Package ‘PKPDsim’

April 1, 2022

Type Package
Title Tools for Performing Pharmacokinetic-Pharmacodynamic Simulations
Version 1.1.1
Date 2022-03-31
Depends R (>= 3.0.2)
Imports Rcpp (>= 0.12.9), BH, data.table, stringr, MASS, randtoolbox, jsonlite, stats, parallel, magrittr
Suggests httr, testthat (>= 3.0.0), mockery
LinkingTo BH, Rcpp (>= 0.12.9)
Description Simulate dose regimens for pharmacokinetic-pharmacodynamic (PK-PD) models described by differential equation (DE) systems. Simulation using ADVAN-style analytical equations is also supported (Abuhelwa et al. (2015) <doi:10.1016/j.vascn.2015.03.004>).
License MIT + file LICENSE
LazyData TRUE
RoxygenNote 7.1.2
Config/testthat/edition 3
Encoding UTF-8
NeedsCompilation yes
Author Ron Keizer [aut, cre], Jasmine Hughes [aut], Dominic Tong [aut], Kara Woo [aut], InsightRX [cph, fnd]
Maintainer Ron Keizer <ron@insight-rx.com>
Repository CRAN
Date/Publication 2022-04-01 08:10:02 UTC
R topics documented:

PKPDsim-package ................................................. 4
add_quotes ....................................................... 4
add_ruv .......................................................... 4
add_ruv_to_quantile ............................................ 5
adherence_binomial ............................................. 6
adherence_markov ............................................... 6
advan ............................................................. 7
advan_create_data .............................................. 7
advan_parse_output ............................................. 8
advan_process_infusion_doses ................................. 8
analytical_eqn_wrapper ....................................... 9
apply_lagtime ................................................... 9
bioavailability_to_R_code .................................. 10
calculate_parameters ....................................... 11
calc_dydP ....................................................... 12
calc_ss_analytic ............................................... 12
check_iov_specification .................................... 13
check_mixture_model ....................................... 14
compile_sim_cpp .............................................. 14
covariates_table_to_list .................................... 15
covariate_last_obs_only ................................... 16
create_event_table .......................................... 16
create_obs_data .............................................. 17
cv_to_omega ................................................... 18
define_tdm_init_model ...................................... 18
detect_ode_syntax .......................................... 19
f_cov ........................................................... 19
generate_ode_model ......................................... 20
get_parameters_from_code ................................... 20
get_t_obs_from_regimen ...................................... 21
get_var_y ....................................................... 21
ifelse0 .......................................................... 23
is_newer_package ............................................ 24
is_positive_definite ........................................ 24
join_cov_and_par ............................................. 25
join_regimen ................................................... 25
merge_regimen ................................................ 26
model_from_api ............................................... 27
model_library ................................................ 28
mvtnorm2 ....................................................... 28
na_locf .......................................................... 29
new_adherence ............................................... 29
new_covariate ............................................... 30
new_covariate_model ....................................... 31
new_ode_model ............................................... 31
new_regimen ................................................... 34
topics documented:

nlmixr_parse_parameters ........................................ 36
nm_to_regimen ......................................................... 36
now_utc .......................................................... 37
OneCompIVbolus ....................................................... 37
OneCompIVinfusion .................................................... 38
OneCompOral .......................................................... 38
parse_obs_types ....................................................... 39
pkdata ............................................................... 39
pkpdsim_to_nlmixr .................................................. 39
pop_regimen .......................................................... 40
print.covariate ....................................................... 41
print.PKPDsim ........................................................ 41
print.regimen ........................................................ 42
print_list ............................................................ 42
read_model_json ...................................................... 43
regimen_to_nm ....................................................... 43
reparametrize ........................................................ 44
search_replace_in_file .............................................. 44
shift_regimen ........................................................ 45
shift_state_indices .................................................. 45
sim ................................................................. 46
sim_core ............................................................. 49
sim_ode ............................................................... 50
sim_ode_shiny ....................................................... 50
table_to_list ........................................................ 51
test_model ........................................................... 51
test_pointer .......................................................... 52
ThreeCompIVbolus .................................................... 52
ThreeCompIVinfusion ................................................ 53
ThreeCompIVinfusionMetab ......................................... 53
ThreeCompOral ......................................................... 54
ThreeCompOralMetab ................................................ 54
translate_ode ........................................................ 55
triangle_to_full ....................................................... 55
TwoCompIVbolus ...................................................... 56
TwoCompIVinfusion .................................................. 56
TwoCompOral .......................................................... 57
vector_to_R_code ..................................................... 57

Index ............................................................... 58
PKPDsim-package

Description
Simulate regimens for PKPD models defined by ODE systems

Author(s)
Ron Keizer <ronkeizer@gmail.com>

add_quotes

Description
Put vector values in quotes

Usage
add_quotes(x, quote = "double")

Arguments
x vector of string / numeric
quote what type of quotes (‘double’ or ‘single’)

Value
Character vector of input with quotation marks around each value

add_ruv

Description
Add residual variability to the dependent variable

Usage
add_ruv(x, ruv = list(), obs_type = 1)
add_ruv_to_quantile

Arguments

- **x**: dependent value without residual variability
- **ruv**: list specifying proportional, additive and/or exponential errors ('prop', 'add', 'exp')
- **obs_type**: vector of observation types

Value

Input vector with residual variability added

---

**add_ruv_to_quantile** Calculate the increase in a specific quantile for a distribution on y when residual variability is added

Description

Calculate the increase in a specific quantile for a distribution on y when residual variability is added

Usage

```r
add_ruv_to_quantile(y, sd_y, log_scale = FALSE, q = NULL, ruv = list(), ...)
```

Arguments

- **y**: y with
- **sd_y**: standard deviation of y without residual variability added. Will add normally distributed variability (potentially on log-scale).
- **log_scale**: add variability on log scale (FALSE by default, DEPRECATED!).
- **q**: quantile
- **ruv**: list of residual variability ('prop' and 'add')
- **...**: passed arguments

Value

Numeric vector of y values with residual variability
**adherence_binomial**  
*Binomial adherence*

**Description**  
Model adherence as a binomial probability at the time of each occasion.

**Usage**  
`adherence_binomial(n = 100, prob)`

**Arguments**  
- `n`: number of occasions  
- `prob`: binomial probability

**Value**  
Returns a vector of length `n` containing values 0 (non-adherent) or 1 (adherent).  
Numeric vector of length `n`

---

**adherence_markov**  
*Markov adherence model*

**Description**  
Model adherence as a markov chain model, based on the probability of staying adherent and of becoming adherent once non-adherent. Assumes all patients start adherent.

**Usage**  
`adherence_markov(n = 100, p11 = 0.9, p01 = 0.7)`

**Arguments**  
- `n`: number of occasions  
- `p11`: probability of staying adherent  
- `p01`: probability of going from non-adherent to adherent state

**Value**  
Returns a vector of length `n` containing values 0 (non-adherent) or 1 (adherent).  
Numeric vector of length `n`
Description

ADVAN-style functions to calculate linear PK systems

Usage

advan(model, cpp = TRUE)

Arguments

model  Standard linear PK model, e.g. ‘pk_1cmt_iv_bolus’.
cpp  use C++-versions of model (~50x faster than R implementations)

Value

Model function

advan_create_data  Create ADVAN-style dataset

Description

Create ADVAN-style dataset

Usage

advan_create_data(
  regimen,
  parameters,
  cmts = 5,
  t_obs = NULL,
  covariates = NULL,
  covariate_model = NULL
)

Arguments

regimen  PKPDsim regimen
parameters  list of parameters
cmts  number of compartments, minimum is 1. Default is 5, which is enough for most linear PK models. It is OK to have more compartments available than are actually being used.
advan_process_infusion_doses

- **t_obs**: add observation timepoints to dataset
- **covariates**: covariate list
- **covariate_model**: covariate model equations, written in C

**Value**
Data frame of ADVAN-style data

---

**advan_parse_output**
*Internal function to parse the raw output from ADVAN-style functions*

**Description**
Internal function to parse the raw output from ADVAN-style functions

**Usage**

```r
advan_parse_output(data, cmts = 1, t_obs, extra_t_obs = TRUE, regimen)
```

**Arguments**
- **data**: simulation output data
- **cmts**: number of compartments
- **t_obs**: observation times
- **extra_t_obs**: leave extra added dose times in dataset?
- **regimen**: PKPDsim regimen

**Value**
Data frame containing parsed simulation data

---

advan_process_infusion_doses

*Add column RATEALL to ADVAN-style dataset to handle infusions*

**Description**
Function adapted from code from Abuhelwa, Foster, Upton JPET 2015. cleaned up and somewhat optimized. Can potentially be optimized more.

**Usage**

```r
advan_process_infusion_doses(data)
```
**analytical_eqn_wrapper**

**Arguments**

- **data**  
  ADVAN-style dataset, e.g. created using ‘advan_create_data’.

**Value**

Data frame containing additional RATEALL column.

**References**


---

**analytical_eqn_wrapper**

Wrap for using analytical equations with PKPD regimens

**Description**

In development. Needs to be optimized significantly to be useful in production.

**Usage**

```r
analytical_eqn_wrapper(analytical, design = NULL, parameters)
```

**Arguments**

- **analytical**  
  analytical equation, taking parameters ‘amt’, ‘parameters’, and ‘t’, and returning a vector of values for ‘y’
- **design**  
  design dataset created by ‘sim_ode’
- **parameters**  
  list of parameters

---

**apply_lagtime**

Apply lagtime to a regimen

**Description**

Apply lagtime to a regimen

**Usage**

```r
apply_lagtime(regimen, lagtime, parameters, cmt_mapping = NULL)
```
bioavailability_to_R_code

Transforms bioavailability specs into appropriate R code

Arguments

regimen PKPDsim regimen
lagtime lagtime object, either single value / parameter name or vector of values/parameter names for all compartments.
parameters parameter list, required if parameters are specified.
cmt_mapping map of administration types to compartments, e.g. `list("oral" = 1, "infusion" = 2, "bolus" = 2)`.

Value

Original regimen with lagtime added to dose times

Description

Specialized wrapper around `vector_to_R_code` that makes reasonable PK assumptions for when the bioavailability specification is NULL.

Usage

bioavailability_to_R_code(bioav)

Arguments

bioav bioavailability specification, either NULL (assume a value of 1 in all compartments), a single value (assume it applies to all compartments), or a vector of values.

Value

character string of length 1
Calculate model-specific variables using a dummy call to sim_ode()

Description

This is a convenience function for PKPDsim users, it is not used inside the `sim_ode()` function in any way. This function is useful for converting from an estimated parameter to actual parameter, e.g. when clearance is specified as ‘CLi = CL * (WT/70) * (1/CR)’ it can be used to calculate ‘CLi’ without having to write that function a second time in R.

Usage

```r
calculate_parameters(
  ode = NULL,
  parameters = NULL,
  covariates = NULL,
  include_parameters = TRUE,
  include_variables = TRUE,
  ...
)
```

Arguments

- **ode**: PKPDsim model object
- **parameters**: parameter list
- **covariates**: covariate list. Make sure to include covariates at the right time point, since only last observed covariate values are used.
- **include_parameters**: boolean, include parameters?
- **include_variables**: boolean, include variables?
- **...**: arguments to pass on to simulation function

Value

List of model-specific variables
### calc_dydP

**Calculate derivative**

**Description**

Calculate derivative

**Usage**

\[
\text{calc_dydP}(\text{dy}, \text{y}, \text{rel\_delta}, \text{log\_y})
\]

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dy</td>
<td>dy</td>
</tr>
<tr>
<td>y</td>
<td>dependent value</td>
</tr>
<tr>
<td>rel_delta</td>
<td>relative delta</td>
</tr>
<tr>
<td>log_y</td>
<td>logical indicating if the dependent variable is log transformed</td>
</tr>
</tbody>
</table>

### calc_ss_analytic

**Returns the state of a linear PK system at steady state (trough) using analytics equations (so for linear PK systems only).**

**Description**

Basically it performs a PK simulation using analytic equations instead of ODEs to steady state (n=45 days, increased if needed).

**Usage**

\[
\text{calc_ss_analytic}(
  f = "1cmt\_oral",
  dose,
  interval,
  t\_inf = NULL,
  model,
  parameters,
  covariates = NULL,
  map = NULL,
  n\_days = 45,
  n\_transit\_compartments = 0,
  auc = FALSE
)
\]
Arguments

- **f**: analytic equation to use, must be one of `names(advan_funcs)`
- **dose**: dose
- **interval**: interval
- **t_inf**: infusion time
- **model**: PKPDsim model
- **parameters**: parameters list
- **covariates**: covariates list
- **map**: list for remapping parameters, ex: ‘list(CL = "CL", V = "V")’
- **n_days**: number of days at which to assume steady state. Default is 45.
- **n_transit_compartments**: number of transit compartments, will insert n compartments between the first (dose) compartment and the second (central) compartment.
- **auc**: add (empty) AUC compartment at end of state vector?

Details

It can also be used for models with transit compartments, however, the assumption is made that at the end of the dosing interval the amount in the transit compartments is negligible (0).

Value

State vector of a linear pharmacokinetic system at steady state

Description

Inter-occasion variability (IOV) is expected to be supplied as a list with ‘cv’ and ‘n_bins’ specified. ‘cv’ is expected to be a named list with IOV for each PK parameter. This function then checks to ensure that the PK code or ODE code contains an IOV term for each PK parameter specified.

Usage

```r
check_iov_specification(iov, code, pk_code)
```

Arguments

- **iov**: IOV specifications, provided as a nested named list.
- **code**: C++ ODE code, supplied as a string
- **pk_code**: C++ PK code, supplied as a string
check_mixture_model  
*Check that mixture model is specified in right format and within constraints (1 parameter, 2 groups)*

**Description**
Check that mixture model is specified in right format and within constraints (1 parameter, 2 groups)

**Usage**
```r
check_mixture_model(mixture, parameters)
```

**Arguments**
- `mixture`: mixture model specification (as list, e.g. `list("CL" = list(values=c(5, 10), probability=0.3))`)
- `parameters`: vector of parameter names

compile_sim_cpp  
*Compile ODE model to c++ function*

**Description**
Compile ODE model to c++ function

**Usage**
```r
compile_sim_cpp(
code,
dose_code,
 pk_code,
size,
p,
cpp_show_code,
code_init = NULL,
state_init = NULL,
declare_variables = NULL,
variables = NULL,
covariates = NULL,
ob = NULL,
dose = NULL,
iov = NULL,
compile = TRUE,
verbose = FALSE,
as_is = FALSE
)
```
covariates_table_to_list

Arguments

code C++ code ODE system
dose_code C++ code per dose event
pk_code C++ code per any event (similar to SPK)
size size of ODE system
p parameters (list)
cpp_show_code show output c++ function?
code_init code for initialization of state
state_init state init vector
declare_variables variable declaration for all required variables (including user-specified)
variables only the user-specified variables
covariates covariates specification
obs observation specification
dose dose specification
iov iov specification
compile compile or not?
verbose show more output
as_is use C-code as-is, don’t substitute line-endings or shift indices

Value
List containing ODE definition in C++ code and simulation function

Description
Can handle time-varying data too, if ‘t’ or ‘time’ is specified as column

Usage
covariates_table_to_list(covariates_table, covariates_implementation = list())

Arguments
covariates_table
   ‘data.frame‘ with covariates in columns. Potentially with ‘id‘ and ‘t‘ columns
covariates_implementation
   ‘list‘ with implementation method per covariate

Value
List of covariates
covariate_last_obs_only

*Use only last observed covariate values*

**Description**

Use only last observed covariate values

**Usage**

```r
covariate_last_obs_only(covariates)
```

**Arguments**

covariates covariates object

**Value**

List containing same elements as input covariate object but including only the last value for each covariate

---

**create_event_table**

*Create an event table*

**Description**

Create an event table

**Usage**

```r
create_event_table(
  regimen,
  t_max = NULL,
  t_obs = NULL,
  t_tte = NULL,
  t_init = 0,
  p,
  covariates,
  model = NULL,
  obs_type = NULL
)
```
create_obs_data

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regimen</td>
<td>regimen</td>
</tr>
<tr>
<td>t_max</td>
<td>t_max</td>
</tr>
<tr>
<td>t_obs</td>
<td>t_obs</td>
</tr>
<tr>
<td>t_tte</td>
<td>t_tte</td>
</tr>
<tr>
<td>t_init</td>
<td>t_init</td>
</tr>
<tr>
<td>p</td>
<td>parameters</td>
</tr>
<tr>
<td>covariates</td>
<td>covariates</td>
</tr>
<tr>
<td>model</td>
<td>model</td>
</tr>
<tr>
<td>obs_type</td>
<td>observation type</td>
</tr>
</tbody>
</table>

create_obs_data Create obs data

Description

Used by sim() to arrange data from ode() function into the correct format.

Usage

create_obs_data(ode_data, obs_attr, id)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ode_data</td>
<td>data frame of output from ode() function</td>
</tr>
<tr>
<td>obs_attr</td>
<td>&quot;obs&quot; attribute from ode() function</td>
</tr>
<tr>
<td>id</td>
<td>ID of the individual</td>
</tr>
</tbody>
</table>

See Also

sim()
cv_to_omega

Create lower-diagonal omega matrix from CV for parameter estimates

Description

Create lower-diagonal omega matrix from CV for parameter estimates

Usage

cv_to_omega(par_cv = NULL, parameters = NULL)

Arguments

par_cv
   list of parameter CVs
parameters
   list of parameters

Value

a vector describing the lower triangle of the omega (between-subject variability) matrix

See Also

sim_ode

define_tdm_init_model

defines C code for TDM before dose conditions

Description

Currently only available for 1-cmt and 2-cmt IV models

Usage

define_tdm_init_model(def)

Arguments

def
   model definition, named recursive list with at least the objects 'misc$model_type', 'parameters' and 'variables'

Value

model definition with 'state_init' object added describing how to initializing the compartments.
detect_ode_syntax

Auto-detect the syntax for the ODE code

Description

Either PKPDsim or RxODE

Usage

detect_ode_syntax(code)

Arguments

code character string with ODE code

Value

List with elements from and to indicating the syntax for the ODE code

f_cov covariate function builder

Description
covariate function builder

Usage

f_cov(...)

Arguments

... parameters to pass to cov

Value

Covariate function
get_ode_model_size

*Get the number of states in the ODE from the code code C++ code for model*

**Description**

Get the number of states in the ODE from the code code C++ code for model

**Usage**

```
get_ode_model_size(code)
```

**Arguments**

- `code` C++ code

**Value**

Number of states in the ODE model

---

get_parameters_from_code

*Get model parameters from code*

**Description**

Get model parameters from code

**Usage**

```
get_parameters_from_code(code, state_init, declare_variables = NULL)
```

**Arguments**

- `code` code
- `state_init` state init vector
- `declare_variables` declared variables

**Value**

Vector of parameter names
get_t_obs_from_regimen

Extract sensible default observation times from a specified regimen

Description

Extract sensible default observation times from a specified regimen

Usage

```r
get_t_obs_from_regimen(
  regimen = NULL,
  obs_step_size = NULL,
  t_max = NULL,
  covariates = NULL,
  extra_t_obs = NULL,
  t_init = 0
)
```

Arguments

- `regimen`: regimen created using `new_regimen()`
- `obs_step_size`: step size between observations. Will be auto-calculated if NULL
- `t_max`: max time value
- `covariates`: covariates object, created using `list(new_covariate(), ...)`
- `extra_t_obs`: add timepoints to _t_obs at which covariate is changing (‘T’/‘F’)
- `t_init`: time of initialization of the ODE system. Usually 0.

get_var_y

*Get expected variance/sd/ci of dependent variable based on PKPDsim model, parameters, and regimen*

Description

Get expected variance/sd/ci of dependent variable based on PKPDsim model, parameters, and regimen
get_var_y

Usage

get_var_y(
    model = NULL,
    parameters = list(),
    regimen = list(),
    t_obs = c(1:48),
    obs_comp = NULL,
    obs_variable = NULL,
    omega = c(0.1, 0.05, 0.1),
    omega_full = NULL,
    n_ind = NULL,
    ruv = NULL,
    y = NULL,
    rel_delta = 1e-04,
    method = "delta",
    sequence = NULL,
    auc = FALSE,
    sd = TRUE,
    q = NULL,
    in_parallel = FALSE,
    n_cores = 3,
    return_all = FALSE,
    ...
)

Arguments

model model, created using ‘PKPDsim::new_ode_model()’
parameters parameters list
regimen regimen, as created using ‘PKPDsim::new_regimen()’
t_obs vector of observation times
obs_comp observation compartment. If NULL will be "obs" (default)
os_var variable observation variable. If NULL, will be ignored, otherwise will override ‘obs_comp’.
omega triangle omega block
omega_full full omega block
n_ind number of individuals to simulate with sim method
ruv residual variability, supplied as a named list, ex: ‘list(prop = 0, add = 0, exp = 0)’
y vector of observations. If NULL, then a new simulation will be performed.
rel_delta rel_delta
method method, ‘delta‘ or ‘sim’
sequence for simulations, if not NULL the pseudo-random sequence to use, e.g. "halton" or "sobol". See ‘mvnorm2’ for more details.
auc is AUC?
ifelse0

sd     return as standard deviation (‘TRUE’) or variance (‘FALSE’)
q     return vector of quantiles instead of sd/var. Will return parametric quantiles when deltamethod is used, non-parametric for simulation-based methods.
in_parallel     run simulations in parallel?
n_cores     if run in parallel, on how many cores?
return_all     return object with all relevant information?
...     passed on to ‘sim_ode()’

Value

Vector of standard deviations or variances (or quantiles thereof) for dependent value variable

ifelse0     ifelse function but then based on whether value is NULL or not

Description

ifelse function but then based on whether value is NULL or not

Usage

ifelse0(value = NULL, alternative = NULL, allow_null = FALSE)

Arguments

value     metadata list object
alternative     alternative value
allow_null     can the alternative be NULL?

Value

value if non-NULL; alternative otherwise
is_newer_package  Check if package number is different from currently installed, and provide some messaging.

Description
Technically it only checks if a package version is different, not necessarily a higher version number.

Usage
is_newer_package(package, new_version)

Arguments
package  R package
new_version  new version number

is_positive_definite  Is matrix positive definite

Description
Is matrix positive definite

Usage
is_positive_definite(x)

Arguments
x  matrix, specified either as vector of lower triangle, or full matrix (as matrix class)

Value
TRUE if x is positive definite; FALSE otherwise.
join_cov_and_par

Combines covariates and parameters into a single list, useful for reparametrization of the model.

**Description**

Combines covariates and parameters into a single list, useful for reparametrization of the model.

**Usage**

```r
join_cov_and_par(covs, pars)
```

**Arguments**

- `covs`: covariates object
- `pars`: model parameters, such as the output of the `parameters()` call from a model library.

**Value**

List containing covariates and parameters

join_regimen

*Join two dosing regimens*

**Description**

Join two dosing regimens

**Usage**

```r
join_regimen(
  regimen1 = NULL,
  regimen2 = NULL,
  interval = NULL,
  dose_update = NULL,
  t_dose_update = NULL,
  continuous = FALSE
)
```
merge_regimen

Arguments

regimen1  first regimen
regimen2  second regimen
interval  interval between regimen1 and regimen2 (if dose_update not specified)
dose_update  dose number at which to override regimen1 with regimen 2 (if interval not specified)
t_dose_update  dose time from which to update regimen
continuous  for joining continuous infusions

Value

Joined regimen

merge_regimen  Merge two regimens together.

Description

In contrast to ‘join_regimen’, which joins two consecutive regimens together, ‘merge_regimen’ merges two or more regimens given at the same time. This can e.g. be used to define regimens for multi-drug models.

Usage

merge_regimen(regimens)

Arguments

regimens  List of PKPDsim regimens created with ‘new_regimen’.

Value

Merged regimens
model_from_api

Load model definition from API, and compile to R library

Description

Load model definition from API, and compile to R library

Usage

model_from_api(
  url,
  model = NULL,
  nonmem = NULL,
  verbose = TRUE,
  get_definition = FALSE,
  to_package = FALSE,
  force = FALSE,
  install_all = FALSE,
  ...
)

Arguments

url URL or file path to JSON representation of model
model model id (used in messages)
nonmem URL or file path to NONMEM file
verbose verbosity (T/F)
get_definition return only the model definition, do not compile
to_package compile to package?
force force install even if same version number of model already installed.
install_all force install all, even if model inactive
... arguments passed to new_ode_model() function

Value

Model object created with new_ode_model()
model_library

Description
Model library

Usage
model_library(name = NULL)

Arguments
name name of model in library. If none specified, will show list of available models.

Value
List containing information about the named model

mvrnorm2
More powerful multivariate normal sampling function

Description
Besides standard multivariate normal sampling (mvrnorm), allows exponential multivariate normal
and quasi-random multivariate normal (using the randtoolbox) all using the same interface.

Usage
mvrnorm2(n, mu, Sigma, exponential = FALSE, sequence = NULL, ...)

Arguments
n number of samples
mu mean
Sigma covariance matrix
exponential exponential distribution (i.e. multiply mu by exponential of sampled numbers)
sequence any sequence available in the randtoolbox, e.g. 'halton', or 'sobol'
... parameters passed to mvrnorm or randtoolbox sequence generator

Value
Multivariate normal samples
**na_locf**  
*Fill in NAs with the previous non-missing value*

**Description**

Inspired by zoo::na.locf0

**Usage**

```r
na_locf(object, fromLast = FALSE)
```

**Arguments**

- `object`: an object
- `fromLast`: logical. Causes observations to be carried backward rather than forward. Default is FALSE.

**Value**

Original object with NAs filled in

**new_adherence**  
*Probabilistically model adherence*

**Description**

Model the drug adherence using either a binomial probability distribution or a markov chain model based on the probability of staying adherent and of becoming adherent once non-adherent.

**Usage**

```r
new_adherence(
  n = 100,
  type = c("markov", "binomial"),
  p_markov_remain_ad = 0.75,
  p_markov_become_ad = 0.75,
  p_binom = 0.7
)
```
Arguments

- **n**: number of occasions to simulate
- **type**: type of adherence simulation, either "markov" or "binomial"
- **p_markov_remain_ad**: markov probability of staying adherent
- **p_markov_become_ad**: markov probability of going from non-adherent to adherent state
- **p_binom**: binomial probability of being adherent

Value

Returns a vector of length `n` containing values 0 (non-adherent) or 1 (adherent).

Numeric vector of length n

---

**new_covariate**  
*New covariate*

Description

Describe data for a covariate, either fixed or time-variant

Usage

```r
new_covariate(
  value = NULL,
  times = NULL,
  implementation = "interpolate",
  unit = NULL,
  interpolation_join_limit = 1,
  remove_negative_times = TRUE,
  comments = NULL,
  verbose = TRUE
)
```

Arguments

- **value**: a numeric vector
- **times**: NULL for time-invariant covariate or a numeric vector specifying the update times for the covariate
- **implementation**: for time-varying covariates either 'LOCF' (last observation carried forward) or 'interpolate' (default)
- **unit**: specify covariate unit (optional, for documentation purposes only)
new_covariate_model

interpolation_join_limit
    for interpolate option, if covariate timepoints are spaced too close together, the ODE solver sometimes chokes. This argument sets a lower limit on the space between timepoints. It will create average values on joint timepoints instead. If undesired set to NULL or 0.

remove_negative_times
    TRUE or FALSE

comments
    NULL, or vector of length equal to value specifying comments to each observation

verbose
    verbosity

Value

Object of class "covariate"

new_covariate_model
    covariate model function

Description

covariate model function

Usage

new_covariate_model(model = list())

Arguments

model
    covariate model specified as list

Value

List containing model function(s)

new_ode_model
    Create new ODE model

Description

Create new ODE model
Usage

new_ode_model(
  model = NULL,
  code = NULL,
  pk_code = NULL,
  dose_code = NULL,
  file = NULL,
  func = NULL,
  state_init = NULL,
  parameters = NULL,
  reparametrization = NULL,
  mixture = NULL,
  units = NULL,
  size = NULL,
  lagtime = NULL,
  obs = list(cmt = 1, scale = 1),
  dose = list(cmt = 1),
  covariates = NULL,
  declare_variables = NULL,
  iiv = NULL,
  iov = NULL,
  omega_matrix = NULL,
  ruv = NULL,
  ltbs = NULL,
  misc = NULL,
  cmt_mapping = NULL,
  int_step_size = NULL,
  default_parameters = NULL,
  fixed = NULL,
  cpp_show_code = FALSE,
  package = NULL,
  test_file = NULL,
  install = TRUE,
  folder = NULL,
  lib_location = NULL,
  verbose = FALSE,
  as_is = FALSE,
  nonmem = NULL,
  comments = NULL,
  version = "0.1.0",
  quiet = ""
)

Arguments

model model name from model library

code C++ code specifying ODE system

pk_code C++ code called at any event
**new_ode_model**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dose_code</td>
<td>C++ code called at dose event only</td>
</tr>
<tr>
<td>file</td>
<td>file containing C++ code</td>
</tr>
<tr>
<td>func</td>
<td>R function to be used with deSolve library</td>
</tr>
<tr>
<td>state_init</td>
<td>vector of state init</td>
</tr>
<tr>
<td>parameters</td>
<td>list or vector of parameter values</td>
</tr>
<tr>
<td>reparametrization</td>
<td>list of parameters with definitions that reparametrize the linear PK model to a 1-, 2-, or 3-compartment PK with standardized parametrization.</td>
</tr>
<tr>
<td>mixture</td>
<td>for mixture models, provide a list of the parameter associated with the mixture and its possible values and probabilities (of the first value), e.g. ‘list(CL = list(value = c(10, 20), probability = 0.3))’.</td>
</tr>
<tr>
<td>units</td>
<td>list or vector of parameter units</td>
</tr>
<tr>
<td>size</td>
<td>size of state vector for model. Size will be extracted automatically from supplied code, use this argument to override.</td>
</tr>
<tr>
<td>lagtime</td>
<td>lag time</td>
</tr>
<tr>
<td>obs</td>
<td>list with &quot;scale&quot; : character string with definition for scale, e.g. &quot;V&quot; or &quot;V*(WT/70)&quot;. If NULL, scale defaults to 1., and &quot;cmt&quot; the observation compartment</td>
</tr>
<tr>
<td>dose</td>
<td>specify default dose compartment, e.g. list(cmt = 1)</td>
</tr>
<tr>
<td>covariates</td>
<td>specify covariates, either as a character vector or a list. if specified as list, it allows use of timevarying covariates (see ‘new_covariate()’ function for more info)</td>
</tr>
<tr>
<td>declare_variables</td>
<td>declare variables</td>
</tr>
<tr>
<td>iiv</td>
<td>inter-individual variability, can optionally be added to library</td>
</tr>
<tr>
<td>iov</td>
<td>inter-occasion variability, can optionally be added to library</td>
</tr>
<tr>
<td>omega_matrix</td>
<td>variance-covariance matrix for inter-individual variability, can optionally be added to library</td>
</tr>
<tr>
<td>ruv</td>
<td>residual variability, can optionally be added to library</td>
</tr>
<tr>
<td>ltbs</td>
<td>log-transform both sides. Not used in simulations, only for fitting (sets attribute 'ltbs').</td>
</tr>
<tr>
<td>misc</td>
<td>a list of miscellaneous model metadata</td>
</tr>
<tr>
<td>cmt_mapping</td>
<td>list indicating which administration routes apply to which compartments. Example: ‘list(&quot;oral&quot; = 1, &quot;infusion&quot; = 2)’</td>
</tr>
<tr>
<td>int_step_size</td>
<td>step size for integrator. Can be pre-specified for model, to override default for ‘sim_ode()’</td>
</tr>
<tr>
<td>default_parameters</td>
<td>population or specific patient values, can optionally be added to library</td>
</tr>
<tr>
<td>fixed</td>
<td>parameters that should not have iiv added.</td>
</tr>
<tr>
<td>cpp_show_code</td>
<td>show generated C++ code</td>
</tr>
<tr>
<td>package</td>
<td>package name when saving as package</td>
</tr>
<tr>
<td>test_file</td>
<td>optional test file to be included with package</td>
</tr>
</tbody>
</table>
install  install package after compilation?
folder  base folder name to create package in
lib_location  install into folder (‘–library’ argument)
verbose  show more output
as_is  use C-code as-is, don’t substitute line-endings or shift indices
nonmem  add nonmem code as attribute to model object
comments  comments for model
version  number of library
quiet  passed on to ‘system2‘ as setting for stderr and stdout; how to output cmd line output. Default (’”‘) is R console, NULL or FALSE discards. TRUE captures the output and saves as a file.

Value

If package name is NULL, returns the model object. Otherwise has no return value.

---

**new_regimen**

* Dose regimen for sim_ode

**Description**

Create a dosing regimen for use with sim_ode

**Usage**

```r
new_regimen(
  amt = 100,
  interval = NULL,
  n = 3,
  times = NULL,
  type = NULL,
  t_inf = NULL,
  rate = NULL,
  t_lag = NULL,
  cmt = NULL,
  checks = TRUE,
  ss = FALSE,
  n_ss = NULL,
  first_dose_time = now_utc()
)
```
new_regimen

Arguments

amt  dosing amount, either a single value (which will repeated for multiple doses), or a vector with doses for each administration.
interval  dosing interval (requires n as argument).
n  number of doses (requires interval as argument).
times  vector describing dosing times. Overrides specified times using interval and n arguments.
type  either "infusion", "oral" or "bolus".
t_inf  infusion time (if 'type'=='infusion').
rate  infusion rate (if 'type'=='infusion'). 'NULL' by default. If specified, overrides 't_inf'.
t_lag  lag time (can be applied to any dose type, not only oral). Will just be added to 'times'.
cmt  vector of dosing compartments (optional, if NULL will dosing compartment defined in model will be used).
checks  input checks. Remove to increase speed (e.g. for population-level estimation or optimal design).
ss  steady state? boolean value whether to simulate out to steady state first (steady state will be based on specified 'amt' and 'interval', 'times' will be ignored).
n_ss  how many doses to simulate before assumed steady state. Default is 4 * 24 / 'interval'.
first_dose_time  datetime stamp of first dose (of class 'POSIXct'). Default is current date time.

Value

a list containing calculated VPC information, and a ggplot2 object

See Also

sim_ode

Examples

r1 <- new_regimen(amt=50, interval=12, n=20) # dose 50mg, q12hrs for 10 days
r2 <- new_regimen(amt=50, times=c(0:19)*12) # same, but using explicit times
r3 <- new_regimen(amt=c(rep(100,4), rep(50,16)), times=c(0:19)*12) # first 4 doses higher dose
nlmixr_parse_parameters

*Function to parse parameters for a model into a structure used by nlmixr*

**Description**

Function to parse parameters for a model into a structure used by nlmixr

**Usage**

```r
nlmixr_parse_parameters(
  parameters = list(CL = 5, V = 50),
  omega = c(0.1, 0.05, 0.1),
  res_var = list(prop = 0.1, add = 1),
  fixed = c(),
  log_transform = TRUE,
  ...
)
```

**Arguments**

- **parameters**: list of parameters
- **omega**: vector describing the lower-diagonal of the between-subject variability matrix
- **res_var**: residual variability. Expected a list with arguments ‘prop’, ‘add’, and/or ‘exp’. NULL by default.
- **fixed**: vector of fixed parameters
- **log_transform**: log-transform estimated parameters in nlmixr?
- **...**: passed on

**Value**

List of parameters that can be used by nlmixr

---

**nm_to_regimen**

*Create a regimen from NONMEM data*

**Description**

Create a regimen based on a NONMEM, or NONMEM-like dataset

**Usage**

```r
nm_to_regimen(data, reset_time = TRUE, first_only = FALSE)
```
Arguments

- **data**: NONMEM-type dataset
- **reset_time**: start time for each simulated patient at 0, irrespective of design in dataset
- **first_only**: use only design from first individual in dataset

Value

Regimen object

---

**now_utc**  
*Current time in UTC*

---

**Description**

Current time in UTC

**Usage**

```r
now_utc()
```

**Value**

POSIXct object containing current time in UTC

---

**OneCompIVbolus**  
*ADVAN-style equations*

---

**Description**

Adapted from Abuhelwa et al. JPET 2015

**Usage**

```r
OneCompIVbolus(d)
```

**Arguments**

- **d**: data, a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, DV, CL, V

**Details**

Functions for calculating drug amount in each compartments of the common pharmacokinetic models (1,2,3 compartment IV bolus, IV infusion, and first-order absorption models)

Definitions:
- $A_{i,last}$: is the initial amount at the beginning of each time interval ($t$, $t=t_2-t_1$) of a corresponding compartment (i.e. drug amount at the end of the last time interval)
- $E_i$: the sum of Exit (elimination) rate constant of the corresponding compartment. IV bolus- 1 compartment
Value

Returns a dataframe with populated columns for A1, and DV

References


OneCompIVinfusion  
IV infusion- 1 compartment

Description

IV infusion- 1 compartment

Usage

OneCompIVinfusion(d)

Arguments

d  
data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, RATE, RATEALL, DV, CL, V

Value

Returns a dataframe with populated columns for A1, and DV

OneCompOral  
first-order absorption 1 compartment

Description

first-order absorption 1 compartment

Usage

OneCompOral(d)

Arguments

d  
data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, DV, CL, V, KA & F1

Value

Returns a dataframe with populated columns for A1, A2 and DV
parse_obs_types

Parse observation types to simulation code

Description

Parse observation types to simulation code

Usage

parse_obs_types(obs)

Arguments

obs specified observation object including at least a description of which variable(s) are associated with a particular compartment, e.g. `list(variable="CONC", scale="1")`.

pkdata

PK dataset

Description

Example PK dataset

Usage

pkdata

Format

A data frame with 624 rows and 12 variables in NONMEM format

pkpdsim_to_nlmixr

Convert a model generated with PKPDsim to an object for nlmixr

Description

Convert a model generated with PKPDsim to an object for nlmixr
Usage

```r
pkpdsim_to_nlmixr(
    model = NULL,
    parameters = NULL,
    omega = NULL,
    res_var = NULL,
    fixed = c(),
    ini_code = NULL,
    model_code = NULL,
    model_par_code = NULL,
    verbose = FALSE,
    ...
)
```

Arguments

- `model`: PKPDsim model
- `parameters`: list of parameters
- `omega`: vector describing the lower-diagonal of the between-subject variability matrix
- `res_var`: residual variability. Expected a list with arguments 'prop', 'add', and/or 'exp'. NULL by default.
- `fixed`: vector of fixed (not estimated) parameter names
- `ini_code`: manually specify the 'ini' block for nlmixr
- `model_code`: manually specify the 'model' block for nlmixr
- `model_par_code`: manually specify the parameters section inside the 'model' block for nlmixr
- `verbose`: verbose, 'TRUE' or 'FALSE'
- `...`: passed on

Value

nlmixr function

---

`pop_regimen` | Remove n doses (from tail) of PKPDsim regimen

Description

Opposite of shift_regimen()

Usage

```r
pop_regimen(regimen, n = 1)
```
print.covariate

Arguments

regimen PKPDsim regimen created using 'new_regimen()'
n number of doses to pop from regimen

Value

Input regimen minus selected number of doses

See Also

shift_regimen

print.covariate  \hspace{1em} \textit{Print function for PKPDsim covariate object}

Description

Print function for PKPDsim covariate object

Usage

```r
## S3 method for class 'covariate'
print(x, ...)
```

Arguments

- \( x \) covariate object
- \( \ldots \) additional arguments

Value

No return value, print function.

print.PKPDsim  \hspace{1em} \textit{Print function for PKPDsim simulation function}

Description

Print function for PKPDsim simulation function

Usage

```r
## S3 method for class 'PKPDsim'
print(x, ...)
```
**print_regimen**

*Print function for PKPDsim regimen*

**Description**

Print function for PKPDsim regimen

**Usage**

```
## S3 method for class 'regimen'
print(x, ...)
```

**Arguments**

- `x` : regimen
- `...` : arguments to pass

**Value**

No return value, print function.

---

**print_list**

*Return a list in R syntax*

**Description**

Return a list in R syntax

**Usage**

```
print_list(x, wrapper = TRUE, quote = FALSE)
```

**Arguments**

- `x` : list to be printed
- `wrapper` : wrap in list object?
- `quote` : add quotes to values in list definition?
**read_model_json**

**Value**

Original list in R syntax

---

**Description**

Does some substitution of escaped characters in strings in the JSON file, then converts to a list with `jsonlite::fromJSON()`

**Usage**

`read_model_json(path)`

**Arguments**

- `path` Path to JSON file

**Value**

List containing contents of original JSON file

---

**regimen_to_nm**

**Convert PKPDsim regimen to NONMEM table (doses only)**

**Description**

Convert PKPDsim regimen to NONMEM table (doses only)

**Usage**

`regimen_to_nm(reg = NULL, dose_cmt = 1, n_ind = 1, t_obs = NULL, obs_cmt = 1)`

**Arguments**

- `reg` ‘PKPDsim’ regimen, created using ‘new_regimen()’ function
- `dose_cmt` dosing compartment, if not specified in ‘reg’ object
- `n_ind` repeat for ‘n_ind’ subjects
- `t_obs` add observation time(s)
- `obs_cmt` observation compartment for added observation time(s)

**Value**

Data frame containing doses
reparametrize

Reparametrize model parameters using a reparametrization defined within the model.

Description
Mostly useful for reparametrizing models into standard parametrizations, e.g. to NONMEM TRANS or clinPK parametrizations.

Usage
reparametrize(model, parameters, covariates)

Arguments
- **model**: PKPDsim model, compiled using 'reparametrization' argument or in metadata object.
- **parameters**: list of model parameters
- **covariates**: covariates list, specified as PKPDsim covariates

Value
Reparameterized model parameters

search_replace_in_file

Find string and replace in file

Description
Find string and replace in file

Usage
search_replace_in_file(files = c(), find = NULL, replacement = NULL)

Arguments
- **files**: vector of files
- **find**: find what string, vector of character
- **replacement**: replace with what, vector of character, should be equal in length to 'find'

Value
Function does not return a value but edits files on disk
**shift_regimen**

*Remove n doses (from start) of PKPDsim regimen*

**Description**

Opposite of pop_regimen()

**Usage**

```r
shift_regimen(regimen, n = 1, reset_time = TRUE)
```

**Arguments**

- `regimen`: PKPDsim regimen created using ‘new_regimen()’
- `n`: number of doses to shift regimen
- `reset_time`: reset the remaining doses to start at t=0?

**Value**

Regimen with selected number of doses removed from start

**See Also**

- pop_regimen

---

**shift_state_indices**

*R starts counting vector indices at 1, c++ starts at 0, so reduce all state numbers in the Cpp function definition by 1*

**Description**

R starts counting vector indices at 1, c++ starts at 0, so reduce all state numbers in the Cpp function definition by 1

**Usage**

```r
shift_state_indices(ode_def, n = -1)
```

**Arguments**

- `ode_def`: ODE definition
- `n`: add/subtract what number, default = -1
Simulate ODE or analytical equation

Description
Simulates a specified regimen using ODE system or analytical equation

Usage
```
sim(
  ode = NULL,
  analytical = NULL,
  parameters = NULL,
  parameters_table = NULL,
  mixture_group = NULL,
  omega = NULL,
  omega_type = "exponential",
  res_var = NULL,
  iov_bins = NULL,
  seed = NULL,
  sequence = NULL,
  n_ind = 1,
  event_table = NULL,
  regimen = NULL,
  lagtime = NULL,
  covariates = NULL,
  covariates_table = NULL,
  covariates_implementation = list(),
  covariate_model = NULL,
  A_init = NULL,
  only_obs = FALSE,
  obs_step_size = NULL,
  int_step_size = 0.01,
  t_max = NULL,
  t_obs = NULL,
  t_tte = NULL,
  t_init = 0,
  obs_type = NULL,
  duplicate_t_obs = FALSE,
  extra_t_obs = TRUE,
  rtte = FALSE,
  checks = TRUE,
  verbose = FALSE,
  return_event_table = FALSE,
  return_design = FALSE,
  output_include = list(parameters = FALSE, covariates = FALSE),
  ...)
```
Arguments

ode function describing the ODE system
analytical string specifying analytical equation model to use (similar to ADVAN1-5 in NONMEM). If specified, will not use ODEs.
parameters model parameters
parameters_table dataframe of parameters (with parameters as columns) containing parameter estimates for individuals to simulate. Formats accepted: data.frame, data.table, or list of lists.
mixture_group mixture group for models containing mixtures. Should be either ‘1’ or ‘2’, since only two groups are currently allowed.
omega vector describing the lower-diagonal of the between-subject variability matrix
omega_type exponential or normal, specified as vector
res_var residual variability. Expected a list with arguments ‘prop’, ‘add’, and/or ‘exp’. NULL by default.
iiov_bins allow override of the default IOV bins for a model. Specified as a vector of timepoints specifying the bin separators, e.g. ‘iiov_bins = c(0, 24, 48, 72, 9999)’.
seed set seed for reproducible results
sequence if not NULL specifies the pseudo-random sequence to use, e.g. "halton" or "sobol". See ‘mvnorm2’ for more details.
n_ind number of individuals to simulate
event_table use a previously created ‘design’ object used for ODE simulation instead of calling create_event_table() to create a new one. Especially useful for repeated calling of sim(), such as in optimizations or optimal design analysis. Also see ‘sim_core()’ for even faster simulations using precalculated ‘design’ objects.
regimen a regimen object created using the regimen() function
lagtime either a value (numeric) or a parameter (character) or NULL.
covariates list of covariates (for single individual) created using ‘new_covariate()’ function
covariates_table data.frame (or unnamed list of named lists per individual) with covariate values
covariates_implementation used only for ‘covariates_table’, a named list of covariate implementation methods per covariate, e.g. ‘list(WT = "interpolate", BIN = "locf")’
covariate_model R code used to pre-calculate effective parameters for use in ADVAN-style analytical equations. Not used in ODE simulations.
A_init vector with the initial state of the ODE system
only_obs only return the observations
obs_step_size the step size between the observations
int_step_size  the step size for the numerical integrator

the step size for the numerical integrator

t_max  maximum simulation time, if not specified will pick the end of the regimen as

maximum

t_obs  vector of observation times, only output these values (only used when t_obs==NULL)

t_tte  vector of observation times for time-to-event simulation

t_init  initialization time before first dose, default 0.

obs_type  vector of observation types. Only valid in combination with equal length vector

'\texttt{t\_obs}'.

duplicate_t_obs  allow duplicate \texttt{t\_obs} in output? E.g. for optimal design calculations when \texttt{t\_obs} = c(0,1,2,2,3). Default is FALSE.

extra_t_obs  include extra \texttt{t\_obs} in output for bolus doses? This is only activated when \texttt{t\_obs} is not specified manually. E.g. for a bolus dose at \texttt{t=24}, if FALSE, \texttt{PKPDsim} will output only the trough, so for bolus doses you might want to switch this setting to TRUE. When set to "auto" (default), it will be TRUE by default, but will switch to FALSE whenever \texttt{t\_obs} is specified manually.

rtte  should repeated events be allowed (FALSE by default)

checks  perform input checks? Default is TRUE. For calculations where \texttt{sim\_ode} is invoked many times (e.g. population estimation, optimal design) it makes sense to switch this to FALSE (after confirming the input is correct) to improve speed.

verbose  show more output

return_event_table  return the event table for the simulation only, does not run the actual simulation. Useful for iterative use of \texttt{sim}().

return_design  returns the design (event table and several other details) for the simulation, does not run the actual simulation. Useful for iterative functions like estimation in combination with \texttt{sim\_core()}, e.g. for estimation and optimal design.

output_include  list specifying what to include in output table, with keys 'parameters' and 'covariates'. Both are FALSE by default.

...  extra parameters

Value

a data frame of compartments with associated concentrations at requested times

Simulated regimen

See Also

\texttt{sim\_ode\_shiny}

Examples

\begin{verbatim}
 p <- list( 
 CL = 38.48,
\end{verbatim}
\begin{verbatim}
V = 7.4,
Q = 7.844,
V2 = 5.19,
Q2 = 9.324,
V3 = 111
)

omega <- c(0.3, # IIV CL
0.1, 0.3) # IIV V

r1 <- new_regimen(
  amt = 100,
  times = c(0, 24, 36),
  type = "infusion"
)

mod <- new_ode_model("pk_3cmt_iv")

dat <- sim(
  ode = mod,
  parameters = p,
  omega = omega,
  n_ind = 20,
  regimen = r1
)
\end{verbatim}

**sim_core**

Only core function of the simulation function, always just returns observations. Mostly useful for estimations / optimal design. Has no checks (for speed)!

**Description**

Only core function of the simulation function, always just returns observations. Mostly useful for estimations / optimal design. Has no checks (for speed)!

**Usage**

\[
sim\_core(sim\_object = NULL, ode, duplicate\_t\_obs = FALSE, t\_init = 0)
\]

**Arguments**

- **sim\_object** list with design and simulation parameters
- **ode** ode
- **duplicate\_t\_obs** allow duplicate t\_obs in output? E.g. for optimal design calculations when t\_obs = c(0,1,2,2,3). Default is FALSE.
- **t\_init** time of initialization of the ODE system. Usually 0.
**Value**

Data frame with simulation results

---

**Description**

Deprecated function, renamed to `sim()`

**Usage**

`sim_ode(...)`

**Arguments**

`...` parameters passed to `sim()` function

**Value**

Output from `sim()`

**See Also**

`sim`

---

**sim_ode_shiny**

*Simulate ODE and create a Shiny app*

---

**Description**

This function has been deprecated and moved to a separate package at https://github.com/ronkeizer/PKPDsimShiny.

**Usage**

`sim_ode_shiny(...)`

**Arguments**

`...` arguments passed to PKPDsimShiny::sim_ode_shiny()

**Value**

No return value

**See Also**

`sim_ode`
**table_to_list**  
*Convert a table to a list*

**Description**  
Convert a table to a list

**Usage**  
table_to_list(table)

**Arguments**  
table data.frame

**Value**  
List containing original table contents

---

**test_model**  
*Test a model*

**Description**  
Test a model

**Usage**  
test_model(url, test_file, package, force = FALSE)

**Arguments**  
url URL or file path to JSON representation of model  
test_file Path to a .R file containing tests to run  
package Package name  
force Run tests even if model is not flagged for building? Defaults to FALSE

**Value**  
Runs test file for a model but does not return a value
**test_pointer**

Test if model still in memory

**Description**
Test if model still in memory

**Usage**

```r
test_pointer(model)
```

**Arguments**

- **model**: pointer to model

**Value**
No return value

---

**ThreeCompIVbolus**

*IV bolus- 3 compartment*

**Description**

IV bolus- 3 compartment

**Usage**

```r
ThreeCompIVbolus(d)
```

**Arguments**

- **d**: data, Accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, DV, CL, V1, Q12, V2, Q13, V3

**Value**

Returns a dataframe with populated columns for A1, A2, A3, and DV
**ThreeCompIVinfusion**  
*IV infusion- 3 compartment*

---

**Description**

IV infusion- 3 compartment

**Usage**

ThreeCompIVinfusion(d)

**Arguments**

- **d**  
  data, Accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, RATE, RATEALL, DV, CL, V1, Q12, V2, Q13, V3

**Value**

Returns a dataframe with populated columns for A1, A2, A3, and DV

---

**ThreeCompIVinfusionMetab**  
*3-compartment IV infusion with first-order metabolite formation*

---

**Description**

3-compartment IV infusion with first-order metabolite formation

**Usage**

ThreeCompIVinfusionMetab(d)

**Arguments**

- **d**  
  data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, RATE, RATEALL, DV, CL, V1, Q12, V2, Q13, V3, CLM, VM, km

**Value**

Returns a dataframe with populated columns for A1, A2, A3, and DV
**ThreeCompOral**

*first-order absorption- 3 compartment*

**Description**

*first-order absorption- 3 compartment*

**Usage**

ThreeCompOral(d)

**Arguments**

- **d**
  
data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT,MDV,DV, CL, V2, Q3, V3, Q4, V4, KA & F1

**Value**

Returns a dataframe with populated columns for A1, A2, A3, A4 and DV

---

**ThreeCompOralMetab**

*first-order absorption- 3 compartment-Metabolite*

**Description**

*first-order absorption- 3 compartment-Metabolite*

**Usage**

ThreeCompOralMetab(d)

**Arguments**

- **d**
  
data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT,MDV,DV, CL, V2, Q3, V3, Q4, V4, KA & F1

**Value**

Returns a dataframe with populated columns for A1, A2, A3, A4 and DV
translate_ode

Translate a model from/to various PKPD simulators

Description
Currently only supports PKDPsim <-> RxODE

Usage
translate_ode(code, auto = TRUE, from = NULL, to = NULL, verbose = TRUE)

Arguments
- code: character string with ODE code
- auto: is auto-detect syntax (‘from’)
- from: from syntax
- to: to syntax
- verbose: verbose, ‘TRUE’ or ‘FALSE’

Value
Translated PKPDsim or RxODE model

triangle_to_full

Convert triangle omega matrix to full omega matrix

Description
Convert triangle omega matrix to full omega matrix

Usage
triangle_to_full(vect)

Arguments
- vect: vector specifying triangle omega matrix

Value
Omega matrix
TwoCompIVbolus  
**IV bolus - 2 compartment**

**Description**

IV bolus - 2 compartment

**Usage**

TwoCompIVbolus(d)

**Arguments**

d  
Data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, DV, CL, V1, Q, V2

**Value**

Returns a dataframe with populated columns for A1, A2, and DV

TwoCompIVinfusion  
**IV infusion - 2 compartment**

**Description**

IV infusion - 2 compartment

**Usage**

TwoCompIVinfusion(d)

**Arguments**

d  
Data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, RATE, RATEALL, DV, CL, V1, Q, V2

**Value**

Returns a dataframe with populated columns for A1, A2, and DV
TwoCompOral

First-order absorption- 2 compartment

Description

First-order absorption- 2 compartment

Usage

TwoCompOral(d)

Arguments

d data, accepts a NONMEM style data frame for 1 subject with columns for TIME, AMT, MDV, DV, CL, V2, Q, V3, KA & F1

Value

Returns a dataframe with populated columns for A1, A2, A3 and DV

vector_to_R_code

Transform a vector into a string that evaluates to the same vector

Description

Collapses a vector into a comma-separated list with strings quoted (and special characters escaped). A general purpose helper function for writing new model code.

Usage

vector_to_R_code(vec)

Arguments

vec a vector

Value

character string of length 1
Index

* datasets
  pkdata, 39
add_quotes, 4
add_ruv, 4
add_ruv_to_quantile, 5
adherence_binomial, 6
adherence_markov, 6
advan, 7
advan_create_data, 7
advan_parse_output, 8
advan_process_infusion_doses, 8
analytical_eqn_wrapper, 9
apply_lagtime, 9
bioavailability_to_R_code, 10
calc_dydP, 12
calc_ss_analytic, 12
calculate_parameters, 11
check_iov_specification, 13
check_mixture_model, 14
compile_sim_cpp, 14
covariate_last_obs_only, 16
covariates_table_to_list, 15
create_event_table, 16
create_obs_data, 17
cv_to_omega, 18
define_tdm_init_model, 18
detect_ode_syntax, 19
f_cov, 19
get_ode_model_size, 20
get_parameters_from_code, 20
get_t_obs_from_regimen, 21
get_var_y, 21
ifelse0, 23
is_newer_package, 24
is_positive_definite, 24
join_cov_and_par, 25
join_regimen, 25
jsonlite::fromJSON(), 43
merge_regimen, 26
model_from_api, 27
model_library, 28
mvnorm2, 28
na_locf, 29
new_adherence, 29
new_covariate, 30
new_covariate_model, 31
new_ode_model, 31
new_ode_model(), 27
new_regimen, 34
nlmixr_parse_parameters, 36
nm_to_regimen, 36
now_utc, 37
OneCompIVbolus, 37
OneCompIVinfusion, 38
OneCompOral, 38
parse_obs_types, 39
pkdata, 39
PKPDsim-package, 4
pkpdsim_to_nlmixr, 39
pop_regimen, 40
print.covariate, 41
print.PKPDsim, 41
print.regimen, 42
print_list, 42
read_model_json, 43
regimen_to_nm, 43
reparametrize, 44
search_replace_in_file, 44
shift_regimen, 45
shift_state_indices, 45
sim, 46
sim(), 17, 50
sim_core, 49
sim_ode, 18, 35, 50, 50
sim_ode_shiny, 48, 50

table_to_list, 51
test_model, 51
test_pointer, 52
ThreeCompIVbolus, 52
ThreeCompIVinfusion, 53
ThreeCompIVinfusionMetab, 53
ThreeCompOral, 54
ThreeCompOralMetab, 54
translate_ode, 55
triangle_to_full, 55
TwoCompIVbolus, 56
TwoCompIVinfusion, 56
TwoCompOral, 57

vector_to_R_code, 57