

Package ‘campsis’

February 9, 2025

Type Package

Title Generic PK/PD Simulation Platform CAMPSIS

Version 1.6.0

Description A generic, easy-to-use and intuitive pharmacokinetic/pharmacodynamic (PK/PD) simulation platform based on R packages 'rxode2' and 'mrgsolve'. CAMPSIS provides an abstraction layer over the underlying processes of writing a PK/PD model, assembling a custom dataset and running a simulation. CAMPSIS has a strong dependency to the R package 'campsismod', which allows to read/write a model from/to files and adapt it further on the fly in the R environment. Package 'campsis' allows the user to assemble a dataset in an intuitive manner. Once the user's dataset is ready, the package is in charge of preparing the simulation, calling 'rxode2' or 'mrgsolve' (at the user's choice) and returning the results, for the given model, dataset and desired simulation settings.

License GPL (>= 3)

URL <https://github.com/Calvagone/campsis>,
<https://calvagone.github.io/>,
<https://calvagone.github.io/campsis.doc/>

BugReports <https://github.com/Calvagone/campsis/issues>

Depends campsismod (>= 1.2.0), R (>= 4.0.0)

Imports assertthat, digest, dplyr, furr, future, ggplot2, MASS,
methods, progressr, purrr, rlang, stats, tibble, tidyr

Suggests bookdown, devtools, gridExtra, knitr, mrgsolve, pkgdown,
rmarkdown, roxygen2, rxode2, stringr, testthat, tictoc, vdiff,
xfun

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Collate 'global.R' 'utilities.R' 'time_utilities.R' 'check.R'
 'generic.R' 'data.R' 'seed.R' 'distribution.R'
 'dataset_config.R' 'time_entry.R' 'occasion.R' 'occasions.R'
 'treatment_iov.R' 'treatment_iovs.R' 'dose_adaptation.R'
 'dose_adaptations.R' 'treatment_entry.R' 'treatment.R'
 'observations.R' 'observations_set.R' 'covariate.R'
 'covariates.R' 'bootstrap.R' 'protocol.R' 'arm.R' 'arms.R'
 'event.R' 'events.R' 'scenario.R' 'scenarios.R'
 'simulation_engine.R' 'dataset.R' 'event_logic.R'
 'dataset_summary.R' 'outfun.R' 'hardware_settings.R'
 'simulation_progress.R' 'solver_settings.R' 'nocb_settings.R'
 'declare_settings.R' 'progress_settings.R'
 'internal_settings.R' 'simulation_settings.R' 'plan_setup.R'
 'simulate_preprocess.R' 'simulate.R' 'results_processing.R'
 'default_plot.R'

NeedsCompilation no

Author Nicolas Luyckx [aut, cre]

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

Repository CRAN

Date/Publication 2025-02-09 17:20:02 UTC

Contents

applyCompartmentCharacteristics	5
Arm	5
arm-class	6
arms-class	6
BinomialDistribution	7
Bolus	7
bolus-class	8
Bootstrap	8
bootstrap-class	9
BootstrapDistribution	9
bootstrap_distribution-class	10
campsis_handler	10
ConstantDistribution	11
constant_distribution-class	11
convertTime	12
Covariate	12
covariate-class	13
covariates-class	13
Dataset	13
dataset-class	14
DatasetConfig	14
dataset_config-class	15

days	15
Declare	16
declare_settings-class	16
DiscreteDistribution	16
distribution-class	17
DoseAdaptation	17
dose_adaptation-class	18
dose_adaptations-class	18
dosingOnly	18
EtaDistribution	19
Event	19
event-class	20
EventCovariate	20
Events	21
events-class	21
event_covariate-class	21
FixedDistribution	22
fixed_covariate-class	22
fixed_distribution-class	22
FunctionDistribution	23
function_distribution-class	23
generateIV	24
generateIV_	24
getAvailableTimeUnits	25
getCovariates	25
getEventCovariates	26
getFixedCovariates	26
getIOVs	27
getOccasions	28
getSeedForDatasetExport	28
getSeedForIteration	29
getSeedForParametersSampling	29
getSplittingConfiguration	30
getTimes	30
getTimeVaryingCovariates	31
Hardware	32
hardware_settings-class	33
hours	33
Infusion	34
infusion-class	35
internal_settings-class	35
IOV	35
length,arm-method	36
length,dataset-method	36
LogNormalDistribution	37
minutes	37
months	38
mrgsolve_engine-class	38

nhanes	38
NOCB	39
nocb_settings-class	40
NormalDistribution	40
Observations	41
observations-class	41
observations_set-class	41
obsOnly	42
Occasion	42
occasion-class	43
occasions-class	43
Outfun	43
output_function-class	44
ParameterDistribution	44
PI	45
Progress	45
progress_settings-class	46
protocol-class	46
retrieveParameterValue	46
rxode_engine-class	47
sample	47
scatterPlot	48
Scenario	49
scenario-class	49
Scenarios	50
scenarios-class	50
seconds	50
setLabel	51
setSubjects	51
Settings	52
setupPlanDefault	52
setupPlanSequential	53
shadedPlot	53
simulate	54
SimulationProgress	58
simulation_engine-class	59
simulation_progress-class	59
simulation_settings-class	60
Solver	60
solver_settings-class	61
spaghettiPlot	61
standardiseTime	62
TimeVaryingCovariate	62
time_varying_covariate-class	63
treatment-class	63
treatment_iov-class	63
treatment_iovs-class	63
undefined_distribution-class	64

<i>applyCompartmentCharacteristics</i>	5
UniformDistribution	64
VPC	65
vpcPlot	65
weeks	66
years	66
Index	67

`applyCompartmentCharacteristics`

Apply compartment characteristics from model. In practice, only compartment infusion duration needs to be applied.

Description

Apply compartment characteristics from model. In practice, only compartment infusion duration needs to be applied.

Usage

```
applyCompartmentCharacteristics(table, properties)
```

Arguments

<code>table</code>	current dataset
<code>properties</code>	compartment properties from model

Value

updated dataset

<code>Arm</code>	<i>Create a treatment arm.</i>
------------------	--------------------------------

Description

Create a treatment arm.

Usage

```
Arm(id = as.integer(NA), subjects = 1, label = as.character(NA))
```

Arguments

id	unique identifier for this arm (available through dataset), integer. If NA (default), this identifier is auto-incremented.
subjects	number of subjects in arm, integer
label	arm label, single character string. If set, this label will be output in the ARM column of CAMPSIS instead of the identifier.

Value

an arm

arm-class	<i>Arm class.</i>
-----------	-------------------

Description

Arm class.

Slots

id arm unique ID, integer
 subjects number of subjects in arm, integer
 label arm label, single character string
 protocol protocol
 covariates covariates
 bootstrap covariates to be bootstrapped

arms-class	<i>Arms class.</i>
------------	--------------------

Description

Arms class.

BinomialDistribution *Binomial distribution.*

Description

Binomial distribution.

Usage

```
BinomialDistribution(trials, prob)
```

Arguments

trials	number of Bernoulli trials per observation (=subject), integer
prob	probability of success for each trial

Value

a binomial distribution

Bolus *Create one or several bolus(es).*

Description

Create one or several bolus(es).

Usage

```
Bolus(  
  time,  
  amount,  
  compartment = NA,  
  f = NULL,  
  lag = NULL,  
  ii = NULL,  
  addl = NULL  
)
```

Arguments

time	treatment time(s), numeric value or vector. First treatment time if used together with ii and addl.
amount	amount to give as bolus, single numeric value
compartment	compartment index, single integer value
f	fraction of dose amount, distribution
lag	dose lag time, distribution
ii	interdose interval, requires argument 'time' to be a single numeric value
addl	number of additional doses, requires argument 'time' to be a single integer value

Value

a single bolus or a list of boluses

bolus-class	<i>Bolus class.</i>
-------------	---------------------

Description

Bolus class.

Bootstrap	<i>Create a bootstrap object.</i>
-----------	-----------------------------------

Description

Create a bootstrap object.

Usage

```
Bootstrap(
  data,
  id = "BS_ID",
  replacement = FALSE,
  random = FALSE,
  export_id = FALSE
)
```


Arguments

data	data frame to be bootstrapped. It must have a unique identifier column named according to the specified argument 'id' (default value is 'BS_ID'). Other columns are covariates to bootstrap. They must all be numeric. Whatever the configuration of the bootstrap, these covariates are always read row by row and belong to a same individual.
id	unique identifier column name in data
replacement	values can be reused or not when drawn, logical
random	values are drawn randomly, logical
export_id	tell CAMPSIS if the identifier 'BS_ID' must be output or not, logical

Value

a bootstrap object

bootstrap-class	<i>Bootstrap class.</i>
-----------------	-------------------------

Description

Bootstrap class.

Slots

data	data frame to be bootstrapped. Column 'BS_ID' is mandatory and corresponds to the original row ID from the bootstrap. It must be numeric and unique. Other columns are covariates to be bootstrapped (row by row).
replacement	values can be reused or not, logical
random	values are drawn randomly, logical
export_id	tell CAMPSIS if 'BS_ID' must be exported into the dataset, logical

BootstrapDistribution	<i>Create a bootstrap distribution. During function sampling, CAMPSIS will generate values depending on the given data and arguments.</i>
-----------------------	---

Description

Create a bootstrap distribution. During function sampling, CAMPSIS will generate values depending on the given data and arguments.

Usage

```
BootstrapDistribution(data, replacement = FALSE, random = FALSE)
```

Arguments

data	values to draw, numeric vector
replacement	values can be reused or not, logical
random	values are drawn randomly, logical

Value

a bootstrap distribution

bootstrap_distribution-class
Bootstrap distribution class.

Description

Bootstrap distribution class.

Slots

data	values to draw, numeric vector
replacement	values can be reused or not, logical
random	values are drawn randomly, logical

campsis_handler *Suggested Campsis handler for showing the progress bar.*

Description

Suggested Campsis handler for showing the progress bar.

Usage

campsis_handler()

Value

a progressr handler list

ConstantDistribution *Create a constant distribution. Its value will be constant across all generated samples.*

Description

Create a constant distribution. Its value will be constant across all generated samples.

Usage

```
ConstantDistribution(value)
```

Arguments

value covariate value, single numeric value

Value

a constant distribution (same value for all samples)

constant_distribution-class
Constant distribution class.

Description

Constant distribution class.

Slots

value covariate value, single numeric value

convertTime	<i>Convert numeric time vector based on the provided units.</i>
-------------	---

Description

Convert numeric time vector based on the provided units.

Usage

```
convertTime(x, from, to)
```

Arguments

x	numeric time vector
from	unit of x, single character value
to	destination unit, single character value

Value

numeric vector with the converted times

Covariate	<i>Create a non time-varying (fixed) covariate.</i>
-----------	---

Description

Create a non time-varying (fixed) covariate.

Usage

```
Covariate(name, distribution)
```

Arguments

name	covariate name, single character value
distribution	covariate distribution

Value

a fixed covariate

covariate-class	<i>Covariate class.</i>
-----------------	-------------------------

Description

Covariate class.

Slots

name covariate name, single character value

distribution covariate distribution

covariates-class	<i>Covariates class.</i>
------------------	--------------------------

Description

Covariates class.

Dataset	<i>Create a dataset.</i>
---------	--------------------------

Description

Create a dataset.

Usage

```
Dataset(subjects = NULL, label = as.character(NA))
```

Arguments

subjects number of subjects in the default arm

label label of the default arm, NA by default

Value

a dataset

dataset-class	<i>Dataset class.</i>
---------------	-----------------------

Description

Dataset class.

Slots

arms a list of treatment arms
 config dataset configuration for export
 iiv data frame containing the inter-individual variability (all ETAS) for the export

DatasetConfig	<i>Create a dataset configuration. This configuration allows CAMPSIS to know which are the default depot and observed compartments.</i>
---------------	---

Description

Create a dataset configuration. This configuration allows CAMPSIS to know which are the default depot and observed compartments.

Usage

```
DatasetConfig(
  defDepotCmt = 1,
  defObsCmt = 1,
  exportTSLD = FALSE,
  exportTDOS = FALSE,
  timeUnitDataset = "hour",
  timeUnitExport = "hour"
)
```

Arguments

defDepotCmt	default depot compartment, integer
defObsCmt	default observation compartment, integer
exportTSLD	export column TSLD (time since last dose), logical
exportTDOS	export column TDOS (time of last dose), logical
timeUnitDataset	unit of time in dataset, character ('hour' by default)
timeUnitExport	unit of time in export, character ('hour' by default)

Value

a dataset configuration

dataset_config-class *Dataset configuration class.*

Description

Dataset configuration class.

Slots

def_depot_cmt default depot compartment, integer
def_obs_cmt default observation compartment, integer
export_tsld export column TSLD, logical
export_tdos export column TDOS, logical
time_unit_dataset unit of time in dataset, character ('hour' by default)
time_unit_export unit of time in export, character ('hour' by default)

days *Convert days to hours.*

Description

Convert days to hours.

Usage

days(x)

Arguments

x numeric vector in days

Value

numeric vector in hours

Declare *Create declare settings.*

Description

Create declare settings.

Usage

Declare(variables = character(0))

Arguments

variables uninitialized variables to be declared, only needed with mrgsolve

Value

Declare settings

declare_settings-class *Declare settings class.*

Description

Declare settings class.

Slots

variables uninitialized variables to be declared, only needed with mrgsolve

DiscreteDistribution *Discrete distribution.*

Description

Discrete distribution.

Usage

DiscreteDistribution(x, prob, replace = TRUE)

Arguments

x	vector of one or more integers from which to choose
prob	a vector of probability weights for obtaining the elements of the vector being sampled
replace	should sampling be with replacement, default is TRUE

Value

a discrete distribution

distribution-class *Distribution class. See this class as an interface.*

Description

Distribution class. See this class as an interface.

DoseAdaptation *Create a dose adaptation.*

Description

Create a dose adaptation.

Usage

```
DoseAdaptation(formula, compartments = integer(0))
```

Arguments

formula	formula to apply, single character string, e.g. "AMT*WT"
compartments	compartment numbers where the formula needs to be applied, integer vector. Default is integer(0) (formula applied on all compartments)

Value

a fixed covariate

dose_adaptation-class *Dose adaptation class.*

Description

Dose adaptation class.

Slots

formula formula to apply, single character string, e.g. "AMT*WT"

compartments compartment numbers where the formula needs to be applied

dose_adaptations-class
Dose adaptations class.

Description

Dose adaptations class.

dosingOnly *Filter CAMPSIS output on dosing rows.*

Description

Filter CAMPSIS output on dosing rows.

Usage

dosingOnly(x)

Arguments

x data frame, CAMPSIS output

Value

a data frame with the dosing rows

EtaDistribution	<i>Create an ETA distribution. The resulting distribution is a normal distribution, with mean=0 and sd=sqrt(OMEGA).</i>
-----------------	---

Description

Create an ETA distribution. The resulting distribution is a normal distribution, with mean=0 and sd=sqrt(OMEGA).

Usage

```
EtaDistribution(model, omega)
```

Arguments

model	model
omega	corresponding THETA name, character

Value

an ETA distribution

Event	<i>Create an interruption event.</i>
-------	--------------------------------------

Description

Create an interruption event.

Usage

```
Event(name = NULL, times, fun, debug = FALSE)
```

Arguments

name	event name, character value
times	interruption times, numeric vector
fun	event function to apply at each interruption
debug	output the variables that were changed through this event

Value

an event definition

event-class	<i>Event class.</i>
-------------	---------------------

Description

Event class.

Slots

name event name, character value
 times interruption times, numeric vector
 fun event function to apply at each interruption
 debug output the variables that were changed through this event

EventCovariate	<i>Create an event covariate. These covariates can be modified further in interruption events.</i>
----------------	--

Description

Create an event covariate. These covariates can be modified further in interruption events.

Usage

```
EventCovariate(name, distribution)
```

Arguments

name covariate name, character
 distribution covariate distribution at time 0

Value

a time-varying covariate

Events	<i>Create a list of interruption events.</i>
--------	--

Description

Create a list of interruption events.

Usage

Events()

Value

a events object

events-class	<i>Events class.</i>
--------------	----------------------

Description

Events class.

event_covariate-class	<i>Event covariate class.</i>
-----------------------	-------------------------------

Description

Event covariate class.

FixedDistribution	<i>Create a fixed distribution. Each sample will be assigned a fixed value coming from vector 'values'.</i>
-------------------	---

Description

Create a fixed distribution. Each sample will be assigned a fixed value coming from vector 'values'.

Usage

```
FixedDistribution(values)
```

Arguments

values covariate values, numeric vector (1 value per sample)

Value

a fixed distribution (1 value per sample)

fixed_covariate-class	<i>Fixed covariate class.</i>
-----------------------	-------------------------------

Description

Fixed covariate class.

fixed_distribution-class	<i>Fixed distribution class.</i>
--------------------------	----------------------------------

Description

Fixed distribution class.

Slots

values covariate values, numeric vector (1 value per sample)

FunctionDistribution *Create a function distribution. During distribution sampling, the provided function will be responsible for generating values for each sample. If first argument of this function is not the size (n), please tell which argument corresponds to the size 'n' (e.g. list(size="n")).*

Description

Create a function distribution. During distribution sampling, the provided function will be responsible for generating values for each sample. If first argument of this function is not the size (n), please tell which argument corresponds to the size 'n' (e.g. list(size="n")).

Usage

```
FunctionDistribution(fun, args)
```

Arguments

fun	function name, character (e.g. 'rnorm')
args	list of arguments (e.g list(mean=70, sd=10))

Value

a function distribution

function_distribution-class
Function distribution class.

Description

Function distribution class.

Slots

fun	function name, character (e.g. 'rnorm')
args	list of arguments (e.g list(mean=70, sd=10))

generateIIV	<i>Generate IIV matrix for the given Campsis model.</i>
-------------	---

Description

Generate IIV matrix for the given Campsis model.

Usage

```
generateIIV(model, n, offset = 0)
```

Arguments

model	Campsis model
n	number of subjects
offset	if specified, resulting ID will be ID + offset

Value

IIV data frame with ID column

generateIIV_	<i>Generate IIV matrix for the given OMEGA matrix.</i>
--------------	--

Description

Generate IIV matrix for the given OMEGA matrix.

Usage

```
generateIIV_(omega, n)
```

Arguments

omega	omega matrix
n	number of subjects

Value

IIV data frame

getAvailableTimeUnits *Return the list of available time units.*

Description

Return the list of available time units.

Usage

```
getAvailableTimeUnits()
```

Value

character vector

getCovariates *Get all covariates (fixed / time-varying / event covariates).*

Description

Get all covariates (fixed / time-varying / event covariates).

Usage

```
getCovariates(object)
```

```
## S4 method for signature 'covariates'  
getCovariates(object)
```

```
## S4 method for signature 'arm'  
getCovariates(object)
```

```
## S4 method for signature 'arms'  
getCovariates(object)
```

```
## S4 method for signature 'dataset'  
getCovariates(object)
```

Arguments

object any object

Value

all covariates from object

getEventCovariates *Get all event-related covariates.*

Description

Get all event-related covariates.

Usage

```
getEventCovariates(object)

## S4 method for signature 'covariates'
getEventCovariates(object)

## S4 method for signature 'arm'
getEventCovariates(object)

## S4 method for signature 'arms'
getEventCovariates(object)

## S4 method for signature 'dataset'
getEventCovariates(object)
```

Arguments

object any object

Value

all event-related covariates from object

getFixedCovariates *Get all fixed covariates.*

Description

Get all fixed covariates.

Usage

```
getFixedCovariates(object)

## S4 method for signature 'covariates'
getFixedCovariates(object)

## S4 method for signature 'arm'
```

```
getFixedCovariates(object)

## S4 method for signature 'arms'
getFixedCovariates(object)

## S4 method for signature 'dataset'
getFixedCovariates(object)
```

Arguments

object any object

Value

all fixed covariates from object

getIOVs	<i>Get all IOV objects.</i>
---------	-----------------------------

Description

Get all IOV objects.

Usage

```
getIOVs(object)

## S4 method for signature 'arm'
getIOVs(object)

## S4 method for signature 'arms'
getIOVs(object)

## S4 method for signature 'dataset'
getIOVs(object)
```

Arguments

object any object

Value

all IOV's from object

get0ccasions	<i>Get all occasions.</i>
--------------	---------------------------

Description

Get all occasions.

Usage

```
get0ccasions(object)

## S4 method for signature 'arm'
get0ccasions(object)

## S4 method for signature 'arms'
get0ccasions(object)

## S4 method for signature 'dataset'
get0ccasions(object)
```

Arguments

object	any object
--------	------------

Value

all occasions from object

getSeedForDatasetExport	<i>Get seed for dataset export.</i>
-------------------------	-------------------------------------

Description

Get seed for dataset export.

Usage

```
getSeedForDatasetExport(seed, progress)
```

Arguments

seed	original seed
progress	simulation progress

Value

the seed value used to export the dataset

`getSeedForIteration` *Get seed for iteration.*

Description

Get seed for iteration.

Usage

`getSeedForIteration(seed, progress)`

Arguments

<code>seed</code>	original seed
<code>progress</code>	simulation progress

Value

the seed value to be used for the given replicate number and iteration

`getSeedForParametersSampling`
Get seed for parameter uncertainty sampling.

Description

Get seed for parameter uncertainty sampling.

Usage

`getSeedForParametersSampling(seed)`

Arguments

<code>seed</code>	original seed
-------------------	---------------

Value

the seed value used to sample parameter uncertainty

`getSplittingConfiguration`*Get splitting configuration for parallel export.*

Description

Get splitting configuration for parallel export.

Usage

```
getSplittingConfiguration(dataset, hardware)
```

Arguments

dataset	Campsis dataset to export
hardware	hardware configuration

Value

splitting configuration list (if 'parallel_dataset' is enabled) or NA (if 'parallel_dataset' disabled or if the length of the dataset is less than the dataset export slice size)

`getTimes`*Get all distinct times for the specified object.*

Description

Get all distinct times for the specified object.

Usage

```
getTimes(object)
```

```
## S4 method for signature 'observations_set'  
getTimes(object)
```

```
## S4 method for signature 'arm'  
getTimes(object)
```

```
## S4 method for signature 'arms'  
getTimes(object)
```

```
## S4 method for signature 'events'  
getTimes(object)
```

```
## S4 method for signature 'dataset'  
getTimes(object)
```

Arguments

object any object

Value

numeric vector with all unique times, sorted

`getTimeVaryingCovariates`
Get all time-varying covariates.

Description

Get all time-varying covariates.

Usage

```
getTimeVaryingCovariates(object)

## S4 method for signature 'covariates'
getTimeVaryingCovariates(object)

## S4 method for signature 'arm'
getTimeVaryingCovariates(object)

## S4 method for signature 'arms'
getTimeVaryingCovariates(object)

## S4 method for signature 'dataset'
getTimeVaryingCovariates(object)
```

Arguments

object any object

Value

all time-varying covariates from object

Hardware

Create hardware settings.

Description

Create hardware settings.

Usage

```
Hardware(
  cpu = 1,
  replicate_parallel = FALSE,
  scenario_parallel = FALSE,
  slice_parallel = FALSE,
  slice_size = NULL,
  dataset_parallel = FALSE,
  dataset_slice_size = 500,
  auto_setup_plan = NULL
)
```

Arguments

cpu	number of CPU cores to use, default is 1
replicate_parallel	enable parallel computing for replicates, default is FALSE
scenario_parallel	enable parallel computing for scenarios, default is FALSE
slice_parallel	enable parallel computing for slices, default is FALSE
slice_size	number of subjects per simulated slice, default is NULL (auto-configured by Campsis depending on the specified engine)
dataset_parallel	enable parallelisation when exporting dataset into a table, default is FALSE
dataset_slice_size	dataset slice size when exporting subjects to a table, default is 500. Only applicable if 'dataset_parallel' is enabled.
auto_setup_plan	auto-setup plan with the library future, if not set (i.e. =NULL), plan will be setup automatically if the number of CPU's > 1.

Value

hardware settings

hardware_settings-class
Hardware settings class.

Description

Hardware settings class.

Slots

cpu number of CPU cores to use, default is 1
 replicate_parallel enable parallel computing for replicates, default is FALSE
 scenario_parallel enable parallel computing for scenarios, default is FALSE
 slice_parallel enable parallel computing for slices, default is FALSE
 slice_size number of subjects per simulated slice, default is NULL (auto-configured by Campsis depending on the specified engine)
 dataset_parallel enable parallelisation when exporting dataset into a table, default is FALSE
 dataset_slice_size dataset slice size when exporting subjects to a table, default is 500. Only applicable if 'dataset_parallel' is enabled.
 auto_setup_plan auto-setup plan with the library future, default is FALSE

hours *Convert hours to hours (do nothing).*

Description

Convert hours to hours (do nothing).

Usage

hours(x)

Arguments

x numeric vector in hours

Value

numeric vector in hours

Infusion *Create one or several infusion(s).*

Description

Create one or several infusion(s).

Usage

```
Infusion(
  time,
  amount,
  compartment = NA,
  f = NULL,
  lag = NULL,
  duration = NULL,
  rate = NULL,
  ii = NULL,
  addl = NULL
)
```

Arguments

time	treatment time(s), numeric value or vector. First treatment time if used together with ii and addl.
amount	total amount to infuse, numeric
compartment	compartment index, integer
f	fraction of infusion amount, distribution
lag	infusion lag time, distribution
duration	infusion duration, distribution
rate	infusion rate, distribution
ii	interdose interval, requires argument 'time' to be a single numeric value
addl	number of additional doses, requires argument 'time' to be a single integer value

Value

a single infusion or a list of infusions.

infusion-class	<i>Infusion class.</i>
----------------	------------------------

Description

Infusion class.

Slots

duration infusion duration, distribution
 rate infusion rate, distribution

internal_settings-class	<i>Internal settings class (transient object from the simulation settings).</i>
-------------------------	---

Description

Internal settings class (transient object from the simulation settings).

Slots

dataset_summary dataset summary
 progress simulation progress
 iterations list of event iterations

IOV	<i>Define inter-occasion variability (IOV) into the dataset. A new variable of name 'colname' will be output into the dataset and will vary at each dose number according to the given distribution.</i>
-----	--

Description

Define inter-occasion variability (IOV) into the dataset. A new variable of name 'colname' will be output into the dataset and will vary at each dose number according to the given distribution.

Usage

IOV(colname, distribution, doseNumbers = NULL)

Arguments

colname	name of the column that will be output in dataset
distribution	distribution
doseNumbers	dose numbers, if provided, IOV is generated at these doses only. By default, IOV is generated for all doses.

Value

an IOV object

length, arm-method *Return the number of subjects contained in this arm.*

Description

Return the number of subjects contained in this arm.

Usage

```
## S4 method for signature 'arm'
length(x)
```

Arguments

x	arm
---	-----

Value

a number

length, dataset-method *Return the number of subjects contained in this dataset.*

Description

Return the number of subjects contained in this dataset.

Usage

```
## S4 method for signature 'dataset'
length(x)
```

Arguments

x	dataset
---	---------

Value

a number

LogNormalDistribution *Create a log normal distribution.*

Description

Create a log normal distribution.

Usage

LogNormalDistribution(meanlog, sdlog)

Arguments

meanlog	mean value of distribution in log domain
sdlog	standard deviation of distribution in log domain

Value

a log normal distribution

minutes *Convert minutes to hours.*

Description

Convert minutes to hours.

Usage

minutes(x)

Arguments

x	numeric vector in minutes
---	---------------------------

Value

numeric vector in hours

months	<i>Convert pharma months (1 month = 4 weeks) to hours.</i>
--------	--

Description

Convert pharma months (1 month = 4 weeks) to hours.

Usage

months(x)

Arguments

x numeric vector in months

Value

numeric vector in hours

mrgsolve_engine-class	<i>mrgsolve engine class.</i>
-----------------------	-------------------------------

Description

mrgsolve engine class.

nhanes	<i>NHANES database (demographics and body measure data combined, from 2017-2018).</i>
--------	---

Description

NHANES database (demographics and body measure data combined, from 2017-2018).

Usage

nhanes

Format

data frame

BS_ID Original identifier

SEX Sex: 1 for males, 2 for females

AGE Age in years

BW Body weight in kg

BMI Body mass index

HT Height in cm

Source

https://wwwn.cdc.gov/Nchs/Nhanes/2017-2018/DEMO_J.XPT

https://wwwn.cdc.gov/Nchs/Nhanes/2017-2018/BMX_J.XPT

NOCB

Create NOCB settings.

Description

Create NOCB settings.

Usage

```
NOCB(enable = NULL, variables = character(0))
```

Arguments

enable enable/disable next-observation carried backward mode (NOCB), default value is TRUE for mrgsolve, FALSE for RxODE

variables variable names subject to NOCB behavior (see vignette for more info)

Value

NOCB settings

nocb_settings-class *NOCB settings class.*

Description

NOCB settings class.

Slots

enable enable/disable next-observation carried backward mode (NOCB), default value is TRUE for mrgsolve, FALSE for RxODE

variables variable names subject to NOCB behavior (see vignette for more info)

NormalDistribution *Create a normal distribution.*

Description

Create a normal distribution.

Usage

NormalDistribution(mean, sd)

Arguments

mean mean value of distribution

sd standard deviation of distribution

Value

a normal distribution

Observations	<i>Create an observations list. Please note that the provided 'times' will automatically be sorted. Duplicated times will be removed.</i>
--------------	---

Description

Create an observations list. Please note that the provided 'times' will automatically be sorted. Duplicated times will be removed.

Usage

```
Observations(times, compartment = NA)
```

Arguments

times	observation times, numeric vector
compartment	compartment index, integer

Value

an observations list

observations-class	<i>Observations class.</i>
--------------------	----------------------------

Description

Observations class.

Slots

times	observation times, numeric vector
compartment	compartment index, integer
dv	observed values, numeric vector (FOR EXTERNAL USE)

observations_set-class	<i>Observations set class.</i>
------------------------	--------------------------------

Description

Observations set class.

obsOnly	<i>Filter CAMPSIS output on observation rows.</i>
---------	---

Description

Filter CAMPSIS output on observation rows.

Usage

```
obsOnly(x)
```

Arguments

x data frame, CAMPSIS output

Value

a data frame with the observation rows

Occasion	<i>Define a new occasion. Occasions are defined by mapping occasion values to dose numbers. A new column will automatically be created in the exported dataset.</i>
----------	---

Description

Define a new occasion. Occasions are defined by mapping occasion values to dose numbers. A new column will automatically be created in the exported dataset.

Usage

```
Occasion(colname, values, doseNumbers)
```

Arguments

colname name of the column that will be output in dataset
 values the occasion numbers, any integer vector
 doseNumbers the related dose numbers, any integer vector of same length as 'values'

Value

occasion object

occasion-class	<i>Occasion class.</i>
----------------	------------------------

Description

Occasion class.

Slots

colname single character value representing the column name related to this occasion

values occasion values, integer vector, same length as dose_numbers

dose_numbers associated dose numbers, integer vector, same length as values

occasions-class	<i>Occasions class.</i>
-----------------	-------------------------

Description

Occasions class.

Outfun	<i>Create a new output function</i>
--------	-------------------------------------

Description

Create a new output function

Usage

```
Outfun(
  fun = function(x, ...) {
    x
  },
  args = list(),
  packages = NULL,
  level = "scenario"
)
```

Arguments

fun	function or purrr-style lambda formula, first argument 'x' must be the results
args	extra arguments, named list
packages	packages that must be loaded to execute the given function, character vector
level	either 'scenario' or 'replicate'. Default is 'scenario'.

Value

an output function

output_function-class *Output function class.*

Description

Output function class.

Slots

fun function or purrr-style lambda formula, first argument 'x' must be the results
 args extra arguments, named list
 packages packages that must be loaded to execute the given function, character vector
 level either 'scenario' or 'replicate'. Default is 'scenario'.

ParameterDistribution *Create a parameter distribution. The resulting distribution is a log-normal distribution, with meanlog=log(THETA) and sdlog=sqrt(OMEGA).*

Description

Create a parameter distribution. The resulting distribution is a log-normal distribution, with meanlog=log(THETA) and sdlog=sqrt(OMEGA).

Usage

```
ParameterDistribution(model, theta, omega = NULL)
```

Arguments

model	model
theta	corresponding THETA name, character
omega	corresponding OMEGA name, character, NULL if not defined

Value

a parameter distribution

PI *Compute the prediction interval summary over time.*

Description

Compute the prediction interval summary over time.

Usage

```
PI(x, output, scenarios = NULL, level = 0.9, gather = TRUE)
```

Arguments

x	data frame
output	variable to show, character value
scenarios	scenarios, character vector, NULL is default
level	PI level, default is 0.9 (90% PI)
gather	FALSE: med, low & up columns, TRUE: metric column

Value

a summary table

Progress *Create progress settings.*

Description

Create progress settings.

Usage

```
Progress(tick_slice = TRUE)
```

Arguments

tick_slice	tick() is called after each simulated slice, default is TRUE. In some cases, when the number of subjects per slice is low, it may be useful disable this flag, to improve performance issues.
------------	---

Value

progress settings

progress_settings-class

Progress settings class.

Description

Progress settings class.

Slots

tick_slice tick() is called after each simulated slice, default is TRUE. In some cases, when the number of subjects per slice is low, it may be useful disable this flag, to improve performance issues.

protocol-class

Protocol class.

Description

Protocol class.

retrieveParameterValue

Retrieve the parameter value (standardized) for the specified parameter name.

Description

Retrieve the parameter value (standardized) for the specified parameter name.

Usage

```
retrieveParameterValue(model, paramName, default = NULL, mandatory = FALSE)
```

Arguments

model	model
paramName	parameter name
default	default value if not found
mandatory	must be in model or not

Value

the standardized parameter value or the given default value if not found

rxode_engine-class	<i>RxODE/rxode2 engine class.</i>
--------------------	-----------------------------------

Description

RxODE/rxode2 engine class.

Slots

rxode2 logical field to indicate if CAMPSIS should use rxode2 (field set to TRUE) or RxODE (field set to FALSE). Default is TRUE.

sample	<i>Sample generic object.</i>
--------	-------------------------------

Description

Sample generic object.

Usage

```
sample(object, n, ...)

## S4 method for signature 'constant_distribution,integer'
sample(object, n)

## S4 method for signature 'fixed_distribution,integer'
sample(object, n)

## S4 method for signature 'function_distribution,integer'
sample(object, n)

## S4 method for signature 'bootstrap_distribution,integer'
sample(object, n)

## S4 method for signature 'bolus,integer'
sample(object, n, ...)

## S4 method for signature 'infusion,integer'
sample(object, n, ...)

## S4 method for signature 'observations,integer'
sample(object, n, ...)

## S4 method for signature 'covariate,integer'
```

```
sample(object, n)

## S4 method for signature 'bootstrap,integer'
sample(object, n)
```

Arguments

object	generic object
n	number of samples required
...	extra arguments

Value

sampling result

scatterPlot	<i>Scatter plot (or X vs Y plot).</i>
-------------	---------------------------------------

Description

Scatter plot (or X vs Y plot).

Usage

```
scatterPlot(x, output, colour = NULL, time = NULL)
```

Arguments

x	data frame
output	the 2 variables to show, character vector
colour	variable(s) to colour
time	the time to look at those 2 variables, if NULL, min time is used (usually 0)

Value

a ggplot object

Scenario	<i>Create an scenario.</i>
----------	----------------------------

Description

Create an scenario.

Usage

```
Scenario(name = NULL, model = NULL, dataset = NULL)
```

Arguments

name	scenario name, single character string
model	either a CAMPSIS model, a function or lambda-style formula
dataset	either a CAMPSIS dataset, a function or lambda-style formula

Value

a new scenario

scenario-class	<i>Scenario class.</i>
----------------	------------------------

Description

Scenario class.

Slots

name	scenario name, single character string
model	either a CAMPSIS model, a function or lambda-style formula
dataset	either a CAMPSIS dataset, a function or lambda-style formula

Scenarios	<i>Create a list of scenarios.</i>
-----------	------------------------------------

Description

Create a list of scenarios.

Usage

Scenarios()

Value

a scenarios object

scenarios-class	<i>Scenarios class.</i>
-----------------	-------------------------

Description

Scenarios class.

seconds	<i>Convert seconds to hours.</i>
---------	----------------------------------

Description

Convert seconds to hours.

Usage

seconds(x)

Arguments

x numeric vector in seconds

Value

numeric vector in hours

setLabel	<i>Set the label.</i>
----------	-----------------------

Description

Set the label.

Usage

```
setLabel(object, x)
```

```
## S4 method for signature 'arm,character'  
setLabel(object, x)
```

Arguments

object	any object that has a label
x	the new label

Value

the updated object

setSubjects	<i>Set the number of subjects.</i>
-------------	------------------------------------

Description

Set the number of subjects.

Usage

```
setSubjects(object, x)
```

```
## S4 method for signature 'arm,integer'  
setSubjects(object, x)
```

```
## S4 method for signature 'dataset,integer'  
setSubjects(object, x)
```

Arguments

object	any object
x	the new number of subjects

Value

the updated object

Settings	<i>Create advanced simulation settings.</i>
----------	---

Description

Create advanced simulation settings.

Usage

Settings(...)

Arguments

... any user-required settings: see ?Hardware, ?Solver, ?NOCB, ?Declare, ?Progress or ?AutoReplicationSettings

Value

advanced simulation settings

setupPlanDefault	<i>Setup default plan for the given simulation or hardware settings. This plan will prioritise the distribution of workers in the following order: 1) Replicates (if 'replicate_parallel' is enabled) 2) Scenarios (if 'scenario_parallel' is enabled) 3) Dataset export / slices (if 'dataset_export' or 'slice_parallel' is enabled)</i>
------------------	--

Description

Setup default plan for the given simulation or hardware settings. This plan will prioritise the distribution of workers in the following order: 1) Replicates (if 'replicate_parallel' is enabled) 2) Scenarios (if 'scenario_parallel' is enabled) 3) Dataset export / slices (if 'dataset_export' or 'slice_parallel' is enabled)

Usage

setupPlanDefault(object)

Arguments

object simulation or hardware settings

Value

nothing

setupPlanSequential *Setup plan as sequential (i.e. no parallelisation).*

Description

Setup plan as sequential (i.e. no parallelisation).

Usage

```
setupPlanSequential()
```

Value

nothing

shadedPlot *Shaded plot (or prediction interval plot).*

Description

Shaded plot (or prediction interval plot).

Usage

```
shadedPlot(  
  x,  
  output,  
  colour = NULL,  
  strat_extra = NULL,  
  level = 0.9,  
  alpha = 0.25  
)
```

Arguments

x	data frame
output	variable to show
colour	variable(s) to colour
strat_extra	variable(s) to stratify, but not to colour (useful for use with facet_wrap)
level	PI level, default is 0.9 (90% PI)
alpha	alpha parameter (transparency) given to geom_ribbon

Value

a ggplot object

simulate	<i>Simulate function.</i>
----------	---------------------------

Description

Simulate function.

Usage

```
simulate(  
  model,  
  dataset,  
  dest = NULL,  
  events = NULL,  
  scenarios = NULL,  
  tablefun = NULL,  
  outvars = NULL,  
  outfun = NULL,  
  seed = NULL,  
  replicates = 1,  
  dosing = FALSE,  
  settings = NULL  
)  
  
## S4 method for signature  
## 'replicated_campsis_model,  
## dataset,  
## character,  
## events,  
## scenarios,  
## function,  
## character,  
## output_function,  
## integer,  
## integer,  
## logical,  
## simulation_settings'  
simulate(  
  model,  
  dataset,  
  dest = NULL,  
  events = NULL,  
  scenarios = NULL,  
  tablefun = NULL,  
  outvars = NULL,  
  outfun = NULL,  
  seed = NULL,
```

```
    replicates = 1,
    dosing = FALSE,
    settings = NULL
  )

## S4 method for signature
## 'campsis_model,
## dataset,
## character,
## events,
## scenarios,
## function,
## character,
## output_function,
## integer,
## integer,
## logical,
## simulation_settings'
simulate(
  model,
  dataset,
  dest = NULL,
  events = NULL,
  scenarios = NULL,
  tablefun = NULL,
  outvars = NULL,
  outfun = NULL,
  seed = NULL,
  replicates = 1,
  dosing = FALSE,
  settings = NULL
)

## S4 method for signature
## 'campsis_model,
## tbl_df,
## character,
## events,
## scenarios,
## function,
## character,
## output_function,
## integer,
## integer,
## logical,
## simulation_settings'
simulate(
  model,
```

```
dataset,
dest = NULL,
events = NULL,
scenarios = NULL,
tablefun = NULL,
outvars = NULL,
outfun = NULL,
seed = NULL,
replicates = 1,
dosing = FALSE,
settings = NULL
)

## S4 method for signature
## 'campsis_model,
## data.frame,
## character,
## events,
## scenarios,
## function,
## character,
## output_function,
## integer,
## integer,
## logical,
## simulation_settings'
simulate(
  model,
  dataset,
  dest = NULL,
  events = NULL,
  scenarios = NULL,
  tablefun = NULL,
  outvars = NULL,
  outfun = NULL,
  seed = NULL,
  replicates = 1,
  dosing = FALSE,
  settings = NULL
)

## S4 method for signature
## 'campsis_model,
## tbl_df,
## rxode_engine,
## events,
## scenarios,
## function,
```



```
## character,  
## output_function,  
## integer,  
## integer,  
## logical,  
## simulation_settings'  
simulate(  
  model,  
  dataset,  
  dest = NULL,  
  events = NULL,  
  scenarios = NULL,  
  tablefun = NULL,  
  outvars = NULL,  
  outfun = NULL,  
  seed = NULL,  
  replicates = 1,  
  dosing = FALSE,  
  settings = NULL  
)  
  
## S4 method for signature  
## 'campsis_model,  
## tbl_df,  
## mrgsolve_engine,  
## events,  
## scenarios,  
## function,  
## character,  
## output_function,  
## integer,  
## integer,  
## logical,  
## simulation_settings'  
simulate(  
  model,  
  dataset,  
  dest = NULL,  
  events = NULL,  
  scenarios = NULL,  
  tablefun = NULL,  
  outvars = NULL,  
  outfun = NULL,  
  seed = NULL,  
  replicates = 1,  
  dosing = FALSE,  
  settings = NULL  
)
```

Arguments

model	generic CAMPSIS model
dataset	CAMPSIS dataset or 2-dimensional table
dest	destination simulation engine, default is 'RxODE'
events	interruption events
scenarios	list of scenarios to be simulated
tablefun	function or lambda formula to apply on exported 2-dimensional dataset
outvars	variables to output in resulting dataframe
outfun	an output function to apply on the simulation results. Type ?Outfun for more info.
seed	seed value
replicates	number of replicates, default is 1
dosing	output dosing information, default is FALSE
settings	advanced simulation settings

Value

dataframe with all results

SimulationProgress *Create a simulation progress object.*

Description

Create a simulation progress object.

Usage

```
SimulationProgress(
  replicates = 1,
  scenarios = 1,
  progressor = NULL,
  hardware = NULL
)
```

Arguments

replicates	total number of replicates to simulate
scenarios	total number of scenarios to simulate
progressor	progressor progressor
hardware	hardware settings

Value

a progress bar

simulation_engine-class
Simulation engine class.

Description

Simulation engine class.

simulation_progress-class
Simulation progress class.

Description

Simulation progress class.

Arguments

replicates	total number of replicates to simulate
scenarios	total number of scenarios to simulate
iterations	total number of iterations to simulate
slices	total number of slices to simulate
replicate	current replicate number being simulated
scenario	current scenario number being simulated
iteration	current iteration number being simulated
slice	current slice number being simulated
progressor	progressr progressor
hardware	hardware settings

```
simulation_settings-class
    Simulation settings class.
```

Description

Simulation settings class.

Slots

```
hardware hardware settings object
solver solver settings object
nocb NOCB settings object
declare declare settings (mrgsolve only)
progress progress settings
replication replication settings
internal internal settings
```

```
Solver
    Create solver settings.
```

Description

Create solver settings.

Usage

```
Solver(
  atol = 1e-08,
  rtol = 1e-08,
  hmax = NA,
  maxsteps = 70000L,
  method = "liblsoda"
)
```

Arguments

atol	absolute solver tolerance, default is 1e-08
rtol	relative solver tolerance, default is 1e-08
hmax	limit how big a solver step can be, default is NA
maxsteps	max steps between 2 integration times (e.g. when observations records are far apart), default is 70000
method	solver method, for RxODE/rxode2 only: 'liblsoda' (default), 'lsoda', 'dop853', 'indLin'. Mrgsolve's method is always 'lsoda'.

Value

solver settings

solver_settings-class *Solver settings class. See ?mrgsolve::update. See ?rxode2::rxSolve.*

Description

Solver settings class. See ?mrgsolve::update. See ?rxode2::rxSolve.

Slots

atol absolute solver tolerance, default is 1e-08

rtol relative solver tolerance, default is 1e-08

hmax limit how big a solver step can be, default is NA

maxsteps max steps between 2 integration times (e.g. when observations records are far apart), default is 70000

method solver method, for RxODE/rxode2 only: 'liblsoda' (default), 'lsoda', 'dop853', 'indLin'. Mrgsolve's method is always 'lsoda'.

spaghettiPlot *Spaghetti plot.*

Description

Spaghetti plot.

Usage

```
spaghettiPlot(x, output, colour = NULL)
```

Arguments

x	data frame
output	variable to show
colour	variable(s) to colour

Value

plot

standardiseTime	<i>Standardise time to hours.</i>
-----------------	-----------------------------------

Description

Standardise time to hours.

Usage

```
standardiseTime(x, unit)
```

Arguments

x	numeric time vector
unit	unit of x, single character value

Value

numeric vector with the times converted to hours

TimeVaryingCovariate	<i>Create a time-varying covariate. This covariate will be implemented using EVID=2 rows in the exported dataset and will not use interruption events.</i>
----------------------	--

Description

Create a time-varying covariate. This covariate will be implemented using EVID=2 rows in the exported dataset and will not use interruption events.

Usage

```
TimeVaryingCovariate(name, table)
```

Arguments

name	covariate name, character
table	data.frame, must contain the mandatory columns 'TIME' and 'VALUE'. An 'ID' column may also be specified. In that case, ID's between 1 and the max number of subjects in the dataset/arm can be used. All ID's must have a VALUE defined for TIME 0.

Value

a time-varying covariate

time_varying_covariate-class
Time-varying covariate class.

Description

Time-varying covariate class.

treatment-class *Treatment class.*

Description

Treatment class.

treatment_iov-class *Treatment IOV class.*

Description

Treatment IOV class.

Slots

colname name of the column that will be output in dataset

distribution distribution

dose_numbers associated dose numbers, integer vector, same length as values

treatment_iovs-class *Treatment IOV's class.*

Description

Treatment IOV's class.

undefined_distribution-class

Undefined distribution class. This type of object is automatically created in method toExplicitDistribution() when the user does not provide a concrete distribution. This is because S4 objects do not accept NULL values.

Description

Undefined distribution class. This type of object is automatically created in method toExplicitDistribution() when the user does not provide a concrete distribution. This is because S4 objects do not accept NULL values.

UniformDistribution *Create an uniform distribution.*

Description

Create an uniform distribution.

Usage

UniformDistribution(min, max)

Arguments

min min value

max max value

Value

an uniform distribution

VPC	<i>Compute the VPC summary. Input data frame must contain the following columns: - replicate: replicate number - low: low percentile value in replicate (and in scenario if present) - med: median value in replicate (and in scenario if present) - up: up percentile value in replicate (and in scenario if present) - any scenario column</i>
-----	--

Description

Compute the VPC summary. Input data frame must contain the following columns: - replicate: replicate number - low: low percentile value in replicate (and in scenario if present) - med: median value in replicate (and in scenario if present) - up: up percentile value in replicate (and in scenario if present) - any scenario column

Usage

```
VPC(x, scenarios = NULL, level = 0.9)
```

Arguments

x	data frame
scenarios	scenarios, character vector, NULL is default
level	PI level, default is 0.9 (90% PI)

Value

VPC summary with columns TIME, <scenarios> and all combinations of low, med, up (i.e. low_low, low_med, low_up, etc.)

vpcPlot	<i>VPC plot.</i>
---------	------------------

Description

VPC plot.

Usage

```
vpcPlot(x, scenarios = NULL, level = 0.9, alpha = 0.15)
```

Arguments

x	data frame, output of CAMPSIS with replicates
scenarios	scenarios, character vector, NULL is default
level	PI level, default is 0.9 (90% PI)
alpha	alpha parameter (transparency) given to geom_ribbon

Value

a ggplot object

weeks	<i>Convert weeks to hours.</i>
-------	--------------------------------

Description

Convert weeks to hours.

Usage

weeks(x)

Arguments

x numeric vector in weeks

Value

numeric vector in hours

years	<i>Convert pharma years (1 year = 12*4 weeks) to hours.</i>
-------	---

Description

Convert pharma years (1 year = 12*4 weeks) to hours.

Usage

years(x)

Arguments

x numeric vector in years

Value

numeric vector in hours

Index

* datasets

- nhanes, [38](#)

- [applyCompartmentCharacteristics](#), [5](#)
- Arm, [5](#)
- arm-class, [6](#)
- arms-class, [6](#)

- [BinomialDistribution](#), [7](#)
- Bolus, [7](#)
- bolus-class, [8](#)
- Bootstrap, [8](#)
- bootstrap-class, [9](#)
- bootstrap_distribution-class, [10](#)
- [BootstrapDistribution](#), [9](#)

- [campsis_handler](#), [10](#)
- constant_distribution-class, [11](#)
- [ConstantDistribution](#), [11](#)
- convertTime, [12](#)
- Covariate, [12](#)
- covariate-class, [13](#)
- covariates-class, [13](#)

- [Dataset](#), [13](#)
- dataset-class, [14](#)
- dataset_config-class, [15](#)
- [DatasetConfig](#), [14](#)
- days, [15](#)
- Declare, [16](#)
- declare_settings-class, [16](#)
- [DiscreteDistribution](#), [16](#)
- distribution-class, [17](#)
- dose_adaptation-class, [18](#)
- dose_adaptations-class, [18](#)
- [DoseAdaptation](#), [17](#)
- dosingOnly, [18](#)

- [EtaDistribution](#), [19](#)
- Event, [19](#)
- event-class, [20](#)

- event_covariate-class, [21](#)
- [EventCovariate](#), [20](#)
- Events, [21](#)
- events-class, [21](#)

- fixed_covariate-class, [22](#)
- fixed_distribution-class, [22](#)
- [FixedDistribution](#), [22](#)
- function_distribution-class, [23](#)
- [FunctionDistribution](#), [23](#)

- [generateIIV](#), [24](#)
- [generateIIV_](#), [24](#)
- [getAvailableTimeUnits](#), [25](#)
- [getCovariates](#), [25](#)
- [getCovariates](#), arm-method
([getCovariates](#)), [25](#)
- [getCovariates](#), arms-method
([getCovariates](#)), [25](#)
- [getCovariates](#), covariates-method
([getCovariates](#)), [25](#)
- [getCovariates](#), dataset-method
([getCovariates](#)), [25](#)
- [getEventCovariates](#), [26](#)
- [getEventCovariates](#), arm-method
([getEventCovariates](#)), [26](#)
- [getEventCovariates](#), arms-method
([getEventCovariates](#)), [26](#)
- [getEventCovariates](#), covariates-method
([getEventCovariates](#)), [26](#)
- [getEventCovariates](#), dataset-method
([getEventCovariates](#)), [26](#)
- [getFixedCovariates](#), [26](#)
- [getFixedCovariates](#), arm-method
([getFixedCovariates](#)), [26](#)
- [getFixedCovariates](#), arms-method
([getFixedCovariates](#)), [26](#)
- [getFixedCovariates](#), covariates-method
([getFixedCovariates](#)), [26](#)

- getFixedCovariates, dataset-method (getFixedCovariates), 26
- getIOVs, 27
- getIOVs, arm-method (getIOVs), 27
- getIOVs, arms-method (getIOVs), 27
- getIOVs, dataset-method (getIOVs), 27
- getOccasions, 28
- getOccasions, arm-method (getOccasions), 28
- getOccasions, arms-method (getOccasions), 28
- getOccasions, dataset-method (getOccasions), 28
- getSeedForDatasetExport, 28
- getSeedForIteration, 29
- getSeedForParametersSampling, 29
- getSplittingConfiguration, 30
- getTimes, 30
- getTimes, arm-method (getTimes), 30
- getTimes, arms-method (getTimes), 30
- getTimes, dataset-method (getTimes), 30
- getTimes, events-method (getTimes), 30
- getTimes, observations_set-method (getTimes), 30
- getTimeVaryingCovariates, 31
- getTimeVaryingCovariates, arm-method (getTimeVaryingCovariates), 31
- getTimeVaryingCovariates, arms-method (getTimeVaryingCovariates), 31
- getTimeVaryingCovariates, covariates-method (getTimeVaryingCovariates), 31
- getTimeVaryingCovariates, dataset-method (getTimeVaryingCovariates), 31
- Hardware, 32
- hardware_settings-class, 33
- hours, 33
- Infusion, 34
- infusion-class, 35
- internal_settings-class, 35
- IOV, 35
- length, arm-method, 36
- length, dataset-method, 36
- LogNormalDistribution, 37
- minutes, 37
- months, 38
- mrgsolve_engine-class, 38
- nhanes, 38
- NOCB, 39
- nocb_settings-class, 40
- NormalDistribution, 40
- Observations, 41
- observations-class, 41
- observations_set-class, 41
- obsOnly, 42
- Occasion, 42
- occasion-class, 43
- occasions-class, 43
- Outfun, 43
- output_function-class, 44
- ParameterDistribution, 44
- PI, 45
- Progress, 45
- progress_settings-class, 46
- protocol-class, 46
- retrieveParameterValue, 46
- rxode_engine-class, 47
- sample, 47
- sample, bolus, integer-method (sample), 47
- sample, bootstrap, integer-method (sample), 47
- sample, bootstrap_distribution, integer-method (sample), 47
- sample, constant_distribution, integer-method (sample), 47
- sample, covariate, integer-method (sample), 47
- sample, fixed_distribution, integer-method (sample), 47
- sample, function_distribution, integer-method (sample), 47
- sample, infusion, integer-method (sample), 47
- sample, observations, integer-method (sample), 47
- scatterPlot, 48
- Scenario, 49
- scenario-class, 49
- Scenarios, 50
- scenarios-class, 50

seconds, [50](#)
setLabel, [51](#)
setLabel, arm, character-method
 (setLabel), [51](#)
setSubjects, [51](#)
setSubjects, arm, integer-method
 (setSubjects), [51](#)
setSubjects, dataset, integer-method
 (setSubjects), [51](#)
Settings, [52](#)
setupPlanDefault, [52](#)
setupPlanSequential, [53](#)
shadedPlot, [53](#)
simulate, [54](#)
simulate, campsis_model, data.frame, character, events, scenarios, function, character, output_function, integer
 (simulate), [54](#)
simulate, campsis_model, dataset, character, events, scenarios, function, character, output_function, integer, in
 (simulate), [54](#)
simulate, campsis_model, tbl_df, character, events, scenarios, function, character, output_function, integer, int
 (simulate), [54](#)
simulate, campsis_model, tbl_df, mrgsolve_engine, events, scenarios, function, character, output_function, integ
 (simulate), [54](#)
simulate, campsis_model, tbl_df, rxode_engine, events, scenarios, function, character, output_function, integer,
 (simulate), [54](#)
simulate, replicated_campsis_model, dataset, character, events, scenarios, function, character, output_function
 (simulate), [54](#)
simulation_engine-class, [59](#)
simulation_progress-class, [59](#)
simulation_settings-class, [60](#)
SimulationProgress, [58](#)
Solver, [60](#)
solver_settings-class, [61](#)
spaghettiPlot, [61](#)
standardiseTime, [62](#)

time_varying_covariate-class, [63](#)
TimeVaryingCovariate, [62](#)
treatment-class, [63](#)
treatment_iov-class, [63](#)
treatment_iovs-class, [63](#)

undefined_distribution-class, [64](#)
UniformDistribution, [64](#)

VPC, [65](#)
vpcPlot, [65](#)

weeks, [66](#)

years, [66](#)