Package 'PKNCA'

November 22, 2015

Type Package

Title Perform Pharmacokinetic Non-Compartmental Analysis

Version 0.6

Date 2015-11-19

Author Bill Denney <bill@denney.ws>, Clare Buckeridge <clare.buckeridge@pfizer.com>

Maintainer Bill Denney <bill@denney.ws>

Imports digest, doBy, plyr, nlme, parallel, lattice, stats, utils

Suggests testthat

Description Compute standard Non-Compartmental Analysis (NCA) parameters and summarize them. In addition to this core work, it also provides standardized plotting routines, basic assessments for biocomparison or drug interaction, and model-based estimation routines for calculating doses to reach specific values of AUC or Cmax.

License AGPL-3

Acknowledgement Sridhar Duvvuri for significant help

NeedsCompilation no

RoxygenNote 5.0.0

Repository CRAN

Date/Publication 2015-11-22 15:50:53

R topics documented:

`add.interval.col` *Add columns for calculations within PKNCA intervals*

Description

Add columns for calculations within PKNCA intervals

Usage

```
add.interval.col(name, FUN, values, depends = c(), desc = "",
  datatype = c("interval", "individual", "population"))
```

Arguments

`name`	The column name
`FUN`	The function to run (as a character string)
`values`	Valid values for the column
`depends`	Character vector of columns that must be run before this column.
`desc`	A human-readable description of the parameter (<=40 characters to comply with SDTM)
`datatype`	The type of data

adj.r.squared *Calculate the adjusted r-squared value*

Description

Calculate the adjusted r-squared value

Usage

```
adj.r.squared(r.sq, n)
```

Arguments

r.sq	The r-squared value
n	The number of points

Value

The numeric adjusted r-squared value

AIC.list *Assess the AIC for all models in a list of models*

Description

Assess the AIC for all models in a list of models

Usage

```
## S3 method for class 'list'
AIC(object, ..., assess.best = TRUE)
```

Arguments

object	the list of models
assess.best	determine which model is the best (by lowest AIC)
...	parameters passed to the underlying AIC function (typically the parameter k)

Value

a data frame with row names matching the names of the list x and columns for degrees of freedom (df) and AIC. If assess.best is true, then there will be another column isBest.

See Also

get.best.model

`as.data.frame.PKNCAresults` *Extract the parameter results from a PKNCAresults and return them as a data frame.*

Description

Extract the parameter results from a PKNCAresults and return them as a data frame.

Usage

```
## S3 method for class 'PKNCAresults'
as.data.frame(x, ...)
```

Arguments

`x`	The object to extract results from
`...`	Ignored (for compatibility with generic `as.data.frame`

Value

A data frame of results

`business.mean` *Generate functions to do the named function (e.g. mean) applying the business rules.*

Description

Generate functions to do the named function (e.g. mean) applying the business rules.

Usage

```
business.mean(x, ...)
```

Arguments

`x`	vector to be passed to the various functions
`...`	Additional arguments to be passed to the underlying function.

Value

The value of the various functions or NA if too many values are missing

See Also

pk.business

check.conc.time *Verify that the concentration and time are valid*

Description

If the concentrations or times are invalid, will provide an error. Reasons for being invalid are

- Any `time` value is NA
- `time` is not monotonically increasing
- `conc` and `time` are not the same length

Usage

```
check.conc.time(conc, time, monotonic.time = TRUE)
```

Arguments

`conc`	Measured concentrations
`time`	Time of the measurement of the concentrations
`monotonic.time`	Must the time be unique and monotonically increasing?

Details

Some cases may generate warnings but allow the data to proceed.

- A negative concentration is often but not always an error; it will generate a warning.

Value

None

check.conversion *Check that the conversion to a data type does not change the number of NA values*

Description

Check that the conversion to a data type does not change the number of NA values

Usage

```
check.conversion(x, FUN, ...)
```

Arguments

x	the value to convert
FUN	the function to use for conversion
...	arguments passed to FUN

Value

FUN(x, ...) or an error if the set of NAs change.

---

check.interval.deps *Take in a single row of an interval specification and return that row updated with any additional calculations that must be done to fulfil all dependencies.*

Description

Take in a single row of an interval specification and return that row updated with any additional calculations that must be done to fulfil all dependencies.

Usage

```
check.interval.deps(x)
```

Arguments

x	A data frame with one or morw rows of the PKNCA interval

Value

The interval specification with additional calculations added where requested outputs require them.

See Also

check.interval.specification

`check.interval.specification` *Check the formatting of a calculation interval specification data frame.*

Description

Calculation interval specifications are data frames defining what calculations will be required and summarized from all time intervals. Note: parameters which are not requested may be calculated if it is required for (or computed at the same time as) a requested parameter.

Usage

```
check.interval.specification(x)
```

Arguments

`x`	The data frame specifying what to calculate during each time interval

Details

The data frame columns (with variable type in parentheses) are:

`start` The starting time (numeric)

`end` The ending time (numeric including Inf)

`aucinf` Compute AUCinf (logical)

`auclast` Compute AUClast (logical)

`aucall` Compute AUCall (logical)

`aumcinf` Compute AUMCinf (logical)

`aumclast` Compute AUMClast (logical)

`aumcall` Compute AUMCall (logical)

`tfirst` The time of the first concentration above the limit of quantification (logical)

`tmax` The time of observed maximum concentration (logical)

`tlast` The time of the last concentration above the limit of quantification (logical)

`cmin` The observed minimum concentration during the interval (logical)

`cmax` The observed maximum concentration (logical)

`clast.obs` The observed last concentration (logical)

`clast.pred` The concentration at `tlast` predicted by the half life (logical)

`half.life` The half-life (logical)

`thalf.eff` The effective half-life (logical)

`aucpext` The percent of the AUCinf that is extrapolated after the AUClast (logical)

`cl` The clearance (force: 'force' indicates that clearance should be calculated even if it is a multiple-dose study and the drug has not reached steady-state.)

`mrt` The mean residence time (logical)

`vz` Terminal volume of distribution (logical)

`vss` Steady-state volume of distribution (logical)

The variable types for each column are:

logical A logical variable.

numeric A numeric (non-factor) column

force logical or the text `'force'`. `'force'` indicates that checking if the calculation is appropriate should be skipped.

character or factor The text suggested as either a character or a factor

`start` and end time must always be given, and the `start` must be before the end.

Value

x The potentially updated data frame with the interval calculation specification.

See Also

`check.interval.deps`

`choose.auc.intervals` *Choose intervals to compute AUCs from time and dosing information*

Description

Intervals for AUC are selected by the following metrics:

1. If only one dose is administered, use the `PKNCA.options("single.dose.auc")`
2. If more than one dose is administered, estimate the AUC between any two doses that have PK taken at both of the dosing times and at least one time between the doses.
3. For the final dose of multiple doses, try to determine the dosing interval (τ) and estimate the AUC in that interval if multiple samples are taken in the interval.
4. If there are samples $> \tau$ after the last dose, calculate the half life after the last dose.

Usage

```
choose.auc.intervals(time.conc, time.dosing, options = list(),
  single.dose.aucs = PKNCA.choose.option("single.dose.aucs", options))
```

Arguments

`time.conc`	Time of concentration measurement
`time.dosing`	Time of dosing
`options`	List of changes to the default `PKNCA.options` for calculations.
`single.dose.aucs`	The AUC specification for single dosing.

Value

A data frame with columns for `start`, `end`, `auc.type`, and `half.life`. See `check.interval.specification` for column definitions. The data frame may have zero rows if no intervals could be found.

See Also

`pk.calc.auc`, `pk.calc.aumc`, `pk.calc.half.life`, `find.tau`, `check.interval.specification`, `PKNCA.options`

`clean.conc.blq` *Handle BLQ values in the concentration measurements as requested by the user.*

Description

Handle BLQ values in the concentration measurements as requested by the user.

Usage

```
clean.conc.blq(conc, time, ..., options = list(),
  conc.blq = PKNCA.choose.option("conc.blq", options),
  conc.na = PKNCA.choose.option("conc.na", options), check = TRUE)
```

Arguments

`conc`	Measured concentrations
`time`	Time of the concentration measurement
`options`	List of changes to the default `PKNCA.options` for calculations.
`conc.blq`	How to handle a BLQ value that is between above LOQ values? See details for description.
`conc.na`	How to handle NA concentrations. (See `clean.conc.na`)
`check`	Run `check.conc.time`?
`...`	Additional arguments passed to clean.conc.na

Details

NA concentrations (and their associated times) will be handled as described in `clean.conc.na` before working with the BLQ values. The method for handling NA concentrations can affect the output of which points are considered BLQ and which are considered "middle". Values are considered BLQ if they are 0.

`conc.blq` can be set either a scalar indicating what should be done for all BLQ values or a list with elements named "first", "middle", and "last" each set to a scalar.

The meaning of each of the list elements is:

first Values up to the first non-BLQ value. Note that if all values are BLQ, this includes all values.

middle Values that are BLQ between the first and last non-BLQ values.

last Values that are BLQ after the last non-BLQ value

The valid settings for each are:

"drop" Drop the BLQ values

"keep" Keep the BLQ values

a number Set the BLQ values to that number

Value

The concentration and time measurements (data frame) filtered and cleaned as requested relative to BLQ in the middle.

See Also

clean.conc.na

`clean.conc.na` *Handle NA values in the concentration measurements as requested by the user.*

Description

NA concentrations (and their associated times) will be removed then the BLQ values in the middle

Usage

```
clean.conc.na(conc, time, ..., options = list(),
  conc.na = PKNCA.choose.option("conc.na", options), check = TRUE)
```

Arguments

conc	Measured concentrations
time	Time of the concentration measurement
options	List of changes to the default PKNCA.options for calculations.
conc.na	How to handle NA concentrations? Either 'drop' or a number to impute.
check	Run check.conc.time?
...	Additional items to add to the data frame

Value

The concentration and time measurements (data frame) filtered and cleaned as requested relative to NA in the concentration.

count.non.missing *Count the number of values that are not NA*

Description

Count the number of values that are not NA

Usage

```
count.non.missing(x)
```

Arguments

x	The object to count non-NA values within.

Value

A scalar count of the non-NA values.

find.tau *Find the repeating interval within a vector of doses*

Description

This is intended to find the interval over which x repeats by the rule unique(mod(x, interval)) is minimized.

Usage

```
find.tau(x, na.action = na.omit, options = list(),
  tau.choices = PKNCA.choose.option("tau.choices", options))
```

Arguments

x	the vector to find the interval within
na.action	What to do with NAs in x
options	List of changes to the default PKNCA.options for calculations.
tau.choices	the intervals to look for if the doses are not all equally spaced.

Value

A scalar indicating the repeating interval with the most repetition.

1. If all values are NA then NA is returned.
2. If all values are the same, then 0 is returned.
3. If all values are equally spaced, then that spacing is returned.
4. If one of the choices can minimize the number of unique values, then that is returned.
5. If none of the choices can minimize the number of unique values, then -1 is returned.

---

findOperator — *Find the first occurrence of an operator in a formula and return the left, right, or both sides of the operator.*

Description

Find the first occurrence of an operator in a formula and return the left, right, or both sides of the operator.

Usage

```
findOperator(x, op, side)
```

Arguments

x	The formula to parse
op	The operator to search for (e.g. +, -, *, /, ...)
side	Which side of the operator would you like to see: 'left', 'right', or 'both'.

Value

The side of the operator requested, NA if requesting the left side of a unary operator, and NULL if the operator is not found.

formula.parseFormula *Convert the parsed formula back into the original*

Description

Convert the parsed formula back into the original

Usage

```
## S3 method for class 'parseFormula'
formula(x, drop.groups = FALSE, drop.lhs = FALSE,
  ...)
```

Arguments

x	The parsed formula object to revert to the original
drop.groups	logical. Should the returned formula drop the groups?
drop.lhs	logical. Should the returned formula be one-sided dropping the left hand side?
...	Unused.

Value

A formula (optionally with portions removed)

formula.PKNCAconc *Extract the formula from a PKNCAconc object.*

Description

Extract the formula from a PKNCAconc object.

Usage

```
## S3 method for class 'PKNCAconc'
formula(x, ...)

## S3 method for class 'PKNCAdose'
formula(x, ...)
```

Arguments

x	The object to extract the formula from.
...	Unused

Value

A formula object

geomean *Compute the geometric mean, sd, and CV*

Description

Compute the geometric mean, sd, and CV

Usage

```
geomean(x, na.rm = FALSE)
```

Arguments

x	A vector to compute the geometric mean of
na.rm	Should missing values be removed?

Value

The scalar value of the geometric mean, geometric standard deviation, or geometric coefficient of variation.

get.best.model *Extract the best model from a list of models using AIC.list.*

Description

Extract the best model from a list of models using AIC.list.

Usage

```
get.best.model(object, ...)
```

Arguments

object	the list of models
...	Parameters passed to AIC.list

Value

The model which is assessed as best. If more than one are equal, the first is chosen.

See Also

AIC.list

get.first.model *Get the first model from a list of models*

Description

Get the first model from a list of models

Usage

```
get.first.model(object)
```

Arguments

object	the list of (lists of, ...) models

Value

The first item in the object that is not a list or NA. If NA is passed in or the list (of lists) is all NA, then NA is returned.

get.interval.cols *Get the columns that can be used in an interval specification*

Description

Get the columns that can be used in an interval specification

Usage

```
get.interval.cols()
```

getData.PKNCAconc *Extract all the original data from a PKNCAconc or PKNCAdose object*

Description

Extract all the original data from a PKNCAconc or PKNCAdose object

Usage

```
## S3 method for class 'PKNCAconc'
getData(object)

## S3 method for class 'PKNCAdose'
getData(object)
```

Arguments

object	R object to extract the data from.

`getDepVar` *Get the dependent variable (left hand side of the formula) from a PKNCA object.*

Description

Get the dependent variable (left hand side of the formula) from a PKNCA object.

Usage

```
getDepVar(x, ...)
```

Arguments

x	The object to extract the formula from
...	Unused

Value

The vector of the dependent variable from the object.

`getGroups.PKNCAconc` *Get the groups (right hand side after the | from a PKNCA object.*

Description

Get the groups (right hand side after the | from a PKNCA object.

Usage

```
## S3 method for class 'PKNCAconc'
getGroups(object, form = formula(object), level,
  data = getData(object), sep)

## S3 method for class 'PKNCAdose'
getGroups(object, form = formula(object), level,
  data = getData(object), sep)

## S3 method for class 'PKNCAresults'
getGroups(object, form = formula(object$data$conc),
  level, data = object$result, sep)
```

Arguments

object	The object to extract the data from
form	The formula to extract the data from (defaults to the formula from object)
level	optional. If included, this specifies the level(s) of the groups to include. If a numeric scalar, include the first level number of groups. If a numeric vector, include each of the groups specified by the number. If a character vector, include the named group levels.
data	The data to extract the groups from (defaults to the data from object)
sep	Unused (kept for compatibility with the nlme package)

Value

A data frame with the (selected) group columns.

---

getIndepVar *Get the independent variable (right hand side of the formula) from a PKNCA object.*

Description

Get the independent variable (right hand side of the formula) from a PKNCA object.

Usage

```
getIndepVar(x, ...)
```

Arguments

x	The object to extract the formula from
...	Unused

Value

The vector of the independent variable from the object.

`interp.extrap.conc` *Interpolate concentrations between measurements or extrapolate concentrations after the last measurement.*

Description

`interpolate.conc` and `extrapolate.conc` returns an interpolated (or extrapolated) concentration. `interp.extrap.conc` will choose whether interpolation or extrapolation is required and will also operate on many concentrations. These will typically be used to estimate the concentration between two measured concentrations or after the last measured concentration. Of note, these functions will not extrapolate prior to the first point.

Usage

```
interp.extrap.conc(conc, time, time.out, lambda.z = NA,
  clast = pk.calc.clast.obs(conc, time), options = list(),
  interp.method = PKNCA.choose.option("auc.method", options),
  extrap.method = "AUCinf", conc.blq = PKNCA.choose.option("conc.blq",
  options), conc.na = PKNCA.choose.option("conc.na", options), check = TRUE)

interpolate.conc(conc, time, time.out, options = list(),
  interp.method = PKNCA.choose.option("auc.method", options),
  conc.blq = PKNCA.choose.option("conc.blq", options),
  conc.na = PKNCA.choose.option("conc.na", options), check = TRUE)

extrapolate.conc(conc, time, time.out, lambda.z = NA,
  clast = pk.calc.clast.obs(conc, time), extrap.method = "AUCinf",
  options = list(), conc.na = PKNCA.choose.option("conc.na", options),
  conc.blq = PKNCA.choose.option("conc.blq", options), check = TRUE)
```

Arguments

`conc`	Measured concentrations
`time`	Time of the concentration measurement
`time.out`	Time when interpolation is requested (vector for `interp.extrap.conc`, scalar otherwise)
`lambda.z`	The elimination rate constant. `NA` will prevent extrapolation.
`clast`	The last observed concentration above the limit of quantification. If not given, `clast` is calculated from `pk.calc.clast.obs`
`options`	List of changes to the default `PKNCA.options` for calculations.
`interp.method`	The method for interpolation (either 'lin up/log down' or 'linear')
`extrap.method`	The method for extrapolation: "AUCinf", "AUClast", or "AUCall". See details for usage.
`conc.blq`	How to handle BLQ values. (See `clean.conc.blq` for usage instructions.)
`conc.na`	How to handle NA concentrations. (See `clean.conc.na`)
`check`	Run `check.conc.time`, `clean.conc.blq`, and `clean.conc.na`?

Details

extrap.method 'AUCinf' Use lambda.z to extrapolate beyond the last point with the half-life.

'AUCall' If the last point is above the limit of quantification or missing, this is identical to 'AUCinf'. If the last point is below the limit of quantification, then linear interpolation between the Clast and the next BLQ is used for that interval and all additional points are extrapolated as 0.

'AUClast' Extrapolates all points after the last above the limit of quantification as 0.

Value

The interpolated or extrapolated concentration value as a scalar float.

Functions

- `interpolate.conc`: Interpolate concentrations through Tlast (inclusive)
- `extrapolate.conc`: Extrapolate concentrations after Tlast

See Also

`pk.calc.clast.obs` `pk.calc.half.life`

`merge.splitByData` *Merge lists of data to make a list of lists.*

Description

Merge lists of data to make a list of lists.

Usage

```
## S3 method for class 'splitByData'
merge(..., missing.value = NULL)
```

Arguments

`missing.value`	The value to place when there is not a matching value. If not given, set to NULL.
`...`	Lists of class `splitByData`. If the arguments are named, the names will be used for the output lists.

Value

A list the length of merging all the `groupid` attributes where each element is a list with the first element coming from the first input list, the second from the second imput list, and proceeding until all inputs are completed. The order of the output changes most slowly with the first and faster with each subsequent argument.

model.frame.PKNCAconc *Extract the columns used in the formula (in order) from a PKNCAconc or PKNCAdose object.*

Description

Extract the columns used in the formula (in order) from a PKNCAconc or PKNCAdose object.

Usage

```
## S3 method for class 'PKNCAconc'
model.frame(formula, ...)

## S3 method for class 'PKNCAdose'
model.frame(formula, ...)
```

Arguments

formula	The object to use (parameter name is formula to use the generic function)
...	Unused

Value

A data frame with the columns from the object in formula order.

parseFormula *Parse a formula into its component parts.*

Description

This function supports parsing

Usage

```
parseFormula(form, require.groups = FALSE, require.two.sided = FALSE)
```

Arguments

form	the formula to extract into its parts
require.groups	is it an error not to have groups?
require.two.sided	is it an error to have a one-sided formula?

Details

This function extracts the left hand side (`lhs`), right hand side (`rhs`), groups (`groups` and as a formula, `grpFormula`), the environment (`env`, and the original left/right hand side of the model (`model`).

This function borrows heavily from the `parseGroupFormula` function in the doBy package.

Value

A parseFormula class list with elements of

model The left~right side of the model (excluding groups)

lhs The call for the left hand side

rhs The call for the right hand side (excluding groups)

groups The call for the groups

groupFormula A formula form of the groups

env The original formula's environment

Examples

```
parseFormula("a~b", require.groups=FALSE)
## parseFormula("a~b", require.groups=TRUE) # This is an error
parseFormula("a~b|c")
parseFormula("a~b|c")$groups
```

---

pk.business *Run any function with a maximum missing fraction of X and 0s possibly counting as missing. The maximum fraction missing comes from* `PKNCA.options("max.missing")`.

Description

Note that all missing values are removed prior to calling the function. The function is called with the

Usage

```
pk.business(FUN, zero.missing = FALSE, max.missing)
```

Arguments

`FUN`	function to run. The function is called as FUN(x, ...) with missing values removed.
`zero.missing`	Are zeros counted as missing? If `TRUE` then include them in the missing count.
`max.missing`	The maximum fraction of the data allowed to be missing (a number between 0 and 1, inclusive).

Value

A version of FUN that can be called with parameters that are checked for missingness (and zeros) with missing (and zeros) removed before the call. If `max.missing` is exceeded, then NA is returned.

---

pk.calc.aucpext *Calculate the AUC percent extrapolated*

Description

Calculate the AUC percent extrapolated

Usage

```
pk.calc.aucpext(auclast, aucinf)
```

Arguments

`auclast`	the area under the curve from time 0 to the last measurement above the limit of quantification
`aucinf`	the area under the curve from time 0 to infinity

Value

the numeric value of the AUC percent extrapolated

---

pk.calc.auxc *A compute the Area Under the (Moment) Curve*

Description

Compute the area under the curve (AUC) and the area under the moment curve (AUMC) for pharmacokinetic (PK) data. AUC and AUMC are used for many purposes when analyzing PK in drug development.

Usage

```
pk.calc.auxc(conc, time, interval = c(0, Inf),
  clast = pk.calc.clast.obs(conc, time, check = FALSE), lambda.z = NA,
  auc.type = "AUClast", options = list(),
  method = PKNCA.choose.option("auc.method", options),
  conc.blq = PKNCA.choose.option("conc.blq", options),
  conc.na = PKNCA.choose.option("conc.na", options), check = TRUE,
  fun.linear, fun.log, fun.inf)

pk.calc.auc(conc, time, ..., options = list())
```

```
pk.calc.auc.last(conc, time, ..., options = list())

pk.calc.auc.inf(conc, time, ..., options = list(), lambda.z)

pk.calc.auc.all(conc, time, ..., options = list())

pk.calc.aumc(conc, time, ..., options = list())

pk.calc.aumc.last(conc, time, ..., options = list())

pk.calc.aumc.inf(conc, time, ..., options = list(), lambda.z)

pk.calc.aumc.all(conc, time, ..., options = list())
```

Arguments

conc	Concentration measured
time	Time of concentration measurement (must be monotonically increasing and the same length as the concentration data)
interval	Numeric vector of two numbers for the start and end time of integration
clast	The last concentration above the limit of quantification; this is used for AUCinf calculations. If provided as clast.obs (observed clast value, default), AUCinf is AUCinf,obs. If provided as clast.pred, AUCinf is AUCinf,pred.
lambda.z	The elimination rate (in units of inverse time) for extrapolation
auc.type	The type of AUC to compute. Choices are 'AUCinf', 'AUClast', and 'AUCall'.
options	List of changes to the default PKNCA.options for calculations.
method	The method for integration (either 'lin up/log down' or 'linear')
conc.blq	How to handle BLQ values in between the first and last above LOQ concentrations. (See clean.conc.blq for usage instructions.)
conc.na	How to handle missing concentration values. (See clean.conc.na for usage instructions.)
check	Run check.conc.time, clean.conc.blq, and clean.conc.na?
fun.linear	The function to use for integration of the linear part of the curve (not required for AUC or AUMC functions)
fun.log	The function to use for integration of the logarithmic part of the curve (if log integration is used; not required for AUC or AUMC functions)
fun.inf	The function to use for extrapolation from the final measurement to infinite time (not required for AUC or AUMC functions.
...	For functions other than pk.calc.auxc, these values are passed to pk.calc.auxc

Details

pk.calc.auc.last is simply a shortcut setting the interval parameter to c(0, "last").

Extrapolation beyond Clast occurs using the half-life and Clast,obs; Clast,pred is not yet supported.

Value

A numeric value for the AU(M)C

Functions

- `pk.calc.auc`: Compute the area under the curve
- `pk.calc.auc.last`: Compute the AUClast.
- `pk.calc.auc.inf`: Compute the AUCinf.
- `pk.calc.auc.all`: Compute the AUCall.
- `pk.calc.aumc`: Compute the area under the moment curve
- `pk.calc.aumc.last`: Compute the AUMClast.
- `pk.calc.aumc.inf`: Compute the AUMCinf.
- `pk.calc.aumc.all`: Compute the AUMCall.

References

Gabrielsson J, Weiner D. "Section 2.8.1 Computation methods - Linear trapezoidal rule." Pharmacokinetic & Pharmacodynamic Data Analysis: Concepts and Applications, 4th Edition. Stockholm, Sweden: Swedish Pharmaceutical Press, 2000. 162-4.

Gabrielsson J, Weiner D. "Section 2.8.3 Computation methods - Log-linear trapezoidal rule." Pharmacokinetic & Pharmacodynamic Data Analysis: Concepts and Applications, 4th Edition. Stockholm, Sweden: Swedish Pharmaceutical Press, 2000. 164-7.

See Also

`pk.calc.auc.all`, `pk.calc.auc.last`, `clean.conc.blq`

Examples

```
myconc <- c(0, 1, 2, 1, 0.5, 0.25, 0)
mytime <- c(0, 1, 2, 3, 4,   5,    6)
pk.calc.auc(myconc, mytime, interval=c(0, 6))
pk.calc.auc(myconc, mytime, interval=c(0, Inf))
```

---

pk.calc.cav *Calculate the average concentration during an interval.*

Description

Calculate the average concentration during an interval.

Usage

```
pk.calc.cav(auclast, start, end)
```

Arguments

auclast	The area under the curve during the interval
start	The starting time of the interval
end	The ending time of the interval

Value

The Cav (average concentration during the interval)

---

pk.calc.cl *Calculate the (observed oral) clearance*

Description

Calculate the (observed oral) clearance

Usage

```
pk.calc.cl(dose, auc, unitconv)
```

Arguments

dose	the dose administered
auc	the area under the curve from 0 to infinity or 0 to tau (the next dose on a regular schedule at steady-state)
unitconv	the multiplied factor to use for unit conversion (e.g. 1000 for mg dose, time*ng/mL for auc, and output in L/time)

Value

the numeric value of the total (CL) or observed oral clearance (CL/F)

References

Gabrielsson J, Weiner D. "Section 2.5.1 Derivation of clearance." Pharmacokinetic & Pharmacodynamic Data Analysis: Concepts and Applications, 4th Edition. Stockholm, Sweden: Swedish Pharmaceutical Press, 2000. 86-7.

pk.calc.clast.obs *Determine the last observed concentration above the limit of quantification (LOQ).*

Description

If Tlast is NA (due to no non-missing above LOQ measurements), this will return NA.

Usage

```
pk.calc.clast.obs(conc, time, check = TRUE)
```

Arguments

conc	Concentration measured
time	Time of concentration measurement
check	Run check.conc.time?

Value

The last observed concentration above the LOQ

pk.calc.cmax *Determine maximum observed PK concentration*

Description

Determine maximum observed PK concentration

Usage

```
pk.calc.cmax(conc, check = TRUE)

pk.calc.cmin(conc, check = TRUE)
```

Arguments

conc	Concentration measured
check	Run check.conc.time?

Value

a number for the maximum concentration or NA if all concentrations are missing

Functions

- pk.calc.cmin: Determine the minimum observed PK concentration

pk.calc.ctrough *Determine the trough (predose) concentration*

Description

Determine the trough (predose) concentration

Usage

```
pk.calc.ctrough(conc, time, start)
```

Arguments

conc	Observed concentrations during the interval
time	Times of conc observations
start	Starting time of the interval

Value

The concentration when `time == start`. If none match, then `NA`

pk.calc.f *Calculate the absolute (or relative) bioavailability*

Description

Calculate the absolute (or relative) bioavailability

Usage

```
pk.calc.f(dose1, auc1, dose2, auc2)
```

Arguments

dose1	The dose administered in route or method 1
auc1	The AUC from 0 to infinity or 0 to tau administered in route or method 1
dose2	The dose administered in route or method 2
auc2	The AUC from 0 to infinity or 0 to tau administered in route or method 2

pk.calc.half.life *Compute the half-life and associated parameters*

Description

The half-life is calculated by computing the best fit line for all available sets of points. The best one is chosen by the following rules in order:

Usage

```
pk.calc.half.life(conc, time, tmax, tlast, options = list(),
  min.hl.points = PKNCA.choose.option("min.hl.points", options),
  adj.r.squared.factor = PKNCA.choose.option("adj.r.squared.factor", options),
  conc.blq = PKNCA.choose.option("conc.blq", options),
  conc.na = PKNCA.choose.option("conc.na", options),
  first.tmax = PKNCA.choose.option("first.tmax", options),
  allow.tmax.in.half.life = PKNCA.choose.option("allow.tmax.in.half.life",
  options), check = TRUE)
```

Arguments

conc	Concentration measured
time	Time of concentration measurement
tmax	Time of maximum concentration (will be calculated and included in the return data frame if not given)
tlast	Time of last concentration above the limit of quantification (will be calculated and included in the return data frame if not given)
options	List of changes to the default PKNCA.options for calculations.
min.hl.points	The minimum number of points that must be included to calculate the half-life
adj.r.squared.factor	The allowance in adjusted r-squared for adding another point.
conc.blq	See clean.conc.blq
conc.na	See clean.conc.na
first.tmax	See pk.calc.tmax.
allow.tmax.in.half.life	Allow the concentration point for tmax to be included in the half-life slope calculation.
check	Run check.conc.time, clean.conc.blq, and clean.conc.na?

Details

- At least min.hl.points points included
- A lambda.z > 0
- The best adjusted r-squared (within adj.r.squared.factor)
- The one with the most points included

Value

A data frame with one row and columns for

tmax Time of maximum observed concentration (only included if not given as an input)

tlast Time of last observed concentration above the LOQ (only included if not given as an input)

r.squared coefficient of determination

adj.r.squared adjusted coefficient of determination

lambda.z elimination rate

lambda.z.time.first first time for half-life calculation

lambda.z.n.points number of points in half-life calculation

clast.pred Concentration at tlast as predicted by the half-life line

half.life half-life

span.ratio span ratio [ratio of half-life to time used for half-life calculation

References

Gabrielsson J, Weiner D. "Section 2.8.4 Strategies for estimation of lambda-z." Pharmacokinetic & Pharmacodynamic Data Analysis: Concepts and Applications, 4th Edition. Stockholm, Sweden: Swedish Pharmaceutical Press, 2000. 167-9.

`pk.calc.kel` *Calculate the elimination rate (Kel)*

Description

Calculate the elimination rate (Kel)

Usage

```
pk.calc.kel(mrt)
```

Arguments

`mrt`	the mean residence time

Value

the numeric value of the elimination rate

pk.calc.mrt *Calcuate the mean residence time (MRT)*

Description

Calcuate the mean residence time (MRT)

Usage

```
pk.calc.mrt(auc, aumc)
```

Arguments

auc	the AUC from 0 to infinity or 0 ot tau at steady-state
aumc	the AUMC from 0 to infinity or 0 ot tau at steady-state

Value

the numeric value of the mean residence time

pk.calc.ptr *Determine the peak-to-trough ratio*

Description

Determine the peak-to-trough ratio

Usage

```
pk.calc.ptr(cmax, cmin)
```

Arguments

cmax	The maximum observed concentration
cmin	The minimum observed concentration

Value

The ratio of cmax to cmin (if cmin == 0, NA)

pk.calc.thalf.eff *Calculate the effective half-life*

Description

Calculate the effective half-life

Usage

```
pk.calc.thalf.eff(mrt)
```

Arguments

mrt	the mean residence time

Value

the numeric value of the effective half-life

pk.calc.tlag *Determine the observed lag time (time before the first concentration above the limit of quantification or above the first concentration in the interval)*

Description

Determine the observed lag time (time before the first concentration above the limit of quantification or above the first concentration in the interval)

Usage

```
pk.calc.tlag(conc, time)
```

Arguments

conc	The observed concentrations
time	The observed times

Value

The time associated with the first increasing concentration

pk.calc.tlast *Determine time of last observed concentration above the limit of quantification.*

Description

NA will be returned if all conc are NA or 0.

Usage

```
pk.calc.tlast(conc, time, check = TRUE)

pk.calc.tfirst(conc, time, check = TRUE)
```

Arguments

conc	Concentration measured
time	Time of concentration measurement
check	Run check.conc.time?

Value

The time of the last observed concentration measurement

Functions

- pk.calc.tfirst: Determine the first concentration above the limit of quantification.

pk.calc.tmax *Determine time of maximum observed PK concentration*

Description

Input restrictions are:

1. the conc and time must be the same length,
2. the time may have no NAs,

NA will be returned if:

1. the length of conc and time is 0
2. all conc is 0 or NA

Usage

```
pk.calc.tmax(conc, time, options = list(),
  first.tmax = PKNCA.choose.option("first.tmax", options), check = TRUE)
```

Arguments

conc	Concentration measured
time	Time of concentration measurement
options	List of changes to the default PKNCA.options for calculations.
first.tmax	If there is more than time that matches the maximum concentration, should the first be considered as Tmax? If not, then the last is considered Tmax.
check	Run check.conc.time?

Value

the time of the maximum concentration

---

pk.calc.vss *Calculate the steady-state volume of distribution (Vss)*

Description

Calculate the steady-state volume of distribution (Vss)

Usage

```
pk.calc.vss(cl, mrt)
```

Arguments

cl	the clearance
mrt	the mean residence time

Value

the volume of distribution at steady-state

pk.calc.vz — *Calculate the terminal volume of distribution (Vz)*

Description

Calculate the terminal volume of distribution (Vz)

Usage

```
pk.calc.vz(dose, auc, kel, unitconv)
```

Arguments

dose	the dose administered
auc	the AUC from 0 to infinity
kel	the elimination rate
unitconv	the factor to use for unit conversion (e.g. 1000 for mg dose, time*ng/mL for auc, 1/time for kel, and output in L)

pk.nca — *Compute NCA parameters for each interval for each subject.*

Description

The computations assume that all calculation options are set in PKNCA.options.

Usage

```
pk.nca(data)
```

Arguments

data	A PKNCAdata object

Value

A data frame with a row for each interval for each grouping in the concentration data.

See Also

PKNCAdata, PKNCA.options

pk.nca.interval *Compute all PK parameters for a single concentration-time data set*

Description

For one subject/time range, compute all available PK parameters. All the internal options should be set by PKNCA.options prior to running. The only part that changes with a call to this function is the concentration and time.

Usage

```
pk.nca.interval(conc, time, interval, options = list())
```

Arguments

conc	Concentration measured
time	Time of concentration measurement
interval	One row of an interval definition (see check.interval.specification for how to define the interval.
options	List of changes to the default PKNCA.options for calculations.

Value

A data frame with the start and end time along with all PK parameters for the interval

See Also

check.interval.specification

pk.tss *Compute the time to steady-state (tss)*

Description

Compute the time to steady-state (tss)

Usage

```
pk.tss(..., type = c("monoexponential", "stepwise.linear"), check = TRUE)
```

Arguments

type	The type of Tss to calculate, either stepwise.linear or monoexponential
check	See pk.tss.data.prep
...	Passed to pk.tss.monoexponential or pk.tss.stepwise.linear.

Value

A data frame with columns as defined from `pk.tss.monoexponential` and/or `pk.tss.stepwise.linear`.

See Also

`pk.tss.monoexponential`, `pk.tss.stepwise.linear`

`pk.tss.data.prep` *Clean up the time to steady-state parameters and return a data frame for use by the tss calculators.*

Description

Clean up the time to steady-state parameters and return a data frame for use by the tss calculators.

Usage

```
pk.tss.data.prep(conc, time, subject, treatment, subject.dosing, time.dosing,
  options = list(), conc.blq = PKNCA.choose.option("conc.blq", options),
  conc.na = PKNCA.choose.option("conc.na", options), check = TRUE, ...)
```

Arguments

`conc`	Concentration measured
`time`	Time of concentration measurement
`subject`	Subject identifiers (used as a random effect in the model)
`treatment`	Treatment description (if missing, all subjects are assumed to be on the same treatment)
`subject.dosing`	Subject number for dosing
`time.dosing`	Time of dosing
`options`	List of changes to the default `PKNCA.options` for calculations.
`conc.blq`	See `clean.conc.blq`
`conc.na`	See `clean.conc.na`
`check`	Run `check.conc.time`?
`...`	Discarded inputs to allow generic calls between tss methods.

Value

a data frame with columns for concentration, `time`, `subject`, and `treatment`.

`pk.tss.monoexponential` *Compute the time to steady state using nonlinear, mixed-effects modeling of trough concentrations.*

Description

Trough concentrations are selected as concentrations at the time of dosing. An exponential curve is then fit through the data with a different magnitude by treatment (as a factor) and a random steady-state concentration and time to stead-state by subject (see `random.effects` argument).

Usage

```
pk.tss.monoexponential(..., tss.fraction = 0.9, output = c("population",
  "popind", "individual", "single"), check = TRUE, verbose = FALSE)
```

Arguments

`tss.fraction`	The fraction of steady-state required for calling steady-state
`output`	Which types of outputs should be produced? `population` is the population estimate for time to steady-state (from an nlme model), `popind` is the individual estimate (from an nlme model), `individual` fits each individual separately with a gnls model, and `single` fits all the data to a single gnls model.
`check`	See `pk.tss.data.prep`.
`verbose`	Describe models as they are run, show convergence of the model (passed to the nlme function), and additional details while running.
`...`	See `pk.tss.data.prep`

Value

A scalar float for the first time when steady-state is achieved or `NA` if it is not observed.

References

Maganti L, Panebianco DL, Maes AL. Evaluation of Methods for Estimating Time to Steady State with Examples from Phase 1 Studies. AAPS Journal 10(1):141-7. doi:10.1208/s12248-008-9014-y

See Also

`pk.tss`, `pk.tss.stepwise.linear`

pk.tss.monoexponential.individual

A helper function to estimate individual and single outputs for mono-exponential time to steady-state.

Description

This function is not intended to be called directly. Please use `pk.tss.monoexponential`.

Usage

```
pk.tss.monoexponential.individual(data, output = c("individual", "single"),
  verbose = FALSE)
```

Arguments

`data`	a data frame as prepared by `pk.tss.data.prep`. It must contain at least columns for `subject`, `time`, `conc`, and `tss.constant`.
`output`	a character vector requesting the output types.
`verbose`	Show verbose output.

Details

If no model converges, then the `tss.monoexponential.single` and/or `tss.monoexponential.individual` column will be set to NA.

Value

A data frame with either one row (if `population` output is provided) or one row per subject (if `popind` is provided). The columns will be named `tss.monoexponential.population` and/or `tss.monoexponential.popind`.

pk.tss.monoexponential.population

A helper function to estimate population and popind outputs for mo-noexponential time to steady-state.

Description

This function is not intended to be called directly. Please use `pk.tss.monoexponential`.

Usage

```
pk.tss.monoexponential.population(data, output = c("population", "popind"),
  verbose = FALSE)
```

Arguments

data	a data frame as prepared by pk.tss.data.prep. It must contain at least columns for subject, time, conc, and tss.constant.
output	a character vector requesting the output types.
verbose	Show verbose output.

Details

If no model converges, then the tss.monoexponential.population column will be set to NA. If the best model does not include a random effect for subject on Tss then the tss.monoexponential.popind column of the output will be set to NA.

Value

A data frame with either one row (if population output is provided) or one row per subject (if popind is provided). The columns will be named tss.monoexponential.population and/or tss.monoexponential.popind.

---

pk.tss.stepwise.linear

Compute the time to steady state using stepwise test of linear trend

---

Description

A linear slope is fit through the data to find when it becomes non-significant. Note that this is less preferrred than the pk.tss.monoexponential due to the fact that with more time or more subjects the performance of the test changes (see reference).

Usage

```
pk.tss.stepwise.linear(..., min.points = 3, level = 0.95, verbose = FALSE,
  check = TRUE)
```

Arguments

min.points	The minimum number of points required for the fit
level	The confidence level required for assessment of steady-state
verbose	Describe models as they are run, show convergence of the model (passed to the nlme function), and additional details while running.
check	See pk.tss.data.prep
...	See pk.tss.data.prep

Details

The model is fit with a different magnitude by treatment (as a factor, if given) and a random slope by subject (if given). A minimum of min.points is required to fit the model.

Value

A scalar float for the first time when steady-state is achieved or NA if it is not observed.

References

Maganti L, Panebianco DL, Maes AL. Evaluation of Methods for Estimating Time to Steady State with Examples from Phase 1 Studies. AAPS Journal 10(1):141-7. doi:10.1208/s12248-008-9014-y

See Also

pk.tss.monoexponential

PKNCA *Compute noncompartmental pharmacokinetics*

Description

This package computes pharmacokinetic (PK) noncompartmental analysis (NCA) parameters.

Details

Typically, you will start with the pk.nca function and then use the summary function to generate tables for reporting.

PKNCA.choose.option *Choose either the value from an option list or the current set value for an option.*

Description

Choose either the value from an option list or the current set value for an option.

Usage

```
PKNCA.choose.option(name, options = list())
```

Arguments

name	The option name requested.
options	The non-default options to choose from.

Value

The value of the option first from the options list and if it is not there then from the current settings.

See Also

PKNCA.options

PKNCA.options *Set default options for PKNCA functions*

Description

This function will set the default PKNCA options. If given no inputs, it will provide the current option set. If given name/value pairs, it will set the option (as in the options function). If given a name, it will return the value for the parameter. If given the default option as true, it will provide the default options.

Usage

```
PKNCA.options(..., default = FALSE, check = FALSE, name, value)
```

Arguments

default	(re)sets all default options
check	check a single option given, but do not set it (for validation of the values when used in another function)
name	An option name to use with the value.
value	An option value (paired with the name) to set or check.
...	options to set or get the value for

Details

Options are either for calculation or summary functions. Calculation options are required for a calculation function to report a result (otherwise the reported value will be NA). Summary options are used during summarization and are used for assessing what values are included in the summary.

Value

If...

no arguments are given returns the current options.

a value is set (including the defaults) returns NULL

a single value is requested the current value of that option is returned as a scalar

multiple values are requested the current values of those options are returned as a list

See Also

PKNCA.choose.option

Examples

```
PKNCA.options()
PKNCA.options(default=TRUE)
PKNCA.options("auc.method")
PKNCA.options(name="auc.method")
PKNCA.options(auc.method="lin up/log down", min.hl.points=3)
```

`PKNCA.set.summary` *Define how NCA parameters are summarized.*

Description

Define how NCA parameters are summarized.

Usage

```
PKNCA.set.summary(name, point, spread, rounding = list(signif = 3),
  reset = FALSE)
```

Arguments

`name`	The parameter name. It must have already been defined (see `add.interval.col`).
`point`	The function to calculate the point estimate for the summary. The function will be called as `point(x)` and must return a scalar value (typically a number, NA, or a string).
`spread`	Optional. The function to calculate the spread (or variability). The function will be called as `spread(x)` and must return a scalar or two-long vector (typically a number, NA, or a string).
`rounding`	Instructions for how to round the value of point and spread. It may either be a list or a function. If it is a list, then it must have a single entry with a name of either "signif" or "round" and a value of the digits to round. If a function, it is expected to return a scalar number or character string with the correct results for an input of either a scalar or a two-long vector.
`reset`	Reset all the summary instructions

Value

All current summary settings (invisibly)

See Also

`summary.PKNCAresults`

PKNCAconc *Create a PKNCAconc object*

Description

Create a PKNCAconc object

Usage

```
PKNCAconc(data, formula, subject, labels, units)
```

Arguments

data	A data frame with concentration, time, and the groups defined in formula.
formula	The formula defining the concentration~time|groups
subject	The column indicating the subject number (used for plotting). If not provided, this defaults to the beginning of the inner groups: For example with concentration~time|Study+Subjec the inner groups start with the first grouping variable before a /, Subject. If there is only one grouping variable, it is assumed to be the subject (e.g. concentration~time|Subject), and if there are multiple grouping variables without a /, subject is assumed to be the last one. For single-subject data, it is assigned as NULL.
labels	(optional) Labels for use when plotting. They are a named list where the names correspond to the names in the data frame and the values are used for xlab and/or ylab as appropriate.
units	(optional) Units for use when plotting and calculating parameters. Note that unit conversions and simplifications are not done; the text is used as-is.

Value

A PKNCAconc object that can be used for automated NCA.

PKNCAdata *Create a PKNCAdata object.*

Description

PKNCAdata combines PKNCAconc and PKNCAdose and adds in the intervals for PK calculations.

Usage

```
PKNCAdata(data.conc, formula.conc, data.dose, formula.dose, intervals,
  options = list())
```

Arguments

data.conc	Concentration data as a PKNCAconc object or a data frame
formula.conc	Formula for making a PKNCAconc object with data.conc. This must be given if data.conc is a data.frame, and it must not be given if data.conc is a PKNCAconc object.
data.dose	Dosing data as a PKNCAdose object
formula.dose	Formula for making a PKNCAdose object with data.dose. This must be given if data.dose is a data.frame, and it must not be given if data.dose is a PKNCAdose object.
intervals	A data frame with the AUC interval specifications as defined in check.interval.specification. If missing, this will be automatically chosen by choose.auc.intervals.
options	List of changes to the default PKNCA.options for calculations.

Value

A PKNCAdata object with concentration, dose, interval, and calculation options stored (note that PKNCAdata objects can also have results after a NCA calculations are done to the data).

See Also

PKNCAconc, PKNCAdose, choose.auc.intervals

PKNCAdose *Create a PKNCAdose object*

Description

Create a PKNCAdose object

Usage

```
PKNCAdose(data, formula, labels, units)
```

Arguments

data	A data frame with time and the groups defined in formula.
formula	The formula defining the ~time|groups where time is the time of the dosing.
labels	(optional) Labels for use when plotting. They are a named list where the names correspond to the names in the data frame and the values are used for xlab and/or ylab as appropriate.
units	(optional) Units for use when plotting and calculating parameters. Note that unit conversions and simplifications are not done; the text is used as-is.

Value

A PKNCAconc object that can be used for automated NCA.

PKNCAresults *Generate a PKNCAresults object*

Description

This function should not be run directly. The object is created for summarization and plotting.

Usage

```
PKNCAresults(result, data, provenance)
```

Arguments

result	a data frame with NCA calculation results and groups. Each row is one interval and each column is a group name or the name of an NCA parameter.
data	The PKNCAdata used to generate the result
provenance	Data and calculation provenance

Value

A PKNCAresults object with each of the above within.

plot.PKNCAconc *Plot a PKNCAconc object*

Description

Plot a PKNCAconc object

Usage

```
## S3 method for class 'PKNCAconc'
plot(x, ..., groups = x$subject,
  panel.formula = parseFormula(x)$groupFormula, panel.formula.update)

## S3 method for class 'PKNCAdata'
plot(x, ...)
```

Arguments

x	The object to plot
groups	The grouping variable for the plot (typically the subject column)
panel.formula	The formula used for the call to xyplot (defaults to the group formula of x)
panel.formula.update	Updates to the panel.formula to simplify modifications without having to fully specify the formula.
...	Additional arguments passed to xyplot

Value

A trellis object of the plot(s)

`print.PKNCAconc` *Print and/or summarize a PKNCAconc or PKNCAdose object.*

Description

Print and/or summarize a PKNCAconc or PKNCAdose object.

Usage

```
## S3 method for class 'PKNCAconc'
print(x, n = 6, summarize = FALSE, ...)

## S3 method for class 'PKNCAconc'
summary(object, n = 0, summarize = TRUE, ...)

## S3 method for class 'PKNCAdose'
print(x, n = 6, summarize = FALSE, ...)

## S3 method for class 'PKNCAdose'
summary(object, n = 0, summarize = TRUE, ...)
```

Arguments

`x`	The object to print
`n`	The number of rows of data to show (see `head`)
`summarize`	Summarize the nested number of groups
`object`	The object to summarize
`...`	Arguments passed to `print.formula` and `print.data.frame`

`print.PKNCAdata` *Print a PKNCAdata object*

Description

Print a PKNCAdata object

Usage

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

Arguments

x	The object to print
...	Arguments passed on to print.PKNCAconc and print.PKNCAdose

---

`roundingSummarize` *During the summarization of PKNCAresults, do the rounding of values based on the instructions given.*

Description

During the summarization of PKNCAresults, do the rounding of values based on the instructions given.

Usage

```
roundingSummarize(x, name)
```

Arguments

x	The values to summarize
name	The NCA parameter name (matching a parameter name in PKNCA.set.summary)

Value

A string of the rounded value

`roundString` *Round a value to a defined number of digits printing out trailing zeros, if applicable.*

Description

Round a value to a defined number of digits printing out trailing zeros, if applicable.

Usage

```
roundString(x, digits = 0)
```

Arguments

x	The number to round
digits	integer indicating the number of decimal places

Value

A string with the value

See Also

round, signifString

`sapplyBy` *Similar to lapplyBy but returning a data frame*

Description

Similar to lapplyBy but returning a data frame

Usage

```
sapplyBy(formula, data = parent.frame(), FUN)
```

Arguments

formula	See splitBy
data	See splitBy
FUN	either a function or a named list of functions

Value

A data frame with one column for each parameter of formula and one for each FUN. If FUN is a named list, then the columns will be named the same; otherwise, the column will be named "FUN".

`signifString` *Round a value to a defined number of significant digits printing out trailing zeros, if applicable.*

Description

Round a value to a defined number of significant digits printing out trailing zeros, if applicable.

Usage

```
signifString(x, digits = 6)
```

Arguments

x	The number to round
digits	integer indicating the number of significant digits

Value

A string with the value

See Also

signif, roundString

`sort.interval.cols` *Sort the interval columns by dependencies.*

Description

Columns are always to the right of columns that they depend on.

Usage

```
## S3 method for class 'interval.cols'
sort()
```

`summary.PKNCAdata` *Summarize a PKNCAdata object showing important details about the concentration, dosing, and interval information.*

Description

Summarize a PKNCAdata object showing important details about the concentration, dosing, and interval information.

Usage

```
## S3 method for class 'PKNCAdata'
summary(object, ...)
```

Arguments

`object`	The PKNCAdata object to summarize.
`...`	arguments passed on to `print.PKNCAdata`

summary.PKNCAresults *Summarize PKNCA results*

Description

Summarize PKNCA results

Usage

```
## S3 method for class 'PKNCAresults'
summary(object, simplify.start = TRUE,
  drop.group = "Subject", not.requested.string = ".",
  not.calculated.string = "NC")
```

Arguments

object	The results to summarize
simplify.start	Should all results with the same starting time for the interval be summarized together? (I.e. If TRUE, AUC[0-24] and AUC[0-Inf] are both summarized in the same row.)
drop.group	Which group(s) should be dropped from the formula?
not.requested.string	A character string to use when a parameter summary was not requested for a parameter within an interval.
not.calculated.string	A character string to use when a parameter summary was requested, but the point estimate AND spread calculations (if applicable) returned NA.

Value

A data frame of NCA parameter results summarized according to the summarization settings.

See Also

PKNCA.set.summary

`superposition` *Compute noncompartmental superposition for repeated dosing*

Description

Compute noncompartmental superposition for repeated dosing

Usage

```
superposition(conc, ...)

## S3 method for class 'PKNCAconc'
superposition(conc, ...)

## S3 method for class 'numeric'
superposition(conc, time, dose.input, tau, dose.times = 0,
  dose.amount, n.tau = Inf, options = list(), lambda.z,
  clast.pred = FALSE, tlast, additional.times = c(), check.blq = TRUE,
  interp.method = PKNCA.choose.option("auc.method", options),
  extrap.method = "AUCinf", steady.state.tol = 0.001, ...)
```

Arguments

`conc`	Concentration measured
`...`	Additional arguments passed to the `half.life` function if required to compute `lambda.z`.
`time`	Time of concentration measurement
`dose.input`	The dose given to generate the `conc` and `time` inputs. If missing, output doses will be assumed to be equal to the input dose.
`tau`	The dosing interval
`dose.times`	The time of dosing within the dosing interval. The `min(dose.times)` must be >= 0, and the `max(dose.times)` must be < `tau`. There may be more than one dose times given as a vector.
`dose.amount`	The doses given for the output. Linear proportionality will be used from the input to output if they are not equal. The length of dose.amount must be either 1 or matching the length of `dose.times`.
`n.tau`	The number of tau dosing intervals to simulate or `Inf` for steady-state.
`options`	The PKNCA.options to use for the calculation (passed on to subsequent functions like `pk.calc.half.life`).
`lambda.z`	The elimination rate (from the half life calculation, used to extrapolate beyond the maximum time observed).
`clast.pred`	To use predicted as opposed to observed Clast, either give the value for clast.pred here or set it to true (for automatic calculation from the half-life).

`tlast` The time of last observed concentration above the limit of quantificaiton. This is calculated if not provided.

`additional.times`
Times to include in the final outputs in addition to the standard times (see details). All `min(additional.times)` must be >= 0, and the `max(additional.times)` must be <= `tau`.

`check.blq` Must the first concentration measurement be below the limit of quantification?

`interp.method` See `interp.extrap.conc`

`extrap.method` See `interp.extrap.conc`

`steady.state.tol`
The tolerance for assessing if steady-state has been achieved (between 0 and 1, exclusive).

Details

The returned superposition times will include all of the following times: 0 (zero), `dose.times`, `time modulo tau` (shifting `time` for each dose time as well), `additional.times`, and `tau`.

Value

A data frame with columns named "conc" and "time".

See Also

`interp.extrap.conc`

`tss.monoexponential.generate.formula`

A helper function to generate the formula and starting values for the parameters in monoexponential models.

Description

A helper function to generate the formula and starting values for the parameters in monoexponential models.

Usage

```
tss.monoexponential.generate.formula(data)
```

Arguments

`data` The data used for the model

Value

a list with elements for each of the variables

Index