Package ‘dynmdl’

June 21, 2019

Type Package

Title Parse a Dynare model an generate R code

Version 0.8

Author Who wrote it

Maintainer Who to complain to <yourfault@somewhere.net>

Description A Dynare parser.

SystemRequirements GNU make

License GPL (>= 2)

LazyData TRUE

RoxygenNote 6.1.1

LinkingTo Rcpp, BH, Rcereal

Imports Rcpp,
nleqslv,
Matrix,
compiler,
R6,
geigen,
gsubfn,
tools,
stringi,
umfpackr,
numDeriv,
data.table,
openxlsx

Depends 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 (or FitMdl) 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 the endogenous or exogenous model data by applying a function.

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 is an DynMdl object

Arguments

fun a function applied each model timeseries specified with argument names or pattern

names a character vector with variable names

pattern a regular expression

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

... arguments passed to fun

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

Methods

changes_endo_data Changes the endogenous model variables

change_exo_data Changes the exogenous model variables

See Also

get_data-methods, set_data and set_values-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())
```

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.

Usage

DynMdl method:

mdl$check()

mdl is an DynMdl object

See Also

solve_steady and get_eigval

Examples

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

clear_fit — FitMdl *method: removes fit targets and turns off fit instruments.*

Description

This method of R6 class FitMdl removes all fit targets, sets the sigma-parameters of the fit-instruments to -1 and sets all Lagrange multipliers to 0.

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 an FitMdl object '

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 FitMdl 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 an DynMdl object

Usage

```
mdl$copy()
```

mdl is an 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

Usage

```
DynMdl
```

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

get_data Returns the model data

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

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 equations of the model.

copy Returns a deep copy of the DynMdl object

get_solve_status Returns the status of the last model solve attempt

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

Description

Creates a DynMdl object from a mod file. If the mod file contains a fit block, then this function returns a FitMdl object, which is an extension of a DynMdl object.

Usage

```
dyn_mdl(mod_file, period, data, base_period = NULL, calc = c("R",
  "bytecode", "dll", "internal"), fit_mod_file, debug = FALSE, dll_dir,
  max_laglead_1 = FALSE, nostrict = FALSE, fit = TRUE)
```

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 `"R"`, `"bytecode"`, `"dll"` and `"internal"`. 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.

`nostrict` Allows Dynare to issue a warning and continue processing when there are more endogenous variables than equations, an undeclared symbol is assigned in initval or endval, or exogenous variables were declared but not used in the model block.

`fit` a logical. If `TRUE`, then the function returns a `FitMdl` object if a fit block has been found in the mod file. If `FALSE` then this function does not return a `FitMdl` object.

Value

an `DynMdl` object or, if the mod file contains a fit block, a `FitMdl` object.

FitMdl — *An R6 class for a Dynare model with Fit targets*

Description

This class is a subclass of a DynMdl objects. It contains special methods for the fit procedure.

Usage

```
FitMdl
```

Format

R6Class object.

Value

Object of R6Class containing a macro-economic model,

Methods

- `get_endo_names` Returns the names of the endogenous variables of the model (excluding the instruments and Lagrange multipliers used in the fit procedure).
- `get_exo_names` Returns the names of the exogenous variables of the model (excluding the fit control variables).
- `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
- `get_fit` Returns the fit targets used in the fit procedure
- `get_fit_instruments` Returns all non-zero fit instruments used in the fit procedure
- `get_sigmas` Returns all sigma parameters >= 0 (rms values) 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.

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 `FitMdl` 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 procedure. Use method `get_lagrange` to obtain these Lagrange multipliers.

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` is an `DynMdl` object

Arguments

`pattern` a regular expression

`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 `FitMdl` 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).

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 an DynMdl object

See Also

check

---

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

Description

This method of R6 class DynMdl returns a character vector with the model equations (excluding local equations).

Usage

DynMdl method:

```
 md$get_equations(i = NULL)
```

mdl is an 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 FitMdl *methods: get variables used in the fit procedure.*

Description

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

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() # fit targets

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

mdl$get_lagrange(names, period = mdl$get_period())
```

mdl is an FitMdl object

Arguments

pattern a regular expression

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 FitMdl object
mdl$set_fit(fit_targets)

mdl$solve()

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

get_instrument_names/get_sigma_names

FitMdl methods: Retrieve the names of the fit instruments or sigma parameters used in the fit procedure.

Description

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

Usage

```
mdl$get_instrument_names()

mdl$get_sigma_names()
```

mdl is an FitMdl object

See Also

get_fit_instruments

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 an 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 an 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 of the model

Description

Methods get_max_lag and get_max_lead of R6 class DynMdl return the maximum lag and lead, respectively. the maximum a character vector

Usage

DynMdl methods:

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

mdl is an DynMdl object

get_name-methods — *DynMdl methods: Retrieve the names of model variables or parameters*

Description

These methods of R6 class DynMdl return the names of the model variables or parameters

If the DynMdl object is also a FitMdl 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 procedure. 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()
```

mdl is an 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).

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

See Also

get_instrument_names and get_sigma_names

get_param — DynMdl *method: Returns model parameters*

Description

This method of R6 class DynMdl returns model parameters

Usage

```
mdl$get_param(pattern, names)
```

mdl is an DynMdl object

Arguments

pattern a regular expression specifying parameter names

names a character vector with parameter names

See Also

set_param, set_param_values and get_vars_pars

Examples

```
mdl <- islm_mdl()

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

# print parameters c0, c1, c2 and c3
print(mdl$get_param(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 and lead 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.

Usage

```
mdl$get_period()

mdl$get_data_period()

mdl$get_lag_period()

mdl$get_lead_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 is the model has no lags
- get_lead_period: Returns the lead period or NULL is the model has no leads

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 an 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 FitMdl object, then the variables do not include the the auxiliary endogenous and exogenous variables used in the fit procedure.

Usage

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

mdl is an 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 the model data.

This method sets the model data period and initializes the model variables with static values of the exogenous and endogenous model variables.

This methods also sets the model period, the standard period for which the model will be 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 period_range object, or an object that can be coerced to period_range,

data a ts or regts object with values for endogogenous and exogenous model variables. If data has labels, then these labels are used to update the model labels.

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 have been specified, then the data period is determined from the model period (which in that case must have been specified before init_data is called).

Examples

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

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 or a FitMdl object is argument fit is TRUE

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 an 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 or FitMdl object.

Usage

```
read_mdl(file, dll_dir)
```

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`).

Value

a `DynMdl` or `FitMdl` 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 an `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_fit_eqs` a logical value (default `FALSE`). This argument is only used if `mdl` is a `FitMdl` object. If `TRUE`, then the fit equations are included in the residual check.

`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

---

set/get_static_endos/exos

DynMdl methods: set and get the static values of the model variables

---

Description

set_static_exos, set_static_exo_values and set_static_endos can be used to set one or more static values of the endogenous or exogenous model variables, respectively. The correspondig 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

Usage

```
DynMdl method:

mdl$set_static_endos(endos)
mdl$set_static_exos(exos)
mdl$set_static_exo_values(value, names, pattern)
mdl$get_static_endos()
mdl$get_static_endos()
mdl$get_static_endos()

mdl is an DynMdl object
```

Arguments

endos A named numerical vector with new static values of the endogenous variables

exos A named numerical vector with new static values of the exogenous variables

value a numeric vector of length 1

names a character vector with names of model variables

pattern a regular expression

See Also

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)

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

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 and change_data-methods

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 *FitMdl method: transfers data from a timeseries object to the fit targets.*

Description

The method set_fit of R6 class FitMdl 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 an FitMdl object

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.

See Also

get_fit, set_fit 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_values *FitMdl 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 an FitMdl object

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

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 an 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_param — DynMdl *method: Sets the model parameters*

Description

This method of R6 class DynMdl sets the model parameters

Usage

```
mdl$set_param(p)
```

mdl is an DynMdl object

Arguments

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

See Also

set_param_values and get_param

Examples

```
mdl <- islm_mdl()
mdl$set_param(c(i0 = 101))
```

---

set_param_values *DynMdl methods: Sets the values of the model parameters*

Description

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

Usage

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

mdl is an DynMdl object

Arguments

value	a numeric vector of length 1
names	a character vector with parameter names
pattern	a regular expression

If neither names nor pattern have been specified, then all model parameters are set to the specified value.

See Also

set_param and get_param

Examples

```
mdl <- islm_mdl()

# set parameters i4 and i5 to zero
mdl$set_param_values(0, names = c("i4", "c5"))

# set the values all parameters starting with "i"
# (i0, i1, i2, i3, i4 and i5) to 0
mdl$set_param_values(0, pattern = "^i")

# set all parameters to zero
mdl$set_param_values(0)
```

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_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 an DynMdl object

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

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
- 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_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:

```
md$solve(control = list(), force_stacked_time = FALSE,
         solver = c("umfpackr", "nleqslv"),
         start = c("current", "previous"), ...)
```

mdl is an DynMdl object

Arguments

control A named list of control parameters passed to function umf_solve_nl or nleqslv, depending on argument solver

force_stacked_time a logical. If TRUE, the the model is solved using the stacked time Newton method, also for purely backward looking models.

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.

start Method used to initialize starting values when solving the model backwards. 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.

... Other arguments passed to the solver

See Also

solve_steady and get_solve_status

Examples

```
mdl <- islm_mdl(period = "2018Q1/2023Q3")
mdl$solve(control = list(trace = TRUE))
```

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, solver = c("umfpackr", "nleqslv"), ...)
```

mdl is an DynMdl object

Arguments

control A named list of control parameters passed to function umf_solve_nl or nleqslv, depending on argument solver

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.

... 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()
```

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 an DynMdl object

Arguments

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

include_fit_eqs a logical value (default FALSE). This argument is only used if mdl is a FitMdl object. If TRUE, then the fit equations are included in the residual check.

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 xlsx file

Examples

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

See Also

read_mdl

Examples

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

Index