

Package ‘NMcalc’

July 21, 2025

Title Basic Calculations for PK/PD Modeling

Version 0.0.4

Maintainer Philip Delff <philip@delff.dk>

Description Essentials for PK/PD (pharmacokinetics/pharmacodynamics) such as area under the curve, (geometric) coefficient of variation, and other calculations that are not part of base R. This is not a noncompartmental analysis (NCA) package.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports stats, data.table

Suggests testthat, ggplot2

BugReports <https://github.com/philipdelff/NMdata/issues>

Language en-US

NeedsCompilation no

Author Philip Delff [aut, cre]

Repository CRAN

Date/Publication 2024-08-26 18:00:02 UTC

Contents

CV	2
CVlnorm	2
invlogit	3
logit	4
means	4
quantbin	5
seqlog	7
signif2	7
trapez	8

Index	9
--------------	----------

CV	<i>Calculate coefficient of variation of data</i>
----	---

Description

Calculate coefficient of variation of data

Usage

```
CV(x, log = FALSE)
```

Arguments

x	The data
log	If TRUE, the geometric coefficient of variation is calculated. This is $\sqrt{\exp(\text{var}(\log(x)) - 1)}$.

Details

This function is intended to be used on data. For a log-normal $\text{THETA1} * \text{EXP}(\text{ETA}(1))$ 'Nonmem' parameter, do $\text{CV} = \sqrt{\exp(\text{OMEGA}[1,1]) - 1}$.

Value

A numeric

Examples

```
set.seed(139)
x1 <- rnorm(1000, mean=5)
CV(x1)
CV(x1, log=TRUE)
x2 <- exp(x1)
CV(x2)
CV(x2, log=TRUE)
```

CVlnorm	<i>CV of log-normal dist baed on omega parameters CV based variance like provided in Nonmem's OMEGA metrix.</i>
---------	---

Description

CV of log-normal dist baed on omega parameters CV based variance like provided in Nonmem's OMEGA metrix.

Usage

```
CVlnorm(omega)
```

Arguments

omega A variance as provided in diagonal om the Nonmem OMEGA matrix

Details

This is a very simple function. All it does is `sqrt(exp(omega)-1)`.

Value

CV of the distribution (numeric)

invlogit	<i>Inverse logit function</i>
----------	-------------------------------

Description

Inverse logit function

Usage

```
invlogit(x)
```

Arguments

x a number to transform

Value

A numeric

See Also

logit

logit	<i>Logit function</i>
-------	-----------------------

Description

Logit function

Usage

```
logit(x)
```

Arguments

x a number to transform

Value

A numeric

See Also

invlogit

means	<i>calculate arithmetic or geometric mean and confidence intervals</i>
-------	--

Description

calculate arithmetic or geometric mean and confidence intervals

Usage

```
means(  
  x,  
  type = "arithmetic",  
  na.rm = FALSE,  
  z.rm = FALSE,  
  ci = FALSE,  
  dist.ci = "t",  
  p.ci = 0.95,  
  colnames = c("est", "ll", "ul"),  
  format = "df"  
)
```

Arguments

<code>x</code>	vector to calculate the geometric mean of
<code>type</code>	type of mean or median. Default is arithmetic, geometric and median are available as well. Only first letters needed, so say "geo" or even "g" is enough.
<code>na.rm</code>	Remove NA's before doing calculations?
<code>z.rm</code>	removes zeros before calculation? Default is FALSE. Can only be TRUE if <code>type="geometric"</code> .
<code>ci</code>	if TRUE, a data.frame including point estimate and confidence interval returned. If FALSE, a numeric representing the mean value returned.
<code>dist.ci</code>	The distribution to use for the confidence interval. Default and only supported is "t". If <code>type=geometric</code> , this is applied after transformation to gaussian.
<code>p.ci</code>	probability covered by confidence interval. Default is 0.95
<code>colnames</code>	If <code>ci</code> , this defines the column names of the resulting data frame. Default is <code>c("est","ll","ul")</code> .
<code>format</code>	The format of the result. Possible values are <code>df</code> and <code>num</code> .

Value

If `ci=FALSE`, a numeric. If `ci=TRUE`, a data.frame.

Examples

```
x <- 1:100
means(x, type="arithmetic", ci=TRUE)
means(x, type="geometric", ci=TRUE)
means(x, type="median", ci=TRUE)
library(data.table)
## CRAN requires examples to run on a single thread
data.table::setDTthreads(1)
data.table(x=x)[,append(means(x,ci=TRUE),list(N=.N))]
```

quantbin

Bin observations by quantiles. Label by bin number or by interval.

Description

This is simple stuff, but I can never remember the exact quantile and `findInterval/cut` commands to use. `quantbin` finds quantiles using `quantile` and then assigns bins using either `findInterval` or `cut`.

Usage

```
quantbin(x, nbins, probs, label = "num", ...)
```

Arguments

<code>x</code>	The observations
<code>nbins</code>	Number of bins to use
<code>probs</code>	Quantiles for construction of bins (optional). The default is to spread <code>nbins</code> quantiles equi-distantly across the observed values.
<code>label</code>	<code>label="num"</code> gives a numeric bin number (<code>findInterval</code>). <code>label="interval"</code> gives a character representation of the interval (<code>cut</code>).
<code>...</code>	additional arguments passed to <code>quantile</code> .

Details

`quantbin` uses `stats::quantile` for quantile estimation. Except for `x` and `probs`, all parameters can be controlled using `na.rm` and `...` arguments. See `?stats::quantile` for details.

`na.rm` `na.rm=TRUE` is needed for `quantile` to be able to estimate the distribution if `x` contains NA's. Notice, if `na.rm=T`, an NA element in `x` will still result in an NA element in return. If `na.rm=F` and there are NA's in `x`, all elements will be NA in result (quantiles cannot be determined, nor can the binning of `x` by those quantiles).

If data is not continuous, this method may not lead to balanced distributions.

Value

If `label="num"`, integers. If `label="interval"`, factors.

Examples

```
set.seed(134)
library(data.table)
## CRAN requires examples to run on a single thread
data.table::setDTthreads(1)
dt1 <- data.table(x=rnorm(n=1000))
dt1[,bin:=quantbin(x,nbins=4,label="num")]
dt1[,int:=quantbin(x,nbins=4,label="interval")]
## perfect - flat distribution
dt1[,.N,keyby=.(bin,int)]

dt2 <- data.table(x=c(rnorm(n=100000),NA))
dt2[,bin:=quantbin(x,nbins=4,label="num",na.rm=TRUE)]
dt2[,int:=quantbin(x,nbins=4,label="interval",na.rm=TRUE)]
## perfect - flat distribution
dt2[,.N,keyby=.(bin,int)]
unique(dt2[,.(bin,int)])[order(bin)]

## we may not get a flat distribution in case of discrete observations
dt3 <- data.table(x=c(sample(1:3,100,replace=TRUE)))
dt3[,bin:=quantbin(x,nbins=2,label="num",na.rm=TRUE)]
dt3[,int:=quantbin(x,nbins=2,label="interval",na.rm=TRUE)]
## Not a flat distribution
dt3[,.N,keyby=.(x,bin,int)]
```

seqlog	<i>Log-scale equidistant sequences</i>
--------	--

Description

Useful for generating sequences to be plotted on log scale. This is really simple - seq is run on from and to after log transformation, then the exponential is reported.

Usage

```
seqlog(from, to, length.out)
```

Arguments

from	start of sequence
to	end of sequence
length.out	length of sequence

Value

A numeric vector.

Examples

```
df <- data.frame(x=seqlog(1,100,100))
df <- transform(df, y=x/(10+x))
## Not run:
library(ggplot2)
## the points are equidistant on the log x scale
ggplot(df,aes(x,y))+geom_point()+scale_x_log10()

## End(Not run)
```

signif2	<i>round to fixed number of significant digits</i>
---------	--

Description

Even if theoretically correct, the built-in 'R' functions 'round' and 'signif' can be confusing (see examples). 'signif2' is a simple solution that can be used for reporting results consistently.

Usage

```
signif2(x, digits = 1, add, ...)
```

Arguments

<code>x</code>	a numeric vector.
<code>digits</code>	number of significant digits to round to. Must be an integer larger than 0.
<code>add</code>	pad with zeros where <code>digits > nchar(x[i])</code> . Currently not used.
<code>...</code>	additional arguments passed to <code>formatC</code> .

Value

A character vector.

Examples

```
x <- c(1.24e-4, 1.1334e6, 1.1, 22.00000, 10.00, 1)
data.frame(x, s.3=signif(x,3), sc.3=as.character(signif(x,3)), s2.3=signif2(x,3))
signif2(c(.2, 11.84), 2)
## digits has no effect when x==0
signif2(0, 1)
signif2(0, 3)
```

trapez

trapezoidal area under the curve on linear scale

Description

This is a numerical integration of `y` with respect to `x` by the trapezoidal method on linear scale.

Usage

```
trapez(x, y, cum = FALSE, na.rm = FALSE)
```

Arguments

<code>x</code>	The vector to integrate <code>y</code> with respect to (typically TIME to get area under the curve).
<code>y</code>	The variable to integrate.
<code>cum</code>	Return the cumulative trapezoidal area under the curve? If false (default) a single number is returned. If true, a vector is returned. Notice, the vector is one element shorter than <code>x</code> and <code>y</code> .
<code>na.rm</code>	Remove indexes in <code>x</code> and <code>y</code> wherever <code>x</code> or <code>y</code> are NA.

Value

a numeric

Index

CV, [2](#)

CVlnorm, [2](#)

invlogit, [3](#)

logit, [4](#)

means, [4](#)

quantbin, [5](#)

seqlog, [7](#)

signif2, [7](#)

trapez, [8](#)