Package: nlmixr2autoinit (via r-universe)

October 13, 2026

Type Package

Title Automatic Generation of Initial Estimates for Population Pharmacokinetic Modeling

Version 1.0.1

Description Provides automated methods for generating initial parameter estimates in population pharmacokinetic modeling. The pipeline integrates adaptive single-point methods, naive pooled graphic approaches, noncompartmental analysis methods, and parameter sweeping across pharmacokinetic models. It estimates residual unexplained variability using either data-driven or fixed-fraction approaches and assigns pragmatic initial values for inter-individual variability. These strategies are designed to improve model robustness and convergence in 'nlmixr2' workflows. For more details see Huang Z, Fidler M, Lan M, Cheng IL, Kloprogge F, Standing JF (2025) <doi:10.1007/s10928-025-10000-z>.

License GPL (>= 3)

URL https://ucl-pharmacometrics.github.io/nlmixr2autoinit, https://github.com/ucl-pharmacometrics/nlmixr2autoinit

BugReports https://github.com/ucl-pharmacometrics/nlmixr2autoinit/issues

Depends nlmixr2data, R (>= 4.0)

Imports nlmixr2, nlmixr2est, rxode2, dplyr, tidyr, crayon, purrr, knitr, magrittr, progressr, tibble, vpc

Encoding UTF-8

Language en-US

NeedsCompilation no

Roxygen list(markdown = TRUE)

Suggests testthat (>= 3.0.0), progress, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libgmp3-dev make libicu-dev libpng-dev libuv1-dev libmpfr-dev libssl-dev libx11-dev zlib1g-dev

Repository https://ucl-pharmacometrics.r-universe.dev

Date/Publication 2026-06-09 11:02:43 UTC

RemoteUrl https://github.com/ucl-pharmacometrics/nlmixr2autoinit

RemoteRef HEAD

RemoteSha 30e48b0acdd4fdf324935149e2a6263a8437e022

Contents

approx.vc . . . 3
bin.time . . . 5
calculate_cl . . . 6
calculate_tad . . . 7
calculate_vd . . . 8
eval_perf_1cmpt . . . 10
fallback_control . . . 11
find_best_lambdaz . . . 12
Fit_1cmpt_iv . . . 14
Fit_1cmpt_mm_iv . . . 15
Fit_1cmpt_mm_oral . . . 17
Fit_1cmpt_oral . . . 19
Fit_2cmpt_iv . . . 20
Fit_2cmpt_oral . . . 22
Fit_3cmpt_iv . . . 23
Fit_3cmpt_oral . . . 25
force_find_lambdaz . . . 27
get_hf . . . 28
get_pooled_data . . . 30
getnca . . . 31
getOmegas . . . 33
getPPKinits . . . 33
getsigma . . . 35
getsigmas . . . 36
graphcal_iv . . . 37
graphcal_oral . . . 38
hybrid_eval_perf_1cmpt . . . 39
initsControl . . . 41
is_ss . . . 42
ka_calculation_md . . . 43
ka_calculation_sd . . . 44
ka_wanger_nelson . . . 45
mark_dose_number . . . 47

metrics. 48
nca_control 49
nmpkconvert 50
pooled_control 51
print.getPPKinits 52
processData 53
run_graphcal 55
run_ka_solution 56
run_pooled_nca 58
run_single_point 59
run_single_point_base 60
run_single_point_extra 62
sim_sens_1cmpt_mm 64
sim_sens_2cmpt 66
sim_sens_3cmpt 68
ss_control 70
trapezoidal_linear 71
trapezoidal_linear_up_log_down 72
trimmed_geom_mean 73

Index **74**

`approx.vc` *Approximate volume of distribution from observed Cmax*

Description

Estimates the volume of distribution (Vd) from observed peak concentrations (Cmax) in single-dose, multiple-dose, or mixed datasets.

Usage

```
approx.vc(
  dat = NULL,
  half_life = NULL,
  single_point_base.lst = NULL,
  route = c("bolus", "oral", "infusion"),
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  ssctrl = ss_control()
)
```

Arguments

`dat`	A data frame containing pharmacokinetic data, including observed concentrations (DV), time after dose (tad), dose, and route information.
`half_life`	The elimination half-life (t1/2) of the compound, used to identify early-phase Cmax values.

`single_point_base.lst`
: Optional list object returned by run_single_point_base(). If not supplied, the function will generate it internally.

`route`
: Route of administration. One of "bolus", "oral", or "infusion" (default = "bolus").

`dose_type`
: Optional string specifying the dosing type, passed to run_single_point_base().

`pooled_ctrl`
: Control object created by pooled_control(), defining data pooling options.

`ssctrl`
: Control object created by ss_control(), defining steady-state control options.

Details

Estimates individual apparent volumes of distribution from observed peak concentrations. Individual estimates are then summarized to obtain a population-level value.

For single-dose data, Vd is calculated according to the route of administration:

- Bolus: $V_d = \text{Dose}/C_{\max}$
- Infusion: $V_d = (\text{Rate} \times t_{\text{inf}})/C_{\max}$
- Oral: $V_d = (\text{Dose} \times F)/C_{\max}$, where $F = 1 - e^{-k_a t}$

For multiple-dose data, observed Cmax values are adjusted to single-dose equivalents using the accumulation ratio:

$$R_{\text{ac}} = \frac{1}{1 - e^{-k_e \tau}}, \quad k_e = \ln(2)/t_{1/2}$$

Adjusted values are used to estimate Vd using the same route-specific equations.

Value

A list containing individual and population Vd estimates and related dose-level data.

Author(s)

Zhonghui Huang

See Also

`run_single_point_base`, `trimmed_geom_mean`, `pooled_control`, `ss_control`

Examples

```
# Process dataset
out <- processData(Bolus_1CPT)
# Get half-life and dose route
hf <- get_hf(dat = out$dat)$half_life_median
rt <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]
# Estimate Vd
approx.vc(dat = out$dat, half_life = hf, route = rt)$approx.vc.value
```

`bin.time` *Bin time-concentration data using quantile or algorithmic binning*

Description

Bins data by time using either equal-frequency (quantile) binning or algorithmic binning methods.

Usage

```
bin.time(
  dat,
  nbins = "auto",
 bin.method = c("quantile", "jenks", "kmeans", "pretty", "sd", "equal", "density")
)
```

Arguments

`dat`	A data frame containing PK data. Must include: • tad: time after dose • DVstd: standardized concentration (DV/dose) • EVID: optional event ID column used to filter observations (EVID == 0)
`nbins`	Number of bins or "auto". If numeric with `bin.method = "quantile"`, specifies equal-frequency bins. If "auto", 10 bins are used for quantile; otherwise binning is determined by `vpc::auto_bin`. Numeric nbins for non-quantile methods is passed to `vpc::auto_bin`.
`bin.method`	Binning strategy (default = "quantile"). Available options are: • quantile: equal-frequency binning by empirical quantiles • jenks: natural breaks minimizing within-bin variance • kmeans, pretty, sd, equal, density: alternative binning methods from vpc::auto_bin

Details

Supports quantile-based binning and other data-driven methods (jenks, kmeans, pretty, sd, equal, density), with optional automatic bin count selection.

Value

A list containing summary results of the time-concentration binning process.

Author(s)

Zhonghui Huang

See Also

`vpc::auto_bin`

Examples

```
dat <- Bolus_1CPT
dat <- nmpkconvert(dat)
dat <- calculate_tad(dat)
dat$DVstd <- dat$DV / dat$dose
bin.time(dat)
```

`calculate_cl` *Calculate clearance using an adaptive single-point method*

Description

Calculates clearance using an adaptive single-point pharmacokinetic method

Usage

```
calculate_cl(
  dat,
  half_life = NULL,
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  ssctrl = ss_control()
)
```

Arguments

`dat`	A data frame containing pharmacokinetic data. Required columns typically include ID, TIME, DV, tad, recent_ii, dose, routeobs, and durationobs.
`half_life`	Optional numeric value for the drug's half-life. If not provided, half-life is estimated using `get_hf()` from pooled observations.
`dose_type`	Specifies the dosing context of the pharmacokinetic observations. Required when half_life is not provided. Classified as first_dose, repeated_doses, or combined_doses based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts.
`pooled_ctrl`	Optional list of control parameters used by `get_pooled_data()` for pooling observations. Defaults to output from `pooled_control()`.
`ssctrl`	A list of control parameters generated by `ss_control()` to guide the detection of steady-state observations.

Details

Estimates individual and population clearance from steady-state pharmacokinetic data. If half-life is not provided, it is estimated from pooled data using `get_hf()` and pooling rules defined in `pooled_control()`.

The procedure:

- Identifies steady-state observations using `is_ss()` and `ss_control()` criteria.
- Selects peak and trough concentrations within each dose interval to represent steady-state behavior.
- Classifies concentration points as Cssmax, Cssmin, or Cssavg based on timing within the interval and decay pattern.
- Computes individual clearance as Dose / (Cssavg × tau).
- Aggregates individual clearance values using a trimmed geometric mean to obtain a population estimate.

Supports bolus, infusion, and oral administration routes.

Value

A list containing:

- dat: the processed dataset with steady-state identification
- cl_df: individual clearance estimates
- trimmed_mean_cl: the population clearance calculated as a trimmed geometric mean with a 5 percent trimming level to reduce the impact of outliers

Author(s)

Zhonghui Huang

See Also

`get_hf`, `get_pooled_data`, `pooled_control`, `is_ss`, `ss_control`, `trimmed_geom_mean`

Examples

```
dat <- processData(Bolus_1CPT)$dat
calculate_cl(dat, get_hf(dat)$half_life_median)$trimmed_mean_cl
```

`calculate_tad` *Calculate time after dose for pharmacokinetic data*

Description

Calculate time after dose (TAD) for pharmacokinetic observations.

Usage

```
calculate_tad(dat, verbose = FALSE)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
verbose	Logical; if TRUE, prints informational messages during processing (e.g., when generating dose numbers). Default is FALSE.

Details

The procedure identifies dosing events based on the event identifier (EVID) and assigns each observation the attributes of the most recent prior dose. The time after dose is then calculated for observation rows. If dose_number column is not present in the input, it is automatically created for each subject.

Value

A modified data frame with added columns:

- tad: time after dose, calculated as the observation time minus the time of the most recent prior dose; set to NA for dosing records
- iiobs: interdose interval inherited from the most recent dosing record
- rateobs: infusion rate inherited from the most recent dosing record
- routeobs (optional): route of administration inherited from the most recent dosing record, included only if route information is present
- dose_number: sequential dose number, generated if not already present

Author(s)

Zhonghui Huang

Examples

```
calculate_tad(Bolus_1CPT)
calculate_tad(Infusion_1CPT)
calculate_tad(Oral_1CPT)
```

---

calculate_vd *Calculates volume of distribution from concentration data*

Description

Calculates the volume of distribution (Vd) using an adaptive single-point approach

Usage

```
calculate_vd(
  dat,
  half_life = NULL,
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  route = c("bolus", "oral", "infusion")
)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
half_life	Optional numeric value for the drug's half-life. If not provided, it will be estimated using `get_hf()` from pooled observations.
dose_type	Specifies the dosing context of the pharmacokinetic observations. Required when half_life is not provided. Classified as first_dose, repeated_doses, or combined_doses based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts.
pooled_ctrl	Optional list of control parameters used by `get_pooled_data()` for pooling observations. Defaults to output from `pooled_control()`.
route	Character string specifying the route of administration. Must be one of bolus, oral, or infusion. Currently, oral is not implemented.

Details

The function uses a concentration observed within the first 20% of the elimination half-life after dosing as the early point for estimating the volume of distribution.

$$Vd = \frac{\text{Dose}}{C_0}$$

For infusion:

$$Vd = \frac{\text{Rate} \times \min(\text{TIME}, \text{durationobs})}{C_0}$$

Here, C_0 represents the early concentration observed within the first 20% of the elimination half-life after dosing, which is used as an approximation of the initial concentration for estimating volume of distribution (Vd). `TIME` refers to time after dose; `durationobs` is the actual infusion duration.

When half_life is not provided, it is estimated from pooled data using the functions `get_pooled_data()` and `get_hf()`.

Value

A list with two elements:

- vd_df: individual volume of distribution estimates
- trimmed_mean_vd: population volume of distribution estimated as a trimmed geometric mean using a 5 percent trimming level

Author(s)

Zhonghui Huang

See Also

get_pooled_data, get_hf, trimmed_geom_mean

Examples

```
dat <- Bolus_1CPT
out <- processData(dat)
fdat<- out$dat
froute <-out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]
half_life <- get_hf(dat = fdat)$half_life_median
calculate_vd(dat = fdat, half_life = half_life,route=froute)$trimmed_mean_vd
```

`eval_perf_1cmpt` *Evaluates predictive performance of a one-compartment model*

Description

Computes predictive error metrics by comparing simulated and observed concentration–time data using specified pharmacokinetic parameters and dosing route.

Usage

```
eval_perf_1cmpt(
  dat,
  est.method = "rxSolve",
  ka = NULL,
  cl = NULL,
  vd = NULL,
  route = c("bolus", "infusion", "oral"),
  ncores = 2
)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.
`est.method`	Estimation method passed to the fitting function. Defaults to using `rxSolve` for model simulation and parameter estimation.
`ka`	Absorption rate constant.
`cl`	Clearance value.
`vd`	Volume of distribution.

route	A character string indicating the route of administration. Must be one of "oral", "infusion", or "bolus". Defaults to "bolus".
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.

Details

Internally selects the appropriate one-compartment model fitting function, using Fit_1cmpt_oral() for oral administration and Fit_1cmpt_iv() for intravenous administration. Predictive performance is quantified using the metrics.() function.

Value

A numeric vector containing absolute prediction error, mean absolute error, mean absolute percentage error, root mean square error, and relative root mean square error.

See Also

Fit_1cmpt_oral, Fit_1cmpt_iv, metrics.

Examples

```
eval_perf_1cmpt(
  dat = pheno_sd,
  est.method = "rxSolve",
  cl = 0.006,
  vd = 1,
  route = "bolus"
)
```

---

fallback_control *Control settings for fallback rules in parameter estimation*

Description

Control settings for fallback rules in parameter estimation

Usage

```
fallback_control(
  enable_ka_fallback = TRUE,
  sigma_method_additive = "model",
  sigma_method_proportional = "model",
  sigma_fallback_fraction = 0.2
)
```

Arguments

`enable_ka_fallback`
: Logical value indicating whether to apply a fallback to ka = 1 if the estimated value is invalid.

`sigma_method_additive`
: Method for additive sigma. Options are "model" or "fixed_fraction".

`sigma_method_proportional`
: Method for proportional sigma. Options are "model" or "fixed_fraction".

`sigma_fallback_fraction`
: Numeric value specifying the fallback fraction, for example, 0.2 corresponds to 20 percent of the mean of observed concentrations.

Value

A list of fallback control parameters.

Examples

```
fallback_control()
```

`find_best_lambdaz` *Find the best terminal elimination rate constant (lambdaz)*

Description

Identifies the optimal terminal phase for lambdaz estimation using a systematic log-linear regression approach with adjusted R-squared optimization criteria.

Usage

```
find_best_lambdaz(
  time,
  conc,
  route = "bolus",
  duration = NULL,
  adj_r_squared_threshold = 0.7,
  nlastpoints = 3,
  tolerance = 1e-04
)
```

Arguments

`time`
: Numeric vector of observation time points.

`conc`
: Numeric vector of concentration measurements corresponding to time points.

`route`
: Administration method specification:

- "bolus" (default) - Excludes time of maximum concentration (Tmax) point
- "infusion" - Includes Tmax point in terminal phase evaluation
- "oral" - Starts regression from Tmax point

`duration` Numeric (optional). Duration of infusion administration, in the same time units as `time`. Required only when `route = "infusion"`. Used to determine the first post-infusion observation time for terminal phase regression.

`adj_r_squared_threshold`
Minimum acceptable adjusted R-squared value for valid estimation (default = 0.7). Values below this threshold will generate warnings.

`nlastpoints` Integer. Minimum number of terminal points (from the end of the profile) to include when evaluating candidate regression segments for λ_z. Default is 3.

`tolerance` Threshold for considering adjusted R-squared values statistically equivalent (default = 1e-4). Used when selecting between fits with similar goodness-of-fit.

Details

The algorithm implements the following decision logic:

1. Identifies the time of maximum observed concentration (Tmax)
2. Defines candidate terminal phases starting from the last 3 measurable concentrations
3. Iteratively evaluates longer time spans by including preceding data points
4. For each candidate phase:
   - Performs log-concentration vs. time linear regression
   - Requires negative regression slope (positive λ_z)
   - Calculates adjusted R-squared metric
5. Selects the optimal phase based on:
   - Highest adjusted R-squared value
   - When R-squared differences are < tolerance, selects the fit with more points
6. Validates final selection against R-squared threshold

Value

A list containing:

- `lambdaz`: Estimated terminal elimination rate constant (λ_z), or NA if no valid fit
- `UsedPoints`: Number of data points used in the optimal fit
- `adj.r.squared`: Adjusted R-squared value (R^2) of the optimal regression
- `message`: Character vector containing diagnostic messages or warnings

Author(s)

Zhonghui Huang

Examples

```
# Basic usage
time <- c(0.5, 1, 2, 4, 6, 8, 10)
conc <- c(12, 8, 5, 3, 2, 1.5, 1)
find_best_lambdaz(time, conc)

# With infusion route specification
find_best_lambdaz(time, conc, route = "bolus",duration=1)

# Custom threshold settings
find_best_lambdaz(time, conc, adj_r_squared_threshold = 0.8, tolerance = 0.001)
```

Fit_1cmpt_iv *Fit intravenous pharmacokinetic data to a one-compartment linear elimination model*

Description

Fits intravenous (IV) pharmacokinetic data to a one-compartment model with first-order elimination using the naive pooled data approach. Supports multiple estimation methods provided by nlmixr2 and can optionally return only predicted concentrations to support efficient simulation workflows.

Usage

```
Fit_1cmpt_iv(
  data,
  est.method,
  input.cl,
  input.vd,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

data	A data frame of IV pharmacokinetic data formatted for nlmixr2.
est.method	Estimation method to use in nlmixr2. Must be one of: "rxSolve", "nls", "nlm", "nlminb", or "focei".
input.cl	Initial estimate of clearance (CL).
input.vd	Initial estimate of volume of distribution (V).
input.add	Initial estimate of the additive residual error.
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.

`return.pred.only`
: Logical; if `TRUE`, returns a data frame with only predicted concentrations (`cp`) for all observations in the input data.

`...`
: Additional arguments passed to `nlmixr2()`, such as a user-defined `control = foceiControl(...)` or other control settings.

Value

If `return.pred.only = TRUE`, returns a `data.frame` with a single column `cp` (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Bolus_1CPT
# Run simulation
Fit_1cmpt_iv(
  data = dat,
  est.method = "rxSolve",
  input.cl = 4,
  input.vd = 70,
  input.add = 1
)
# Return only predicted concentrations
Fit_1cmpt_iv(
 data = dat,
 est.method = "rxSolve",
 input.cl = 4,
 input.vd = 70,
 input.add = 0,
 return.pred.only = TRUE
)
```

Fit_1cmpt_mm_iv *Fit intravenous pharmacokinetic data to a one-compartment model with Michaelis-Menten elimination*

Description

Fits intravenous (IV) pharmacokinetic data to a one-compartment model with Michaelis-Menten (nonlinear) elimination using the naive pooled data approach. Supports multiple estimation methods available in nlmixr2, and optionally returns only predicted concentrations to reduce memory use in simulation workflows.

Usage

```
Fit_1cmpt_mm_iv(
  data,
  est.method,
  input.vmax,
  input.km,
  input.vd,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

data	A data frame of IV pharmacokinetic data formatted for nlmixr2.
est.method	Estimation method to use in nlmixr2, one of: "rxSolve", "nls", "nlm", "nlminb", or "focei".
input.vmax	Initial estimate of the maximum elimination rate (Vmax).
input.km	Initial estimate of the Michaelis constant (Km).
input.vd	Initial estimate of the volume of distribution (V).
input.add	Initial estimate of the additive residual error.
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.
return.pred.only	Logical; if TRUE, returns a data frame with only predicted concentrations (cp) for all observations in the input data.
...	Optional arguments passed to nlmixr2(), such as a custom control = foceiControl(...) or other control objects.

Value

If return.pred.only = TRUE, returns a data.frame with a single column cp (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Bolus_1CPTMM
# Run simulation
Fit_1cmpt_mm_iv(
  data = dat,
  est.method = "rxSolve",
  input.vmax = 1000,
```

```
  input.km = 250,
  input.vd = 70,
  input.add = 10)

# Return only predicted concentrations
Fit_1cmpt_mm_iv(
  data = dat,
  est.method = "rxSolve",
  input.vmax = 1000,
  input.km = 250,
  input.vd = 70,
  input.add = 0,
  return.pred.only = TRUE
  )
```

`Fit_1cmpt_mm_oral` *Fit oral pharmacokinetic data to a one-compartment model with Michaelis-Menten elimination*

Description

Fits oral pharmacokinetic data to a one-compartment model with first-order absorption and Michaelis-Menten (nonlinear) elimination using the naive pooled data approach. Supports multiple estimation methods available in nlmixr2, and optionally returns only predicted concentrations to reduce memory use in simulation workflows.

Usage

```
Fit_1cmpt_mm_oral(
  data,
  est.method,
  input.ka,
  input.vmax,
  input.km,
  input.vd,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

`data`	A data frame of oral pharmacokinetic data formatted for nlmixr2.
`est.method`	Estimation method to use in nlmixr2, one of: `"rxSolve"`, `"nls"`, `"nlm"`, `"nlminb"`, or `"focei"`.
`input.ka`	Initial estimate of the absorption rate constant (ka).

input.vmax	Initial estimate of the maximum elimination rate (Vmax).
input.km	Initial estimate of the Michaelis constant (Km).
input.vd	Initial estimate of the volume of distribution (V).
input.add	Initial estimate of the additive residual error.
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.
return.pred.only	Logical; if TRUE, returns a data frame with only predicted concentrations (cp) for all observations in the input data.
...	Optional arguments passed to nlmixr2(), such as a custom control = foceiControl(...) or other control objects.

Value

If return.pred.only = TRUE, returns a data.frame with columns cp (predicted concentration) and DV (observed data). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Oral_1CPTMM
# Run simulation
Fit_1cmpt_mm_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
  input.vmax = 1000,
  input.km = 250,
  input.vd = 70,
  input.add = 10
)
# Return only predicted concentrations
Fit_1cmpt_mm_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
  input.vmax = 1000,
  input.km = 250,
  input.vd = 70,
  input.add = 10
)
```

`Fit_1cmpt_oral` *Fit oral pharmacokinetic data to a one-compartment linear elimination model*

Description

Fits oral pharmacokinetic data to a one-compartment model with first-order absorption and first-order elimination using the naive pooled data approach. Supports multiple estimation methods provided by nlmixr2 and can optionally return only predicted concentrations to support efficient simulation workflows.

Usage

```
Fit_1cmpt_oral(
  data,
  est.method,
  input.ka,
  input.cl,
  input.vd,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

`data`	A data frame containing oral pharmacokinetic data formatted for nlmixr2,
`est.method`	Estimation method to use in nlmixr2. Must be one of: `"rxSolve"`, `"nls"`, `"nlm"`, `"nlminb"`, or `"focei"`.
`input.ka`	Initial estimate of the absorption rate constant (ka).
`input.cl`	Initial estimate of clearance (CL).
`input.vd`	Initial estimate of volume of distribution (V).
`input.add`	Initial estimate of the additive residual error.
`ncores`	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
`return.pred.only`	Logical; if `TRUE`, returns a data frame with only predicted concentrations (`cp`) for all observations in the input data.
`...`	Additional arguments passed to `nlmixr2()`, such as a user-defined `control = foceiControl(...)` or other control settings.

Value

If `return.pred.only = TRUE`, returns a `data.frame` with a single column `cp` (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Oral_1CPT
# Run simulation
Fit_1cmpt_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
  input.cl = 4,
  input.vd = 70,
  input.add = 10
)
# Return only predicted concentrations
Fit_1cmpt_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
  input.cl = 4,
  input.vd = 70,
  input.add = 0,
  return.pred.only = TRUE
)
```

`Fit_2cmpt_iv` *Fit intravenous pharmacokinetic data to a two-compartment linear elimination model*

Description

Fits intravenous (IV) pharmacokinetic data to a two-compartment model with first-order elimination using the naive pooled data approach. Supports multiple estimation methods provided by nlmixr2 and can optionally return only predicted concentrations to support efficient simulation workflows.

Usage

```
Fit_2cmpt_iv(
  data,
  est.method,
  input.cl,
  input.vc2cmpt,
  input.vp2cmpt,
  input.q2cmpt,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
```

```
  ...
)
```

Arguments

data	A data frame containing IV pharmacokinetic data formatted for nlmixr2,
est.method	Estimation method to use in nlmixr2. Must be one of: "rxSolve", "nls", "nlm", "nlminb", or "focei".
input.cl	Initial estimate of clearance (CL).
input.vc2cmpt	Initial estimate of central volume of distribution (V1).
input.vp2cmpt	Initial estimate of peripheral volume of distribution (V2).
input.q2cmpt	Initial estimate of inter-compartmental clearance (Q).
input.add	Initial estimate of the additive residual error.
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.
return.pred.only	Logical; if TRUE, returns a data frame with only predicted concentrations (cp) for all observations in the input data.
...	Additional arguments passed to nlmixr2(), such as a user-defined control = foceiControl(...) or other control settings.

Value

If return.pred.only = TRUE, returns a data.frame with a single column cp (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Bolus_2CPT
# Run simulation
Fit_2cmpt_iv(
  data = dat,
  est.method = "rxSolve",
  input.cl = 4,
  input.vc2cmpt = 70,
  input.vp2cmpt = 40,
  input.q2cmpt = 4,
  input.add = 10
)
# Return only predicted concentrations
Fit_2cmpt_iv(
  data = dat,
  est.method = "rxSolve",
  input.cl = 4,
```

```
  input.vc2cmpt = 70,
  input.vp2cmpt = 40,
  input.q2cmpt = 4,
  input.add = 0,
  return.pred.only = TRUE
)
```

`Fit_2cmpt_oral` *Fit oral pharmacokinetic data to a two-compartment model*

Description

Fits oral pharmacokinetic data to a two-compartment model with first-order absorption and first-order elimination using the naive pooled data approach. Supports multiple estimation methods available in nlmixr2, and optionally returns only predicted concentrations to support simulation workflows.

Usage

```
Fit_2cmpt_oral(
  data,
  est.method,
  input.ka,
  input.cl,
  input.vc2cmpt,
  input.vp2cmpt,
  input.q2cmpt,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

`data`	A data frame containing oral pharmacokinetic data formatted for nlmixr2,
`est.method`	Estimation method to use in nlmixr2, one of: `"rxSolve"`, `"nls"`, `"nlm"`, `"nlminb"`, or `"focei"`.
`input.ka`	Initial estimate of the absorption rate constant (ka).
`input.cl`	Initial estimate of clearance (CL).
`input.vc2cmpt`	Initial estimate of central volume of distribution (V1).
`input.vp2cmpt`	Initial estimate of peripheral volume of distribution (V2).
`input.q2cmpt`	Initial estimate of inter-compartmental clearance (Q).
`input.add`	Initial estimate of the additive residual error.

ncores — Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.

return.pred.only — Logical; if `TRUE`, returns a data frame with only predicted concentrations (`cp`) for all observations in the input data.

... — Additional arguments passed to `nlmixr2()`, such as a user-defined `control = foceiControl(...)` or other control settings.

Value

If `return.pred.only = TRUE`, returns a `data.frame` with a single column `cp` (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Oral_2CPT[Oral_2CPT$ID<11,]
# Run simulation
Fit_2cmpt_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
  input.cl = 4,
  input.vc2cmpt = 70,
  input.vp2cmpt = 40,
  input.q2cmpt = 10,
  input.add = 10
)
```

Fit_3cmpt_iv — *Fit intravenous pharmacokinetic data to a three-compartment linear elimination model*

Description

Fits intravenous (IV) pharmacokinetic data to a three-compartment model with linear (first-order) elimination using the naive pooled data approach. Supports multiple estimation methods provided by nlmixr2 and can optionally return only predicted concentrations to support efficient simulation workflows.

Usage

```
Fit_3cmpt_iv(
  data,
  est.method,
  input.cl,
  input.vc3cmpt,
  input.vp3cmpt,
  input.vp23cmpt,
  input.q3cmpt,
  input.q23cmpt,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

`data`	A data frame containing IV pharmacokinetic data formatted for nlmixr2.
`est.method`	Estimation method to use in nlmixr2. Must be one of: `"rxSolve"`, `"nls"`, `"nlm"`, `"nlminb"`, or `"focei"`.
`input.cl`	Initial estimate of clearance (CL).
`input.vc3cmpt`	Initial estimate of central volume of distribution (V1).
`input.vp3cmpt`	Initial estimate of first peripheral volume of distribution (V2).
`input.vp23cmpt`	Initial estimate of second peripheral volume of distribution (V3).
`input.q3cmpt`	Initial estimate of first inter-compartmental clearance (Q1).
`input.q23cmpt`	Initial estimate of second inter-compartmental clearance (Q2).
`input.add`	Initial estimate of the additive residual error.
`ncores`	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
`return.pred.only`	Logical; if `TRUE`, returns a data frame with only predicted concentrations (`cp`) for all observations in the input data.
`...`	Additional arguments passed to `nlmixr2()`, such as a user-defined `control = foceiControl(...)` or other control settings.

Value

If `return.pred.only = TRUE`, returns a `data.frame` with a single column `cp` (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Bolus_2CPT
# Run simulation
Fit_3cmpt_iv(
  data = dat,
  est.method = "rxSolve",
  input.cl = 4,
  input.vc3cmpt = 70,
  input.vp3cmpt = 35,
  input.vp23cmpt = 5,
  input.q3cmpt = 4,
  input.q23cmpt = 4,
  input.add = 10
)
# Return only predicted concentrations
Fit_3cmpt_iv(
  data = dat,
  est.method = "rxSolve",
  input.cl = 4,
  input.vc3cmpt = 70,
  input.vp3cmpt = 35,
  input.vp23cmpt = 35,
  input.q3cmpt = 4,
  input.q23cmpt = 4,
  input.add = 10,
  return.pred.only = TRUE
)
```

Fit_3cmpt_oral *Fit oral pharmacokinetic data to a three-compartment linear elimination model*

Description

Fits oral pharmacokinetic data to a three-compartment model with first-order absorption and first-order elimination using the naive pooled data approach. Supports multiple estimation methods provided by nlmixr2 and can optionally return only predicted concentrations to support efficient simulation workflows.

Usage

```
Fit_3cmpt_oral(
  data,
  est.method,
  input.ka,
  input.cl,
  input.vc3cmpt,
  input.vp3cmpt,
```

```
  input.vp23cmpt,
  input.q3cmpt,
  input.q23cmpt,
  input.add,
  ncores = 2,
  return.pred.only = FALSE,
  ...
)
```

Arguments

data	A data frame containing oral pharmacokinetic data formatted for nlmixr2.
est.method	Estimation method to use in nlmixr2. Must be one of: "rxSolve", "nls", "nlm", "nlminb", or "focei".
input.ka	Initial estimate of the absorption rate constant (ka).
input.cl	Initial estimate of clearance (CL).
input.vc3cmpt	Initial estimate of central volume of distribution (V1).
input.vp3cmpt	Initial estimate of first peripheral volume of distribution (V2).
input.vp23cmpt	Initial estimate of second peripheral volume of distribution (V3).
input.q3cmpt	Initial estimate of first inter-compartmental clearance (Q1).
input.q23cmpt	Initial estimate of second inter-compartmental clearance (Q2).
input.add	Initial estimate of the additive residual error.
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.
return.pred.only	Logical; if TRUE, returns a data frame with only predicted concentrations (cp) for all observations in the input data.
...	Additional arguments passed to nlmixr2(), such as a user-defined control = foceiControl(...) or other control settings.

Value

If return.pred.only = TRUE, returns a data.frame with a single column cp (predicted concentrations). Otherwise, returns a fitted model object produced by nlmixr2.

Author(s)

Zhonghui Huang

Examples

```
dat <- Oral_2CPT[Oral_2CPT$ID<11,]
Fit_3cmpt_oral(
  data = dat,
  est.method = "rxSolve",
  input.ka = 1,
```

```
  input.cl = 4,
  input.vc3cmpt = 70,
  input.vp3cmpt = 35,
  input.vp23cmpt = 35,
  input.q3cmpt = 4,
  input.q23cmpt = 4,
  input.add = 10
)
```

`force_find_lambdaz` *Forceful estimation of terminal slope*

Description

Estimates the terminal elimination rate constant (lambda_z) of a pharmacokinetic profile. The function first attempts to use the `find_best_lambdaz` method. If no valid estimate is obtained, it falls back to a simplified log-linear regression using progressively fewer data points to enforce a negative slope.

Usage

```
force_find_lambdaz(time, conc, ...)
```

Arguments

`time`	Numeric vector of time points.
`conc`	Numeric vector of concentration values corresponding to time.
`...`	Additional arguments passed to find_best_lambdaz (e.g., nlastpoints).

Details

This function implements a two-step strategy to ensure estimation of the terminal elimination slope:

- First, it applies `find_best_lambdaz` to automatically select the best fitting terminal phase segment based on adjusted R-squared optimization.
- If `find_best_lambdaz` fails (e.g., limited data), the function forcibly fits simplified linear models using progressively fewer points (starting from n-1 down to 2) until a negative slope is identified. In fallback mode, adjusted R-squared is not considered.

Value

A list containing:

- lambdaz: Estimated terminal elimination rate constant (1/time)
- intercept: Intercept of the log-linear regression, used to extrapolate concentration at time zero
- method: Method used (`find_best_lambdaz` or fallback regression)

- UsedPoints: Number of time-concentration points used for estimation
- adj.r.squared: Adjusted R-squared (available only when using `find_best_lambdaz`)
- message: Diagnostic message summarizing the outcome
- slopefit: Fitted linear model object

Author(s)

Zhonghui Huang

See Also

find_best_lambdaz

Examples

```
time <- c(0.5, 1, 2, 4, 6, 8, 10)
conc <- c(12, 8, 5, 3, 2, 1.5, 1)
force_find_lambdaz(time, conc)
```

---

get_hf *Estimate half-life from pooled pharmacokinetic data*

Description

Estimates the terminal half-life of a drug using pooled pharmacokinetic data. The method supports analysis based on first-dose, repeated-dose, or combined dosing profiles.

Usage

```
get_hf(dat, dose_type = "first_dose", pooled = NULL, verbose = TRUE, ...)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.
`dose_type`	Specifies the dosing context of the pharmacokinetic observations. Classified as: • first_dose: observations occur only after the initial administration • repeated_doses: observations occur only after multiple administrations or at steady state • combined_doses: observations include both first-dose and repeated-dose intervals
`pooled`	Optional pooled data object generated by get_pooled_data. If not provided, pooled data are automatically created using the input dataset.

verbose	Logical; if TRUE (default), progress and completion messages are printed to the console. Set to FALSE to suppress all informational messages, for example when running automated scripts or tests.
...	Additional arguments passed to bin.time for pooling operations or to find_best_lambdaz for elimination slope estimation.

Details

The function estimates terminal half-life using the following procedure:

- Generate or use existing pooled pharmacokinetic data
- Identify the terminal phase using elimination slope estimation
- Calculate the terminal elimination rate constant (lambda_z)
- Derive half-life using:

$$t_{1/2} = \frac{\log(2)}{\lambda_z}$$

Value

A list containing individual and median half-life estimates for first-dose, repeated-dose, and combined dosing profiles.

Author(s)

Zhonghui Huang

See Also

get_pooled_data, bin.time, find_best_lambdaz

Examples

```
dat <- Bolus_1CPT
dat <- processData(dat)$dat
get_hf(dat, dose_type = "combined_doses")
```

`get_pooled_data` *Generate pooled data for pharmacokinetic analysis*

Description

Processes pharmacokinetic data and produces pooled datasets according to the dosing context. Data can be grouped based on first dose, repeated dosing, or a combination of both, with control over binning and time alignment.

Usage

```
get_pooled_data(
  dat,
  dose_type = c("first_dose", "repeated_doses", "combined_doses"),
  pooled_ctrl = pooled_control()
)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.
`dose_type`	Specifies the dosing context of the pharmacokinetic observations. Classified as: • first_dose: data include only observations following the initial administration • repeated_doses: data include only observations during repeated or steady-state dosing • combined_doses: data include observations from both first-dose and repeated-dose intervals
`pooled_ctrl`	A list of control parameters created by 'pooled_control', including settings for binning and time rounding.

Details

For repeated-doses and combined-doses classifications, the most common interdose interval is identified from dosing records and used to determine whether observations fall within the relevant interval. If tad_rounding is TRUE, both time after dose and dosing interval are rounded before comparison.

Value

A list containing pooled pharmacokinetic datasets depending on the specified dose type:

- datpooled_fd: pooled data for first-dose observations
- datpooled_efd: pooled data for repeated dosing
- datpooled_all: pooled data combining first-dose and repeated-dose observations

Author(s)

Zhonghui Huang

See Also

pooled_control, trimmed_geom_mean

Examples

```
dat <- processData(Bolus_1CPT)$dat
get_pooled_data(dat, dose_type = "combined_doses")
```

---

getnca *Perform non-compartmental pharmacokinetic analysis*

Description

Calculates key pharmacokinetic parameters using non-compartmental methods for both intravenous and oral administration data.

Usage

```
getnca(
  x,
  y,
  dose = 1,
  trapezoidal.rule = c("linear_up_log_down", "linear"),
  ss = 0,
  duration = NULL,
  nlastpoints = 3,
  slope.method = c("bestfitforce", "bestfit"),
  route = c("bolus", "oral", "infusion")
)
```

Arguments

x	Numeric vector of observation times.
y	Numeric vector of drug concentration measurements.
dose	Administered dose (default = 1).
trapezoidal.rule	Method for AUC calculation: • "linear" - Linear trapezoidal method (default) • "linear_up_log_down" - Linear-up/log-down method (linear for ascending concentrations, logarithmic for descending)
ss	Steady-state flag:

- 0 - Use extrapolated $AUC_{0\to\infty}$ for clearance (default)
- 1 - Use observed $AUC_{0\to\text{last}}$ for clearance

`duration` Infusion duration (required if `route = "infusion"`).

`nlastpoints` Number of terminal points for slope estimation (default = 3).

`slope.method` Method for estimating terminal slope (λ_z):

- `"bestfitforce"` - Force estimation using decreasing number of terminal points if best-fit fails (default)
- `"bestfit"` - Use automated best-fit selection based on adjusted R-squared

`route` Administration route:

- `"bolus"` - Intravenous bolus (default)
- `"oral"` - Oral administration
- `"infusion"` - Intravenous infusion

Value

A list containing:

- cl - Clearance (CL), calculated as Dose/AUC
- vz - volume of distribution (Vz), calculated as CL / lambdaz
- half_life - Terminal elimination half-life, computed as ln(2) / lambdaz
- auct - Area under the concentration–time curve from time 0 to last measurable concentration
- auc0_inf - AUC extrapolated to infinity
- C_last - Last non-zero measurable concentration
- lambdaz - Terminal elimination rate constant
- aumc_0_t - Area under the first moment curve from time 0 to last measurable concentration
- aumc_0_inf - AUMC extrapolated to infinity
- used_points - Number of time–concentration points used to estimate lambdaz
- adj.r.squared - Adjusted R-squared of the terminal phase regression
- messages - Warning or diagnostic messages returned during the calculation

Examples

```
# IV bolus example
dat <- data.frame(TIME = c(0.5, 1, 2, 4, 6, 8),
                  DV = c(12, 8, 5, 3, 2, 1))
getnca(x = dat$TIME, y = dat$DV, dose = 1)

# IV infusion example
dat <- data.frame(TIME = c(0.5, 1, 2, 4, 6, 8),
                  DV = c(2, 8, 5, 3, 2, 1))
getnca(x = dat$TIME, y = dat$DV, dose = 1, route = "infusion", duration = 1)

# Oral administration example
dat <- data.frame(TIME = c(0, 1, 2, 4, 6, 8),
                  DV = c(0, 9, 12, 8, 6, 2))
getnca(x = dat$TIME, y = dat$DV, route = "oral")
```

getOmegas — *Generate ETA variance and covariance table*

Description

This function constructs a combined table containing:

- **ETA variances** (e.g., `eta.cl`, `eta.vc`), which represent inter-individual variability (IIV) in pharmacokinetic parameters;
- **Derived covariances** (e.g., `cov.eta_cl_vc`) computed from ETA variances and assumed pairwise correlations.

Usage

```
getOmegas()
```

Details

ETA variances are initialized to 0.1 by default. Correlations within defined omega blocks (block 1: `eta.vmax`, `eta.km`; block 2: `eta.cl`, `eta.vc`, `eta.vp`, `eta.vp2`, `eta.q`, `eta.q2`) are assumed to be 0.1 and used to compute covariances as:

$$Cov(i,j) = sqrt(Var_i) * sqrt(Var_j) * Corr(i,j)$$

The resulting output format aligns with `Recommended_initial_estimates`.

Value

A `data.frame` with columns: `Parameters`, `Methods`, and `Values`.

Examples

```
getOmegas()
```

getPPKinits — *Automated pipeline for generating initial estimates in population PK models*

Description

Provides a unified and fully automated workflow to generate initial pharmacokinetic and residual variability parameters for population PK models using concentration–time data from bolus, infusion, or oral administration.

Usage

```
getPPKinits(dat, control = initsControl(), ncores = 2, verbose = TRUE)
```

Arguments

`dat`	A data frame containing pharmacokinetic records in standard nlmixr2 format, including ID, TIME, EVID, and DV.
`control`	A list created by `initsControl()` specifying configuration for pooling, non-compartmental analysis, steady-state detection, fallback rules, statistical model components, and parameter selection metrics.
`ncores`	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
`verbose`	Logical (default = TRUE); when TRUE, displays key progress messages and stepwise updates during the initialization process. When FALSE, the function runs quietly without printing intermediate information.

Details

The pipeline integrates four model-informed analytical components applied to raw or pooled concentration–time profiles:

1. Adaptive single-point methods
2. Naive pooled graphic methods
3. Naive pooled non-compartmental analysis (NCA) with optional Wagner–Nelson Ka calculation for oral dosing
4. Parameter sweeping across one-, two-, three-compartment and Michaelis–Menten models

In addition to structural PK parameters, the framework also initializes statistical model components:

- Inter-individual variability (IIV): pragmatic fixed ω^2 values are assigned to random effects.
- Residual unexplained variability (RUV): estimated either using a data-driven method based on trimmed residual summaries or a fixed-fraction approach consistent with NONMEM User Guide recommendations.
- Model applicability: the automated and model-informed strategy generates robust initial values suitable for both linear and nonlinear mixed-effects pharmacokinetic models.

Value

An object of class `getPPKinits` containing recommended initial parameter estimates, intermediate results, and computation diagnostics.

Author(s)

Zhonghui Huang

See Also

`initsControl`, `run_single_point`, `run_graphcal`, `run_pooled_nca`, `sim_sens_1cmpt_mm`, `sim_sens_2cmpt`, `sim_sens_3cmpt`, `metrics`.

Examples

```
getPPKinits(pheno_sd[pheno_sd$ID<11,])
```

`getsigma` *Compute overall residual variability from elimination phase*

Description

Applies `getsigmas` to each individual and dose group after filtering observation records (EVID == 0), and calculates trimmed mean estimates of additive and proportional residual variability.

Usage

```
getsigma(df, nlastpoints = 3, sigma_trim = 0.05)
```

Arguments

`df`	Full pharmacokinetic dataset containing at least the columns: EVID, ID, TIME, DV, and routeobs.
`nlastpoints`	Number of terminal points used for elimination phase regression in each group (passed to getsigmas).
`sigma_trim`	Trimming proportion used when calculating trimmed means of residual standard deviations. Default is 0.05.

Details

The function groups the dataset by subject and dose occasion, applies elimination-phase residual analysis using `getsigmas`, and summarizes the individual residual standard deviations by their trimmed means. This provides population-level estimates of additive and proportional residual unexplained variability (RUV).

Value

A list containing:

- summary: Named list with trimmed mean values of additive and proportional residual variability
- full: Data frame with residual estimates for each individual-dose group

Author(s)

Zhonghui Huang

See Also

getsigmas

Examples

```
dat <- Bolus_1CPT
dat <- processData(dat)$dat
getsigma(dat)
```

`getsigmas` *Estimate individual-level residual error from the elimination phase*

Description

Performs log-linear regression on the elimination phase of a single individual's or one group's pharmacokinetic concentration–time data to estimate additive and proportional residual standard deviations.

Usage

```
getsigmas(group_df, nlastpoints = 3)
```

Arguments

`group_df`	A data frame for a single group (e.g., one subject or dose), containing columns: EVID (event ID), DV (observed concentration), TIME (time after dose), and routeobs (administration route).
`nlastpoints`	Integer specifying the number of terminal data points used for regression.

Details

Residuals are computed from individual-predicted concentrations (IPRED) and observed concentrations (DV) using the following definitions:

$$\sigma_{add} = \sqrt{Var(C_{obs} - C_{pred})}$$

$$\sigma_{prop} = \sqrt{Var\left(\frac{C_{obs}}{C_{pred}} - 1\right)}$$

where C_{obs} is the observed concentration and C_{pred} is the model-predicted concentration obtained by back-transformation of the log-linear regression. The additive residual standard deviation (σ_{add}) and proportional residual standard deviation (σ_{prop}) are calculated per individual.

Value

A tibble with the following columns:

- intercept: Intercept of the log-linear regression line
- slope: Estimate of the terminal elimination rate constant
- residual_sd_additive: Standard deviation of additive residuals
- residual_sd_proportional: Standard deviation of proportional residuals

Author(s)

Zhonghui Huang

Examples

```
dat <- Bolus_1CPT
dat <- processData(dat)$dat
getsigmas(dat[dat$ID == 1 & dat$dose_number == 1 & dat$resetflag == 1 &
              dat$EVID == 0, ])
```

---

graphcal_iv *Graphical calculation of clearance and volume of distribution (IV route)*

Description

Estimates clearance (CL), volume of distribution (Vd), terminal slope (lambdaz), and extrapolated concentration at time zero (C0exp) from intravenous pharmacokinetic data using graphical methods.

Usage

```
graphcal_iv(dat, dose = 1, ...)
```

Arguments

`dat`	A data frame containing TIME (time after dosing) and DV (observed concentration).
`dose`	Administered dose amount. Defaults to 1.
`...`	Additional arguments passed to `force_find_lambdaz()`.

Details

Terminal slope (lambdaz) is estimated using `force_find_lambdaz()`, which applies an automated phase selection strategy with fallback regression when required.

Value

A list containing graphical estimates of CL, Vd, lambda_z, and C0exp.

Author(s)

Zhonghui Huang

See Also

`force_find_lambdaz`

Examples

```
dat <- data.frame(TIME = c(0.5, 1, 2, 4, 6, 8, 10),
                  DV = c(12, 8, 5, 3, 2, 1.5, 1))
graphcal_iv(dat, dose = 100)
```

`graphcal_oral` *Graphical calculation of pharmacokinetic parameters for oral administration*

Description

Estimates key pharmacokinetic parameters from oral concentration–time data using graphical methods, including absorption rate constant (ka), elimination rate constant (kel), terminal slope, extrapolated concentration (C0exp), apparent volume of distribution (Vd/F), and clearance (Cl/F).

Usage

```
graphcal_oral(dat, dose = 1, ...)
```

Arguments

`dat`	A data frame containing TIME (time after dosing) and DV (observed concentration).
`dose`	Administered dose amount. Defaults to 1.
`...`	Additional arguments passed to `find_best_lambdaz()`.

Details

The terminal slope (lambdaz) is estimated using `force_find_lambdaz()`. The apparent volume of distribution and clearance are computed using the following relationships:

$$Vd/F = \frac{Dose \times ka}{C_0 \times (ka - kel)}$$

$$Cl/F = kel \times Vd/F$$

where ka is estimated from the absorption phase.

Value

A list containing graphical estimates of ka, kel, lambda_z, C0exp, Vd/F, and Cl/F.

Author(s)

Zhonghui Huang

See Also

find_best_lambdaz

Examples

```
dat <- data.frame(TIME = c(0.5, 1, 2, 4, 6, 8, 10),
                  DV = c(1, 2, 5, 3, 2, 1.5, 1))
graphcal_oral(dat, dose = 100, route = "oral")
```

---

hybrid_eval_perf_1cmpt

Generate Unique Mixture Parameter Grid (with Deduplication and NA Removal)

Description

Constructs a grid of all combinations of ka, cl, and vd parameters from different sources (e.g., simpcal, graph, NCA methods), and removes combinations that are redundant based on relative tolerance. Only oral routes consider ka value for deduplication. Any parameter value set that includes NA is removed up front.

Usage

```
hybrid_eval_perf_1cmpt(
  route = "bolus",
  dat,
  sp_out_ka,
  sp_out_cl,
  sp_out_vd,
  graph_out_ka,
  graph_out_cl,
  graph_out_vd,
  nca_fd_ka,
  nca_fd_cl,
  nca_fd_vd,
  nca_efd_ka,
  nca_efd_cl,
  nca_efd_vd,
  nca_all_ka,
  nca_all_cl,
  nca_all_vd,
  ncores = 2,
  verbose = TRUE
)
```

Arguments

route	Route of administration. Must be one of bolus, oral, or infusion.
dat	A data.frame containing PK data with columns such as EVID and DV.
sp_out_ka	Numeric; ka estimated from adaptive single-point methods.
sp_out_cl	Numeric; clearance estimated from adaptive single-point methods.
sp_out_vd	Numeric; volume of distribution estimated from adaptive single-point methods.
graph_out_ka	Numeric; ka estimated from naive pooled graphic methods.
graph_out_cl	Numeric; clearance estimated from naive pooled graphic methods.
graph_out_vd	Numeric; volume of distribution estimated from naive pooled graphic methods.
nca_fd_ka	Numeric; ka estimated from naive pooled NCA using first-dose data.
nca_fd_cl	Numeric; clearance estimated from naive pooled NCA using first-dose data.
nca_fd_vd	Numeric; volume of distribution estimated from naive pooled NCA using first-dose data.
nca_efd_ka	Numeric; ka estimated from naive pooled NCA using repeated-dose data.
nca_efd_cl	Numeric; clearance estimated from naive pooled NCA using repeated-dose data.
nca_efd_vd	Numeric; volume of distribution estimated from naive pooled NCA using repeated-dose data.
nca_all_ka	Numeric; ka estimated from naive pooled NCA using combined first- and repeated-dose data.
nca_all_cl	Numeric; clearance estimated from naive pooled NCA using combined first- and repeated-dose data.
nca_all_vd	Numeric; volume of distribution estimated from naive pooled NCA using combined first- and repeated-dose data.
ncores	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
verbose	Logical; if TRUE (default), displays a textual progress bar during model evaluation using the 'progressr' package. Set to FALSE to run silently without showing progress information, which is recommended for automated analyses or CRAN checks.

Value

A `data.frame` of unique parameter combinations with source labels and values.

Examples

```
dat <- Oral_1CPT[Oral_1CPT$ID<11,]
# Example parameter estimates from different methods
sp_out_ka <- 1.2; sp_out_cl <- 3.5; sp_out_vd <- 50
graph_out_ka <- 1.1; graph_out_cl <- 3.6; graph_out_vd <- 52
nca_fd_ka <- 1.3; nca_fd_cl <- 3.4; nca_fd_vd <- 49
nca_efd_ka <- NA;  nca_efd_cl <- NA;  nca_efd_vd <- NA
nca_all_ka <- 1.25; nca_all_cl <- 3.55; nca_all_vd <- 51
```

```
# Run hybrid evaluation
 hybrid_eval_perf_1cmpt(
  route = "oral",
  dat = dat,
  sp_out_ka = sp_out_ka, sp_out_cl = sp_out_cl, sp_out_vd = sp_out_vd,
  graph_out_ka = graph_out_ka, graph_out_cl = graph_out_cl, graph_out_vd = graph_out_vd,
  nca_fd_ka = nca_fd_ka, nca_fd_cl = nca_fd_cl, nca_fd_vd = nca_fd_vd,
  nca_efd_ka = nca_efd_ka, nca_efd_cl = nca_efd_cl, nca_efd_vd = nca_efd_vd,
  nca_all_ka = nca_all_ka, nca_all_cl = nca_all_cl, nca_all_vd = nca_all_vd,
  verbose = FALSE
)
```

`initsControl` *Create full control list for initial parameter estimation*

Description

Aggregates modular control functions into a structured list for use in population pharmacokinetic parameter initialization.

Usage

```
initsControl(
  ss.control = ss_control(),
  pooled.control = pooled_control(),
  nca.control = nca_control(),
  fallback.control = fallback_control(),
  selmetrics = "rRMSE2",
  hybrid.base = TRUE,
  preferNCA = TRUE
)
```

Arguments

`ss.control`	A control list consistent with the structure returned by ss_control().
`pooled.control`	A control list consistent with the structure returned by pooled_control().
`nca.control`	A control list consistent with the structure returned by nca_control().
`fallback.control`	A control list consistent with the structure returned by fallback_control().
`selmetrics`	A character string or vector specifying model performance metrics to evaluate. Must be one or more of "APE", "MAE", "MAPE", "RMSE", "rRMSE1", or "rRMSE2". Default is "rRMSE2".
`hybrid.base`	Logical. If TRUE, enables hybrid evaluation mode in which model performance is assessed using mixed parameter combinations across methods. If FALSE, each method is evaluated independently. Default is TRUE.

preferNCA — Logical. If TRUE and selmetrics equals "rRMSE2", the lowest rRMSE2 is selected first. If the best-performing method is not NCA-based, the function then checks whether an NCA-based method offers a lower rRMSE1. If so, the NCA method is selected. Default is TRUE.

Value

A named list combining all control modules for parameter estimation.

See Also

ss_control, pooled_control, nca_control, fallback_control

Examples

```
initsControl(
  pooled.control = pooled_control(nbins = 8),
  fallback.control = fallback_control(
    sigma_method_additive = "fixed_fraction"
  )
)
```

---

is_ss *Determine steady state for pharmacokinetic observations*

Description

Evaluates whether pharmacokinetic observations have reached steady state based on user-defined control settings. The classification can be based on a fixed number of doses, the number of half-lives relative to the dosing interval, or a combination of both criteria.

Usage

```
is_ss(df, ssctrl = ss_control(), half_life = NA)
```

Arguments

df — A data frame containing pharmacokinetic data. It should include columns for ID, EVID, SSflag, TIME, AMT, and tad.

ssctrl — A control list consistent with the structure returned by ss_control(). It specifies the method and thresholds for steady-state evaluation.

half_life — Numeric value representing the drug half-life. Required when the method in ss_control() is based on half-life or uses a combined approach.

Details

The function determines steady state by examining each observation in relation to prior dosing history. The required number of doses is calculated based on the specified method in ss_control(). Observation times are evaluated to confirm that dose interval and dose amount variability fall within acceptable limits and that the time after dose is within the most recent dosing interval. Observations manually marked as steady state using SSflag are also recognized as steady state.

Value

A data frame with added columns indicating steady-state status, the dosing interval for steady-state observations, and the method used to classify steady state.

See Also

ss_control()

Examples

```
dat <- pheno_sd
dat <- processData(dat)$dat
out <- is_ss(df = dat)
out[out$SteadyState == TRUE & !is.na(out$SteadyState),
    c("ID", "TIME", "DV", "EVID", "SteadyState")]
```

---

ka_calculation_md *Calculate absorption rate constant (ka) in a multiple-dose one-compartment model*

Description

This estimates the absorption rate constant in a multiple-dose oral model using first-order pharmacokinetics.

Usage

```
ka_calculation_md(cl, ke, t, Ct, Fbio = 1, Dose, tau)
```

Arguments

`cl`	Numeric. Clearance of the drug (in L/hr).
`ke`	Numeric. Elimination rate constant (in 1/hr).
`t`	Numeric. Time after the last dose (in hours) at which the concentration is measured.
`Ct`	Numeric. Observed concentration of the drug at time `t` (in mg/L).
`Fbio`	Numeric. Bioavailability fraction (default = 1, meaning 100% bioavailability).
`Dose`	Numeric. Administered dose of the drug (in mg).
`tau`	Numeric. Dosing interval (in hours) between successive doses.

Details

The value of ka is obtained numerically using the uniroot unction by solving the following equation:

$$Ct = \frac{Fbio \cdot Dose \cdot ka}{Vd \cdot (ka - ke)} \left(\frac{e^{-ke \cdot t}}{1 - e^{-ke \cdot \tau}} - \frac{e^{-ka \cdot t}}{1 - e^{-ka \cdot \tau}} \right)$$

ka is estimated using `uniroot()`, which solves for the root of the residual function (predicted Ct - observed Ct) within a bounded interval (ka > ke and ka <= 1000)

Value

A list containing the following components:

`ka`	The calculated absorption rate constant.
`full_solution`	The full solution object returned by the `uniroot()` function, which includes additional details about the root-finding process.

Author(s)

Zhonghui Huang

Examples

```
# Example from Oral_1CPT dataset (ID = 1, 5th dose, t = 2 h)
ka_calculation_md(cl = 4, ke = 0.057, t = 2, Ct = 852, Dose = 60000, tau = 24)
```

---

ka_calculation_sd *Estimate absorption rate constant in a one-compartment oral model*

Description

This estimates the absorption rate constant in a single-dose oral model using first-order pharmacokinetics.

Usage

```
ka_calculation_sd(cl, ke, t, Ct, Fbio = 1, Dose)
```

Arguments

`cl`	Numeric. Clearance of the drug.
`ke`	Numeric. Elimination rate constant.
`t`	Numeric. Time after administration.
`Ct`	Numeric. Observed plasma concentration at time t.
`Fbio`	Numeric. Absolute bioavailability fraction. Default is 1.
`Dose`	Numeric. Administered oral dose.

Details

The model assumes a one-compartment structure with first-order absorption and first-order elimination.

The concentration-time relationship is:

$$Ct = \frac{Fbio \cdot Dose \cdot ka}{Vd \cdot (ka - ke)} \left(e^{-ke \cdot t} - e^{-ka \cdot t}\right)$$

where the volume of distribution is defined as:

$$Vd = \frac{cl}{ke}$$

ka is estimated using `uniroot()`, which solves for the root of the residual function (predicted Ct - observed Ct) within a bounded interval (ka > ke and ka <= 1000)

Value

A list containing:

`ka`	Estimated absorption rate constant.
`full_solution`	The full result object returned by the root-finding process.
`message`	A character string indicating the status of the estimation or any warnings.

Author(s)

Zhonghui Huang

Examples

```
# Example from Oral_1CPT dataset (ID = 1, 1st dose, t = 0.5 h)
ka_calculation_sd(cl = 3.62, ke = 0.0556, t = 0.5, Ct = 310.6, Dose = 60000)
```

---

ka_wanger_nelson *Calculate the absorption rate constant using the Wagner-Nelson method*

Description

Calculates absorption rate constant using the Wagner–Nelson method for single-dose extravascular pharmacokinetics.

Usage

```
ka_wanger_nelson(dat, nca.out = NULL)
```

Arguments

`dat`	A data frame containing two columns: 'TIME' for sampling time points and 'DV' for observed plasma drug concentrations.
`nca.out`	Optional object containing results from a previous noncompartmental analysis. It must include 'auc0_inf' for the area under the concentration-time curve extrapolated to infinity and 'lambdaz' for the terminal elimination rate constant. If not provided, the function calls 'getnca' internally using the input data.

Details

The Wagner-Nelson method estimates the fraction of drug absorbed over time based on the principle of mass balance, where the unabsorbed fraction is quantified as the proportion of the administered dose that has not yet entered systemic circulation. A linear regression is applied to the natural logarithm of the unabsorbed fraction versus time, and the negative slope of this regression corresponds to the first-order absorption rate constant 'ka'.

Key assumptions:

- Single-dose oral or extravascular administration
- First-order absorption and first-order elimination
- Linear pharmacokinetics with 'ka' greater than 'ke'

Computational steps:

- AUC is calculated using trapezoidal integration.
- The fraction absorbed is calculated from AUC and the terminal elimination rate constant.
- The remaining fraction is transformed using the natural logarithm.
- Linear regression of log(remaining fraction) against time yields 'ka'.

Value

A list containing:

- ka: Estimated absorption rate constant
- dat_out_wanger_nelson: Input data frame augmented with calculated pharmacokinetic variables including cumulative AUC, fraction absorbed, and fraction remaining

Author(s)

Zhonghui Huang

References

Wagner JG and Nelson E (1963). Percent absorbed time plots derived from blood level and/or urinary excretion data. Journal of Pharmaceutical Sciences, 52(6), 610-611.

Examples

```
# Simulated one-compartment oral absorption data
Dose <- 100
Fbio <- 1
Vd   <- 70
CL   <- 4
ka   <- 1.2
ke   <- CL / Vd
t  <- seq(0.5, 8, by = 0.5)
Ct <- (Fbio * Dose * ka) / (Vd * (ka - ke)) * (exp(-ke * t) - exp(-ka * t))

dat <- data.frame(TIME = t, DV = Ct)

ka_wanger_nelson(dat)
```

---

mark_dose_number *Mark dose number*

Description

Assigns sequential dose numbers based on dosing events (EVID) within each subject.

Usage

```
mark_dose_number(dat)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.

Value

A modified data frame with an added column named dose_number, indicating the sequential dose count within each subject and reset group.

Author(s)

Zhonghui Huang

Examples

```
mark_dose_number(Bolus_1CPT)
mark_dose_number(Infusion_1CPT)
mark_dose_number(Oral_1CPT)
```

`metrics.` *Calculate metrics for model predictive performance evaluation*

Description

Computes common error metrics that quantify the predictive performance of pharmacometric models by comparing predicted (pred.x) and observed (obs.y) concentration values.

Usage

```
metrics.(pred.x, obs.y)
```

Arguments

`pred.x`	Numeric vector of model-predicted values.
`obs.y`	Numeric vector of corresponding observed values.

Details

The function stops with an error if pred.x and obs.y have unequal lengths. The following metrics are calculated:

$$APE = \sum |pred.x - obs.y|$$

Absolute prediction error (APE) is the sum of absolute differences.

$$MAE = \frac{1}{n} \sum |pred.x - obs.y|$$

Mean absolute error (MAE) expresses the average absolute deviation.

$$MAPE = \frac{100}{n} \sum \left| \frac{pred.x - obs.y}{obs.y} \right|$$

Mean absolute percentage error (MAPE) normalizes the error by observed values.

$$RMSE = \sqrt{\frac{1}{n} \sum (pred.x - obs.y)^2}$$

Root mean squared error (RMSE) penalizes larger deviations.

$$rRMSE1 = \frac{RMSE}{\bar{obs.y}} \times 100$$

Relative RMSE type 1 is the RMSE normalized by the mean observed value.

$$rRMSE2 = 100 \times \sqrt{\frac{1}{n} \sum \left(\frac{pred.x - obs.y}{(pred.x + obs.y)/2} \right)^2}$$

Relative RMSE type 2 is symmetric and normalizes by the mean of each predicted–observed pair.

Value

A numeric vector with named elements:

- APE: absolute prediction error
- MAE: mean absolute error
- MAPE: mean absolute percentage error
- RMSE: root mean squared error
- rRMSE1: relative RMSE (type 1)
- rRMSE2: relative RMSE (type 2)

Examples

```
obs.y  <- rnorm(100, mean = 100, sd = 10)
pred.x <- obs.y + rnorm(100, mean = 0, sd = 5)
metrics.(pred.x = pred.x, obs.y = obs.y)
```

---

nca_control *Control options for non-compartmental analysis*

Description

Control options for non-compartmental analysis (NCA)

Usage

```
nca_control(
  trapezoidal.rule = c("linear_up_log_down", "linear"),
  duration = NULL,
  nlastpoints = 3,
  slope.method = "bestfitforce"
)
```

Arguments

trapezoidal.rule	Character. Method for trapezoidal AUC integration: • "linear" - Linear trapezoidal rule (default) • "linear_up_log_down" - Linear-up / log-down rule
duration	Numeric. Optional. Duration of the observation window (same units as time). Used to restrict the integration or define the evaluation range.
nlastpoints	Integer. Number of terminal points for half-life regression (default = 3).
slope.method	Character. Method for estimating the terminal slope (lambdaz). Options are: • "bestfit": Performs automated terminal phase selection based on adjusted R-squared. If no acceptable segment is found, lambdaz is returned as NA.

- "bestfitforce": First attempts the "bestfit" method. If no valid lambdaz is obtained, the function applies a fallback log-linear regression using progressively fewer terminal points to force estimation. This is the default.

Value

A list with NCA control parameters.

Author(s)

Zhonghui Huang

Examples

```
nca_control()
```

`nmpkconvert` *Expand additional dosing (ADDL) records for pharmacokinetic analysis*

Description

Expands dosing records that contain additional doses (ADDL) using the specified interdose interval (II). Each additional dose is converted into an explicit record to provide a complete dosing history suitable for population pharmacokinetic modeling.

Usage

```
nmpkconvert(dat)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.

Details

Dosing records with ADDL greater than zero are expanded using the formula: TIME_new = TIME + n × II, where n ranges from 1 up to ADDL. Observation records (EVID = 0) are not modified.

Value

A data frame with expanded dosing records. The columns ADDL and II are reset to zero after expansion.

Examples

```
# Dataset with a single subject and additional dosing
dat <- data.frame(
  ID   = rep(1, 6),
  EVID = c(1, 0, 0, 1, 0, 0),
  ADDL = c(2, 0, 0, 0, 0, 0),
  TIME = c(0, 1, 2, 3, 4, 5),
  II   = c(24, 0, 0, 0, 0, 0),
  AMT  = c(100, 0, 0, 0, 0, 0)
)
nmpkconvert(dat)
```

---

pooled_control *Control settings for pooled data analysis*

Description

Defines control parameters for time binning and preprocessing in pooled data analysis. These parameters are typically passed to 'get_pooled_data'.

Usage

```
pooled_control(
  nbins = 10,
 bin_method = c("quantile", "jenks", "kmeans", "pretty", "sd", "equal", "density"),
  tad_rounding = TRUE
)
```

Arguments

nbins	Integer or the character string auto. Number of time bins used to group observations. Default is 10.
bin_method	Character string specifying the binning method. Must be one of "quantile", "jenks", "kmeans", "pretty", "sd", "equal", or "density".
tad_rounding	Logical value indicating whether tad and the most common dosing interval should be rounded to the nearest whole unit before comparison. Default is TRUE, allowing small deviations (for example, a tad of 24.3 is treated as within a 24-unit interval).

Value

A named list containing control parameters for pooled pharmacokinetic analysis.

See Also

get_pooled_data

Examples

```
pooled_control()
```

`print.getPPKinits` *Print method for* `getPPKinits` *objects*

Description

Prints a summary of the results from the initial parameter estimation pipeline, including recommended initial estimates, ETA variance estimates, and parameter descriptions. It is the default S3 `print` method for objects of class `getPPKinits`.

Usage

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

Arguments

`x`	An object of class `getPPKinits` containing the initial parameter estimation results. Expected components include: • `Recommended_initial_estimates`: A data frame with estimated values and selection methods. • `Parameter.descriptions`: A character vector explaining the meaning of each parameter. • `time.spent`: Time taken to compute the estimates.
`...`	Additional arguments (for compatibility with the generic `print()`).

Value

Prints a formatted summary to the console.

Examples

```
## Oral example
inits.out <- getPPKinits(Bolus_1CPT)
print(inits.out)
```

`processData` *Process time–concentration dataset for pharmacokinetic analysis*

Description

Processes a time–concentration dataset to derive analysis-ready variables and structured output for pharmacokinetic evaluation.

Usage

```
processData(dat, verbose = TRUE)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format. The following columns are required (case-insensitive) and must be present: **ID** Subject identifier (required) **TIME** Nominal or actual time after dose (required) **DV** Observed concentration (dependent variable) (required) **EVID** Event ID indicating observation (0) or dosing event (1) (required) **AMT** Dose amount for dosing records (required) **RATE** Infusion rate (optional) **DUR** Infusion duration (optional) **MDV** Missing dependent variable flag (optional) **CMT** Compartment number (optional) **ADDL** Number of additional doses (optional) **II** Interdose interval (optional) **SS** Steady-state indicator (optional) **CENS** Censoring indicator (optional)
`verbose`	Logical (default = TRUE). When TRUE, the function prints detailed processing messages and summary tables to the console, including notes on data cleaning and event handling. When FALSE, these messages are suppressed and only the returned list is produced.

Details

This function standardizes and preprocesses time–concentration data to ensure compatibility with pharmacokinetic modeling workflows in nlmixr2. The operations follow these steps:

1. Standardize data
   - convert column names to uppercase
   - coerce key columns (TIME, DV, EVID, AMT, RATE, etc.) to numeric
2. Process events and observations

- impute EVID from MDV if missing
- handle censored data (CENS) by converting them to excluded records
- remove or recode invalid EVID values (e.g., DV = 0 observations)

3. Expand dose events
   - expand dosing records using nmpkconvert() when ADDL and II are present
   - assign dose occasions using mark_dose_number()
4. Determine administration route and infusion logic
   - derive RATE and DUR when needed
   - identify route (bolus, infusion, oral) based on compartment and rate
5. Generate derived variables
   - calculate time after dose using calculate_tad()
   - compute dose-normalized concentration (DVstd)
   - flag eligible records for terminal elimination phase
6. Summarize dataset
   - classify dataset as first-dose, repeated-dose, or mixed
   - generate summary metrics for nlmixr2 analysis

   Classification of dosing context is based on pharmacokinetic observation records (EVID equal to 0), determining whether observed concentrations occur after the first dose, during repeated dosing, or across both contexts. The categories are:

   - first_dose: observations occur only after the initial administration, without repeated-dose or steady-state intervals.
   - repeated_doses: observations occur only after multiple administrations or under steady-state conditions.
   - combined_doses: observations include both first-dose and repeated-dose intervals and are analyzed together.

Value

A list with two elements:

- dat: A data frame containing the processed time–concentration dataset with standardized and derived pharmacokinetic variables, including resetflag, SSflag, route, dose_number, DVstd, indiv_lambda_z_eligible, and others.
- Datainfo: A data frame summarizing the dataset structure, including subject counts and observation counts for first-dose and multiple-dose conditions, with contextual notes.

Author(s)

Zhonghui Huang

See Also

nmpkconvert, mark_dose_number, calculate_tad

Examples

```
dat <- Bolus_1CPT
processData(dat)
```

`run_graphcal` *Run graphical analysis of pharmacokinetic parameters*

Description

Performs graphical estimation of pharmacokinetic parameters based on pooled concentration–time data and the specified route of administration.

Usage

```
run_graphcal(
  dat,
  route,
  dose_type = c("first_dose", "repeated_doses", "combined_doses"),
  pooled = NULL,
  pooled_ctrl = pooled_control(),
  ...
)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.
`route`	Route of administration. Must be one of bolus, oral, or infusion.
`dose_type`	Specifies the dosing context of the pharmacokinetic observations. Classified as first_dose, repeated_doses, or combined_doses based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts.
`pooled`	Optional pooled dataset. If NULL, pooling is performed internally.
`pooled_ctrl`	Control settings created by `pooled_control()` for time binning and pooling.
`...`	Additional arguments passed to graphical calculation functions.

Details

The function pools individual profiles using `get_pooled_data()` when needed, and then applies route-specific graphical methods (`graphcal_iv` or `graphcal_oral`) to estimate parameters such as clearance, volume of distribution, terminal slope, and absorption rate constant (for oral data).

Value

A list containing graphical estimates of key pharmacokinetic parameters.

Author(s)

Zhonghui Huang

See Also

graphcal_iv, graphcal_oral, get_pooled_data

Examples

```
# Example 1 (iv case)
dat <- Bolus_1CPT
dat <- processData(dat)$dat
run_graphcal(dat, route="bolus")

# Example 2 (oral case)
dat <- Oral_1CPT
dat <- processData(dat)$dat
run_graphcal(dat, route="oral")

# Example 3 (infusion case).
# Approximate calculation. only use when the infusion duration is very short

dat <- Infusion_1CPT
dat <- processData(dat)$dat
run_graphcal(dat, route="infusion")
```

---

run_ka_solution *Estimate the absorption rate constant using pointwise methods*

Description

It implements pointwise estimation of absorption rate constants for single-dose and multiple-dose pharmacokinetic models.

Usage

```
run_ka_solution(df, cl, ke, Fbio = 1)
```

Arguments

df	A data frame containing raw time–concentration data in the standard nlmixr2 format.
cl	A numeric value for drug clearance. It is assumed constant across subjects unless pre-specified per subject.
ke	A numeric value for the elimination rate constant. This is assumed known or estimated from the terminal phase.
Fbio	A numeric value for bioavailability (F). Default is 1.

Details

For each subject, the time of maximum observed concentration (Tmax) is identified as the time corresponding to the highest DV. Only records with TIME less than or equal to Tmax are retained, representing the absorption phase.

Two scenario-specific calculations are implemented: single-dose and multiple-dose at steady state.

- For single-dose data (dose_number == 1 and SteadyState == FALSE), the function uses `ka_calculation_sd()`, which applies a one-compartment oral absorption model under first-order absorption and elimination.
- For steady-state multiple-dose data (dose_number > 1 and SteadyState == TRUE), the function uses `ka_calculation_md()`, which accounts for accumulation using the dosing interval (dose_interval).

This function does not perform model fitting. The median is recommended for use in pharmacokinetic modeling.

Value

A list with the following elements:

ka_calc_median Median ka value across all valid observations.

ka_calc_dat_sd Data frame containing absorption-phase single-dose data, with estimated ka and diagnostic messages.

ka_calc_dat_md Data frame containing absorption-phase steady-state multiple-dose data, with ka estimates.

ka_calc_dat Combined data frame (single-dose and multiple-dose) containing all ka estimates.

Author(s)

Zhonghui Huang

Examples

```
# Single-dose
df <- Oral_1CPT[Oral_1CPT$SD == 1, ]
df <- processData(df)$dat
df <- is_ss(df)
run_ka_solution(df = df, cl = 4, ke = 4/70, Fbio = 1)$ka_calc_median

# Mixed doses
dat <- Oral_1CPT
df_ss <- processData(dat)$dat
df_ss <- is_ss(df_ss)
run_ka_solution(df = df_ss, cl = 4, ke = 4/70, Fbio = 1)$ka_calc_median
```

`run_pooled_nca` *Performs non-compartmental analysis on pooled data*

Description

Implements pooled concentration–time profiling followed by non-compartmental analysis (NCA) to derive pharmacokinetic parameters across single-dose, multiple-dose, or combined dosing scenarios under bolus, oral, or infusion routes.

Usage

```
run_pooled_nca(
  dat,
  route = c("bolus", "oral", "infusion"),
  dose_type = c("first_dose", "repeated_doses", "combined_doses"),
  pooled = NULL,
  pooled_ctrl = pooled_control(),
  nca_ctrl = nca_control()
)
```

Arguments

`dat`	A data frame containing raw time–concentration data in the standard nlmixr2 format.
`route`	Route of administration. Must be one of bolus, oral, or infusion.
`dose_type`	Classified as first_dose, repeated_doses, or combined_doses based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts.
`pooled`	Optional pre-pooled data returned by `get_pooled_data`.
`pooled_ctrl`	Optional list of control parameters used by `get_pooled_data()` for pooling observations. Defaults to output from `pooled_control()`.
`nca_ctrl`	List of options created by `nca_control` for NCA settings.

Details

The function first pools individual subject data into representative concentration–time profiles using `get_pooled_data` based on the settings in `pooled_ctrl`. The pooled profiles are then passed to `getnca`, which computes non-compartmental parameters using rules specified in `nca_ctrl`.

Value

A list containing NCA results according to the selected dose_type.

Author(s)

Zhonghui Huang

See Also

get_pooled_data, bin.time, getnca

Examples

```
out   <- processData(Bolus_1CPT)
dat   <- out$dat
route <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]

run_pooled_nca(
  dat       = dat,
  dose_type = "first_dose",
  route     = route
)$nca.fd.results
```

---

run_single_point *Run full adaptive single-point PK analysis*

Description

Runs full adaptive single-point pharmacokinetic analysis to estimate CL, Vd, and ka across bolus, oral, and infusion dosing scenarios.

Usage

```
run_single_point(
  dat,
  route = c("bolus", "oral", "infusion"),
  half_life = NULL,
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  ssctrl = ss_control()
)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
route	Character string specifying the route of administration. Must be one of "bolus", "oral", or "infusion".
half_life	Optional numeric value representing the elimination half-life of the drug. If not provided, half-life is estimated within run_single_point_base() using get_hf() applied to the pharmacokinetic observations.
dose_type	Classified as "first_dose", "repeated_doses", or "combined_doses" based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts. This parameter is passed directly to run_single_point_base().

pooled_ctrl	A list of pooled-analysis control options created using pooled_control(). These control time binning and time-after-dose rounding when pooled summaries are required.
ssctrl	A list of steady-state control options created using ss_control(). These govern assumptions and thresholds used when interpreting steady-state behavior in single-point logic.

Value

An object of class "single.point.lst" with results from the base and extended steps.

Author(s)

Zhonghui Huang

See Also

run_single_point_base, run_single_point_extra

Examples

```
# Example: Adaptive single-point PK analysis for bolus data
# Step 1: Preprocess the data
dat <- Oral_1CPT
out <- processData(dat)
# Step 2: Extract route and dose type info
froute <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]
fdose_type <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Type"]
# Step 3: Estimate half life
half_life <- get_hf(dat = out$dat)$half_life_median
# Step 4: Run single-point analysis (CL, Vd, Ka if oral)
result <- run_single_point(
  dat = out$dat,
  route = froute,
  dose_type = fdose_type,
  half_life = half_life
)
# Step 5: View results
print(result$singlepoint.results)
```

---

run_single_point_base *Run adaptive single-point pharmacokinetic analysis*

Description

Implements adaptive single-point pharmacokinetic analysis to calculate clearance and volume of distribution.

Usage

```
run_single_point_base(
  dat,
  route = c("bolus", "oral", "infusion"),
  half_life = NULL,
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  ssctrl = ss_control()
)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
route	Route of administration. Must be one of bolus, oral, or infusion.
half_life	Optional numeric value for drug half-life. If not provided, it is estimated from the dataset.
dose_type	Specifies the dosing context of the pharmacokinetic observations. Required when half_life is not provided. Classified as first_dose, repeated_doses, or combined_doses based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts.
pooled_ctrl	Optional list of control parameters used by `get_pooled_data()` for pooling observations. Defaults to output from `pooled_control()`.
ssctrl	A list of control parameters generated by `ss_control()` to guide the detection of steady-state observations.

Details

This function integrates clearance and volume estimation into a unified adaptive workflow, using steady-state pharmacokinetic observations and trimmed mean statistics to reduce the influence of outliers.

Value

A list containing:

- summary: a data frame with trimmed mean clearance and volume of distribution, and run time information
- dat: the processed dataset used for analysis
- cl_df: individual clearance estimates
- vd_df: individual volume of distribution estimates

See Also

calculate_cl, calculate_vd, pooled_control, ss_control

Examples

```
dat <- Bolus_1CPT
out <- processData(dat)
fdat <- out$dat
route <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]
half_life <- get_hf(dat = fdat)$half_life_median
run_single_point_base(dat = fdat, half_life = half_life, route = route)$summary
```

---

run_single_point_extra

Perform extended single-point pharmacokinetic calculations

Description

Extended single-point pharmacokinetic analysis for deriving CL, Vd, and ka

Usage

```
run_single_point_extra(
  dat = NULL,
  half_life = NULL,
  single_point_base.lst = NULL,
  route = c("bolus", "oral", "infusion"),
  dose_type = NULL,
  pooled_ctrl = pooled_control(),
  ssctrl = ss_control()
)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
half_life	Optional numeric value representing the elimination half-life of the drug. If not provided, half-life is estimated within `run_single_point_base()` using `get_hf()` applied to the pharmacokinetic observations.
single_point_base.lst	A list object returned from the base single-point calculation that includes input data, preprocessing information, and initial estimates of CL and Vd. If not supplied, the function will internally call the base routine using dat.
route	Character string specifying the route of administration. Must be one of "bolus", "oral", or "infusion".
dose_type	Classified as "first_dose", "repeated_doses", or "combined_doses" based on whether observed concentrations occur following the first administration, during repeated dosing, or across both contexts. This parameter is passed directly to `run_single_point_base()`.

`pooled_ctrl` A list of pooled-analysis control options created using `pooled_control()`. These control time binning and time-after-dose rounding when pooled summaries are required.

`ssctrl` A list of steady-state control options created using `ss_control()`. These govern assumptions and thresholds used when interpreting steady-state behavior in single-point logic.

Details

The function derives pharmacokinetic parameters using the following logic:

- When both clearance (CL) and volume of distribution (Vd) are available from the base calculation, these values are directly used without modification.
- If Vd is missing but CL and elimination half-life are provided, Vd is calculated using:

$$V_d = \frac{CL \cdot t_{1/2}}{\ln(2)}$$

- If CL is missing but Vd and half-life are available, CL is calculated using:

$$CL = \frac{V_d \cdot \ln(2)}{t_{1/2}}$$

- If both CL and Vd are unavailable but half-life is provided, Vd is estimated using dose and Cmax:

$$V_d = \frac{\text{Dose}}{C_{\max}}$$

and CL is subsequently derived:

$$CL = \frac{V_d \cdot \ln(2)}{t_{1/2}}$$

- For oral administration, the absorption rate constant (ka) is estimated using a solution-based approach implemented in `run_ka_solution()`. Only concentration–time data from the absorption phase are used, defined as time after dose (tad) less than 20% of the terminal half-life and occurring prior to Tmax, where absorption is the dominant process.

The function supports bolus, oral, and infusion administration routes. For oral dosing, only data within the absorption phase are used to estimate the absorption rate constant (ka), specifically using concentration-time observations prior to the maximum concentration (Tmax).

Value

A list containing:

- singlepoint.results: A data frame with estimated ka, CL, Vd, computation start time, processing time, and a descriptive message of the applied logic.
- dat: The dataset used for processing.
- single_point.ka.out: Output used for estimating the absorption rate constant (for oral administration), if applicable.
- single_point_cl_df: Data used for clearance estimation.

- single_point_vd_df: Data used for volume of distribution estimation.
- approx.vc.out: Data used for estimating the volume of distribution from maximum concentration (Cmax) and dose when needed.

Author(s)

Zhonghui Huang

See Also

run_single_point_base, run_single_point, run_ka_solution

Examples

```
dat <- Bolus_1CPT
out <- processData(dat)
fdat <- out$dat
froute <- out$Datainfo$Value[out$Datainfo$Infometrics == "Dose Route"]
half_life <- get_hf(dat = fdat)$half_life_median
run_single_point_extra(
  dat = fdat,
  half_life = half_life,
  single_point_base.lst = run_single_point_base(
    dat = fdat,
    half_life = half_life,
    route = froute
  ),
  route = froute
)
```

---

sim_sens_1cmpt_mm *Parameter sweeping for a one-compartment Michaelis-Menten model*

Description

Performs parameter sweeping by varying pharmacokinetic parameters in a one-compartment model with Michaelis-Menten elimination.

Usage

```
sim_sens_1cmpt_mm(
  dat,
  sim_vmax = list(mode = "auto", values = NULL, est.cl = NULL),
  sim_km = list(mode = "auto", values = NULL),
  sim_vd = list(mode = "manual", values = NULL),
  sim_ka = list(mode = "manual", values = NULL),
  route = c("iv", "oral"),
  ncores = 2,
```

```
  verbose = TRUE
)
```

Arguments

dat	A data frame containing raw time–concentration data in the standard nlmixr2 format.
sim_vmax	List specifying Vmax: • mode: "manual" or "auto" • values: numeric vector if mode = "manual" • est.cl: required if mode = "auto"
sim_km	List specifying Km: • mode: "manual" or "auto" • values: numeric vector if mode = "manual"
sim_vd	List specifying Vd: • mode: must be "manual" • values: numeric vector
sim_ka	List specifying Ka (oral route only): • mode: must be "manual" • values: numeric vector
route	Dosing route, either "iv" or "oral". Default is "iv".
ncores	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
verbose	Logical (default = TRUE). Controls whether progress information is displayed during parameter sweeping. When TRUE, a dynamic progress bar is shown using the `progressr` package to indicate simulation status and elapsed time. When FALSE, progress output is suppressed and the function runs silently.

Details

The function generates a parameter grid and performs model fitting for each combination using `Fit_1cmpt_mm_iv`. Parameters can be specified manually or automatically derived. Model predictions and fit metrics are computed for each simulation to assess parameter sensitivity.

Value

A data frame containing parameter combinations and model fit metrics.

Author(s)

Zhonghui Huang

See Also

`Fit_1cmpt_mm_iv`, `Fit_1cmpt_mm_oral`

Examples

```
# Example: IV dosing scenario with automatic Vmax and Km
out <- sim_sens_1cmpt_mm(
  dat = Bolus_1CPTMM[Bolus_1CPTMM$ID<50,],
  sim_vmax = list(mode = "auto", est.cl = 4),
  sim_km   = list(mode = "auto"),
  sim_vd   = list(mode = "manual", values = 70),
  sim_ka   = list(mode = "manual", values = NA),
  route = "iv"
)
head(out[out$rRMSE2==min(out$rRMSE2),])
```

---

sim_sens_2cmpt *Parameter sweeping for a two-compartment pharmacokinetic model*

Description

Performs parameter sweeping by varying pharmacokinetic parameters in a two-compartment model under IV or oral dosing. Model fit is evaluated across combinations of CL, Vc, Vp, Q, and Ka (oral only).

Usage

```
sim_sens_2cmpt(
  dat,
  sim_ka = list(mode = "manual", values = NULL),
  sim_cl = list(mode = "manual", values = NULL),
  sim_vc = list(mode = "manual", values = NULL),
  sim_vp = list(mode = c("auto", "manual"), values = NULL),
 sim_q = list(mode = c("auto", "manual"), values = NULL, auto.strategy = c("scaled",
    "fixed")),
  route = c("iv", "oral"),
  ncores = 2,
  verbose = TRUE
)
```

Arguments

dat	Pharmacokinetic dataset.
sim_ka	List specifying Ka (oral route only): • mode: must be "manual" • values: numeric vector
sim_cl	List specifying clearance (CL): • mode: must be "manual" • values: numeric vector

sim_vc	List specifying central volume (Vc): • mode: must be "manual" • values: numeric vector
sim_vp	List specifying peripheral volume (Vp): • mode: "manual" or "auto" • values: numeric vector if manual
sim_q	List specifying inter-compartmental clearance (Q): • mode: "manual" or "auto" • values: numeric vector if manual
route	Dosing route, either "iv" or "oral". Default is "iv".
ncores	Number of cores to use for parallelization, passed to rxControl(). Default is 2.
verbose	Logical (default = TRUE). Controls whether progress information is displayed during parameter sweeping. When TRUE, a dynamic progress bar is shown using the progressr package to indicate simulation status and elapsed time. When FALSE, progress output is suppressed and the function runs silently.

Details

The function generates a parameter grid and performs model fitting for each combination using Fit_2cmpt_iv or Fit_2cmpt_oral. Parameters can be specified manually or automatically derived. Model predictions and fit metrics are computed for each simulation to assess parameter sensitivity.

Value

A data frame containing parameter combinations with model fit metrics.

Author(s)

Zhonghui Huang

See Also

Fit_2cmpt_iv, Fit_2cmpt_oral

Examples

```
out <- sim_sens_2cmpt(
  dat = Bolus_2CPT[Bolus_2CPT$ID<6,],
  sim_cl = list(mode = "manual", values = 4),
  sim_vc = list(mode = "manual", values = 50),
  sim_vp = list(mode = "auto"),
  sim_q  = list(mode = "auto"),
  sim_ka = list(mode = "manual", values = NA),
  route = "iv",verbose=FALSE
)
head(out[out$rRMSE2==min(out$rRMSE2),])
```

`sim_sens_3cmpt` *Parameter sweeping for a three-compartment pharmacokinetic model*

Description

Performs parameter sweeping by varying pharmacokinetic parameters in a three-compartment model under IV or oral dosing. Parameter combinations include Vc, Vp1, Vp2, Q1, Q2, CL, and Ka (oral only).

Usage

```
sim_sens_3cmpt(
  dat,
  sim_vc = list(mode = "manual", values = NULL),
  sim_vp = list(mode = c("auto", "manual"), values = NULL),
  sim_vp2 = list(mode = c("auto", "manual"), values = NULL),
 sim_q = list(mode = c("auto", "manual"), values = NULL, auto.strategy = c("scaled",
    "fixed")),
 sim_q2 = list(mode = c("auto", "manual"), values = NULL, auto.strategy = c("scaled",
    "fixed")),
  sim_cl = list(mode = "manual", values = NULL),
  sim_ka = list(mode = "manual", values = NULL),
  route = c("iv", "oral"),
  ncores = 2,
  verbose = TRUE
)
```

Arguments

`dat`	Pharmacokinetic dataset.
`sim_vc`	List specifying Vc: • mode: must be "manual" • values: numeric vector
`sim_vp`	List specifying Vp1: • mode: "manual" or "auto" • values: numeric vector if manual
`sim_vp2`	List specifying Vp2: • mode: "manual" or "auto" • values: numeric vector if manual
`sim_q`	List specifying Q1: • mode: "manual" or "auto" • values: numeric vector if manual • auto.strategy: "scaled" or "fixed" when auto

sim_q2	List specifying Q2: • mode: "manual" or "auto" • values: numeric vector if manual • auto.strategy: "scaled" or "fixed" when auto
sim_cl	List specifying CL: • mode: must be "manual" • values: numeric vector
sim_ka	List specifying Ka (oral route only): • mode: must be "manual" • values: numeric vector
route	Dosing route, either "iv" or "oral". Default is "iv".
ncores	Number of cores to use for parallelization, passed to `rxControl()`. Default is 2.
verbose	Logical (default = TRUE). Controls whether progress information is displayed during parameter sweeping. When TRUE, a dynamic progress bar is shown using the `progressr` package to indicate simulation status and elapsed time. When FALSE, progress output is suppressed and the function runs silently.

Details

The function generates a parameter grid and evaluates each combination using `Fit_3cmpt_iv` or `Fit_3cmpt_oral`. Model predictions and fit metrics are calculated for each simulation to assess parameter influence and identify optimal regions of the parameter space. Parameters can be provided manually or derived automatically.

Value

A data frame containing parameter combinations with model fit metrics.

Author(s)

Zhonghui Huang

See Also

`Fit_3cmpt_iv`, `Fit_3cmpt_oral`

Examples

```
out <- sim_sens_3cmpt(
  dat = Bolus_2CPT[Bolus_2CPT$ID<6,],
  sim_cl = list(mode = "manual", values = 4),
  sim_vc = list(mode = "manual", values = 50),
  sim_vp = list(mode = "auto"),
  sim_vp2 = list(mode = "auto"),
  sim_q  = list(mode = "auto", auto.strategy = "scaled"),
  sim_q2 = list(mode = "auto", auto.strategy = "scaled"),
```

```
  route = "iv",verbose=FALSE
)
head(out[out$rRMSE2==min(out$rRMSE2),])
```

`ss_control` *Internal control builder for steady-state evaluation*

Description

Constructs a list of control parameters used by is_ss() to determine pharmacokinetic steady state.

Usage

```
ss_control(
  ss_method = c("combined", "fixed_doses", "half_life_based"),
  no.doses = 5,
  no.half_lives = 5,
  allowed_interval_variation = 0.25,
  allowed_dose_variation = 0.2,
  min_doses_required = 3,
  tad_rounding = TRUE
)
```

Arguments

`ss_method`	Character string specifying the method used to determine steady state. One of: • "combined" (default): uses the smaller of the dose-based estimate (no.doses) and the half-life-based estimate (no.half_lives) • "fixed_doses": considers steady state reached after no.doses administrations • "half_life_based": uses the drug half-life and dosing interval to estimate the required number of doses
`no.doses`	Integer indicating the number of doses assumed necessary to reach steady state when using the "fixed_doses" method or as part of the "combined" method. Default is 5.
`no.half_lives`	Integer indicating the number of half-lives required to reach steady state when using the "half_life_based" or "combined" method. Default is 5.
`allowed_interval_variation`	Numeric value specifying the acceptable fractional variation in dose interval. For example, 0.25 allows plus or minus 25 percent variation. Default is 0.25.
`allowed_dose_variation`	Numeric value specifying the acceptable fractional variation in dose amount. For example, 0.20 allows plus or minus 20 percent variation. Default is 0.20.

`min_doses_required`
: Integer specifying the minimum number of doses that must be administered regardless of method. Default is 3.

`tad_rounding`
: Logical value. If TRUE (default), rounding is applied when comparing time after dose (tad) to dosing intervals to allow small numerical deviations.

Value

A named list containing the steady-state control parameters, typically passed as the ssctrl argument to `is_ss()`.

See Also

is_ss

Examples

```
ss_control()
ss_control(ss_method = "fixed_doses", no.doses = 4)
```

`trapezoidal_linear` *Linear trapezoidal rule*

Description

Computes the area under the curve (AUC) or the area under the moment curve (AUMC) using the linear trapezoidal rule. If `moment = TRUE`, the function estimates AUMC by integrating `time * concentration`.

Usage

```
trapezoidal_linear(x, y, moment = FALSE)
```

Arguments

`x`
: A numeric vector representing the time points.

`y`
: A numeric vector representing the corresponding concentration values at each time point.

`moment`
: Logical. If `TRUE`, computes AUMC by integrating `t * C(t)` instead of just `C(t)`.

Value

A numeric value representing the estimated AUC or AUMC using the linear trapezoidal rule.

Examples

```
x <- c(0.5, 1, 2, 4, 6, 8)
y <- c(12, 8, 5, 3, 2, 1)
trapezoidal_linear(x, y)                # AUC
trapezoidal_linear(x, y, moment = TRUE) # AUMC
```

`trapezoidal_linear_up_log_down`

Linear-up and log-down trapezoidal rule

Description

Computes the area under the curve (AUC) or the area under the moment curve (AUMC) using a hybrid trapezoidal rule. The method uses linear interpolation for increasing or constant concentration segments, and logarithmic interpolation for decreasing segments.

Usage

```
trapezoidal_linear_up_log_down(x, y, moment = FALSE)
```

Arguments

`x`	A numeric vector representing the time points.
`y`	A numeric vector representing the corresponding concentration values at each time point.
`moment`	Logical. If `TRUE`, computes AUMC by integrating `t * C(t)` instead of `C(t)`.

Details

If `moment = TRUE`, the function calculates the area under the moment curve (AUMC), i.e., it integrates `t * C(t)` over time instead of just `C(t)`.

Value

A numeric value representing the estimated AUC or AUMC using the linear-up/log-down trapezoidal method.

Examples

```
x <- c(0, 0.5, 1, 2, 4, 6, 8)
y <- c(0, 2, 8, 5, 3, 2, 1)
trapezoidal_linear_up_log_down(x, y)                # AUC
trapezoidal_linear_up_log_down(x, y, moment = TRUE) # AUMC
```

`trimmed_geom_mean` *Computes the trimmed geometric mean*

Description

Computes the trimmed geometric mean of a numeric vector

Usage

```
trimmed_geom_mean(x, trim = 0, na.rm = TRUE)
```

Arguments

`x`	A numeric vector containing the values for geometric mean calculation.
`trim`	A numeric value between 0 and 0.5 indicating the proportion of values to be trimmed from each end of the vector. Default is 0.
`na.rm`	Logical value indicating whether missing values should be removed before computation. Default is TRUE.

Value

A numeric value representing the trimmed geometric mean. Returns NA if no values remain after trimming.

Examples

```
x <- c(1, 2, 3, 4, 5, 100)
trimmed_geom_mean(x, trim = 0.05)
```

Index

approx.vc, 3

bin.time, 5, *29*, *59*

calculate_cl, 6, *61*
calculate_tad, 7, *54*
calculate_vd, 8, *61*

eval_perf_1cmpt, 10

fallback_control, 11, *42*
find_best_lambdaz, 12, *28, 29*, *39*
Fit_1cmpt_iv, *11*, 14
Fit_1cmpt_mm_iv, 15, *65*
Fit_1cmpt_mm_oral, 17, *65*
Fit_1cmpt_oral, *11*, 19
Fit_2cmpt_iv, 20, *67*
Fit_2cmpt_oral, 22, *67*
Fit_3cmpt_iv, 23, *69*
Fit_3cmpt_oral, 25, *69*
force_find_lambdaz, 27, *37*

get_hf, *7*, *10*, 28
get_pooled_data, *7*, *10*, *29*, 30, *51*, *56*, *59*
getnca, 31, *59*
getOmegas, 33
getPPKinits, 33
getsigma, 35
getsigmas, *35*, 36
graphcal_iv, 37, *56*
graphcal_oral, 38, *56*

hybrid_eval_perf_1cmpt, 39

initsControl, *34*, 41
is_ss, *7*, 42, *71*

ka_calculation_md, 43
ka_calculation_sd, 44
ka_wanger_nelson, 45

mark_dose_number, 47, *54*
metrics., *11*, *34*, 48

nca_control, *42*, 49
nmpkconvert, 50, *54*

pooled_control, *4*, *7*, *31*, *42*, 51, *61*
print.getPPKinits, 52
processData, 53

run_graphcal, *34*, 55
run_ka_solution, 56, *64*
run_pooled_nca, *34*, 58
run_single_point, *34*, 59, *64*
run_single_point_base, *4*, 60, *60*, *64*
run_single_point_extra, *60*, 62

sim_sens_1cmpt_mm, *34*, 64
sim_sens_2cmpt, *34*, 66
sim_sens_3cmpt, *34*, 68
ss_control, *4*, *7*, *42*, *61*, 70

trapezoidal_linear, 71
trapezoidal_linear_up_log_down, 72
trimmed_geom_mean, *4*, *7*, *10*, *31*, 73

vpc::auto_bin, *5*