

# Package ‘PKPDsim’

October 12, 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](https://doi.org/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](mailto: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
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
merge_regimen	26
model_from_api	27
model_library	28
mvrnorm2	28
na_locf	29
new_adherence	29
new_covariate	30
new_covariate_model	31
new_ode_model	31
new_regimen	34

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

---

PKPDsim-package	<i>PKPDsim package</i>
-----------------	------------------------

---

**Description**

Simulate regimens for PKPD models defined by ODE systems

**Author(s)**

Ron Keizer <ronkeizer@gmail.com>

---

add_quotes	<i>Put vector values in quotes</i>
------------	------------------------------------

---

**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	<i>Add residual variability to the dependent variable</i>
---------	---

---

**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

**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

---

advan	<i>ADVAN-style functions to calculate linear PK systems</i>
-------	---

---

**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	<i>Create ADVAN-style dataset</i>
-------------------	-----------------------------------

---

**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.

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	<i>Internal function to parse the raw output from ADVAN-style functions</i>
--------------------	---

---

**Description**

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

**Usage**

```
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	<i>Add column RATEALL to ADVAN-style dataset to handle infusions</i>
------------------------------	--

---

**Description**

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

**Usage**

```
advan_process_infusion_doses(data)
```



**Arguments**

data                    ADVAN-style dataset, e.g. created using 'advan\_create\_data'.

**Value**

Data frame containing additional RATEALL column.

**References**

Abuhelwa, A. Y., Foster, D. J. R., Upton, R. N. (2015) ADVAN-style analytical solutions for common pharmacokinetic models. *J Pharmacol Toxicol Methods* 73:42-8. DOI: 10.1016/j.vascn.2015.03.004

---

analytical\_eqn\_wrapper

*Wrapper for using analytical equations with PKPD regimens*

---

**Description**

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

**Usage**

```
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**

```
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

---

bioavailability\_to\_R\_code

*Transforms bioavailability specs into appropriate R code*

---

**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\_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 ‘ $CL_i = CL * (WT/70) * (1/CR)$ ’ it can be used to calculate ‘ $CL_i$ ’ without having to write that function a second time in R.

## Usage

```
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	<i>Calculate derivative</i>
-----------	-----------------------------

---

**Description**

Calculate derivative

**Usage**

```
calc_dydP(dy, y, rel_delta, log_y)
```

**Arguments**

dy	dy
y	dependent value
rel_delta	relative delta
log_y	logical indicating if the dependent variable is log transformed

---

calc_ss_analytic	<i>Returns the state of a linear PK system at steady state (trough) using analytics equations (so for linear PK systems only).</i>
------------------	--

---

**Description**

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

**Usage**

```
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

---

check\_iov\_specification

*Checks that IOV was specified appropriately*

---

**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**

```
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	<i>Check that mixture model is specified in right format and within constraints (1 parameter, 2 groups)</i>
---------------------	---

---

**Description**

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

**Usage**

```
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	<i>Compile ODE model to c++ function</i>
-----------------	--

---

**Description**

Compile ODE model to c++ function

**Usage**

```
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
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

---

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

---

create\_event\_table      *Create an event table*

---

**Description**

Create an event table

**Usage**

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



**Arguments**

regimen	regimen
t_max	t_max
t_obs	t_obs
t_tte	t_tte
t_init	t_init
p	parameters
covariates	covariates
model	model
obs_type	observation type

---

create_obs_data	<i>Create obs data</i>
-----------------	------------------------

---

**Description**

Used by [sim\(\)](#) to arrange data from `ode()` function into the correct format.

**Usage**

```
create_obs_data(ode_data, obs_attr, id)
```

**Arguments**

ode_data	data frame of output from <code>ode()</code> function
obs_attr	"obs" attribute from <code>ode()</code> function
id	ID of the individual

**See Also**

[sim\(\)](#)

---

cv_to_omega	<i>Create lower-diagonal omega matrix from CV for parameter estimates</i>
-------------	---

---

**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	<i>defines C code for TDM before dose conditions</i>
-----------------------	--

---

**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	<i>Auto-detect the syntax for the ODE code</i>
-------------------	--

---

**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	<i>covariate function builder</i>
-------	-----------------------------------

---

**Description**

covariate function builder

**Usage**

```
f_cov(...)
```

**Arguments**

...                    parameters to pass to cov

**Value**

Covariate function

---

get_ode_model_size	<i>Get the number of states in the ODE from the code code C++ code for model</i>
--------------------	--

---

**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	<i>Get model parameters from code</i>
--------------------------	---------------------------------------

---

**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

```
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

**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 'mvrnorm2' 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 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	<i>ifelse function but then based on whether value is NULL or not</i>
---------	---

---

**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	<i>Check if package number is different from currently installed, and provide some messaging.</i>
------------------	---

---

**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	<i>Is matrix positive definite</i>
----------------------	------------------------------------

---

**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	<i>Combines covariates and parameters into a single list, useful for reparametrization of the model.</i>
------------------	--

---

**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	<i>Join two dosing regimens</i>
--------------	---------------------------------

---

**Description**

Join two dosing regimens

**Usage**

```
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

---

merge_regimen	<i>Merge two regimens together.</i>
---------------	-------------------------------------

---

**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	<i>Load model definition from API, and compile to R library</i>
----------------	---

---

### 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 <code>new_ode_model()</code> function

### Value

Model object created with `new_ode_model()`

---

model_library	<i>Model library</i>
---------------	----------------------

---

**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	<i>More powerful multivariate normal sampling function</i>
----------	--

---

**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	<i>Fill in NAs with the previous non-missing value</i>
---------	--

---

**Description**

Inspired by zoo::na.locf0

**Usage**

```
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	<i>Probabilistically model adherence</i>
---------------	--

---

**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**

```
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	<i>New covariate</i>
---------------	----------------------

---

**Description**

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

**Usage**

```
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)

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



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 it's 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
iiv	inter-individual variability, can optionally be added to library
iov	inter-occasion variability, can optionally be added to library
omega_matrix	variance-covariance matrix for inter-individual variability, can optionally be added to library
ruv	residual variability, can optionally be added to library
ltbs	log-transform both sides. Not used in simulations, only for fitting (sets attribute 'ltbs').
misc	a list of miscellaneous model metadata
cmt_mapping	list indicating which administration routes apply to which compartments. Example: 'list("oral" = 1, "infusion" = 2)'
int_step_size	step size for integrator. Can be pre-specified for model, to override default for 'sim_ode()'
default_parameters	population or specific patient values, can optionally be added to library
fixed	parameters that should not have iiv added.
cpp_show_code	show generated C++ code
package	package name when saving as package
test_file	optional test file to be included with package

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	<i>Dose regimen for sim_ode</i>
-------------	---------------------------------

---

### 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**

amt	dosing amount, either a single value (which will be 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 $4 * 24 / \text{'interval'}$ .
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

```
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

```
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	<i>Current time in UTC</i>
---------	----------------------------

---

**Description**

Current time in UTC

**Usage**

now\_utc()

**Value**

POSIXct object containing current time in UTC

---

OneCompIVbolus	<i>ADVAN-style equations</i>
----------------	------------------------------

---

**Description**

Adapted from Abuhelwa et al. JPET 2015

**Usage**

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: -  $A^*_{last}$ : 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) -  $E^*$ : 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**

Abuhelwa, A. Y., Foster, D. J. R., Upton, R. N. (2015) ADVAN-style analytical solutions for common pharmacokinetic models. J Pharmacol Toxicol Methods 73:42-8. DOI: 10.1016/j.vascn.2015.03.004

---

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	<i>Parse observation types to simulation code</i>
-----------------	---

---

**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	<i>PK dataset</i>
--------	-------------------

---

**Description**

Example PK dataset

**Usage**

```
pkdata
```

**Format**

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

---

pkpsim_to_nlmixr	<i>Convert a model generated with PKPSim to an object for nlmixr</i>
------------------	--

---

**Description**

Convert a model generated with PKPSim 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.
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**

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



**Arguments**

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

**Value**

Input regiment minus selected number of doses

**See Also**

shift\_regimen

---

print.covariate	<i>Print function for PKPDsim covariate object</i>
-----------------	--

---

**Description**

Print function for PKPDsim covariate object

**Usage**

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

**Arguments**

x	covariate object
...	additional arguments

**Value**

No return value, print function.

---

print.PKPDsim	<i>Print function for PKPDsim simulation function</i>
---------------	---

---

**Description**

Print function for PKPDsim simulation function

**Usage**

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

**Arguments**

x	function
...	additional arguments

**Value**

No return value, print function.

---

print.regimen	<i>Print function for PKPDsim regimen</i>
---------------	---

---

**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	<i>Return a list in R syntax</i>
------------	----------------------------------

---

**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?

**Value**

Original list in R syntax

---

read_model_json	<i>Read model definition from JSON</i>
-----------------	--

---

**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	<i>Convert PKPDsim regimen to NONMEM table (doses only)</i>
---------------	---

---

**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	<i>Reparametrize model parameters using a reparametrization defined within the model.</i>
---------------	---

---

**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	<i>Find string and replace in file</i>
------------------------	--

---

**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	<i>Remove n doses (from start) of PKPDsim regimen</i>
---------------	---

---

**Description**

Opposite of pop\_regimen()

**Usage**

```
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	<i>R starts counting vector indices at 1, c++ starts at 0, so reduce all state numbers in the Cpp function definition by 1</i>
---------------------	--

---

**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**

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

**Arguments**

ode_def	ODE definition
n	add/subtract what number, default = -1

---

`sim`*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.
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 'mvrnorm2' 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

<code>int_step_size</code>	the step size for the numerical integrator
<code>t_max</code>	maximum simulation time, if not specified will pick the end of the regimen as maximum
<code>t_obs</code>	vector of observation times, only output these values (only used when <code>t_obs==NULL</code> )
<code>t_tte</code>	vector of observation times for time-to-event simulation
<code>t_init</code>	initialization time before first dose, default 0.
<code>obs_type</code>	vector of observation types. Only valid in combination with equal length vector <code>'t_obs'</code> .
<code>duplicate_t_obs</code>	allow duplicate <code>t_obs</code> in output? E.g. for optimal design calculations when <code>t_obs = c(0,1,2,2,3)</code> . Default is FALSE.
<code>extra_t_obs</code>	include extra <code>t_obs</code> in output for bolus doses? This is only activated when <code>'t_obs'</code> is not specified manually. E.g. for a bolus dose at <code>t=24</code> , 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 <code>'t_obs'</code> is specified manually.
<code>rtte</code>	should repeated events be allowed (FALSE by default)
<code>checks</code>	perform input checks? Default is TRUE. For calculations where <code>sim_ode</code> 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.
<code>verbose</code>	show more output
<code>return_event_table</code>	return the event table for the simulation only, does not run the actual simulation. Useful for iterative use of <code>sim()</code> .
<code>return_design</code>	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 <code>'sim_core()'</code> , e.g. for estimation and optimal design.
<code>output_include</code>	list specifying what to include in output table, with keys <code>'parameters'</code> and <code>'covariates'</code> . Both are FALSE by default.
<code>...</code>	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 <- 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
)

```

---

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

---

## 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

---

sim_ode	<i>Deprecated function, renamed to sim()</i>
---------	--

---

**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	<i>Simulate ODE and create a Shiny app</i>
---------------	--

---

**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	<i>Convert a table to a list</i>
---------------	----------------------------------

---

**Description**

Convert a table to a list

**Usage**

```
table_to_list(table)
```

**Arguments**

table	data.frame
-------	------------

**Value**

List containing original table contents

---

test_model	<i>Test a model</i>
------------	---------------------

---

**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	<i>Test if model still in memory</i>
--------------	--------------------------------------

---

**Description**

Test if model still in memory

**Usage**

test\_pointer(model)

**Arguments**

model	pointer to model
-------	------------------

**Value**

No return value

---

ThreeCompIVbolus	<i>IV bolus- 3 compartment</i>
------------------	--------------------------------

---

**Description**

IV bolus- 3 compartment

**Usage**

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	<i>first-order absorption- 3 compartment</i>
---------------	--

---

**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	<i>first-order absorption- 3 compartment-Metabolite</i>
--------------------	---

---

**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	<i>Translate a model from/to various PKPD simulators</i>
---------------	--

---

**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 PKDPsim or RxODE model

---

triangle_to_full	<i>Convert triangle omega matrix to full omega matrix</i>
------------------	---

---

**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	<i>IV bolus- 2 compartment</i>
----------------	--------------------------------

---

**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	<i>IV infusion- 2 compartment</i>
-------------------	-----------------------------------

---

**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
- mvrnorm2, 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](#)