

Package ‘nlmixr2plot’

September 23, 2026

Title Nonlinear Mixed Effects Models in Population PK/PD, Plot Functions

Version 5.2.0

Description Fit and compare nonlinear mixed-effects models in differential equations with flexible dosing information commonly seen in pharmacokinetics and pharmacodynamics (Almquist, Leander, and Jirstrand 2015 <[doi:10.1007/s10928-015-9409-1](https://doi.org/10.1007/s10928-015-9409-1)>). Differential equation solving is by compiled C code provided in the 'rxode2' package (Wang, Hallow, and James 2015 <[doi:10.1002/psp4.12052](https://doi.org/10.1002/psp4.12052)>). This package is for 'ggplot2' plotting methods for 'nlmixr2' objects.

License GPL (>= 3)

URL <https://github.com/nlmixr2/nlmixr2plot>,
<https://nlmixr2.github.io/nlmixr2plot/>

BugReports <https://github.com/nlmixr2/nlmixr2plot/issues/>

Depends R (>= 4.3.0)

Imports ggforce, ggplot2 (>= 3.4.0), ggtibble (>= 1.0.4), nlmixr2est, nlmixr2extra (>= 5.0.0), rxode2, stats, tidyr, utils, xgxr

Suggests DiagrammeR, knitr, rmarkdown, testthat (>= 3.0.0), data.table, dplyr, withr, vpc, tidyvpc, nlmixr2data

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Matthew Fidler [aut, cre] (ORCID: <<https://orcid.org/0000-0001-8538-6691>>),
Bill Denney [ctb] (ORCID: <<https://orcid.org/0000-0002-5759-428X>>),
Wenping Wang [aut],
Vipul Mann [aut]

Maintainer Matthew Fidler <matthew.fidler@gmail.com>

Repository CRAN

Date/Publication 2026-09-22 23:50:08 UTC

Contents

modelDiagram	2
modelGraph	4
plot.nlmixr2AugPred	6
plot.nlmixr2FitData	8
traceplot	9
vpcPlot	10
Index	14

modelDiagram	<i>Automatic model diagram</i>
--------------	--------------------------------

Description

Draws a compartment diagram of a model from its differential equations (see [modelGraph\(\)](#) for how the equations are interpreted).

Usage

```
modelDiagram(  
  object,  
  dosing = NULL,  
  data = NULL,  
  engine = getOption("nlmixr2plot.diagram.engine"),  
  labels = FALSE,  
  ...  
)  
  
## S3 method for class 'nlmixr2ModelGraph'  
plot(x, ..., engine = getOption("nlmixr2plot.diagram.engine"), labels = FALSE)  
  
## S3 method for class 'rxUi'  
plot(  
  x,  
  ...,  
  dosing = NULL,  
  data = NULL,  
  engine = getOption("nlmixr2plot.diagram.engine"),  
  labels = FALSE  
)  
  
## S3 method for class 'rxode2'  
plot(  
  x,  
  ...,  
  dosing = NULL,
```

```

    data = NULL,
    engine = getOption("nlmixr2plot.diagram.engine"),
    labels = FALSE
  )

```

Arguments

object	model to diagram (see modelGraph()) or a <code>nlmixr2ModelGraph</code> object.
dosing	optional character vector naming the dosing compartments. When <code>NULL</code> the dosing compartments are detected from the dosing records in <code>data</code> (a dataset without dose records has no dosing compartment); when there is no <code>data</code> , or it has no <code>evid/amt</code> columns, the first compartment (the default <code>rxode2</code> dosing compartment) is used.
data	optional dataset used to detect the dosing compartments (from the dosing records' <code>cmt</code>). For fitted models this defaults to the data the model was fit with.
engine	drawing engine: "DiagrammeR" (a Graphviz <code>htmlwidget</code> from the 'DiagrammeR' package), "ggplot2" (a <code>ggplot</code> object) or "dot" (the Graphviz DOT source as a character string, to customize or render elsewhere). The default is "DiagrammeR" when that package is installed and "ggplot2" otherwise; it may be changed with <code>options(nlmixr2plot.diagram.engine = ...)</code> .
labels	logical; when <code>TRUE</code> label the arrows with the model terms.
...	ignored.
x	a <code>nlmixr2ModelGraph</code> object, an <code>rxode2</code> user interface (<code>rxUi</code>) object or a compiled <code>rxode2</code> model

Details

The layout follows common pharmacometric conventions: dosing and absorption/transit compartments are above the compartment they feed; the central compartment is in the middle with the compartments it exchanges mass with (peripheral compartments) to its left; unidirectional transfer (e.g. to a metabolite) and eliminations go below; compartments that interact with the model without mass transfer (e.g. effect compartments or pharmacodynamic models) go to the right, with their own inputs above, outputs below and exchange compartments further right.

Mass transfer is drawn with solid arrows; interactions without mass transfer are dashed (with a "tee" arrow head for inhibition and a "dot" arrow head when the direction is undetermined with `DiagrammeR`; dotted for inhibition and dot-dashed when undetermined with `ggplot2`).

`plot()` of an `rxode2` user interface (`rxUi`) object, like `rxode2::rxode2(modelFunction)`, or of a compiled `rxode2` model draws its model diagram, so `plot(rxode2(model))` is the same as `modelDiagram(model)`. (A fitted `nlmixr2` model keeps its goodness-of-fit `plot()`; use `modelDiagram(fit)` for its diagram.)

Value

the diagram drawn by the requested engine.

Author(s)

Matthew L. Fidler

See Also

Other model diagrams: [modelGraph\(\)](#)

Examples

```
pk.turnover.emax <- function() {
  ini({
    tktr <- log(1)
    tka <- log(1)
    tcl <- log(0.1)
    tv <- log(10)
    poplogit <- 2
    tec50 <- log(0.5)
    tkout <- log(0.05)
    te0 <- log(100)
    prop.err <- 0.1
    pkadd.err <- 0.1
    pdadd.err <- 10
  })
  model({
    ktr <- exp(tktr)
    ka <- exp(tka)
    cl <- exp(tcl)
    v <- exp(tv)
    emax <- expit(poplogit)
    ec50 <- exp(tec50)
    kout <- exp(tkout)
    e0 <- exp(te0)
    DCP <- center / v
    PD <- 1 - emax * DCP / (ec50 + DCP)
    effect(0) <- e0
    kin <- e0 * kout
    d/dt(depot) <- -ktr * depot
    d/dt(gut) <- ktr * depot - ka * gut
    d/dt(center) <- ka * gut - cl / v * center
    d/dt(effect) <- kin * PD - kout * effect
    cp <- center / v
    cp ~ prop(prop.err) + add(pkadd.err)
    effect ~ add(pdadd.err)
  })
}
modelDiagram(pk.turnover.emax, engine = "ggplot2")
if (requireNamespace("DiagrammeR", quietly = TRUE)) {
  modelDiagram(pk.turnover.emax, engine = "DiagrammeR")
}
```

Description

The differential equations are parsed into additive terms. A term that is subtracted from one compartment and added (identically) to another is mass transfer between the two compartments. A remaining subtracted term that contains the compartment's own amount is an elimination (output); a remaining term that depends on other compartments is an interaction that does not transport mass (for example an effect compartment or a pharmacodynamic stimulation/inhibition); a remaining added term that does not depend on any other compartment is a (zero-order) input. Dependencies through intermediate variables (like $cp \leftarrow central/v$) are followed. A compartment that drives a transfer between two other compartments (like an enzyme) is drawn as an interaction with the destination. Production or loss driven only by another compartment (like $ke_0 * cp$ in an effect compartment) is represented by the interaction arrow alone, without a separate input/output arrow.

Usage

```
modelGraph(object, dosing = NULL, data = NULL)
```

Arguments

object	model to diagram: a model function, an rxode2 user interface (rxUi) object, a compiled rxode2 model or a fitted nlmixr2 object.
dosing	optional character vector naming the dosing compartments. When NULL the dosing compartments are detected from the dosing records in data (a dataset without dose records has no dosing compartment); when there is no data, or it has no evid/amt columns, the first compartment (the default rxode2 dosing compartment) is used.
data	optional dataset used to detect the dosing compartments (from the dosing records' cmt). For fitted models this defaults to the data the model was fit with.

Details

Some limitations on how equations must be written:

- Mass transfer is only detected when the same term (up to reordering of the factors of a product) is subtracted from the source and added to the destination, e.g. $d/dt(\text{depot}) \leftarrow -ka * \text{depot}$ and $d/dt(\text{central}) \leftarrow ka * \text{depot} - \dots$. Scaled transfer (like a stoichiometric or volume conversion in only one of the equations) is shown as an elimination plus an interaction.
- `linCmt()` models are converted to ODEs with `rxode2::linToOde()`, which requires a version of 'rxode2' that provides it.

Value

a `nlmixr2ModelGraph` object; a list with:

- `nodes`: data frame with the compartment name, its role ("dosing", "central", "peripheral", "transit", "metabolite", "effect" or "other"), whether it is dosing and the layout coordinates `x` and `y`, and an annotation with the compartment's dosing properties (`lag`, `F`, `rate`, `dur`; "" when there are none), which the diagrams show next to the compartment.

- edges: data frame with from, to (NA for inputs/eliminations), type ("transfer", "elimination", "input" or "interaction"), sign (1 when the term is added, -1 when it is subtracted; for interactions 1 is stimulation, -1 inhibition and 0 an effect whose direction cannot be determined from the equations, e.g. through ifelse() or a conditionally assigned variable), bidirectional (for transfers) and label (the model term(s)).

Author(s)

Matthew L. Fidler

See Also

Other model diagrams: [modelDiagram\(\)](#)

Examples

```
one.cmt <- function() {
  ini({
    tka <- 0.45
    tcl <- 1
    tv <- 3.45
    add.sd <- 0.7
  })
  model({
    ka <- exp(tka)
    cl <- exp(tcl)
    v <- exp(tv)
    d/dt(depot) <- -ka * depot
    d/dt(central) <- ka * depot - cl / v * central
    cp <- central / v
    cp ~ add(add.sd)
  })
}
modelGraph(one.cmt)
```

plot.nlmixr2AugPred *Plot a.nlmixr2 augPred object*

Description

Plot a.nlmixr2 augPred object

Usage

```
## S3 method for class '.nlmixr2AugPred'
plot(x, y, ..., log = "")
```

Arguments

x	augPred object
y	ignored, used to mach plot generic
...	Other arguments (ignored)
log	a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic and "xy" or "yx" if both axes are to be logarithmic (as in <code>graphics::plot.default()</code>). The default "" uses linear axes. Non-positive values cannot be shown on a log axis and are dropped.

Value

A `ggtribble::gglist` object (a list of `ggplot2` objects, one per page of individual plots)

Examples

```
library(nlmixr2est)
## The basic model consiss of an ini block that has initial estimates
one.compartment <- function() {
  ini({
    tka <- 0.45 # Log Ka
    tcl <- 1 # Log Cl
    tv <- 3.45 # Log V
    eta.ka ~ 0.6
    eta.cl ~ 0.3
    eta.v ~ 0.1
    add.sd <- 0.7
  })
  # and a model block with the error sspecification and model specification
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl + eta.cl)
    v <- exp(tv + eta.v)
    d/dt(depot) = -ka * depot
    d/dt(center) = ka * depot - cl / v * center
    cp = center / v
    cp ~ add(add.sd)
  })
}

## The fit is performed by the function nlmixr/nlmix2 specifying the model, data and estimate
fit <- nlmixr2est::nlmixr2(one.compartment, theo_sd, est="saem",
  saemControl(print=0, nBurn = 10, nEm = 20))

# augPred shows more points for the fit:

a <- nlmixr2est::augPred(fit)

# you can plot it with plot(augPred object)
plot(a)
```

```
# or with a log-scaled y axis
plot(a, log = "y")
```

plot.nlmixr2FitData *Plot a.nlmixr2 data object*

Description

Plot some standard goodness of fit plots for the `focei` fitted object. When the model has between-subject variability (BSV), the returned collection also includes a nested `"bsv"` element (inside each data/compartiment group) with QQ plots for each BSV parameter, BSV-BSV correlation plots (when more than one BSV parameter is present) and, when `covariate` is supplied, BSV-by-covariate plots.

Usage

```
## S3 method for class '.nlmixr2FitData'
plot(x, covariate = NULL, ...)
```

Arguments

<code>x</code>	a <code>focei</code> fit object
<code>covariate</code>	Optional character vector of covariate column names (from the model input data) to plot against each between-subject variability (BSV) parameter. Default <code>NULL</code> (no covariate plots). The first row per individual is used, so covariates are assumed time-invariant (a time-varying covariate is represented by its baseline value).
<code>...</code>	additional arguments (currently ignored)

Value

A named, nested `ggtribble::gglist` object (a list of `ggplot2` objects with easier plotting of all of them at the same time)

Author(s)

Wenping Wang & Matthew Fidler

Examples

```
library.nlmixr2est)
one.compartment <- function() {
  ini({
    tka <- 0.45
    tcl <- 1
```

```

      tv <- 3.45
      eta.ka ~ 0.6
      eta.cl ~ 0.3
      eta.v ~ 0.1
      add.sd <- 0.7
    })
    model({
      ka <- exp(tka + eta.ka)
      cl <- exp(tcl + eta.cl)
      v <- exp(tv + eta.v)
      d/dt(depot) = -ka * depot
      d/dt(center) = ka * depot - cl / v * center
      cp = center / v
      cp ~ add(add.sd)
    })
  }

## The fit is performed by the function nlmixr/nlmix2 specifying the model, data and estimate
fit <- nlmixr2(one.compartment, theo_sd, est="saem", saemControl(print=0, nBurn = 10, nEm = 20))

# This shows many goodness of fit plots
plot(fit)

```

traceplot

Produce trace-plot for fit if applicable

Description

Produce trace-plot for fit if applicable

Usage

```
traceplot(x, ...)
```

```
## S3 method for class 'nlmixr2FitCore'
traceplot(x, ...)
```

Arguments

x	fit object
...	other parameters

Value

Fit traceplot or nothing.

Author(s)

Rik Schoemaker, Wenping Wang & Matthew L. Fidler

Examples

```
library(nlmixr2est)
## The basic model consists of an ini block that has initial estimates
one.compartment <- function() {
  ini({
    tka <- 0.45 # Log Ka
    tc1 <- 1 # Log Cl
    tv <- 3.45 # Log V
    eta.ka ~ 0.6
    eta.cl ~ 0.3
    eta.v ~ 0.1
    add.sd <- 0.7
  })
  # and a model block with the error specification and model specification
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tc1 + eta.cl)
    v <- exp(tv + eta.v)
    d/dt(depot) = -ka * depot
    d/dt(center) = ka * depot - cl / v * center
    cp = center / v
    cp ~ add(add.sd)
  })
}

## The fit is performed by the function nlmixr/nlmix2 specifying the model, data and estimate
fit <- nlmixr2(one.compartment, theo_sd, est="saem",
              saemControl(print=0, nBurn = 10, nEm = 20))

# This shows the traceplot of the fit (useful for saem)
traceplot(fit)
```

vpcPlot

VPC based on ui model

Description

VPC based on ui model

Usage

```
vpcPlot(
  fit,
  data = NULL,
  n = 300,
  bins = "jenks",
```

```

    n_bins = "auto",
    bin_mid = "mean",
    show = NULL,
    stratify = NULL,
    pred_corr = FALSE,
    pred_corr_lower_bnd = 0,
    pi = c(0.05, 0.95),
    ci = c(0.05, 0.95),
    uloq = fit$dataUloq,
    lloq = fit$dataLloq,
    log_y = FALSE,
    log_y_min = 0.001,
    xlab = NULL,
    ylab = NULL,
    title = NULL,
    smooth = TRUE,
    vpc_theme = NULL,
    facet = "wrap",
    scales = "fixed",
    labeller = NULL,
    vpcdb = FALSE,
    verbose = FALSE,
    ...,
    seed = 1009,
    idv = "time",
    cens = FALSE,
    method = c("vpc", "tidyvpc")
)

vpcPlotTad(..., idv = "tad")

vpcCensTad(..., cens = TRUE, idv = "tad")

vpcCens(..., cens = TRUE, idv = "time")

```

Arguments

<code>fit</code>	nlmixr2 fit object, or a simulation from vpcSim() . A supplied simulation is used as-is (n is then ignored). For <code>pred_corr = TRUE</code> it must have been created with <code>vpcSim(..., pred = TRUE)</code> , and the observed data are pred-corrected by re-solving the population predictions of the simulation's fit with ..., so pass the same ... that was given to <code>vpcSim()</code> .
<code>data</code>	this is the data to use to augment the VPC fit. By default is the fitted data, (can be retrieved by getData), but it can be changed by specifying this argument.
<code>n</code>	Number of VPC simulations (ignored when <code>fit</code> is a <code>vpcSim()</code> simulation)
<code>bins</code>	either "density", "time", or "data", "none", or one of the approaches available in <code>classInterval()</code> such as "jenks" (default) or "pretty", or a numeric vector specifying the bin separators.

n_bins	when using the "auto" binning method, what number of bins to aim for
bin_mid	either "mean" for the mean of all timepoints (default) or "middle" to use the average of the bin boundaries.
show	what to show in VPC (obs_dv, obs_ci, pi, pi_as_area, pi_ci, obs_median, sim_median, sim_median_ci)
stratify	character vector of stratification variables. Only 1 or 2 stratification variables can be supplied.
pred_corr	perform prediction-correction?
pred_corr_lower_bnd	lower bound for the prediction-correction
pi	simulated prediction interval to plot. Default is c(0.05, 0.95),
ci	confidence interval to plot. Default is (0.05, 0.95)
uloq	Number or NULL indicating upper limit of quantification. Default is NULL.
lloq	Number or NULL indicating lower limit of quantification. Default is NULL.
log_y	Boolean indicting whether y-axis should be shown as logarithmic. Default is FALSE.
log_y_min	minimal value when using log_y argument. Default is 1e-3.
xlab	label for x axis
ylab	label for y axis
title	title
smooth	"smooth" the VPC (connect bin midpoints) or show bins as rectangular boxes. Default is TRUE.
vpc_theme	theme to be used in VPC. Expects list of class vpc_theme created with function vpc_theme()
facet	either "wrap", "columns", or "rows"
scales	Are scales shared across all facets (the default, "fixed"), or do they vary across rows ("free_x"), columns ("free_y"), or both rows and columns ("free")?
labeller	ggplot2 labeller function to be passed to underlying ggplot object
vpcdb	Boolean whether to return the underlying vpcdb rather than the plot
verbose	show debugging information (TRUE or FALSE)
...	Args sent to rxSolve
seed	an object specifying if and how the random number generator should be initialized
idv	Name of independent variable. For vpcPlot() and vpcCens() the default is "time" for vpcPlotTad() and vpcCensTad() this is "tad"
cens	is a boolean to show if this is a censoring plot or not. When cens=TRUE this is actually a censoring vpc plot (with vpcCens() and vpcCensTad()). When cens=FALSE this is traditional VPC plot (vpcPlot() and vpcPlotTad()).
method	the method to use for VPC plotting; can be "vpc" (uses the vpc package) or "tidyvpc" (uses the tidyvpc package). By default, "vpc" is used when the vpc package is available, otherwise "tidyvpc" is used.

Value

Simulated dataset (invisibly)

Author(s)

Matthew L. Fidler

Examples

```
one.cmt <- function() {  
  ini({  
    tka <- 0.45; label("Ka")  
    tcl <- log(c(0, 2.7, 100)); label("Cl")  
    tv <- 3.45; label("V")  
    eta.ka ~ 0.6  
    eta.cl ~ 0.3  
    eta.v ~ 0.1  
    add.sd <- 0.7; label("Additive residual error")  
  })  
  model({  
    ka <- exp(tka + eta.ka)  
    cl <- exp(tcl + eta.cl)  
    v <- exp(tv + eta.v)  
    linCmt() ~ add(add.sd)  
  })  
}  
  
fit <-  
  nlmixr2est::nlmixr(  
    one.cmt,  
    data = nlmixr2data::theo_sd,  
    est = "saem",  
    control = nlmixr2est::saemControl(print = 0, nBurn = 10, nEm = 20)  
  )  
  
vpcPlot(fit, n = 100)
```

Index

* **model diagrams**

modelDiagram, [2](#)

modelGraph, [4](#)

getData, [11](#)

graphics::plot.default(), [7](#)

modelDiagram, [2](#)

modelDiagram(), [6](#)

modelGraph, [4](#)

modelGraph(), [2–4](#)

plot.nlmixr2AugPred, [6](#)

plot.nlmixr2FitData, [8](#)

plot.nlmixr2ModelGraph(modelDiagram), [2](#)

plot.rxode2(modelDiagram), [2](#)

plot.rxUi(modelDiagram), [2](#)

rxSolve, [12](#)

traceplot, [9](#)

vpcCens(vpcPlot), [10](#)

vpcCensTad(vpcPlot), [10](#)

vpcPlot, [10](#)

vpcPlotTad(vpcPlot), [10](#)

vpcSim, [11](#)