL, shock_m = NULL,
               periods = NULL, path_length = 40,
               sim_type = NULL, model_name = NULL)
```

Arguments

model	an object of the gecon_model class.
shock_list	the shock names for the rows shock_m specified by the user. The default names are the names of the first shocks from the list of shocks up to the number of shock_m matrix rows.
var_list	the list of variables for which the impact has to be computed. By default, the impact of shocks is evaluated for the state variables only.

shock_m	a matrix or vector of shocks given by the user. Values for different shocks should be stored in rows and values for periods in columns.
periods	the number of periods for which, shocks in the shock_m function have been specified. The default values are from 1 to the number of columns of the shock matrix.
path_length	the number of periods for which the model is simulated. The default number is 40.
sim_type	the type of simulation performed on model. It does not have to be specified when user invokes this function directly.
model_name	the name of the gecon_model-class object based on which simulations are created. The user does not have to specify the name explicitly (by default, the variable is deparsed and name is retrieved automatically). It does not have to be specified when user invokes this function directly.

Details

The random_path and compute_irf functions are wrappers for this function. They generate a path of shock(s) values and pass it on to simulate_model function, which performs computations and returns relevant results.

Value

An object of the gecon_simulation class with simulated paths of variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

the random_path function to use random path of shocks for the simulation of the model. The function returns an object of the gecon_simulation class. Generic functions such as print and summary provide information about the simulations. The plot_simulation function allows to visualize the impact on variables.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
```

```
# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, shock_matrix = matrix(0.01, 1, 1), shock_order = 'epsilon_Z')
irf_rbc <- simulate_model(rbc, shock_m = matrix(c(-0.05, -0.05), 1, 2),
                               periods=c(1, 4), var_list = c('K_s', 'C', 'Z', 'I', 'Y'))
plot_simulation(irf_rbc)
```

`solve_pert` *Solve the model in a linearised form (1st order perturbation)*

Description

This function solves the model in a linearised form using Christopher Sims' `gensys` solver.

Usage

```
solve_pert(model, loglin = TRUE, not_loglin_var = NULL,
           norm_tol = 1e-08, solver = "sims_solver")
```

Arguments

`model`	an object of the `gecon_model` class.
`loglin`	an option to log-linearise the perturbation. If FALSE, the model is only linearised.
`not_loglin_var`	a vector of variables that will not be log-linearised.
`norm_tol`	the tolerance for residuals of model (default 1e-08).
`solver`	the name linear RE solver. The default solver is Christopher Sims' solver. Currently no other solvers are available.

Details

Cf. gEcon manual, chapter "Solving the model in linearised form".

Value

an object of the `gecon_model` class representing the model. Generic functions such as `print` and `summary` allow to show the model elements. The `get_pert_solution` function returns computed recursive laws of motion for the model's variables. The `check_bk` function displays the eigenvalues of the system and checks the Blanchard-Kahn conditions.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

Sims, Ch. A. (2002), "Solving Linear Rational Expectations Models.", *Computational Economics*

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())
# make and load the model
rbc <- make_model('rbc.gcn')
# find the steady state
rbc <- steady_state(rbc)

# solve in log-linearised form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)

# solve in linearised form
rbc <- solve_pert(rbc, loglin = FALSE)
get_pert_solution(rbc)

# solve with all variables except L_s log-linearised
rbc <- solve_pert(rbc, not_loglin_var = c('L_s'))
get_pert_solution(rbc)
```

`steady_state` *Compute the steady state (equilibrium) of the dynamic (static) model*

Description

The `steady_state` function solves for the steady state of a dynamic model (equilibrium for static model) and calibrates the chosen parameters using a set of solvers from the `nleqslv` package.

Usage

```
steady_state(model, solver = "slv1_nleqslv", use_jac = TRUE,
             calibration = TRUE, options_list = NULL, solver_status = FALSE)
```

Arguments

`model`	an object of the gecon_model class.
`solver`	the name of nonlinear equations solver. In the current version only an interface to `slv1_nleqslv` function has been implemented.
`use_jac`	the option to use the Jacobian generated by the symbolic library. If FALSE, numerical derivatives are computed.
`calibration`	if FALSE, calibrating equations will not be taken into account in the computation of the steady state (equilibrium in case of static model). The initial values of calibrated parameters will be then treated as their values.
`options_list`	a list containing one or more of the following fields: • `method` a character, can be set to "Newton" or "Broyden", the default option is "Newton".

- `global` a character, search strategy can be set to "dbldog", "pwldog", "qline", "gline", "none". The default option is "qline".
- `xscalm` a character, a method of scaling x. It can be set to "fixed", "auto". The default option is "fixed".
- `max_iter` a numeric value denoting max. number of iterations. The default value is 150.
- `tol` a numeric value setting the numeric tolerance for a solution (function value tolerance). The default value is 1e-6.
- `xtol` a numeric value setting the numeric tolerance for a solution (iteration relative step length tolerance). The default value is 1e-6.

`solver_status` the information about the solver exit code.

Details

Cf. gEcon Manual, chapter "Deterministic steady state & calibration".

Value

An object of the `gecon_model` class representing the model. Generic functions such as `print` and `summary` allow to show model elements. The `get_ss_values` and `get_par_values` functions return the steady state (equilibrium) and parameter values respectively.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`nleqslv` for the detailed description of the `nleqslv` solver capabilities. If the steady state has not been found, the `get_residuals` function can be used to check initial and final residuals.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# find the steady state and calibrate alpha
rbc <- initval_calibr_par(rbc, list(alpha = 0.33))
rbc <- steady_state(rbc, use_jac=TRUE,
                        options_list=list(method='Broyden', global='gline',
                                          max_iter = 300, tol = 1e-7))
get_ss_values(rbc)

# find the steady state without calibrating alpha
rbc <- initval_calibr_par(rbc, list(alpha = 0.4))
```

```
rbc <- steady_state(rbc, calibration = FALSE, use_jac = FALSE,
                    options_list = list(method = 'Newton', global = 'gline',
                                        max_iter = 100, tol = 1e-5))
get_ss_values(rbc)
```

---

summary-methods *Summary methods for classes in the gEcon package*

Description

This method summarizes the objects of the gecon_simulation, gecon_model, gecon_var_info, gecon_shock_info, and gecon_par_info classes.

Methods

signature(object = "gecon_simulation") Prints a summary of a gecon_simulation class object consisting of a shock matrix and the simulation for each shock.

signature(object = "gecon_model") Prints a summary of a gecon_model class object consisting of all the computed statistics and values.

signature(object = "gecon_var_info") Prints all the available information (short listing, the incidence matrix, the statistics) about the variables, retrieved from the model when creating a gecon_var_info-class object.

signature(object = "gecon_shock_info") Prints all the available information (short listing, the incidence matrix , the variance-covariance matrix) about the shocks, retrieved from the model when creating a gecon_shock_info-class object.

signature(object = "gecon_par_info") Prints all the available information (short listing, the values, the type, and the incidence) about the parameters, retrieved from the model when creating a gecon_par_info-class object.

---

var_info *Information about variables*

Description

The var_info function allows to create an object of gecon_var_info class, which contains information about the chosen variables. It allows to check which equations given variables appear in. In addition, this function prints the already computed statistics of the given set of variables.

Usage

```
var_info(model, var_names = NULL, all_variables = FALSE)
```

Arguments

model	an object of the gecon_model class.
var_names	the names of the variables of interest.
all_variables	the logical value. If set to TRUE, the var_names argument is overwritten with a vector of all variables appearing in the model. The default value is FALSE.

Details

The var_info function may be useful in debugging model and quick retrieval of information when the model is large. If the object returned by the function is not assigned to any variable, the information about the requested parameters is printed to the console. One or more of the following elements may be printed:

- incidence information,
- steady-state (equilibrium) values,
- variables info (which ones are log-linearised and which are state variables),
- state variables impact on the chosen variables,
- shocks impact on the chosen variables,
- moments,
- correlations,

depending on which operations have been performed on gecon_model class object.

Value

An object of gecon_var_info-class class.

Note

The function only displays the elements of a model that have been already set or computed. Eg. if the model has been solved but the statistics have not been computed, the correlations will not be passed to the gecon_var_info class.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

shock_info for information about the shocks.
get_ss_values, get_pert_solution, get_moments to extract the steady-state (equilibrium) values, the solution, and various moments and statistics of the model.

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = 'gEcon'),
          'rbc.gcn'), to = getwd())

# make and load the model
rbc <- make_model('rbc.gcn')

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# get information about variables
rbc_var_info <- var_info(rbc, var_names = c('K_s'))
print(rbc_var_info)
show(rbc_var_info)
summary(rbc_var_info)
```

Index