Package 'readsdr'

October 14, 2022

Type Package

Title Translate Models from System Dynamics Software into 'R'

Version 0.2.0

Description The goal of 'readsdr' is to bridge the design capabilities from specialised System Dynamics software with the powerful numerical tools offered by 'R' libraries. The package accomplishes this goal by parsing 'XMILE' files ('Vensim' and 'Stella') models into 'R' objects to construct networks (graph theory); 'ODE' functions for 'Stan'; and inputs to simulate via 'deSolve' as described in Duggan (2016) <doi:10.1007/978-3-319-34043-2>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Suggests testthat (>= 2.1.0), igraph, knitr, rmarkdown, ggplot2, tidyr

Imports stringr, xml2, purrr, dplyr, rlang, stringi, magrittr, stats, deSolve, parallel

BugReports https://github.com/jandraor/readsdr/issues

VignetteBuilder knitr

NeedsCompilation no

Author Jair Andrade [aut, cre] (<https://orcid.org/0000-0002-1412-7868>)

Maintainer Jair Andrade <jair.albert.andrade@gmail.com>

Repository CRAN

Date/Publication 2021-01-08 16:00:02 UTC

R topics documented:

`create_stan_function` *Create a Stan's ODE function from an XMILE file*

Description

`create_stan_function` returns a string with the code for a Stan's ODE function

Usage

```
create_stan_function(
  filepath,
  func_name,
  pars = NULL,
  override.consts = NULL,
  additional_funs = NULL
)
```

Arguments

`filepath`	A string that indicates a path to a file with extension .stmx or .xmile. Vensim files (.mdl) are not xmile files. They must be exported from Vensim with extension .xmile
`func_name`	A string for naming the ODE function
`pars`	A character vector that indicates which constants will be considered as parameters in the ODE function
`override.consts`	A list in which each element is a name-value pair that replaces values of constants.
`additional_funs`	A vector of strings. Each string corresponds to a user-defined function.

Details

This function extracts the xml from the file specified via `filepath` to generate the code for an equivalent model in Stan.

Value

A string with the code containing the model's equations in the format required by Stan.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
create_stan_function(path, "my_model")
```

`extract_timeseries_stock` *Extract the values over time of a stock from a Stan fit*

Description

Extract the values over time of a stock from a Stan fit

Usage

```
extract_timeseries_stock(stock_name, posterior_df, all_stocks, ODE_output)
```

Arguments

`stock_name`	A string that indicates the stock's name for which the function will construct the timeseries.
`posterior_df`	A Stan fit object converted into a data frame
`all_stocks`	A vector of strings that contains the names of all the stocks in the model. This vector must have the same order as the differential equations in the Stan code.
`ODE_output`	A string that indicates the name of the variable where model's output in stored in Stan.

Value

A data frame

Examples

```
posterior_df <- data.frame(`yhat[1,2]` = rep(0, 2), `yhat[2,2]` = rep(1, 2),
                           check.names = FALSE)
stocks       <- c("S1", "S2")
extract_timeseries_stock("S2", posterior_df, stocks, "yhat")
```

extract_timeseries_var *Extract the values over time of a variable from a Stan fit*

Description

Extract the values over time of a variable from a Stan fit

Usage

```
extract_timeseries_var(var_name, posterior_df)
```

Arguments

var_name	A string that indicates the variable's name for which the function will construct the timeseries.
posterior_df	A Stan fit object converted into a data frame

Value

A data frame

Examples

```
posterior_df <- data.frame(`var[1]` = rep(0, 2), `var[2]` = rep(1, 2),
                           check.names = FALSE)
extract_timeseries_var("var", posterior_df)
```

read_xmile *Read an XMILE file into R*

Description

read_xmile returns a list for constructing deSolve functions and graphs

Usage

```
read_xmile(filepath, stock_list = NULL, const_list = NULL)
```

Arguments

filepath	A string that indicates a path to a file with extension .stmx or .xmile. Vensim files (.mdl) are not xmile files. They must be exported from Vensim with extension .xmile
stock_list	A list in which each element's name is the name of the stock to override and the element's value correspond to the new init value.
const_list	A list in which each element's name is the name of the constant to override and the element's value correspond to the new value.

Details

This function extracts the xml from the file specified via `filepath` to generate a list of objects. Such a list contains a summary of the model, the inputs for simulating through deSolve, and the inputs for creating a igraph object.

Value

This function returns a list with three elements. The first element, *description*, is a list that contains the simulation parameters, and the names and equations (including graphical functions) for each stock or level, variable and constant. The second element, *deSolve_components*, is a list that contains initial values, constants and the function for simulating via deSolve. The third element, *igraph* contains the data.frames for creating a graph with igraph.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
read_xmile(path)
```

---

sd_constants *Summarise the information of a model's constants in a data frame*

Description

Summarise the information of a model's constants in a data frame

Usage

```
sd_constants(mdl)
```

Arguments

`mdl`	A list which is the output from read_xmile.

Value

A data frame.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
mdl  <- read_xmile(path)
sd_constants(mdl)
```

sd_pulse_s *Replicate the behaviour of the PULSE function from Stella*

Description

This function must be placed inside the object that will be passed as the argument `func` to deSolve's `ode` function.

Usage

```
sd_pulse_s(time, volume, start_p, interval)
```

Arguments

`time`	A number
`volume`	A number
`start_p`	A number
`interval`	A number

Value

A number

Examples

```
timestep <- function() 0.25 # replicates timestep() from deSolve
sd_pulse_s(2, 1, 2, 0)
```

sd_pulse_train *PULSE TRAIN*

Description

PULSE TRAIN

Usage

```
sd_pulse_train(time, start_pulse, duration_pulse, repeat_pt, end_pulse)
```

Arguments

`time`	A numeric argument that indicates the current simulation time
`start_pulse`	A numeric argument that indicates the start of the pulse
`duration_pulse`	A numeric argument that indicates the width of the pulse
`repeat_pt`	A numeric argument that indicates the repetition pattern
`end_pulse`	A numeric argument that indicates the end of the sequence

Value

1 during the pulse, 0 otherwise.

Examples

```
sd_pulse_train(5, 5, 3, 10, 20)
```

`sd_pulse_v` *Replicate the behaviour of the PULSE function from Vensim*

Description

Replicate the behaviour of the PULSE function from Vensim

Usage

```
sd_pulse_v(time, startPulse, duration)
```

Arguments

`time`	A number
`startPulse`	A number
`duration`	A number

Value

A number

Examples

```
timestep <- function() 0.25 # replicates timestep() from deSolve
sd_pulse_v(1, 1, 2)
```

`sd_sensitivity_run` *Perform a sensitivity run on a System Dynamics model*

Description

`sd_sensitivity_run` returns a data frame with the simulation of a model for several iterations of different inputs.

Usage

```
sd_sensitivity_run(
  ds_inputs,
  consts_df = NULL,
  stocks_df = NULL,
  start_time = NULL,
  stop_time = NULL,
  timestep = NULL,
  integ_method = "euler",
  multicore = FALSE,
  n_cores = NULL
)
```

Arguments

ds_inputs	A list of deSolve inputs generated by read_xmile
consts_df	A data frame that contains the values of constants to simulate. Each column corresponds to a constant and each row to an iteration.
stocks_df	A data frame that containts the initial value of stocks to be explored. Each column corresponds to a stock and each row to an iteration.
start_time	A number
stop_time	A number
timestep	A number
integ_method	A string. Either "euler" or "rk4"
multicore	A boolean value that indicates whether the process is parallelised. This option only works for Unix-based systems.
n_cores	An integer.

Value

A data frame

Examples

```
path      <- system.file("models", "SIR.stmx", package = "readsdr")
ds_inputs <- xmile_to_deSolve(path)
consts_df <- data.frame(i = c(0.25, 0.30))
sd_sensitivity_run(ds_inputs, consts_df)
```

sd_simulate *Simulate a System Dynamics model*

Description

Simulate a System Dynamics model

Usage

```
sd_simulate(
  ds_inputs,
  start_time = NULL,
  stop_time = NULL,
  timestep = NULL,
  integ_method = "euler"
)
```

Arguments

ds_inputs	A list of deSolve inputs generated by read_xmile
start_time	A number
stop_time	A number
timestep	A number
integ_method	A string. Either "euler" or "rk4"

Value

a data frame

Examples

```
path      <- system.file("models", "SIR.stmx", package = "readsdr")
ds_inputs <- xmile_to_deSolve(path)
sd_simulate(ds_inputs, 0, 1, 0.25, "rk4")
```

sd_stocks *Summarise the information of a model's stocks in a data frame*

Description

Summarise the information of a model's stocks in a data frame

Usage

```
sd_stocks(mdl)
```

Arguments

mdl	A list which is the output from read_xmile.

Value

A data frame.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
mdl  <- read_xmile(path)
sd_stocks(mdl)
```

---

stan_data *Stan's data block for ODE models*

Description

Stan's data block for ODE models

Usage

```
stan_data(vars_vector, type, inits = TRUE)
```

Arguments

vars_vector	a string vector. Each element corresponds to a vector's name for which users will supply data.
type	a string vector. It must have the same length as vars_vector . This parameter indicates the type of the variables declared by vars_vector.
inits	a boolean. Indicates whether the block includes the declaration for stocks' init values.

Value

a string that contains the Stan code for the data block.

Examples

```
stan_data("y", "int")
stan_data("y", "real", FALSE)
```

stan_ode_function *Create Stan ODE function*

Description

Create Stan ODE function

Usage

```
stan_ode_function(
  filepath,
  func_name,
  pars = NULL,
  const_list = NULL,
  extra_funs = NULL
)
```

Arguments

filepath	A string that indicates a path to a file with extension .stmx or .xmile. Vensim files (.mdl) are not xmile files. They must be exported from Vensim with extension .xmile
func_name	A string for naming the ODE function
pars	A character vector that indicates which constants will be considered as parameters in the ODE function
const_list	A list in which each element's name is the name of the constant to override and the element's value correspond to the new value.
extra_funs	A vector of strings. Each string corresponds to a user-defined function.

Value

A string with the code containing a function with the model's equations in the format required by cmdstan 2.24+.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
stan_ode_function(path, "my_model")
```

stan_transformed_data *Stan's transformed data block for ODE models*

Description

Stan's transformed data block for ODE models

Usage

```
stan_transformed_data()
```

Value

a string

Examples

```
td <- stan_transformed_data()
```

xmile_to_deSolve *Parse XMILE to deSolve components*

Description

`xmile_to_deSolve` returns a list that serves as an input for deSolve's ODE function.

Usage

```
xmile_to_deSolve(filepath)
```

Arguments

`filepath`	A string that indicates a path to a file with extension .stmx or .xmile. Vensim files (.mdl) are not xmile files. They must be exported from Vensim with extension .xmile

Details

This function extracts the xml from the file specified via `filepath` to generate a list with the necessary elements to simulate with deSolve.

Value

This function returns a list with at least four elements. *stocks*, a numeric vector that contains initial values. *consts*, a numeric vector with the model's constants. *func*, the function that wraps the model's equations. *sim_params*, a list with control parameters. If the model includes a table or graphical function, this function returns the element *graph_funs*, a list with these functions.

Examples

```
path <- system.file("models", "SIR.stmx", package = "readsdr")
xmile_to_deSolve(path)
```

Index