Package 'mapbayr'

October 13, 2022

Title MAP-Bayesian Estimation of PK Parameters

Version 0.8.0

Description Performs maximum a posteriori Bayesian estimation of individual pharmacokinetic parameters from a model defined in 'mrgsolve', typically for model-based therapeutic drug monitoring. Internally computes an objective function value from model and data, performs optimization and returns predictions in a convenient format. The performance of the package was described by Le Louedec et al (2021) <doi:10.1002/psp4.12689>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.0

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

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

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

Suggests knitr, minqa, testthat (>= 3.0.0), tibble

Depends R (>= 2.10)

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Felicien Le Louedec [aut, cre]
(<`https://orcid.org/0000-0003-3699-2729`>),
Kyle T Baron [ctb] (<`https://orcid.org/0000-0001-7252-5656`>)

Maintainer Felicien Le Louedec <`felicienlelouedec@proton.me`>

Repository CRAN

Date/Publication 2022-09-29 11:50:06 UTC

R topics documented:

`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` *Compute full PK profile prediction from mapbayr estimates.*

Description

Compute full PK profile prediction from mapbayr estimates.

Usage

```
augment(x, ...)
```

Arguments

`x`	object to augment
`...`	additional arguments

Value

an augmented object (depending on the object passed).

`augment.mapbayests` *Compute full PK profile prediction from mapbayr estimates.*

Description

Compute full PK profile prediction from mapbayr estimates.

Usage

```
## 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

Checks that the model respects points related exclusively to 'mapbayr'. Useful at the time you wish to convert a "regular" 'mrgsolve' model you used for simulation into a model to perform MAP-Bayesian estimation. Note that some elements cannot be checked:

- In `$MAIN` block, make sure that you added `ETA1, ETA2...` in the code. For instance: `double CL = TVCL * exp(ETA(1) + ETA1) ;`.
- In `$OMEGA` block, make sure the order of the (diagonal) values is the same as for ETAs in `$PARAM`. For instance, if `ETA1` corresponds to clearance, the first value in `$OMEGA` must be the variance of clearance.
- In `$SIGMA` block, make sure the order is respected: proportional error first, and additive error secondly.

Usage

```
check_mapbayr_model(x, check_compile = TRUE)
```

Arguments

`x`	model file
`check_compile`	check if model is compiled

Value

`TRUE` (invisibly) if checks are passed, errors otherwise.

Examples

```
library(mapbayr)
library(mrgsolve)
## Not run: check_mapbayr_model(house())
```

`compute_ofv` *Compute the objective function value*

Description

Compute the objective function value

Usage

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

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` *Data helpers*

Description

Helpers to build data set.

Usage

```
adm_lines(x, ...)

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

obs_lines(x, time, DV, mdv = 0, cmt = NULL, DVmet = NULL, ...)

## S3 method for class 'mrgmod'
obs_lines(x, time, DV, mdv = NULL, cmt = NULL, DVmet = NULL, ...)

add_covariates(x, covariates, ...)

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

see_data(x, ...)
```

Arguments

`x`	model object
`...`	passed to `mrgsolve::ev()` in `adm_lines()`
`time`, `DV`, `mdv`, `cmt`, `DVmet`	passed to `obs_lines()`
`covariates`	a list of named covariates, with a single value or same number of lines than data

Details

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 :

- 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 `mapbayests` 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 `mapbayests` of length 9.

See Also

`mapbayest`

`eta` *Generate a vector of "ETA"*

Description

Generate a vector of "ETA" values. If `x` is a `mrgsolve` model, these will be extracted from values defined in `$PARAM`. Otherwise, any numeric values passed to `x` and `...` as vector(s) or list(s) will be coerced as a single vector. Alternatively, if `x` and `...` are missing, generate a vector of ETA equal to `val` of length `n`.

Usage

```
eta(x, ..., n, val = 0)
```

Arguments

`x`	either a `mrgsolve` model object, or a numeric
`...`	additional numeric(s)
`n, val`	generate a sequence of `val` of length `n`

Value

a single named vector of numeric

Examples

```
# Extract ETA from the model
mod <- exmodel()
eta(mod)

# Coerce numeric values
eta(0.1, 0.2, c(0.3, 0.4), list(0.5, 0.6))
eta(rnorm(4))

# Generate a sequence from scratch
eta(n = 3)
eta(n = 3, val = 0.001)
```

`exmodel_exdata` *Example model and data*

Description

A collection of example models and corresponding data to test and explore `mapbayr`.

Usage

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

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

Arguments

`num`	model number (see details)
`add_exdata`	should data be automatically loaded with the model
`cache`	read the model with `mrgsolve::mread_cache()`
`quiet`	don't print messages when compiling
`...`	passed to `mrgsolve::mread()` or `mrgsolve::mread_cache()`
`ID`	individual number to include in the data (from 1 to 8)
`clean_data`	remove useless columns and rows from the original data

Details

Available models are:

- 1: Base model. A simple monocompartmental 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`

Examples

```
# 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

```
get_data(x, ...)

## S3 method for class 'mrgmod'
get_data(x, ...)

## S3 method for class 'mapbayests'
get_data(x, ..., output = "df")

get_eta(x, ...)

## S3 method for class 'mapbayests'
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)
get_data(est, output = "list")

get_eta(est)
get_eta(est, output = "list")

get_cov(est)

get_param(est)

get_phi(est)
```

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

Description

Plot posterior distribution of bayesian estimates

Usage

```
## 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

```
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

```
mapbayest(
  x,
  data = NULL,
  method = c("L-BFGS-B", "newuoa"),
  hessian = stats::optimHess,
  force_initial_eta = NULL,
  quantile_bound = 0.001,
  control = list(),
  check = TRUE,
  verbose = TRUE,
  progress = TRUE,
  reset = 50,
  output = NULL
)

mbrest(...)
```

Arguments

`x`	the model object
`data`	NMTRAN-like data set
`method`	optimization method; the default is `"L-BFGS-B"` (from `stat::optim()`), alternatively `"newuoa"` for `minqa::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 `stats::optim()` or `minqa::newuoa()` 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`	maximum allowed reset of the optimizer with new initial eta values if numerical difficulties, or with new bounds (L-BFGS-B) if estimate equal to a bound. (a numeric, default is 50)
`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)
get_param(my_est)
as.data.frame(my_est)
use_posterior(my_est)
```

`plot.mapbayests` *Plot predictions from mapbayests object*

Description

Plot predictions from mapbayests object

Usage

```
## S3 method for class 'mapbayests'
plot(x, ..., PREDICTION = c("IPRED", "PRED"))
```

Arguments

`x`	A `mapbayests` object.
`...`	additional arguments (passed to `augment.mapbayests`)
`PREDICTION`	plot either "IPRED", "PRED" or both.

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")
```

`preprocess.ofv` *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

```
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

```
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 = c("L-BFGS-B", "newuoa"),
  control = list(),
  force_initial_eta = NULL,
  quantile_bound = 0.001
)
```

Arguments

`x`	the model object
`method`	optimization method; the default is `"L-BFGS-B"` (from `stat::optim()`), alternatively `"newuoa"` for `minqa::newuoa()`
`control`	a list passed to the optimizer (see `stats::optim()` or `minqa::newuoa()` documentation)
`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%)

Value

a list of named arguments passed to optimizer (i.e. 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.

`use_posterior` *Use posterior estimation*

Description

Use posterior estimation

Usage

```
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 $PARAM.
- If `update_cov` is `TRUE`, the covariates values are updated to the values of the individual (instead of default model values) in $PARAM.
- 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 reflects 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

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

`vs_nonmem` *Compare results to NONMEM .phi*

Description

Compare results to NONMEM .phi

Usage

```
read_nmphi(x)

merge_phi(mapbayr_phi, nonmem_phi)

plot_phi(merged_phi, only_ETA = TRUE)

summarise_phi(
  merged_phi,
  group,
  only_ETA = TRUE,
  levels = c(Excellent = 0, Acceptable = 0.001, Discordant = 0.1)
)

bar_phi(summarised_phi, xaxis = NULL, facet = NULL)
```

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"` (a logical, default is `TRUE`)
`group`	one or multiple variables to `group_by()`
`levels`	a named vector of length 3 in order to classify the absolute differences. Default cut-offs are 0.1% and 10% in the parameters space.
`summarised_phi`	summarized results of estimations, typically obtained from `summarise_phi()`
`xaxis`	optional. A character value, that correspond to a variable in data, passed to the x-axis to plot multiple bars side-by-side.
`facet`	a formula, that will be passed to `ggplot2::facet_wrap()`

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 to 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.

Use `summarise_phi()` to classify the estimation as "Excellent", "Acceptable" or "Discordant", over the whole dataset or by `group`.

Use `bar_phi()` to graphically represent the proportion of the aforementioned classification as bar plot.

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
- summarise_phi: a summarized tibble data.frame classifying the performance of mapbayr and NONMEM
- plot_phi, bar_phi: a `ggplot2` object

Examples

```
library(mapbayr)
nmphi <- read_nmphi(system.file("nm001", "run001.phi", package = "mapbayr"))
mapbayrphi <- get_phi(est001)

merged <- merge_phi(mapbayrphi, nmphi)
plot_phi(merged)

summarised <- summarise_phi(merged)
bar_phi(summarised)


# Analyse the results of multiple runs simultaneously

#Example dataset that represents 3 runs
merge3 <- dplyr::bind_rows(merged, merged, merged, .id = "RUN")
merge3$adiff <- 10 ^ runif(nrow(merge3), -8, 0)

summarised3 <- summarise_phi(merge3, group = RUN)
bar_phi(summarised3, xaxis = "RUN")
```

`x_cmt` *Read compartment options in a model*

Description

Read compartment options in a model

Usage

```
adm_cmt(x)

obs_cmt(x)
```

Arguments

`x`	model object

Details

In a mrgsolve model, it is possible to specify options in `$CMT`. If `[ADM]` or `[OBS]` are set, mapbayr will interpret these as defaults administration and observation compartments, respectively.

Value

a vector of compartment identified as default "administration" or "observation" compartments.

Examples

```
#Administration:  Both 1st and 0- order
model <- exmodel(6, compile = FALSE)
mrgsolve::see(model)
adm_cmt(model)

#Observation: Both parent drug and metabolite
model <- exmodel(401, compile = FALSE)
mrgsolve::see(model)
obs_cmt(model)
```

Index