Package 'MCSimMod'

April 10, 2025

Title Working with 'MCSim' Models

Version 0.9

Description Tools that facilitate ordinary differential equation (ODE) modeling in 'R'. This package allows one to perform simulations for ODE models that are encoded in the GNU 'MCSim' model specification language (Bois, 2009) <doi:10.1093/bioinformatics/btp162> using ODE solvers from the 'R' package 'deSolve' (Soetaert et al., 2010) <doi:10.18637/jss.v033.i09>.

Depends methods, tools

Imports deSolve

URL https://CRAN.R-project.org/package=MCSimMod, https://github.com/USEPA/MCSimMod

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/Needs/dev devtools, styler (== 1.10.3), testthat, covr

Config/Needs/website r-lib/pkgdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Dustin F. Kapraun [aut, cre] (<https://orcid.org/0000-0001-5570-6383>), Todd J. Zurlinden [aut] (<https://orcid.org/0000-0003-1372-3913>), Andrew J. Shapiro [aut] (<https://orcid.org/0000-0002-5233-8092>), Ryan D. Friese [aut] (<https://orcid.org/0000-0002-4121-2195>), Frederic Y. Bois [ctb] (<https://orcid.org/0000-0002-4154-0391>), Free Software Foundation, Inc. [cph]

Maintainer Dustin F. Kapraun <kapraun.dustin@epa.gov>

Repository CRAN

Date/Publication 2025-04-10 14:40:06 UTC

Contents

`compileModel` *Function to translate and compile MCSim model specification text*

Description

This function translates MCSim model specification text to C and then compiles the resulting C file to create a dynamic link library (DLL) file (on Windows) or a shared object (SO) file (on Unix).

Usage

```
compileModel(model_file, c_file, dll_name, dll_file, hash_file = NULL)
```

Arguments

`model_file`	Name of an MCSim model specification file.
`c_file`	Name of a C source code file to be created by compiling the MCSim model specification file.
`dll_name`	Name of a DLL or SO file without the extension (".dll" or ".so").
`dll_file`	Name of the same DLL or SO file with the appropriate extension (".dll" or ".so").
`hash_file`	Name of a file containing a hash key for determining if `model_file` has changed since the previous translation and compilation.

Value

No return value. Creates files and saves them in locations specified by function arguments.

`createModel` *Function to create an MCSimMod Model object*

Description

This function creates a `Model` object using an MCSim model specification file or an MCSim model specification string.

Usage

```
createModel(mName = character(0), mString = character(0), writeTemp = TRUE)
```

Arguments

mName	Name of an MCSim model specification file, excluding the file name extension .model.
mString	A character string containing MCSim model specification text.
writeTemp	Boolean specifying whether to write model files to a temporary directory. If value is TRUE (the default), model files will be Written to a temporary directory; if value is FALSE, model files will be Written to the same directory that contains the model specification file.

Value

Model object.

Examples

```
## Not run:
# Simple model
mod <- createModel("path/to/model")

# Load/compile the model
mod$loadModel()

# Update parameters (P1 and P2)
mod$updateParms(c(P1 = 3, P2 = 1))

# Define times for ODE simulation
times <- seq(from = 0, to = 24, by = 0.1)

# Run the simulation
out <- mod$runModel(times)

## End(Not run)
```

Model-class *MCSimMod Model class*

Description

A class for managing MCSimMod models.

Arguments

mName	Name of an MCSim model specification file, excluding the file name extension .model.
mString	A character string containing MCSim model specification text.

Details

Instances of this class represent ordinary differential equation (ODE) models. A `Model` object has both attributes (i.e., things the object "knows" about itself) and methods (i.e., things the object can "do"). Model attributes include: the name of the model (`mName`); a vector of parameter names and values (`parms`); and a vector of initial conditions (`Y0`). Model methods include functions for: translating, compiling, and loading the model (`loadModel`); updating parameter values (`updateParms`); updating initial conditions (`updateY0`); and running model simulations (`runModel`). So, for example, if `mod` is a Model object, it will have an attribute called `parms` that can be accessed using the R expression `mod$parms`. Similarly, `mod` will have a method called `updateParms` that can be accessed using the R expression `mod$updateParms()`. Use the `createModel()` function to create `Model` objects.

Fields

`mName` Name of an MCSim model specification file, excluding the file name extension `.model`.

`mString` Character string containing MCSim model specification text.

`initParms` Function that initializes values of parameters defined for the associated MCSim model.

`initStates` Function that initializes values of state variables defined for teh associated MCSim model..

`Outputs` Names of output variables defined for the associated MCSim model.

`parms` Named vector of parameter values for the associated MCSim model.

`Y0` Named vector of initial conditions for the state variables of the associated MCSim model.

`paths` List of character strings that are names of files associated with the model.

`writeTemp` Boolean specifying whether to write model files to a temporary directory. If value is TRUE, model files will be Written to a temporary directory; if value is FALSE, model files will be Written to the same directory that contains the model specification file.

Methods

`cleanup(deleteModel = FALSE)` Delete files created during the translation and compilation steps performed by `loadModel`. If `deleteModel = TRUE`, delete the MCSim model specification file, as well.

`initialize(...)` Initialize the Model object using an MCSim model specification file (mName) or an MCSim model specification string (mString).

`loadModel(force = FALSE)` Translate (if necessary) the model specification text to C, compile (if necessary) the resulting C file to create a dynamic link library (DLL) file (on Windows) or a shared object (SO) file (on Unix), and then load all essential information about the Model object into memory (for use in the current R session).

`runModel(times, ...)` Perform a simulation for the Model object using the `deSolve` function `ode` for the specified `times`.

`updateParms(new_parms = NULL)` Update values of parameters for the Model object.

`updateY0(new_states = NULL)` Update values of initital conditions of state variables for the Model object.

Index