

# Package ‘campsis’

February 16, 2024

**Type** Package

**Title** Generic PK/PD Simulation Platform CAMPSIS

**Version** 1.5.1

**Description** A generic, easy-to-use and intuitive pharmacokinetic/pharmacodynamic (PK/PD) simulation platform based on R packages 'rxode2', 'RxODE' 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', 'RxODE' 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/>

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

**Depends** campsismod (>= 1.1.0), R (>= 4.0.0)

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

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

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.2.3

**Collate** 'global.R' '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' 'parameter\_uncertainty.R'  
 'event\_logic.R' 'dataset\_summary.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** 2024-02-16 11:20:08 UTC

## R topics documented:

applyCompartmentCharacteristics . . . . .	4
Arm . . . . .	5
arm-class . . . . .	6
arms-class . . . . .	6
BinomialDistribution . . . . .	6
Bolus . . . . .	7
bolus-class . . . . .	7
Bootstrap . . . . .	8
bootstrap-class . . . . .	8
BootstrapDistribution . . . . .	9
bootstrap_distribution-class . . . . .	9
campsis_handler . . . . .	10
ConstantDistribution . . . . .	10
constant_distribution-class . . . . .	10
Covariate . . . . .	11
covariate-class . . . . .	11
covariates-class . . . . .	11
Dataset . . . . .	12
dataset-class . . . . .	12
DatasetConfig . . . . .	13
dataset_config-class . . . . .	13
Declare . . . . .	14
declare_settings-class . . . . .	14
DiscreteDistribution . . . . .	14
distribution-class . . . . .	15
DoseAdaptation . . . . .	15
dose_adaptation-class . . . . .	16

dose_adaptations-class	16
dosingOnly	16
EtaDistribution	17
Event	17
event-class	18
EventCovariate	18
Events	19
events-class	19
event_covariate-class	19
FixedDistribution	20
fixed_covariate-class	20
fixed_distribution-class	20
FunctionDistribution	21
function_distribution-class	21
generateIIV	22
generateIIV_	22
getCovariates	23
getEventCovariates	23
getFixedCovariates	24
getIOVs	25
getOccasions	25
getSeedForDatasetExport	26
getSeedForIteration	26
getSeedForParametersSampling	27
getSplittingConfiguration	27
getTimes	28
getTimeVaryingCovariates	28
Hardware	29
hardware_settings-class	30
Infusion	31
infusion-class	32
internal_settings-class	32
IOV	32
length,arm-method	33
length,dataset-method	33
LogNormalDistribution	34
mrgsolve_engine-class	34
nhanes	35
NOCB	35
nocb_settings-class	36
NormalDistribution	36
Observations	37
observations-class	37
observations_set-class	37
obsOnly	38
Occasion	38
occasion-class	39
occasions-class	39

ParameterDistribution . . . . .	39
PI . . . . .	40
Progress . . . . .	40
progress_settings-class . . . . .	41
protocol-class . . . . .	41
retrieveParameterValue . . . . .	41
rxode_engine-class . . . . .	42
sample . . . . .	42
scatterPlot . . . . .	43
Scenario . . . . .	44
scenario-class . . . . .	44
Scenarios . . . . .	45
scenarios-class . . . . .	45
setLabel . . . . .	45
setSubjects . . . . .	46
Settings . . . . .	46
setupPlanDefault . . . . .	47
setupPlanSequential . . . . .	47
shadedPlot . . . . .	48
simulate . . . . .	48
SimulationProgress . . . . .	52
simulation_engine-class . . . . .	53
simulation_progress-class . . . . .	53
simulation_settings-class . . . . .	54
Solver . . . . .	54
solver_settings-class . . . . .	55
spaghettiPlot . . . . .	55
TimeVaryingCovariate . . . . .	56
time_varying_covariate-class . . . . .	56
treatment-class . . . . .	56
treatment_iov-class . . . . .	57
treatment_iovs-class . . . . .	57
undefined_distribution-class . . . . .	57
UniformDistribution . . . . .	58
VPC . . . . .	58
vpcPlot . . . . .	59

**Index** **60**

---

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**

table	current dataset
properties	compartment properties from model

**Value**

updated dataset

---

Arm	<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	<i>Binomial distribution.</i>
----------------------	-------------------------------

---

**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	<i>Create one or several bolus(es).</i>
-------	---

---

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

---

**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	<i>Suggested Campsis handler for showing the progress bar.</i>
-----------------	--

---

**Description**

Suggested Campsis handler for showing the progress bar.

**Usage**

```
campsis_handler()
```

**Value**

a progressr handler list

---

ConstantDistribution	<i>Create a constant distribution. Its value will be constant across all generated samples.</i>
----------------------	---

---

**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	<i>Constant distribution class.</i>
-----------------------------	-------------------------------------

---

**Description**

Constant distribution class.

**Slots**

value covariate value, single numeric value

---

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)
```

**Arguments**

subjects      number of subjects in the default arm

**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
)
```

**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

**Value**

a dataset configuration

---

dataset_config-class	<i>Dataset configuration class.</i>
----------------------	-------------------------------------

---

**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

---

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

---

**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      *Fixed covariate class.*

---

**Description**

Fixed covariate class.

---

fixed\_distribution-class      *Fixed distribution class.*

---

**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

---

getCovariates	<i>Get all covariates (fixed / time-varying / event covariates).</i>
---------------	--

---

**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	<i>Get all event-related covariates.</i>
--------------------	--

---

**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

---

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

---

**Description**

Get all occasions.

**Usage**

```
getOccasions(object)

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

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

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

**Arguments**

object            any object

**Value**

all occasions from object

---

getSeedForDatasetExport

*Get seed for dataset export.*

---

**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**

seed            original seed  
 progress       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**

seed                    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)

---

getTime	<i>Get all distinct times for the specified object.</i>
---------	---

---

**Description**

Get all distinct times for the specified object.

**Usage**

```
getTime(object)

## S4 method for signature 'observations_set'
getTime(object)

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

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

## S4 method for signature 'events'
getTime(object)

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

**Arguments**

object            any object

**Value**

numeric vector with all unique times, sorted

---

getTimeVaryingCovariates	<i>Get all time-varying covariates.</i>
--------------------------	---

---

**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	<i>Create hardware settings.</i>
----------	----------------------------------

---

**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

---

Infusion	<i>Create one or several infusion(s).</i>
----------	---

---

**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

---

mrgsolve\_engine-class *mrgsolve engine class.*

---

**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

**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/DEMO_J.XPT)

[https://wwwn.cdc.gov/Nchs/Nhanes/2017-2018/BMX\\_J.XPT](https://wwwn.cdc.gov/Nchs/Nhanes/2017-2018/BMX_J.XPT)

---

NOCB	<i>Create NOCB settings.</i>
------	------------------------------

---

**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.

---

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

---

**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

<code>model</code>	model
<code>paramName</code>	parameter name
<code>default</code>	default value if not found
<code>mandatory</code>	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)

## S4 method for signature 'campsis_model,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.

---

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 or ?Progress settings
-----	---

**Value**

advanced simulation settings

---

setupPlanDefault	<i>Setup default plan for the given simulation or hardware settings. This plan will prioritise the ditribution 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 ditribution 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	<i>Setup plan as sequential (i.e. no parallelisation).</i>
---------------------	--

---

**Description**

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

**Usage**

```
setupPlanSequential()
```

**Value**

nothing

---

shadedPlot	<i>Shaded plot (or prediction interval plot).</i>
------------	---

---

### 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  
## 'campsis_model,  
##   dataset,  
##   character,  
##   events,  
##   scenarios,  
##   `function`,  
##   character,  
##   `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,
## `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,
## `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,
## `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,
## `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	function or lambda formula to apply on resulting dataframe after each replicate
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	progressr 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  
 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

---

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

---

### 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

# Index

## \* datasets

- nhanes, 35
  
- applyCompartmentCharacteristics, 4
- Arm, 5
- arm-class, 6
- arms-class, 6
  
- BinomialDistribution, 6
- Bolus, 7
- bolus-class, 7
- Bootstrap, 8
- bootstrap-class, 8
- bootstrap\_distribution-class, 9
- BootstrapDistribution, 9
  
- campsis\_handler, 10
- constant\_distribution-class, 10
- ConstantDistribution, 10
- Covariate, 11
- covariate-class, 11
- covariates-class, 11
  
- Dataset, 12
- dataset-class, 12
- dataset\_config-class, 13
- DatasetConfig, 13
- Declare, 14
- declare\_settings-class, 14
- DiscreteDistribution, 14
- distribution-class, 15
- dose\_adaptation-class, 16
- dose\_adaptations-class, 16
- DoseAdaptation, 15
- dosingOnly, 16
  
- EtaDistribution, 17
- Event, 17
- event-class, 18
- event\_covariate-class, 19
- EventCovariate, 18
  
- Events, 19
- events-class, 19
  
- fixed\_covariate-class, 20
- fixed\_distribution-class, 20
- FixedDistribution, 20
- function\_distribution-class, 21
- FunctionDistribution, 21
  
- generateIIV, 22
- generateIIV\_, 22
- getCovariates, 23
- getCovariates, arm-method (getCovariates), 23
- getCovariates, arms-method (getCovariates), 23
- getCovariates, covariates-method (getCovariates), 23
- getCovariates, dataset-method (getCovariates), 23
- getEventCovariates, 23
- getEventCovariates, arm-method (getEventCovariates), 23
- getEventCovariates, arms-method (getEventCovariates), 23
- getEventCovariates, covariates-method (getEventCovariates), 23
- getEventCovariates, dataset-method (getEventCovariates), 23
- getFixedCovariates, 24
- getFixedCovariates, arm-method (getFixedCovariates), 24
- getFixedCovariates, arms-method (getFixedCovariates), 24
- getFixedCovariates, covariates-method (getFixedCovariates), 24
- getFixedCovariates, dataset-method (getFixedCovariates), 24
- getIOVs, 25
- getIOVs, arm-method (getIOVs), 25

- getIOVs, arms-method (getIOVs), 25
- getIOVs, dataset-method (getIOVs), 25
- getOccasions, 25
- getOccasions, arm-method (getOccasions), 25
- getOccasions, arms-method (getOccasions), 25
- getOccasions, dataset-method (getOccasions), 25
- getSeedForDatasetExport, 26
- getSeedForIteration, 26
- getSeedForParametersSampling, 27
- getSplittingConfiguration, 27
- getTimes, 28
- getTimes, arm-method (getTimes), 28
- getTimes, arms-method (getTimes), 28
- getTimes, dataset-method (getTimes), 28
- getTimes, events-method (getTimes), 28
- getTimes, observations\_set-method (getTimes), 28
- getTimeVaryingCovariates, 28
- getTimeVaryingCovariates, arm-method (getTimeVaryingCovariates), 28
- getTimeVaryingCovariates, arms-method (getTimeVaryingCovariates), 28
- getTimeVaryingCovariates, covariates-method (getTimeVaryingCovariates), 28
- getTimeVaryingCovariates, dataset-method (getTimeVaryingCovariates), 28
  
- Hardware, 29
- hardware\_settings-class, 30
  
- Infusion, 31
- infusion-class, 32
- internal\_settings-class, 32
- IOV, 32
  
- length, arm-method, 33
- length, dataset-method, 33
- LogNormalDistribution, 34
  
- mrgsolve\_engine-class, 34
  
- nhanes, 35
- NOCB, 35
- nocb\_settings-class, 36
- NormalDistribution, 36
  
- Observations, 37
- observations-class, 37
- observations\_set-class, 37
- obsOnly, 38
- Occasion, 38
- occasion-class, 39
- occasions-class, 39
  
- ParameterDistribution, 39
- PI, 40
- Progress, 40
- progress\_settings-class, 41
- protocol-class, 41
  
- retrieveParameterValue, 41
- rxode\_engine-class, 42
  
- sample, 42
- sample, bolus, integer-method (sample), 42
- sample, bootstrap, integer-method (sample), 42
- sample, bootstrap\_distribution, integer-method (sample), 42
- sample, campsis\_model, integer-method (sample), 42
- sample, constant\_distribution, integer-method (sample), 42
- sample, covariate, integer-method (sample), 42
- sample, fixed\_distribution, integer-method (sample), 42
- sample, function\_distribution, integer-method (sample), 42
- sample, infusion, integer-method (sample), 42
- sample, observations, integer-method (sample), 42
- scatterPlot, 43
- Scenario, 44
- scenario-class, 44
- Scenarios, 45
- scenarios-class, 45
- setLabel, 45
- setLabel, arm, character-method (setLabel), 45
- setSubjects, 46
- setSubjects, arm, integer-method (setSubjects), 46
- setSubjects, dataset, integer-method (setSubjects), 46

