Package 'dynmdl'

July 21, 2020

Type Package

Title An R version of Dynare

Version 1.4.0

Author Rob van Harrevelt [aut, cre]

Maintainer Rob van Harrevelt <rvanharrevelt@gmail.com>

Description An R version of Dynare.

SystemRequirements GNU make

License GPL (>= 2)

LazyData TRUE

RoxygenNote 7.1.0

Roxygen list(markdown = TRUE, r6 = FALSE)

LinkingTo Rcpp, BH, Rcereal

Imports Rcpp,
    nleqslv,
    Matrix,
    compiler,
    R6,
    geigen,
    gsubfn,
    tools,
    stringi,
    umfpackr,
    numDeriv,
    openxlsx,
    tictoc

Depends R (>= 3.5.0),
    regts,
    methods

Suggests testthat,
    knitr,
    rmarkdown,
    caret

VignetteBuilder knitr

R **topics documented:**

---

all.equal *Test if two* DynMdl *objects are (nearly) equal*

Description

all.equal(x,y) is a utility to compare R objects x and y testing near equality. If they are different, comparison is still made to some extent, and a report of the differences is returned. Do not use all.equal directly in if expressions - use isTRUE(all.equal(...)).

Usage

```
## S3 method for class 'DynMdl'
all.equal(target, current, ...)
```

Arguments

target	and DynMdl object
current	another DynMdl object, to be compared with target
...	Arguments passed to the internal call of all.equal.

Details

The implementation of all.equal for DynMdl objects first serialized the model using the DynMdl method serialize and then uses all.equal of the base package.

Value

Either TRUE or a character vector describing the differences between target and current.

See Also

all.equal

Examples

```
mdl <- islm_mdl("2017Q2/2018Q2")
mdl2 <- mdl$copy()
print(all.equal(mdl, mdl2))

# now modify mdl2
mdl2$set_endo_values(600, names = "c")
print(all.equal(mdl, mdl2))
```

---

change_data-methods   [DynMdl](#) *methods: changes the endogenous or exogenous model data by applying a function.*

Description

These methods of R6 class [DynMdl](#) changes endogenous and/or exogenous model data by applying a function. If the model has trends, then the change is applied to the trended model variables.

Usage

```
mdl$change_endo_data(fun, names, pattern, period = mdl$get_data_period(), ...)

mdl$change_exo_data(fun names, pattern, period = mdl$get_data_period(), ...)

mdl$change_data(fun, names, pattern, period = mdl$get_data_period(), ...)
```

mdl is a [DynMdl](#) object

Arguments

fun   a function applied each model variable specified with argument names or pattern. See Details.

names   a character vector with variable names

pattern   a regular expression for selecting the names of variables whose values must be changed

period   an [period_range](#) object or an object that can be coerced to a period_range: the period for which the function will be applied

...   arguments passed to fun

If neither names nor pattern have been specified, then the function is applied to all endogenous or exogenous variables.

Details

The function specified with argument fun should be a function with at least one argument, for example fun = function(x) {x + 0.1}. The first argument (named x in the example) will be the model variable. The function is evaluated for each model variable separately. The values of the model variables for period range period are passed as a normal numeric vector (not a timeseries) to the first argument.

An example may help to clarify this. Consider the following statement

```
mdl$change_endo_data(fun = myfun, names = c("c", "y"),
                     period = "2017q1/2017q2"),
```

where mdl is a DynMdl object and myfun some function whose details are not relevant here. Method change_endo_data evaluates this as

```
data <- mdl$get_endo_data(names = c("c", "y"), period = "2017q1/2017q2")
data[, "c"] <- myfun(as.numeric(data[, "c"]))
data[, "y"] <- myfun(as.numeric(data[, "y"]))
mdl$set_data(data)
```

The function result must be a vector (or timeseries) of length one or with the same length as the number of periods in the period range `period`.

Methods

change_endo_data Changes endogenous model variables, including fit instruments and Lagrange multipliers used in the fit method (if present).

change_exo_data Changes exogenous model variables

change_data Changes endogenous and/or exogenous model variables, including fit instruments and Lagrange multipliers used in the fit method (if present).

See Also

[get_data-methods](), [set_data](), [set_values-methods]() and \link{change_static_data-methods}.

Examples

```
mdl <- islm_mdl(period = "2017Q1/2017Q3")

# increase y and yd with 10% for the full data period
mdl$change_endo_data(pattern = "^y.?$", fun = function(x) {x * 1.1})
print(mdl$get_endo_data())

# increase ms in 2017Q1 and 2017Q2 with 10 and 20, resp.
mdl$change_exo_data(names = "ms", fun = function(x, dx) {x + dx},
                    dx = c(10, 20), period = "2017Q1/2017Q2")
print(mdl$get_exo_data())
```

---

change_static_data-methods

> [DynMdl]() *methods: changes static values of the endogenous or exogenous model data by applying a function.*

Description

These methods of R6 class [DynMdl]() changes the static values of the endogenous and/or exogenous model data by applying a function.

Usage

```
mdl$change_static_endos(fun, names, pattern, ...)

mdl$change_static_exos(fun names, pattern, ...)

mdl$change_static_data(fun, names, pattern, ...)
```
mdl is a [DynMdl](#) object

Arguments

fun a function applied each model variable specified with argument `names` or `pattern`. See Details.

names a character vector with variable names

pattern a regular expression for selecting the names of variables whose values must be changed

... arguments passed to `fun`

If neither `names` nor `pattern` have been specified, then the function is applied to all endogenous or exogenous variables.

Details

The function specified with argument `fun` should be a function with at least one argument, for example `fun = function(x) {x + 0.1}`. The first argument (named `x` in the example) will be the model variable. The function is evaluated for each model variable separately.

The function result must be a vector of length one.

Methods

change_static_endos Changes the static values of endogenous model variables, including fit instruments and Lagrange multipliers used in the fit method (if present).

change_static_exos Changes the static values of exogenous model variables

change_static_data Changes the static values of endogenous and/or exogenous model variables, including fit instruments and Lagrange multipliers used in the fit method (if present).

See Also

[set_static_data](#), [set_static_values-methods](#) and [change_data-methods](#)

Examples

```
mdl <- islm_mdl()

# increase y the static values of y and yd with 10% for the full data period
mdl$change_static_endos(pattern = "^y.?$", fun = function(x) {x * 1.1})
print(mdl$get_static_endos())

# increase ms with 10
mdl$change_static_exos(names = "ms", fun = function(x, dx) {x + dx},
                dx = 10)
```

```
print(mdl$get_static_exos())
```

check	*DynMdl method: Compute the eigenvalues of the linearized model around the steady state.*

Description

This method of R6 class DynMdl computes the steady state, constructs a linear model around the state steady and finally computes the eigenvalues of the linearized model around the steady state. It also checks if the Blachard and Kahn conditions are satisfied. The Blanchard and Kahn conditions state that the number of eigenvalues larger than 1 should be equal to the number of forward looking variables (variables with leads).

Usage

```
mdl$check(tol = sqrt(.Machine$double.eps))
```

mdl is a DynMdl object

Arguments

tol   The tolerance parameter used to test if an eigenvalue is significantly larger than 1 when checking the Blanchard-Kahn conditions. The default is the square root of the machine precision (typically about 1.5e-8). See Details.

Details

To test if the Blanchard-Kahn conditions are satisfied, we need to determine the number of eigenvalues larger than 1. If an eigenvalue is exactly equal to 1, the actually calculated eigenvalue may be slightly larger than 1 because of rounding errors. To check the Blanchard-Kahn conditions, we therefore count the number of eigenvalues larger than 1 + tol, where tol is a small number (by default the square root of the machine precision). Use argument tol to change this tolerance parameter.

Method check also prints the number of eigenvalues with absolute values between 1 -tol and 1 + tol. Thus the values of these eigenvalues are not significantly different from 1.

Warning

Method check is only possible for models with a maximum lag and lead of 1. If the original model has lags or leads greater than 1, use argument max_lag_lead_1 = TRUE of function dyn_mdl to create a transformed model with maximum lag and lead 1.

See Also

solve_steady and get_eigval

Examples

```
mdl <- islm_mdl()
mdl$check()
print(mdl$get_eigval())
```

---

clear_fit   [DynMdl](#) *method: removes fit targets and turns off fit instruments.*

Description

This method of R6 class DynMdl removes all fit targets, sets the sigma-parameters of the fit-instruments to -1 and sets all Lagrange multipliers to 0, for both the dynamic and static version of the model.

By removing the fit targets (which is equivalent to setting all fit targets to NA), all endogenous variables are calculated according to the equations of the model, while the fit instruments stay fixed at their current value, and are efficitively exogenous (even though they are still implemented as endogenous variables).

If the model had been solved before clear_fit was called, then the model is still solved after clear_fit has been called.

Usage

```
mdl$clear_fit()
```

mdl is a [DynMdl](#) object implementing the fit method. '

See Also

[get_data-methods](#), [set_fit](#), [set_fit_steady](#), [set_fit_values](#) and [clear_fit](#).

Examples

```
mdl <- islm_mdl(period = "2016Q1/2017Q3", fit = TRUE)

# create a regts with fit targets
y <- regts(c(1250, 1255, 1260), start = "2016Q1")
t <- regts(c(250, 255), start = "2016Q1")
fit_targets <- cbind(y, t)

# register the fit targets in the DynMdl object
mdl$set_fit(fit_targets)

mdl$solve()
mdl$clear_fit()

# the next statements gives 0 iterations.
mdl$solve()
```

---

copy   [DynMdl](#) *method: Returns a copy of this* DynMdl *object*

Description

This method of R6 class [DynMdl](#) returns a deep copy of a DynMdl object

Usage

```
mdl$copy()
```

mdl is a [DynMdl](#) object

Details

mdl$copy() is equivalent to mdl$clone(deep = TRUE)

Examples

```
mdl <- islm_mdl("2017Q1/2019Q2")
mdl2 <- mdl$copy()
```

DynMdl   *An R6 class for a Dynare model*

Description

An R6 class for a Dynare model

Format

[R6Class](#) object.

Value

Object of [R6Class](#) containing a macro-economic model,

Methods

[get_max_lag](#) Returns the maximum lag

[get_max_lead](#) Returns the maximum lead

[get_endo_names](#) Returns the names of the endogenous variables.

[get_exo_names](#) Returns the names of the exogenous variables.

[set_labels](#) Set labels for the model variables

get_labels Returns the labels of the model variables and parameters

get_tex_names Returns the LaTeX names of the model variables and parameters

get_par_names Returns the names of the parameters.

set_param Sets the parameters of the model.

set_param_values Sets the values of one or more model parameters.

get_param Returns the parameters of the model.

get_all_param Returns the parameters of the model, including the sigma parameters used in the fit method.

set_static_exos Sets the static values of the exogenous variables used to compute the steady state.

set_static_endo_values Sets the values of one or more static endogenous variables

set_static_exo_values Sets the values of one or more static exogenous variables

get_static_exos Returns the static values of the exogenous variables.

set_static_endos Sets the static values of the endogenous variables.

get_static_endos Returns the static values of the endogenous variables.

set_static_data Sets the static values of the model variables.

change_static_endos Changes the static values of endogenous model variables by applying a function

change_static_exos Changes the static values of exogenous model variables by applying a function

change_data Changes the static values of endogenous and exogenous model variables by applying a function.

get_static_data Returns the static values of the model variables.

get_all_static_endos Returns the static values of the endogenous model variables, including fit instruments and lagrange multupliers.
#

get_all_static_data Returns the static values of the exogenous and endogenous model variables, including fit instruments and lagrange multupliers.

init_data Initializes the model data

set_period Sets the model period

get_period Returns the model period

get_data_period Returns the model data period.

get_lag_period Returns the lag period.

get_lead_period Returns the lead period

set_endo_values Sets the values of endogenous model variables

set_exo_values Sets the values of exogenous model variables

set_data Transfer timeseries to the model data

change_endo_data Changes the values of endogenous model variables by applying a function

change_exo_data Changes the values of exogenous model variables by applying a function

`change_data` Changes the values of endogenous and exogenous model variables by applying a function.

`get_data` Returns the model data, including exogenous end exogenous variables, fit instruments and trend variables (but excluding Lagrange multipliers).

`get_all_endo_data` Returns all endogenous model variables, including exogenous end exogenous variables, fit instruments and trend variables (but excluding Lagrange multipliers).

`get_all_data` Returns all endogenous end exogenous model varieables, including fit instruments and Lagrange multipliers

`get_endo_data` Returns the endogenous model data

`get_exo_data` Returns the exogenous model data

`get_vars_pars` Returns a list with model variables and parameters

`solve_steady` Solves the steady state

`solve` Solves the model

`solve_steady_dynare` Solves the steady state and eigenvalues with Dynare (emplpying Matlab or Octave)

`solve_dynare` Solves the model with Dynare (employing Octave or Matlab)

`check` Compute the eigenvalues of the linear system and check if the Blachard and Kahn conditions are satisfied.

`residual_check` Calculates the residuals of the equations and reports the differences larger than a tolerance parameters

`static_residual_check` Calculates the residuals of the static model equations and reports the differences larger than a tolerance parameters

solve_perturbation Solves the model using the perturbation theory used in the Dynare function stoch_simul. Only shocks in the first solution period are allowed.

`get_jacob` Returns the Jacobian for the dynamic model

`get_static_jacob` Returns the Jacobian for the static version of the model

`get_back_jacob` Returns the Jacobian for a backward looking model at a specific period

`get_eigval` Returns the eigenvalues computed with method `check` or solve_perturbation

`get_equations` Returns a character vector with the parsed equations of the model.

`get_original_equations` Returns a character vector with the equations of the original model.

`copy` Returns a deep copy of the `DynMdl` object

`get_solve_status` Returns the status of the last model solve attempt

`get_mdldef` Returns a list with technical details of the model.

Methods for the fit method

`get_instrument_names` Returns the names of the fit instruments.

`get_sigma_names` Returns the names of the sigma parameters used in the fit procedure.

`set_fit_values` Sets the values of the fit targets

`set_fit` Sets the targets for the fit procedure

set_fit_steady Sets the targets for the fit procedure for the steady state.

get_fit Returns the fit targets used in the fit procedure

get_fit_steady Returns the fit targets used in the fit procedure for the steady state.

get_fit_instruments Returns all non-zero fit instruments used in the fit procedure

set_sigma Sets one or more sigma parameters used in the fit method

set_sigma_values Sets the values of sigma parameters used in the fit method

get_sigmas Returns all sigma parameters >= 0 used in the fit procedure. If a sigma parameter is negative, then the corresponding fit instrument is not included

get_lagrange Returns the Lagrange multipliers used in the fit procedure.

run_initval Run the initval equations to obtain new values for endogenous and exogenous static variables for the current values or parameters.

---

dyn_mdl   *Creates a* DynMdl *object from a mod file*

Description

Creates a DynMdl object from a mod file. If the mod file contains a fit block, then the DynMdl object implements the fit procedure (except if argument fit is FALSE).

Usage

```
dyn_mdl(
  mod_file,
  period,
  data,
  base_period = NULL,
  calc = c("internal", "R", "bytecode", "dll"),
  fit_mod_file,
  debug = FALSE,
  dll_dir,
  max_laglead_1 = FALSE,
  strict = TRUE,
  warn_uninit_param = TRUE,
  init_param_na = FALSE,
  fit = TRUE,
  fit_fixed_period = FALSE,
  check_static_eqs = TRUE,
  latex = TRUE,
  latex_options,
  nostrict,
  silent = FALSE
)
```

Arguments

mod_file	the name of the model file (including extension .mod)
period	a [period_range](#) object specifying the model period, i.e. the period range for which the model will be solved. Thus this period range excludes the lag and lead period.
data	the model data as a [regts](#) object with column names
base_period	a [period](#) object specifying the base period for the trends. This is used if the model has trend variables. All trend variables will be equal to 1 at the base period.
calc	Method used to evaluate the model equations. Possible values are `"internal"`, `"R"`, `"bytecode"` and `"dll"`. See Details.
fit_mod_file	the name of the generated fit mod file. If not specified, then the fit mod file is destroyed after the model has been parsed. This argument should not be specified if the model contains trends, since in that case the fit mod file cannot be used a input mod file for function dyn_mdl or for Dynare. If wou want to check the equations in the fit mod file, use argument DEBUG (see below).
debug	If logical (default FALSE), only used when the model is a fit model. If TRUE, then intermediate files created when preparing the fit model are written to the current directory. By default these files are written in a temporary directory and deleted when the R session terminates.
dll_dir	the directory where the dynamically linked library is stored. Primarily used for testing. Only used if argument use_dll is TRUE.
max_laglead_1	a logical indicating whether the model should be transformed internally to a model with a maximum lag and lead of 1. The default is FALSE. This option has no effect if the maximum lag and lead of the original model is 1. Set this argument to TRUE if you want to analyse the stability of the steady state with method [check](#) for models with a maximum lag or lead larger than 1.
strict	A logical. If TRUE (the default), then an error is given when endogenous or exogenous variables are not used in the model block or when an undeclared symbol is assigned in the initval block. If FALSE, then only a warning is issued, unused endogenous variables are removed, and the assignments of undeclared symbols in the initval block are ignored.
warn_uninit_param	
	A logical. If TRUE (the default) then a warning is given for each parameter that has not been initialized in the mod file. Uninitialized parameters are set to zero or NA, depending on argument init_param_na.
init_param_na	A logical (default FALSE). If TRUE, then the parameters that have not been initialized in the mod file are set to NA. Otherwise these parameters are initialised with zero.
fit	a logical (default TRUE) indicating if the DynMdl object returned by this function should implement the fit procedure if the mod file contains a fit block. Specify FALSE if the mod file has a fit block while you do not want to use the fit procedure.

fit_fixed_period

a logical. If TRUE, then the fit conditions are derived for a fixed period, treating lags and leads as exogenous variables. If FALSE (the default), the fit conditions are derived from the stacked-time equations. This option is particularly useful for backward looking models (models without leads). If fit_fixed_period is TRUE, then the fit equations will not contain lags or leads. If fit_fixed_period is TRUE, then some fit equation may contain leads.

check_static_eqs

a logical. If TRUE (the default), then we check if the mod file contains separate static and dynamic equations (i.e. equations tagged with static and dynamic). If this is the case, separate static and dynamic fit equations are generated when necessary (separate equations are not generated if the static version is simply equal to the dynamic version when lags and leads are removed).

latex   A logical. If TRUE (the default), then LaTeX files are created if the model block contains a write_latex_static_model, write_latex_dynamic_model or write_latex_original_model statement.

latex_options   a list with options for writing LaTeX files. See Details.

nostrict   Obsolete: the logical negation of argument strict. This argument should not be used in new code: use argument strict instead.

silent   A logical (default FALSE). If TRUE, then output of the Dynare parser is suppressed except for warnings.

Details

Evaluation of model equations:

There are several methods available for evaluating the model equations and the Jacobian. These methods can be specified with argument calc. The possible methods are

R the model equations and Jacobian are evaluated using R functions . This is very slow for large models. This method should therefore only be used for small models.

bytcode the same as R, except that the R function is turned into byte code. This is usually only slightly faster than using the R method

dll The model equations and Jacobian are evaluated using a shared library created for this specific model. Function dyn_mdl generates C code for evaluating the equations and the Jacobian, and subsequently compiles the C code to create a shared library. The evaluation of the equations and Jacobian is much faster than for the R and bytecode methods. However, the compilation of the C code can take a considerable amount time for large models.

internal This methods converts the equations to internal byte code using reverse Polish notation. The internal byte code is evaluated using compiled C++ code that is part of package dynmdl. For this method the evaluation of the equations and Jacobian is as fast as for the dll method, but the compilation time is much faster. However, the internal method does not yet support all features of Dynare models. For example, it cannot handle model-local variables and some built-in functions (the only supported built-in functions are currently exp, log, sqrt, abs and sign).

If possible, use the internal method, because this method is faster than the other methods for both compiling the model and for evaluating the equations and Jacobian. The internal method is therefore the default method. However, as explained above, the mod file may contain features not yet supported for the internal method, in which case another method must be selected.

Latex options: When the mod file contains a `write_latex_static_model`, `write_latex_dynamic_model` or `write_latex_oiriginal_model` statement, then the Dynare parser of package dynmdl generates LaTeX files in directory `latex`. Argument `latex_options` can be used to change the format of the LaTeX files. It should be a named list containing one or more of the following components:

par_as_num A logical. If `TRUE`, then the parameters are written as numerical constants to the LaTeX file, using the numerical values as specified in the mod file. The default is `FALSE`.

ndigits The number of significant digits used when parameters as written as numerical values (default 4). This argument is only used if `par_as_num` is `TRUE`. For example, if `ndigits` is 4, then the number π is printed as `3.142`, the number 120.25 as `120.2`, and the number 10.1234 as `1.012e+05`

Value

A `DynMdl` object.

---

get_data-methods   [DynMdl](#) *methods: Retrieve timeseries from the model data*

Description

These methods of R6 class [DynMdl](#) can be used to retrieve timeseries from the model data.

If the `DynMdl` object is also a [DynMdl](#) object, then `get_data` also returns the fit instruments. In contrast, `get_endo_data` does not return these fit instruments. Both `get_data` and `get_endo_data` do not return the Lagrange multipliers used in the fit method. Use method [get_lagrange](#) to obtain these Lagrange multipliers. [get_all_endo_data](#) returns all endogenous variables, fit instruments and Lagrange multipliers, for the complete data period. [get_all_data](#) also returns all exogenous variables. These functions can be useful to save the complete solution of a model that be used as initial values for the endogenous variables in another model.

Usage

```
mdl$get_data(pattern, names, period = mdl$get_data_period(),
            trend = TRUE)

mdl$get_endo_data(pattern, names, period = mdl$get_data_period(),
                  trend = TRUE)

mdl$get_exo_data(pattern, names, period = mdl$get_data_period())

mdl$get_trend_data(pattern, names, period = mdl$get_data_period())

mdl$get_all_endo_data(trend = TRUE)

mdl$get_all_data(trend = TRUE)
```

mdl is a [DynMdl](#) object.

Arguments

pattern a regular expression for selecting the names of variables.

names a character vector with variable names.

period an [period_range](#) object or an object that can be coerced to a period_range.

trend a logical. This argument is used for model with trend variables. If TRUE (the default), then the endogenous variables are multiplied with their trends (called deflators in the mod file).

If neither names nor pattern have been specified, then all variables with the specific type are returned.

Methods

- get_data: All model variables: exogenous and endogenous model variables, trends variables, and fit instruments for [DynMdl](#) objects
- get_endo_data: Endogenous model variables, excluding fit instruments.
- get_exo_data: Exogenous model variables
- get_trend_data: Trend variables (variables declared with trend_var in the mod file).
- get_all_endo_data: All endogenous variables, including fit instruments and lagrange multipliers.
- get_all_data: All endogenous and exogenous variables, including fit instruments and lagrange multipliers.

See Also

[get_fit-methods](#), [get_fit](#), [get_fit_instruments](#), [get_lagrange](#) and [get_vars_pars](#).

Examples

```
mdl <- islm_mdl(period = "2017Q1/2017Q3")

mdl$get_data(names = "c", pattern  = "y.", period = "2017Q1/2017Q2")
```

---

get_eigval   [DynMdl](#) *method: Return the eigenvalues computed with method* check

---

Description

This method of R6 class [DynMdl](#) returns the eigenvalues computed with method [check](#), ordered with increasing absolute value

Usage

DynMdl method:

mdl$get_eigval()

mdl is a [DynMdl](#) object

See Also

check

---

get_equations	DynMdl *method: Returns a character vector with the dynamic model equations*

Description

These method of R6 class DynMdl returns a character vector with the model equations (excluding local equations) of the dynamic version of the model. get_equations returns the parsed equations, where trends (if present) have been removed. get_original_equations returns the equations as defined in the mod file.

Usage

DynMdl method:

```
md$get_equations(i)
mdl$get_original_equations(i)
```

mdl is a DynMdl object

Arguments

i A numeric vector with the indices of the non-local equations. If not specified, then the function returns all equations

Examples

```
mdl <- islm_mdl(period = "2018Q1/2023Q3")

# print the 4th equation nicely to the screen
cat(mdl$get_equations(4))

# print all equations
print(mdl$get_equations())
```

---

get_fit-methods   *DynMdl methods: get variables used in the fit method.*

Description

These methods of R6 class DynMdl can be used to retrieve the variables used in the fit method: the fit targets, fit instruments or Lagrange multipliers.

By default, function get_fit only returns fit targets with any non-NA value for the period range with any non-NA value. Thus columns with only NA values and leading and trailing rows with only NA values are removed. If all values are NA, then the function returns NULL. If argument period has been specified, then the function always returns a timeseries with the specified period range. If names or pattern has been specified, it always returns a timeseries with the specified variables, except if all values are NA.

For method get_fit there are corresponding set_fit and set_fit_values methods. There are currently no special methods to set or change the fit instruments and Lagrange multipliers. However, since they are internally implemented as endogenous variables you can use methods set_data, set_endo_values, and change_endo_data to change the fit instruments or Lagrange multipliers.

Usage

```
mdl$get_fit(pattern, names, period) # fit targets

mdl$get_fit_instruments(pattern, names, period = mdl$get_period())

mdl$get_lagrange(names, period = mdl$get_period())
```
mdl is a DynMdl object implementing the fit method.

Arguments

pattern a regular expression for selecting the names of the variables

names a character vector with variable names

period an period_range object or an object that can be coerced to a period_range

See Also

get_data-methods, set_fit, set_fit_values and clear_fit.

Examples

```
mdl <- islm_mdl(period = "2016Q1/2017Q3", fit = TRUE)

# create a regts with fit targets
y <- regts(c(1250, 1255, 1260), start = "2016Q1")
```

```
t <- regts(c(250, 255), start = "2016Q1")
fit_targets <- cbind(y, t)

# register the fit targets in the DynMdl object
mdl$set_fit(fit_targets)

mdl$solve()

print(mdl$get_fit())
print(mdl$get_fit_instruments())
print(mdl$get_lagrange())
```

---

get_fit_steady   [DynMdl](#) *method: get fit targets used in the steady state calculation*

Description

This methods of R6 class `DynMdl` returns the static fit targets, i.e. the fit targets used when the steady state is solved.

By default, function `get_fit` only returns fit targets with any non-NA value. If all values are NA, then the function returns NULL. If `names` or `pattern` has been specified, it always returns a timeseries with the specified variables.

For method `get_fit_syteady` the is a corresponding [set_fit_](#)steady method. There are currently no special methods to set or change the fit instruments and Lagrange multipliers. However, since they are internally implemented as endogenous variables you can use methods [set_static_data](#) to change the static fit instruments or Lagrange multipliers.

Usage

```
mdl$get_fit_steady(pattern, names) # fit targets
```

mdl is a [DynMdl](#) object implementing the fit method.

Arguments

pattern a regular expression for selecting the names of the fit targets.

names a character vector with variable names.

See Also

[set/get_static_endos/exos](#), [set_fit_steady](#), and [clear_fit](#).

Examples

```
mdl <- islm_mdl(fit = TRUE)

fit_targets <- c(y = 1250, t = 255)

# register the fit targets in the DynMdl object
mdl$set_fit_steady(fit_targets)

mdl$solve_steady()

print(mdl$get_fit_steady())
```

---

get_instrument_names/get_sigma_names

> DynMdl *methods: Retrieve the names of the fit instruments or sigma*
> *parameters used in the fit method.*

Description

These methods of R6 class DynMdl return the names of the fit instruments or sigma parameters used in the fit method.

Usage

```
mdl$get_instrument_names(all = FALSE)

mdl$get_sigma_names()
```

mdl is a DynMdl object implementing the fit method.

Arguments

all   A logical (default FALSE). If TRUE, the names of all fit instruments are returned, including the inactive fit instruments.

See Also

get_fit_instruments, get_sigmas

Examples

```
mdl <- islm_mdl(period = "2017Q1/2018Q3", fit = TRUE)
print(mdl$get_instrument_names())
print(mdl$get_sigma_names())
```

---

get_jacob   [DynMdl](#) *methods: Return the Jacobian for the static or dynamic model*

Description

These methods of R6 class [DynMdl](#) can be used to retrieve the Jacobian for the static or dynamic model.

The methods return a `matrix` object which can be further analysed using standard linear algebra functions. This is particularly useful when method [solve](#) complains about (nearly) singular Jacobians. For example, an SVD decomposition using function [svd](#) can be used to identify linearly dependent rows or columns.

Usage

```
mdl$get_static_jacob(sparse = FALSE)

mdl$get_jacob(sparse = FALSE)

mdl$get_back_jacob(period, sparse = FALSE)

mdl is a DynMdl object
```

Arguments

sparse   a logical. If `TRUE`, then the matrix is returned as a sparse matrix

period   an [period](#) object or an object that can be coerced to a `period`

Methods

- `get_jacob`: The Jacobian for the dynamic model This is the Jacobian used when solving the model with the stacked-time Newton method.
- `get_static_jacob`: The Jacobian for the static version of the model. This Jacobian is used when solving the steady state.
- `get_back_jacob`: The Jacobian at a specific period for backward looking models, treating the lags as exogenous. This Jacobian is used to solve backward looking models.

Examples

```
mdl <- islm_mdl("2018Q1/2019Q4")
print(mdl$get_static_jacob())
print(mdl$get_jacob())

## Not run:
# print the Jacobian for a backward looking model at period 2018Q3
print(backwards_mdl$get_back_jacob("2018Q3"))

## End(Not run)
```

---

get_labels/get_tex_names

> DynMdl *method: Returns the labels or LaTeX names of the model variables and parameters*

Description

These methods of R6 class DynMdl return the labels (long names) or LaTeX names of the model variables and parameters. The return value is a named character vector.

The labels and LaTeX names are defined in the mod file (consult the documentation of Dynare, in Dynare labels are called 'long names'). Method set_labels can be used to modify these labels. By default the labels are equal to the variable names.

Usage

```
mdl$get_labels()

mdl$get_tex_names()
```

mdl is a DynMdl object

Methods

- get_labels: Returns the labels (long names), e.g. "Disposable income"
- get_tex_names: Returns the LaTeX names (e.g. "Y_d")

See Also

set_labels

---

get_max_lag/get_max_lead

> DynMdl *methods: Returns the maximum lag or lead*

Description

Methods get_max_lag and get_max_lead of R6 class DynMdl return the maximum lag and lead of the original model, respectively. These are the maximum lag and lead in the equations specified in the mod file. The actual maximum lag or lead will be different if max_laglead_1 == TRUE and if there are endogenous lags or leads greater than 1.

Usage

```
DynMdl methods:

 mdl$get_max_lag()
 mdl$get_max_lead()
```

mdl is a [DynMdl](#) object

---

get_mdldef   [DynMdl](#) *method: Returns technical details about the model.*

Description

This function returns a list with various components containing technical details of the model, such as the names of the variables, the lead-lag incidence matrix, etc.

Usage

```
DynMdl method:

mdl$sget_mdldef()
```

mdl is a [DynMdl](#) object

---

get_name-methods   [DynMdl](#) *methods: Retrieve the names of model variables or parame-
                          ters*

Description

These methods of R6 class [DynMdl](#) return the names of the model variables or parameters

If the DynMdl object is also a [DynMdl](#) object, then get_endo_names and get_exo_names do not include the names of the auxiliary endogenous and exogenous variables used in the fit method. Use [get_instrument_names](#) to obtain the names of the fit instruments.

Usage

```
mdl$get_endo_names(type = c("all", "lags", "leads"))

mdl$get_exo_names()

mdl$get_par_names(pattern = ".+")
```

mdl is a [DynMdl](#) object

Arguments

type   a character describing the type of the endogenous variables: `"lags"` or `"leads"` for endogenous variables with lags or leads, respectively. The default is `"all"` (all endogenous variables).

pattern   A regular expression. If specified, then only names matching the regular expression are returned.

Methods

- get_endo_names: Names of the endogenous model variables.

- get_exo_names: Names of the endogenous model variables.

- get_par_names: Names of the model parameters (excluding the sigma parameters used in the fit method).

See Also

get_instrument_names and get_sigma_names

Examples

```
mdl <- islm_mdl()

# print the names of all variables with leads
print(mdl$get_endo_names(type = "lead"))

# print parameters starting with "c"
print(mdl$get_par_names(pattern = "^c.*"))
```

---

get_period-methods   *DynMdl* *method: return the model, data, lead or lag period*

Description

These methods of R6 class DynMdl return the model period, data period, lag period, lead period, or base period respectively.

The *model period* is the default period for which the model will be solved. The *data period* is the period for which the model contains the values for the endogenous and exogenous variables. If a model has lags, then the data period always include the *lag period*: the period before the model period where the lags needed to solve the model in the model period are stored. For a model with leads the model data period also includes a *lead_period*. Thus, the data period always contains the lag period, model period and lead period, but it may also be longer. See the example below.

The *base_period* is used when the model has trend variables. All trend variables will be equal to 1 at the base period.

Usage

```
mdl$get_period()

mdl$get_data_period()

mdl$get_lag_period()

mdl$get_lead_period()

mdl$get_base_period()

mdl is a DynMdl object
```

Methods

- get_period: Returns the model period
- get_data_period: Returns the data period
- get_lag_period: Returns the lag period, or NULL if the model has no lags
- get_lead_period: Returns the lead period or NULL if the model has no leads
- get_base_period: Returns the base period.

See Also

set_period and init_data

Examples

```
# For this example we first create a model with a data period
# starting many periods before the model period.
mdl <- islm_mdl()
mdl$init_data("1997Q1/2022Q4")
mdl$set_period("2017Q4/2022Q3")

print(mdl$get_period())        # result: "2017Q1/2022Q3"
print(mdl$get_data_period())   # result: "1997Q1/2022Q4"

# This model has a maximum lag and lead of 1, so the lag
# and lag period are simple the period before and after the model period.
print(mdl$get_lag_period())    # result: "2017Q3"
print(mdl$get_lead_period())   # result: "2022Q4"
```

get_solve_status   [DynMdl] *method: Returns the solve status of the last model solve.*

Description

This method of R6 class [DynMdl] returns the status of the last model solve as a text string. If the last model solve was succesfull, it returns the string "OK".

Usage

DynMdl method:

```
mdl$get_solve_status()
```

mdl is a [DynMdl] object

Details

The possible return values are:

- NA_character_ (method solve has not yet been called)
- "OK"
- "ERROR" (an error has occurred, check the warnings).

See Also

[solve] and [solve_steady]

Examples

```
## Not run:
mdl <- islm_mdl(period = "2017Q1/2018Q4")
mdl$set_endo_values(NA, names = "y", period = "2017Q1")
mdl$solve()
if (mdl$get_solve_status() != "OK") {
   stop("Error solving the model. Check the warnings!")
}

## End(Not run)
```

---

get_vars_pars   `DynMdl` *methods: Returns a list of all model variables and parameters*

Description

This method of R6 class `DynMdl` returns a list of all model variables and parameters. This makes it easy to directly evaluate expressions involving both model variables and parameters.

If the DynMdl object is also a `DynMdl` object, then the variables do not include the the auxiliary endogenous and exogenous variables used in the fit method.

Usage

```
mdl$get_vars_pars(period = mdl$get_data_period(), trend = TRUE)
```

mdl is a `DynMdl` object

Arguments

period an `period_range` object or an object that can be coerced to a period_range #'

trend a logical. This argument is used for model with trend variables. If TRUE (the default), then the endogenous variables are multiplied with their trends (called deflators in the mod file)

See Also

`get_data-methods`, `get_param`, `get_fit`, `get_fit_instruments` and `get_lagrange`

Examples

```
mdl <- islm_mdl(period = "2017Q1/2017Q3")

# create a list of all parameters and model variables for
# period 2017q1/2017q2
vars_pars <- mdl$get_vars_pars(period = "2017Q1/2017Q2")
print(vars_pars)

# evaluate an expression within list vars_pars
with(vars_pars, print(t0 + t1 * y))

# copy all parameters to the global environment, and evaluate
# an expresions in the global environment:
list2env(vars_pars, .GlobalEnv)
print(md - ms)
```

---

init_data   [DynMdl] *method: initializes the model data.*

Description

This method of R6 class [DynMdl] initializes all model variables with static values for the whole data period. All endogenous and exogenous variables, fit instruments and lagrange multipliers are set to the static values. Fit targets specified for the dynamic model are removed. If static fit targets have been specified with method [set_fit_steady], then the dynamic fit targets are set to the static fit targets for the whole data period.

If argument data has been specified, then the model data are subsequently updated with the timeseries in data. For models implementing the fit method, data may include fit instruments and lagrange multipliers. All timeseries in data that are no model variables, fit instruments or lagrange multipliers are silently skipped.

If the model period has not yet been specified (in function [dyn_mdl] or method [set_period]), then this method also sets the model period, the standard period for which the model is solved. The model period is obtained from the data period by subtracting the lag and lead periods.

Usage

```
mdl$init_data(data_period = NULL, data = NULL, upd_mode = c("upd", "updval"))
```

mdl is a DynMdl object

Arguments

data_period a [period_range] object, or an object that can be coerced to [period_range]. The (new) data period, i.e. the period range of all model timeseries. If not specified, then the data period is based on the period range of argument data (if this argument has been specified) and the model period.

data a [ts] or [regts] object with values for endogogenous and exogenous model variables, including fit instruments and Lagrange multipliers used in the fit method. If data has labels, then these labels are used to update the model labels. If the model has trends, then the timeseries in data should include the trends.

upd_mode the update mode, a character string specifying how the timeseries in object data are transferred to the model data. For "upd" (standard update, default), the timseseries in data are used to replace the steady state values of the exogenous and endogenous model variables. For "updval", the static model variables are only replaced by valid (i.e. non-NA) values in data).

If neither data_period nor data has been specified, then the data period is unchanged. In that case the data period must have been set before with method init_data or set_period.

Warning

Method init_data removes all fit targets.

See Also

[set_period](#)

Examples

```
mdl <- islm_mdl()
mdl$init_data("2017Q2/2021Q3")
print(mdl)

# since are variables have the steady state values, a subsequent solve will
# converge in 0 iterations
mdl$solve()
```

---

islm_mdl   *Returns an example ISLM model*

Description

This function returns an example ISLM model, If argument `period` has been specified, then this function also initializes the model data with the steady state values.

Usage

```
islm_mdl(period, fit = FALSE)
```

Arguments

period	the model period for the ISLM model
fit	a logical indicating whether the dynamical fit procedure should be used

Value

a [DynMdl](#) object.

Examples

```
mdl <- islm_mdl("2017Q1/2019Q4")
```

put_static_endos	DynMdl *method: Transfers the static endogenous variables to the model data.*

Description

This method of R6 class DynMdl transfers the static endogenous variables to the model data.

Usage

DynMdl method:

```
mdl$put_static_endos(period = mdl$get_data_period())
```

mdl is a DynMdl object

Arguments

period A period_range object or an object that can be coerced to a period_range, specifying the period for which the endogenous model data will be updated with the static endogenous variables.

See Also

solve_steady, set_static_endos and get_static_endos.

Examples

```
mdl <- islm_mdl(period = "2018Q1/2040Q3")

# transfer static endogenous variables for the full data period
mdl$put_static_endos()

# now only for the lead period
mdl$put_static_endos(period = mdl$get_lead_period())
```

read_mdl	*Reads a model from a RDS file*

Description

This function reads a model from an RDS file that has been written by method write_mdl of an DynMdl object.

Usage

```
read_mdl(file, dll_dir, silent = FALSE)
```

Arguments

file	the name of the RDS file
dll_dir	the directory where the dynamically linked library is stored. Primarily used for testing. Only used if the model was created with the dll option (see function [dyn_mdl](#)).
silent	A logical (default FALSE). It TRUE, then no output is written.

Value

a [DynMdl](#) object.

See Also

[write_mdl](#)

Examples

```
mdl <- islm_mdl("2017Q1/2019Q2")
mdl$write_mdl("islm_mod.rds")
mdl2 <- read_mdl("islm_mod.rds")
```

---

residual_check	[DynMdl](#) *method: Calculates the residuals of the equations*

Description

This method of R6 class [DynMdl](#) calculates the residuals for the full model period and returns the result as a [regts](#) timeseries object.

Usage

```
mdl$residual_check(tol, include_fit_eqs = FALSE)
```

mdl is a [DynMdl](#) object

Arguments

tol the tolerance parameter. If specified, then the return value does not include columns for the equations whose residuals are smaller than tol

include_all_eqs a logical value (default FALSE). If TRUE, then the all equations, including fit equations and auxiliary equations (if present), are included in the residual check. The auxiliary equations are extra equations created when the model has lags or leads greater than 1 and if dynmdl was called with max_laglead_1 = TRUE.

include_fit_eqs a logical value (default FALSE). If TRUE, then fit equations (if present) are included in the residual check. Ignored if include_all_eqs is TRUE.

debug_eqs Debug equations (default FALSE). Only used for the internal calculation mode (calc == "internal", see [dyn_mdl](#)). If TRUE then numerical problems in evaluation of mathematical functions or operators such a log are reported.

See Also

[static_residual_check](#)

---

run_dynare   *Run a mod file Dynare with Matlab or Octave.*

Description

This function runs a mod file with Dynare using either Matlab or Octave. The input files for Dynare and Octave/Matlab are generated automatically. The command octave or matlab should be in the search path.

Usage

```
run_dynare(
  mod_file,
  period,
  data,
  steady = TRUE,
  perfect_foresight = !missing(period) || !missing(data),
  scratch_dir = tempfile(),
  dynare_path = NULL,
  steady_options,
  perfect_foresight_solver_options,
  initval_type = c("m", "xlsx"),
  use_octave = Sys.which("matlab") == "",
  exit_matlab = FALSE
)
```

Arguments

mod_file   The name of a Dynare mod file. For models with trends, this can also be a fit mod file created with function dyn_mdl when argument fit_mod_file was specified. However, for model with trends the fit mod file is not a correct Dynare mod file, because the fit equations are already detrended while the original equations still contain trends.

period   A [period_range](#) object or object that can be coerced to a period_range. This argument must be specified when the perfect foresight solver is called.

data	a [ts](#) or [regts](#) object with values for endogogenous and exogenous model variables used in the perfect foresight solver. If the model has trends, then the timeseries in data should include the trends.
steady	A logical, indicating wether the steady state should be calculated (default is TRUE). If the steady state is calculated then also the eigenvalues are calculated.
perfect_foresight	
	A logical, indicating wether the perfect foresight solver should be called. The default is TRUE if argument period or data have been specified and FALSE otherwise.
scratch_dir	Directory where the Matlablab and Dynare scripts are created. By default this is a temporary directory that is automatically deleted when the R session terminates.
dynare_path	Character string specifying the name of the directory of the Dynare installation. On Linux it is usually not necessary to the specify this argument. On Windows it is necessary to specify the path of the Dynare installation. In you are running R in the CPB environment the path to Dynare is set automatically.
steady_options	Options passed to the steady command of Dynare. This should be a named list, which names corresponding to the Dynare options. Specify a NULL value if the option has no value. Consult the documentation of Dynare for a list of available options. Example: steady_options = list(tolf = 1e-7,nocheck = NULL)
perfect_foresight_solver_options	
	Options passed to the perfect_foresight_solver command of Dynare. This should be a named list, which names corresponding to the Dynare options. Specify a NULL value if the option has no value. Consult the documentation of Dynare for a list of available options. Example: steady_options = list(tolf = 1e-7,no_homotopy = NULL).
initval_type	A character specifying the type of initval file used in the Matlab/Octave job: "m" for a Matlab file and "xlsx" for an xlsx-file.
use_octave	A logical. If TRUE, then Dynare is envoked with Octave, otherwise Matlab is used. By default Matlab is used if available.
exit_matlab	A logical specifying if Matlab should immediately exit when the calcultions have finished Matlab writes the output to a separate console. If exit_matlab is FALSE (the default), then the R job waits until the user has closed this console, or entered exit in the console. Otherwise the console is automatically closed at the end of the calculation and all output is lost. This argument is ignored if Dynare is run with Octave. Octave does not open a seperate console: all output appears in the same console used by R.

Value

A list with the following components

steady_endos	(only if steady == TRUE): a steady state endogenous variables
eigval	(only if steady == TRUE): the eigenvalues of the steady state
endo_data	(only if perfect_foresight_solver == TRUE): the endogenous variables for the solution of the perfect foresight solver

Examples

```
## Not run:
data <- regts(matrix(280, nrow = 1, ncol = 1), names = "g", period = "2017Q1")

result <- run_dynare("mod/islm.mod", period = "2017q1/2019q3", data = data,
                     steady_options = list(tolf = 1e-8),
                     perfect_foresight_solver_options = list(tolf = 1e-8,
                                                             tolx = 1e-8,
                                                             no_homotopy = NULL))


## End(Not run)
```

---

run_initval   [DynMdl](#) *method: Evaluate to initval block to obtain new values for*
                          the static endogenous and exogenous variables based on the current
                          parameters.

Description

This function runs the equations in the initval block using the current values of parameters and static
model variables. The static values of the exogenous and endogenous variables are updated with the
new values calculated using the initval equations. Static endogenous variables are not updated in
argument update_endos = FALSE.

Variables that do not occur at the left hand side of an equation in the initval block are *not* modified.
Thus they are not initialized to zero, in contrast to the evaluation of the initval block by the Dynare
parser.

Usage

DynMdl method:

```
mdl$run_initval(update_endos = TRUE)
```

mdl is a [DynMdl](#) object

Arguments

update_endos A logical. If TRUE, then the static value of the endogenous variables are updated
with the new values computed from the initval equations.

---

set/get_param   [DynMdl](#) *methods: Set and get model parameters*

Description

[DynMdl](#) methods set_param and set_param_values can be used to set the model parameters. Method get_param returns model parameters (excluding sigma parameters used in the fit method), and method get_all_param returns parameters including the sigma parameters.

Methods set_param and set_param_values treat the sigma parameters used in the fit procedure the same as ordinary parameters. However, the recommended methods to set or get sigma parameters are [set_sigma](#), [set_sigma_values](#) and [get_sigmas](#).

Usage

```
mdl$set_param(params, names, name_err = c("stop", "warn", "silent"))

mdl$set_param_values(value, names, pattern)

mdl$get_param(pattern, names)

mdl$get_all_param(pattern, names)
```

mdl is a [DynMdl](#) object

Arguments

params a named numeric vector with parameter values. The names are the names of the parameters.

names a character vector with names of the parameters. For method set_param, this argument *must* be specified if params is a vector without names.

name_err this option specifies the action that should be taken when a specified name is not the name of a model parameter. For "stop" (the default), the execution of this function is stopped. For "warn" and "silent" the names that are no model parameters are skipped. "warn" does, however, give a warning.

pattern a regular expression. The action (get or set parameter values) is applied to all parameters wiht names matching pattern.

If neither names nor pattern has been specified in methods set_param_values or get_param, then the action is applied to all model parameters.

Methods

- set_param: Set the parameters using a named numeric vector. The names of the vector should be the parameter names.
- set_param_values: Give one or more parameter a specified value.
- get_param: Returns the parameters (excluding the sigma parameters used in the fit method)
- get_all_param: Returns parameters including the sigma parameters used in the fit method.

See Also

[get_par_names](), [set_sigma](), [set_sigma_values]() and [get_sigmas]()

Examples

```
mdl <- islm_mdl()

# print parameters c0, c1, c2 and c3
print(mdl$get_param(pattern = "^c.*"))

# set parameter i0 to 101 and c0 to 110
mdl$set_param(c(i0 = 101, c0 = 110))

# set all parameters with names ending with "5" to zero:
mdl$set_param_values(0, pattern = "5$")

# print all parameters
print(mdl$get_param())
```

---

set/get_sigma   [DynMdl]() *methods: Set and get the sigma parameters for the fit method*

Description

DynMdl methods `set_sigma` and `set_sigma_values` can be used to set the sigma parameters for the fit instruments used in the fit method. The names of the sigma parameters are the names of the instruments suffixed with `sigma_`. For example, the name of the sigma parameter corresponding to instrument `"uc"` is `"sigma_uc"`.

If a sigma parameter is smaller than 0, then the corresponding fit instrument is not active, and is keeped fixed at the current value, even though it is an endogenous model variable.

Method `get_sigmas` returns all sigma parameters larger than or equal to zero.

Usage

```
mdl$set_sigma(sigmas, names, name_err = c("stop", "warn", "silent"))

mdl$set_sigma_values(value, names, pattern)

mdl$get_sigma(pattern, names)

mdl$get_sigmas()
```
mdl is a [DynMdl]() object.

Arguments

sigmas a named numeric vector with values of the sigma parameters. The names are the name of the instruments (not the names of the sigma parameters).

names a character vector with names of fit instruments. For method set_sigma, this argument *must* be specified if sigmas is a vector without names.

name_err this option specifies the action that should be taken when a variable name is not a fit instrument. For "stop" (the default), the execution of this function is stopped. For "warn" and "silent", the names that are no fit instrument names are skipped. "warn" does, however, give a warning.

value a numeric vector of length 1.

pattern a regular expression. The action (get or set sigma parameter values) is applied to all sigma parameters wiht names matching pattern.

If neither names nor pattern has been specified in methods set_param_values or get_param, then the action is applied to all model parameters.

Methods

- set_sigmas: Set the sigma parameters using a named numeric vector. The names of the vector should be the names of the corresponding fit instruments.
- set_sigma_values: Give one or more sigma parameters a specified value.
- get_sigma: Return sigma parameters.
- get_sigmas: Returns all sigma parameters greater than or equal to zero.

See Also

get_instrument_names, get_sigma_names, set_fit and clear_fit

Examples

```
mdl <- islm_mdl(fit = TRUE)
mdl$set_sigma(c(umd = 12))

# print the sigma parameter for umd
print(mdl$get_sigma(names = "umd"))

# disable fit instruments umd and umc
mdl$set_sigma_values(-1, names = c("umd", "uc"))

# print all sigma parameters for active fit instruments
print(mdl$get_sigmas())

# print names of all active instruments (sigma >= 0):
print(mdl$get_instrument_names())

# set all sigmas parameters to 1
mdl$set_sigma_values(1)
```

---

set/get_static_endos/exos

> [DynMdl](#) *methods: set and get the static values of the model variables*

Description

set_static_exos, set_static_endos and set_static_data can be used to set one or more static values of the endogenous and/or exogenous model variables. The corresponding get methods can be used to retrieve them.

Each [DynMdl](#) object contains a set of static values for the exogenous and endogenous model variables. The static exogenous values are used to compute the steady state with function methode [solve_steady](#). The static endogenous values are both input and output of solve_steady: they are used as an initial guess for the steady state, and replaced by the steady state solution.

The static values are initialized to the values specified in the initval block of the mod file, or to zero if they are not specified in the initval block. The static values can be modified with methods set_static_endos, set_static_endos, set_static_data or set_static_exo_values.

[get_all_static_endos](#) returns the static values for all endogenous variables, fit instruments and Lagrange multipliers. [get_all_data](#) also returns all static values of exogenous variables. These functions can be useful to save the complete solution of a model that be used as initial values for the endogenous variables in another model.

Usage

DynMdl method:

```
mdl$set_static_endos(endos, names = names(endos), name_err = c("stop", "warn", "silent"))
mdl$set_static_exos(exos, names = names(endos), name_err = c("stop", "warn", "silent"))
mdl$set_static_data(data, names = names(data), name_err = c("stop", "warn", "silent"))
mdl$get_static_endos(pattern, names)
mdl$get_static_endos(pattern, names)
mdl$get_static_data(pattern, names)
mdl$get_all_static_endos()
mdl$get_all_static_data()
```

mdl is a [DynMdl](#) object

Arguments

endos   A (named) numerical vector with new static values of the endogenous variables. If the vector has no names, than argument names must be specified.

exos   A (named) numerical vector with new static values of the exogenous variables. If the vector has no names, than argument names must be specified.

data   A (named) numerical vector with new static values of both endogenous and exogenous variables. If the vector has no names, than argument names must be specified.

names   a character vector with names of model variables

pattern a regular expression

name_err this option specifies the action that should be taken when a variable name is not a model
variable. For ″stop″ (the default), the execution of this function is stopped. For ″warn″ and
″silent″ the names that are no model variables are skipped. ″warn″ does however give a
warning.

Methods

- set_static_endos: Set the static values of one or more endogenous variables (including
  static fit instruments and static Lagrange multipliers).
- set_static_exos: Set the static values of one or more exogenous variables.
- set_static_data: Set the static values of one or more endogenous or exogenous variable
  (including static fit instruments and static Lagrange multipliers.)
- get_static_endos: Returns the static values of one or more endogenous variables, excluding
  fit instruments and Lagrange multipliers.
- get_static_exos: Returns the static values of one or more exogenous variables.
- get_static_data: Returns the static values of the model variables: exogenous and endoge-
  nous model variables and static fit instruments.
- get_all_static_endos: Returns all static endogenous variables, including fit instruments
  and lagrange multipliers.
- get_all_static_data: Returns all static endogenous and exogenous variables, including fit
  instruments and lagrange multipliers.

See Also

[set_static_values-methods](), [change_static_data-methods](), [solve_steady](), [check]()

Examples

```
mdl <- islm_mdl()
mdl$set_static_endos(c(y = 1250))

#  set static values of all exogenous variables starting with m
# (for this model only ″ms″) to zero.
mdl$set_static_exo_values(333, pattern = ″^m″)

print(mdl$get_static_endos())
```

---

set_data   [DynMdl]() *method: transfers data from a timeseries object to the model*
                              data

Description

This method of R6 class [DynMdl]() transfers data from a timeseries object to the model data (both
endogenous and exogenous). If the model implements the fit method, then set_data can also be
used to modify fit instruments and the Lagrange multipliers.

Usage

```
mdl$set_data(data, names = colnames(data),
             upd_mode = c("update", "updval"), fun,
             name_err = c("stop", "warn", "silent"))
```

mdl is a [DynMdl](#) object

Arguments

data a [ts](#) or [regts](#) object. If data has labels, then set_data will also update the labels of the corresponding model variables. If the model has trends, then the timeseries in data should include the trends.

names a character vector with variable names. Defaults to the column names of data. If data does not have column names, then argument names is mandatory

upd_mode the update mode, a character string specifying how the timeseries are updated: "upd" (standard update, default) or "updval" (update only with valid numbers). See details.

fun a function used to update the model data. This should be a function with two arguments. The original model data is passed to the first argument of the function and data to the second argument. See the examples.

name_err this option specifies the action that should be taken when a variable name is not a model variable. For "stop" (the default), the execution of this function is stopped. For "warn" and "silent" the timeseries that are no model variables are skipped. "warn" does however give a warning.

Details

Method set_data transfers data from a timeseries object to the model data. If data is a multivariate timeseries object, then each column is used to update the model variable with the same name as the column name. If data does not have column names, or if the column names do not correspond to the model variable names, then argument names should be specified.

By default, all values in data are used to update the corresponding model variable. Sometimes it is desirable to skip the NA values in data. This can be achieved by selecting "updval" for argument upd_mode. Other non finite numbers (NaN, Inf, and -Inf) are also disregarded for this update mode. The argument upd_mode controls how the timeseries are updated:

"update" Model variables are updated with the timeseries in data

"updval" Model variables are updated with the non NA values in data

See Also

[get_data-methods](#), [set_values-methods](#), [change_data-methods](#) and [put_static_endos](#).

Examples

```
mdl <- islm_mdl(period = "2017Q1/2017Q3")
```

```
# create a multivariate regts object for exogenous variables g and md
exo <- regts(matrix(c(200, 210, 220, 250, 260, 270), ncol = 2),
             start = "2017Q1", names = c("g", "ms"))

# set and print data
mdl$set_data(exo)
print(mdl$get_exo_data())

# create a univariate regts object for exogenous variable ms,
# with a missing value in 2017Q2
ms <- regts(c(255, NA, 273), start = "2017Q1")

# update with update mode updval (ignore NA)
# note that here we have to specify argument names,
# because ms does not have column names
mdl$set_data(ms, names = "ms", upd_mode = "updval")
print(mdl$get_exo_data())

# in the next example, we use argument fun to apply an additive shock to the
# exogenous variables g and ms.
shock <- regts(matrix(c(-5, -10, -15, 3 , 6, 6), ncol = 2),
             start = "2017Q1", names = c("g", "ms"))
mdl$set_data(shock, fun = function(x1, x2) {x1 + x2})

# the statement above can be more concisely written as
mdl$set_data(shock, fun = `+`)
#`+` is a primitive function that adds its two arguments.
```

---

set_fit   [DynMdl](#) *method: transfers data from a timeseries object to the fit tar-*
                           gets.

Description

The method `set_fit` of R6 class [DynMdl](#) transfers data from a timeseries object to the fit targets.

Usage

```
mdl$set_fit(data, names = colnames(data),
            name_err = c("stop", "warn", "silent"))
```

mdl is a [DynMdl](#) object implementing the fit method.

Arguments

data a [ts](#) or [regts](#) timeseries object

names a character vector with variable names, with the same length as the number of timeseries
       in data. Defaults to the column names of data. If data does not have column names, then
       argument names is mandatory

name_err this option specifies the action that should be taken when a variable name is not an endogenous model variable. For "stop" (the default), the execution of this function is stopped. For "warn" and "silent" the timeseries that are no endogenous model variables are skipped. "warn" does however give a warning.

Details

Method set_fit transfers data from a timeseries object to the fit targets. It works similarly as method [set_data](#). If data is a multivariate timeseries object, then each column is used to update the fit target with the same name as the column name. If data does not have column names, or if the column names do not correspond to the model variable names, then argument names should be specified.

If data contains NA values, then the variable is not a fit target for the corresponding periods, which implies that the variable will be calculated according to the equations of the model.

Warning

Method [init_data](#) removes all fit targets.

See Also

[get_fit](#), [set_fit_steady](#), [set_sigma](#), [set_sigma_values](#), [get_sigma](#) and [clear_fit](#)

Examples

```
mdl <- islm_mdl(period = "2016Q1/2017Q3", fit = TRUE)

# create a regts with fit targets
y <- regts(c(1250, 1255, 1260), start = "2016Q1")
t <- regts(c(250, 255), start = "2016Q1")
fit_targets <- cbind(y, t)

# register the fit targets in the DynMdl object
mdl$set_fit(fit_targets)

print(mdl$get_fit())
```

---

set_fit_steady   [DynMdl](#) *method: set fit targets for the steady state*

Description

This method of R6 class [DynMdl](#) sets the static fit targets, i.e. the fit targets used in the steady state calculation.

Usage

```
mdl$set_fit_steady(data, names = names(data),
name_err = c("stop", "warn", "silent"))
```

mdl is a [DynMdl](#) object implementing the fit method.

Arguments

data a named numeric vector with the fit target values. The names coresspond to the names of the endogenous model variables.

names a character vector with variable names, with the same length as the vector `data`. Defaults to the cnames of `data`. If `data` does not have names, then argument `names` is mandatory

name_err this option specifies the action that should be taken when a variable name is not an endogenous model variable. For `"stop"` (the default), the execution of this function is stopped. For `"warn"` and `"silent"` the timeseries that are no endogenous model variables are skipped. `"warn"` does however give a warning.

Details

If `data` contains NA values, then the corresponding model variable is not a fit target.

Warning

Method [init_data](#) removes all fit targets.

See Also

[get_fit_steady](#) and [clear_fit](#)

Examples

```
mdl <- islm_mdl(fit = TRUE)

# create a regts with fit targets
y <- regts(c(1250, 1255, 1260), start = "2016Q1")
t <- regts(c(250, 255), start = "2016Q1")
fit_targets <- c(y = 1250, t = 250)

# register the static fit targets in the DynMdl object
mdl$set_fit_steady(fit_targets)

print(mdl$get_fit_steady())

mdl$solve_steady()

print(mdl$get_static_endos())
```

---

set_fit_values   [DynMdl](#) *method: Sets the values of the fit targets*

Description

This method of R6 class [DynMdl](#) can be used to set the values of the fit targets. See the documentation of function [set_fit](#) for more information about fit targets.

Usage

```
mdl$set_fit_values(value, names, pattern, period = mdl$get_data_period())
```

mdl is a [DynMdl](#) object implementing the fit method.

Arguments

value a numeric vector of length 1 or with the same length as the length of the range of period.

names a character vector with variable names

pattern a regular expression for selecting the names of fit targets.

period a [period_range](#) object or an object that can be coerced to a period_range.

See Also

[set_fit](#) and [clear_fit](#)

Examples

```
mdl <- islm_mdl(period = "2017Q1/2018Q3", fit = TRUE)

# set the values of ms in 2017Q1 and 2017Q2
mdl$set_fit_values(c(190, 195), names = "i", period = "2017Q1/2017Q2")

print(mdl$get_fit())
```

---

set_labels   [DynMdl](#) *method: Sets labels for the model variables.*

Description

This method of R6 class [DynMdl](#) sets labels for the model variables.

Usage

```
mdl$set_labels(labels)
```

mdl is a `DynMdl` object

Arguments

`labels` a named character vector. The names are the names of the model variables

See Also

`get_labels`

Examples

```
mdl <- islm_mdl()
mdl$set_labels(c(c = "Consumption", i = "investments"))
```

---

set_period   `DynMdl` *method: sets the model period*

Description

This method of R6 class `DynMdl` sets the model period. This is the default period used when solving the model.

If the model data has not already been initialized with method `init_data`, then set_period also initializes the model data. In that case the model data period is set to the specified model period extended with a lag and lead period. Model timeseries are initialized with with the static values of the exogenous and endogenous model variables.

If the model data has already been initialized with method `init_data`, then the new model period should be compatible with the model data period. In particular, the new model period extended with a lag and lead period should not contain periods outside the model data period.

Usage

```
mdl$set_period(period)
```

mdl is a DynMdl object

Arguments

period `period_range` object, or an object that can be coerced to `period_range`

Examples

```
mdl <- islm_mdl()
mdl$set_period("2017Q2/2021Q3")
```

---

set_static_values-methods

DynMdl methods: Sets the values of the static model data

Description

This method of R6 class DynMdl can be used to set the static values of the model data.

Usage

```
mdl$set_static_endo_values(value, names, pattern)

mdl$set_static_exo_values(value, names, pattern)
```

mdl is a DynMdl object

Arguments

value a numeric vector of length 1

names a character vector with variable names.

pattern a regular expression for selecting the names of variables.

If neither names nor pattern have been specified, then the static values of all endogenous or exogenous variables are set to the specified value.

Methods

- set_static_endo_values: Endogenous model variables (including fit instruments and Lagrange multipliers for the fit method).
- set_static_exo_values: Exogenous model variables.

See Also

set_static_data and change_static_data-methods and

Examples

```
mdl <- islm_mdl()

# set the static value of ms
mdl$set_static_exo_values(205, names = "ms")

# set the static values for y and yd to 1000
mdl$set_static_endo_values(1000, pattern = "^yd?$")
```

---

set_values-methods   [DynMdl](#) *methods: Sets the values of the model data*

Description

This method of R6 class [DynMdl](#) can be used to set the values of the model data.

Usage

```
mdl$set_endo_values(value, names, pattern, period = mdl$get_data_period())

mdl$set_exo_values(value, names, pattern, period = mdl$get_data_period())
```

mdl is a [DynMdl](#) object

Arguments

value   a numeric vector of length 1 or with the same length as the length of the range of `period`. If the model has trends, then the values should include the trends.

names   a character vector with variable names.

pattern   a regular expression for selecting the names of variables.

period   a [period_range](#) object or an object that can be coerced to a `period_range`.

If neither `names` nor `pattern` have been specified, then all endogenous or exogenous variables are set to the specified value.

Methods

- set_endo_values: Endogenous model variables (including fit instruments and Lagrange multipliers for the fit method).
- set_exo_values: Exogenous model variables.

See Also

[change_data-methods](#) and [set_data](#)

Examples

```
mdl <- islm_mdl(period = "2017Q1/2018Q3")

# set the values of ms in 2017Q1 and 2017Q2
mdl$set_exo_values(c(205, 206), names = "ms", period = "2017Q1/2017Q2")

# set the values for y and yd to 1000 for  the full data period
mdl$set_endo_values(1000, pattern = "^yd?$")
```

---

solve   [DynMdl](#) *method: Solves the model*

Description

This method of R6 class [DynMdl](#) solves the model.

solve does *not* raise an error when the solve was not successful. In that case a warning may be issued. Method [get_solve_status](#) can be used to check whether the solve was successfully terminated or not.

Usage

DynMdl method:

```
 md$solve(control = list(), mode, solver = c("umfpackr", "nleqslv"),
         start = c("current", "previous"), debug_eqs = FALSE,
         homotopy = TRUE, silent = FALSE, backrep = c("period", "total"),
         ...)
```

mdl is a [DynMdl](#) object

Arguments

control A named list of control parameters passed to function [umf_solve_nl](#) or [nleqslv](#), depending on argument solver. If control parameter trace has not been specified, then that parameter is set to TRUE when the model is solved using the stacked time method and FALSE when the model is solved using the backwards method (see also argument mode).

mode A character specifying the solve mode. Possible values are "stacked_time" and "backwards". By default, models with leads are solved with the stacked time method and models without leads are solved backwards.

solver Specifies the solver employed to solve the model: "umfpackr" (sparse linear algebra) or "nleqslv" (dense linear algebra). For large models, the umfpackr solve can be much faster.

start Method used to initialize starting values when solving the model with the backwards method. For "current" (the default) the current values of the endogenous variables are used as starting values. For "previous" the solution of the previous period is used to create starting values (except for the first period when the model is solved). This argument is ignored if the model if solved with the stacked time Newton method

debug_eqs Debug equations (default FALSE). Only used for the internal calculation mode (calc == "internal", see [dyn_mdl](#)). If TRUE then numerical problems in evaluation of mathematical functions or operators such a log are reported.

homotopy A logical. If TRUE (the default), then the homotopy approach is used when directly solving the model fails. Consult the documentation of Dynare for more information about the homotopy aproach.

silent A logical. If TRUE then all output is suppressed. In that case control parameters silent and trace (see argument control) are ignored.

backrep A character specifying the type of iteration report when the model is solved using the backwards mode (see argument mode). If ″period″, then the number of iterations per period is printed. If ″total″, then only the total number of iterations is printed. This argument is ignored if argument silent is TRUE.

... Other arguments passed to the solver function ([umf_solve_nl](#) when the solver is″umfpackr″), Useful arguments for umf_solve_nl are global (select a global strategy) and umf_control (this option can be used to specify the UMFPACK ordering method, see the example below). See the documentation of [umf_solve_nl](#) for more details.

See Also

[residual_check](#), [solve_steady](#) and [get_solve_status](#)

Examples

```
islm <- islm_mdl(period = ″2018Q1/2023Q3″)
islm$set_exo_values(260, period = ″2018q1″, names = ″g″)
islm$solve(control = list(trace = TRUE))

# use the METIS ordering method of UMFACK. This method can handle
# larger matrices than the standard AMD ordering, and is therefore
# suitable for larghe models.
# WARNING: The METIS ordering is not available on Windows.
## Not run:
islm$solve(control = list(trace = TRUE),
           umf_control = list(ordering = ″METIS″))

## End(Not run)
```

---

solve_dynare   [DynMdl](#) *method: Solves the model with Dynare*

Description

Solve the model with Dynare using Matlab or Octave.

Usage

DynMdl method:

```
mdl$solve_dynare(scratch_dir = tempfile(),
                 dynarere_path = NULL, model_options = list(),
                 solve_options = list(tolf = 1e-8, tolx = 1e-8)),
                 initval_type = c(″m″, ″xlsx″),
                 use_octave = Sys.which(″matlab″) == ″″,
                 exit_matlab = FALSE)
```

mdl is a [DynMdl](#) object

Arguments

scratch_dir Directory where the Matlablab and Dynare scripts are created. By default this is a
      temporary directory that is automatically deleted when the R session terminates.

dynare_path Character string specifying the name of the directory of the Dynare installation. On
      Linux it is usually not necessary to the specify this argument. On Windows it is necessary to
      specify the path of the Dynare installation. In you are running R in the CPB environment the
      path to Dynare is set automatically.

model_options Options passed to the `model` command of Dynare. This should be a named list,
      which names corresponding to the Dynare options. Specify a `NULL` value if the option has
      no value. Consult the documentation of Dynare for a list of available options. Example:
      `model_options = list(block = NULL, mfs = 2)`

solve_options Options passed to the `perfect_foresight_solver` command of Dynare. This
      should be a named list, which names corresponding to the Dynare options. Specify a `NULL`
      value if the option has no value. Consult the documentation of Dynare for a list of avail-
      able options. Example: `steady_options = list(tolf = 1e-7, no_homotopy = NULL)`. The
      default options are list(tolf = 1e-8, tolx = 1e-8)

initval_type A character specifying the type of initval file used in the Matlab/Octave job: `"m"`
      for a Matlab file and `"xlsx"` for an xlsx-file.

use_octave A logical. If `TRUE`, then Dynare is envoked with Octave, otherwise Matlab is used. By
      default Matlab is used if available.

exit_matlab A logical specifying if Matlab should immediately exit when the calcultions have
      finished Matlab writes the output to a separate console. If `exit_matlab` is `FALSE` (the default),
      then the R job waits until the user has closed this console, or entered `exit` in the console.
      Otherwise the console is automatically closed at the end of the calculation and all output is
      lost. This argument is ignored if Dynare is run with Octave. Octave does not open a seperate
      console: all output appears in the same console used by R.

Examples

```
## Not run:
 islm <- islm_mdl(period = "2018Q1/2023Q3")
 islm$set_exo_values(260, period = "2018q1", names = "g")
 islm$solve_dynare()

## End(Not run)
```

---

solve_steady   [DynMdl](#) *method: Solves the steady state.*

Description

This method of R6 class [DynMdl](#) solves the steady state.

This function uses the static exogenous and endogenous variables stored in the `DynMdl` object. The
static endogenous variables are used as an initial guess for solving the steady state. After creating a
`DynMdl` object, the static exogenous and endogenous variables are initialized to the values specified

in the initval block of the mod file, or to zero if they are not specified in the initval block. The static variables can be modified with methods `set_static_exos` and `set_static_endos`.

The function `get_static_endos` can be used to retrieve the steady state solution.

`solve_steady` does *not* raise an error when the solve was not successful. In that case a warning may be issued. Method `get_solve_status` can be used to check whether the solve was successfully terminated or not.

Usage

`DynMdl` method:

```
mdl$solve_steady(control = list(), solver = c("umfpackr", "nleqslv"),
                 debug_eqs = FALSE, silent = FALSE, ...)
```

`mdl` is a `DynMdl` object

Arguments

`control` A named list of control parameters passed to function `umf_solve_nl` or `nleqslv`, depending on argument `solver`. If control parameter `trace` has not been specified, then that parameter is set to `TRUE`.

`solver` Specifies the solver employed to solve the model: umfpackr (sparse linear algebra) or nleqslv (dense linear algebra). For large model, the `umfpackr` solve can be much faster.

`debug_eqs` Debug equations (default `FALSE`). Only used for the internal calculation mode (calc == "internal", see `dyn_mdl`). If `TRUE` then numerical problems in evaluation of mathematical functions or operators such a `log` are reported.

`silent` A logical. If `TRUE` then all output is suppressed. In that case control parameters `silent` and `trace` (see argument `control`) are ignored.

`...` Other arguments passed to the solver

See Also

`set_static_endos`, `set_static_exos`, `get_static_endos`, `get_static_exos`, `put_static_endos` and `get_solve_status`

Examples

```
mdl <- islm_mdl(period = "2018Q1/2080Q1")
mdl$solve_steady(control = list(trace = 1))

# print the solution
print(mdl$get_static_endos())

# update the model data with steady state values of endogenous variables
mdl$put_static_endos()
```

---

solve_steady_dynare   DynMdl *method: Solves the steady state with Dynare*

Description

Solve the steady state and calculate the eigenvalues with Dynare using Matlab or Octave.

Usage

DynMdl method:

```
mdl$solve_steady_dynare(scratch_dir = tempfile(), dynare_path = NULL,
                        model_options = list(),
                        solve_options = list(tolf = 1e-8),
                        use_octave = Sys.which("matlab") == "",
                        exit_matlab = FALSE)
```

mdl is a DynMdl object

Arguments

scratch_dir Directory where the Matlablab and Dynare scripts are created. By default this is a
     temporary directory that is automatically deleted when the R session terminates.

dynare_path Character string specifying the name of the directory of the Dynare installation. On
     Linux it is usually not necessary to the specify this argument. On Windows it is necessary to
     specify the path of the Dynare installation. In you are running R in the CPB environment the
     path to Dynare is set automatically.

model_options Options passed to the model command of Dynare. This should be a named list,
     which names corresponding to the Dynare options. Specify a NULL value if the option has
     no value. Consult the documentation of Dynare for a list of available options. Example:
     model_options = list(block = NULL, mfs = 2)

solve_options Options passed to the steady command of Dynare. This should be a named
     list, which names corresponding to the Dynare options. Specify a NULL value if the option
     has no value. Consult the documentation of Dynare for a list of available options. Exam-
     ple: steady_options = list(tolf = 1e-7, no_homotopy = NULL). The default is list(tolf
     = 1e-8)

use_octave A logical. If TRUE, then Dynare is envoked with Octave, otherwise Matlab is used. By
     default Matlab is used if available.

exit_matlab A logical specifying if Matlab should immediately exit when the calcultions have
     finished Matlab writes the output to a separate console. If exit_matlab is FALSE (the default),
     then the R job waits until the user has closed this console, or entered exit in the console.
     Otherwise the console is automatically closed at the end of the calculation and all output is
     lost. This argument is ignored if Dynare is run with Octave. Octave does not open a seperate
     console: all output appears in the same console used by R.

Value

The eigenvalues of the steady state.

Examples

```
## Not run:
islm <- islm_mdl()
islm$solve_steady_dynare(solve_options = list(tolf = 1e-8))

## End(Not run)
```

---

static_residual_check   [DynMdl](#) *method: Calculates the residuals of the equations of the static model*

Description

This method of R6 class [DynMdl](#) calculates the residuals for the static version of the model. The result is a named numeric vector, where the names are the equation numbers.

Usage

```
mdl$static_residual_check(tol, include_fit_eqs = FALSE)
```

mdl is a [DynMdl](#) object

Arguments

tol   the tolerance parameter. If specified, then return value does not include equations whose residuals are smaller than `tol`.

include_all_eqs a logical value (default FALSE). If TRUE, then the all equations, including fit equations and auxiliary equations (if present), are included in the residual check. The auxiliary equations are extra equations created when the model has lags or leads greater than 1 and if dynmdl was called with `max_laglead_1 = TRUE`.

include_fit_eqs a logical value (default FALSE). If TRUE, then fit equations (if present) are included in the residual check. Ignored if `include_all_eqs` is TRUE.

debug_eqs Debug equations (default FALSE). Only used for the internal calculation mode (`calc == "internal"`, see [dyn_mdl](#)). If TRUE then numerical problems in evaluation of mathematical functions or operators such a `log` are reported.

See Also

[residual_check](#)

---

svd_analysis   *Perform an SVD analysis of a jacobian matrix.*

Description

Find linear combinations of rows and columns of the jacobian using an Singular Value Decomposition (SVD) of the jacobian.

Usage

```
svd_analysis(jac, sd_tol = 1e-12, coef_tol = 1e-12)
```

Arguments

jac	a square matrix, for example the matrix returned by [DynMdl] methods [get_static_jacob], [get_jacob] or [get_back_jacob].
sd_tol	singular value tolerance. Singular values smaller than this tolerance are ignored.
coef_tol	coefficient tolerance. The returned singular vector matrices do no include rows for which all elements are smaller than coef_tol.

Value

a list with class svd_analysis, containing the following components

d	a vector with singular values smaller than sd_tol, in decreasing order.
u	a matrix with the left singular vectors corresponding to the singular values d. Rows for which all elements are smaller than coef_tol have been removed. The columns of this matrix can be interpreted as the (near) linear relations between the rows of the jacobian.
v	a matrix with the right singular vectors corresponding to the singular values d. Rows for which all elements are smaller than coef_tol have been removed. The columns of this matrix can be interpreted as the (near) linear relations between the columns of the jacobian.
svd	the result of the SVD decomposition as returned by function [svd].
sd_tol	the value of argument sd_tol
coef_tol	the value of argument coef_tol

See Also

[svd] and function [findLinearCombos] in package caret.

Examples

```
# create a singular matrix with linearly dependend rows
set.seed(123)
x1 <- rnorm(4)
x2 <- rnorm(4)
x3 <- rnorm(4)
mat1 <- rbind(x1, x2, x3, x4 = x2 + x3)
mat1

svd_analysis(mat1)

# function findLinearCombos in package caret is also useful.
# this function resolves linear relations between the columns
# of matrix (therefore we pass the transpose of mat1)
caret::findLinearCombos(t(mat1))

# now example with linearly dependent columns
x1 <- rnorm(4)
x2 <- rnorm(4)
mat2 <- cbind(x1, x2, x3 = x2)
svd_analysis(mat2)
```

---

write_initval_file   *Writes the model data to a Dynare initval file*

Description

This method of R6 class `DynMdl` writes all endogous and exogenous model variables to a so called
"initval file" that can be read by Dynare. An initval_file contains the paths of all model variables.

Arguments

`file`   the name of the initval file

Examples

```
mdl <- islm_mdl("2017Q1/2019Q2")

# write initval file as matlab file
mdl$write_initval_file("dynare_input/islm_initval.xlsx")

# write initval file in xlsx format
mdl$write_initval_file("dynare_input/islm_initval.xlsx")
```

write_mdl

Writes the model to an RDS file

Description

This method of R6 class DynMdl serializes the model object and writes it to an RDS file. The model can be read back by function read_mdl.

Arguments

file   the name of the RDS file

silent   A logical (default FALSE). It TRUE, then no output is written.

See Also

read_mdl

Examples

```
mdl <- islm_mdl("2017Q1/2019Q2")
mdl$write_mdl("islm_mdl.rds")
```

Index