Package ‘mapbayr’

May 26, 2022

Title MAP-Bayesian Estimation of PK Parameters

Version 0.7.3


License GPL-3

Encoding UTF-8

RoxygenNote 7.2.0

Imports dplyr, ggplot2, magrittr, mrgsolve (>= 1.0.0), optimx, progress, purrr, rlang, stringr, tibble, tidyr

URL https://github.com/FelicienLL/mapbayr

BugReports https://github.com/FelicienLL/mapbayr/issues

Suggests knitr, minqa, testthat (>= 3.0.0)

Depends R (>= 2.10)

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Felicien Le Louedec [aut, cre]

Kyle T Baron [ctb]

Maintainer Felicien Le Louedec <felicienlelouedec@proton.me>

Repository CRAN

Date/Publication 2022-05-26 13:30:02 UTC
R topics documented:

- adm_0_cmt
- adm_cmt
- as.data.frame.mapbayests
- augment
- augment.mapbayests
- check_mapbayr_model
- compute_ofv
- data_helpers
- est001
- exmodel_exdata
- get_x
- h
- hist.mapbayests
- mapbayest
- obs_cmt
- plot.mapbayests
- postprocess
- preprocess.ofv
- preprocess.optim
- print.mapbayests
- qparam
- use_posterior
- vs_nonmem

Index

| adm_0_cmt | Get zero-order infusion compartment from mrgsolve model |

Description

Get zero-order infusion compartment from mrgsolve model

Usage

adm_0_cmt(x)

Arguments

x model object

Value

vector of integer
Examples

# Both 1st and 0-order administration
cmodel <- exmodel(6, compile = FALSE)
adm_cmt(cmodel)

Description

Get administration compartment numbers from mrgsolve model

Usage

adm_cmt(x)

Arguments

x model object

Value

vector of integer

Examples

# Both 1st and 0-order administration
cmodel <- exmodel(6, compile = FALSE)
adm_cmt(cmodel)

as.data.frame.mapbayests

Return the mapbay_tab as a data.frame

Description

Return the mapbay_tab as a data.frame

Usage

## S3 method for class 'mapbayests'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)


Arguments

x  A mapbayests object.
row.names, optional, ...
    passed to as.data.frame

Value

a data.frame (the mapbay_tab from estimation)

augment

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>object to augment</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments</td>
</tr>
</tbody>
</table>

Value

an augmented object (depending on the object passed).

augment.mapbayests

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Compute full PK profile prediction from mapbayr estimates.</td>
</tr>
<tr>
<td>row.names</td>
<td>Optional, passed to as.data.frame</td>
</tr>
</tbody>
</table>

Description

Compute full PK profile prediction from mapbayr estimates.
Usage

```r
## S3 method for class 'mapbayests'
augment(
  x,
  data = NULL,
  start = NULL,
  end = NULL,
  delta = NULL,
  ci = FALSE,
  ci_width = 90,
  ci_method = "delta",
  ci_sims = 500,
  ...
)
```

Arguments

- **x**: A `mapbayests` object.
- **data**: dataset to pass to `mrgsolve` for simulation (default is dataset used for estimation)
- **start, end, delta**: start, end and delta of simulation time passed to `mrgsim()` (see details)
- **ci**: a logical. If TRUE, compute a confidence interval around the prediction (default is FALSE)
- **ci_width**: a number between 0 and 100, width of the confidence interval (default is "90" for a 90%CI)
- **ci_method**: method to compute the confidence interval. Can be "delta" (the default) to use the Delta approximation. Alternatively "simulations" for a more accurate approach, but also more time-consuming.
- **ci_sims**: number of replicates to simulate in order to derive the confidence interval (default is 500)
- **...**: additional arguments passed to `mrgsim()`

Details

This function is called in the background by `plot()` in order to simulate the full PK profile, and return a `mapbayests` object with an additional `aug_tab` data.frame inside. The latter is used with by the `plot` method. The time grid, for each PK profile (i.e. patient) is defaulted with the minimum time in the dataset for `start` and the maximum time in the dataset +20% for `end`. `delta` is a power of 10 (e.g. 0.1, 1, 10 etc...), automatically chosen to render visually appealing graphs with a reasonable computing time (about 200 time points). Additional arguments can be passed to `mrgsim()` through `...`. Note that `recsort` is set to 3 (see `mrgsolve` documentation for more details).

Value

A `mapbayests` object, augmented of an `aug_tab` data.frame.
Examples

#x is the result of `mapbayest()`.
#Default plot is returned by:
#plot(x)
#Argument passed to `plot()` are passed to `augment()` in the background:
#plot(x, end = 240, ci = TRUE)
#Save the augmented object if simulation time is long
# x2 <- augment(x, ci = TRUE, ci_method = "simulations", ci_sims = 10000) %>%
# plot(x2)

---

check_mapbayr_model  Check if model is valid for mapbayr

Description

Check if model is valid for mapbayr

Usage

check_mapbayr_model(x, check_compile = TRUE)

Arguments

x  model file
check_compile  check if model is compiled (used internally)

Value

TRUE value if check is passed, a vector of character with errors otherwise.

Examples

library(mapbayr)
library(mrgsolve)
check_mapbayr_model(house())
**compute_ofv**

Compute the objective function value

### Description

Compute the objective function value

### Usage

```r
compute_ofv(
  eta,
  qmod,
  sigma,
  omega_inv,
  all_cmt,
  log_transformation,
  idvaliddata,
  idDV,
  idcmt,
  ...
)
```

```r
do_compute_ofv(eta, argofv, ...)
```

### Arguments

- **eta**: a named vector/list of parameters
- **qmod, sigma, log_transformation, omega_inv, all_cmt**: generated by `preprocess.ofv.fix`
- **idvaliddata, idDV, idcmt**: generated by `preprocess.ofv.id`
- **...**: for compatibility (not used)
- **argofv**: above mentioned arguments as a list

### Details

This function is called iteratively by the optimization function. Arguments should not be passed directly, but generated by the pre-processing functions (see `preprocess.ofv`).

### Value

A single numeric value (the objective function value)
data_helpers

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helpers to build data set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>adm_lines(x, ...)</td>
</tr>
</tbody>
</table>

```r
## S3 method for class 'mrgmod'
adm_lines(x, ...)
```

<table>
<thead>
<tr>
<th>obs_lines(x, time, DV, mdv = 0, cmt = NULL, DVmet = NULL, ...)</th>
</tr>
</thead>
</table>

```r
## S3 method for class 'mrgmod'
ob_lines(x, time, DV, mdv = 0, cmt = NULL, DVmet = NULL, ...)
```

<table>
<thead>
<tr>
<th>add_covariates(x, covariates, ...)</th>
</tr>
</thead>
</table>

```r
## S3 method for class 'mrgmod'
add_covariates(x, covariates = list(), ...)
```

<table>
<thead>
<tr>
<th>see_data(x, ...)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>time, DV, mdv, cmt, DVmet</td>
</tr>
<tr>
<td>covariates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helpful functions build the data set. Instead of painfully build a data set and mind how to format the data, you can pass information about:</td>
</tr>
</tbody>
</table>

- administrations with adm_lines(),
- observations with obs_lines()
- covariates with add_covariates().

These functions are passed to a mrgmod object (mrgsolve model), and return a mrgmod object with a data set inside with the correct formatting (so-called NM-TRAN format), so that mrgsolve or mapbayr functions can be passed along within a pipe-friendly workflow.

These functions are meant to be used for one single patient at a time. Multiple ID is accepted, but the user is asked to check if the output is acceptable.
Value

a mrgmod object, with a dataset in the @args$data slot.

Examples

library(magrittr)
# First, import a model
mod <- exmodel(add_exdata = FALSE)

mod %>%
  adm_lines(amt = 10000, cmt = 1) %>%
  obs_lines(time = c(1.5, 4.4, 7.5, 24.6), DV = c(91.2904, 110.826, 79.384, 20.6671), cmt = 2) %>%
# get_data() # for curiosity, you can extract the data set at this step
  mapbayest()

# If `adm` or `obs` are set in `$cmt` in model code, the `cmt` argument are superfluous.

---

est001                         Estimation object

Description

An example of mapbayest object, corresponding to the parameter estimation of the 8 subjects from model 1. Note that the model object within is not associated to a shared object, which make some features unavailable. This object can be re-generated by executing est001 <- mapbayest(exmodel(ID = 1:8))

Usage

est001

Format

An object of class mapbayest of length 9.

See Also

mapbayest
exmodel_exdata  

Example model and data

Description

A collection of example models and corresponding data to test and explore mapb ayr.

Usage

```r
exmodel(
    num = 1,
    add_exdata = TRUE,
    cache = TRUE,
    quiet = getOption("mrgsolve_mread_quiet", TRUE),
    ...
)

exdata(num = 1, ID = 1, clean_data = TRUE)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>model number (see details)</td>
</tr>
<tr>
<td>add_exdata</td>
<td>should data be automatically loaded with the model</td>
</tr>
<tr>
<td>cache</td>
<td>read the model with mrgsolve::mread_cache()</td>
</tr>
<tr>
<td>quiet</td>
<td>don’t print messages when compiling</td>
</tr>
<tr>
<td>...</td>
<td>passed to mrgsolve::mread() or mrgsolve::mread_cache()</td>
</tr>
<tr>
<td>ID</td>
<td>individual number to include in the data (from 1 to 8)</td>
</tr>
<tr>
<td>clean_data</td>
<td>remove useless columns and rows from the original data</td>
</tr>
</tbody>
</table>

Details

Available models are:

- **1**: Base model. A simple mono compartmental PK model with inter-individual variability on absorption constant (KA), volume of distribution (VC) and clearance (CL). The residual error model is proportional.
- **6**: Complex absorption model. Dual 0- and 1st orders absorption phenomenons.
- **301**: Time-varying covariates. A continuous covariate (body weight "BW") and a categorical one (sex "SEX") influence the clearance parameter. In the corresponding dataset, the values randomly changes from one record to another within a single individual.
- **401**: Metabolite. The PK model of both a parent drug and its metabolite.
An example dataset of eight (simulated) individuals is available for each model. Individuals differ in terms of sampling times (sparse or rich) and dosing regimen (single or multiple dosing).

Model code and data files are stored at the location given by `system.file("exmodel", package = "mapbayr")`.

These models and data were created for the validation study of `mapbayr` published in CPT:Pharmacometrics & System Pharmacology. More models and full datasets can be accessed in a dedicated repository.

**Value**

`exmodel()` reads and compiles code, and returns a (mrgmod) model object. `exdata()` returns a data.frame.

**Source**

[https://github.com/FelicienLL/mapbayr-CPTPSP-2021](https://github.com/FelicienLL/mapbayr-CPTPSP-2021)

**Examples**

```r
# Models can be loaded with data (the default), ready for parameter estimation
est <- mapbayest(exmodel())

# Number of subjects in dataset can be chosen up to 8 individuals
exdata(301, ID = c(5,8))
```

---

**get_x**  

*Get content from object*

**Description**

Helpful functions to get content from a mrgmod object (i.e. data) or from a mapbayests object (data, eta, cov, param, phi).

**Usage**

```r
get_data(x, ...)
```

## S3 method for class 'mrgmod'

```r
get_data(x, ...)
```

## S3 method for class 'mapbayests'

```r
get_data(x, ..., output = "df")
```

```r
get_eta(x, ...)
```

## S3 method for class 'mapbayests'

```r
get_eta(x, ..., output = NULL)
```
get_cov(x, ...)

## S3 method for class 'mapbayests'
get_cov(x, ..., simplify = TRUE)

get_param(x, ...)

## S3 method for class 'mapbayests'
get_param(x, ..., output = NULL, keep_ID = NULL, keep_names = NULL)

get_phi(x, ...)

## S3 method for class 'mapbayests'
get_phi(x, ...)

**Arguments**

- **x**: mapbayests object
- **...**: not used
- **output**: either a data.frame ("df") or a vector of numeric ("num"). Default to "num" if only one ID
- **simplify**: a logical. If TRUE (the default) and only one ID, one matrix is returned instead of a list of length 1
- **keep_ID**: a logical. By default, the ID variable is dropped if one ID in data.
- **keep_names**: a logical. By default, names are dropped if one parameter is requested, and output is not a data frame.

**Value**

the class of the object returned depends on the function, and on their arguments. Typically, a data.frame or a vector if the output can be reduced to one line.

**Examples**

# From a model object (mrgmod)
mod <- exmodel(ID = 1:2, cache = FALSE, capture = "CL")
get_data(mod)

# From an estimation object (mapbayests)
est <- mapbayest(mod)
get_data(est)
gest <- mapbayest(mod)
gest <- est

geta(est)
gest <- est

get_data(est, output = "list")

geta(est, output = "list")
gest <- est

get_data(est)

get_param(est)
**h**

Compute the H matrix

### Description
Partial derivative of predictions with respect to epsilon

### Usage
h(pred, cmt, all_cmt, ...)

### Arguments
- **pred**: predictions (typically obtained from f())
- **cmt**: compartments predictions belong to (typically the cmt column of the dataset)
- **all_cmt**: all possible compartments with observations as defined in the model in $SIGMA$
  ... for compatibility (not used)

### Value
a matrix of dimensions [length(pred), 2 * length(all_cmt)]

### Examples
```r
mapbayr:::h(
pred = c(400, 40, 200, 20),
cmt = c(2, 3, 2, 3),
all_cmt = c(2, 3)
)
```

**hist.mapbayests** Plot posterior distribution of bayesian estimates

### Description
Plot posterior distribution of bayesian estimates

### Usage
```r
## S3 method for class 'mapbayests'
hist(x, ...)
```
Arguments

- `x`: A `mapbayests` object.
- `...`: additional arguments (not used)

Details

Use this function to plot the results of the estimations, in the form of histograms with the *a priori* distribution in the background. For every parameter, the inter-individual variability is displayed, as well as the percentile of the patient in the corresponding distribution (if `n = 1` patient). For additional modifications, you can add extra `+function(...)` in order to modify the plot as a regular `ggplot2` object.

Value

- a `ggplot` object.

Examples

```r
est <- mapbayest(exmodel(ID = 1))
hist(est) +
  ggplot2::labs(title = "Awesome estimations")
```

mapbayest

*Estimate parameters (maximum a posteriori)*

Description

The main function of the mapbayr package. Performs a *maximum a posteriori* Bayesian estimation of parameters, from a mrgsolve model object and a dataset containing information about administrations and observed concentrations.

Usage

```r
mapbayest(
  x,
  data = NULL,
  method = "L-BFGS-B",
  hessian = stats::optimHess,
  force_initial_eta = NULL,
  quantile_bound = 0.001,
  control = list(),
  check = TRUE,
  verbose = TRUE,
  progress = TRUE,
  reset = TRUE,
  output = NULL
)
```

mbrest(...)

Arguments

- **x**: the model object
- **data**: NMTRAN-like data set
- **method**: optimization method; possible values are L-BFGS-B (the default) and newuoa
- **hessian**: function used to compute the Hessian and variance-covariance matrix with (default is stats::optimHess, alternatively use nlmixr::nlmixrHess)
- **force_initial_eta**: a vector of numeric values to start the estimation from (default to 0 for "L-BFGS-B")
- **quantile_bound**: a numeric value representing the quantile of the normal distribution admitted to define the bounds for L-BFGS-B (default is 0.001, i.e. 0.1%)
- **control**: a list passed to the optimizer (see optimx documentation)
- **check**: check model code for mapbayr specification (a logical, default is TRUE)
- **verbose**: print a message whenever optimization is reset (a logical, default is TRUE)
- **progress**: print a progress bar (a logical, default is TRUE)
- **reset**: reset optimizer with new initial eta values if numerical difficulties, or with new bounds (L-BFGS-B) if estimate equal to a bound. (a logical, default is TRUE)
- **output**: if NULL (the default) a mapbayests object is returned; if df a mapbay_tab dataframe is returned
- **...**: passed to mapbayest

Value

A mapbayests object. Basically a list containing:

- **model**: the model object
- **arg.ofv.optim, arg.ofv.fix, arg.ofv.id**: arguments passed to the optimization function. Useful for debugging but not relevant for a basic usage. Access to the data with get_data(x)
- **opt.value**: the original output of the optimization function
- **final_eta**: a list of individual vectors of final estimates. Access it with x$final_eta or get_eta(x).
- **covariance**: a list of individual variance-covariance matrix of estimation. Access it with x$covariance or get_cov(x).
- **mapbay_tab**: an output table containing the results of your estimations (data, IPRED, PRED, covariates, captured items, ETA etc...). Access it with x$mapbay_tab, as.data.frame(x) or as_tibble(x).
- **information**: run times and package versions.

See Also

- hist.mapbayests
- plot.mapbayests
- use_posterior
Examples

# First, code a model
code1 <- "PARAM ETA1 = 0, ETA2 = 0, KA = 0.5, TVCL = 1.1, TVV = 23.3
OMEGA 0.41 0.32
SIGMA 0.04 0
CMT DEPOT CENT
PK
double CL=TVCL*exp(ETA1+ETA(1));
double V=TVV*exp(ETA2+ETA(2));
ERROR
double DV=CENT/V*(1+EPS(1)) + EPS(2);
PKMODEL ncmt = 1, depot = TRUE
CAPTURE DV CL"

my_model <- mrgsolve::mcode("my_model", code1)

# Then, import your data
my_data <- data.frame(ID = 1, TIME = c(0, 1.1, 5.2, 12.3), EVID = c(1, 0, 0, 0), AMT = c(500, 0, 0, 0),
                    CMT = c(1, 2, 2, 2), DV = c(0, 15.1, 29.5, 22.3))
print(my_data)

# And estimate
my_est <- mapbayest(x = my_model, data = my_data)
print(my_est)
# see also plot(my_est) and hist(my_est)

# Use your estimation
get_eta(my_est)
ge_param(my_est)
as.data.frame(my_est)
use_posterior(my_est)

-----

obs_cmt

Get observation compartment numbers from mrgsolve model

Description

Get observation compartment numbers from mrgsolve model

Usage

obs_cmt(x)

Arguments

x model object
Value
vector of integer

Examples
#Both parent drug and metabolite
model <- exmodel(401, compile = FALSE)
obs_cmt(model)

plot.mapbayests
Plot predictions from mapbayests object

Description
Plot predictions from mapbayests object

Usage
## S3 method for class 'mapbayests'
plot(x, ...)

Arguments
x  A mapbayests object.
...   additional arguments (passed to augment.mapbayests)

Details
Use this function to plot the results of the estimations, in the form of concentration vs time profiles
for every patient of the data set. For additional modifications, you can:

• see augment.mapbayests to modify the simulation output.
• add extra +function(...) in order to modify the plot as a regular ggplot2 object.

Value
a ggplot object.

Examples
est <- mapbayest(exmodel(ID = 1))
plot(est, end = 48) +
ggplot2::labs(title = "Awesome prediction")
**Postprocess mapbayr**

**Description**

Functions to generate postprocess after optimization.

**Usage**

```r
postprocess.optim(x, data, opt.value, arg.ofv, arg.optim, hessian)

postprocess.output(
  x,
  arg.optim,
  arg.ofv.fix,
  arg.ofv.id,
  opt.value,
  post,
  output,
  times
)
```

**Arguments**

- `x` the model object
- `data` NMTRAN-like data set
- `opt.value` value returned by optimizer
- `arg.optim, arg.ofv, arg.ofv.fix, arg.ofv.id` argument passed to optimizer
- `hessian` function used to compute the Hessian and variance-covariance matrix with (default is `stats::optimHess`, alternatively use `nlmixr::nlmixrHess`)
- `post, times` output of the post.process function
- `output` if NULL (the default) a mapbayests object is returned; if df a `mapbay_tab` dataframe is returned

**Value**

`postprocess.optim()` returns a list with final parameters and `mapbay_tab`. `postprocess.output()` returns a `mapbayests` class object.
Preprocess model and data for ofv computation

Description

Functions to generate arguments passed to `compute_ofv`. Arguments that are fixed between individuals are created once (preprocess.ofv.fix), while others are specific of each individual (preprocess.ofv.id).

Usage

```r
preprocess.ofv.fix(x, data)
preprocess.ofv.id(x, iddata)
```

Arguments

- `x` the model object
- `data`, `iddata` NMTRAN-like data set. `iddata` is likely a dataset of one individual

Value

A list of arguments used to compute the objective function value.

The following arguments are fixed between individuals:

- `qmod`: model object, modified to simulate without random effects and with controlled outputs
- `sigma`: a single matrix object
- `log_transformation`: a logical, whether predictions need to be log-transformed for ofv computation
- `omega_inv`: a single matrix object
- `all_cmt`: a vector of compartment numbers where observations can be expected

The following arguments differs between individuals:

- `idvaliddata`: a matrix, individual data set (with administrations and covariates), validated with `valid_data_set`
- `idDV`: a vector of (possibly log-transformed) observations
- `idcmt`: a vector of compartments where observations belong to

Examples

```r
mod <- exmodel(add_exdata = FALSE, compile = FALSE)
dat <- exdata(ID = c(1,4))

preprocess.ofv.fix(x = mod, data = dat)
preprocess.ofv.id(x = mod, iddata = dat[dat$ID == 1,])
preprocess.ofv.id(x = mod, iddata = dat[dat$ID == 4,])
```
preprocess.optim  

Pre-process: arguments for optimization function

Description
Pre-process: arguments for optimization function

Usage
preprocess.optim(x, method, control, force_initial_eta, quantile_bound)

Arguments
- **x**: the model object
- **method**: optimization method; possible values are \texttt{L-BFGS-B} (the default) and \texttt{newuoa}
- **control**: a list passed to the optimizer (see \texttt{optimx} documentation)
- **force_initial_eta**: a vector of numeric values to start the estimation from (default to 0 for \texttt{"L-BFGS-B"})
- **quantile_bound**: a numeric value representing the quantile of the normal distribution admitted to define the bounds for \texttt{L-BFGS-B} (default is 0.001, i.e. 0.1%)

Value
a list of named arguments passed to optimizer (i.e. \texttt{arg.optim})

print.mapbayests  

Print a mapbayests object

Description
Print a mapbayests object

Usage

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

Arguments
- **x**: A mapbayests object.
- **...**: additional arguments

Value
print the results of the estimation to the console, and returns it invisibly.
Quick parameter update

Description

Quick parameter update

Usage

$qparam(x, p)$

Arguments

- **x**: model object
- **p**: a named list/vector of parameters to update

Details

Faster update of parameters inside the model object. Useful in the context of parameter optimization, otherwise consider the regular `param()` because speed comes at the cost of safety.

Value

model object

Examples

```r
library(mrgsolve)
ho <- house()
## Not run:
system.time(replicate(1000, param(ho, c(CL = .1, VC = 1))))
system.time(replicate(1000, qparam(ho, c(CL = .1, VC = 1))))
## End(Not run)
```

Use posterior estimation

Description

Use posterior estimation
**Usage**

```r
use_posterior(
  x,
  update_omega = FALSE,
  update_cov = TRUE,
  update_eta = TRUE,
  .zero_re = NULL,
  simplify = TRUE
)
```

**Arguments**

- `x`: A `mapbayests` object.
- `update_omega`: Update the OMEGA matrix with the variance-covariance matrix of estimation (a logical, default is `FALSE`).
- `update_cov`: Update the values of covariates with the individual values (a logical, default is `TRUE`).
- `update_eta`: Update the values of ETA with the final estimates (a logical, default is `TRUE`).
- `.zero_re`: Set all elements of the OMEGA or SIGMA matrix to zero. Default is "both" if `update_omega` is `FALSE`, "sigma" otherwise. (possible values are "both", "sigma", "omega", "none")
- `simplify`: a logical. If `TRUE` (the default) and only one ID, one `mrgmod` is returned instead of a list of length 1

**Details**

This function takes the results of an estimation (i.e. a `mapbayests` object) and return a modified `mrgmod` in order to perform a posteriori simulations. Modifications are:

- If `update_eta` is `TRUE`, the values of ETA are updated to the estimated values (instead of 0) in `SPARAM`.
- If `update_cov` is `TRUE`, the covariates values are updated to the values of the individual (instead of default model values) in `SPARAM`.
- If `update_omega` is `TRUE`, the values of OMEGA are updated with the variance-covariance matrix of estimation (i.e. an approximation of the a posteriori distribution) instead of the inter-individual variability (i.e. the a priori distribution). Use this command in order to derive a confidence interval of concentrations that反映s the uncertainty about parameter estimation when a large number of profiles are simulated. Note that if inter-individual variability was initially defined in multiple $OMEGA$ blocks in the model, they will be collapsed to a single full matrix (this is irreversible).
- Depending on the values of `.zero_re`, the elements of $OMEGA$ or $SIGMA$ can be set to zero, whether you want to simulate one profile, or several in order to derive confidence/prediction intervals. It does not handle time-varying covariates: only the first value will be used as the individual value.
Value

a mrgmod, or a list of mrgmod if there is more than 1 ID

Examples

```r
library(magrittr)
est <- mapbayest(exmodel())
est %>%
  use_posterior() %>%
mrgsolve::ev(amt = 50000) %>%
mrgsolve::mrgsim()
```

---

**Description**

Compare results to NONMEM .phi

**Usage**

```r
read_nmphi(x)
merge_phi(mapbayr_phi, nonmem_phi)
plot_phi(merged_phi, only_ETA = TRUE)
```

**Arguments**

- `x` full path to a .phi file generated by NONMEM
- `mapbayr_phi` results of mapbayr estimations, in the form of a tibble data.frame, typically obtained from get_phi()
- `nonmem_phi` results of NONMEM estimations, in the form of a tibble data.frame, typically obtained from read_nmphi()
- `merged_phi` merged results of estimations, typically obtained from merge_phi()
- `only_ETA` filter the data with type=="ETA" before the plot (a logical, default is TRUE)

**Details**

These functions were made to easily compare the results of mapbayr to NONMEM. For instance, it could be useful in the case of the transposition of a pre-existing NONMEM model into mapbayr. For this, you need to code your model in both mapbayr and NONMEM, and perform the MAP-Bayesian estimation on the same dataset. Ideally, the latter contains a substantial number of patients. NONMEM returns the estimations results into a .phi file.

Use `read_nmphi()` to parse the NONMEM .phi file into a convenient tibble data.frame with the columns:
• **SUBJECT_NO, ID**: Subject identification.
• **ETA1, ETA2, ..., ETAn**: Point estimates of eta.
• **ETC1_1, ETC2_1, ETC2_2, ..., ETCn_n**: Variance-covariance matrix of estimation.
• **OBJ**: Objective function value

Use `get_phi()` to access the estimations of mapbayr with the same "phi" format.

Use `merge_phi()` to combine mapbayr and NONMEM "phi files" into a single long-form data.frame with the columns:

• **SUBJECT_NO, ID**: Subject identification.
• **variable name and its type**: ETA (point estimate), VARIANCE (on-diagonal element of the matrix), COVARIANCE (off-diagonal), and OBJ.
• **mapbayr and nonmem**: corresponding values
• **adiff**: absolute difference between mapbayr and nonmem values.

Use `plot_phi()` to graphically represent adiff vs variable. Alternatively, the table returned by `merge_phi()` is easy to play with in order to derive performance statistics or the graphical plot of your choice.

**Value**

• **read_nmphi**: a tibble data.frame with a format close to the original .phi file
• **merge_phi**: a long-form tibble data.frame with results of mapbayr and NONMEM
• **plot_phi**: a ggplot2 object

**Examples**

```r
library(mapbayr)
nmphi <- read_nmphi(system.file("nm001", "run001.phi", package = "mapbayr"))
est001 |> get_phi() |> merge_phi(nmphi) |> plot_phi()
```
Index

* datasets
  - est001, 9
  - add_covariates (data_helpers), 8
  - adm_0_cmt, 2
  - adm_cmt, 3
  - adm_lines (data_helpers), 8
  - as.data.frame.mapbayests, 3
  - augment, 4
  - augment.mapbayests, 4, 17
  - check_mapbayr_model, 6
  - compute_ofv, 7, 19
  - data_helpers, 8
  - do_compute_ofv (compute_ofv), 7
  - est001, 9
  - exdata (exmodel_exdata), 10
  - exmodel (exmodel_exdata), 10
  - exmodel_exdata, 10
  - get_cov (get_x), 11
  - get_data (get_x), 11
  - get_eta (get_x), 11
  - get_param (get_x), 11
  - get_phi (get_x), 11
  - get_x, 11
  - h, 13
  - hist.mapbayests, 13, 15
  - mapbayest, 9, 14
  - mbrest (mapbayest), 14
  - merge_phi (vs_nonmem), 23

  - obs_cmt, 16
  - obs_lines (data_helpers), 8
  - optimx, 15, 20

  - plot.mapbayests, 15, 17

plot_phi (vs_nonmem), 23
postprocess, 18
preprocess.ofv, 7, 19
preprocess.ofv.fix, 7
preprocess.ofv.id, 7
preprocess.optim, 20
print.mapbayests, 20
qparam, 21
read_nmphi (vs_nonmem), 23
see_data (data_helpers), 8
use_posterior, 15, 21
valid_data_set, 19
vs_nonmem, 23