Package ‘campsismod’

February 7, 2025

Type Package

Title Generic Implementation of a PK/PD Model

Version 1.2.0

Description A generic, easy-to-use and expandable implementation of a pharmacokinetic (PK) / pharmacodynamic (PD) model based on the S4 class system. This package allows the user to read/write a pharmacometric model from/to files and adapt it further on the fly in the R environment. For this purpose, this package provides an intuitive API to add, modify or delete equations, ordinary differential equations (ODE's), model parameters or compartment properties (like infusion duration or rate, bioavailability and initial values). Finally, this package also provides a useful export of the model for use with simulation packages 'rxode2' and 'mrgsolve'. This package is designed and intended to be used with package 'campsis', a PK/PD simulation platform built on top of 'rxode2' and 'mrgsolve'.

License GPL (>= 3)

URL `https://github.com/Calvagone/campsismod`, `https://calvagone.github.io/`, `https://calvagone.github.io/campsismod.doc/`

BugReports `https://github.com/Calvagone/campsismod/issues`

Depends R (>= 4.0.0)

Imports assertthat, dplyr, LaplacesDemon, magrittr, MASS, methods, purrr, readr, rlang, tibble, utils

Suggests devtools, knitr, pkgdown, rmarkdown, roxygen2, testthat, xfun

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Collate 'global.R' 'data.R' 'utilities.R' 'special_operators.R' 'check.R' 'generic.R' 'generic_element_list.R' 'generic_element_position.R' 'generic_list.R' 'pattern.R' 'model_statement.R' 'model_statements.R' 'model_unknown_statement.R' 'model_comment.R' 'model_line_break.R' 'model_equation.R' 'model_ode.R' 'model_if_statement.R' 'compartment.R' 'compartment_property.R' 'compartment_properties.R' 'compartment_bioavailability.R' 'compartment_lag_time.R' 'compartment_infusion_duration.R' 'compartment_infusion_rate.R' 'compartment_initial_condition.R' 'compartments.R' 'parameter.R' 'parameters.R' 'code_record.R' 'code_records.R' 'model_parser.R' 'campsis_model.R' 'omega_block.R' 'omega_blocks.R' 'parameter_uncertainty_utils.R' 'replication_settings.R' 'replicated_campsis_model.R' 'model_add_suffix.R' 'rxode_model.R' 'rxode_conversion.R' 'mrgsolve_model.R' 'mrgsolve_conversion.R'

NeedsCompilation no

Author Nicolas Luyckx [aut, cre]

Maintainer Nicolas Luyckx <`nicolas.luyckx@calvagone.com`>

Repository CRAN

Date/Publication 2025-02-07 14:40:02 UTC

Contents

add *Add element to list.*

Description

Add element to list.

Usage

```
add(object, x, ...)

## S4 method for signature 'pmx_list,pmx_element'
add(object, x, pos = NULL)

## S4 method for signature 'pmx_list,pmx_list'
add(object, x)

## S4 method for signature 'pmx_list,list'
add(object, x)

## S4 method for signature 'compartments,compartment_property'
add(object, x)

## S4 method for signature 'compartments,compartments'
add(object, x)

## S4 method for signature 'parameters,single_array_parameter'
add(object, x)

## S4 method for signature 'parameters,double_array_parameter'
add(object, x)

## S4 method for signature 'parameters,parameters'
add(object, x)

## S4 method for signature 'code_record,model_statement'
add(object, x, pos = NULL)

## S4 method for signature 'code_record,code_record'
add(object, x)

## S4 method for signature 'code_records,code_records'
add(object, x)

## S4 method for signature 'code_records,model_statement'
add(object, x, pos = NULL)
```

```
## S4 method for signature 'campsis_model,compartment_property'
add(object, x)

## S4 method for signature 'campsis_model,parameter'
add(object, x)

## S4 method for signature 'campsis_model,code_record'
add(object, x)

## S4 method for signature 'campsis_model,model_statement'
add(object, x, pos = NULL)

## S4 method for signature 'campsis_model,campsis_model'
add(object, x)

## S4 method for signature 'omega_block,double_array_parameter'
add(object, x)

## S4 method for signature 'omega_blocks,omega_block'
add(object, x)

## S4 method for signature 'omega_blocks,parameters'
add(object, x)
```

Arguments

`object`	list object
`x`	element to add
`...`	extra arguments, unused by this generic list
`pos`	position where x needs to be added in list

Value

modified list object

`addRSE` *Add relative standard error (RSE) to the specified parameter.*

Description

Add relative standard error (RSE) to the specified parameter.

Usage

```
addRSE(object, parameter, value, ...)

## S4 method for signature 'parameters,parameter,numeric'
addRSE(object, parameter, value, ...)

## S4 method for signature 'campsis_model,parameter,numeric'
addRSE(object, parameter, value, ...)
```

Arguments

`object`	model or parameters object
`parameter`	parameter object (Theta, Omega or Sigma)
`value`	RSE value, in percent
`...`	extra arguments, unused

Value

updated object

`addSuffix` *Generic function to add a suffix to various objects like parameters, code records, compartment names or a model (all previous objects at the same time). This makes it an extremely powerful function to combine 2 models or more (using function 'add'), that have similar equation, parameter or compartment names.*

Description

Generic function to add a suffix to various objects like parameters, code records, compartment names or a model (all previous objects at the same time). This makes it an extremely powerful function to combine 2 models or more (using function 'add'), that have similar equation, parameter or compartment names.

Usage

```
addSuffix(object, suffix, separator = NULL, ...)

## S4 method for signature 'parameters,character,character'
addSuffix(object, suffix, separator = NULL, ...)

## S4 method for signature 'code_records,character,character'
addSuffix(object, suffix, separator = NULL, ...)

## S4 method for signature 'code_record,character,character'
addSuffix(object, suffix, separator = NULL, ...)
```

```
## S4 method for signature 'compartments,character,character'
addSuffix(object, suffix, separator = NULL, ...)

## S4 method for signature 'campsis_model,character,character'
addSuffix(object, suffix, separator = NULL, ...)
```

Arguments

`object`	generic object
`suffix`	suffix to be appended, single character value
`separator`	separator to use before the suffix, default is the underscore
`...`	extra arguments like 'model' if the changes need to be reflected in the model

Value

updated object of the same class as the provided object, unless 'model' was specified, in that case the model is returned

---

`as.data.frame` *As data frame method.*

Description

As data frame method.

Usage

```
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'theta,character,logical'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'omega,character,logical'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature 'sigma,character,logical'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	generic object
`row.names`	row names
`optional`	optional
`...`	extra arguments

Value

data frame

`assertSingleCharacterString` *Assert the given character vector is a single character string.*

Description

Assert the given character vector is a single character string.

Usage

```
assertSingleCharacterString(x)
```

Arguments

`x`	single character string

Value

no return value

`autoDetectNONMEM` *Auto-detect special variables from NONMEM as compartment properties. Bioavailabilities, infusion durations/rates and lag times will be automatically detected.*

Description

Auto-detect special variables from NONMEM as compartment properties. Bioavailabilities, infusion durations/rates and lag times will be automatically detected.

Usage

```
autoDetectNONMEM(object, ...)

## S4 method for signature 'campsis_model'
autoDetectNONMEM(object, ...)
```

Arguments

`object`	object that has NONMEM special variables to be identified
`...`	extra arguments, unused

Value

updated object

`AutoReplicationSettings`

Create auto replication settings.

Description

By default, all model parameters are sampled from a multivariate normal distribution, whose characteristics are specified by the variance-covariance matrix. OMEGAs and SIGMAs can be sampled from scaled inverse chi-squared or Wishart distributions by setting the `wishart` argument to `TRUE`. In that case, THETAs are still sampled from a multivariate normal distribution, while OMEGAS and SIGMAs are sampled from scaled inverse chi-squared (univariate OMEGA/SIGMA distribution) and Wishart (block of OMEGAs/SIGMAs) distribution, respectively. When `wishart` is set to `TRUE`, the degrees of freedom of the distribution must be specified, respectively, `odf` for the OMEGAs and `sdf` for the SIGMAs.

Usage

```
AutoReplicationSettings(
  wishart = FALSE,
  odf = NA,
  sdf = NA,
  checkMinMax = TRUE,
  checkPosDef = FALSE,
  quiet = NA
)
```

Arguments

`wishart`	logical, sample OMEGAs and SIGMAs from scaled inverse chi-squared (univariate OMEGA distribution) or Wishart distribution (block of OMEGAs)
`odf`	the degrees of freedom for the scaled inverse chi-squared/Wishart distribution with regards to the OMEGAs
`sdf`	the degrees of freedom for the scaled inverse chi-squared/Wishart distribution with regards to the SIGMAs
`checkMinMax`	logical, check for min/max values when sampling the parameters, default is TRUE
`checkPosDef`	logical, check for positive definiteness when sampling the OMEGA/SIGMA parameters from the variance-covariance matrix (i.e. when `wishart=FALSE`), default is FALSE (requires extra time)
`quiet`	logical, suppress info messages, default is NA. By default, messages will be printed out when the success rate of sampling the parameters is below 95%.

Value

replication settings

`auto_replication_settings-class` *Auto replication settings class.*

Description

Auto replication settings class.

Slots

- `wishart` logical, sample OMEGAs and SIGMAs from scaled inverse chi-squared or Wishart distributions
- `odf` the degrees of freedom for the scaled inverse chi-squared/Wishart distribution with regards to the OMEGAs
- `sdf` the degrees of freedom for the scaled inverse chi-squared/Wishart distribution with regards to the SIGMAs
- `quiet` logical, suppress info messages
- `max_iterations` number of iterations maximum to sample the parameters
- `max_chunk_size` maximum number of rows to sample at once, default value will be the number of replicates, unless specified.
- `check_min_max` logical, check for min/max values when sampling the parameters
- `check_pos_def` logical, check for positive definiteness when sampling the OMEGA/SIGMA parameters

---

`Bioavailability` *Create a bioavailability for the specified compartment.*

Description

Create a bioavailability for the specified compartment.

Usage

```
Bioavailability(compartment, rhs = "")
```

Arguments

`compartment`	compartment index
`rhs`	right-hand side part of the equation

Value

a bioavailability property

`CampsisModel` *Create a new Campsis model.*

Description

Create a new Campsis model.

Usage

```
CampsisModel()
```

Value

a Campsis model, empty

`campsis_model-class` *Campsis model class.*

Description

Campsis model class.

Slots

`model` a list of code records

`parameters` model parameters

`compartments` model compartments

`CodeRecords` *Create a list of code records.*

Description

Create a list of code records.

Usage

```
CodeRecords()
```

Value

an empty list of code records

`code_record-class` *Code record class. See this code record as an abstract class. 2 implementations are possible: - properties record (lag, duration, rate & bioavailability properties) - statements record (main, ode & error records)*

Description

Code record class. See this code record as an abstract class. 2 implementations are possible: - properties record (lag, duration, rate & bioavailability properties) - statements record (main, ode & error records)

Slots

`comment` a comment, single character value

`statements` model statements

`Comment` *Create a new comment.*

Description

Create a new comment.

Usage

```
Comment(x)
```

Arguments

`x`	comment, single character string

Value

a comment

`comment-class` *Comment class. A statement starting with #.*

Description

Comment class. A statement starting with #.

`Compartment` *Create a compartment.*

Description

Create a compartment.

Usage

```
Compartment(index, name = NA)
```

Arguments

`index`	compartment index
`name`	compartment name (without prefix)

Value

an empty list of compartments

`compartment-class` *Compartment class.*

Description

Compartment class.

Slots

`name` compartment name (without prefix)

`index` compartment index

`Compartments` *Create a list of compartments*

Description

Create a list of compartments

Usage

```
Compartments()
```

Value

an empty list of compartments

`compartments-class` *Compartments class.*

Description

Compartments class.

Slots

`properties` compartment properties of the compartments defined in this class

---

`compartment_bioavailability-class` *Compartment bioavailability class.*

Description

Compartment bioavailability class.

`compartment_infusion_duration-class` *Compartment infusion duration class.*

Description

Compartment infusion duration class.

`compartment_infusion_rate-class` *Compartment infusion rate class.*

Description

Compartment infusion rate class.

`compartment_initial_condition-class`

Compartment initial condition class.

Description

Compartment initial condition class.

`compartment_lag_time-class`

Compartment lag time class.

Description

Compartment lag time class.

`compartment_properties-class`

Compartment properties class.

Description

Compartment properties class.

`compartment_property-class`

Compartment property class.

Description

Compartment property class.

Slots

`compartment` related compartment index

`rhs` right-hand side formula

`comment` comment if any, single character string

`default` *Get default element from list.*

Description

Get default element from list.

Usage

```
default(object, ...)
```

Arguments

`object`	list object
`...`	additional arguments

Value

the default element from list

`delete` *Delete an element from this list.*

Description

Delete an element from this list.

Usage

```
delete(object, x)

## S4 method for signature 'pmx_list,pmx_element'
delete(object, x)

## S4 method for signature 'pmx_list,integer'
delete(object, x)

## S4 method for signature 'compartments,compartment_property'
delete(object, x)

## S4 method for signature 'parameters,single_array_parameter'
delete(object, x)

## S4 method for signature 'parameters,double_array_parameter'
delete(object, x)
```

```
## S4 method for signature 'statements_record,model_statement'
delete(object, x)

## S4 method for signature 'statements_record,integer'
delete(object, x)

## S4 method for signature 'code_records,model_statement'
delete(object, x)

## S4 method for signature 'campsis_model,compartment_property'
delete(object, x)

## S4 method for signature 'campsis_model,parameter'
delete(object, x)

## S4 method for signature 'campsis_model,code_record'
delete(object, x)

## S4 method for signature 'campsis_model,model_statement'
delete(object, x)
```

Arguments

`object`	list object
`x`	element to delete or element index

Value

the updated list

`disable` *Disable.*

Description

Disable.

Usage

```
disable(object, x, ...)

## S4 method for signature 'parameters,character'
disable(object, x, ...)

## S4 method for signature 'campsis_model,character'
disable(object, x, ...)
```

Arguments

`object`	generic object
`x`	what needs to be disabled
`...`	extra arguments needed for disabling

Value

object with some disabled features

`double_array_parameter-class` *Double-array parameter class. This parameter has 2 indexes. It can thus be used to define correlations between parameters.*

Description

Double-array parameter class. This parameter has 2 indexes. It can thus be used to define correlations between parameters.

`duration_record-class` *(Infusion)-duration record class.*

Description

(Infusion)-duration record class.

`Equation` *Create a new equation.*

Description

Create a new equation.

Usage

```
Equation(lhs, rhs = "", comment = as.character(NA))
```

Arguments

`lhs`	left-hand side variable corresponding to the assigned variable name
`rhs`	right-hand side expression corresponding to a formula
`comment`	comment if any, single character string

Value

an equation

`equation-class` *Equation class. Any statement in the form A = B.*

Description

Equation class. Any statement in the form A = B.

Slots

`lhs` left-hand side expression

`rhs` right-hand side expression

`ErrorRecord` *Create ERROR code record.*

Description

Create ERROR code record.

Usage

```
ErrorRecord(code = character())
```

Arguments

`code`	code record

Value

an ERROR code record

`error_record-class` *Error record class.*

Description

Error record class.

`export` *Export function.*

Description

Export function.

Usage

```
export(object, dest, ...)

## S4 method for signature 'campsis_model,character'
export(object, dest, ...)

## S4 method for signature 'replicated_campsis_model,campsis_model'
export(object, dest = CampsisModel(), index, ...)

## S4 method for signature 'campsis_model,rxode_type'
export(object, dest, ...)

## S4 method for signature 'campsis_model,mrgsolve_type'
export(object, dest, outvars = NULL, extra_params = character(0))
```

Arguments

`object`	generic object
`dest`	destination
`...`	optional arguments
`index`	index of the replicated Campsis model to export
`outvars`	additional variables to capture
`extra_params`	extra parameter names to be added. By default, they will be assigned a zero value.

Value

specific object depending on given destination

`export_type-class` *Export type class.*

Description

Export type class.

`extractLhs` *Extract left-hand-side expression.*

Description

Extract left-hand-side expression.

Usage

```
extractLhs(x, split = "=")
```

Arguments

`x`	character value
`split`	character where to split

Value

left-hand-side expression, not trimmed

`extractRhs` *Extract right-hand-side expression.*

Description

Extract right-hand-side expression.

Usage

```
extractRhs(x, split = "=")
```

Arguments

`x`	character value
`split`	character where to split

Value

right-hand side expression

`extractTextBetweenBrackets` *Extract text between brackets.*

Description

Extract text between brackets.

Usage

```
extractTextBetweenBrackets(x)
```

Arguments

`x`	character value

Value

text between brackets (trimmed)

`find` *Find an element in list.*

Description

Find an element in list.

Usage

```
find(object, x)

## S4 method for signature 'pmx_list,pmx_element'
find(object, x)

## S4 method for signature 'compartments,compartment_property'
find(object, x)

## S4 method for signature 'statements_record,model_statement'
find(object, x)

## S4 method for signature 'code_records,model_statement'
find(object, x)

## S4 method for signature 'campsis_model,compartment'
find(object, x)
```

```
## S4 method for signature 'campsis_model,compartment_property'
find(object, x)

## S4 method for signature 'campsis_model,parameter'
find(object, x)

## S4 method for signature 'campsis_model,code_record'
find(object, x)

## S4 method for signature 'campsis_model,model_statement'
find(object, x)
```

Arguments

`object`	list object
`x`	element to find, only key slots need to be filled in

Value

the element from the list that has same name as x, or NULL if no element was found

---

`fixOmega` *Fix omega matrix for SAME OMEGA parameters that have NA values due to imperfections in Pharmpy import.*

Description

Fix omega matrix for SAME OMEGA parameters that have NA values due to imperfections in Pharmpy import.

Usage

```
fixOmega(object)

## S4 method for signature 'parameters'
fixOmega(object)
```

Arguments

`object`	generic object

Value

the parameter that matches

`f_record-class` *Bioavailability record class.*

Description

Bioavailability record class.

`getByIndex` *Get element by index.*

Description

Get element by index.

Usage

```
getByIndex(object, x)

## S4 method for signature 'pmx_list,integer'
getByIndex(object, x)

## S4 method for signature 'pmx_list,numeric'
getByIndex(object, x)

## S4 method for signature 'compartment_properties,compartment_property'
getByIndex(object, x)

## S4 method for signature 'parameters,parameter'
getByIndex(object, x)
```

Arguments

`object`	list object
`x`	element index

Value

element from the list whose index matches with provided index

---

`getByName` *Get an element from a list by name. Never return more than 1 element.*

Description

Get an element from a list by name. Never return more than 1 element.

Usage

```
getByName(object, name)

## S4 method for signature 'pmx_list,character'
getByName(object, name)
```

Arguments

`object`	list object
`name`	element name to search for

Value

the element that was found or NULL if no element was found with the same name

---

`getCompartmentIndex` *Get the compartment index for the specified compartment name.*

Description

Get the compartment index for the specified compartment name.

Usage

```
getCompartmentIndex(object, name)

## S4 method for signature 'compartments,character'
getCompartmentIndex(object, name)

## S4 method for signature 'campsis_model,character'
getCompartmentIndex(object, name)
```

Arguments

`object`	generic object that contains compartments information
`name`	compartment name

Value

the corresponding compartment index

`getName` *Get element name.*

Description

Get element name.

Usage

```
getName(x)

## S4 method for signature 'unknown_statement'
getName(x)

## S4 method for signature 'comment'
getName(x)

## S4 method for signature 'line_break'
getName(x)

## S4 method for signature 'equation'
getName(x)

## S4 method for signature 'ode'
getName(x)

## S4 method for signature 'if_statement'
getName(x)

## S4 method for signature 'compartment'
getName(x)

## S4 method for signature 'compartment_bioavailability'
getName(x)

## S4 method for signature 'compartment_lag_time'
getName(x)

## S4 method for signature 'compartment_infusion_duration'
getName(x)

## S4 method for signature 'compartment_infusion_rate'
getName(x)
```

```
## S4 method for signature 'compartment_initial_condition'
getName(x)

## S4 method for signature 'theta'
getName(x)

## S4 method for signature 'omega'
getName(x)

## S4 method for signature 'sigma'
getName(x)

## S4 method for signature 'main_record'
getName(x)

## S4 method for signature 'ode_record'
getName(x)

## S4 method for signature 'f_record'
getName(x)

## S4 method for signature 'lag_record'
getName(x)

## S4 method for signature 'duration_record'
getName(x)

## S4 method for signature 'rate_record'
getName(x)

## S4 method for signature 'init_record'
getName(x)

## S4 method for signature 'error_record'
getName(x)

## S4 method for signature 'omega_block'
getName(x)
```

Arguments

`x`	element to know the name

Value

the name of this element

`getNameInModel` *Get the name of the given parameter in the CAMPSIS model.*

Description

Get the name of the given parameter in the CAMPSIS model.

Usage

```
getNameInModel(x)

## S4 method for signature 'theta'
getNameInModel(x)

## S4 method for signature 'omega'
getNameInModel(x)

## S4 method for signature 'sigma'
getNameInModel(x)
```

Arguments

`x`	element to know the name

Value

the name of this parameter

`getNames` *Get element names from list.*

Description

Get element names from list.

Usage

```
getNames(object)

## S4 method for signature 'pmx_list'
getNames(object)
```

Arguments

`object`	list object

Value

character vector with all the element names of this list

`getNONMEMName` *Get NONMEM name.*

Description

Get NONMEM name.

Usage

```
getNONMEMName(object)

## S4 method for signature 'theta'
getNONMEMName(object)

## S4 method for signature 'omega'
getNONMEMName(object)

## S4 method for signature 'sigma'
getNONMEMName(object)
```

Arguments

`object`	generic object

Value

the NONMEM name associated with this object

`getOmegaBlock` *Get the right block of OMEGA’s.*

Description

Get the right block of OMEGA’s.

Usage

```
getOmegaBlock(object, x)

## S4 method for signature 'omega_blocks,double_array_parameter'
getOmegaBlock(object, x)
```

Arguments

`object`	list of OMEGA blocks
`x`	omega param

Value

the corresponding OMEGA block or NULL if not found

`getOmegaIndexes` *Get the indexes of the omegas.*

Description

Get the indexes of the omegas.

Usage

```
getOmegaIndexes(object)

## S4 method for signature 'omega_block'
getOmegaIndexes(object)
```

Arguments

`object`	omega block

Value

a list of integers

`getPrefix` *Get prefix.*

Description

Get prefix.

Usage

```
getPrefix(object, ...)

## S4 method for signature 'compartment_bioavailability'
getPrefix(object, ...)

## S4 method for signature 'compartment_lag_time'
getPrefix(object, ...)

## S4 method for signature 'compartment_infusion_duration'
getPrefix(object, ...)

## S4 method for signature 'compartment_infusion_rate'
getPrefix(object, ...)

## S4 method for signature 'compartment_initial_condition'
getPrefix(object, ...)
```

Arguments

`object`	generic object
`...`	e.g. `dest='mrgsolve'`

Value

the prefix of this object

`getRecordDelimiter` *Get record delimiter.*

Description

Get record delimiter.

Usage

```
getRecordDelimiter(line)
```

Arguments

`line`	any line, single character value

Value

the record delimiter between brackets

`getRecordName` *Get record name.*

Description

Get record name.

Usage

```
getRecordName(object)

## S4 method for signature 'compartment_bioavailability'
getRecordName(object)

## S4 method for signature 'compartment_lag_time'
getRecordName(object)

## S4 method for signature 'compartment_infusion_duration'
getRecordName(object)

## S4 method for signature 'compartment_infusion_rate'
getRecordName(object)

## S4 method for signature 'compartment_initial_condition'
getRecordName(object)
```

Arguments

`object`	generic object

Value

the name of the record

`getUncertainty` *Get uncertainty on the parameters.*

Description

Get uncertainty on the parameters.

Usage

```
getUncertainty(object, ...)

## S4 method for signature 'parameter'
getUncertainty(object, varcov, ...)

## S4 method for signature 'parameters'
getUncertainty(object, ...)

## S4 method for signature 'campsis_model'
getUncertainty(object, ...)
```

Arguments

`object`	generic object
`...`	extra arguments
`varcov`	variance covariance matrix

Value

data frame with standard error (se) and relative standard error (rse

---

`getVarCov` *Get variance-covariance matrix.*

Description

Get variance-covariance matrix.

Usage

```
getVarCov(object)

## S4 method for signature 'parameters'
getVarCov(object)

## S4 method for signature 'campsis_model'
getVarCov(object)
```

Arguments

`object`	generic object

Value

a variance-covariance matrix (data frame) or NULL if no matrix present

---

`hasComment` *Check if string contains CAMPSIS-style comments.*

Description

Check if string contains CAMPSIS-style comments.

Usage

```
hasComment(x)
```

Arguments

`x`	character vector

Value

logical value

`hasOffDiagonalOmegas` *Has off-diagonal omegas.*

Description

Has off-diagonal omegas.

Usage

```
hasOffDiagonalOmegas(object)

## S4 method for signature 'omega_block'
hasOffDiagonalOmegas(object)
```

Arguments

`object`	omega block

Value

TRUE or FALSE

`IfStatement` *Create a new IF-statement.*

Description

Create a new IF-statement.

Usage

```
IfStatement(condition, equation, comment = as.character(NA))
```

Arguments

`condition`	condition, single character string
`equation`	equation if condition is met
`comment`	comment if any, single character string

Value

an IF-statement

`if_statement-class` *If-statement class. Any statement in the form if (condition) A = B.*

Description

If-statement class. Any statement in the form if (condition) A = B.

Slots

`condition` IF statement condition

`equation` any equation or ODE

`indexOf` *Get the index of an element in list.*

Description

Get the index of an element in list.

Usage

```
indexOf(object, x)

## S4 method for signature 'pmx_list,pmx_element'
indexOf(object, x)
```

Arguments

`object`	list object
`x`	element to know the index

Value

index of this element

`InfusionDuration` *Create an infusion duration.*

Description

Create an infusion duration.

Usage

```
InfusionDuration(compartment, rhs = "")
```

Arguments

`compartment`	compartment index
`rhs`	right-hand side part of the equation

Value

an infusion duration property

`InfusionRate` *Create an infusion rate.*

Description

Create an infusion rate.

Usage

```
InfusionRate(compartment, rhs = "")
```

Arguments

`compartment`	compartment index
`rhs`	right-hand side part of the equation

Value

an infusion rate property

`InitialCondition` *Create an initial condition.*

Description

Create an initial condition.

Usage

```
InitialCondition(compartment, rhs = "")
```

Arguments

`compartment`	compartment index
`rhs`	right-hand side part of the equation

Value

an initial condition property

`init_record-class` *Init record class.*

Description

Init record class.

`isComment` *Check if string is a CAMPSIS comment (i.e. not an equation).*

Description

Check if string is a CAMPSIS comment (i.e. not an equation).

Usage

```
isComment(x)
```

Arguments

`x`	character vector

Value

logical value

`isDiag` *Is diagonal.*

Description

Is diagonal.

Usage

```
isDiag(object)

## S4 method for signature 'double_array_parameter'
isDiag(object)
```

Arguments

`object`	generic object

Value

logical value

`isEmptyLine` *Check if string is an empty line.*

Description

Check if string is an empty line.

Usage

```
isEmptyLine(x)
```

Arguments

`x`	character vector

Value

logical value

`isEquation` *Say if line in record is an equation not.*

Description

Say if line in record is an equation not.

Usage

```
isEquation(x)
```

Arguments

`x`	character value

Value

logical value

`isIfStatement` *Say if line in record is an IF-statement.*

Description

Say if line in record is an IF-statement.

Usage

```
isIfStatement(x)
```

Arguments

`x`	character value

Value

logical value

`isMatrixPositiveDefinite` *Is matrix positive definite. Same check as* `mvtnorm` *does.*

Description

Is matrix positive definite. Same check as `mvtnorm` does.

Usage

```
isMatrixPositiveDefinite(matrix, tol = 1e-06)
```

Arguments

`matrix`	matrix to check
`tol`	tolerance when checking the eigenvalues

`isODE` *Say if line(s) in record is/are ODE or not.*

Description

Say if line(s) in record is/are ODE or not.

Usage

```
isODE(x)
```

Arguments

`x`	character vector

Value

logical vector

`isRecordDelimiter` *Is record delimiter. A record delimiter is any line starting with [...].*

Description

Is record delimiter. A record delimiter is any line starting with [...].

Usage

```
isRecordDelimiter(line)
```

Arguments

`line`	any line, single character value

Value

a logical value

`isStrictRecordDelimiter` *Is strict record delimiter. A strict record delimiter is any line starting with [...] and followed by nothing but spaces or a possible comment.*

Description

Is strict record delimiter. A strict record delimiter is any line starting with [...] and followed by nothing but spaces or a possible comment.

Usage

```
isStrictRecordDelimiter(line)
```

Arguments

`line`	any line, single character value

Value

a logical value

`LagTime` *Create a lag time for the specified compartment.*

Description

Create a lag time for the specified compartment.

Usage

```
LagTime(compartment, rhs = "")
```

Arguments

`compartment`	compartment index
`rhs`	right-hand side part of the equation

Value

a lag time property

`lag_record-class` *Lag record class.*

Description

Lag record class.

`LineBreak` *Create a new line break.*

Description

Create a new line break.

Usage

```
LineBreak()
```

Value

a line break

`line_break-class` *Line-break class. A linebreak in the model.*

Description

Line-break class. A linebreak in the model.

`MainRecord` *Create MAIN code record.*

Description

Create MAIN code record.

Usage

```
MainRecord(code = character())
```

Arguments

`code`	code record

`main_record-class` *Main record class.*

Description

Main record class.

`ManualReplicationSettings` *Create manual replication settings.*

Description

Create manual replication settings.

Usage

```
ManualReplicationSettings(data)
```

Arguments

`data`	data frame with 1 row per replicate, must contain a column named 'REPLICATE' with unique integers from 1 to nrow(data), other columns are model parameters to use.

Details

Use these settings to import custom replicated model parameters.

Value

replication settings

`manual_replication_settings-class` *Manual replication settings class.*

Description

Manual replication settings class.

Slots

`replicated_parameters`	data frame, 1 row per replicate

`maxIndex` *Max index.*

Description

Max index.

Usage

```
maxIndex(object)

## S4 method for signature 'parameters'
maxIndex(object)
```

Arguments

`object`	generic object

Value

max index

`minIndex` *Min index.*

Description

Min index.

Usage

```
minIndex(object)

## S4 method for signature 'parameters'
minIndex(object)
```

Arguments

`object`	generic object

Value

min index

`ModelStatements` *Create an empty list of model statements.*

Description

Create an empty list of model statements.

Usage

```
ModelStatements()
```

Value

a model statements object

`model_statement-class` *Model statement class. Any statement in a code record.*

Description

Model statement class. Any statement in a code record.

Slots

`comment` a comment associated to this model statement

`model_statements-class` *Model statements class. A list of statements.*

Description

Model statements class. A list of statements.

`model_suite` *CAMPSIS model suite.*

Description

A library of models of all kinds, ready to be simulated in Campsis. These model templates are sorted into the following categories: pharmacokinetic (PK), pharmacodynamic (PD), target-mediated drug disposition (TMDD), NONMEM, literature and other (custom models).

Usage

```
model_suite
```

Format

A list with all the models:

pk extensive list of pharmacokinetic (PK) model templates

pd list of pharmacodynamic (PD) model templates, to be plugged into any pharmacokinetic (PK) model

tmdd extensive list of target-mediated drug disposition (TMDD) model templates

nonmem list of model templates translated from standard NONMEM control streams

literature a couple of models coming from the literature

other a couple of custom models

Source

`https://calvagone.github.io/campsis.doc/`

`https://www.iconplc.com/solutions/technologies/nonmem/`

`move` *Move element ’x’ from object to a certain place.*

Description

Move element ’x’ from object to a certain place.

Usage

```
move(object, x, to, ...)

## S4 method for signature 'code_records,model_statement,pmx_position'
move(object, x, to, ...)

## S4 method for signature 'code_records,list,pmx_position'
move(object, x, to, ...)

## S4 method for signature 'code_records,model_statements,pmx_position'
move(object, x, to, ...)

## S4 method for signature 'code_records,code_record,pmx_position'
move(object, x, to, ...)

## S4 method for signature 'campsis_model,ANY,pmx_position'
move(object, x, to, ...)
```

Arguments

`object`	generic object (e.g. model, code records, etc.)
`x`	element to move
`to`	destination (e.g. a position)
`...`	extra arguments, unused

Value

updated object

`mrgsolveBlock` *Convert code record for mrgsolve.*

Description

Convert code record for mrgsolve.

Usage

```
mrgsolveBlock(record, init = NULL, capture = FALSE)
```

Arguments

`record`	code record
`init`	name of mrgsolve block
`capture`	'capture' instead of 'double'

Value

translated record for mrgsolve

`mrgsolveCapture` *Get the CAPTURE block for mrgsolve.*

Description

Get the CAPTURE block for mrgsolve.

Usage

```
mrgsolveCapture(outvars, model)
```

Arguments

`outvars`	outvars in method simulate
`model`	CAMPSIS model

Value

CAPTURE block or character(0) if no variable in outvars

`mrgsolveCompartment` *Get the compartment block for mrgsolve.*

Description

Get the compartment block for mrgsolve.

Usage

```
mrgsolveCompartment(model)
```

Arguments

`model`	CAMPSIS model

Value

character vector, each value is a line

`mrgsolveMain` *Get the MAIN block for mrgsolve.*

Description

Get the MAIN block for mrgsolve.

Usage

```
mrgsolveMain(model)
```

Arguments

`model`	CAMPSIS model

Value

MAIN block

`mrgsolveMatrix` *Get the OMEGA/SIGMA matrix for mrgsolve.*

Description

Get the OMEGA/SIGMA matrix for mrgsolve.

Usage

```
mrgsolveMatrix(model, type = "omega")
```

Arguments

`model`	CAMPSIS model
`type`	either omega or sigma

Value

named matrix or character(0) if matrix is empty

`mrgsolveOde` *Get the ODE block for mrgsolve.*

Description

Get the ODE block for mrgsolve.

Usage

```
mrgsolveOde(model)
```

Arguments

`model`	CAMPSIS model

Value

ODE block

`mrgsolveParam` *Get the parameters block for mrgsolve.*

Description

Get the parameters block for mrgsolve.

Usage

```
mrgsolveParam(model, extra_params = character(0))
```

Arguments

`model`	CAMPSIS model
`extra_params`	extra parameter names to be added. By default, they will be assigned a zero value.

Value

character vector, 1 parameter per line. First one is header [PARAM].

`mrgsolveTable` *Get the TABLE block for mrgsolve.*

Description

Get the TABLE block for mrgsolve.

Usage

```
mrgsolveTable(model)
```

Arguments

`model`	CAMPSIS model

Value

TABLE block if at least one line in error record, character(0) otherwise

---

`mrgsolve_type-class` *Mrgsolve export type class.*

Description

Mrgsolve export type class.

`Ode` *Create a new ordinary differential equation (ODE).*

Description

Create a new ordinary differential equation (ODE).

Usage

```
Ode(lhs, rhs = "", comment = as.character(NA))
```

Arguments

`lhs`	left-hand side variable corresponding to derivative name, must start with 'A_'
`rhs`	right-hand side expression corresponding to derivative value
`comment`	comment if any, single character string

Value

an ODE

`ode-class` *ODE class. Any statement in the form d/dt(A_CMT) = B.*

Description

ODE class. Any statement in the form d/dt(A_CMT) = B.

`OdeRecord` *Create ODE code record.*

Description

Create ODE code record.

Usage

```
OdeRecord(code = character())
```

Arguments

`code`	code record

Value

an ODE code record

`ode_record-class` *ODE record class.*

Description

ODE record class.

`Omega` *Create an OMEGA parameter.*

Description

Create an OMEGA parameter.

Usage

```
Omega(
  name = NA,
  index = NA,
  index2 = NA,
  value = NA,
  min = NA,
  max = NA,
  fix = FALSE,
  type = NULL,
  same = NA,
  label = NA,
  comment = NA
)
```

Arguments

`name`	parameter name, e.g. CL (prefix OMEGA will be added automatically)
`index`	parameter index
`index2`	second parameter index
`value`	parameter value
`min`	minimum value for this parameter when parameter uncertainty is enabled
`max`	maximum value for this parameter when parameter uncertainty is enabled
`fix`	parameter was fixed in estimation, logical value
`type`	variance type: 'var', 'sd', 'covar', 'cor', 'cv' or 'cv%'
`same`	NA by default, FALSE for first OMEGA followed by 'SAME' OMEGA's, TRUE for 'SAME' OMEGA's
`label`	parameter label, optional
`comment`	any comment, optional

Value

an OMEGA parameter

`omega-class` *Omega parameter class.*

Description

Omega parameter class.

Slots

`same` logical value, tell if this omega is the same as the previous one

`OmegaBlock` *Create a block of OMEGA's.*

Description

Create a block of OMEGA's.

Usage

```
OmegaBlock()
```

`OmegaBlocks` *Create a list of OMEGA blocks.*

Description

Create a list of OMEGA blocks.

Usage

```
OmegaBlocks()
```

`parameter-class` *Parameter class. Any parameter in a pharmacometric model.*

Description

Parameter class. Any parameter in a pharmacometric model.

Slots

- `name` parameter name, optional (although recommended)
- `index` parameter index, integer
- `value` parameter value (e.g. the estimated value from a modelling tool)
- `min` minimum value for this parameter when parameter uncertainty is enabled
- `max` maximum value for this parameter when parameter uncertainty is enabled
- `fix` logical value, say if parameter was fixed in the modelling phase
- `label` parameter label, any string
- `comment` any comment on this parameter, any string

`Parameters` *Create a list of parameters.*

Description

Create a list of parameters.

Usage

```
Parameters()
```

Value

an empty list of parameters

`parameters-class` *Parameters class.*

Description

Parameters class.

Slots

- `varcov` associated variance-covariance matrix

`parseIfStatement` *Parse IF-statement. Assumption:* `isIfStatement` *method already called and returned TRUE.*

Description

Parse IF-statement. Assumption: `isIfStatement` method already called and returned TRUE.

Usage

```
parseIfStatement(line, comment = as.character(NA))
```

Arguments

`line`	IF-statement as single character string value, comment omitted
`comment`	any comment, NA by default

Value

an IF statement object

`parseStatements` *Parse statements code and return CAMPSIS statements.*

Description

Parse statements code and return CAMPSIS statements.

Usage

```
parseStatements(code)
```

Arguments

`code`	character vector containing all statements (text form)

Value

a list of CAMPSIS statements

`Pattern` *Create a pattern.*

Description

Create a pattern.

Usage

```
Pattern(x)
```

Arguments

`x`	regular expression

Value

a pattern

`pattern-class` *Pattern class.*

Description

Pattern class.

`pmx_element-class` *PMX element class.*

Description

PMX element class.

`pmx_position-class` *PMX position class.*

Description

PMX position class.

`pmx_position_by_element-class` *PMX position by element class.*

Description

PMX position by element class.

`pmx_position_by_index-class` *PMX position by index class.*

Description

PMX position by index class.

`Position` *Element position in list.*

Description

Element position in list.

Usage

```
Position(x, after = TRUE)
```

Arguments

`x`	either an integer position (useful to add an element in a code record at a specified position) or an model element (element can be a model statement or a code record)
`after`	element to be added will be added after x (if after is TRUE) or before x (if after is FALSE)

Value

a position object

`processExtraArg` *Process extra arguments.*

Description

Process extra arguments.

Usage

```
processExtraArg(args, name, default = NULL, mandatory = FALSE)
```

Arguments

`args`	arguments list
`name`	argument name to retrieve
`default`	default value if argument is not present
`mandatory`	mandatory argument, logical value

Value

requested argument value

`properties_record-class` *Properties record class.*

Description

Properties record class.

`rate_record-class` *(Infusion)-rate record class.*

Description

(Infusion)-rate record class.

`read` *Generic read method to read data from a file or a folder.*

Description

Generic read method to read data from a file or a folder.

Usage

```
read(file, ...)
```

Arguments

`file`	path to the file or folder to be read
`...`	extra arguments

Value

the object representation of the data contained in the file

`read.allparameters` *Read all parameters files at once.*

Description

Read all parameters files at once.

Usage

```
read.allparameters(folder)
```

Arguments

`folder`	path to folder or path to zipped project

Value

parameters object

`read.campsis` *Read a CAMPSIS model.*

Description

Read a CAMPSIS model.

Usage

```
read.campsis(file)
```

Arguments

`file`	path to folder

Value

a CAMPSIS model

`read.model` *Read model file.*

Description

Read model file.

Usage

```
read.model(file = NULL, text = NULL)
```

Arguments

`file`	path to file 'model.campsis'
`text`	model file as text, character (single or multiple lines)

Value

records object

`read.parameters` *Read parameters file.*

Description

Read parameters file.

Usage

```
read.parameters(file, type)
```

Arguments

`file`	path to CSV file
`type`	parameter type: 'theta', 'omega' or 'sigma'

Value

parameters sub list

`read.varcov` *Read variance-covariance file.*

Description

Read variance-covariance file.

Usage

```
read.varcov(file)
```

Arguments

`file`	path to CSV file

Value

variance-covariance matrix

`replace` *Replace element by another in list.*

Description

Replace element by another in list.

Usage

```
replace(object, x)

## S4 method for signature 'pmx_list,pmx_element'
replace(object, x)

## S4 method for signature 'pmx_list,pmx_list'
replace(object, x)

## S4 method for signature 'pmx_list,list'
replace(object, x)

## S4 method for signature 'compartments,compartment_property'
replace(object, x)

## S4 method for signature 'parameters,single_array_parameter'
replace(object, x)

## S4 method for signature 'parameters,double_array_parameter'
replace(object, x)

## S4 method for signature 'statements_record,model_statement'
replace(object, x)

## S4 method for signature 'code_records,model_statement'
replace(object, x)

## S4 method for signature 'campsis_model,compartment'
replace(object, x)

## S4 method for signature 'campsis_model,compartment_property'
replace(object, x)

## S4 method for signature 'campsis_model,parameter'
replace(object, x)

## S4 method for signature 'campsis_model,code_record'
replace(object, x)
```

```
## S4 method for signature 'campsis_model,model_statement'
replace(object, x)
```

Arguments

`object`	list object
`x`	element to replace

Value

list object or an error if the element does not exist in the list

---

`replaceAll` *Replace all occurrences in object.*

Description

Replace all occurrences in object.

Usage

```
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'character,variable_pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'character,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'model_statement,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'equation,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'if_statement,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'compartment_property,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'code_record,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'code_records,pattern,character'
replaceAll(object, pattern, replacement, ...)
```

```
## S4 method for signature 'campsis_model,pattern,character'
replaceAll(object, pattern, replacement, ...)

## S4 method for signature 'campsis_model,character,character'
replaceAll(object, pattern, replacement, ...)
```

Arguments

`object`	generic object (e.g. model, code_record(s), etc.)
`pattern`	pattern to be replaced
`replacement`	replacement string
`...`	extra arguments

Value

the same object with all occurrences replaced

`replicate` *Replicate generic object.*

Description

Replicate generic object.

Usage

```
replicate(object, n, settings = NULL, ...)

## S4 method for signature 'campsis_model,integer,auto_replication_settings'
replicate(object, n, settings)

## S4 method for signature 'campsis_model,integer,manual_replication_settings'
replicate(object, n, settings)
```

Arguments

`object`	generic object
`n`	number of replicates required
`settings`	settings for replication
`...`	extra arguments

Value

object replicated n times

`replicated_campsis_model-class` *Replicated Campsis model class.*

Description

Replicated Campsis model class.

`replication_settings-class` *Replication settings interface.*

Description

Replication settings interface.

`rxodeCode` *Get code for rxode2*

Description

Get code for rxode2

Usage

```
rxodeCode(model)
```

Arguments

`model`	CAMPSIS model

Value

corresponding model code for rxode2

`rxodeMatrix` *Get the OMEGA/SIGMA matrix for rxode2.*

Description

Get the OMEGA/SIGMA matrix for rxode2.

Usage

```
rxodeMatrix(model, type = "omega")
```

Arguments

`model`	Campsis model or Campsis parameters
`type`	either omega or sigma

Value

omega/sigma named matrix

`rxodeParams` *Get the parameters vector for rxode2.*

Description

Get the parameters vector for rxode2.

Usage

```
rxodeParams(model)
```

Arguments

`model`	CAMPSIS model

Value

named vector with THETA values

`rxode_type-class` *RxODE/rxode2 export type class.*

Description

RxODE/rxode2 export type class.

`select` *Get a subset of an object.*

Description

Get a subset of an object.

Usage

```
select(object, ...)

## S4 method for signature 'data.frame'
select(object, ...)

## S4 method for signature 'compartment_properties'
select(object, ...)

## S4 method for signature 'parameters'
select(object, ...)
```

Arguments

`object`	generic object
`...`	arguments to select

Value

subset of an object

`setMinMax` *Set the minimum and maximum value on a model parameter.*

Description

Set the minimum and maximum value on a model parameter.

Usage

```
setMinMax(object, parameter, min, max, ...)

## S4 method for signature 'parameters,parameter,numeric,numeric'
setMinMax(object, parameter, min, max, ...)

## S4 method for signature 'parameters,character,numeric,numeric'
setMinMax(object, parameter, min, max, ...)
```

```
## S4 method for signature 'campsis_model,parameter,numeric,numeric'
setMinMax(object, parameter, min, max, ...)

## S4 method for signature 'campsis_model,character,numeric,numeric'
setMinMax(object, parameter, min, max, ...)
```

Arguments

`object`	model or parameters object
`parameter`	parameter object (Theta, Omega or Sigma)
`min`	minimum value for this parameter when parameter uncertainty is enabled
`max`	maximum value for this parameter when parameter uncertainty is enabled
`...`	extra arguments, unused

Value

updated object

`shiftOmegaIndexes` *Shift OMEGA indexes.*

Description

Shift OMEGA indexes.

Usage

```
shiftOmegaIndexes(object)

## S4 method for signature 'omega_block'
shiftOmegaIndexes(object)
```

Arguments

`object`	omega block

Value

same block but shifted

`Sigma` *Create a SIGMA parameter.*

Description

Create a SIGMA parameter.

Usage

```
Sigma(
  name = NA,
  index = NA,
  index2 = NA,
  value = NA,
  min = NA,
  max = NA,
  fix = FALSE,
  type = NULL,
  label = NA,
  comment = NA
)
```

Arguments

`name`	parameter name, e.g. CL (prefix SIGMA will be added automatically)
`index`	parameter index
`index2`	second parameter index
`value`	parameter value
`min`	minimum value for this parameter when parameter uncertainty is enabled
`max`	maximum value for this parameter when parameter uncertainty is enabled
`fix`	parameter was fixed in estimation, logical value
`type`	variance type: ’var’, ’sd’, ’covar’, ’cv’ or ’cv%’
`label`	parameter label, optional
`comment`	any comment, optional

Value

a SIGMA parameter

`sigma-class` *Sigma parameter class.*

Description

Sigma parameter class.

`single_array_parameter-class` *Single-array parameter class. This parameter has a single index value.*

Description

Single-array parameter class. This parameter has a single index value.

`sort` *Sort the specified list.*

Description

Sort the specified list.

Usage

```
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'compartment_properties'
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'compartments'
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'parameters'
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'code_records'
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'campsis_model'
sort(x, decreasing = FALSE, ...)

## S4 method for signature 'omega_blocks'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	list object
`decreasing`	increasing or decreasing order
`...`	extra arguments

Value

same list but ordered

`standardise` *Standardise.*

Description

Standardise.

Usage

```
standardise(object, ...)

## S4 method for signature 'theta'
standardise(object, ...)

## S4 method for signature 'double_array_parameter'
standardise(object, parameters = NULL, ...)

## S4 method for signature 'parameters'
standardise(object, ...)

## S4 method for signature 'campsis_model'
standardise(object, ...)
```

Arguments

`object`	generic object
`...`	extra arguments needed for standardisation
`parameters`	the list of parameters, to be provided only if parameter type is 'cor'

Value

standardised object

`statements_record-class` *Statements record class.*

Description

Statements record class.

`Theta` *Create a THETA parameter.*

Description

Create a THETA parameter.

Usage

```
Theta(
  name = NA,
  index = NA,
  value = NA,
  min = NA,
  max = NA,
  fix = FALSE,
  label = NA,
  unit = NA,
  comment = NA
)
```

Arguments

`name`	parameter name, e.g. CL (prefix THETA will be added automatically)
`index`	parameter index
`value`	parameter value
`min`	minimum value for this parameter when parameter uncertainty is enabled
`max`	maximum value for this parameter when parameter uncertainty is enabled
`fix`	parameter was fixed in estimation, logical value
`label`	parameter label, optional
`unit`	parameter unit, optional
`comment`	any comment, optional

Value

a THETA parameter

`theta-class` *Theta parameter class.*

Description

Theta parameter class.

Slots

`unit` parameter unit

`toString` *ToString generic method.*

Description

ToString generic method.

Usage

```
toString(object, ...)

## S4 method for signature 'unknown_statement'
toString(object, ...)

## S4 method for signature 'comment'
toString(object, ...)

## S4 method for signature 'line_break'
toString(object, ...)

## S4 method for signature 'equation'
toString(object, ...)

## S4 method for signature 'ode'
toString(object, ...)

## S4 method for signature 'if_statement'
toString(object, ...)

## S4 method for signature 'compartment'
toString(object, ...)

## S4 method for signature 'compartment_property'
toString(object, ...)
```

```
## S4 method for signature 'compartment_initial_condition'
toString(object, ...)

## S4 method for signature 'mrgsolve_model'
toString(object, ...)
```

Arguments

`object`	generic object
`...`	extra arguments needed for toString conversion

Value

character value/vector

`trim` *Trim character vector. Remove all leading and trailing spaces.*

Description

Trim character vector. Remove all leading and trailing spaces.

Usage

```
trim(x)
```

Arguments

`x`	character vector

Value

character vector without leading and trailing spaces

`UnknownStatement` *Create a new ordinary differential equation (ODE).*

Description

Create a new ordinary differential equation (ODE).

Usage

```
UnknownStatement(line, comment = as.character(NA))
```

Arguments

`line`	line which was not recognised
`comment`	comment if any, single character string

Value

an unknown statement

`unknown_statement-class` *Unknown statement class. Any statement not recognized by campsismod.*

Description

Unknown statement class. Any statement not recognized by campsismod.

`updateCompartments` *Update compartments list from the persisted records. Exported especially for package `campsistrans`. However, this method should not be called.*

Description

Update compartments list from the persisted records. Exported especially for package `campsistrans`. However, this method should not be called.

Usage

```
updateCompartments(model)
```

Arguments

`model`	Campsis model

Value

an updated Campsis model, with an updated compartments list

`VariablePattern` *Create a variable pattern.*

Description

Create a variable pattern.

Usage

```
VariablePattern(x)
```

Arguments

`x`	variable name

Value

a variable pattern

`variable_pattern-class` *Variable pattern class.*

Description

Variable pattern class.

`write` *Write generic object to files.*

Description

Write generic object to files.

Usage

```
write(object, file, ...)

## S4 method for signature 'parameters,character'
write(object, file, ...)

## S4 method for signature 'code_records,character'
write(object, file, ...)

## S4 method for signature 'campsis_model,character'
write(object, file, ...)
```

Arguments

`object`	generic object
`file`	path of the output file or directory
`...`	extra arguments

Value

logical value, TRUE for success, FALSE for failure

`writeParameters` *Write subset of parameters (theta, omega or sigma).*

Description

Write subset of parameters (theta, omega or sigma).

Usage

```
writeParameters(object, file, ...)
```

Arguments

`object`	subset of parameters
`file`	filename
`...`	extra arguments, like `defaultDf` for empty parameters list

Value

TRUE if success

`writeVarcov` *Write variance-covariance matrix.*

Description

Write variance-covariance matrix.

Usage

```
writeVarcov(object, file)
```

Arguments

`object`	matrix
`file`	filename

Value

TRUE if success

Index