Package ‘sfcr’

October 14, 2022

Title Simulate Stock-Flow Consistent Models

Version 0.2.1

Description Routines to write, simulate, and validate stock-flow consistent (SFC) models. The accounting structure of SFC models are described in Godley and Lavoie (2007, ISBN:978-1-137-08599-3). The algorithms implemented to solve the models (Gauss-Seidel and Broyden) are described in Kinsella and O'Shea (2010) <doi:10.2139/ssrn.1729205> and Peressini and Sullivan (1988, ISBN:0-387-96614-5).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

URL `https://github.com/joaomacalos/sfcr`

BugReports `https://github.com/joaomacalos/sfcr/issues`

Imports dplyr (>= 1.0.2), expm (>= 0.999.5), forcats (>= 0.5.0), igraph (>= 1.2.6), kableExtra (>= 1.3.1), magrittr (>= 1.5), purrr (>= 0.3.4), Rdpack (>= 2.1), rootSolve (>= 1.8.2.1), rlang (>= 0.4.7), tibble (>= 3.0.3), tidyr (>= 1.1.2), tidyselect (>= 1.1.0), stringr (>= 1.4.0), utils, vctrs (>= 0.3.5), Rcpp (>= 1.0.5)

Suggests ggraph, ggplot2, grDevices, knitr, pkgdown, rmarkdown, RColorBrewer, testthat (>= 2.3.2), tidygraph, tidyverse, networkD3

VignetteBuilder knitr

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Joao Macalos [aut, cre] (<`https://orcid.org/0000-0001-6050-6394`>)

Maintainer Joao Macalos <`joaomacalos@gmail.com`>

Repository CRAN

Date/Publication 2021-10-11 08:00:02 UTC

R topics documented:

`sfcr_baseline` *Simulate the baseline scenario of a stock-flow consistent model*

Description

The `sfcr_baseline()` function is used to simulate a SFC model.

Usage

```
sfcr_baseline(
  equations,
  external,
  periods,
  initial = NULL,
  hidden = NULL,
  max_iter = 350,
  .hidden_tol = 0.1,
  tol = 1e-08,
  method = "Broyden",
  rhtol = FALSE,
  ...
)
```

Arguments

`equations`	A `sfcr_set` containing all the equations of the model to be simulated. The equations must be written with the R formula syntax, with the left-hand side separated from the right-hand side by a twiddle `~`.
`external, initial`	A `sfcr_set` of external variables (exogenous and parameters) or of initial values. They should be written as equations using the R syntax.
`periods`	A number specifying the total number of periods of the model to be simulated.
`hidden`	Named object that identify the two variables that make the hidden equality in the SFC model, e.g., `c("H_h" = "H_s")`. Defaults to NULL. If `hidden` is supplied, the model will evaluate if the hidden equation is satisfied.
`max_iter`	Maximum iterations allowed per period.
`.hidden_tol`	Error tolerance to accept the equality of the hidden equation. Defaults to 1. In growth models, computational errors might buildup in the hidden equation, which renders any absolute comparison inadequate. For such models, please turn `rhtol` to `TRUE`, and set the value of `.hidden_tol` accordingly. See details for further information.
`tol`	Tolerance accepted to determine convergence.
`method`	The method to use to find a solution. Defaults to "Broyden".
`rhtol`	A logical argument that defines whether the a relative measure is used to evaluate the hidden equation or not. Defaults to `FALSE`, i.e., a absolute measure is used.
`...`	Extra arguments to pass to `rootSolve::multiroot()` function if "Newton" method is selected.

Details

The output of a `sfcr_baseline()` is a `sfcr_tbl`. The only difference between a `sfcr_tbl` and a standard `tbl_df` is that the former has two extra attributes: `matrix` and `call`. The `matrix` attribute, for example, can be accessed by calling `attributes(sfcr_sim_object)$matrix`. It is possible to see, in the matrix, the number of iterations required to calculate each block of equations in the model. The `call` attribute shows the blocks of equations and preserve the call that are used internally.

The `equations`, `exogenous`, and `parameters` arguments must be written with the R formula syntax, i.e., the left-hand side of each item is separated to the right-hand side by a twiddle. Variables that represent lags of endogenous or exogenous variables must be followed by `[-1]`. See examples for details on the syntax.

Before solving the system of equations, two consecutive depth-first searches identify and order the blocks of independent equations in the system. The system is then solved sequentially, i.e., the variables that depend only on lagged or exogenous values are evaluated first, and then the variables that depends on these variables, etc. The solving algorithms are only applied to the blocks of mutually dependent equations. The great `igraph` package is used to implement the two consecutive depth-first searches.

- Methods:

The `sfcr` package provides three algorithms to solve the blocks of cyclical equations: the Gauss-Seidel algorithm, the Broyden algorithm, and the Newton-Raphson algorithm. The default method is "Broyden" as it tends to be fastest one.

See (Kinsella and OShea 2010) for details on the Gauss-Seidel algorithm and (Peressini et al. 1988) for details on the Broyden and Newton-Raphson algorithms.

The "Broyden" algorithm uses the `rootSolve::jacobian.full()` function to get the initial Jacobian matrix, and compiled code from `RcppArmadillo` to invert the jacobians. See also https://www.math.usm.edu/lambers/ma

The Gauss Seidel algorithm is implemented as described by (Kinsella and OShea 2010). Finally, the "Newton" method uses the `rootSolve::multiroot()` function to solve the system.

- Hidden equation:

One of the defining aspects of a SFC model is its water tight accounting. One way to check whether the model was correctly defined is to see if the hidden (redundant) equation is satisfied after the model is simulated. In stationary models, an absolute comparison should suffice as the model converges to a stationary state. However, growth models converge to a stable growth rate where stocks are perpetually increasing. It is inadequate to use a absolute comparison in such models. In these cases, the `rhtol` argument ("relative hidden tolerance") must be set to `TRUE` in order to perform a relative comparison. The relative comparison evaluates the numerical discrepancy in the hidden equation as a ratio of one of its elements. For example, if `hidden = c("Bbs" = "Bbd")`, the hidden equation will be evaluated according to the following steps:

1. `d = (Bbs - Bbd)`
2. `isTRUE(d/Bbs < .hidden_tol)`

In general, the `.hidden_tol` argument should be set to a small number (e.g. 1e-6). The function will check that this proportion remains the same for all simulated periods.

Value

A `sfcr_tbl`.

Author(s)

João Macalós, <joaomacalos@gmail.com>

References

Kinsella S, OShea T (2010). “Solution and Simulation of Large Stock Flow Consistent Monetary Production Models via the Gauss Seidel Algorithm.” *SSRN Electronic Journal*. doi: 10.2139/ssrn.1729205, `https://doi.org/10.2139/ssrn.1729205`. Peressini AL, Sullivan FE, Uhl JJ (1988). *The Mathematics of Nonlinear Programming*. Springer-Verlag, Berlin, Heidelberg. ISBN 0387966145.

Examples

```
eqs <- sfcr_set(
  TXs ~ TXd,
  YD ~ W * Ns - TXs,
  Cd ~ alpha1 * YD + alpha2 * Hh[-1],
  Hh ~ YD - Cd + Hh[-1],
  Ns ~ Nd,
  Nd ~ Y / W,
  Cs ~ Cd,
  Gs ~ Gd,
  Y ~ Cs + Gs,
  TXd ~ theta * W * Ns,
  Hs ~ Gd - TXd + Hs[-1]
)

external <- sfcr_set(Gd ~ 20, W ~ 1, alpha1 ~ 0.6, alpha2 ~ 0.4, theta ~ 0.2)

# Periods is set to 10 to run faster. A usual model should run at
# least 50 periods to find a steady state
sfcr_baseline(equations = eqs, external = external, periods = 10)
```

sfcr_dag_blocks — *Create a* tbl_graph *object blocks and cycles information*

Description

Create a tbl_graph object blocks and cycles information

Usage

```
sfcr_dag_blocks(equations)
```

Arguments

equations	A sfcr_set containing all the equations of the model to be simulated. The equations must be written with the R formula syntax, with the left-hand side separated from the right-hand side by a twiddle ~.

Details

This function creates a tbl_graph with information about the blocks and cycles attached to it. This object can then be used to plot the DAG of the model.

Value

A tbl_graph

Author(s)

João Macalós, <joaomacalos@gmail.com>

`sfcr_dag_blocks_plot` *Plot the DAG with blocks and cycles information*

Description

Plot the DAG with blocks and cycles information

Usage

```
sfcr_dag_blocks_plot(equations, title = NULL, size = 10)
```

Arguments

`equations`	A `sfcr_set` containing all the equations of the model to be simulated. The equations must be written with the R formula syntax, with the left-hand side separated from the right-hand side by a twiddle `~`.
`title`	Title of the plot.
`size`	Size of the points.

Details

This function creates a `tbl_graph` with information about the cycles attached to it. This object can then be used to plot the DAG of the model.

Value

A `tbl_graph`

Author(s)

João Macalós, <`joaomacalos@gmail.com`>

`sfcr_dag_cycles` *Create a* `tbl_graph` *object with cycles information*

Description

Create a `tbl_graph` object with cycles information

Usage

```
sfcr_dag_cycles(equations)
```

Arguments

`equations`	A `sfcr_set` containing all the equations of the model to be simulated. The equations must be written with the R formula syntax, with the left-hand side separated from the right-hand side by a twiddle ~.

Details

This function creates a `tbl_graph` with information about the cycles attached to it. This object can then be used to plot the DAG of the model.

Value

A `tbl_graph`

Author(s)

João Macalós, <`joaomacalos@gmail.com`>

`sfcr_dag_cycles_plot` *Plot the DAG with cycles information*

Description

Plot the DAG with cycles information

Usage

```
sfcr_dag_cycles_plot(equations, title = NULL, size = 10)
```

Arguments

`equations`	A `sfcr_set` containing all the equations of the model to be simulated. The equations must be written with the R formula syntax, with the left-hand side separated from the right-hand side by a twiddle ~.
`title`	Title of the plot.
`size`	Size of the points.

Author(s)

João Macalós

sfcr_expand *Expand variables to implement sensitivity analysis*

Description

The sfcr_expand() function is a s3 **generic** that takes a list of external objects and returns a expanded set of these lists. It has methods for sfcr_set objects and for sfcr_shock objects.

Usage

```
sfcr_expand(x, variable, values)
```

Arguments

x	A external set created with sfcr_set() or a shock set created with sfcr_shock()
variable	the name of variable to be expanded.
values	a vector containing the new values of the external or shock variable.

Details

There are two available methods for the sfcr_expand() function:

- sfcr_set: Takes a sfcr_set object with **external** variables and creates a list of sets that inherits all the aspects of the x set supplied but set the values of the variable to the each element of value. The output is a sfcr_mlt_set object.
- sfcr_shock: Takes a sfcr_shock object and creates a list of shocks that inherits all the aspects of the x shock but set the values of the variable to each element of value. The output of this method is a sfcr_mlt_shock object.

Author(s)

João Macalós

Examples

```
# 1. Expand a external set:
external <- sfcr_set(G_d ~ 20, W ~ 1, alpha1 ~ 0.6, alpha2 ~ 0.4, theta ~ 0.2)
sfcr_expand(external, alpha2, c(0.1, 0.2))

# 2. Expand a shock:
shock <- sfcr_shock(variables = sfcr_set(alpha1 ~ 0.8), start = 5, end = 50)
sfcr_expand(shock, alpha1, c(0.7, 0.8, 0.9))
```

`sfcr_get_blocks` *Get block structure of a* `sfcr_tbl` *object*

Description

Get block structure of a `sfcr_tbl` object

Usage

```
sfcr_get_blocks(sfcr_tbl)
```

Arguments

`sfcr_tbl`	A `sfcr_tbl` object.

Author(s)

João Macalós

`sfcr_get_matrix` *Get Matrix form of* `sfcr_tbl` *object*

Description

Get Matrix form of `sfcr_tbl` object

Usage

```
sfcr_get_matrix(sfcr_tbl)
```

Arguments

`sfcr_tbl`	A `sfcr_tbl` object.

Author(s)

João Macalós

`sfcr_matrix` *Create balance-sheet or transactions-flow matrices*

Description

Create balance-sheet or transactions-flow matrices

Usage

```
sfcr_matrix(columns, codes, ...)
```

Arguments

`columns`	Vector containing the name of the columns in the matrix.
`codes`	A vector containing the abbreviation of the column names that is going to be used as a reference to build the rows. They must be provided in the same order as the `columns`.
`...`	Vectors that fill the rows of the matrix. The first element of each vector **must** be the name of the row in the respective matrix. The remaining elements of the vector must be name-value pairs that exactly matches the `codes` argument. See the examples for further details.

Note

This function can be used to generate a transactions- flow matrix as well as a balance-sheet matrix. If the user wishes to validate these matrices with the simulated data, please pay attention to the following details:

- Transactions-flow Matrix: In the transactions-flow matrix, the `sum` column is going to be generated automatically by the validation function. Please do not add it by hand.
- Balance-sheet Matrix: In the balance-sheet matrix, it might be the case that some rows do not sum to zero. Therefore, the user must supply by hand the non-zero values of the `sum` column. This column should always be the last column of the matrix and should always be named as "Sum". If there's no column named as "Sum", it will be generated automatically by the validation function with all entries equal to zero.

Author(s)

João Macalós, <joaomacalos@gmail.com>

Examples

```
# Balance-sheet matrix

bs_pc <- sfcr_matrix(
  columns = c("Households", "Firms", "Government", "Central bank", "sum"),
  codes = c("h", "f", "g", "cb", "s"),
  r1 = c("Money", h = "+Hh", cb = "-Hs"),
  r2 = c("Bills", h = "+Bh", g = "-Bs", cb = "+Bcb"),
  r3 = c("Balance", h = "-V", g = "+V")
)


# Transactions-flow matrix
tfm_pc <- sfcr_matrix(
  columns = c("Households", "Firms", "Government", "CB current", "CB capital"),
  codes = c("h", "f", "g", "cbc", "cbk"),
  c("Consumption", h = "-C", f = "+C"),
  c("Govt. Expenditures", f = "+G", g = "-G"),
  c("Income", h = "+Y", f = "-Y"),
 c("Int. payments", h = "+r[-1] * Bh[-1]", g = "-r[-1] * Bs[-1]", cbc = "+r[-1] * Bcb[-1]"),
  c("CB profits", g = "+r[-1] * Bcb[-1]", cbc = "-r[-1] * Bcb[-1]"),
  c("Taxes", h = "-TX", g = "+TX"),
  c("Ch. Money", h = "-(Hh - Hh[-1])", cbk = "+(Hs - Hs[-1])"),
  c("Ch. Bills", h = "-(Bh - Bh[-1])", g = "+(Bs - Bs[-1])", cbk = "-(Bcb - Bcb[-1])")
)
```

`sfcr_matrix_display` *Print matrix to screen*

Description

Print matrix to screen

Usage

```
sfcr_matrix_display(matrix, which = "tfm")
```

Arguments

`matrix`	A balance sheet or transactions-flow matrix
`which`	A character string for the matrix. Is it a balance-sheet or a transactions-flow matrix? here are two options: `"bs"` for balance-sheet matrices, and `"tfm"` for transactions- flow matrices. The default is `"tfm"`.

Details

This function takes a matrix as input and generate a `kableExtra` table with math symbols displayed in latex style.

Note

This function converts the math expressions used to build the `sfcr_matrix` into a latex format, but cannot add modifications to it. The user is invited to explore the source code and the `kableExtra` package in order to personalize his/her own matrices.

Author(s)

João Macalós

Examples

```
# Balance-sheet matrix

bs_insout <- sfcr_matrix(
  columns = c("Households", "Firms", "Government", "Central bank", "Banks", "Sum"),
  codes = c("h", "f", "g", "cb", "b", "s"),
  r1 = c("Inventories", f = "+INV", s = "+INV"),
  r2 = c("HPM", h = "+Hhd", cb = "-Hs", b = "+Hbd"),
  r3 = c("Advances", cb = "+As", b = "-Ad"),
  r4 = c("Checking deposits", h = "+M1h", b = "-M1s"),
  r5 = c("Time deposits", h = "+M2h", b = "-M2s"),
  r6 = c("Bills", h = "+Bhh", g = "-Bs", cb = "+Bcb", b = "+Bbd"),
  r7 = c("Bonds", h = "+BLh * pbl", g = "-BLs * pbl"),
  r8 = c("Loans", f = "-Ld", b = "+Ls"),
  r9 = c("Balance", h = "-V", f = 0, g = "+GD", cb = 0, b = 0, s = "-INV")
)

sfcr_matrix_display(bs_insout, "bs")
```

`sfcr_multis` *Simulate multiple SFC models at the same time*

Description

The `sfcr_multis()` function is used to simulate multiple models at the same time, returning a list of `sfcr_tbl`s.

Usage

```
sfcr_multis(expanded, fixed, periods, ...)
```

Arguments

`expanded`	A `sfcr_mlt_set`, `sfcr_mlt_shock`, or a `sfcr_mlt` object.
`fixed`	A `sfcr_set`, `sfcr_tbl`, or `sfcr_shock` object.
`periods`	A number specifying the total number of periods of the model to be simulated.
`...`	Additional arguments to pass to the underlying implementation of the `sfcr_baseline()` or `sfcr_scenario()` functions.

Details

The `sfcr_multis()` function takes an `expanded` object and a `fixed` to simulate multiple models that will share the content of `fixed` but vary on the `expanded`.

This function is a **generic**, which means that its implementation depends on the class of the `expanded` argument.

The available methods for the `sfcr_multis()` function depends on the `expanded` argument. There are three possible methods:

- `sfcr_mlt_set`: When the `sfcr_multis()` takes an `sfcr_mlt_set` class as the input of `expanded`, it must take a list of equations of the `sfcr_set` class as the `fixed` input. This method simulates many baseline models that accept the same set of equations and vary on the external variables supplied with the `expanded` argument.
- `sfcr_mlt_shock`: When the `sfcr_multis()` takes an `sfcr_mlt_shock` class as the input of `expanded`, it must also take an object of `sfcr_tbl` class as the input of `fixed`. It will simulate multiple scenario models that takes the same baseline model and diverge on the content of the multiple shocks provided with the `expanded` argument that are applied to it.
- `sfcr_mlt`: When the `sfcr_multis()` function takes a `sfcr_mlt` class object as the input of the `expanded` argument, a `sfcr_shock` object must be supplied with the `fixed` argument. This method simulates multiple scenario models that applies the same shock to a varying number of baseline models.

Author(s)

João Macalós

Examples

```
eqs <- sfcr_set(
  TX_s ~ TX_d,
  YD ~ W * N_s - TX_s,
  C_d ~ alpha1 * YD + alpha2 * H_h[-1],
  H_h ~ YD - C_d + H_h[-1],
  N_s ~ N_d,
  N_d ~ Y / W,
  C_s ~ C_d,
  G_s ~ G_d,
  Y ~ C_s + G_s,
  TX_d ~ theta * W * N_s,
  H_s ~ G_d - TX_d + H_s[-1]
)

external <- sfcr_set(G_d ~ 20, W ~ 1, alpha1 ~ 0.6, alpha2 ~ 0.4, theta ~ 0.2)

shock <- sfcr_shock(
  variables = sfcr_set(
    alpha2 ~ 0.3
  ),
  start = 1,
  end = 3
)

baseline <- sfcr_baseline(eqs, external, periods = 5)

# Example 1: Many external sets, 1 set of equations:
expanded1 <- sfcr_expand(external, alpha1, c(0.7, 0.8))
multis1 <- sfcr_multis(expanded = expanded1, fixed = eqs, periods = 5)

# Example 2: Many shocks, 1 baseline model:
expanded2 <- sfcr_expand(shock, alpha2, c(0.1, 0.2))
multis2 <- sfcr_multis(expanded = expanded2, fixed = baseline, periods = 5)

# Example 3: Many baseline models, 1 shock:
multis3 <- sfcr_multis(expanded = multis1, fixed = shock, periods = 5)
```

`sfcr_portfolio` *Find a valid matrix of portfolio parameters*

Description

The `sfcr_portfolio()` function calculates a valid matrix of portfolio parameters by applying the symmetry condition and then filling the missing rows accordingly to the vertical and horizontal adding-up constraints.

Usage

```
sfcr_portfolio(m, known)
```

Arguments

`m`	A matrix of parameter names
`known`	A named vector of known parameters. One entry for each symmetry condition is enough to find a valid matrix.

Details

This function calculates only the values of the rates of return matrix, i.e., the internal matrix. The adding-up constraint number 1, that calculates the share of assets in the net wealth and the impact of regular income to wealth ratio must be calculated separately.

If supplied with insufficient parameters, the function will return a matrix with NA values.

This function requires at least (n^2 - n)/2 known parameters to find a valid portfolio matrix, where n is the number of rows/columns. This is achieved by setting known parameters outside the diagonal and not on symmetrical entries, i.e., not lambda12 and lambda21, for example.

Author(s)

João Macalós

Examples

```
j1 <- matrix(paste0("lambda", c(11:14, 21:24, 31:34, 41:44)), ncol = 4, nrow = 4, byrow = TRUE)
j2 <- c(lambda12 = 0, lambda13 = 0, lambda14 = 0, lambda23 = -15, lambda24 = -15, lambda34 = -15)

sfcr_portfolio(j1, j2)
```

`sfcr_random` *Generate random sequences inside* `sfcr_set()`

Description

This function can only be used inside `sfcr_set()` when generating variables. It smartly guesses the length of the `sfcr_baseline()` model or of the `sfcr_shock()` that it is inserted.

Usage

```
sfcr_random(.f, ...)
```

Arguments

`.f`	This argument accepts three options: "rnorm", "rbinom", and "runif", and implement the respective functions from the built-in `stats` package.
`...`	Extra arguments to be passed to the `stats` generator functions

Author(s)

João Macalós

Examples

```
# Create a random normal series to pass along an endogenous series
# Example taken from model PC EXT 2.
sfcr_set(
    Ra ~ sfcr_random("rnorm", mean=0, sd=0.05)
)
```

`sfcr_sankey` *Plot Sankey's diagram representation of transactions-flow matrix*

Description

Plot Sankey's diagram representation of transactions-flow matrix

Usage

```
sfcr_sankey(tfm, baseline, when = "start")
```

Arguments

`tfm`	A transactions-flow matrix
`baseline`	A baseline model
`when`	When the Sankey's diagram should be evaluated? • "start": Fifth and fourth periods. • "end": last two periods of the simulation (stationary state).

Author(s)

João Macalós

`sfcr_scenario` *Add scenarios to a* `sfcr` *model.*

Description

Add scenarios to a `sfcr` model.

Usage

```
sfcr_scenario(
  baseline,
  scenario,
  periods,
  max_iter = 350,
  tol = 1e-10,
  method = "Broyden",
  ...
)
```

Arguments

`baseline`	A model generated with the `sfcr_baseline()` function.
`scenario`	Either a shock created with `sfcr_shock()`, a list of shocks, or `NULL`. If `scenario = NULL`, the model will just extend the baseline model.
`periods`	A number specifying the total number of periods of the model to be simulated.
`max_iter`	Maximum iterations allowed per period.
`tol`	Tolerance accepted to determine convergence.
`method`	The method to use to find a solution. Defaults to "Broyden".
`...`	Extra arguments to pass to `rootSolve::multiroot()` function if "Newton" method is selected.

Details

Add scenario(s) to a model generated with `sfcr_baseline()` functions.

This function inherits the block structure from the steady state model. See `sfcr_baseline` for further details on the algorithms.

Author(s)

João Macalós, <`joaomacalos@gmail.com`>

See Also

`sfcr_baseline`

Examples

```
eqs <- sfcr_set(
  TX_s ~ TX_d,
  YD ~ W * N_s - TX_s,
  C_d ~ alpha1 * YD + alpha2 * H_h[-1],
  H_h ~ YD - C_d + H_h[-1],
  N_s ~ N_d,
  N_d ~ Y / W,
  C_s ~ C_d,
  G_s ~ G_d,
  Y ~ C_s + G_s,
  TX_d ~ theta * W * N_s,
  H_s ~ G_d - TX_d + H_s[-1]
)

external <- sfcr_set(G_d ~ 20, W ~ 1, alpha1 ~ 0.6, alpha2 ~ 0.4, theta ~ 0.2)

# t is set to 10 to run faster. A usual model should run at least 50 periods to find a steady state
steady_state <- sfcr_baseline(eqs, external, periods = 10)

# Increase G_d from 20 to 30 between periods 5 and 10
shock1 <- sfcr_shock(sfcr_set(G_d ~ 30), 5, 10)
```

```
sfcr_scenario(steady_state, scenario = list(shock1), 10)

# Increase W to 2, alpha2 to 0.5, and decrease theta to 0.15
shock2 <- sfcr_shock(
  variables = sfcr_set(
  W ~ 2,
  alpha2 ~ 0.5,
  theta ~ 0.15
  ),
  start = 5,
  end = 10)

sfcr_scenario(steady_state, list(shock2), 10)
```

sfcr_set *Define the formulas of the model*

Description

The sfcr_set() function is used to create the lists of equations, external variables, initial values, and also to modify the variables inside the sfcr_shock() function.

Usage

```
sfcr_set(..., exclude = NULL)
```

Arguments

...	The formulas used to define the equations and external values of the system
exclude	One or more indices of equations to be excluded. The correct indices can be found with sfcr_set_index().

Details

This function is a S3 generic that applicable to only two inputs: formula and sfcr_set. It is used to create a new set of equations or to modify an existing one.

Therefore, the equations must be written using the R formula syntax, i.e., the left-hand side of each equation is separated from the right-hand side with a ~ ("twiddle") instead of a =.

Furthermore, the sfcr_set() function recognizes two symbols that are not native to R language: [-1], and d().

- If a variable defined with sfcr_set() is followed by [-1], it will be recognized as a lagged variable.
- If a variable is defined inside d(), the sfcr engines will transform them into a first difference equation. For example, d(Hh) is internally transformed into (Hh - Hh[-1]).

Random variables can be created using the `sfcr_random()` function. See `sfcr_random` for further details.

Author(s)

João Macalós

Examples

```
# Endogenous set
equations <- sfcr_set(
  TXs ~ TXd,
  YD ~ W * Ns - TXs,
  Cd ~ alpha1 * YD + alpha2 * Hh[-1],
  Hh ~ YD - Cd + Hh[-1],
  Ns ~ Nd,
  Nd ~ Y / W,
  Cs ~ Cd,
  Gs ~ Gd,
  Y ~ Cs + Gs,
  TXd ~ theta * W * Ns,
  Hs ~ Gd - TXd + Hs[-1]
  )

# Exogenous set
exogenous <- sfcr_set(alpha1 ~ 0.8, alpha2 ~ 0.15)

# Modify an existing set
equations2 <- sfcr_set(equations, Hh ~ Hh[-1] + d(Hs), exclude = 4)

# Add normal random variable
sfcr_set(Ra ~ sfcr_random("rnorm", mean=10, sd=2))
```

`sfcr_set_index` *Get names of endogenous vars and their index*

Description

The `sfcr_set_index()` function takes a list of equations as its input and returns a tibble containing the name of the variable on the left-hand side of the equations and their position in the equations list.

Usage

```
sfcr_set_index(eqs)
```

Arguments

`eqs`	A list of equations created with `sfcr_set()`

Details

This function aims to facilitate locating a specific equation in the list in order to modify the list of equations.

To add random variation to endogenous variables, use `sfcr_random()`.

Author(s)

João Macalós

`sfcr_shock` *Create shock(s) to add to a* `sfcr_scenario()`.

Description

Create shock(s) to add to a `sfcr_scenario()`.

Usage

```
sfcr_shock(variables, start, end)
```

Arguments

`variables`	A `sfcr_set()` with formula(e) containing the name of the variable(s) that will be shocked on the left-hand side and their new values on the right- hand side. It is possible to add exogenous series a shock instead of constant variables. However, the length of such series must be exactly the same as the period of the shock (i.e., the difference between start and end).
`start`	An integer indicating the period when the shock takes place.
`end`	An integer indicating the period when the shock ends.

Author(s)

João Macalós, <`joaomacalos@gmail.com`>

Examples

```
sfcr_shock(
 variables = sfcr_set(G_d ~ 30, W ~ 1.5),
 start = 5,
 end = 66)

sfcr_shock(
 variables = sfcr_set(G_d ~ seq(30, 40, length.out=62)),
 start = 5,
 end = 66)
```

`sfcr_validate` *Validate a transactions-flow or balance-sheet matrix*

Description

This function validates a transactions-flow or balance-sheet matrix with the simulated data obtained with `sfcr_baseline()` function

Usage

```
sfcr_validate(matrix, baseline, which, tol = 1, rtol = FALSE)
```

Arguments

`matrix`	A transactions-flow or balance sheet matrix
`baseline`	A baseline model.
`which`	Either "bs" (balance-sheet matrix) or "tfm" (transactions-flow matrix).
`tol`	A numerical value indicating the absolute accepted discrepancy accepted to validate whether the rows and columns are equal to their expected values.
`rtol`	A logical value indicating whether relative discrepancies should be evaluated. It defaults to `FALSE`. Stationary models should pass the test using a absolute level while growth models might need a relative validation since computational discrepancies tend to get larger with the model. See details for further information.

Details

The relative discrepancy is calculated differently if we are dealing with a transactions-flow matrix or with a balance-sheet matrix. If `which` is set to `tfm`, the sum of the row/column is evaluated against the sum of the positive entries of that row/column.

For example, in a transactions-flow matrix with three entries in the "change in the stock of bills" row (-Delta (Bhd), + Delta (Bs), and + Delta (Bbd)), the discrepancy d = Delta Bs - Delta Bhd - Delta Bbd is evaluated against Delta Bs, i.e., the row is validated if d/Delta Bs < tol.

In a balance-sheet matrix, all the rows/columns that sum to zero are validated exactly as in a transactions-flow matrix. The exception to this rule is when there is a expected value. In this case, the discrepancy is evaluated as a proportion of the expected. value

To prevent unnecessary calculations, a absolute check with tolerance defined as 1e-3 is executed prior to this evaluation.

The absolute discrepancy set with `tol` should be enough to validate a stationary SFC Model.

Author(s)

João Macalós

Index