Package ‘PKNCA’

June 1, 2020

Type Package
Title Perform Pharmacokinetic Non-Compartmental Analysis
Version 0.9.4
Imports dplyr (>= 0.5.0), digest, nlme, parallel, rlang, stats, tidyr, utils
Suggests covr, cowplot, ggplot2, knitr, rmarkdown, testthat
Description Compute standard Non-Compartmental Analysis (NCA) parameters for typical pharmacokinetic analyses and summarize them.
License AGPL-3
URL https://github.com/billdenney/pknca
BugReports https://github.com/billdenney/pknca/issues
NeedsCompilation no
VignetteBuilder knitr
RoxygenNote 7.1.0
Encoding UTF-8
Author Bill Denney [aut, cre] (<https://orcid.org/0000-0002-5759-428X>), Clare Buckeridge [aut], Sridhar Duvvuri [ctb]
Maintainer Bill Denney <wdenney@humanpredictions.com>
Repository CRAN
Date/Publication 2020-06-01 17:00:02 UTC

R topics documented:

  add.interval.col .................................................. 4
  addProvenance .................................................... 6
  adj.r.squared ..................................................... 7
  AIC.list ............................................................ 7
  as.data.frame.PKNCAresults ..................................... 8
  business.mean ..................................................... 8
check.conc.time ................................................. 9
check.conversion .............................................. 10
check.interval.deps ......................................... 11
check.interval.specification ................................. 11
checkProvenance ............................................. 12
choose.auc.intervals .......................................... 13
clean.conc.blq ................................................ 14
clean.conc.na .................................................. 15
exclude .......................................................... 16
exclude_nca ...................................................... 17
find.tau .......................................................... 18
findOperator ..................................................... 19
fit_half_life ..................................................... 19
formula.parseFormula .......................................... 20
formula.PKNCACconc ........................................... 21
geomean .......................................................... 21
get.best.model ................................................... 22
get.first.model .................................................. 23
get.interval.cols .............................................. 23
get.parameter.deps ............................................ 24
getAttributeColumn ............................................ 24
columnValueOrNot .............................................. 25
data.PKNCACconc ............................................... 25
data.PKNCAdata .................................................. 26
data.PKNCACresults ............................................ 26
dataName.PKNCACconc .......................................... 27
gDepVar .......................................................... 28
gGroups.PKNCACconc ........................................... 28
gIndepVar ........................................................ 29
interp.extrap.conc ............................................. 30
merge.splitlist .................................................. 33
model.frame.PKNCACconc ....................................... 33
normalize_exclude ............................................. 34
parseFormula ..................................................... 34
pk.business ..................................................... 35
pk.calc.ae ....................................................... 36
pk.calc.aucint .................................................. 37
pk.calc.aucpext ............................................... 39
pk.calc.auxc .................................................... 40
pk.calc.c0 ....................................................... 43
pk.calc.cav ..................................................... 44
pk.calc.ceoi ..................................................... 45
pk.calc.cl ......................................................... 45
pk.calc.clast.obs .............................................. 46
pk.calc.clr ....................................................... 46
pk.calc.cmax ..................................................... 47
pk.calc.ctrrough ............................................... 48
pk.calc.deg.fluc ............................................... 48
R topics documented:

pk.calc.dn .................................................. 49
pk.calc.f .................................................. 49
pk.calc.fe ............................................... 50
pk.calc.half.life ......................................... 50
pk.calc.kel ............................................... 52
pk.calc.mrt ............................................... 53
pk.calc.mrt.md ......................................... 54
pk.calc.ptr ............................................... 54
pk.calc.swing ........................................... 55
pk.calc.thalf.eff ......................................... 55
pk.calc.tlag ............................................. 56
pk.calc.tlast ........................................... 56
pk.calc.tmax ............................................ 57
pk.calc.vd ............................................... 58
pk.calc.vss ............................................. 58
pk.calc.vz ............................................... 59
pk.nca .................................................... 59
pk.nca.interval ........................................ 60
pk.tss ...................................................... 62
pk.tss.data.prep ........................................ 62
pk.tss.monoexponential ................................ 63
pk.tss.monoexponential.individual ..................... 64
pk.tss.monoexponential.population ....................... 65
pk.tss.stepwise.linear .................................. 66
PKNCA ....................................................... 67
PKNCA.choose.option .................................... 67
PKNCA.options ............................................ 68
PKNCA.options.describe .................................. 69
PKNCA.set.summary ....................................... 70
PKNCAconc .................................................. 71
PKNCAdata ................................................ 72
PKNCAdose ................................................ 74
PKNCAresults ............................................. 75
print.PKNCAconc ........................................ 76
print.PKNCAdata ......................................... 76
print.provenance ......................................... 77
print.summary_PKNCAresults ............................. 77
roundingSummarize ...................................... 78
roundString .............................................. 78
setAttributeColumn ...................................... 79
setDuration .............................................. 80
setExcludeColumn ....................................... 81
setRoute ................................................... 81
signifString ............................................. 82
sort.interval.cols ....................................... 83
split.PKNCAconc ......................................... 83
summary.PKNCAdata ...................................... 84
summary.PKNCAresults .................................. 84
add.interval.col

Description
Add columns for calculations within PKNCA intervals

Usage
add.interval.col(
    name,
    FUN,
    values = c(FALSE, TRUE),
    depends = c(),
    desc = "",
    formalsmap = list(),
    datatype = c("interval", "individual", "population")
)

Arguments
name The column name as a character string
FUN The function to run (as a character string) or NA if the parameter is automatically calculated when calculating another parameter.
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)
formalsmap A named list mapping parameter names in the function call to NCA parameter names. See the details for information on use of formalsmap.
datatype The type of data used for the calculation

Details
The formalsmap argument enables mapping some alternate formal argument names to parameters. It is used to generalize functions that may use multiple similar arguments (such as the variants of mean residence time). The names of the list should correspond to function formal parameter names and the values should be one of the following:

- For the current interval:
character strings of NCA parameter name  The value of the parameter calculated for the current interval.

"conc"  Concentration measurements for the current interval.
"time"  Times associated with concentration measurements for the current interval (values start at 0 at the beginning of the current interval).
"volume"  Volume associated with concentration measurements for the current interval (typically applies for excretion parameters like urine).
"duration.conc"  Durations associated with concentration measurements for the current interval.
"dose"  Dose amounts associated with the current interval.
"time.dose"  Time of dose start associated with the current interval (values start at 0 at the beginning of the current interval).
"duration.dose"  Duration of dose (typically infusion duration) for doses in the current interval.
"route"  Route of dosing for the current interval.
"start"  Time of interval start.
"end"  Time of interval end.
"options"  PKNCA.options governing calculations.

- For the current group:
  "conc.group"  Concentration measurements for the current group.
  "time.group"  Times associated with concentration measurements for the current group (values start at 0 at the beginning of the current interval).
  "volume.group"  Volume associated with concentration measurements for the current interval (typically applies for excretion parameters like urine).
  "duration.conc.group"  Durations associated with concentration measurements for the current group.
  "dose.group"  Dose amounts associated with the current group.
  "time.dose.group"  Time of dose start associated with the current group (values start at 0 at the beginning of the current interval).
  "duration.dose.group"  Duration of dose (typically infusion duration) for doses in the current group.
  "route.group"  Route of dosing for the current group.

Value

NULL (Calling this function has a side effect of changing the available intervals for calculations)

See Also

Other Interval specifications: check.interval.deps(), check.interval.specification(), choose.auc.intervals(), get.interval.cols(), get.parameter.deps()

Examples

```R
## Not run:
add.interval.col("cmax",
```
addProvenance

Add a hash and associated information to enable checking object provenance.

Description

Add a hash and associated information to enable checking object provenance.

Usage

addProvenance(object, replace = FALSE)

Arguments

object The object to add provenance
replace Replace provenance if the object already has a provenance attribute. (If the object already has provenance and replace is FALSE, then an error will be raised.)

Value

The object with provenance as an added item

See Also

checkProvenance
adj.r.squared

| 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

| 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
- ...: parameters passed to the underlying AIC function (typically the parameter k)
- assess.best: determine which model is the best (by lowest AIC)

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

give.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, ..., out.format = c("long", "wide"))

Arguments

x The object to extract results from
...
out.format Should the output be 'long' (default) or 'wide'?

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, ...)
business.sd(x, ...)
business.cv(x, ...)
business.geomean(x, ...)
business.geocv(x, ...)
business.min(x, ...)
check.conc.time

business.max(x, ...)  
business.median(x, ...)  
business.range(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

Functions

• business.sd: Compute the standard deviation with business rules.
• business.cv: Compute the coefficient of variation with business rules.
• business.geomean: Compute the geometric mean with business rules.
• business.geocv: Compute the geometric coefficient of variation with business rules.
• business.min: Compute the minimum with business rules.
• business.max: Compute the maximum with business rules.
• business.median: Compute the median with business rules.
• business.range: Compute the range with business rules.

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

• time is not a number
• conc is not a number
• 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 fulfill 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 fulfill all dependencies.

Usage

check.interval.deps(x)

Arguments

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

Value

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

See Also

Other Interval specifications: add.interval.col(), check.interval.specification(), choose.auc.intervals(), get.interval.cols(), get.parameter.deps()

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

start and end time must always be given as columns, and the start must be before the end. Other columns define the parameters to be calculated and the groupings to apply the intervals to.

Value

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

See Also

The vignette "Selection of Calculation Intervals"

Other Interval specifications: add.interval.col(), check.interval.deps(), choose.auc.intervals(), get.interval.cols(), get.parameter.deps()

checkProvenance

Check the hash of an object to confirm its provenance.

Description

Check the hash of an object to confirm its provenance.

Usage

checkProvenance(object)

Arguments

object The object to check provenance for

Value

TRUE if the provenance is confirmed to be consistent, FALSE if the provenance is not consistent, or NA if provenance is not present.

See Also

addProvenance
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.aucs")
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 > τ 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 = NULL
)

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, PKNCA.options
Other Interval specifications: add.interval.col(), check.interval.deps(), check.interval.specification(), get.interval.cols(), get.parameter.deps()
Other Interval determination: find.tau()
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 = NULL,
  conc.na = NULL,
  check = TRUE
)

Arguments

conc  Measured concentrations
time  Time of the concentration measurement
...   Additional arguments passed to clean.conc.na
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?

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
Other Data cleaners: 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 = NULL, check = TRUE)
```

**Arguments**
- `conc` Measured concentrations
- `time` Time of the concentration measurement
- `...` Additional items to add to the data frame
- `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?

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

**See Also**
Other Data cleaners: clean.conc.blq()
exclude

Exclude data points or results from calculations or summarization.

Description

Exclude data points or results from calculations or summarization.

Usage

```r
exclude(object, reason, mask, FUN)
```

## Default S3 method:
```r
exclude(object, reason, mask, FUN)
```

Arguments

- **object**: The object to exclude data from.
- **reason**: The reason to add as a reason for exclusion.
- **mask**: A logical vector or numeric index of values to exclude (see details).
- **FUN**: A function to operate on the data (one group at a time) to select reasons for exclusions (see details).

Details

Only one of `mask` or `FUN` may be given. If `FUN` is given, it will be called with two arguments: a data.frame (or similar object) that consists of a single group of the data and the full object (e.g. the PKNCAconc object), `FUN(current_group, object)`, and it must return a logical vector equivalent to `mask` or a character vector with the reason text given when data should be excluded or `NA_character_` when the data should be included (for the current exclusion test).

Value

The object with updated information in the exclude column. The exclude column will contain the reason if `mask` or `FUN` indicate. If a previous reason for exclusion was given, then subsequent reasons for exclusion will be added to the first with a semicolon space ("; ") separator.

Methods (by class)

- default: The general case for data exclusion

See Also

Other Result exclusions: `exclude_nca`
**Examples**

myconc <- PKNCAconc(data.frame(subject=1, 
                         time=0:6, 
                         conc=c(1, 2, 3, 2, 1, 0.5, 0.25)), 
                         conc=time|subject)

exclude(myconc, 
        reason="Carryover", 
        mask=c(TRUE, rep(FALSE, 6))

---

**exclude_nca**

*Exclude NCA parameters based on examining the parameter set.*

**Description**

Exclude NCA parameters based on examining the parameter set.

**Usage**

```r
exclude_nca_span.ratio(min.span.ratio)
exclude_nca_max.aucinf.pext(max.aucinf.pext)
exclude_nca_min.hl.r.squared(min.hl.r.squared)
```

**Arguments**

- `min.span.ratio`  The minimum acceptable span ratio (uses PKNCA.options("min.span.ratio") if not provided).
- `max.aucinf.pext`  The maximum acceptable percent AUC extrapolation (uses PKNCA.options("max.aucinf.pext") if not provided).
- `min.hl.r.squared`  The minimum acceptable r-squared value for half-life (uses PKNCA.options("min.hl.r.squared") if not provided).

**Functions**

- `exclude_nca_span.ratio`: Exclude based on span ratio
- `exclude_nca_max.aucinf.pext`: Exclude based on AUC percent extrapolated (both observed and predicted)
- `exclude_nca_min.hl.r.squared`: Exclude based on half-life r-squared

**See Also**

Other Result exclusions: `exclude()`
Examples

```r
my_conc <- PKNCAconc(data.frame(conc=1.1^(3:0),
                         time=0:3,
                         subject=1),
                         conc~time|subject)
my_data <- PKNCAdata(my_conc,
                     intervals=data.frame(start=0, end=Inf,
                                          aucinf.obs=TRUE,
                                          aucpext.obs=TRUE))
my_result <- pk.nca(my_data)
my_result_excluded <- exclude(my_result,
                              FUN=exclude_nca_max.aucinf.pext())
as.data.frame(my_result_excluded)
```

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

```r
find.tau(x, na.action = stats::na.omit, options = list(), tau.choices = NULL)
```

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.

See Also

Other Interval determination: choose.auc.intervals()
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**

```r
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.

**See Also**

Other Formula parsing: `formula.parseFormula()`, `parseFormula()`

---

fit_half_life

Perform the half-life fit given the data. The function simply fits the data without any validation. No selection of points or any other components are done.

**Description**

Perform the half-life fit given the data. The function simply fits the data without any validation. No selection of points or any other components are done.

**Usage**

```r
fit_half_life(data, tlast)
```

**Arguments**

- `data`: The data to fit. Must have two columns named "log_conc" and "time"
- `tlast`: The time of last observed concentration above the limit of quantification.
Value

A data.frame with one row and columns named "r.squared", "adj.r.squared", "PROB", "lambda.z", "clast.pred", "lambda.z.n.points", "half.life", "span.ratio"

See Also

pk.calc.half.life

---

formular.parseFormula \textit{Convert the parsed formula back into the original}

Description

Convert the parsed formula back into the original

Usage

\begin{verbatim}
## S3 method for class 'parseFormula'
formula(x, drop.groups = FALSE, drop.lhs = FALSE, ...)
\end{verbatim}

Arguments

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

Value

A formula (optionally with portions removed)

See Also

Other Formula parsing: \texttt{findOperator()}, \texttt{parseFormula()}
**formula.PKNCAconc**

Extract the formula from a PKNCAconc object.

### Description

Extract the formula from a PKNCAconc object.

### Usage

```r
## 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

```r
geomean(x, na.rm = FALSE)
geosd(x, na.rm = FALSE)
geocv(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.

Functions

- `geosd`: Compute the geometric standard deviation, \( \exp(\text{sd}(\log(x))) \).
- `geocv`: Compute the geometric coefficient of variation, \( \sqrt{\exp(\text{sd}(\log(x))^2)-1}100 \).

References


Examples

```r
geomean(1:3)
geosd(1:3)
geocv(1:3)
```

get.best.model

```
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()

Value
A list with named elements for each parameter. Each list element contains the parameter definition.

See Also
check.interval.specification() and the vignette "Selection of Calculation Intervals"
Other Interval specifications: add.interval.col(), check.interval.deps(), check.interval.specification(), choose.auc.intervals(), get.parameter.deps()

Examples
get.interval.cols()
**getAttributeColumn**

---

**getAttributeColumn**

Retrieves the value of an attribute column.

**Description**

Retrieve the value of an attribute column.

**Usage**

```
getAttributeColumn(object, attr_name, warn_missing = c("attr", "column"))
```

**Arguments**

- `object` The object to extract the attribute value from.
- `attr_name` The name of the attribute to extract.
- `warn_missing` Give a warning if the "attribute" or "column" is missing. Character vector with zero, one, or both of "attr" and "column".

**Value**

The value of the attribute (or NULL if the attribute is not set or the column does not exist).

---

**get.parameter.deps**

Get all columns that depend on a parameter

**Description**

Get all columns that depend on a parameter

**Usage**

```
get.parameter.deps(x)
```

**Arguments**

- `x` The parameter name (as a character string)

**Value**

A character vector of parameter names that depend on the parameter `x`. If none depend on `x`, then the result will be an empty vector.

**See Also**

Other Interval specifications: `add.interval.col()`, `check.interval.deps()`, `check.interval.specification()`, `choose.auc.intervals()`, `get.interval.cols()`
**getColumnValueOrNot**

Get the value from a column in a data frame if the value is a column there, otherwise, the value should be a scalar or the length of the data.

**Description**

Get the value from a column in a data frame if the value is a column there, otherwise, the value should be a scalar or the length of the data.

**Usage**

```r
getColumnValueOrNot(data, value, prefix = "X")
```

**Arguments**

- `data`: A data.frame or similar object
- `value`: A character string giving the name of a column in the data, a scalar, or a vector the same length as the data
- `prefix`: The prefix to use if a column must be added (it will be used as the full column name if it is not already in the dataset or it will be prepended to the maximum column name if not.)

**Value**

A list with elements named "data", "name" giving the data with a column named "name" with the value in that column.

---

**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**

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

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

**Arguments**

- `object`: R object to extract the data from.
**getData.PKNCAdata**  
Extract all the original data from a PKNCAconc or PKNCAdose object

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

**Usage**
```r
## S3 method for class 'PKNCAdata'
getData(object)
```

**Arguments**
- object: R object to extract the data from.

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

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

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

**Arguments**
- object: R object to extract the data from.
getDataName.PKNCAconc

`getDataName.PKNCAconc`  
*Get the name of the element containing the data for the current object.*

### Description

Get the name of the element containing the data for the current object.

### Usage

```r
## S3 method for class 'PKNCAconc'
ggetDataName(object)

## S3 method for class 'PKNCAdata'
ggetDataName(object)

## S3 method for class 'PKNCAdose'
ggetDataName(object)

## S3 method for class 'PKNCAresults'
ggetDataName(object)

## Default S3 method:
ggetDataName(object)
```

### Arguments

- **object**
  
The object to get the data name from.

### Value

A character scalar with the name of the data object (or NULL if the method does not apply).

### Methods (by class)

- **default**: If no data name exists, returns NULL.

### See Also

Other PKNCA object extractors: `getDepVar()`, `getIndepVar()`
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
...

Value

The vector of the dependent variable from the object.

See Also

Other PKNCA object extractors: getDatName.PKNCAconc(), getIndepVar()

groups.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'
groups(object, form = formula(object), level, data = getData(object), sep)

## S3 method for class 'PKNCAdose'
groups(object)

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

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)
... Arguments passed to other getGroups functions

Value

A data frame with the (selected) group columns.

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
...

Value

The vector of the independent variable from the object.

See Also

Other PKNCA object extractors: getDataName.PKNCAconc(), getDepVar()
Interpolate concentrations between measurements or extrapolate concentrations after the last measurement.

**Description**

`interp.extrap.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**

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

```r
interpolate.conc(
  conc,  
  time,  
  time.out,  
  options = list(),  
  interp.method = NULL,  
  conc.blq = NULL,  
  conc.na = NULL,  
  conc.origin = 0,  
  ...,
  check = TRUE
)
```

```r
extrapolate.conc(
  conc,  
  time,  
  time.out,  
  lambda.z = NA,
```

interp.extrap.conc

clast = pk.calc.clast.obs(conc, time),
extrap.method = "AUCinf",
options = list(),
conc.na = NULL,
conc.blq = NULL,
...
check = TRUE
)

interp.extrap.conc.dose(
conc,
time,
time.dose,
route.dose = "extravascular",
duration.dose = NA,
time.out,
out.after = FALSE,
options = list(),
conc.blq = NULL,
conc.na = NULL,
...
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.
... Additional arguments passed to interpolate.conc() or extrapolate.conc().
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()?
conc.origin The concentration before the first measurement. conc.origin is typically used to set predose values to zero (default), set a predose concentration for endogenous compounds, or set predose concentrations to NA if otherwise unknown.
time.dose Time of the dose
route.dose  What is the route of administration ("intravascular" or "extravascular"). See the details for how this parameter is used.

duration.dose  What is the duration of administration? See the details for how this parameter is used.

out.after  Should interpolation occur from the data before (FALSE) or after (TRUE) the interpolated point? See the details for how this parameter is used. It only has a meaningful effect at the instant of an IV bolus dose.

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.

duration.dose and direction.out are ignored if route.dose == "extravascular", direction.out is ignored if duration.dose > 0.

route.dose and duration.dose affect how interpolation/extrapolation of the concentration occurs at the time of dosing. If route.dose == "intravascular" and duration.dose == 0 then extrapolation occurs for an IV bolus using \texttt{pk.calc.c0()}; with the data after dosing. Otherwise (either route.dose == "extravascular" or duration.dose > 0), extrapolation occurs using the concentrations before dosing and estimating the half-life (or more precisely, estimating lambda.z). Finally, direction.out can change the direction of interpolation in cases with route.dose == "intravascular" and duration.dose == 0. When direction.out == "before" interpolation occurs only with data before the dose (as is the case for route.dose == "extravascular"), but if direction.out == "after" interpolation occurs from the data after dosing.

Value

The interpolated or extrapolated concentration value as a scalar double (or vector for \texttt{interp.extrap.conc()}).

Functions

- \texttt{interpolate.conc}: Interpolate concentrations through Tlast (inclusive)
- \texttt{extrapolate.conc}: Extrapolate concentrations after Tlast
- \texttt{interp.extrap.conc.dose}: Interpolate and extrapolate concentrations without interpolating or extrapolating beyond doses.

See Also

\texttt{pk.calc.clast.obs()}, \texttt{pk.calc.half.life()}, \texttt{pk.calc.c0()}

merge.splitlist

Merge two or more lists with a data.frame 'groupid' attribute defining the matching.

Description

Merge two or more lists with a data.frame 'groupid' attribute defining the matching.

Usage

```r
## S3 method for class 'splitlist'
merge(...)
```

Arguments

... lists with 'groupid' attributes

Details

The merge is equivalent to a full_join where items missing from one or the other item will be missing, but the element(s) will exist.

Value

A list of lists with elements for the matching items between the 'groupid’s of each of the input lists. The output list will have a new 'groupid' attribute added with additional columns to indicate the of each input to its output location. If the inputs are named, then each list item will be named the same as the input name.

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

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

```r
## 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.

---

`normalize_exclude`  
*Normalize the exclude column by setting blanks to NA*

Description

Normalize the exclude column by setting blanks to NA

Usage

`normalize_exclude(object)`

Arguments

- `object`: The object to extract the exclude column from

Value

The exclude vector where NA indicates not to exclude and anything else indicates to exclude.

---

`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

See Also

Other Formula parsing: `findOperator()`, `formula.parseFormula()`

Examples

```r
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.

Usage

```r
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.

Examples

```r
my_mean <- pk.business(FUN=mean)
mean(c(1:3, NA))
# Less than half missing results in the summary statistic of the available # values.
my_mean(c(1:3, NA))
# More than half missing results in a missing value
my_mean(c(1:3, rep(NA, 4)))
```

---

**pk.calc.ae**

*Calculate amount excreted (typically in urine or feces)*

Description

Calculate amount excreted (typically in urine or feces)

Usage

```r
pk.calc.ae(conc, volume, check = TRUE)
```

Arguments

- `conc`: The concentration in the sample
- `volume`: The volume (or mass) of the sample
- `check`: Should the concentration and volume data be checked?

Details

```r
ae is sum(conc*volume).
```

The units for the concentration and volume should match such that `sum(conc*volume)` has units of mass or moles.

Value

The amount excreted during the interval

See Also

`pk.calc.clr, pk.calc.fe`
Calculate the AUC over an interval with interpolation and/or extrapolation of concentrations for the beginning and end of the interval.

Description

Calculate the AUC over an interval with interpolation and/or extrapolation of concentrations for the beginning and end of the interval.

Usage

pk.calc.aucint(
  conc,
  time,
  interval = NULL,
  start = NULL,
  end = NULL,
  clast = pk.calc.clast.obs(conc, time),
  lambda.z = NA,
  time.dose = NULL,
  route = "extravascular",
  duration.dose = 0,
  method = NULL,
  auc.type = "AUClast",
  conc.blq = NULL,
  conc.na = NULL,
  check = TRUE,
  ...
)

pk.calc.aucint.last(
  conc,
  time,
  start = NULL,
  end = NULL,
  time.dose,
  ...
)

pk.calc.aucint.all(
  conc,
  time,
  start = NULL,
  end = NULL,
  time.dose,
pk.calc.aucint

pk.calc.aucint.inf.obs(
conc,  
start = NULL,  
end = NULL,  
time.dose,  
lambda.z,  
clast.obs,
...
)

pk.calc.aucint.inf.pred(
conc,  
start = NULL,  
end = NULL,  
time.dose,  
lambda.z,  
clast.pred,
...
)

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
  start, end    The start and end of the interval (cannot be given if interval is given)
  clast, clast.obs, clast.pred
                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
  time.dose, route, duration.dose
                The time of doses, route of administration, and duration of dose used with inter-
                polation and extrapolation of concentration data (see interp.extrap.conc.dose).
                If NULL, interp.extrap.conc will be used instead (assuming that no doses af-
                fecting concentrations are in the interval).
  method        The method for integration (either 'lin up/log down' or 'linear')
  auc.type      The type of AUC to compute. Choices are 'AUCinf', 'AUClast', and 'AUCall'.

pk.calc.aucint
How to handle BLQ values in between the first and last above LOQ concentrations. (See `clean.conc.blq` for usage instructions.)

How to handle missing concentration values. (See `clean.conc.na` for usage instructions.)

Run `check.conc.time`, `clean.conc.blq`, and `clean.conc.na`?

Additional arguments passed to `pk.calc.auxc` and `interp.extrap.conc` options

List of changes to the default `PKNCA.options` for calculations.

**Functions**

- `pk.calc.aucint.last`: Interpolate or extrapolate concentrations for AUClast
- `pk.calc.aucint.all`: Interpolate or extrapolate concentrations for AUCall
- `pk.calc.aucint.inf.obs`: Interpolate or extrapolate concentrations for AUCinf.obs
- `pk.calc.aucint.inf.pred`: Interpolate or extrapolate concentrations for AUCinf.pred

**See Also**

`PKNCA.options`, `interp.extrap.conc.dose`

Other AUC calculations: `pk.calc.auxc()`

---

**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

**Details**

\[ \text{aucpext} = 100 \times (1 - \frac{\text{auclast}}{\text{aucinf}}). \]

**Value**

The numeric value of the AUC percent extrapolated or `NA_real_` if any of the following are true: `is.na(aucinf)`, `is.na(auclast)`, `aucinf <= 0`, or `auclast <= 0`. 
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 = NULL,
  conc.blq = NULL,
  conc.na = NULL,
  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.inf.obs(conc, time, clast.obs, ..., options = list(), lambda.z)

pk.calc.auc.inf.pred(conc, time, clast.pred, ..., 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.inf.obs(conc, time, clast.obs, ..., options = list(), lambda.z)
pk.calc.auxc 41

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

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

Arguments

conc                  Concentration measured

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, clast.obs, clast.pred
                       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 concentra-
                       tions. (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.

If all conc input are zero, then the AU(M)C is zero.

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.inf.obs`: Compute the AUCinf with the observed Clast.
- `pk.calc.auc.inf.pred`: Compute the AUCinf with the predicted Clast.
- `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.inf.obs`: Compute the AUMCinf with the observed Clast.
- `pk.calc.aumc.inf.pred`: Compute the AUMCinf with the predicted Clast.
- `pk.calc.aumc.all`: Compute the AUMCall.

References


See Also

`clean.conc.blq`

Other AUC calculations: `pk.calc.aucint()`

Examples

```r
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))
```
Estimate the concentration at dosing time for an IV bolus dose.

Usage

pk.calc.c0(conc, time, time.dose = 0, method = c("c0", "logslope", "c1", "cmin", "set0"), check = TRUE)

pk.calc.c0.method.logslope(conc, time, time.dose = 0, check = TRUE)

pk.calc.c0.method.c0(conc, time, time.dose = 0, check = TRUE)

pk.calc.c0.method.c1(conc, time, time.dose = 0, check = TRUE)

pk.calc.c0.method.set0(conc, time, time.dose = 0, check = TRUE)

pk.calc.c0.method.cmin(conc, time, time.dose = 0, check = TRUE)

Arguments

conc The observed concentrations

time The observed times

time.dose The time when dosing occurred

method The order of methods to test (see details)

check Check the conc and time inputs

Details

Methods available for interpolation are below, and each has its own specific function.

c0 If the observed conc at time.dose is nonzero, return that. This method should usually be used first for single-dose IV bolus data in case nominal time zero is measured.

logslope Compute the semilog line between the first two measured times, and use that line to extrapolate backward to time.dose

c1 Use the first point after time.dose

cmin Set c0 to cmin during the interval. This method should usually be used for multiple-dose oral data and IV infusion data.
set0  Set c0 to zero (regardless of any other data). This method should usually be used first for single-dose oral data.

Value

The estimated concentration at time 0.

Functions

- `pk.calc.c0.method.logslope`: Semilog regress the first and second points after time.dose. This method will return NA if the second conc after time.dose is 0 or greater than the first.
- `pk.calc.c0.method.c0`: Use \( C_0 = \text{conc}[\text{time}\text{.dose}] \) if it is nonzero.
- `pk.calc.c0.method.c1`: Use \( C_0 = C_1 \).
- `pk.calc.c0.method.set0`: Use \( C_0 = 0 \) (typically used for single dose oral and IV infusion)
- `pk.calc.c0.method.cmin`: Use \( C_0 = \text{Cmin} \) (typically used for multiple dose oral and IV infusion but not IV bolus)

---

**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

**Details**

\( \text{cav} = \frac{\text{auclast}}{(\text{end}-\text{start})}. \)

**Value**

The Cav (average concentration during the interval)
**pk.calc.ceoi**

*Determine the concentration at the end of infusion*

**Description**

Determine the concentration at the end of infusion

**Usage**

```r
pk.calc.ceoi(conc, time, duration.dose = NA, check = TRUE)
```

**Arguments**

- **conc**: Concentration measured
- **time**: Time of concentration measurement
- **duration.dose**: The duration for the dosing administration (typically from IV infusion)
- **check**: Run `check.conc.time`?

**Value**

The concentration at the end of the infusion, NA if duration.dose is NA, or NA if all time != duration.dose

---

**pk.calc.cl**

*Calculate the (observed oral) clearance*

**Description**

Calculate the (observed oral) clearance

**Usage**

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

**Arguments**

- **dose**: the dose administered
- **auc**: The area under the concentration-time curve.

**Details**

cl is dose/auc.

If dose is the same length as the other inputs, then the output will be the same length as all of the inputs; the function assumes that you are calculating for multiple intervals simultaneously. If the inputs other than dose are scalars and dose is a vector, then the function assumes multiple doses were given in a single interval, and the sum of the doses will be used for the calculation.
pk.calc.clr

Value

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

References


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.clr

Calculate renal clearance

Description

Calculate renal clearance

Usage

pk.calc.clr(ae, auc)

Arguments

ae | The amount excreted in urine (as a numeric scalar or vector)
auc | The area under the curve (as a numeric scalar or vector)
pk.calc.cmax

Details

clr is \( \text{sum(ae)/auc} \).
The units for the ae and auc should match such that \( \text{ae/auc} \) has units of volume/time.

Value

The renal clearance as a number

See Also

pk.calc.ae, pk.calc.fe

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, end)

**Arguments**
- **conc**: Observed concentrations during the interval
- **time**: Times of conc observations
- **end**: End time of the interval

**Value**
The concentration when time == end. If none match, then NA

---

pk.calc.deg.fluc  
*Determine the degree of fluctuation*

**Description**
Determine the degree of fluctuation

**Usage**
pk.calc.deg.fluc(cmax, cmin, cav)

**Arguments**
- **cmax**: The maximum observed concentration
- **cmin**: The minimum observed concentration
- **cav**: The average concentration in the interval

**Details**
deg.fluc is 100*(cmax - cmin)/cav.

**Value**
The degree of fluctuation around the average concentration.
**Description**

Determine dose normalized NCA parameter

**Usage**

\texttt{pk.calc.dn}(parameter, dose)

**Arguments**

- \texttt{parameter} Parameter to dose normalize
- \texttt{dose} Dose in units compatible with the area under the curve

**Value**

A number for dose normalized AUC

**Examples**

\texttt{pk.calc.dn}(90, 10)

---

**Description**

Calculate the absolute (or relative) bioavailability

**Usage**

\texttt{pk.calc.f}(dose1, auc1, dose2, auc2)

**Arguments**

- \texttt{dose1} The dose administered in route or method 1
- \texttt{auc1} The AUC from 0 to infinity or 0 to tau administered in route or method 1
- \texttt{dose2} The dose administered in route or method 2
- \texttt{auc2} The AUC from 0 to infinity or 0 to tau administered in route or method 2

**Details**

\[ f = \frac{\text{auc2}/\text{dose2}}{(\text{auc1}/\text{dose1})}. \]
pk.calc.fe

*Calculate fraction excreted (typically in urine or feces)*

**Description**

Calculate fraction excreted (typically in urine or feces)

**Usage**

pk.calc.fe(ae, dose)

**Arguments**

- `ae` The amount excreted (as a numeric scalar or vector)
- `dose` The dose (as a numeric scalar or vector)

**Details**

fe is sum(ae)/dose

The units for `ae` and `dose` should be the same so that `ae/dose` is a unitless fraction.

**Value**

The fraction of dose excreted.

**See Also**

`pk.calc.ae, pk.calc.clr`

---

pk.calc.half.life

*Compute the half-life and associated parameters*

**Description**

The terminal elimination half-life is estimated from the final points in the concentration-time curve using semi-log regression (log(conc)~time) with automated selection of the points for calculation (unless manually.selected.points is TRUE).
Usage

pk.calc.half.life(
  conc,
  time,
  tmax,
  tlast,
  manually.selected.points = FALSE,
  options = list(),
  min.hl.points = NULL,
  adj.r.squared.factor = NULL,
  conc.blq = NULL,
  conc.na = NULL,
  first.tmax = NULL,
  allow.tmax.in.half.life = NULL,
  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)
	
   manually.selected.points
   Have the points being input been manually selected? The impact of setting this to TRUE is that no selection for the best points will be done. When TRUE, this option causes the options of adj.r.squared.factor, min.hl.points, and allow.tmax.in.half.life to be ignored.

toptions 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

If `manually.selected.points` is FALSE (default), the half-life is calculated by computing the best fit line for all points at or after tmax (based on the value of `allow.tmax.in.half.life`). The best half-life is chosen by the following rules in order:

- 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

If `manually.selected.points` is TRUE, the conc and time data are used as-is without any form of selection for the best-fit half-life.

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


---

**pk.calc.kel**

*Calculate the elimination rate (Kel)*

**Description**

Calculate the elimination rate (Kel)

**Usage**

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

Arguments

mrt the mean residence time
kel is 1/mrt, not to be confused with lambda.z.

Value

the numeric value of the elimination rate

Description

Calculate the mean residence time (MRT) for single-dose data or linear multiple-dose data.

Usage

pk.calc.mrt(auc, aumc)
pk.calc.mrt.iv(auc, aumc, duration.dose)

Arguments

auc the AUC from 0 to infinity or 0 to tau
aumc the AUMC from 0 to infinity or 0 to tau
duration.dose The duration of the dose (usually an infusion duration for an IV infusion)

Details

mrt is aumc/auc - duration.dose/2 where duration.dose = 0 for oral administration.

Value

the numeric value of the mean residence time

Functions

• pk.calc.mrt.iv: MRT for an IV infusion

See Also

pk.calc.mrt.md
Description

Calculate the mean residence time (MRT) for multiple-dose data with nonlinear kinetics.

Usage

pk.calc.mrt.md(auctau, aumctau, aucinf, tau)

Arguments

- `auctau`: the AUC from time 0 to the end of the dosing interval (tau).
- `aumctau`: the AUMC from time 0 to the end of the dosing interval (tau).
- `aucinf`: the AUC from time 0 to infinity (typically using single-dose data)
- `tau`: the dosing interval

Details

\[ \text{mrt.md} = \frac{\text{aumctau}}{\text{auctau}} + \tau \cdot \frac{\text{aucinf} - \text{auctau}}{\text{auctau}} \]

and should only be used for multiple dosing with equal intervals between doses.

Note that if \( \text{aucinf} = \text{auctau} \) (as would be the assumption with linear kinetics), the equation becomes the same as the single-dose MRT.

See Also

pk.calc.mrt

---

Description

Determine the peak-to-trough ratio

Usage

pk.calc.ptr(cmax, ctrough)

Arguments

- `cmax`: The maximum observed concentration
- `ctrough`: The last concentration in an interval
pk.calc.swing

Details
ptr is cmax/ctrough.

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

---

pk.calc.swing  

Determine the PK swing

Description
Determine the PK swing

Usage
pk.calc.swing(cmax, cmin)

Arguments

- cmax  The maximum observed concentration
- cmin  The minimum observed concentration

Details
swing is 100*(cmax - cmin)/cmin.

Value
The swing above the minimum concentration. If cmin is zero, then the result is infinity.

---

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 to infinity
Details

\[ \text{thalf.eff} = \log(2) \times mrt. \]

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

\( \text{NA} \) will be returned if all \( \text{conc} \) are \( \text{NA} \) or 0.

Usage

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

pk.calc.tfirstr(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.

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 = NULL, 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.vd  
*Calculate the volume of distribution (Vd) or observed volume of distribution (Vd/F)*

**Description**

Calculate the volume of distribution (Vd) or observed volume of distribution (Vd/F)

**Usage**

`pk.calc.vd(dose, aucinf, lambda.z)`

**Arguments**

- **dose**: One or more doses given during an interval
- **aucinf**: Area under the curve to infinity (either predicted or observed).
- **lambda.z**: Elimination rate constant

**Details**

\[ \text{vd} = \frac{\text{dose}}{\text{aucinf} \times \text{lambda.z}}. \]

If `dose` is the same length as the other inputs, then the output will be the same length as all of the inputs; the function assumes that you are calculating for multiple intervals simultaneously. If the inputs other than `dose` are scalars and `dose` is a vector, then the function assumes multiple doses were given in a single interval, and the sum of the doses will be used for the calculation.

**Value**

The observed volume of distribution

---

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
pk.calc.vz

Details
vss is cl*mrt.

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(cl, lambda.z)

Arguments
cl the clearance (or apparent observed clearance)
lambda.z the elimination rate

Details
vz is cl/lambda.z.

---

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

---

Description
The pk.nca function computes the NCA parameters from a PKNCAdata object. All options for the calculation and input data are set in prior functions (PKNCAconc, PKNCAdose, and PKNCAdata). Options for calculations are set either in PKNCAdata or with the current default options in PKNCA.options.

Usage
pk.nca(data, verbose = FALSE)

Arguments
data A PKNCAdata object
verbose Indicate, by message(), the current state of calculation.
Details

When performing calculations, all time results are relative to the start of the interval. For example, if an interval starts at 168 hours, ends at 192 hours, and the maximum concentration is at 169 hours, tmax=\(169-168=1\).

Value

A PKNCAresults object.

See Also

PKNCAdata, PKNCA.options, summary.PKNCAresults, as.data.frame.PKNCAresults, exclude

---

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,
  volume,
  duration.conc,
  dose,
  time.dose,
  duration.dose,
  route,
  conc.group = NULL,
  time.group = NULL,
  volume.group = NULL,
  duration.conc.group = NULL,
  dose.group = NULL,
  time.dose.group = NULL,
  duration.dose.group = NULL,
  route.group = NULL,
  include_half.life = NULL,
  exclude_half.life = NULL,
  interval,
  options = list()
)
Arguments

conc, conc.group
Concentration measured for the current interval or all data for the group

time, time.group
Time of concentration measurement for the current interval or all data for the group

volume, volume.group
The volume (or mass) of the concentration measurement for the current interval or all data for the group (typically for urine and fecal measurements)

duration.conc, duration.conc.group
The duration of the concentration measurement for the current interval or all data for the group (typically for urine and fecal measurements)

dose, dose.group
Dose amount (may be a scalar or vector) for the current interval or all data for the group

time.dose, time.dose.group
Time of the dose for the current interval or all data for the group (must be the same length as dose or dose.group)

duration.dose, duration.dose.group
The duration of the dose administration for the current interval or all data for the group (typically zero for extravascular and intravascular bolus and nonzero for intravascular infusion)

route, route.group
The route of dosing for the current interval or all data for the group

include_half.life
An optional boolean vector of the concentration measurements to include in the half-life calculation. If given, no half-life point selection will occur.

exclude_half.life
An optional boolean vector of the concentration measurements to exclude from the half-life calculation.

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 data prep

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

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

Value

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

See Also

Other Time to steady-state calculations: 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 = NULL,
  conc.na = NULL,
  check = TRUE,
  ...
)

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 steady-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

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.
Arguments

... See pk.tss.data.prep

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 (requires more than one individual; use single for one individual), 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.

Value

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

References


See Also

Other Time to steady-state calculations: pk.tss.stepwise.linear(), pk.tss()

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
)
pk.tss.monoexponential.population

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` 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 monoexponential time to steady-state._

Description

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

Usage

```r
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.
pk.tss.stepwise.linear

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

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.

Description

A linear slope is fit through the data to find when it becomes non-significant. Note that this is less preferred 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

... See pk.tss.data.prep
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

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

See Also

Other Time to steady-state calculations: `pk.tss.monoexponential()`, `pk.tss()`

---

**PKNCA**  
*Compute noncompartmental pharmacokinetics*

---

**Description**

Compute pharmacokinetic (PK) noncompartmental analysis (NCA) parameters.

**Details**

PKNCA has been cross-validated with both Phoenix WinNonlin(R) and Pumas (click here for the cross-validation article)

A common workflow would load data from a file or database into a data.frame then run the following code.

**Examples**

```r
## Not run:
# Load concentration-time data into a data.frame called d.conc
# with columns named "conc", "time", and "subject".
my.conc <- PKNCAconc(d.conc, conc~time|subject)
# Load dose-time data into a data.frame called d.dose
# with columns named "dose", "time", and "subject".
my.dose <- PKNCAdose(d.dose, dose~time|subject)
# Combine the concentration-time and dose-time data into an object
# ready for calculations.
my.data <- PKNCAdata(my.conc, my.dose)
# Perform the calculations
my.results <- pk.nca(my.data)
# Look at summary results
summary(my.results)
# Look at a listing of results
as.data.frame(my.results)

## End(Not run)
```

---

**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

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

Arguments

- `name`  The option name requested.
- `value` A value to check for the option (NULL to choose not to check the value).
- `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

Other PKNCA calculation and summary settings: `PKNCA.options()`, `PKNCA.set.summary()`

---

### 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**

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

**Arguments**

- `...` options to set or get the value for
- `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 (if NULL, ).

**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. See the vignette 'Options for Controlling PKNCA' for a current list of options.
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.options.describe`

Other PKNCA calculation and summary settings: `PKNCA.choose.option()`, `PKNCA.set.summary()`

Examples

```r
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)
```

---

### Description

Describe a `PKNCA.options` option by name.

### Usage

```r
PKNCA.options.describe(name)
```

### Arguments

- `name` The option name requested.

### Value

A character string of the description.

### See Also

- `PKNCA.options`
PKNCA.set.summary

Define how NCA parameters are summarized.

Description

Define how NCA parameters are summarized.

Usage

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

Arguments

name The parameter name or a vector of parameter names. It must have already been defined (see add.interval.col).
description A single-line description of the summary
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

Other PKNCA calculation and summary settings: PKNCA.choose.option(), PKNCA.options()
## Not run:
```
PKNCA.set.summary(
    name="half.life",
    description="arithmetic mean and standard deviation",
    point=business.mean,
    spread=business.sd,
    rounding=list(signif=3)
)
```
## End(Not run)

---

### Description

Create a PKNCAconc object

#### Usage

```
PKNCAconc(data, ...)
```

#### Arguments

- **data**: A data frame with concentration (or amount for urine/feces), time, and the groups defined in `formula`. 

---

)::

---
... Ignored.

**formula**
The formula defining the concentration-time|groups or amount-time|groups for urine/feces (In the remainder of the documentation, "concentration" will be used to describe concentration or amount.) One special aspect of the groups part of the formula is that the last group is typically assumed to be the subject; see the documentation for the subject argument for exceptions to this assumption.

**subject**
The column indicating the subject number. If not provided, this defaults to the beginning of the inner groups: For example with concentration-time|Study+Subject/Analyte, 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.

**time.nominal** (optional) The name of the nominal time column (if the main time variable is actual time. The time.nominal is not used during calculations; it is available to assist with data summary and checking.

**exclude** (optional) The name of a column with concentrations to exclude from calculations and summarization. If given, the column should have values of NA or "" for concentrations to include and non-empty text for concentrations to exclude.

**duration** (optional) The duration of collection as is typically used for concentration measurements in urine or feces.

**volume** (optional) The volume (or mass) of collection as is typically used for urine or feces measurements.

**exclude_half.life, include_half.life**
Points to exclude from the half-life calculation (still using normal selection rules for the other points) or to include for the half-life (using specifically those points and bypassing automatic point selection).

**Value**
A PNKCAconc object that can be used for automated NCA.

**See Also**
Other PNKCA objects: **PNKCAdata()**, **PNKCAdose()**, **PNKCAresults()**
**Usage**

```r
PKNCAdata(data.conc, data.dose, ...)  
```

## S3 method for class 'PKNCAconc'
`PKNCAdata(data.conc, data.dose, ...)`

## S3 method for class 'PKNCAdose'
`PKNCAdata(data.conc, data.dose, ...)`

## Default S3 method:
`PKNCAdata(
  data.conc,
  data.dose,
  ...,  
  formula.conc,
  formula.dose,
  intervals,
  options = list()
)
```

**Arguments**

- `data.conc`  Concentration data as a PKNCAconc object or a data frame
- `data.dose`  Dosing data as a PKNCAdose object (see details)
- `...`        arguments passed to `PKNCAdata.default`
- `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.
- `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`. (see details)
- `options`  List of changes to the default `PKNCA.options` for calculations.

**Details**

If `data.dose` is not given or is `NA`, then the intervals must be given. At least one of `data.dose` and intervals must be given.

**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).
PKNCAdose

Create a PKNCAdose object

Description

Create a PKNCAdose object

Usage

PKNCAdose(data, ...)

## Default S3 method:
PKNCAdose(data, ...)

## S3 method for class 'tbl_df'
PKNCAdose(data, ...)

## S3 method for class 'data.frame'
PKNCAdose(data, formula, route, rate, duration, time.nominal, exclude, ...)

Arguments

data A data frame with time and the groups defined in formula.

... Ignored.

formula The formula defining the dose.amount~time|groups where time is the time
of the dosing and dose.amount is the amount administered at that time (see
Details).

route Define the route of administration. The value may be either a column name
from the data (checked first) or a character string of either "extravascular"
or "intravascular" (checked second). If given as a column name, then every
value of the column must be either "extravascular" or "intravascular".

rate, duration (optional) for "intravascular" dosing, the rate or duration of dosing. If given
as a character string, it is the name of a column from the data, and if given as
a number, it is the value for all doses. Only one may be given, and if neither is
given, then the dose is assumed to be a bolus (duration=0). If rate is given,
then the dose amount must be given (the left hand side of the formula).

time.nominal (optional) The name of the nominal time column (if the main time variable is
actual time. The time.nominal is not used during calculations; it is available to
assist with data summary and checking.

exclude (optional) The name of a column with concentrations to exclude from calcula-
tions and summarization. If given, the column should have values of NA or ""
for concentrations to include and non-empty text for concentrations to exclude.
Details

The formula for a PKNCAdose object can be given three ways: one-sided (missing left side), one-sided (missing right side), or two-sided. Each of the three ways can be given with or without groups. When given one-sided missing the left side, the left side can either be omitted or can be given as a period (.): ~time|treatment+subject and .~time|treatment+subject are identical, and dose-related NCA parameters will all be reported as not calculable (for example, clearance). When given one-sided missing the right side, the right side must be specified as a period (.): dose~.|treatment+subject, and only a single row may be given per group. When the right side is missing, PKNCA assumes that the same dose is given in every interval. When given as a two-sided formula

Value

A PKNCAconc object that can be used for automated NCA.

See Also

Other PKNCA objects: PKNCAconc(), PKNCAdata(), PKNCAresults()

---

PKNCAresults  Generate a PKNCAresults object

Description

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

Usage

PKNCAresults(result, data, exclude)

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

exclude  (optional) The name of a column with concentrations to exclude from calculations and summarization. If given, the column should have values of NA or "" for concentrations to include and non-empty text for concentrations to exclude.

Value

A PKNCAresults object with each of the above within.

See Also

Other PKNCA objects: PKNCAconc(), PKNCAdata(), PKNAdose()
print.PKNCAconc  
Print and/or summarize a PKNCAconc or PKNCAdose object.

Description
Print and/or summarize a PKNCAconc or PKNCAdose object.

Usage
```r
## 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
- `...`  
  Arguments passed to `print.formula` and `print.data.frame`
- `object`  
  The object to summarize

print.PKNCAdose  
Print a PKNCAdose object

Description
Print a PKNCAdose object

Usage
```r
## S3 method for class 'PKNCAdose'
print(x, ...)  
```

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

Print the summary of a provenance object

Description

Print the summary of a provenance object

Usage

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

Arguments

x The object to be printed
... Ignored

Value

invisible text of the printed information

---

print.summary_PKNCAresults

Print the results summary

Description

Print the results summary

Usage

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

Arguments

x A summary_PKNCAresults object
... passed to print.data.frame (row.names is always set to FALSE)

Value

x invisibly

See Also

summary.PKNCAresults
roundingSummarize

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

Description

During the summarization of PKNCA results, 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, sci_range = Inf, sci_sep = "e", si_range)

Arguments

- x: The number to round
- digits: integer indicating the number of decimal places
- sci_range: See help for `signifString` (and you likely want to round with `signifString` if you want to use this argument)
- sci_sep: The separator to use for scientific notation strings (typically this will be either "e" or "x10^" for computer- or human-readable output).
- si_range: Deprecated, please use `sci_range`
setAttributeColumn

Details

Values that are not standard numbers like Inf, NA, and NaN are returned as "Inf", "NA", and NaN.

Value

A string with the value

See Also

round, signifString

setAttributeColumn  Add an attribute to an object where the attribute is added as a name to the names of the object.

Description

Add an attribute to an object where the attribute is added as a name to the names of the object.

Usage

setAttributeColumn(
  object,       
  attr_name,    
  col_or_value, 
  col_name,     
  default_value, 
  stop_if_default, 
  warn_if_default, 
  message_if_default
)

Arguments

object       The object to set the attribute column on.
attr_name    The attribute name to set
col_or_value If this exists as a column in the data, it is used as the col_name. If not, this becomes the default_value.
col_name     The name of the column within the dataset to use (if missing, uses attr_name)
default_value The value to fill in the column if the column does not exist (the column is filled with NA if it does not exist and no value is provided).
stop_if_default, warn_if_default, message_if_default
  A character string to provide as an error, a warning, or a message to the user if the default_value is used. They are tested in order (if stop, the code stops; if warning, the message is ignored; and message last).
Value

The object with the attribute column added to the data.

See Also

getAttributeColumn

---

**setDuration**  
*Set the duration of dosing or measurement*

### Description

Set the duration of dosing or measurement

### Usage

```r
setDuration(object, ...)
```

```r
## S3 method for class 'PKNCAdose'
setDuration(object, duration, rate, dose, ...)
```

### Arguments

- **object**  
  An object to set a duration on

- **...**  
  Arguments passed to another setDuration function

- **duration**  
  The value to set for the duration or the name of the column in the data to use for the duration.

- **rate**  
  (for PKNCAdose objects only) The rate of infusion

- **dose**  
  (for PKNCAdose objects only) The dose amount

### Value

The object with duration set
setExcludeColumn  
Set the exclude parameter on an object

Description

This function adds the exclude column to an object. To change the exclude value, use the exclude function.

Usage

setExcludeColumn(object, exclude, dataname = "data")

Arguments

object  The object to set the exclude column on.
exclude The column name to set as the exclude value.
dataname The name of the data.frame within the object to add the exclude column to.

Value

The object with an exclude column and attribute

setRoute  
Set the dosing route

Description

Set the dosing route

Usage

setRoute(object, ...)

## S3 method for class 'PKNCAdose'
setRoute(object, route, ...)

Arguments

object A PKNCAdose object
... Arguments passed to another setRoute function
route A character string indicating one of the following: the column from the data which indicates the route of administration, a scalar indicating the route of administration for all subjects, or a vector indicating the route of administration for each dose in the dataset.
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, ...)

## S3 method for class 'data.frame'

signifString(x, ...)

## Default S3 method:

signifString(x, digits = 6, sci_range = 6, sci_sep = "e", si_range, ...)

Arguments

x
The number to round

...  
Arguments passed to methods.

digits
integer indicating the number of significant digits

sci_range
integer (or Inf) indicating when to switch to scientific notation instead of floating point. Zero indicates always use scientific; Inf indicates to never use scientific notation; otherwise, scientific notation is used when abs(log10(x)) > si_range.

sci_sep
The separator to use for scientific notation strings (typically this will be either "e" or "x10^" for computer- or human-readable output).

si_range
Deprecated, please use sci_range

Details

Values that are not standard numbers like Inf, NA, and NaN are returned as "Inf", "NA", and NaN.

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

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

split.PKNCAconc  

Divide into groups

Description

split.PKNCAconc divides data into individual groups defined by `getGroups.PKNCAconc`.

Usage

```r
## S3 method for class 'PKNCAconc'
split(x, f = getGroups(x), drop = TRUE, ...)
## S3 method for class 'PKNCAdata'
split(x, ...)
## S3 method for class 'PKNCAdose'
split(x, f = getGroups(x), drop = TRUE, ...)
```

Arguments

- `x`  
  the object to split
- `f`  
  the groups to use for splitting the object
- `drop`  
  logical indicating if levels that do not occur should be dropped.
- `...`  
  Ignored.

Details

If `x` is NA then a list with NA as the only element and a "groupid" attribute of an empty data.frame is returned.

Value

A list of objects with an attribute of groupid consisting of a data.frame with columns for each group.
### 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**

```r
## 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**

```r
## S3 method for class 'PKNCAresults'
summary(
  object,
  ...,  
  drop.group = object$data$conc$subject,
  summarize.n.per.group = TRUE,
  not.requested.string = ",",
  not.calculated.string = "NC"
)
```

**Arguments**

- `object`: The results to summarize
- `...`: Ignored.
- `drop.group`: Which group(s) should be dropped from the formula?
```R
conc_obj <- PKNCAconc(as.data.frame(datasets::Theoph), conc~Time|Subject)
d_dose <- unique(datasets::Theoph[datasets::Theoph$Time == 0,
  c("Dose", "Time", "Subject")])
dose_obj <- PKNCAdose(d_dose, Dose~Time|Subject)
data_obj_automatic <- PKNCAdata(conc_obj, dose_obj)
results_obj_automatic <- pk.nca(data_obj_automatic)
# To get standard results run summary
summary(results_obj_automatic)
# To enable numeric conversion and extraction, do not give a spread function
# and subsequently run as.numeric on the result columns.
PKNCA.set.summary(
  name=c("auclast", "cmax", "half.life", "aucinf.obs"),
  point=business.geomean,
  description="geometric mean"
)
PKNCA.set.summary(
  name=c("tmax"),
  point=business.median,
  description="median"
)
summary(results_obj_automatic, not.requested.string="NA")
```
**superposition**  
*Compute noncompartmental superposition for repeated dosing*

**Description**

Compute noncompartmental superposition for repeated dosing

**Usage**

```
superposition(conc, ...)
```

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

```R
## 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 = NULL,
  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
The time of dosing within the dosing interval. The \( \min(\text{dose.times}) \) must be \( \geq 0 \), and the \( \max(\text{dose.times}) \) 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 \( \text{Inf} \) for steady-state.

options

The PKNCA.options to use for the calculation (passed on to subsequent functions like \( \text{pk.calc.half.life} \)).

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 quantification. This is calculated if not provided.

additional.times

Times to include in the final outputs in addition to the standard times (see details). All \( \min(\text{additional.times}) \) must be \( \geq 0 \), and the \( \max(\text{additional.times}) \) must be \( \leq \tau \).

check.blq

Must the first concentration measurement be below the limit of quantification?

interp.method

See \( \text{interp.extrap.conc} \)

extrap.method

See \( \text{interp.extrap.conc} \)

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

\( \text{interp.extrap.conc} \)
### time_calc

**Times relative to an event (typically dosing)**

#### Description

Times relative to an event (typically dosing)

#### Usage

```r
time_calc(time_event, time_obs, units = NULL)
```

#### Arguments

- **time_event**: A vector of times for events
- **time_obs**: A vector of times for observations
- **units**: Passed to `base::as.numeric.difftime()`

#### Value

A data.frame with columns for:

- **event_number_before**: The index of 'time_event' that is the last one before 'time_obs' or 'NA' if none are before.
- **event_number_after**: The index of 'time_event' that is the first one after 'time_obs' or 'NA' if none are after.
- **time_before**: The minimum time that the current 'time_obs' is before a 'time_event', 0 if at least one 'time_obs == time_event'.
- **time_after**: The minimum time that the current 'time_obs' is after a 'time_event', 0 if at least one 'time_obs == time_event'.
- **time_after_first**: The time after the first event (may be negative or positive).

'time_after' and 'time_before' are calculated if they are at the same time as a dose, they equal zero, and otherwise, they are calculated relative to the dose number in the 'event_number_*' columns.

### 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.
tss.monoexponential.generate.formula

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

add.interval.col, 4, 11–13, 23, 24, 70
addProvenance, 6, 12
adj.r.squared, 7
AIC.list, 7, 22
as.data.frame, 8
as.data.frame.PKNCAresults, 8, 60
business.cv (business.mean), 8
business.geocv (business.mean), 8
business.geomean (business.mean), 8
business.max (business.mean), 8
business.median (business.mean), 8
business.min (business.mean), 8
business.range (business.mean), 8
business.sd (business.mean), 8
check.conc.time, 9, 14, 15, 31, 39, 41, 45–47, 51, 57, 63
check.conversion, 10
check.interval.deps, 5, 11, 12, 13, 23, 24
check.interval.specification, 5, 11, 11, 13, 23, 24, 61, 73
checkProvenance, 6, 12
choose.auc.intervals, 5, 11, 12, 13, 18, 23, 24, 73, 74
clean.conc.blq, 14, 15, 31, 39, 41, 42, 51, 63
clean.conc.na, 14, 15, 15, 31, 39, 41, 42, 51, 63
exclude, 16, 17, 60, 81
exclude_nca, 16, 17
exclude_nca_max.aucinf.pext (exclude_nca), 17
exclude_nca_min.hl.r.squared (exclude_nca), 17
exclude_nca_span.ratio (exclude_nca), 17
extrapolate.conc (interp.extrap.conc), 30
find.tau, 13, 18
findOperator, 19, 20, 35
fit_half_life, 19
formula.parseFormula, 19, 20, 35
formula.PKNCAconc, 21
formula.PKNCAdose (formula.PKNCAconc), 21
geocv (geomean), 21
geomean, 21
gesd (geomean), 21
get.best.model, 7, 22
get.first.model, 23
get.interval.cols, 5, 11–13, 23, 24
get.parameter.deps, 5, 11–13, 23, 24
getAttributeColumn, 24, 80
columnValueOrNot, 25
data.PKNCAconc, 25
data.PKNCAdata, 26
data.PKNCAdose (data.PKNCAconc), 25
data.PKNCAresults, 26
dataName (dataName.PKNCAconc), 27
dataName.PKNCAconc, 27, 28, 29
data.PKNCAdose, 25
dataName.PKNCAconc, 27, 28, 29
dataName.PKNCAresults (dataName.PKNCAconc), 28
dataGroups.PKNCAconc, 28, 83
dataGroups.PKNCAdose (dataGroups.PKNCAconc), 28
dataGroups.PKNCAresults (dataGroups.PKNCAconc), 28
dataIndepVar, 27, 28, 29
head, 76
interp.extrap.conc, 30, 38, 87
interp.extrap.conc.dose, 38, 39
interpolate.conc (interp.extrap.conc), 30
merge.splitlist, 33
model.frame.PKNCAconc, 33
INDEX

model.frame.PKNCAdose
  (model.frame.PKNCAconc), 33

normalize_exclude, 34

options, 68

parseFormula, 19, 20, 34
pk.business, 9, 35
pk.calc.ae, 36, 47, 50
pk.calc.auc, 13
pk.calc.auc (pk.calc.auxc), 40
pk.calc.aucint, 37, 42
pk.calc.aucext, 39
pk.calc.auxc, 39, 40
pk.calc.c0, 32, 43
pk.calc.cav, 44
pk.calc.ceoi, 45
pk.calc.cl, 45
pk.calc.clast.obs, 31, 32, 46
pk.calc.clr, 36, 46, 50
pk.calc.cmax, 47
pk.calc.cmin (pk.calc.cmax), 47
pk.calc.ctrough, 48
pk.calc.deg.fluc, 48
pk.calc.dn, 49
pk.calc.f, 49
pk.calc.fe, 36, 47, 50
pk.calc.half.life, 13, 20, 32, 50
pk.calc.kel, 52
pk.calc.mrt, 53, 54
pk.calc.mrt.md, 53, 54
pk.calc.ptr, 54
pk.calc.swing, 55
pk.calc.tfirst (pk.calc.tlast), 56
pk.calc.thalf.eff, 55
pk.calc.tlag, 56
pk.calc.tlast, 56
pk.calc.tmax, 51, 57
pk.calc.vd, 58
pk.calc.vss, 58
pk.calc.vz, 59
pk.nca, 59, 74
pk.nca.interval, 60
pk.tss, 62, 64, 67
pk.tss.data.prep, 62, 62, 64–66
pk.tss.monoexponential, 62, 63, 67

pk.tss.monoexponential.individual, 64
pk.tss.monoexponential.population, 65
pk.tss.stepwise.linear, 62, 64, 66
PKNCA, 67
PKNCA.choose.option, 67, 69, 70
PKNCA.options, 13–15, 18, 31, 39, 41, 51, 57, 60, 61, 63, 68, 69, 70, 73
PKNCA.options.describe, 69, 69
PKNCA.set.summary, 68, 69, 70, 78, 85
PKNCAconc, 71, 74, 75
PKNCAdata, 60, 72, 72, 75
PKNCAdose, 72, 74, 74, 75
PKNCAresults, 72, 74, 75, 75
print.PKNCAconc, 76, 76
print.PKNCAdata, 76, 84
print.PKNCAdose, 76
print.PKNCAdose (print.PKNCAconc), 76
print.provenance, 77
print.summary_PKNCAresults, 77, 85

round, 79
roundingSummarize, 78
roundString, 78, 82

setAttributeColumn, 79
setDuration, 80
setExcludeColumn, 81
setRoute, 81
signif, 82
signifString, 78, 79, 82

sort.interval.cols, 83
split.PKNCAconc, 83
split.PKNCAdata (split.PKNCAconc), 83
split.PKNCAdose (split.PKNCAconc), 83
summary.PKNCAconc (print.PKNCAconc), 76
summary.PKNCAdata, 84
summary.PKNCAdose (print.PKNCAconc), 76
summary.PKNCAresults, 60, 70, 77, 84
superposition, 86

time_calc, 88
tss.monoexponential.generate.formula, 88