Package ‘PKPDsim’
March 2, 2023

Type Package
Title Tools for Performing Pharmacokinetic-Pharmacodynamic Simulations
Version 1.3.0
Date 2023-02-24
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, knitr, rmarkdown
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 7.2.3
Config/testthat/edition 3
Config/Needs/website tidyverse, nlmixr2
Encoding UTF-8
VignetteBuilder knitr
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 2023-03-02 12:20:28 UTC
R topics documented:

add_quotes .................................................. 3
add_ruv ....................................................... 4
add_ruv_to_quantile ........................................ 4
adherence_binomial ........................................... 5
adherence_markov ............................................. 5
advan ........................................................ 6
advan_create_data ........................................... 6
advan_parse_output .......................................... 7
advan_process_infusion_doses ................................ 7
apply_lagtime ............................................... 8
calculate_parameters ....................................... 9
calc_dydP .................................................... 10
calc_ss_analytic ............................................ 10
check_obs_input ........................................... 11
compile_sim_cpp ............................................ 12
covariates_table_to_list ................................... 13
covariate_last_obs_only .................................. 13
cv_to_omega ................................................ 14
detect_ode_syntax .......................................... 14
f_cov ......................................................... 15
get_fixed_parameters ....................................... 15
get_ode_model_size ........................................ 16
get_parameters_from_code .................................. 16
get_var_y .................................................... 17
ifelse0 ........................................................ 18
is_positive_definite ....................................... 19
join_cov_and_par .......................................... 19
join_regimen ............................................... 20
lower_triangle_mat_size ................................... 20
merge_regimen .............................................. 21
model_from_api ............................................. 21
model_library .............................................. 22
mvrnorm2 .................................................... 23
na_locf ...................................................... 23
new_adherence .............................................. 24
new_covariate ............................................... 24
new_covariate_model ...................................... 25
new_ode_model ............................................. 26
new_regimen ................................................ 29
nlmixr_parse_parameters ................................... 30
nm_to_regimen ............................................. 31
pkdata ....................................................... 32
pkpdsim_to_nlmixr ........................................ 32
pop_regimen ................................................ 33
print_list ................................................... 34
read_model_json ............................................ 34
add_quotes

Put vector values in 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

Add residual variability to the dependent variable

Description

Add residual variability to the dependent variable

Usage

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

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

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
**adherence_binomial**

**Value**

Numeric vector of y values with residual variability

**Description**

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

**Usage**

```r
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**

```r
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
advan  
**ADVAN-style functions to calculate linear PK systems**

Description

ADVAN-style functions to calculate linear PK systems

Usage

```r
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

```r
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_parse_output

- `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

---

**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)
```
apply_lagtime

Description

Apply lagtime to a regimen

Usage

apply_lagtime(regimen, lagtime, parameters, cmt_mapping = NULL)

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
calculate_parameters  
*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 `\text{CL}_i = \text{CL} \times (\text{WT}/70) \times (1/\text{CR})` it can be used to calculate `\text{CL}_i` 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**

```r
calc_dydP(dy, y, rel_delta, 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**

```r
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
)
```
check_obs_input

Arguments

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

---

check_obs_input Checks obs input for valid combinations of cmt, var, scale

Description

Checks obs input for valid combinations of cmt, var, scale

Usage

\texttt{check_obs_input(\textit{obs})}

Arguments

- \texttt{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")`.
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,
  obs = NULL,
  dose = NULL,
  iov = NULL,
  compile = TRUE,
  verbose = FALSE,
  as_is = FALSE
)
```

Arguments

- `code`: C++ code ODE system
- `dose_code`: C++ code per dose event
- `pk_code`: C++ code per any event (similar to PK)
- `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
**covariates_table_to_list**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dose</td>
<td>dose specification</td>
</tr>
<tr>
<td>iov</td>
<td>iov specification</td>
</tr>
<tr>
<td>compile</td>
<td>compile or not?</td>
</tr>
<tr>
<td>verbose</td>
<td>show more output</td>
</tr>
<tr>
<td>as_is</td>
<td>use C-code as-is, don’t substitute line-endings or shift indices</td>
</tr>
</tbody>
</table>

**Value**

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

---

**covariates_table_to_list**

Convert covariate table specified as data.frame

**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**

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

---

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

---
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
**f_cov**

**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_fixed_parameters**  
_Get fixed parameters_

---

**Description**

Get fixed parameters listed in model definition if present. If not present, use size of omega matrix to determine fixed parameters.

**Usage**

get_fixed_parameters(def)

**Arguments**

def  
Model definition as output by `read_model_json()`
get_ode_model_size  
*Get the number of states in the ODE from the code C++ code for model*

**Description**
Get the number of states in the ODE from the 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_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

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)
obs_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?
sd  return as standard deviation (‘TRUE’) or variance (‘FALSE’)
q  return vector of quantiles instead of sd/var. Will return parametric quantiles when delta-method 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

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_positive_definite  Is matrix positive definite

Description
Is matrix positive definite

Usage
is_positive_definite(x)

Arguments
x  

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
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*

**Usage**

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

**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

---

### lower_triangle_mat_size

*Size of the lower triangle of the matrix*

**Description**

Size of the lower triangle of the matrix

**Usage**

```r
lower_triangle_mat_size(mat)
```
merge_regimen

Arguments

mat omega matrix as a vector

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()`
**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**

```r
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
new_covariate_model

Usage

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

Arguments

value a numeric vector

value is NULL for a time-invariant covariate or a numeric vector specifying
the update times for the covariate.

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)

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\`

round_times round times to specified number of digits. If NULL, will not round.

comments NULL, or vector of length equal to value specifying comments to each observa-
tion

verbose verbosity

Value

Object of class "covariate"

Description

covariate model function
Usage

new_covariate_model(model = list())

Arguments

model covariate model specified as list

Value

List containing model function(s)

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,
  development = NULL,
  omega_matrix = NULL,
  ruv = NULL,
  ltbs = NULL,
  misc = NULL,
  cmt_mapping = NULL,
  int_step_size = NULL,
)
new_ode_model

```r
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,
onnomem = NULL,
comments = NULL,
version = "0.1.0",
quiet = "",
definition = NULL
)
```

**Arguments**

- **model**: model name from model library
- **code**: C++ code specifying ODE system
- **pk_code**: C++ code called at any event
- **dose_code**: C++ code called at dose event only
- **file**: file containing C++ code
- **func**: R function to be used with deSolve library
- **state_init**: vector of state init
- **parameters**: list or vector of parameter values
- **reparametrization**: list of parameters with definitions that reparametrize the linear PK model to a 1-, 2- or 3-compartment PK with standardized parametrization.
- **mixture**: 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))`.
- **units**: list or vector of parameter units
- **size**: size of state vector for model. Size will be extracted automatically from supplied code, use this argument to override.
- **lagtime**: lag time
- **obs**: list with "scale": character string with definition for scale, e.g. "V" or "V*(WT/70)". If NULL, scale defaults to 1., and "cmt" the observation compartment
- **dose**: specify default dose compartment, e.g. list(cmt = 1)
- **covariates**: 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)
- **declare_variables**: declare variables
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>development</td>
<td>Information about the model development population, 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>
<tr>
<td>install</td>
<td>install package after compilation?</td>
</tr>
<tr>
<td>folder</td>
<td>base folder name to create package in</td>
</tr>
<tr>
<td>lib_location</td>
<td>install into folder (‘--library’ argument)</td>
</tr>
<tr>
<td>verbose</td>
<td>show more output</td>
</tr>
<tr>
<td>as_is</td>
<td>use C-code as-is, don’t substitute line-endings or shift indices</td>
</tr>
<tr>
<td>nonmem</td>
<td>add nonmem code as attribute to model object</td>
</tr>
<tr>
<td>comments</td>
<td>comments for model</td>
</tr>
<tr>
<td>version</td>
<td>number of library</td>
</tr>
<tr>
<td>quiet</td>
<td>passed on to ‘system2’ as setting for stderr and stdout; how to output cmd line output. Default (&quot;&quot;') is R console, NULL or FALSE discards. TRUE captures the output and saves as a file.</td>
</tr>
<tr>
<td>definition</td>
<td>optional, filename for the JSON file the full definition for the model. The definition file will be stored as ‘definition.json’ in the resulting package.</td>
</tr>
</tbody>
</table>

**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

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()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>amt</td>
<td>dosing amount, either a single value (which will repeated for multiple doses), or a vector with doses for each administration</td>
</tr>
<tr>
<td>interval</td>
<td>dosing interval (requires n as argument)</td>
</tr>
<tr>
<td>n</td>
<td>number of doses (requires interval as argument)</td>
</tr>
<tr>
<td>times</td>
<td>vector describing dosing times. Overrides specified times using interval and n arguments</td>
</tr>
<tr>
<td>type</td>
<td>either &quot;infusion&quot;, &quot;bolus&quot;, &quot;oral&quot;, &quot;sc&quot; (subcutaneous), or &quot;im&quot; (intramuscular)</td>
</tr>
<tr>
<td>t_inf</td>
<td>infusion time (if 'type'=='infusion')</td>
</tr>
<tr>
<td>rate</td>
<td>infusion rate (if 'type'=='infusion'). ‘NULL’ by default. If specified, overrides 't_inf'</td>
</tr>
<tr>
<td>t_lag</td>
<td>lag time (can be applied to any dose type, not only oral). Will just be added to 'times'</td>
</tr>
<tr>
<td>cmt</td>
<td>vector of dosing compartments (optional, if NULL will dosing compartment defined in model will be used)</td>
</tr>
<tr>
<td>checks</td>
<td>input checks. Remove to increase speed (e.g. for population-level estimation or optimal design)</td>
</tr>
</tbody>
</table>
nlmixr_parse_parameters

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

Usage

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,
  ...
)

Description

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

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
Arguments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameters</td>
<td>list of parameters</td>
</tr>
<tr>
<td>omega</td>
<td>vector describing the lower-diagonal of the between-subject variability matrix</td>
</tr>
<tr>
<td>res_var</td>
<td>residual variability. Expected a list with arguments 'prop', 'add', and/or 'exp'. NULL by default.</td>
</tr>
<tr>
<td>fixed</td>
<td>vector of fixed parameters</td>
</tr>
<tr>
<td>log_transform</td>
<td>log-transform estimated parameters in nlmixr?</td>
</tr>
<tr>
<td>...</td>
<td>passed on</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>NONMEM-type dataset</td>
</tr>
<tr>
<td>reset_time</td>
<td>start time for each simulated patient at 0, irrespective of design in dataset</td>
</tr>
<tr>
<td>first_only</td>
<td>use only design from first individual in dataset</td>
</tr>
</tbody>
</table>

Value

Regimen object
pkpdsim_to_nlmixr

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
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.
**pop_regimen**

<table>
<thead>
<tr>
<th>fixed</th>
<th>vector of fixed (not estimated) parameter names</th>
</tr>
</thead>
<tbody>
<tr>
<td>ini_code</td>
<td>manually specify the ‘ini’ block for nlmixr</td>
</tr>
<tr>
<td>model_code</td>
<td>manually specify the ‘model’ block for nlmixr</td>
</tr>
<tr>
<td>model_par_code</td>
<td>manually specify the parameters section inside the ‘model’ block for nlmixr</td>
</tr>
<tr>
<td>verbose</td>
<td>verbose, ‘TRUE’ or ‘FALSE’</td>
</tr>
<tr>
<td>...</td>
<td>passed on</td>
</tr>
</tbody>
</table>

**Description**

Opposite of shift_regimen()

**Usage**

```r
pop_regimen(regimen, n = 1)
```

**Arguments**

<table>
<thead>
<tr>
<th>regimen</th>
<th>PKPDsim regimen created using ‘new_regimen()’</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>number of doses to pop from regimen</td>
</tr>
</tbody>
</table>

**Value**

Input regiment minus selected number of doses

**See Also**

shift_regimen
print_list \hspace{10em} \textit{Return a list in R syntax}

\textbf{Description}

Return a list in R syntax

\textbf{Usage}

\texttt{print_list(x, wrapper = TRUE)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} \hspace{10em} list to be printed
  \item \texttt{wrapper} \hspace{10em} wrap in list object?
\end{itemize}

\textbf{Value}

Original list in R syntax

\textbf{read_model_json} \hspace{10em} \textit{Read model definition from JSON}

\textbf{Description}

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

\textbf{Usage}

\texttt{read_model_json(path)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{path} \hspace{10em} Path to JSON file
\end{itemize}

\textbf{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**

```r
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**

```r
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**
```r
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`
Simulate ODE or analytical equation

Description

Simulates a specified regimen using ODE system or analytical equation

Usage

```r
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.
iov_bins allow override of the default IOV bins for a model. Specified as a vector of timepoints specifying the bin separators, e.g. ‘iov_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

$t_{\text{max}}$  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)

$\mathbf{t}_{\text{tte}}$  vector of observation times for time-to-event simulation

$t_{\text{init}}$  initialization time before first dose, default 0.

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

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.

extra_t_obs  include extra t_obs in output for bolus doses? This is only activated when ‘t_obs’ is not specified manually. E.g. for a bolus dose at t=24, if FALSE, 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 ‘t_obs’ is specified manually.

rtte  should repeated events be allowed (FALSE by default)

checks  perform input checks? Default is TRUE. For calculations where 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 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 ‘sim_core()’, e.g. for estimation and optimal design.

output_include  list specifying what to include in output table, with keys ‘parameters’ and ‘co-variates’. Both are FALSE by default.

...  extra parameters

Value

a data frame of compartments with associated concentrations at requested times

Simulated regimen

See Also

sim_ode_shiny

Examples

```
p <- list(
  CL = 38.48,
```
\( V = 7.4, \)  
\( Q = 7.844, \)  
\( V2 = 5.19, \)  
\( Q2 = 9.324, \)  
\( V3 = 111 \)

\[
\omega \leftarrow c(0.3, \quad \text{# IIV CL} \\
0.1, 0.3) \quad \text{# IIV V}
\]

\[
r1 \leftarrow \text{new_regimen(} \\
\text{amt = 100,} \\
\text{times = c(0, 24, 36),} \\
\text{type = "infusion")}
\]

\[
\text{mod} \leftarrow \text{new_ode_model("pk_3cmt_iv")}
\]

\[
dat \leftarrow \text{sim(} \\
\text{ode = mod,} \\
\text{parameters = p,} \\
\text{omega = omega,} \\
\text{n_ind = 20,} \\
\text{regimen = r1)}
\]

---

**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**

\[
\text{sim_core(sim_object = NULL, ode, duplicate_t_obs = FALSE, t_init = 0)}
\]

**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

sim_ode

Deprecated function, renamed to sim()

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**
```r
table_to_list(table)
```

**Arguments**
- `table`: data.frame

**Value**
List containing original table contents

---

test_model  
*Test a model*

**Description**
Test a model

**Usage**
```r
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

---

translate_ode

Translate a model from/to various PKPD simulators

**Description**

Currently only supports PKDPsim <-> RxODE

**Usage**

```r
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 PKDPsim or RxODE model
Description

Convert triangle omega matrix to full omega matrix

Usage

triangle_to_full(vect)

Arguments

vect vector specifying triangle omega matrix

Value

Omega matrix
Index

* datasets
  pkdata, 32

add_quotes, 3
add_ruv, 4
add_ruv_to_quantile, 4
adherence_binomial, 5
adherence_markov, 5
advan, 6
advan_create_data, 6
advan_parse_output, 7
advan_process_infusion_doses, 7
apply_lagtime, 8
calc_dyDP, 10
calc_ss_analytic, 10
calculate_parameters, 9
check_obs_input, 11
compile_sim_cpp, 12
covariate_last_obs_only, 13
covariates_table_to_list, 13
cv_to_omega, 14
detect_ode_syntax, 14
f_cov, 15
get_fixed_parameters, 15
get_ode_model_size, 16
get_parameters_from_code, 16
get_var_y, 17
ifelse0, 18
is_positive_definite, 19
join_cov_and_par, 19
join_regimen, 20
jsonlite::fromJSON(), 34
lower_triangle_mat_size, 20
merge_regimen, 21
model_from_api, 21
model_library, 22
mvrnorm2, 23
na_locf, 23
new_adherence, 24
new_covariate, 24
new_covariate_model, 25
new_ode_model, 26
new_ode_model(), 22
new_regimen, 29
nlmixr_parse_parameters, 30
nm_to_regimen, 31
pkdata, 32
pkpdsim_to_nlmixr, 32
pop_regimen, 33
print_list, 34
read_model_json, 34
read_model_json(), 15
regimen_to_nm, 35
reparametrize, 35
search_replace_in_file, 36
shift_regimen, 36
sim, 37
sim(), 41
sim_core, 40
sim_ode, 14, 30, 41, 41
sim_ode_shiny, 39, 41
table_to_list, 42
test_model, 42
test_pointer, 43
translate_ode, 43
triangle_to_full, 44