

Package: nlmixr2lib (via r-universe)

October 14, 2026

Type Package

Version 0.3.2.9000

Title A Model Library for 'nlmixr2'

Description A model library for 'nlmixr2'. The models include (and plan to include) pharmacokinetic, pharmacodynamic, and disease models used in pharmacometrics. Where applicable, references for each model are included in the meta-data for each individual model. The package also includes model composition and modification functions to make model updates easier.

Depends R (>= 4.1)

Imports checkmate, cli, methods, nlmixr2est, rxode2 (>= 5.1.8)

License GPL (>= 2)

LazyData true

VignetteBuilder knitr

URL <https://github.com/nlmixr2/nlmixr2lib>,
<https://nlmixr2.github.io/nlmixr2lib/>

Encoding UTF-8

Language en-US

Suggests callr, covr, dplyr, ggplot2, knitr, mvtnorm, nlmixr2data, pkgload, PKNCA, rmarkdown, scales, survival, testthat (>= 3.0.0), tibble, tidyr, withr, yaml

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libgmp3-dev make libicu-dev

Repository <https://nlmixr2.r-universe.dev>

Date/Publication 2026-10-10 20:43:11 UTC

RemoteUrl <https://github.com/nlmixr2/nlmixr2lib>

RemoteRef HEAD

RemoteSha f6c55c60f0e385e5bd7418c1959db41dace9e887

Contents

addBaseline1exp	3
addBaselineConst	4
addBaselineExp	5
addBaselineLin	6
addCmtProp	7
addDepot	8
addDirectLin	9
addDirToModelDb	10
addEffectCmtLin	11
addEta	12
addIndirect	13
addIndirectLin	14
addLogEstimates	16
addPeriph	17
addResErr	18
addSecondAbsorption	19
addTransit	20
addWeibullAbs	21
addZeroOrderAbs	23
checkModelConventions	24
checkNamingRegisters	25
combinePaste2	26
convertAbsForceLongerDelay	27
convertAbsSequential	28
convertEmax	29
convertEmaxHill	30
convertKinR0	31
convertLogLin	32
convertMM	33
convertQuad	34
createMarkovModel	35
createMarkovModelDataset	35
createMarkovModelFromSingleState	37
createMarkovTransitionMatrix	37
cvToBsv	38
defaultCombine	39
fakeCc	40
Mann2022Equilibrate	41
modeldb	42
modellib	43
ncaComparisonTable	44
ncaParamLabel	45
pkTrans	46
Poels_2025_elranatamab_qsp_cauc_multiplier	49
Poels_2025_elranatamab_qsp_events	50
readModelDb	51

addBaseline1exp 3

removeDepot	52
removeLinesAndInis	53
removePeriph	53
removeTransit	54
removeZeroOrderAbs	55
searchReplace	56
setCombineType	57
simMarkov	57
simMarkovId	58
updateOncologyXenograftSimeoni2004	59

Index 61

<code>addBaseline1exp</code>	<i>Add baseline that decays exponential with time</i>
------------------------------	---

Description

Add baseline that decays exponential with time

Usage

```
addBaseline1exp(ui, effect = "effect", eb = "Eb", time = "time", kb = "kb")
```

Arguments

<code>ui</code>	rxode2 model
<code>effect</code>	the effect variable that will be modeled
<code>eb</code>	baseline constant parameter
<code>time</code>	the time or other variable used for baseline decay
<code>kb</code>	the first order baseline decay constant

Value

model with baseline constant

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addDirectLin() |>
  convertQuad() |>
  addBaseline1exp()
```

addBaselineConst	<i>Add an estimated baseline constant</i>
------------------	---

Description

Add an estimated baseline constant

Usage

```
addBaselineConst(ui, effect = "effect", eb = "Eb")
```

Arguments

ui	rxode2 model
effect	the effect variable that will be modeled
eb	baseline constant parameter

Value

model with baseline constant

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addDirectLin() |>
  convertQuad() |>
  addBaselineConst()
```

addBaselineExp	<i>Add baseline that decays exponential with time</i>
----------------	---

Description

Add baseline that decays exponential with time

Usage

```
addBaselineExp(ui, effect = "effect", eb = "Eb", time = "time", kb = "kb")
```

Arguments

ui	rxode2 model
effect	the effect variable that will be modeled
eb	baseline constant parameter
time	the time or other variable used for baseline decay
kb	the first order baseline decay constant

Value

model with baseline constant

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>  
  addDirectLin() |>  
  convertQuad() |>  
  addBaselineExp()
```

addBaselineLin	<i>Add an estimated baseline linear constant</i>
----------------	--

Description

Add an estimated baseline linear constant

Usage

```
addBaselineLin(ui, effect = "effect", eb = "Eb", time = "time")
```

Arguments

ui	rxode2 model
effect	the effect variable that will be modeled
eb	baseline constant parameter
time	the time or other variable used for baseline decay

Value

model with baseline linear constant

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>  
  addDirectLin() |>  
  convertQuad() |>  
  addBaselineLin()
```

addCmtProp	<i>Add a property to a compartment</i>
------------	--

Description

Add a property to a compartment

Usage

```
addCmtProp(ui, prop = c("f", "lag", "dur", "rate", "ini"), cmt)

addBioavailability(ui, cmt, scale = c("log", "logit"), cmt2 = NULL, f = 0.8)

addLag(ui, cmt)

addDur(ui, cmt)

addRate(ui, cmt)

addIni(ui, cmt)
```

Arguments

ui	rxode2 ui object
prop	property to add to a compartment: - F: bioavailability - lag: absorption lag time - dur: modeled duration of infusion - rate: modeled infusion rate - ini: initial value of the compartment
cmt	compartment to apply the property to
scale	parameterization of the bioavailability: "log" (default) estimates it unboundedly on the log scale, while "logit" constrains it to (0,1) with a logit/expit parameterization (Monolix-style F, oral or subcutaneous)
cmt2	optional second compartment for the logit dose-split form; see [addBioavailabilityLogit()]
f	initial bioavailability fraction, in (0,1), for the logit form, or NULL to leave the initial estimate unset

Value

rxode2 ui object with property applied

Functions

- `addBioavailability()`: Adds the bioavailability to a compartment in the model
- `addLag()`: Adds the lag-time to a compartment in the model
- `addDur()`: Adds the modeled duration to a compartment in the model
- `addRate()`: Adds the modeled rate to a compartment in the model
- `addIni()`: Adds the initial value to the compartment

Author(s)

Matthew L. Fidler

Examples

```
readModelDb("PK_3cmt_des") |> addCmtProp("f", "depot")

readModelDb("PK_3cmt_des") |> addBioavailability(depota)

readModelDb("PK_3cmt_des") |> addLag(depota)

readModelDb("PK_3cmt_des") |> addDur(depota)

readModelDb("PK_3cmt_des") |> addRate(depota)

readModelDb("PK_3cmt_des") |> addIni(depota)

# bioavailability bounded to (0,1) with a logit parameterization
readModelDb("PK_1cmt_des") |> addBioavailability(depota, scale = "logit")

# dose split between two absorption paths, fraction named for depota
readModelDb("PK_1cmt_des") |>
  addDepot(depota = "depota2", ka = "ka2") |>
  addBioavailability(depota, depota2, scale = "logit")
```

<code>addDepot</code>	<i>To convert from infusion/intravenous administration to first-order oral absorption</i>
-----------------------	---

Description

To convert from infusion/intravenous administration to first-order oral absorption

Usage

```
addDepot(ui, central = "central", depota = "depota", ka = "ka", model)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
central	central compartment name
depot	depot compartment name
ka	absorption rate parameter name
model	Deprecated alias for ui. Supplying model instead of ui still works but emits a deprecation warning.

Value

a model with the depot added

Examples

```
# most of the examples in the model library already have a depot
# the PK_2cmt_no_depot is an exception
readModelDb("PK_2cmt_no_depot") |> addDepot()
```

addDirectLin	<i>Add direct linear effect with baseline=0</i>
--------------	---

Description

Add direct linear effect with baseline=0

Usage

```
addDirectLin(ui, ek = "Ek", cc = c("Ce", "Cc"), effect = "effect")
```

Arguments

ui	rxode2 model
ek	simulation linear constant
cc	the concentration value
effect	the effect variable that will be modeled

Value

model with direct linear effect added (baseline=0)

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
# Direct linear model
readModelDb("PK_2cmt_no_depot") |>
  addDirectLin()

# Direct emax model
readModelDb("PK_2cmt_no_depot") |>
  addDirectLin() |>
  convertEmax()
```

addDirToModelDb	<i>Add a directory to the modeldb</i>
-----------------	---------------------------------------

Description

Add a directory to the modeldb

Usage

```
addDirToModelDb(dir, modeldb = data.frame())

addFileToModelDb(dir, file, modeldb)
```

Arguments

dir	Directory name containing model files
modeldb	The starting modeldb data.frame
file	The file name (without the directory name)

Value

The updated modeldb data.frame
the model database

Functions

- `addFileToModelDb()`: Add a file to the modeldb

addEffectCmtLin	<i>Add effect compartment</i>
-----------------	-------------------------------

Description

Add effect compartment

Usage

```
addEffectCmtLin(  
  ui,  
  ke0 = "ke0",  
  cc = "Cc",  
  ce = "Ce",  
  ek = "Ek",  
  effect = "effect"  
)
```

Arguments

ui	rxode2 model
ke0	This is the effect compartment keo rate
cc	the concentration value
ce	This is the concentration in the effect compartment
ek	simulation linear constant
effect	the effect variable that will be modeled

Value

a model with an effect compartment attached

Author(s)

Matthew L. Fidler

Examples

```
readModelDb("PK_2cmt_no_depot") |>  
  addEffectCmtLin()  
  
# Can also be changed to the more typical Emax with constant (estimated) baselie  
readModelDb("PK_2cmt_no_depot") |>  
  addEffectCmtLin() |>  
  convertEmaxHill() |>  
  addBaselineConst()
```

addEta	<i>Add random effects to a model</i>
--------	--------------------------------------

Description

Add random effects to a model

Usage

```
addEta(  
  ui,  
  eta,  
  priorName = getOption("nlmixr2lib.priorEta", TRUE),  
  etaCombineType = c("default", "snake", "camel", "dot", "blank"),  
  model  
)
```

Arguments

ui	The model as a function
eta	vector with the parameters to add random effects (sometimes referred to as inter-individual variability, IIV) on
priorName	logical, if TRUE, the parameter name specified in 'eta' will be used to add the eta value prior name is used instead of the left handed side of the equation.
etaCombineType	the option for the how to combine the eta with the parameter name. Can be: "default", "snake", "camel", "dot", "blank"
model	Deprecated alias for ui. Supplying model instead of ui still works but emits a deprecation warning.

Value

The model with eta added to the requested parameters

Author(s)

Bill Denney, Richard Hooijmaijers & Matthew L. Fidler

Examples

```
library(rxode2)  
readModelDb("PK_1cmt") |> addEta("ka")
```

addIndirect	<i>Add an indirect response model to a PK model</i>
-------------	---

Description

Add an indirect response model to a PK model

Usage

```
addIndirect(
  ui,
  stim = c("in", "out"),
  inhib = c("in", "out"),
  hill = FALSE,
  ek = "Ek",
  ik = "Ik",
  emax = "Emax",
  ec50 = "EC50",
  imax = "Imax",
  ic50 = "IC50",
  kin = "kin",
  kout = "kout",
  g = "g",
  cc = "Cc",
  R = "R",
  effect = "effect"
)
```

Arguments

ui	rxode2 model
stim	what type of stimulation indirect response model: - 'in': stimulation of input - 'out': stimulation of output
inhib	what type of inhibition indirect response model: - 'in': inhibition of input - 'out': inhibition of output
hill	Boolean stating if a hill sigmoid coefficient will be added
ek	simulation linear constant
ik	inhibition linear constant
emax	Emax parameter
ec50	EC50 parameter
imax	maximum inhibitory concentration
ic50	concentration where half of the I _{max} occurs

kin	this is the kin parameter name
kout	this is the kout parameter name
g	hill coefficient
cc	the concentration value
R	drug response compartment
effect	the effect variable that will be modeled

Value

pk model with indirect response model added

Author(s)

Matthew L. Fidler

See Also

Other Indirect response: [addIndirectLin\(\)](#), [convertKinR0\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addIndirect(stim = "in", hill = TRUE)

readModelDb("PK_2cmt_no_depot") |>
  addIndirect(inhib = "out", imax = 1)
```

addIndirectLin	<i>Add linear indirect response model</i>
----------------	---

Description

Add linear indirect response model

Usage

```
addIndirectLin(
  ui,
  stim = c("in", "out"),
  inhib = c("in", "out"),
  ek = "Ek",
  ik = "Ik",
  kin = "kin",
  kout = "kout",
  cc = "Cc",
  R = "R",
  effect = "effect"
)
```

Arguments

ui	rxode2 model
stim	what type of stimulation indirect response model: - 'in': stimulation of input - 'out': stimulation of output
inhib	what type of inhibition indirect response model: - 'in': inhibition of input - 'out': inhibition of output
ek	simulation linear constant
ik	inhibition linear constant
kin	this is the kin parameter name
kout	this is the kout parameter name
cc	the concentration value
R	drug response compartment
effect	the effect variable that will be modeled

Value

model with linear indirect effect added

Note that while linear indirect effects are not common, it allows an easier hook to produce other standard effect curves like Emax/Imax, Hill, etc.

Author(s)

Matthew L. Fidler

See Also

Other Indirect response: [addIndirect\(\)](#), [convertKinR0\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |> addIndirectLin(stim = "in")  
readModelDb("PK_2cmt_no_depot") |> addIndirectLin(stim = "out")  
readModelDb("PK_2cmt_no_depot") |> addIndirectLin(inhib = "in")  
readModelDb("PK_2cmt_no_depot") |> addIndirectLin(inhib = "out")
```

addLogEstimates	<i>Add log estimates to a model</i>
-----------------	-------------------------------------

Description

Add log estimates to a model

Usage

```
addLogEstimates(ui, vars, extraLines = NULL, beforeCmt = NULL)
```

Arguments

ui	rxode2 model
vars	estimates to add they will be parameterized as: var <- exp(lvar) where var is the variable name in the model and lvar is the log transformed variable that will be estimated
extraLines	this is a list of additional lines to add to the model just after the variables are defined. It must be NULL or a list of language objects.
beforeCmt	if the model is compartmental you can specify the preferred names where the estimates and extra lines are added before

Value

rxode2 model with log estimates added (and possibly extra lines)

Author(s)

Matthew L. Fidler

Examples

```
# Change the transformation of the PK model from cl to k

readModelDb("PK_3cmt_des") |>
  removeLinesAndInis(c("kel", "k12", "k21", "k13", "k31", "vc")) |>
  addLogEstimates(c("kel", "k12", "k21", "k13", "k31", "vc"))

# You can also label the parameters by using a named character
# vector with the names of the parameters representing the
# variables and the values representing the labels:

readModelDb("PK_3cmt_des") |>
  removeLinesAndInis(c("kel", "k12", "k21", "k13", "k31", "vc")) |>
  addLogEstimates(c(kel = "elimination", k12 = "k12 constant",
    k21 = "k21 constant",
    k13 = "k13 constant",
```

```
k31 = "k31 constant",
vc = "volume of central compartment"))
```

addPeriph

*Add a peripheral compartment to a model***Description**

Adds a peripheral distribution compartment ('peripheral1', then 'peripheral2') to a central compartment, following the 'PK_2cmt_des'/'PK_3cmt_des' seed conventions: intercompartmental clearance 'q' ('q2' for the second) and peripheral volume 'vp' ('vp2'), with 'k12 <- q/vc', 'k21 <- q/vp' (and 'k13'/'k31' for the second). At most two peripheral compartments are supported, matching the seeds.

Usage

```
addPeriph(ui, n = NULL, central = "central", model)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
n	which peripheral to add: 1 ('peripheral1', 'q', 'vp') or 2 ('peripheral2', 'q2', 'vp2'); defaults to the first missing one
central	central compartment name
model	Deprecated alias for ui. Supplying model instead of ui still works but emits a deprecation warning.

Value

a model with the peripheral compartment added

Author(s)

Matthew L. Fidler

See Also

Other distribution: [removePeriph\(\)](#)

Examples

```
readModelDb("PK_1cmt_des") |> addPeriph()

readModelDb("PK_2cmt_des") |> addPeriph()
```

addResErr	<i>Add residual error to a model</i>
-----------	--------------------------------------

Description

Add residual error to a model

Usage

```
addResErr(ui, reserr, endpoint, model)
```

Arguments

ui	The model as a function
reserr	The type or types of residual error (currently "addSd", "propSd", and "lnormSd" are accepted)
endpoint	the endpoint to apply the error; will default to the first error in the model
model	Deprecated alias for ui. Supplying model instead of ui still works but emits a deprecation warning.

Details

For reserr, the parameter will be named with the dependent variable from the model as a prefix. For example, if the dependent variable in the model is Cc, the parameter name for propSd will become CcpropSd.

Value

The model with residual error modified

Examples

```
library(rxode2)
readModelDb("PK_1cmt") |> addResErr("addSd")
readModelDb("PK_1cmt") |> addResErr("lnormSd")
readModelDb("PK_1cmt") |> addResErr(c("addSd", "propSd"))
```

addSecondAbsorption *Add a second absorption path for double-absorption models*

Description

Adds a parallel absorption path ('depot2') to a model that already has first-order absorption through 'depot', for Monolix-style double-absorption models: one dose record is split at translation time into two parallel inputs with a logit-parameterized 'F1' apportionment. The second path can itself be zero-order ('type = "zero"', a modeled 'dur(depot2)' input), first-order ('type = "first"', 'ka2'), or a transit chain ('delay = "transit"', with an optional lag.

Usage

```
addSecondAbsorption(
  ui,
  type = c("first", "zero"),
  delay = c("none", "lag", "transit"),
  n = NULL,
  central = "central",
  depot = "depot",
  depot2 = "depot2",
  ka = "ka",
  ka2 = "ka2",
  tk0 = "tk0",
  f1 = 0.7
)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
type	second-path input type: "zero" for a zero-order (modeled-duration) input or "first" for a first-order ('ka2') input
delay	second-path delay: "none", "lag" (an 'alag' on the second path), or "transit" (a transit chain feeding the second path)
n	number of transit compartments when 'delay = "transit"'
central	central compartment name
depot	depot compartment name
depot2	name of the second depot compartment
ka	absorption rate parameter name
ka2	name of the second first-order absorption rate
tk0	name of the zero-order duration on the second path
f1	initial fraction of the dose entering the first path, in (0,1)

Details

Only one second absorption path is supported per model; applying this twice raises an error.

Value

a model with two parallel absorption paths

Author(s)

Matthew L. Fidler

See Also

Other absorption: [addTransit\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [convertAbsSequential\(\)](#), [removeTransit\(\)](#), [removeZeroOrderAbs\(\)](#)

Examples

```
# simultaneous zero-order + first-order (Monolix double absorption)
readModelDb("PK_1cmt_des") |>
  addSecondAbsorption(type = "first", delay = "lag", f1 = 0.7)
```

addTransit

To add transit compartments to the model

Description

To add transit compartments to the model

Usage

```
addTransit(
  ui,
  ntransit,
  central = "central",
  depot = "depot",
  transit = "transit",
  ktr = "ktr",
  ka = "ka",
  model
)
```

Arguments

<code>ui</code>	The model as a function
<code>ntransit</code>	the number of transit compartments to be added
<code>central</code>	central compartment name
<code>depot</code>	depot compartment name
<code>transit</code>	the transit compartment prefix
<code>ktr</code>	the parameter name for the transit compartment rate
<code>ka</code>	absorption rate parameter name
<code>model</code>	Deprecated alias for <code>ui</code> . Supplying <code>model</code> instead of <code>ui</code> still works but emits a deprecation warning.

Value

a model with transit compartment added

This matches

```
'dose->a0->a1->abs cmt->central'
```

But 'a0' is depot so dosing records labeled depot do not need to be changed

The abs cmt becomes the last "transit" compartment

This is simply for convenience

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [convertAbsSequential\(\)](#), [removeTransit\(\)](#), [removeZeroOrderAbs\(\)](#)

Examples

```
readModelDb("PK_1cmt_des") |> addTransit(3)
```

addWeibullAbs

Converts first order absorption model to Weibull absorption model

Description

Converts first order absorption model to Weibull absorption model

Usage

```
addWeibullAbs(  
  ui,  
  ntransit,  
  central = "central",  
  depot = "depot",  
  transit = "transit",  
  wa = "wa",  
  wb = "wb",  
  ka = "ka",  
  ktr = "ktr"  
)
```

Arguments

ui	The model as a function
ntransit	the number of transit compartments to be added
central	central compartment name
depot	depot compartment name
transit	the transit compartment prefix
wa	Weibull alpha parameter name
wb	Weibull beta parameter name
ka	absorption rate parameter name
ktr	the parameter name for the transit compartment rate

Value

model where first order absorption is changed to Weibull absorption model

Author(s)

Matthew L. Fidler

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addTransit\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [convertAbsSequential\(\)](#), [removeTransit\(\)](#), [removeZeroOrderAbs\(\)](#)

Examples

```
readModelDb("PK_1cmt_des") |>  
  addWeibullAbs()
```

addZeroOrderAbs	<i>Convert a model to zero-order absorption (Monolix Tk0)</i>
-----------------	---

Description

Zero-order absorption releases the dose into the central compartment at a constant rate over a modeled duration Tk0, matching Monolix's absorption(type=2, Tk0) oral route. Following monolix2rx's translation, this is expressed as a modeled duration on the central compartment (`dur(central) <- tk0`), so a depot compartment (and its ka) is removed first. Dosing records for the zero-order route must request the modeled duration in the event table (`RATE = -2`; see `et(rate=-2)`); ordinary bolus records bypass the modeled duration.

Usage

```
addZeroOrderAbs(
  ui,
  central = "central",
  depot = "depot",
  transit = "transit",
  ktr = "ktr",
  ka = "ka",
  tk0 = "tk0"
)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
central	central compartment name
depot	depot compartment name
transit	the transit compartment prefix
ktr	the parameter name for the transit compartment rate
ka	absorption rate parameter name
tk0	zero-order absorption duration parameter name (Monolix's Tk0)

Details

Lag time and bioavailability can be combined with `[addLag()]` and `[addLogitBioavailability()]`.

Value

a model with zero-order absorption

Author(s)

Matthew L. Fidler

See Also

Other absorption: `addSecondAbsorption()`, `addTransit()`, `addWeibullAbs()`, `convertAbsForceLongerDelay()`, `convertAbsSequential()`, `removeTransit()`, `removeZeroOrderAbs()`

Examples

```
readModelDb("PK_1cmt_des") |> addZeroOrderAbs()

# a model without a depot gets the same modeled duration input
readModelDb("PK_2cmt_no_depot") |> addZeroOrderAbs()
```

checkModelConventions *Check a model against nlmixr2lib conventions*

Description

Parses a model and reports deviations from the nlmixr2lib conventions documented in ‘vignettes/create-model-library.Rmd’ and the ‘inst/references/*.md’ registers (especially ‘inst/references/covariate-columns.md’). The checker inspects: file-level metadata (description, reference, units, covariate-Data); parameter names (log-prefix PK params, ‘eta’-prefix IIV, ‘propSd’/‘addSd’ residual error); parameter labels; covariates (canonical register, units, declared aliases); compartment names; the observation variable (‘Cc’); and a syntactic dosing-vs-concentration unit cross-check.

Usage

```
checkModelConventions(model, verbose = TRUE)
```

Arguments

model	Model to check. Accepted forms: - character scalar: resolved via <code>[readModelDb()]</code> . - function: evaluated via <code>'nlmixr2est::nlmixr()'</code> . - <code>'rxUi'</code> object: used directly. - missing / <code>'NULL'</code> : iterate over every model in <code>'modeldb'</code> .
verbose	Logical; if <code>'TRUE'</code> (default) print the per-category report to the console via <code>'cli'</code> .

Details

When any issue of severity `"error"` or `"warning"` is found, a `'warning()'` is emitted whose message directs the user to call this function for the full report.

Value

Invisibly, a `'data.frame'` with one row per issue and columns `'model'`, `'category'`, `'severity'`, `'name'`, `'message'`, `'suggestion'`.

Examples

```
## Not run:
checkModelConventions("PK_1cmt_des")
checkModelConventions()          # check every model in modeldb

## End(Not run)
```

checkNamingRegisters *Check the naming registers for structural defects*

Description

Validates ‘inst/references/*.md’, the registers that define canonical covariate, parameter and compartment names, rather than the models that use them. Complements [checkModelConventions()], which checks the reverse direction.

Usage

```
checkNamingRegisters(root = NULL)
```

Arguments

root Package root to check. ‘NULL’ (default) uses the installed package.

Details

Checks performed:

- duplicate canonical entries (globally for covariates; per section for compartments, where a token may legitimately be both a bare compartment and a suffix);
- citations of model files that do not exist on disk;
- canonicals that no model uses;
- ‘[[cross-references]]’ with no matching entry;
- entries with no ‘Example models:’ line, outside an explicit exemption list of universal parameters;
- entries with no ‘Type:’ line, or whose ‘Type:’ is outside the known vocabulary. A missing ‘Type:’ is the quietest defect of the set: the entry drops out of the canonical name list [checkModelConventions()] builds, so the model-facing check stops recognising the name while every other register check stays green.
- covariate canonicals that are not ALL CAPS and are not listed under ‘## Case convention’ in ‘covariate-columns.md’, and names listed there that no longer have an entry. Checked for that register only: parameter canonicals are deliberately lower case (‘lka’, ‘lcl’).

Value

A data.frame with one row per issue and columns ‘register’, ‘check’, ‘name’, ‘line’, ‘detail’. Zero rows means clean.

`combinePaste2`*Combine two strings using a naming convention*

Description

Combine two in a manner similar to `'paste()'` strings using the default combine type

Usage

```
combinePaste2(  
  a,  
  b,  
  combineType = c("default", "snake", "camel", "dot", "blank")  
)
```

Arguments

<code>a</code>	first string to combine
<code>b</code>	second string to combine
<code>combineType</code>	is the type of combination; can be: - <code>"default"</code>: default combine (set with <code>'defaultCombine()'</code>) - <code>"camel"</code>: camelCase combine - <code>"snake"</code>: snake_case combine - <code>"dot"</code>: dot combine (i.e. <code>"a.b"</code>) - <code>"blank"</code>: no separator (i.e. <code>"ab"</code>)

Value

Combined strings separated with `'defaultCombine()'`

Author(s)

Matthew L. Fidler

Examples

```
combinePaste2("f", "depot")  
  
combinePaste2("f", "depot", "snake")  
  
combinePaste2("f", "depot", "dot")  
  
combinePaste2("f", "depot", "blank")
```

`convertAbsForceLongerDelay`*Force the second absorption delay longer than the first*

Description

Reparameterizes a double-absorption model so the second path's lag is estimated as a positive increment over the first path's lag ($Tlag2 = Tlag1 + diffTlag2$), matching Monolix's "force delay2 longer than delay1" models. Both paths must have lag times and the model must not already be sequential.

Usage

```
convertAbsForceLongerDelay(  
  ui,  
  central = "central",  
  depot = "depot",  
  depot2 = "depot2"  
)
```

Arguments

<code>ui</code>	The model as a function (or something convertible to an rxUi object)
<code>central</code>	central compartment name
<code>depot</code>	depot compartment name
<code>depot2</code>	name of the second depot compartment

Value

a model with the second lag tied above the first

Author(s)

Matthew L. Fidler

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addTransit\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsSequential\(\)](#), [removeTransit\(\)](#), [removeZeroOrderAbs\(\)](#)

convertAbsSequential *Convert double absorption to sequential delays*

Description

Reparameterizes a double-absorption model so the second path's delay is expressed relative to the first path's zero-order duration ('Tlag2 = Tk01'), matching Monolix's "sequential" double-absorption models. Only valid when the first path absorbs zero-order; raises an error otherwise.

Usage

```
convertAbsSequential(  
  ui,  
  central = "central",  
  depot = "depot",  
  depot2 = "depot2"  
)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
central	central compartment name
depot	depot compartment name
depot2	name of the second depot compartment

Value

a model with the second delay tied to the first duration

Author(s)

Matthew L. Fidler

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addTransit\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [removeTransit\(\)](#), [removeZeroOrderAbs\(\)](#)

`convertEmax`*Convert linear effect to Emax effect*

Description

Convert linear effect to Emax effect

Usage

```
convertEmax(  
  ui,  
  emax = "Emax",  
  ec50 = "EC50",  
  imax = "Imax",  
  ic50 = "IC50",  
  ek = c("Ik", "Ek"),  
  cc = c("Ec", "Cc")  
)
```

Arguments

<code>ui</code>	rxode2 model
<code>emax</code>	Emax parameter
<code>ec50</code>	EC50 parameter
<code>imax</code>	Imax parameter used when input model contains "Ik" instead of "Ek"
<code>ic50</code>	IC50 parameter used when input model contains "Ik" instead of "Ek"
<code>ek</code>	simulation linear constant
<code>cc</code>	the concentration value

Value

Model with the linear effect converted to an Emax effect

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertLogLin\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "in") |>
  convertEmax()

# When emax=1
readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "in") |>
  convertEmax(emax = 1)
```

convertEmaxHill	<i>Convert linear effect to Emax-Hill effect</i>
-----------------	--

Description

Convert linear effect to Emax-Hill effect

Usage

```
convertEmaxHill(
  ui,
  emax = "Emax",
  ec50 = "EC50",
  g = "g",
  imax = "Imax",
  ic50 = "IC50",
  ek = c("Ik", "Ek"),
  cc = c("Ec", "Cc")
)
```

Arguments

ui	rxode2 model
emax	Emax parameter
ec50	EC50 parameter
g	hill coefficient
imax	Imax parameter used when input model contains "Ik" instead of "Ek"
ic50	IC50 parameter used when input model contains "Ik" instead of "Ek"
ek	simulation linear constant
cc	the concentration value

Value

Model with the linear effect converted to an Emax effect

Author(s)

Matthew L. Fidler

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "in") |>
  convertEmaxHill()
```

```
# can also specify as emax=1
```

```
readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "in") |>
  convertEmaxHill(emax = 1)
```

convertKinR0

*Convert a kin/kout indirect response model to R0 and kout***Description**

This replaces the kin/kout parameterization to the R0 and kout parameterization

Usage

```
convertKinR0(ui, kin = "kin", kout = "kout", R = "R", R0 = "R0")
```

Arguments

ui	a rxode2 user function
kin	the kin variable (by default is "kin")
kout	the kout variable (by default is "kout")
R	the compartment variable (by default is "R")
R0	the R0 variable (by default is "R0")

Value

a model where the estimated kin is changed to the estimated R0

Author(s)

Matthew L. Fidler

See Also

Other Indirect response: [addIndirect\(\)](#), [addIndirectLin\(\)](#)

Examples

```
addIndirect(stim = "in") |> convertKinR0()
```

 convertLogLin

Converts a linear effect to a log-linear effect

Description

Converts a linear effect to a log-linear effect

Usage

```
convertLogLin(ui, ek = c("Ik", "Ek"), cc = c("Ce", "Cc"))
```

Arguments

ui	rxode2 model
ek	simulation linear constant
cc	the concentration value

Value

model converted from linear to log-linear effect

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertQuad\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addDirectLin() |>
  convertLogLin()

readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "out") |>
  convertLogLin()
```

convertMM	<i>Convert models from linear elimination to Michaelis-Menten elimination</i>
-----------	---

Description

Convert models from linear elimination to Michaelis-Menten elimination

Usage

```
convertMM(  
  ui,  
  central = "central",  
  elimination = "kel",  
  vm = "vm",  
  km = "km",  
  vc = "vc"  
)
```

Arguments

ui	model to convert
central	the central compartment where the elimination is present
elimination	variable for the elimination constant in the model
vm	variable name for Vmax in the model
km	variable name for Km in the model
vc	variable name for Vc in the model

Value

new model changing linear elimination to Michaelis-Menten elimination

Author(s)

Matthew L. Fidler

Examples

```
readModelDb("PK_1cmt_des") |> convertMM()  
  
readModelDb("PK_2cmt_des") |> convertMM()  
  
readModelDb("PK_3cmt_des") |> convertMM()  
  
readModelDb("PK_3cmt_des") |> removeDepot() |> convertMM()
```

`convertQuad`*Convert linear elimination to quadratic elimination*

Description

Convert linear elimination to quadratic elimination

Usage

```
convertQuad(ui, ek = c("Ik", "Ek"), cc = c("Ce", "Cc"), ek2 = "Ek2")
```

Arguments

<code>ui</code>	rxode2 model
<code>ek</code>	simulation linear constant
<code>cc</code>	the concentration value
<code>ek2</code>	quadratic coefficient

Value

model with linear effect converted to quadratic effect

Author(s)

Matthew L. Fidler

See Also

Other PD: [addBaseline1exp\(\)](#), [addBaselineConst\(\)](#), [addBaselineExp\(\)](#), [addBaselineLin\(\)](#), [addDirectLin\(\)](#), [convertEmax\(\)](#), [convertLogLin\(\)](#)

Examples

```
readModelDb("PK_2cmt_no_depot") |>
  addIndirectLin(stim = "out") |>
  convertQuad()

readModelDb("PK_2cmt_no_depot") |>
  addDirectLin() |>
  convertQuad()

readModelDb("PK_2cmt_no_depot") |>
  addEffectCmtLin() |>
  convertQuad()
```

createMarkovModel	Create a Markov model for 'nlmixr2'
-------------------	-------------------------------------

Description

Create a Markov model for 'nlmixr2'

Usage

```
createMarkovModel(..., ignoreProbLt = 0, ignoreProbGt = 1, transitions = NULL)
```

Arguments

...	Passed to 'createMarkovTransitionMatrix()'
ignoreProbLt	Do not estimate probabilities less than the given probability. Set to 0 to estimate all probabilities with any transition chance, use 'estimateZeroTransitions' to give an initial estimate when there is zero probability of transition into the state.
ignoreProbGt	Do not estimate probabilities on a row where any probability is greater than the given probability (treating it as a collector row). Set to 1 to estimate all probabilities with any transition chance away from the state, use 'estimateZeroTransitions' to give an initial estimate when there is zero probability of transition from the state.
transitions	Use this manually-created transition matrix rather than an automatically-creating one

Value

A template 'nlmixr2' model as a character string with 'ini()' and 'model()' blocks for the Markov model

See Also

Other Markov models: [createMarkovModelDataset\(\)](#), [createMarkovModelFromSingleState\(\)](#), [createMarkovTransitionMatrix\(\)](#), [simMarkov\(\)](#), [simMarkovId\(\)](#)

createMarkovModelDataset	Create a Markov model dataset for use in estimating Markov models
--------------------------	---

Description

Create a Markov model dataset for use in estimating Markov models

Usage

```
createMarkovModelDataset(x, ...)

## Default S3 method:
createMarkovModelDataset(
  x,
  colCur,
  colPrev = x,
  prefixPrev = "prev",
  prefixCur = "cur",
  ...
)

## S3 method for class 'data.frame'
createMarkovModelDataset(x, colPrev, colCur, ...)
```

Arguments

x	The object to create a Markov dataset within
...	Passed to methods
colPrev, colCur	Column names in the dataset for the previous and current states (may be any R object that can be coerced to a character string, ordered objects are usually preferred)
prefixPrev, prefixCur	Column name prefixes for the previous and current states

Value

The data.frame modified by adding one-hot columns for the previous and current states

The data.frame modified by adding one-hot columns for the previous and current states

Methods (by class)

- `createMarkovModelDataset(default)`: Create a Markov dataset from a pair of vectors
- `createMarkovModelDataset(data.frame)`: Create a Markov dataset from a data.frame

See Also

Other Markov models: [createMarkovModel\(\)](#), [createMarkovModelFromSingleState\(\)](#), [createMarkovTransitionMatrix\(\)](#), [simMarkov\(\)](#), [simMarkovId\(\)](#)

Examples

```
createMarkovModelDataset(c(1, 2, 1), colCur = c(1, 2, 3))
```

`createMarkovModelFromSingleState`*Create the parts of a Markov model for transitioning from a single state*

Description

Create the parts of a Markov model for transitioning from a single state

Usage

```
createMarkovModelFromSingleState(transitionRow, stateNames)
```

Arguments

<code>transitionRow</code>	a single-element named list with a named vector of all transitions
<code>stateNames</code>	a named vector of state names where the name is the name for use in the model parameters

Value

A list with two elements, "ini" and "model", where each element is a character vector of lines of code for the model

See Also

Other Markov models: [createMarkovModel\(\)](#), [createMarkovModelDataset\(\)](#), [createMarkovTransitionMatrix\(\)](#), [simMarkov\(\)](#), [simMarkovId\(\)](#)

`createMarkovTransitionMatrix`*Create a Markov transition matrix with probabilities of transitioning between every state*

Description

Create a Markov transition matrix with probabilities of transitioning between every state

Usage

```
createMarkovTransitionMatrix(  
  colPrev,  
  colCur,  
  estimateZeroTransitions = FALSE,  
  estimateZeroTransitionsInitial = FALSE,  
  ...  
)
```

Arguments

- colPrev, colCur Column names in the dataset for the previous and current states (may be any R object that can be coerced to a character string, ordered objects are usually preferred)
- estimateZeroTransitions
Should transitions that have zero occurrences be estimated? This is done by setting the state to have a single transition.
- estimateZeroTransitionsInitial
Should transitions that are only initial states be estimated (ignored if 'estimateZeroTransitions = FALSE')
- ... Ignored

Value

A square matrix with row and column names for each state where rows are the prior state and columns are the current state.

See Also

Other Markov models: [createMarkovModel\(\)](#), [createMarkovModelDataset\(\)](#), [createMarkovModelFromSingleState\(\)](#), [simMarkov\(\)](#), [simMarkovId\(\)](#)

cvToBsv	<i>Convert coefficient of variation (percent) to between-subject variance</i>
---------	---

Description

'cvToBsv()' converts a log-normal coefficient of variation (expressed as a percent) to the variance on the log scale, ' $\omega^2 = \log((CV/100)^2 + 1)$ '. 'bsvToCv()' is the inverse transform.

Usage

```
cvToBsv(cv)

bsvToCv(bsv)
```

Arguments

- cv The coefficient of variation, expressed as a percent.
- bsv The between-subject variability on the variance (ω^2) scale.

Value

The between-subject variability on the variance (ω^2) scale.
The coefficient of variation on the percent scale.

Functions

- `bsvToCv()`: Convert log-scale between-subject variance back to a percent coefficient of variation.

defaultCombine	<i>Default combine strings</i>
----------------	--------------------------------

Description

Default combine strings

Usage

```
defaultCombine(...)
```

```
snakeCombine(...)
```

```
camelCombine(...)
```

```
dotCombine(...)
```

```
blankCombine(...)
```

Arguments

... uses default to combine strings

Value

combined strings

Functions

- `snakeCombine()`: use `snake_case` to combine 2 strings
- `camelCombine()`: use `camelCase` to combine strings
- `dotCombine()`: use the default method for combining two strings
- `blankCombine()`: combine using a blank separator

Author(s)

Matthew L. Fidler

Examples

```
# default combine

defaultCombine("f", "depot")

defaultCombine(list(c("a", "funny", "c")))

defaultCombine(c("a", "funny", "c"))

# snake combine

snakeCombine("f", "depot")

snakeCombine(list(c("a", "funny", "c")))

snakeCombine(c("a", "funny", "c"))

# dot combine

dotCombine("f", "depot")

dotCombine(list(c("a", "funny", "c")))

dotCombine(c("a", "funny", "c"))

# blank combine

blankCombine("f", "depot")

blankCombine(list(c("a", "funny", "c")))

blankCombine(c("a", "funny", "c"))
```

fakeCc

Fake blank Cc for creating PD only models

Description

Fake blank Cc for creating PD only models

Usage

```
fakeCc(fun, ..., cc = "Cc")
```

Arguments

fun	function to that requires Cc
...	arguments sent PD function
cc	character name of the concentration in the central compartment that will be faked to allow models that require Cc to change to models with Cc as a covariate

Value

Model where Cc is a covariate

Author(s)

Matthew L. Fidler

Examples

```
fakeCc(addDirectLin) |> convertEmaxHill()
```

Mann2022Equilibrate *Pre-equilibrate the Mann 2022 respiratory-physiology states*

Description

The Mann 2022 / FDA delaymymod.c physiology layer is sensitive to initial conditions because the chemoreflex feedback sees a 48.6 mm Hg error in ‘p_b_co2 - p_b_co2_0’ at t = 0 when the FDA delaystates.R nominal values (‘palv_co2 = 40.28’, ‘cb_co2 = 0.645’) are used verbatim. The transient over-ventilation that follows leaves the system in a state that subsequent opioid doses cannot push into sustained PaO2 < 15 mm Hg, severely under-estimating cardiac arrest incidence.

Usage

```
Mann2022Equilibrate(model, params = NULL, duration_min = 90)
```

Arguments

model	An ‘rxode2’ model containing at least the Mann 2022 respiratory physiology states (‘palv_co2’, ‘palv_o2’, ‘cb_co2’, ‘cb_o2’, ‘ct_co2’, ‘ct_o2’, ‘yco2’, ‘yo2’, ‘dp_state’, ‘dc_state’, ‘alpha_h’). Works with the standalone ‘Mann_2022_respiratory_physiology’ model, with composed chains (PK + binding + physiology), or with any user-extended variant.
params	Named vector of model parameters. For a composed chain, supply the full PK + binding + physiology parameter set (the opioid PK states stay at zero throughout the pre-equilibration because no dose is given).
duration_min	Pre-equilibration duration in minutes. The default ‘90’ is sufficient for the Mann 2022 physiology to settle to four-decimal precision; the slowest state is ‘alpha_h’ with a 5-minute time constant.

Details

The FDA ‘simulateToGetOD_IM.R’ reference script handles this by running a no-drug pre-simulation via ‘fundede’ per subject and using the FINAL state as the dose-time initial state. This function does the equivalent: it runs the supplied ‘model’ for ‘duration_min’ minutes with zero opioid dose and zero antagonist input, then returns the final-time values of the standard Mann 2022 physiology states as a named list suitable for passing as ‘inits’ to ‘rxode2::rxSolve’.

The equilibrium does NOT depend on opioid PK parameters (no drug is present during pre-equilibration). The standalone ‘Mann_2022_respiratory_physiology’ model and the Laffont 2025 vignette inline chain ship with the FDA nominal initial conditions as ‘ini()’ defaults (matching ‘delaystates.R’), so any code that uses these models for downstream overdose simulation MUST call this function and pass the result via the ‘inits’ argument of ‘rxode2::rxSolve’; without it the FDA-published cardiac-arrest rates cannot be reproduced.

Value

Named list of physiology equilibrium state values, suitable for passing as the ‘inits’ argument to ‘rxode2::rxSolve’ on subsequent dose-bearing simulations.

modeldb	<i>Model library for nlmixr2</i>
---------	----------------------------------

Description

This is a data frame of the available models in nlmixr2lib, it is generated with the package. Custom modeldb may be used.

Usage

modeldb

Format

- A data frame with 4014 rows and 12 columns
- name** Model name that can be used to extract the model from the model library
 - description** Model description in free form text; in model itself
 - parameters** A comma separated string listing either the parameter in the model defined by population/individual effects or a population effect parameter
 - DV** The definition of the dependent variable(s)
 - linCmt** Logical flag indicating if solved models are used (TRUE) or not (FALSE)
 - algebraic** Logical flag indicating if the model is purely algebraic: TRUE no linCmt() and no ODEs; FALSE otherwise
 - dosing** A comma separated string of identified dosing compartments
 - depends** A comma separated string of objects the model depends on

- vignette** Basename of the vignette associated with this model (without path or extension); NA if none
- label** Human-readable navbar label derived from the filename: 'Drug (Author Year)' for the canonical '<Author>_<Year>_<drug>' form, 'DDMoRe: <drug>' for the parameterless NA_NA_<drug> ddmores entries, otherwise the basename with underscores replaced by spaces
- category** Coarse-grained model bucket derived from the filename path: 'specificDrugs', 'ddmore', or 'other' (anything not under those two directories)
- filename** Filename of the model. By default these are installed in the model library and read on demand

modellib

Get the model from the model library

Description

This function gets a model from the available model library

Usage

```
modellib(name = NULL, eta = NULL, reserr = NULL)
```

Arguments

- | | |
|--------|--|
| name | character with the name of the model to load (if NULL, lists all available base models) |
| eta | vector with the parameters to add random effects (sometimes referred to as inter-individual variability, IIV) on |
| reserr | The type or types of residual error (currently "addSd", "propSd", and "lnormSd" are accepted) |

Details

This is a very first draft just to look at the proof of concept

Value

The function returns a function the model code (or NULL if the model = NULL)

Examples

```
modellib(name = "PK_1cmt")
modellib(name = "PK_1cmt", eta = c("ka", "vc"), reserr = "addSd")
modellib(name = "PK_1cmt", reserr = "addSd")
```

ncaComparisonTable	<i>Build a side-by-side NCA comparison table (simulated vs. reference)</i>
--------------------	--

Description

Combines model-predicted NCA results with a reference set (e.g. paper values, prior estimates, alternative model runs, observed data) into a single tidy comparison frame ready for ‘knitr::kable()’ or interactive review. Each row is one NCA parameter $\times$ optional grouping level; columns show the reference value, simulated value, and percent difference. Rows whose discrepancy exceeds ‘tolerance_pct’ are flagged with a trailing asterisk and the function attaches a footnote string to the result.

Usage

```
ncaComparisonTable(
  simulated,
  reference,
  by = NULL,
  params = NULL,
  tolerance_pct = 20,
  units = NULL,
  label_first_column = "NCA parameter"
)
```

Arguments

simulated	Model-predicted NCA. One of: a ‘PKNCAresults’ object, a ‘data.frame’ with columns ‘PPTTESTCD’ and ‘PPORRES’ plus any grouping columns named in ‘by’, or a wide ‘data.frame’ (one row per group, one column per PKNCA code).
reference	Reference NCA. Same shapes as ‘simulated’. A wide ‘data.frame’ is the typical form when transcribing values from a paper table.
by	Optional character vector of grouping column name(s) (e.g. “treatment”, ‘c(“cohort”, “weight_band”)’). When ‘NULL’, the inputs are treated as a single ungrouped comparison.
params	Optional character vector of PKNCA codes restricting which parameters appear in the output. Defaults to the intersection of codes present in both inputs.
tolerance_pct	Numeric scalar; rows with ‘ ’ attaches a footnote string via the “footnote” attribute on the result. Pass ‘Inf’ to disable flagging.
units	Optional named character vector keyed by PKNCA code, passed through to [ncaParamLabel()].
label_first_column	Header for the parameter column. Defaults to “NCA parameter”.

Details

Designed for any nlmixr2 user comparing predicted NCA to a reference – not solely a vignette utility. Accepts a ‘PKNCAresults’ object, the ‘\$result’ data.frame, or a tidy long frame with columns ‘PTESTCD’ and ‘PPORRES’ for ‘simulated’; the same shapes plus a wide ‘data.frame’ (one row per group, one column per PKNCA parameter code) for ‘reference’. Multiple rows per group in ‘simulated’ (the typical per-subject case from [PKNCA::pk.nca()]) are aggregated to the group level using [stats::median()] before joining; pass a pre-aggregated frame if a different summary statistic is required.

Value

A ‘data.frame’ whose first column is the friendly parameter label (under ‘label_first_column’); when ‘by’ is non-NULL, the grouping column(s) follow; the final three columns are ‘Reference’, ‘Simulated’, and ‘parameter order then by group’. Carries a “footnote” attribute when any row exceeds the tolerance.

Author(s)

Bill Denney

Examples

```
simulated <- data.frame(
  treatment = rep(c("50 mg", "100 mg"), each = 3),
  PTESTCD = rep(c("cmax", "tmax", "auclast"), 2),
  PPORRES = c(15.2, 2.0, 96.0, 29.1, 2.1, 191.0)
)
reference <- data.frame(
  treatment = c("50 mg", "100 mg"),
  cmax = c(14.8, 28.5),
  tmax = c(2.0, 2.1),
  auclast = c(95.0, 190.0)
)
tbl <- ncaComparisonTable(
  simulated, reference,
  by = "treatment",
  units = c(cmax = "ug/mL", auclast = "ug*h/mL", tmax = "h")
)
tbl
attr(tbl, "footnote")
```

ncaParamLabel

Map PKNCA parameter codes to friendly human-readable labels

Description

Vectorized recoder that maps the PKNCA parameter codes (‘PTESTCD’ values such as “cmax”, “aucinf.obs”, “half.life”) to the friendly pharmacometric labels readers expect in tables and prose (“Cmax”, “AUC0-Inf (obs)”, “t1/2”). Codes that are not in the mapping table are returned unchanged with a warning, so unfamiliar parameters never get silently dropped.

Usage

```
ncaParamLabel(code, units = NULL)
```

Arguments

code	Character vector of PKNCA parameter codes. ‘NA’ values are passed through unchanged.
units	Optional named character vector keyed by PKNCA code. For each code with a matching entry, the friendly label is suffixed with the unit string in parentheses (e.g. “Cmax (ng/mL)”).

Value

Character vector the same length as ‘code’.

Author(s)

Bill Denney

Examples

```
ncaParamLabel(c("cmax", "tmax", "auclast", "aucinf.obs", "half.life"))
ncaParamLabel(
  c("cmax", "auclast"),
  units = c(cmax = "ng/mL", auclast = "ng*h/mL")
)
```

pkTrans

Change the transformation type for PK models

Description

Change the transformation type for PK models

Usage

```
pkTrans(
  ui,
  type = c("k", "k21", "vss", "aob", "alpha"),
  k13 = "k13",
  k31 = "k31",
  k12 = "k12",
  k21 = "k21",
  kel = "kel",
  vc = "vc",
  cl = "cl",
  vp = "vp",
```

```

q = "q",
vp2 = "vp2",
q2 = "q2",
vss = "vss",
aob = "aob",
alpha = "alpha",
beta = "beta",
gam = "gam",
A = "A",
B = "B",
C = "C",
s = "s",
p = "p",
tmp = "tmp",
beforeCmt = c("depot", "central")
)

```

Arguments

ui	A model in terms of Clearance
type	the type of PK transformation to make: - "k": Change to rate constants (kel, k12, k21, k13, k31) - "vss": Change to volume of distribution at steady state (cl, vc, q, vss) - "aob": Change to A/B ratio (aob, alpha, beta, vc) - "k21": Change to k21 constant (k21, alpha, beta, vc) or (k21, k31, alpha, beta, gam, vc) - "alpha": Change to macro constants (alpha, beta, gam, A, B, C, vc)
k13	name of rate constant from central to periph2
k31	name of rate constant from periph2 to central
k12	name of rate constant from central to periph1
k21	name of rate constant from periph1 to central
kel	name of elimination rate constant
vc	name of central compartment volume
cl	name of clearance
vp	name of volume of periph1
q	name of intercompartmental clearance between central and periph1
vp2	name of volume of periph2
q2	name of intercompartmental clearance between central and periph2
vss	name of volume of distribution at steady state
aob	A/B ratio
alpha	macro constant name for first exponential decay term
beta	macro constant name for second exponential decay term
gam	macro constant name for third exponential decay term

A	macro coefficient for the first exponential decay term (corresponds with alpha)
B	macro coefficient for the second exponential decay term (corresponds with beta)
C	macro coefficient for the third exponential decay term (corresponds with gam)
s	sum constant name for the k12 three compartment
p	product constant name for the k12 three compartment
tmp	name of temporary variable for the three compartment with 'A', 'B', 'C', 'alpha', 'beta' and 'gam'.
beforeCmt	if the model is compartmental you can specify the preferred names where the estimates and extra lines are added before

Value

ui with no PK parameters estimated

Author(s)

Matthew L. Fidler

Examples

```
# Three compartment model translations

readModelDb("PK_3cmt_des") |>
  pkTrans("k")

readModelDb("PK_3cmt_des") |>
  pkTrans("k21")

readModelDb("PK_3cmt_des") |>
  pkTrans("alpha")

# The most types of transformations are
# available for 2 compartment models

readModelDb("PK_2cmt_des") |>
  pkTrans("k")

readModelDb("PK_2cmt_des") |>
  pkTrans("vss")

readModelDb("PK_2cmt_des") |>
  pkTrans("aob")

readModelDb("PK_2cmt_des") |>
  pkTrans("k21")

readModelDb("PK_2cmt_des") |>
  pkTrans("alpha")

# One compartment transformations are also available:
```

```
readModelDb("PK_1cmt_des") |>
  pkTrans("k")

readModelDb("PK_1cmt_des") |>
  pkTrans("alpha")

# also works without depot:

readModelDb("PK_3cmt_des") |>
  removeDepot() |>
  pkTrans("k")
```

Poels_2025_elranatamab_qsp_cauc_multiplier

Eq 37 cumulative-cytokine-exposure multiplier

Description

Reciprocal of the Poels 2025 Supplementary Eq 37 divisor, $5 * (1 - (n + 1)^2 / (1.3^2 + (n + 1)^2))$, evaluated at dose index 'n'. Exported so the reset applied by [Poels_2025_elranatamab_qsp_events()] can be inspected and tested directly.

Usage

```
Poels_2025_elranatamab_qsp_cauc_multiplier(n)
```

Arguments

n Integer vector of dose indices (1 for the first dose).

Value

Numeric vector of multipliers applied to the 'cauc' state.

Examples

```
Poels_2025_elranatamab_qsp_cauc_multiplier(1:5)
```

Poels_2025_elranatamab_qsp_events

Build an event table for the Poels 2025 elranatamab QSP model

Description

‘Poels_2025_elranatamab_qsp’ reproduces the dose-to-dose attenuation of cytokine release described by Poels et al. (2025) Supplementary Eq 37, which rescales the cumulative cytokine exposure state ‘cauc’ at the start of every dosing interval:

Usage

```
Poels_2025_elranatamab_qsp_events(
  dose_time,
  dose_mg,
  obs_time,
  mw = 148500,
  id = 1L,
  cytokine_reset = TRUE,
  obs_cmt = "central"
)
```

Arguments

dose_time	Numeric vector of dose times, in hours.
dose_mg	Numeric vector of subcutaneous doses, in mg. Either length 1 (recycled) or the same length as ‘dose_time’.
obs_time	Numeric vector of observation times, in hours.
mw	Elranatamab molecular weight in g/mol. Defaults to 148500, the "approximately 148.5 kDa" of the ELREXFIO US prescribing information (Description section).
id	Subject identifier written to the ‘id’ column.
cytokine_reset	Logical; when ‘TRUE’ (the default) the Eq 37 reset records are included. Set to ‘FALSE’ to simulate the model without the dose-to-dose attenuation of cytokine release.
obs_cmt	Compartment used for observation records. Must be an ODE state of the model; every algebraic observable (‘Cc’, ‘mProtein’, ‘tumorBurden’, ...) is returned as a column at those rows regardless of which state is named.

Details

$$C_{auc,N}(0) = \frac{C_{auc,N-1}(\tau)}{5 \left(1 - \frac{(N+1)^2}{1.3^2 + (N+1)^2} \right)}$$

where N is the number of doses given. That is a discrete state reset, which cannot be written inside an rxode2 ODE. This helper therefore emits it as an ‘evid = 6’ (multiply) record on ‘cauc’ at each dose time, carrying the reciprocal of the Eq 37 divisor as its ‘amt’, alongside the subcutaneous dose records and the requested observation records.

Doses are supplied in mg and converted to pmol using the elranatamab molecular weight, because the model carries ‘depot’ as a pmol amount (Poels 2025 Supplementary Eq 1 divides the absorption flux by ‘Vc’).

Value

A ‘data.frame’ with columns ‘id’, ‘time’, ‘amt’, ‘evid’ and ‘cmt’, ordered by time, suitable for [rxode2::rxSolve()].

References

Poels KE, Elmeliegy M, Hibma J, Wang D, Musante CJ, Shtylla B. Leveraging quantitative systems pharmacology modeling for elranatamab regimen optimization in relapsed or refractory multiple myeloma. npj Syst Biol Appl. 2025;11:102. doi:10.1038/s4154002500585z

Examples

```
# MagnetisMM-3 Cohort A: 12/32/76 mg on C1D1/C1D4/C1D8, then 76 mg weekly
dose_time <- c(0, 72, 168, seq(336, by = 168, length.out = 4))
dose_mg <- c(12, 32, rep(76, 5))
ev <- Poels_2025_elranatamab_qsp_events(
  dose_time = dose_time,
  dose_mg = dose_mg,
  obs_time = seq(0, 1344, by = 24)
)
head(ev)
```

readModelDb

Read a model from the nlmixr2 model database

Description

Read a model from the nlmixr2 model database

Usage

```
readModelDb(name)
```

Arguments

name The name of the model (must be one of modeldb\$name)

Value

The model as a function

Examples

```
readModelDb("PK_1cmt")
```

 removeDepot

To convert from first order oral absorption to IV/Intravenous

Description

Any compartment property lines for the depot (`f()`, `lag()`, `dur()`, `rate()` or an initial condition) are removed with its differential equation; the variables that defined them are left in place (use `[removeLinesAndInis()]` to drop them as well).

Usage

```
removeDepot(ui, central = "central", depot = "depot", ka = "ka", model)
```

Arguments

<code>ui</code>	The model as a function (or something convertible to an <code>rxUi</code> object)
<code>central</code>	central compartment name
<code>depot</code>	depot compartment name
<code>ka</code>	absorption rate parameter name
<code>model</code>	Deprecated alias for <code>ui</code> . Supplying <code>model</code> instead of <code>ui</code> still works but emits a deprecation warning.

Value

Returns a model with the depot from a first order absorption model removed

Examples

```
readModelDb("PK_1cmt_des") |> removeDepot()
```

removeLinesAndInis	<i>Removes lines and inis from a model</i>
--------------------	--

Description

Removes lines and inis from a model

Usage

```
removeLinesAndInis(ui, vars)
```

Arguments

ui	A rxode2 model
vars	A character vector of variables to remove

Value

model with rxode2 lines and any estimates associate with lines removed

Author(s)

Matthew L. Fidler

Examples

```
readModelDb("PK_3cmt_des") |> removeLinesAndInis(c("kel", "k12", "k21"))
```

removePeriph	<i>Remove a peripheral compartment from a model</i>
--------------	---

Description

Removes ‘peripheral2’ (or ‘peripheral1’ when it is the only one), rewiring the central compartment and dropping the associated parameters. ‘peripheral1’ cannot be removed while ‘peripheral2’ is still present.

Usage

```
removePeriph(ui, n = NULL, central = "central", model)
```

Arguments

ui	The model as a function (or something convertible to an rxUi object)
n	which peripheral to remove: 1 or 2; defaults to the highest-numbered one present
central	central compartment name
model	Deprecated alias for ui. Supplying model instead of ui still works but emits a deprecation warning.

Value

a model with the peripheral compartment removed

Author(s)

Matthew L. Fidler

See Also

Other distribution: [addPeriph\(\)](#)

Examples

```
readModelDb("PK_2cmt_des") |> removePeriph()
```

removeTransit	<i>To remove transit compartments from the model</i>
---------------	--

Description

To remove transit compartments from the model

Usage

```
removeTransit(  
  ui,  
  ntransit,  
  central = "central",  
  depot = "depot",  
  transit = "transit",  
  ktr = "ktr",  
  ka = "ka",  
  model  
)
```

Arguments

<code>ui</code>	The model as a function
<code>ntransit</code>	the number of transit compartments to be added
<code>central</code>	central compartment name
<code>depot</code>	depot compartment name
<code>transit</code>	The number of transit compartments to remove
<code>ktr</code>	the parameter name for the transit compartment rate
<code>ka</code>	absorption rate parameter name
<code>model</code>	Deprecated alias for <code>ui</code> . Supplying <code>model</code> instead of <code>ui</code> still works but emits a deprecation warning.

Value

rxode2 model with transit compartment removed

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addTransit\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [convertAbsSequential\(\)](#), [removeZeroOrderAbs\(\)](#)

Examples

```
# In this example the transit is added and then a few are removed

readModelDb("PK_1cmt_des") |> addTransit(4) |> removeTransit(3)

readModelDb("PK_1cmt_des") |> addTransit(4) |> removeTransit()
```

<code>removeZeroOrderAbs</code>	<i>Remove zero-order absorption from a model</i>
---------------------------------	--

Description

Removes the modeled duration on the central compartment, leaving the dose as an intravenous bolus input. The duration variable is taken from the right hand side of the `dur(central)` line: when it is defined by an assignment in the model (the `tk0 <- exp(1tk0)` added by `[addZeroOrderAbs()]` or the `durCentral` added by `[addDur()]`) the variable and its initial estimate are dropped too; a bare estimated parameter that is not used elsewhere in the model is dropped from the initial estimates. First-order absorption can be restored with `[addDepot()]`.

Usage

```
removeZeroOrderAbs(ui, central = "central")
```

Arguments

`ui` The model as a function (or something convertible to an rxUi object)
`central` central compartment name

Value

a model where the zero-order absorption is removed

Author(s)

Matthew L. Fidler

See Also

Other absorption: [addSecondAbsorption\(\)](#), [addTransit\(\)](#), [addWeibullAbs\(\)](#), [addZeroOrderAbs\(\)](#), [convertAbsForceLongerDelay\(\)](#), [convertAbsSequential\(\)](#), [removeTransit\(\)](#)

Examples

```
readModelDb("PK_1cmt_des") |> addZeroOrderAbs() |> removeZeroOrderAbs()
```

searchReplace

Search within a model to replace part of the model

Description

Search within a model to replace part of the model

Usage

```
searchReplace(object, find, replace)
```

```
searchReplaceHelper(object, find, replace)
```

Arguments

`object` function specifying the nlmixr2 model
`find, replace` Character scalars of parts of the model to replace

Value

object with find replaced with replace

Functions

- `searchReplaceHelper()`: A helper function for `searchReplace` (not intended for users to use directly)

setCombineType	<i>Change the default combine type for the package</i>
----------------	--

Description

Change the default combine type for the package

Usage

```
setCombineType(combineType = c("snake", "camel", "dot", "blank"))
```

Arguments

combineType	this is the default combine type: - "default": default combine - "snake": snake_case combine - "camel": camelCase combine - "dot": dot combine (i.e. "a.b") - "blank": no separator (i.e. "ab")
-------------	---

Author(s)

Matthew L. Fidler

Examples

```
# Change to the popular snake_case
setCombineType("snake")
defaultCombine("a", "b")

# Change back to nlmixr2/rxode2 default camelCase

setCombineType("camel")
defaultCombine("a", "b")

# This is used to change the naming convention for parameters
# produced by this package
```

simMarkov	<i>Convert nlmixr2 simulations to Markov states</i>
-----------	---

Description

Simulations are performed per sim.id (if applicable) and per subject id within the simulation. The 'sim.id' column must be named exactly that, and the 'id' column is detected case-insensitively.

Usage

```
simMarkov(  
  ui,  
  initialState,  
  states,  
  colPrev = "previous",  
  colCur = "current",  
  ...  
)
```

Arguments

- ui a model fit or simulation object to simulate from or a data.frame containing the transition probability columns
- initialState The initial Markov state
- states All states in the model as a character vector
- colPrev, colCur Column names in the dataset for the previous and current states (may be any R object that can be coerced to a character string, ordered objects are usually preferred)
- ... Passed to methods

Details

See the Markov vignette (vignette("markov", package = "nlmixr2lib")) for an example.

Value

A data.frame with the original data along with the new state columns ('colPrev' and 'colCur') added

See Also

Other Markov models: [createMarkovModel\(\)](#), [createMarkovModelDataset\(\)](#), [createMarkovModelFromSingleState\(\)](#), [createMarkovTransitionMatrix\(\)](#), [simMarkovId\(\)](#)

simMarkovId	<i>Simulate a Markov model broken down by ID</i>
-------------	--

Description

Simulate a Markov model broken down by ID

Usage

```
simMarkovId(data, initialState, prCols)
```

Arguments

data	A data.frame for the individual with columns for each of the state transition probabilities
initialState	The initial Markov state
prCols	A named list of probability columns. List names are the previous state, and list values are a character vector of probability columns.

Value

A data.frame with two columns named "prev" and "cur"

See Also

Other Markov models: [createMarkovModel\(\)](#), [createMarkovModelDataset\(\)](#), [createMarkovModelFromSingleState\(\)](#), [createMarkovTransitionMatrix\(\)](#), [simMarkov\(\)](#)

updateOncologyXenograftSimeoni2004

Update an oncology xenograft model based on Simeoni 2004

Description

Update an oncology xenograft model based on Simeoni 2004

Usage

```
updateOncologyXenograftSimeoni2004(
  object,
  ncmt,
  damagedCmtName = "damaged_cells",
  drugEffectName = "drugEffectCyclingCells",
  undamagedCmtName = "cycling_cells",
  tumorVolName = "tumor_vol",
  transitRateName = "damageTransit"
)
```

Arguments

object	Fitted object or function specifying the model.
ncmt	The desired number of damaged cell compartments
damagedCmtName, undamagedCmtName, tumorVolName	character string names for the compartments for damaged cells, undamaged cells, and the calculated tumor volume (the sum of undamaged and damaged cells)
drugEffectName, transitRateName	character string names of the drug effect and transit rate (as used in the model block)

Value

An updated model with the new number of compartments

Examples

```
library(rxode2)
readModelDb("oncology_xenograft_simeoni_2004") |>
  updateOncologyXenograftSimeoni2004(ncmt = 5)
```

Index

- * **Indirect response**
 - addIndirect, [13](#)
 - addIndirectLin, [14](#)
 - convertKinR0, [31](#)
- * **Internal**
 - createMarkovModelFromSingleState, [37](#)
 - searchReplace, [56](#)
 - simMarkovId, [58](#)
- * **Markov models**
 - createMarkovModel, [35](#)
 - createMarkovModelDataset, [35](#)
 - createMarkovModelFromSingleState, [37](#)
 - createMarkovTransitionMatrix, [37](#)
 - simMarkov, [57](#)
 - simMarkovId, [58](#)
- * **PD**
 - addBaseline1exp, [3](#)
 - addBaselineConst, [4](#)
 - addBaselineExp, [5](#)
 - addBaselineLin, [6](#)
 - addDirectLin, [9](#)
 - convertEmax, [29](#)
 - convertLogLin, [32](#)
 - convertQuad, [34](#)
- * **absorption**
 - addSecondAbsorption, [19](#)
 - addTransit, [20](#)
 - addWeibullAbs, [21](#)
 - addZeroOrderAbs, [23](#)
 - convertAbsForceLongerDelay, [27](#)
 - convertAbsSequential, [28](#)
 - removeTransit, [54](#)
 - removeZeroOrderAbs, [55](#)
- * **datasets**
 - modeldb, [42](#)
- * **distribution**
 - addPeriph, [17](#)
 - removePeriph, [53](#)
 - addBaseline1exp, [3](#)
 - addBaseline1exp(), [4–6](#), [10](#), [29](#), [32](#), [34](#)
 - addBaselineConst, [4](#)
 - addBaselineConst(), [3](#), [5](#), [6](#), [10](#), [29](#), [32](#), [34](#)
 - addBaselineExp, [5](#)
 - addBaselineExp(), [3](#), [4](#), [6](#), [10](#), [29](#), [32](#), [34](#)
 - addBaselineLin, [6](#)
 - addBaselineLin(), [3–5](#), [10](#), [29](#), [32](#), [34](#)
 - addBioavailability (addCmtProp), [7](#)
 - addCmtProp, [7](#)
 - addDepot, [8](#)
 - addDirectLin, [9](#)
 - addDirectLin(), [3–6](#), [29](#), [32](#), [34](#)
 - addDirToModelDb, [10](#)
 - addDur (addCmtProp), [7](#)
 - addEffectCmtLin, [11](#)
 - addEta, [12](#)
 - addFileToModelDb (addDirToModelDb), [10](#)
 - addIndirect, [13](#)
 - addIndirect(), [15](#), [31](#)
 - addIndirectLin, [14](#)
 - addIndirectLin(), [14](#), [31](#)
 - addIni (addCmtProp), [7](#)
 - addLag (addCmtProp), [7](#)
 - addLogEstimates, [16](#)
 - addPeriph, [17](#)
 - addPeriph(), [54](#)
 - addRate (addCmtProp), [7](#)
 - addResErr, [18](#)
 - addSecondAbsorption, [19](#)
 - addSecondAbsorption(), [21](#), [22](#), [24](#), [27](#), [28](#), [55](#), [56](#)
 - addTransit, [20](#)
 - addTransit(), [20](#), [22](#), [24](#), [27](#), [28](#), [55](#), [56](#)
 - addWeibullAbs, [21](#)
 - addWeibullAbs(), [20](#), [21](#), [24](#), [27](#), [28](#), [55](#), [56](#)
 - addZeroOrderAbs, [23](#)
 - addZeroOrderAbs(), [20–22](#), [27](#), [28](#), [55](#), [56](#)

- blankCombine (defaultCombine), 39
- bsvToCv (cvToBsv), 38
- camelCombine (defaultCombine), 39
- checkModelConventions, 24
- checkNamingRegisters, 25
- combinePaste2, 26
- convertAbsForceLongerDelay, 27
- convertAbsForceLongerDelay(), 20–22, 24, 28, 55, 56
- convertAbsSequential, 28
- convertAbsSequential(), 20–22, 24, 27, 55, 56
- convertEmax, 29
- convertEmax(), 3–6, 10, 32, 34
- convertEmaxHill, 30
- convertKinR0, 31
- convertKinR0(), 14, 15
- convertLogLin, 32
- convertLogLin(), 3–6, 10, 29, 34
- convertMM, 33
- convertQuad, 34
- convertQuad(), 3–6, 10, 29, 32
- createMarkovModel, 35
- createMarkovModel(), 36–38, 58, 59
- createMarkovModelDataset, 35
- createMarkovModelDataset(), 35, 37, 38, 58, 59
- createMarkovModelFromSingleState, 37
- createMarkovModelFromSingleState(), 35, 36, 38, 58, 59
- createMarkovTransitionMatrix, 37
- createMarkovTransitionMatrix(), 35–37, 58, 59
- cvToBsv, 38
- defaultCombine, 39
- dotCombine (defaultCombine), 39
- fakeCc, 40
- Mann2022Equilibrate, 41
- modeldb, 42
- modllib, 43
- ncaComparisonTable, 44
- ncaParamLabel, 45
- pkTrans, 46
- Poels_2025_elranatamab_qsp_cauc_multiplier, 49
- Poels_2025_elranatamab_qsp_events, 50
- readModelDb, 51
- removeDepot, 52
- removeLinesAndInis, 53
- removePeriph, 53
- removePeriph(), 17
- removeTransit, 54
- removeTransit(), 20–22, 24, 27, 28, 56
- removeZeroOrderAbs, 55
- removeZeroOrderAbs(), 20–22, 24, 27, 28, 55
- searchReplace, 56
- searchReplaceHelper (searchReplace), 56
- setCombineType, 57
- simMarkov, 57
- simMarkov(), 35–38, 59
- simMarkovId, 58
- simMarkovId(), 35–38, 58
- snakeCombine (defaultCombine), 39
- updateOncologyXenograftSimeoni2004, 59