Package: mcradds (via r-universe)

September 5, 2024

Type Package

Title Processing and Analyzing of Diagnostics Trials

Version 1.1.1.9000

Maintainer Kai Gu <gukai1212@163.com>

Description Provides methods and functions to analyze the quantitative or qualitative performance for diagnostic assays, and outliers detection, reader precision and reference range are discussed. Most of the methods and algorithms refer to CLSI (Clinical & Laboratory Standards Institute) recommendations and NMPA (National Medical Products Administration) guidelines. In additional, relevant plots are constructed by 'ggplot2'.

License GPL (>= 3)

URL https://github.com/kaigu1990/mcradds, https://kaigu1990.github.io/mcradds/

BugReports https://github.com/kaigu1990/mcradds/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.6)

Imports boot, checkmate, DescTools, dplyr, formatters, ggplot2, lifecycle, magrittr, methods, mcr, pROC, purrr, rlang, stats, tibble, tidyr, VCA

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0), usethis, vdiffr

Config/testthat/edition 3

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Collate 'pkg_class.R' 'pkg_methods.R' 'autoplot.R' 'blandAltman.R' 'correlation.R' 'data.R' 'desc.R' 'mcr.R' 'outlier.R' 'package.R' 'pairedroc.R' 'qual_stat.R' 'referenceInterval.R' 'samplesize.R' 'utils.R' 'vca.R'

Language en-US

Repository https://kaigu1990.r-universe.dev

RemoteUrl https://github.com/kaigu1990/mcradds

RemoteRef HEAD

RemoteSha 097752819ca7e6df2a0a9443aa810b99b89fc492

Contents

mcradds-package . . . 3
adsl_sub . . . 3
anovaVCA . . . 4
aucTest . . . 5
autoplot . . . 7
BAsummary-class . . . 11
blandAltman . . . 12
calcBias . . . 13
calcium . . . 14
cat_with_newline . . . 14
Desc-class . . . 15
descfreq . . . 16
descvar . . . 18
diagTab . . . 20
dixon_outlier . . . 22
esd.critical . . . 22
ESD_test . . . 23
getAccuracy . . . 24
getCoefficients . . . 27
getOutlier . . . 28
glucose . . . 29
h_difference . . . 30
h_factor . . . 31
h_fmt_count_perc . . . 31
h_fmt_est . . . 32
h_fmt_num . . . 33
h_fmt_range . . . 34
h_summarize . . . 35
ldlroc . . . 35
mcreg . . . 36
MCTab-class . . . 38
nonparRanks . . . 39
nonparRI . . . 39
PDL1RP . . . 40
pearsonTest . . . 41
platelet . . . 42
printSummary . . . 43
qualData . . . 43

RefInt-class 44
refInterval 45
robustRI 47
SampleSize-class 48
show,SampleSize-method 49
size_ci_corr 50
size_ci_one_prop 52
size_corr 53
size_one_prop 54
spearmanTest 55
tpROC-class 56
tukey_outlier 57
VCAinference 58

Index **60**

`mcradds-package` `mcradds` *Package*

Description

`mcradds` Processing and analyzing of In Vitro Diagnostic Data.

Author(s)

Maintainer: Kai Gu <gukai1212@163.com> [copyright holder]

See Also

Useful links:

- https://github.com/kaigu1990/mcradds
- https://kaigu1990.github.io/mcradds/
- Report bugs at https://github.com/kaigu1990/mcradds/issues

`adsl_sub` *CDISC ADSL Subsetting Data*

Description

[Experimental]

This ADSL is created by subsetting the CDISC ADSL with 60 subjects in each of two treatments like placebo and Xanomeline (this one corresponds to high dose level in the original ADSL).

Usage

```
adsl_sub
```

Format

A adsl_sub data set contains 120 observations and 14 variables. And the description of each variable has been labeled in data set.

---

anovaVCA *ANOVA-Type Estimation of Variance Components for Random Models*

Description

[Experimental]

A copy from VCA::anovaVCA in VCA package

Usage

```
anovaVCA(...)
```

Arguments

...	Arguments passed on to VCA::anovaVCA
	form (formula) specifying the model to be fit, a response variable left of the '~' is mandatory
	Data (data.frame) containing all variables referenced in 'form'
	by (factor, character) variable specifying groups for which the analysis should be performed individually, i.e. by-processing
	NegVC (logical) FALSE = negative variance component estimates (VC) will be set to 0 and they will not contribute to the total variance (as done in SAS PROC NESTED, conservative estimate of total variance). The original ANOVA estimates can be found in element 'VCoriginal'. The degrees of freedom of the total variance are based on adapted mean squares (MS), i.e. adapted MS are computed as $D * VC$, where VC is the column vector with negative VCs set to 0. TRUE = negative variance component estimates will not be set to 0 and they will contribute to the total variance (original definition of the total variance).
	VarVC.method (character) string specifying whether to use the algorithm given in Searle et al. (1992) which corresponds to VarVC.method="scm" or in Giesbrecht and Burns (1985) which can be specified via "gb". Method "scm" (Searle, Casella, McCulloch) is the exact algorithm, "gb" (Giesbrecht, Burns) is termed "rough approximation" by the authors, but sufficiently exact compared to e.g. SAS PROC MIXED (method=type1) which uses the inverse of the Fisher-Information matrix as approximation. For balanced designs all methods give identical results, only in unbalanced designs differences occur.

`MME` (logical) TRUE = (M)ixed (M)odel (E)quations will be solved, i.e. 'VCA' object will have additional elements "RandomEffects", "FixedEffects", "VarFixed" (variance-covariance matrix of fixed effects) and the "Matrices" element has addional elements corresponding to intermediate results of solving MMEs. FALSE = do not solve MMEs, which reduces the computation time for very complex models significantly.

`quiet` (logical) TRUE = will suppress any warning, which will be issued otherwise

`order.data` (logical) TRUE = class-variables will be ordered increasingly, FALSE = ordering of class-variables will remain as is

Value

a class of `VCA` for downstream analysis.

See Also

`VCA::anovaVCA()`

Examples

```
data(glucose)
anovaVCA(value ~ day / run, glucose)
```

---

aucTest *AUC Test for Paired Two-sample Measurements*

Description

[Experimental]

This function compares two AUC of paired two-sample diagnostic assays by standardized difference method, which has a little difference in SE calculation with unpaired design. In order to compare the two assays, this function provides three assessments including 'difference', 'non-inferiority' and 'superiority'. This method of comparing is referred from Liu(2006)'s article that can be found in reference section below.

Usage

```
aucTest(
  x,
  y,
  response,
  h0 = 0,
  conf.level = 0.95,
  method = c("difference", "non-inferiority", "superiority"),
  ...
)
```

Arguments

x	(numeric) reference/standard diagnostic assay.
y	(numeric) test diagnostic assay.
response	(numeric or factor) a vector of responses to represent the type of classes, typically encoded with 0(controls) and 1(cases).
h0	(numeric) a specified hypothesized value of the margin between the two assays, default is 0 for difference method. If you select the non-inferiority method, the h0 should be negative value. And if select superiority method, then it's non-negative value.
conf.level	(numeric) significance level between 0 and 1 (non-inclusive) for the returned confidence interval.
method	(string) string specifying the type of hypothesis test, must be one of "difference" (default), "non-inferiority" or "superiority".
...	other arguments to be passed to pROC::roc().

Details

If the samples are not considered independent, such as in a paired design, the SE can not be computed by the method of Delong provided in pROC package. Here the aucTest function use the standardized difference approach from Liu(2006) publication to compute the SE and corresponding hypothesis test statistic for a paired design study.

- difference is to test the difference between two diagnostic tests, the default h0 is zero.
- non-inferiority is to test the new diagnostic tests is no worse than the standard diagnostic test in a specific margin, but the same time maybe it's safer, easier to administer or cost less.
- superiority is to test the test the new diagnostic tests is better than the standard diagnostic test in a specific margin(default is zero), having better efficacy.

Value

A RefInt object contains relevant results in comparing the paired ROC of two-sample assays.

Note

The test of significance for the difference is not equal to the result of EP24A2 Appendix D. Table D2. Because the Table D2 uses the method of Hanley & McNeil (1982), whereas this function here uses the method of DeLong et al. (1988), which results in the difference of SE. Thus the corresponding Z statistic and P value will be not equal as well.

References

Jen-Pei Liu (2006) "Tests of equivalence and non-inferiority for diagnostic accuracy based on the paired areas under ROC curves". *Statist. Med.* , 25:1219–1238. DOI: 10.1002/sim.2358.

Examples

```
data("ldlroc")
# H0 : Difference between areas = 0:
aucTest(x = ldlroc$LDL, y = ldlroc$OxLDL, response = ldlroc$Diagnosis)

# H0 : Superiority margin <= 0.1:
aucTest(
  x = ldlroc$LDL, y = ldlroc$OxLDL, response = ldlroc$Diagnosis,
  method = "superiority", h0 = 0.1
)

# H0 : Non-inferiority margin <= -0.1:
aucTest(
  x = ldlroc$LDL, y = ldlroc$OxLDL, response = ldlroc$Diagnosis,
  method = "non-inferiority", h0 = -0.1
)
```

---

autoplot *Generate a* ggplot *for Bland-Altman Plot and Regression Plot*

Description

[Experimental]

Draw a ggplot-based difference Bland-Altman plot of reference assay vs. test assay for `BAsummary` object, and a regression plot for `MCResult`. Also Providing the necessary and useful option arguments for presentation.

Usage

```
autoplot(object, ...)

## S4 method for signature 'BAsummary'
autoplot(
  object,
  type = c("absolute", "relative"),
  color = "black",
  fill = "lightgray",
  size = 1.5,
  shape = 21,
  jitter = FALSE,
  ref.line = TRUE,
  ref.line.params = list(col = "blue", linetype = "solid", size = 1),
  ci.line = FALSE,
  ci.line.params = list(col = "blue", linetype = "dashed"),
  loa.line = TRUE,
  loa.line.params = list(col = "blue", linetype = "dashed"),
  label = TRUE,
```

```
  label.digits = 4,
  label.params = list(col = "black", size = 4),
  x.nbreak = NULL,
  y.nbreak = NULL,
  x.title = NULL,
  y.title = NULL,
  main.title = NULL
)

## S4 method for signature 'MCResult'
autoplot(
  object,
  color = "black",
  fill = "lightgray",
  size = 1.5,
  shape = 21,
  jitter = FALSE,
  identity = TRUE,
  identity.params = list(col = "gray", linetype = "dashed"),
  reg = TRUE,
  reg.params = list(col = "blue", linetype = "solid"),
  equal.axis = FALSE,
  legend.title = TRUE,
  legend.digits = 2,
  x.nbreak = NULL,
  y.nbreak = NULL,
  x.title = NULL,
  y.title = NULL,
  main.title = NULL
)
```

Arguments

object	(BAsummary, MCResult) input, depending on which function you have done, blandAltman() or mcreg().
...	not used.
type	(string) difference type from input, default is 'absolute'.
color, fill	(string) point colors.
size	(numeric) the size of points.
shape	(integer) the ggplot shape of points.
jitter	(logical) whether to add a small amount of random variation to the location of points.

`ref.line` (logical)
whether to plot a 'mean' line, default is TRUE.

`ref.line.params`, `ci.line.params`, `loa.line.params`
(list)
parameters (color, linetype, linewidth) for the argument 'ref.line', 'ci.line' and 'loa.line'; eg. ref.line.params = list(col = "blue", linetype = "solid", linewidth = 1).

`ci.line` (logical)
whether to plot a confidence interval line of 'mean', default is FALSE.

`loa.line` (logical)
whether to plot limit of agreement line, default is TRUE.

`label` (logical)
whether to add specific value label for each line (ref.line, ci.line and loa.line). Only be shown when the line is defined as TRUE.

`label.digits` (integer)
the number of digits after the decimal point in the each label.

`label.params` (list)
parameters (color, size, fontface) for the argument 'label'.

`x.nbreak`, `y.nbreak`
(integer)
an integer guiding the number of major breaks of x-axis and y-axis.

`x.title`, `y.title`, `main.title`
(string)
the x-axis, y-axis and main title of plot.

`identity` (logical)
whether to add identity line, default is TRUE.

`identity.params`, `reg.params`
(list)
parameters (color, linetype) for the argument 'identity' and 'reg'; eg. identity.params = list(col = "gray", linetype = "dashed").

`reg` (logical)
whether to add regression line where the slope and intercept are obtained from `mcr::mcreg()` function, default is TRUE.

`equal.axis` (logical)
whether to adjust the ranges of x-axis and y-axis are identical. If `equal.axis = TRUE`, x-axis will be equal to y-axis.

`legend.title` (logical)
whether to present the title in the legend.

`legend.digits` (integer)
the number of digits after the decimal point in the legend.

Value

A `ggplot` based Bland-Altman plot or regression plot that can be easily customized using additional `ggplot` functions.

Note

If you'd like to alter any part that this `autoplot` function haven't provided, adding other `ggplot` statements are suggested.

See Also

`h_difference()` to see the type details.

`mcr::mcreg()` to see the regression parameters.

Examples

```
# Specify the type for difference plot
data("platelet")
object <- blandAltman(x = platelet$Comparative, y = platelet$Candidate)
autoplot(object)
autoplot(object, type = "relative")

# Set the addition parameters for `geom_point`
autoplot(object,
  type = "relative",
  jitter = TRUE,
  fill = "lightblue",
  color = "grey",
  size = 2
)

# Set the color and line type for reference and limits of agreement lines
autoplot(object,
  type = "relative",
  ref.line.params = list(col = "red", linetype = "solid"),
  loa.line.params = list(col = "grey", linetype = "solid")
)

# Set label color, size and digits
autoplot(object,
  type = "absolute",
  ref.line.params = list(col = "grey"),
  loa.line.params = list(col = "grey"),
  label.digits = 2,
  label.params = list(col = "grey", size = 3, fontface = "italic")
)

# Add main title, X and Y axis titles, and adjust X ticks.
autoplot(object,
  type = "absolute",
  x.nbreak = 6,
  main.title = "Bland-Altman Plot",
  x.title = "Mean of Test and Reference Methods",
  y.title = "Reference - Test"
)
# Using the default arguments for regression plot
```

```
data("platelet")
fit <- mcreg(
  x = platelet$Comparative, y = platelet$Candidate,
  method.reg = "Deming", method.ci = "jackknife"
)
autoplot(fit)

# Only present the regression line and alter the color and shape.
autoplot(fit,
  identity = FALSE,
  reg.params = list(col = "grey", linetype = "dashed"),
  legend.title = FALSE,
  legend.digits = 4
)
```

`BAsummary-class` *BAsummary Class*

Description

[Experimental]

The `BAsummary` class is used to display the BlandAltman analysis and outliers.

Usage

```
BAsummary(call, data, stat, param)
```

Arguments

`call`	(`call`) function call.
`data`	(`data.frame`) stores the raw data from input.
`stat`	(`list`) contains several statistics for numeric data.
`param`	(`list`) list of relevant parameters.

Value

An object of class `BAsummary`.

Slots

`call` call

`data` data

`outlier` outlier

`param` param

`blandAltman` *Calculate Statistics for Bland-Altman*

Description

[Experimental]

Calculate the Bland-Altman related statistics with specific difference type, such as difference, limited of agreement and confidence interval. And the outlier detecting function and graphic function will get the difference result from this.

Usage

```
blandAltman(x, y, sid = NULL, type1 = 3, type2 = 5, conf.level = 0.95)
```

Arguments

`x`	(`numeric`) reference method.
`y`	(`numeric`) test method.
`sid`	(`numeric` or `string`) sample id.
`type1`	(`integer`) specifying a specific difference for absolute difference, default is 3.
`type2`	(`integer`) specifying a specific difference for relative difference, default is 5.
`conf.level`	(`numeric`) significance level for two side, default is 0.95.

Value

A object with `BAsummary` class that contains the BlandAltman analysis.

- `data` a data frame contains the raw data from the input.
- `stat` a list contains the summary table (`tab`) of Bland-Altman analysis, vector (`absolute_diff`) of absolute difference and vector (`relative_diff`) of relative difference.

See Also

`h_difference()` to see the type details.

Examples

```
data("platelet")
blandAltman(x = platelet$Comparative, y = platelet$Candidate)

# with sample id as input sid
blandAltman(x = platelet$Comparative, y = platelet$Candidate, sid = platelet$Sample)

# Specifiy the type for difference
blandAltman(x = platelet$Comparative, y = platelet$Candidate, type1 = 1, type2 = 4)
```

`calcBias` *Systematical Bias Between Reference Method and Test Method*

Description

[Experimental]

A copy from mcr::calcBias in `mcr` package

Usage

```
calcBias(...)
```

Arguments

`...`	Arguments passed on to `mcr::calcBias`
	`.Object` object of class "MCResult".

Value

Bis and corresponding confidence interval for the specific medical decision levels (`x.levels`).

See Also

`mcr::calcBias()`

Examples

```
data(platelet)
fit <- mcreg(
  x = platelet$Comparative, y = platelet$Candidate,
  method.reg = "Deming", method.ci = "jackknife"
)
calcBias(fit, x.levels = c(30, 200))
calcBias(fit, x.levels = c(30, 200), type = "proportional")
calcBias(fit, x.levels = c(30, 200), type = "proportional", percent = FALSE)
```

calcium *Reference Interval Data*

Description

[Experimental]

This example `calcium` can be used to compute the reference range of Calcium in 240 medical students by sex.

Usage

```
calcium
```

Format

A calcium data set contains 240 observations and 3 variables.

Sample Sample id

Value Measurements from target subjects

Group Sex group of target subjects

Source

CLSI-EP28A3 Table 4. is cited in this data set.

cat_with_newline *Concatenate and Print with Newline*

Description

[Experimental]

This function concatenates inputs like `cat()` and prints them with newline.

Usage

```
cat_with_newline(...)
```

Arguments

...	inputs to concatenate.

Value

None, only used for the side effect of producing the concatenated output in the R console.

See Also

This is similar to `cli::cat_line()`.

Examples

```
cat_with_newline("hello", "world")
```

`Desc-class` *Descriptive Statistics Class*

Description

[Experimental]

The `Desc` class serves as the store for results from frequency and univariate statistics analysis.

Usage

```
Desc(func, mat, stat)
```

Arguments

`func`	(`character`) name of function.
`mat`	(`data.frame`) intermediate data with long form, easy for post-processing.
`stat`	(`data.frame`) final data with wide form, easy for presentation.

Value

An object of class `Desc`.

Slots

`func` func

`mat` mat

`stat` stat

descfreq *Summarize Frequency Counts and Percentages*

Description

[Experimental]

Create a summary table for one or more variables by one group, as well as a total column if necessary.

Usage

```
descfreq(
  data,
  denom = NULL,
  var,
  bygroup,
  format,
  fctdrop = FALSE,
  addtot = FALSE,
  na_str = NULL
)
```

Arguments

data	(data.frame) a data frame that contains the variables to be summarized and grouped.
denom	(data.frame) the denominator to use for the percentage, but not use temporarily. By default, it's NULL, meaning the function will use the number of values of the data, including missing value.
var	(vector) a character vector of variables to be summarized within data.
bygroup	(string) a character variable for grouping within data.
format	(string) formatting string from formatters::list_valid_format_labels() for frequency counts and percentages.
fctdrop	(logical) whether to include the levels of the variables but with no records.
addtot	(logical) whether to add total column in the output or not.
na_str	(string) a string to replace NA in the output if no records will be counted for any category.

Value

A object `Desc` contains an intermediate data with long form for post-processing and final data with wide form for presentation.

Note

By default, the each category is sorted based on the corresponding factor level of `var` variable. If the variable is not a factor, that will be sorted alphabetically.

Examples

```
data(adsl_sub)

# Count the age group by treatment with 'xx (xx.x%)' format
adsl_sub %>%
  descfreq(
    var = "AGEGR1",
    bygroup = "TRTP",
    format = "xx (xx.x%)"
  )

# Count the race by treatment with 'xx (xx.xx)' format and replace NA with '0'
adsl_sub %>%
  descfreq(
    var = "RACE",
    bygroup = "TRTP",
    format = "xx (xx.xx)",
    na_str = "0"
  )

# Count the sex by treatment adding total column
adsl_sub %>%
  descfreq(
    var = "SEX",
    bygroup = "TRTP",
    format = "xx (xx.x%)",
    addtot = TRUE
  )

# Count multiple variables by treatment and sort category by corresponding factor levels
adsl_sub %>%
  dplyr::mutate(
    AGEGR1 = factor(AGEGR1, levels = c("<65", "65-80", ">80")),
    SEX = factor(SEX, levels = c("M", "F")),
    RACE = factor(RACE, levels = c(
      "WHITE", "AMERICAN INDIAN OR ALASKA NATIVE",
      "BLACK OR AFRICAN AMERICAN"
    ))
  ) %>%
  descfreq(
    var = c("AGEGR1", "SEX", "RACE"),
    bygroup = "TRTP",
```

```
  format = "xx (xx.x%)",
  addtot = TRUE,
  na_str = "0"
)
```

---

descvar *Summarize Descriptive Statistics*

Description

[Experimental]

Create a summary table with a set of descriptive statistics for one or more variables by one group, as well as a total column if necessary.

Usage

```
descvar(
  data,
  var,
  bygroup,
  stats = getOption("mcradds.stats.default"),
  autodecimal = TRUE,
  decimal = 1,
  addtot = FALSE,
  .perctype = 2
)
```

Arguments

data	(data.frame) a data frame that contains the variables to be summarized and grouped.
var	(vector) a character vector of variables to be summarized within data.
bygroup	(string) a character variable for grouping within data.
stats	(vector) a statistics character vector must be chosen from c("N", "MEAN", "SD", "MEDIAN", "MAX", "MIN", "Q1", "Q3", "MEANSD", "RANGE", "IQR", "MEDRANGE", "MEDIQR"), and the default are top six items.
autodecimal	(logical) whether to capture the variable's maximum decimal, and the final decimal precision is equal to the variable decimal plus the definition of each statistic from getOption("mcradds.precision.default").
decimal	(integer) a integer number to define the decimal precision for each variable.

addtot (logical)
whether to add total column in the output or not.

.perctype (integer)
an integer between 1 and 9 selecting one of the nine quantile algorithms, also see the details in quantile(). The default is 2, so that it can be consistent with SAS quantile calculation.

Value

A object Desc contains an intermediate data with long form for post-processing and final data with wide form for presentation.

Note

The decimal precision is based on two aspects, one is the original precision from the variable or the decimal argument, and the second is the common use that has been defined in getOption("mcradds.precision.default")
So if you want to change the second decimal precision, you can alter it manually with option().

Examples

```
data(adsl_sub)

# Compute the default statistics of AGE by TRTP group
adsl_sub %>%
  descvar(
    var = "AGE",
    bygroup = "TRTP"
  )

# Compute the specific statistics of BMI by TRTP group, adding total column
adsl_sub %>%
  descvar(
    var = "BMIBL",
    bygroup = "TRTP",
    stats = c("N", "MEANSD", "MEDIAN", "RANGE", "IQR"),
    addtot = TRUE
  )

# Set extra decimal to define precision
adsl_sub %>%
  descvar(
    var = "BMIBL",
    bygroup = "TRTP",
    stats = c("N", "MEANSD", "MEDIAN", "RANGE", "IQR"),
    autodecimal = FALSE,
    decimal = 2,
    addtot = TRUE
  )

# Set multiple variables together
adsl_sub %>%
```

```
descvar(
  var = c("AGE", "BMIBL", "HEIGHTBL"),
  bygroup = "TRTP",
  stats = c("N", "MEANSD", "MEDIAN", "RANGE", "IQR"),
  autodecimal = TRUE,
  addtot = TRUE
)
```

---

diagTab *Creates Contingency Table*

Description

[Experimental]

Creates a 2x2 contingency table from the data frame or matrix for the qualitative performance and reader precision of downstream analysis.

Usage

```
diagTab(
  formula = ~.,
  data,
  bysort = NULL,
  dimname = NULL,
  levels = NULL,
  rep = FALSE,
  across = NULL
)
```

Arguments

formula	(numeric) a formula object with the cross-classifying variables (separated by +) on the right hand side. If data is wide structure, the row name of contingency is represented by the variable to the left of the + sign, and the col name is the right. If data is long structure, the classified variable is put on the left of the formula, and the value variable is put on the right.
data	(data.frame or matrix) a data frame or matrix.
bysort	(string) a sorted variable from the col names of data, and a grouped variable for reproducibility analysis.
dimname	(vector) a character vector define the row name of contingency table in first variable, and col name in second variable.

levels	(vector) a vector of known levels for measurements.
rep	(logical) whether to implement the reproducibility like reader precision or not.
across	(string) the across variable to split original data set to subsets. The between-reader and within-reader precision's across variable is site commonly.

Value

A object MCTab contains the 2x2 contingency table.

Note

To be attention that if you would like to generate the 2x2 contingency table for reproducibility analysis, the original data should be long structure and using the corresponding formula.

See Also

Summary() for object to calculate diagnostic accuracy criteria.

Examples

```
# For qualitative performance with wide data structure
data("qualData")
qualData %>% diagTab(formula = ~ CandidateN + ComparativeN)
qualData %>%
  diagTab(
    formula = ~ CandidateN + ComparativeN,
    levels = c(1, 0)
  )

# For qualitative performance with long data structure
dummy <- data.frame(
  id = c("1001", "1001", "1002", "1002", "1003", "1003"),
  value = c(1, 0, 0, 0, 1, 1),
  type = c("Test", "Ref", "Test", "Ref", "Test", "Ref")
)
dummy %>%
  diagTab(
    formula = type ~ value,
    bysort = "id",
    dimname = c("Test", "Ref"),
    levels = c(1, 0)
  )

# For Between-Reader precision performance
data("PDL1RP")
reader <- PDL1RP$btw_reader
reader %>%
  diagTab(
```

```
    formula = Reader ~ Value,
    bysort = "Sample",
    levels = c("Positive", "Negative"),
    rep = TRUE,
    across = "Site"
  )
```

`dixon_outlier` *Detect Dixon Outlier*

Description

[Experimental]

Help function detects the potential outlier with Dixon method, following the rules of EP28A3 and NMPA guideline for establishment of reference range.

Usage

```
dixon_outlier(x)
```

Arguments

x	(numeric) numeric input.

Value

A list contains outliers and vector without outliers.

Examples

```
x <- c(13.6, 44.4, 45.9, 11.9, 41.9, 53.3, 44.7, 95.2, 44.1, 50.7, 45.2, 60.1, 89.1)
dixon_outlier(x)
```

`esd.critical` *Compute Critical Value for ESD Test*

Description

[Experimental]

A helper function to find the lambda for all potential outliers in each iteration.

Usage

```
esd.critical(alpha, N, i)
```

Arguments

alpha	(numeric) type-I-risk, α.
N	(integer) the total number of samples.
i	(integer) the iteration number, less than the number of biggest potential outliers.

Value

a lambda value calculated from the formula.

Examples

```
esd.critical(alpha = 0.05, N = 100, i = 1)
```

ESD_test *EDS Test for Outliers*

Description

[Experimental]

Perform Rosner's generalized extreme Studentized deviate (ESD) test, which assumes that the distribution is normal (Gaussian), can be used when the number of outliers is unknown, and becomes more robust as the number of samples increases.

Usage

```
ESD_test(x, alpha = 0.05, h = 5)
```

Arguments

x	(numeric) vector of observations that can be the difference from Bland-Altman analysis. Normally the relative difference is preferred in IVD trials. Missing(NA) is allowed but will be removed. There must be at least 10 available observations in x.
alpha	(numeric) type-I-risk, α.
h	(integer) the positive integer indicating the number of suspected outliers. The argument h must be between 1 and n-2 where n denotes the number of available values in x. The default value is h = 5.

Value

A list class containing the results of the ESD test.

- `stat` a data frame contains the several statistics about ESD test that includes the index(`i`), Mean, SD, raw data(`x`), the location(`Obs`) in `x`, ESD statistics(ESDi), Lambda and Outliers(`TRUE` or `FALSE`).
- `ord` a vector with the order index of outliers that is equal to `Obs` in the `stat` data frame.

Note

The algorithm for determining the number of outliers is as follows:

- Compare ESDi with Lambda. If ESDi > Lambda then the observations will be regards as outliers.
- The order index corresponds to the available `x` data that has been removed the missing (NA) value.
- As we should compare if the ESD(h) and ESD(h+1) are equal, the h+1 ESD values will be shown. If they are identical, both of them can not be regarded as outliers.

References

CLSI EP09A3 Appendix B. Detecting Aberrant Results (Outliers).

Examples

```
data("platelet")
res <- blandAltman(x = platelet$Comparative, y = platelet$Candidate)
ESD_test(x = res@stat$relative_diff)
```

`getAccuracy` *Summary Method for* `MCTab` *Objects*

Description

[Experimental]

Provides a concise summary of the content of `MCTab` objects. Computes sensitivity, specificity, positive and negative predictive values and positive and negative likelihood ratios for a diagnostic test with reference/gold standard. Computes positive/negative percent agreement, overall percent agreement and Kappa when the new test is evaluated by comparison to a non-reference standard. Computes average positive/negative agreement when the both tests are all not the reference, such as paired reader precision.

Usage

```
getAccuracy(object, ...)

## S4 method for signature 'MCTab'
getAccuracy(
  object,
  ref = c("r", "nr", "bnr"),
  alpha = 0.05,
  r_ci = c("wilson", "wald", "clopper-pearson"),
  nr_ci = c("wilson", "wald", "clopper-pearson"),
  bnr_ci = "bootstrap",
  bootCI = c("perc", "norm", "basic", "stud", "bca"),
  nrep = 1000,
  rng.seed = NULL,
  digits = 4,
  ...
)
```

Arguments

object	(MCTab) input from diagTab function to create 2x2 contingency table.
...	other arguments to be passed to DescTools::BinomCI.
ref	(character) reference condition. It is possible to choose one condition for your require. The r indicates that the comparative test is standard reference, nr indicates the comparative test is not a standard reference, and bnr indicates both the new test and comparative test are not references.
alpha	(numeric) type-I-risk, α.
r_ci	(string) string specifying which method to calculate the confidence interval for a diagnostic test with reference/gold standard. Default is wilson. Options can be wilson, wald and clopper-pearson, see DescTools::BinomCI.
nr_ci	(string) string specifying which method to calculate the confidence interval for the comparative test with non-reference standard. Default is wilson. Options can be wilson, wald and clopper-pearson, see DescTools::BinomCI.
bnr_ci	(string) string specifying which method to calculate the confidence interval for both tests are not reference like reader precision. Default is bootstrap. But when the point estimate of ANA or APA is equal to 0 or 100%, the method will be changed to transformed wilson.
bootCI	(string) string specifying the which bootstrap confidence interval from boot.ci() function in boot package. Default is perc(bootstrap percentile), options can be

norm(normal approximation), boot(basic bootstrap), stud(studentized bootstrap) and bca(adjusted bootstrap percentile).

`nrep` (integer)
number of replicates for bootstrapping, default is 1000.

`rng.seed` (integer)
number of the random number generator seed for bootstrap sampling. If set to NULL currently in the R session used RNG setting will be used.

`digits` (integer)
the desired number of digits. Default is 4.

Value

A data frame contains the qualitative diagnostic accuracy criteria with three columns for estimated value and confidence interval.

- sens: Sensitivity refers to how often the test is positive when the condition of interest is present.
- spec: Specificity refers to how often the test is negative when the condition of interest is absent.
- ppv: Positive predictive value refers to the percentage of subjects with a positive test result who have the target condition.
- npv: Negative predictive value refers to the percentage of subjects with a negative test result who do not have the target condition.
- plr: Positive likelihood ratio refers to the probability of true positive rate divided by the false negative rate.
- nlr: Negative likelihood ratio refers to the probability of false positive rate divided by the true negative rate.
- ppa: Positive percent agreement, equals to sensitivity when the candidate method is evaluated by comparison with a comparative method, not reference/gold standard.
- npa: Negative percent agreement, equals to specificity when the candidate method is evaluated by comparison with a comparative method, not reference/gold standard.
- opa: Overall percent agreement.
- kappa: Cohen's kappa coefficient to measure the level of agreement.
- apa: Average positive agreement refers to the positive agreements and can be regarded as weighted ppa.
- ana: Average negative agreement refers to the negative agreements and can be regarded as weighted npa.

Examples

```
# For qualitative performance
data("qualData")
tb <- qualData %>%
  diagTab(
    formula = ~ CandidateN + ComparativeN,
```

```
    levels = c(1, 0)
  )
getAccuracy(tb, ref = "r")
getAccuracy(tb, ref = "nr", nr_ci = "wilson")

# For Between-Reader precision performance
data("PDL1RP")
reader <- PDL1RP$btw_reader
tb2 <- reader %>%
  diagTab(
    formula = Reader ~ Value,
    bysort = "Sample",
    levels = c("Positive", "Negative"),
    rep = TRUE,
    across = "Site"
  )
getAccuracy(tb2, ref = "bnr")
getAccuracy(tb2, ref = "bnr", rng.seed = 12306)
```

`getCoefficients` *Get Regression Coefficients*

Description

[Experimental]

A copy from mcr::getCoefficients in `mcr` package

Usage

```
getCoefficients(...)
```

Arguments

`...`	Arguments passed on to `mcr::getCoefficients`
	`.Object` object of class "MCResult".

Examples

```
data(platelet)
fit <- mcreg(
  x = platelet$Comparative, y = platelet$Candidate,
  method.reg = "Deming", method.ci = "jackknife"
)
getCoefficients(fit)
```

`getOutlier` *Detect Outliers From* `BAsummary` *Object*

Description

[Experimental]

Detect the potential outliers from the absolute and relative differences in `BAsummary` object with 4E and ESD method.

Usage

```
getOutlier(object, ...)

## S4 method for signature 'BAsummary'
getOutlier(
  object,
  method = c("ESD", "4E"),
  difference = c("abs", "rel"),
  alpha = 0.05,
  h = 5
)
```

Arguments

`object`	(`BAsummary`) input from blandAltman function to generate the Bland-Altman analysis result that contains the absolute and relative differences.
`...`	not used.
`method`	(`string`) string specifying which method to use. Default is `ESD`.
`difference`	(`string`) string specifying which difference type to use for `ESD` method. Default is `abs` that means absolute difference, and `rel` is relative difference.
`alpha`	(`numeric`) type-I-risk. Only used when the method is defined as `ESD`.
`h`	(`integer`) the positive integer indicating the number of suspected outliers. Only used when the method is defined as `ESD`.

Value

A list contains the statistics results (`stat`), outliers' ord id (`ord`), sample id (`sid`), matrix with outliers (`outmat`) and matrix without outliers (`rmmat`).

Note

Bland-Altman analysis is used as the input data regardless of the 4E and ESD method because it's necessary to determine the absolute and relative differences beforehand. For the 4E method, both of the absolute and relative differences are required to be define, and the bias exceeds the 4 fold of the absolute and relative differences. However for the ESD method, only one of them is necessary (the latter is more recommended), and the bias needs to meet the ESD test.

Examples

```
data("platelet")
# Using `blandAltman` function with default arguments
ba <- blandAltman(x = platelet$Comparative, y = platelet$Candidate)
getOutlier(ba, method = "ESD", difference = "rel")

# Using sample id as input
ba2 <- blandAltman(x = platelet$Comparative, y = platelet$Candidate, sid = platelet$Sample)
getOutlier(ba2, method = "ESD", difference = "rel")

# Using `blandAltman` function when the `tyep2` is 2 with `X vs. (Y-X)/X` difference
ba3 <- blandAltman(x = platelet$Comparative, y = platelet$Candidate, type2 = 4)
getOutlier(ba3, method = "ESD", difference = "rel")

# Using "4E" as the method input
ba4 <- blandAltman(x = platelet$Comparative, y = platelet$Candidate)
getOutlier(ba4, method = "4E")
```

---

glucose *Inermediate Precision Data*

Description

[Experimental]

This data set consists of the Glucose intermediate precision data in the CLSI EP05-A3 guideline.

Usage

```
glucose
```

Format

A glucose data set contains 80 observations and 3 variables.

day day number

run run number

value measurement value

Source

CLSI-EP05A3 Table A1. Glucose Precision Evaluation Measurements (mg/dL) is cited in this data set.

References

EP05A3: Evaluation of Precision of Quantitative Measurement Procedures.

`h_difference` *Compute Difference for Bland-Altman*

Description

[Experimental]

Helper function computes the difference with specific type.

Usage

```
h_difference(x, y, type)
```

Arguments

`x`	(`numeric`) reference method.
`y`	(`numeric`) test method.
`type`	(`integer`) integer specifying a specific difference for Bland-Altman (default is 3). Possible choices are: 1 - difference with X vs. Y-X (absolute differences). 2 - difference with X vs. (Y-X)/X (relative differences). 3 - difference with 0.5*(X+Y) vs. Y-X (absolute differences). 4 - difference with 0.5*(X+Y) vs. (Y-X)/X (relative differences). 5 - difference with 0.5*(X+Y) vs. (Y-X)/(0.5*(X+Y)) (relative differences).

Value

a matrix contains the x and y measurement data and corresponding difference.

Examples

```
h_difference(x = c(1.1, 1.2, 1.5), y = c(1.2, 1.3, 1.4), type = 5)
```

h_factor *Factor Variable Per Levels*

Description

[Experimental]

Helper function factor inputs in order of appearance, or per the levels that you provide.

Usage

```
h_factor(df, var, levels = NULL, ...)
```

Arguments

df	(data.frame) input data.
var	(string) variable to factor.
levels	(vector) a character vector of known levels.
...	other arguments to be passed to factor().

Value

A factor variable

Examples

```
df <- data.frame(a = c("aa", "a", "aa"))
h_factor(df, var = "a")
h_factor(df, var = "a", levels = c("aa", "a"))
```

h_fmt_count_perc *Format count and percent*

Description

[Experimental]

Help function to format the count and percent into one string.

Usage

```
h_fmt_count_perc(cnt, perc = NULL, format, ...)
```

Arguments

`cnt`	(`numeric`) numeric vector for count.
`perc`	(`numeric`) numeric vector for percent, if Null only format count.
`format`	(`string`) formatting string from `formatters::list_valid_format_labels()` for `formatters::format_value(` function.
`...`	other arguments to be passed to formatters::format_value.

Value

A character vector of formatted counts and percents.

Examples

```
h_fmt_count_perc(cnt = c(5, 9, 12, 110, 0), format = "xx")
h_fmt_count_perc(
  cnt = c(5, 9, 12, 110, 0),
  perc = c(0.0368, 0.0662, 0.0882, 0.8088, 0),
  format = "xx (xx.x%)"
)
```

---

h_fmt_est *Format and Concatenate to String*

Description

[Experimental]

Help function to format numeric data as strings and concatenate into a single character.

Usage

```
h_fmt_est(num1, num2, digits = c(2, 2), width = c(6, 6))
```

Arguments

`num1`	(`numeric`) first numeric input.
`num2`	(`numeric`) second numeric input.
`digits`	(`integer`) the desired number of digits after the decimal point.
`width`	(`integer`) the total field width.

Value

A single character.

See Also

h_fmt_num()

Examples

```
h_fmt_est(num1 = 3.14, num2 = 3.1415, width = c(4, 4))
```

---

h_fmt_num *Format Numeric Data*

Description

[Experimental]

Help function to format numeric data with `formatC` function.

Usage

```
h_fmt_num(x, digits, width = digits + 4)
```

Arguments

`x`	(`numeric`) numeric input.
`digits`	(`integer`) the desired number of digits after the decimal point (format = "f").
`width`	(`integer`) the total field width.

Value

A character object with specific digits and width.

See Also

formatC()

Examples

```
h_fmt_num(pi * 10^(-2:2), digits = 2, width = 6)
```

h_fmt_range *Format and Concatenate to Range*

Description

[Experimental]

Help function to format numeric data as strings and concatenate into a single character range.

Usage

```
h_fmt_range(num1, num2, digits = c(2, 2), width = c(6, 6))
```

Arguments

num1	(numeric) first numeric input.
num2	(numeric) second numeric input.
digits	(integer) the desired number of digits after the decimal point.
width	(integer) the total field width.

Value

A single character.

See Also

h_fmt_num()

Examples

```
h_fmt_range(num1 = 3.14, num2 = 3.14, width = c(4, 4))
```

h_summarize *Summarize Basic Statistics*

Description

[Experimental]

Help function summarizes the statistics as needed.

Usage

```
h_summarize(x, conf.level = 0.95)
```

Arguments

x	(numeric) input numeric vector.
conf.level	(numeric) significance level, default is 0.95.

Value

a verctor contains several statistics, such as n, mean, median, min, max, q25, q75, sd, se, limit of agreement of limit and confidence interval .

Examples

```
h_summarize(1:50)
```

ldlroc *Two-sampled Paired Test Data*

Description

[Experimental]

This data set consists the measurements of low-density lipoprotein (LDL), oxidized low-density lipoprotein (OxLDL) and the corresponding diagnosis. OxLDL is thought to be the active molecule in the process of atherosclerosis, so its proponents believe that its serum concentration should provide more accurate risk stratification than the traditional LDL assay.

Usage

```
ldlroc
```

Format

A ldlroc data set contains 50 observations and 3 variables.

Diagnosis the diagnosis, 1 represents a subject has the disease or condition of interest is present, 0 is absent

OxLDL oxidized low-density lipoprotein(OxLDL) measurement value

LDL low-density lipoprotein(LDL) measurement value

Source

CLSI-EP24A2 Table D1. OxLDL and LDL Assay Values (in U/L) for 50 Subjects.

References

EP24A2 Assessment of the Diagnostic Accuracy of Laboratory Tests Using Receiver Operating Characteristic Curves.

`mcreg` *Comparison of Two Measurement Methods Using Regression Analysis*

Description

[Experimental]

A copy from mcr::mcreg in `mcr` package

Usage

```
mcreg(...)
```

Arguments

`...`	Arguments passed on to `mcr::mcreg` `x` measurement values of reference method, or two column matrix. `y` measurement values of test method. `error.ratio` ratio between squared measurement errors of reference and test method, necessary for Deming regression (Default 1). `alpha` value specifying the 100(1-alpha)% confidence level for confidence intervals (Default is 0.05). `mref.name` name of reference method (Default "Method1"). `mtest.name` name of test Method (Default "Method2"). `sample.names` names of cases (Default "S##").

`method.reg` regression method. It is possible to choose between five regression methods: `"LinReg"` - ordinary least square regression.
`"WLinReg"` - weighted ordinary least square regression.
`"Deming"` - Deming regression.
`"WDeming"` - weighted Deming regression.
`"TS"` - Theil-Sen regression.
`"PBequi"` - equivariant Passing-Bablok regression.
`"PaBa"` - Passing-Bablok regression.
`"PaBaLarge"` - approximative Passing-Bablok regression for large datasets, operating on `NBins` classes of constant slope angle which each slope is classified to instead of building the complete triangular matrix of all N*N/2 slopes.

`method.ci` method of confidence interval calculation. The function contains four basic methods for calculation of confidence intervals for regression coefficients. `"analytical"` - with parametric method.
`"jackknife"` - with leave one out resampling.
`"bootstrap"` - with ordinary non-parametric bootstrap resampling.
`"nested bootstrap"` - with ordinary non-parametric bootstrap resampling.

`method.bootstrap.ci` bootstrap based confidence interval estimation method.

`nsamples` number of bootstrap samples.

`nnested` number of nested bootstrap samples.

`rng.seed` integer number that sets the random number generator seed for bootstrap sampling. If set to NULL currently in the R session used RNG setting will be used.

`rng.kind` type of random number generator for bootstrap sampling. Only used when rng.seed is specified, see set.seed for details.

`iter.max` maximum number of iterations for weighted Deming iterative algorithm.

`threshold` numerical tolerance for weighted Deming iterative algorithm convergence.

`na.rm` remove measurement pairs that contain missing values (Default is FALSE).

`NBins` number of bins used when 'reg.method="PaBaLarge"' to classify each slope in one of 'NBins' bins covering the range of all slopes

`slope.measure` angular measure of pairwise slopes used for exact PaBa regression (see below for details).
`"radian"` - for data sets with even sample numbers median slope is calculated as average of two central slope angles.
`"tangent"` - for data sets with even sample numbers median slope is calculated as average of two central slopes (tan(angle)).

`methodlarge` Boolean. This parameter applies only to regmethod="PBequi" and "TS". If TRUE, a quasilinear algorithm is used. If FALSE, a quadratic algorithm is used which is faster for less than several hundred data pairs.

Value

A regression fit model.

See Also

mcr::mcreg()

Examples

```
data(platelet)
fit <- mcreg(
  x = platelet$Comparative, y = platelet$Candidate,
  method.reg = "Deming", method.ci = "jackknife"
)
printSummary(fit)
getCoefficients(fit)
```

MCTab-class *MCTab Class*

Description

[Experimental]

The MCTab class serves as the store for 2x2 contingency table

Usage

```
MCTab(data, tab, levels)
```

Arguments

data	(data.frame) original data set.
tab	(table) table class converted from table() to display 2x2 contingency table.
levels	(character) levels of measurements.

Value

An object of class MCTab.

Slots

data data

tab candidate

levels levels

nonparRanks *Nonparametric Rank Number of Reference Interval*

Description

[Experimental]

This data shows the rank number for computing the confidence interval of nonparametric reference limit when the samples within 119-1000 values. But the reference interval must be 95% and the confidence interval is 90%.

Usage

```
nonparRanks
```

Format

A nonparRanks data set contains 882 observations and 3 variables.

SampleSize sample size

Lower lower rank

Upper upper rank

Source

CLSI-EP28A3 Table 8. is cited in this data set.

References

EP28-A3c: Defining, Establishing, and Verifying Reference Intervals in the Clinical Laboratory.

nonparRI *Nonparametric Method in Calculation of Reference Interval*

Description

[Experimental]

This nonparametric method is used to calculate the reference interval when the distribution is skewed and the sample size is above to 120 observations.

Usage

```
nonparRI(x, ind = 1:length(x), conf.level = 0.95)
```

Arguments

x	(numeric) numeric measurements from target population.
ind	(integer) integer vector for boot process, default is all elements in x.
conf.level	(numeric) the percentile of reference limit.

Value

a vector of nonparametric reference interval

Examples

```
data("calcium")
x <- calcium$Value
nonparRI(x)
```

PDL1RP *PD-L1 Reader Precision Data*

Description

[Experimental]

This dummy data set is from a PD-L1 HE stained study to estimate the reproducibility of one assay in determining the PD-L1 status of NSCLC tissue specimens. It contains three sub-data to compute the reproducibility within reader (one pathologists, also called reader here, scores one specimen three times), between reader (three readers scores the same specimen) and between site (one reader in three sites scores the same specimens). These data sets don't have the reference for the each score so it can be only used in the pairwise comparison to calculate the APA, ANA and OPA which don't reply on the reference.

Usage

```
PDL1RP
```

Format

A PDL1RP data set contains 3 sub set, each sub set includes 150 specimens, 450 observations and 4 variables.

Sample Sample id

Site Site id

Order Order of reader scoring

Reader Reader id, the first character represents the site id, and the second character is the reader number

Value Result of scoring, Positive or Negative

pearsonTest *Hypothesis Test for Pearson Correlation Coefficient*

Description

[Experimental]

Adjust the `cor.test` function so that it can define the specific H0 as per your request, that is based on Fisher's Z transformation of the correlation.

Usage

```
pearsonTest(
  x,
  y,
  h0 = 0,
  conf.level = 0.95,
  alternative = c("two.sided", "less", "greater"),
  ...
)
```

Arguments

`x`	(`numeric`) one measurement.
`y`	(`numeric`) another measurement.
`h0`	(`numeric`) a specified hypothesized value of the difference between the two correlations, default is 0.
`conf.level`	(`numeric`) significance level for the returned confidence interval and hypothesis.
`alternative`	(`string`) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
`...`	other arguments to be passed to `cor.test()`.

Value

a named vector contains correlation coefficient (`cor`), confidence interval(`lowerci` and `upperci`), Z statistic (`Z`) and p-value (`pval`)

References

NCSS correlation document

See Also

cor.test() to see the detailed arguments.

Examples

```
x <- c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1)
y <- c(2.6, 3.1, 2.5, 5.0, 3.6, 4.0, 5.2, 2.8, 3.8)
pearsonTest(x, y, h0 = 0.5, alternative = "greater")
```

---

platelet *Quantitative Measurement Data*

Description

[Experimental]

This example platelet can be used to create a data set comparing Platelet results from two analyzers in cells.

Usage

```
platelet
```

Format

A platelet data set contains 120 observations and 3 variables.

Sample Sample id

Comparative Measurements from comparative analyzer

Candidate Measurements from candidate analyzer

Source

CLSI-EP09 A3 Appendix H, Table H2 is cited in this data set.

See Also

From mcr package, mcr::creatinine data set contains data with with serum and plasma creatinin measurements in mg/dL for each sample.

printSummary *Print Summary of a Regression Analysis*

Description

[Experimental]

A copy from mcr::printSummary in `mcr` package

Usage

```
printSummary(...)
```

Arguments

`...`	Arguments passed on to `mcr::printSummary`
	`.Object` object of type "MCResult".

Examples

```
data(platelet)
fit <- mcreg(
  x = platelet$Comparative, y = platelet$Candidate,
  method.reg = "Deming", method.ci = "jackknife"
)
printSummary(fit)
```

qualData *Simulated Qualitative Data*

Description

[Experimental]

This simulated data `qualData` can be used to calculate the qualitative performance such as sensitivity and specificity.

Usage

```
qualData
```

Format

A qualData data set contains 200 observations and 3 variables.

Sample Sample id

ComparativeN Measurements from comparative analyzer with `1=positive` and `0=negative`

CandidateN Measurements from candidate analyzer with `1=positive` and `0=negative`

See Also

platelet that contains quantitative data comparing platelet results from two analyzers.

`RefInt-class` *Reference Interval Class*

Description

[Experimental]

The `RefInt` class serves as the store for results in reference Interval calculation.

Usage

```
RefInt(call, method, n, data, outlier, refInt, confInt)
```

Arguments

`call`	(`call`) function call.
`method`	(`character`) method names of reference interval and confidence interval.
`n`	(`numeric`) number of available samples.
`data`	(`numeric`) numeric raw measurements, no outlier removed.
`outlier`	(`list`) list of outliers that contains the index and number of outliers, and the data without outliers.
`refInt`	(`numeric`) number of reference interval.
`confInt`	(`list`) list of the confidence interval of lower and upper of reference limit.

Value

An object of class `RefInt`.

Slots

`call` call

`method` method

`n` n

`data` data

`outlier` outlier

`refInt` refInt

`confInt` confInt

refInterval *Calculate Reference Interval and Corresponding Confidence Interval*

Description

[Experimental]

This function is used to establish the reference interval for target population with parametric, non-parametric and robust methods that follows the CLSI-EP28A3 and NMPA guideline. In additional, it also provides the corresponding confidence interval for lower/upper reference limit if needed. Given that outliers should be identified beforehand, Tukey and Dixon methods can be applied depending on distribution of the data.

Usage

```
refInterval(
  x,
  out_method = c("doxin", "tukey"),
  out_rm = FALSE,
  RI_method = c("parametric", "nonparametric", "robust"),
  CI_method = c("parametric", "nonparametric", "boot"),
  refLevel = 0.95,
  bootCI = c("perc", "norm", "basic", "stud", "bca"),
  confLevel = 0.9,
  rng.seed = NULL,
  tol = 1e-06,
  R = 10000
)
```

Arguments

x	(numeric) numeric measurements from target population.
out_method	(string) string specifying the which outlier detection to use.
out_rm	(logical) whether the outliers is removed or not.
RI_method	(string) string specifying the which method for computing reference interval to use. Default is parametric, options can be nonparametric and robust.
CI_method	(string) string specifying the which method for computing confidence interval of reference limit(lower or upper) to use. Default is parametric, options can be nonparametric and boot.
refLevel	(numeric) reference range/interval, usual is 0.95.

bootCI (string)
string specifying the which bootstrap confidence interval from boot.ci() function in boot package. Default is perc(bootstrap percentile), options can be norm(normal approximation), boot(basic bootstrap), stud(studentized bootstrap) and bca(adjusted bootstrap percentile).

confLevel (numeric)
significance level for the confidence interval of reference limit.

rng.seed (integer)
number of the random number generator seed for bootstrap sampling. If set to NULL currently in the R session used RNG setting will be used.

tol (numeric)
tolerance for when the iterative process can be stopped in robust method.

R (integer)
number of bootstrap replicates, is used in boot() function.

Value

A RefInt object contains relevant results in establishing of reference interval.

Note

There are some conditions of use to be aware of:

- If parametric method is used to calculate reference interval, confidence interval should be the same method as well.
- If non-parametric method is used to calculate the reference interval and the sample size is up to 120 observations, the non-parametric is suggested for confidence interval. Otherwise if the sample size is below to 120, the bootstrap method is the better choice. Beside the non-parametric method for confidence interval only allows the refLevel=0.95 and confLevel=0.9 arguments, if not the bootstrap methods will be used automatically.
- If robust method is used to calculate the reference interval, the method for confidence interval must be bootstrap.

Examples

```
data("calcium")
x <- calcium$Value
refInterval(x, RI_method = "parametric", CI_method = "parametric")
refInterval(x, RI_method = "nonparametric", CI_method = "nonparametric")
refInterval(x, RI_method = "robust", CI_method = "boot", R = 1000)
```

robustRI *Robust Method in Calculation of Reference Interval*

Description

[Experimental]

This robust method is used to calculate the reference interval on small sample size (below to 120 observations).

Usage

```
robustRI(x, ind = 1:length(x), conf.level = 0.95, tol = 1e-06)
```

Arguments

x	(numeric) numeric measurements from target population.
ind	(integer) integer vector for boot process, default is all elements in x.
conf.level	(numeric) significance level for the internal t statistic.
tol	(numeric) tolerance for when the iterative process can be stopped.

Value

a vector of robust reference interval

References

This robust algorithm is referring to CLSI document EP28A3.

Examples

```
# This example data is taken from EP28A3 Appendix B. to ensure the result is in accordance.
x <- c(8.9, 9.2, rep(9.4, 2), rep(9.5, 3), rep(9.6, 4), rep(9.7, 5), 9.8, rep(9.9, 2), 10.2)
robustRI(x)
```

SampleSize-class *SampleSize Class*

Description

[Experimental]

The SampleSize class serves as the store for results and parameters in sample size calculation.

Usage

```
SampleSize(call, method, n, param)
```

Arguments

call	(call) function call.
method	(character) method name.
n	(numeric) number of sample size.
param	(list) list of relevant parameters.

Value

An object of class SampleSize.

Slots

call call

method method

n n

param param

show,SampleSize-method *Show Method for Objects*

Description

[Experimental]

A show method that displays essential information of objects.

Usage

```
## S4 method for signature 'SampleSize'
show(object)

## S4 method for signature 'MCTab'
show(object)

## S4 method for signature 'BAsummary'
show(object)

## S4 method for signature 'RefInt'
show(object)

## S4 method for signature 'tpROC'
show(object)

## S4 method for signature 'Desc'
show(object)
```

Arguments

object	(any) input.

Value

None (invisible NULL), only used for the side effect of printing to the console.

Examples

```
# Sample zie calculation
size_one_prop(p1 = 0.95, p0 = 0.9, alpha = 0.05, power = 0.8)
size_ci_corr(r = 0.9, lr = 0.85, alpha = 0.025, alternative = "greater")

# Get 2x2 Contingency Table
qualData %>% diagTab(formula = ~ CandidateN + ComparativeN)

# Bland-Altman analysis
```

```
data("platelet")
blandAltman(x = platelet$Comparative, y = platelet$Candidate)

# Reference Interval
data("calcium")
refInterval(x = calcium$Value, RI_method = "nonparametric", CI_method = "nonparametric")

# Comparing the Paired ROC when Non-inferiority margin <= -0.1
data("ldlroc")
aucTest(
  x = ldlroc$LDL, y = ldlroc$OxLDL, response = ldlroc$Diagnosis,
  method = "non-inferiority", h0 = -0.1
)
data(adsl_sub)

# Count multiple variables by treatment
adsl_sub %>%
  descfreq(
    var = c("AGEGR1", "SEX", "RACE"),
    bygroup = "TRTP",
    format = "xx (xx.x%)",
    addtot = TRUE,
    na_str = "0"
  )

# Summarize multiple variables by treatment
adsl_sub %>%
  descvar(
    var = c("AGE", "BMIBL", "HEIGHTBL"),
    bygroup = "TRTP",
    stats = c("N", "MEANSD", "MEDIAN", "RANGE", "IQR"),
    autodecimal = TRUE,
    addtot = TRUE
  )
```

`size_ci_corr` *Sample Size for Testing Confidence Interval of Pearson's correlation*

Description

[Experimental]

This function performs sample size computation for testing Pearson's correlation when a lower confidence interval is provided.

Usage

```
size_ci_corr(
  r,
  lr,
```

```
  alpha = 0.05,
  interval = c(10, 1e+05),
  tol = 1e-05,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

r	(numeric) expected correlation coefficient of the evaluated assay.
lr	(numeric) acceptable correlation coefficient of the evaluated assay.
alpha	(numeric) type-I-risk, α.
interval	(numeric) a numeric vector containing the end-points of the interval to be searched for the root(sample size). The defaults are set to c(1, 100000).
tol	(numeric) tolerance for searching the root(sample size).
alternative	(string) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".

Value

an object of `size` class that contains the sample size and relevant parameters.

References

Fisher (1973, p. 199).

See Also

`size_one_prop()` `size_ci_one_prop()` `size_corr()`

Examples

```
size_ci_corr(r = 0.9, lr = 0.85, alpha = 0.025, alternative = "greater")
```

`size_ci_one_prop` *Sample Size for Testing Confidence Interval of One Proportion*

Description

[Experimental]

This function performs sample size computation for testing a given lower confidence interval of one proportion with the using of the Simple Asymptotic(Wald), Wilson score, clopper-pearson and other methods.

Usage

```
size_ci_one_prop(
  p,
  lr,
  alpha = 0.05,
  interval = c(1, 1e+05),
  tol = 1e-05,
  alternative = c("two.sided", "less", "greater"),
  method = c("simple-asymptotic", "wilson", "wald", "clopper-pearson")
)
```

Arguments

`p`	(`numeric`) expected criteria of the evaluated assay.
`lr`	(`numeric`) acceptable criteria of the evaluated assay.
`alpha`	(`numeric`) type-I-risk, α.
`interval`	(`numeric`) a numeric vector containing the end-points of the interval to be searched for the root(sample size). The defaults are set to c(1, 100000).
`tol`	(`numeric`) tolerance for searching the root(sample size).
`alternative`	(`string`) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
`method`	(`string`) string specifying the which method to use. Simple Asymptotic is default, equal to Wald. Options can be "wilson", "clopper-pearson" and other method, see DescTools::BinomCIn

Value

an object of `size` class that contains the sample size and relevant parameters.

References

Newcombe, R. G. 1998. 'Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods.' Statistics in Medicine, 17, pp. 857-872.

See Also

`size_one_prop()` `size_corr()` `size_ci_corr()`

Examples

```
size_ci_one_prop(p = 0.85, lr = 0.8, alpha = 0.05, method = "wilson")
size_ci_one_prop(p = 0.85, lr = 0.8, alpha = 0.05, method = "simple-asymptotic")
size_ci_one_prop(p = 0.85, lr = 0.8, alpha = 0.05, method = "wald")
```

---

size_corr *Sample Size for Testing Pearson's correlation*

Description

[Experimental]

This function performs sample size computation for testing Pearson's correlation, using uses Fisher's classic z-transformation to normalize the distribution of Pearson's correlation coefficient.

Usage

```
size_corr(
  r1,
  r0,
  alpha = 0.05,
  power = 0.8,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

`r1`	(`numeric`) expected correlation coefficient of the evaluated assay.
`r0`	(`numeric`) acceptable correlation coefficient of the evaluated assay.
`alpha`	(`numeric`) type-I-risk, α.
`power`	(`numeric`) Power of test, equal to 1 minus type-II-risk (β).
`alternative`	(`string`) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".

Value

an object of `size` class that contains the sample size and relevant parameters.

References

Fisher (1973, p. 199).

See Also

`size_one_prop()` `size_ci_one_prop()` `size_ci_corr()`

Examples

```
size_corr(r1 = 0.95, r0 = 0.9, alpha = 0.025, power = 0.8, alternative = "greater")
```

`size_one_prop` *Sample Size for Testing One Proportion*

Description

[Experimental]

This function performs sample size computation for testing one proportion in accordance with Chinese NMPA's IVD guideline.

Usage

```
size_one_prop(
  p1,
  p0,
  alpha = 0.05,
  power = 0.8,
  alternative = c("two.sided", "less", "greater")
)
```

Arguments

`p1`	(`numeric`) expected criteria of the evaluated assay.
`p0`	(`numeric`) acceptable criteria of the evaluated assay.
`alpha`	(`numeric`) type-I-risk, α.
`power`	(`numeric`) Power of test, equal to 1 minus type-II-risk (β).
`alternative`	(`string`) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".

Value

an object of `size` class that contains the sample size and relevant parameters.

References

Chinese NMPA's IVD technical guideline.

See Also

`size_ci_one_prop()` `size_corr()` `size_ci_corr()`

Examples

```
size_one_prop(p1 = 0.95, p0 = 0.9, alpha = 0.05, power = 0.8)
```

---

spearmanTest *Hypothesis Test for Spearman Correlation Coefficient*

Description

[Experimental]

Providing the confidence interval of Spearman's rank correlation by Bootstrap, and define the specific H0 as per your request, that is based on Fisher's Z transformation of the correlation but with the variance recommended by Bonett and Wright (2000), not the same as Pearson's.

Usage

```
spearmanTest(
  x,
  y,
  h0 = 0,
  conf.level = 0.95,
  alternative = c("two.sided", "less", "greater"),
  nrep = 1000,
  rng.seed = NULL,
  ...
)
```

Arguments

`x`	(`numeric`) one measurement.
`y`	(`numeric`) another measurement.
`h0`	(`numeric`) a specified hypothesized value of the difference between the two correlations, default is 0.

conf.level	(numeric) significance level for the returned confidence interval and hypothesis.
alternative	(string) string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
nrep	(integer) number of replicates for bootstrapping, default is 1000.
rng.seed	(integer) number of the random number generator seed for bootstrap sampling. If set to NULL currently in the R session used RNG setting will be used.
...	other arguments to be passed to cor.test().

Value

a named vector contains correlation coefficient (cor), confidence interval(lowerci and upperci), Z statistic (Z) and p-value (pval)

References

NCSS correlation document

See Also

cor.test() boot::boot() to see the detailed arguments.

Examples

```
x <- c(44.4, 45.9, 41.9, 53.3, 44.7, 44.1, 50.7, 45.2, 60.1)
y <- c(2.6, 3.1, 2.5, 5.0, 3.6, 4.0, 5.2, 2.8, 3.8)
spearmanTest(x, y, h0 = 0.5, alternative = "greater")
```

---

tpROC-class *Test for Paired ROC Class*

Description

[Experimental]

The tpROC class serves as the store for results in testing the AUC of paired two-sample assays.

Usage

```
tpROC(testROC, refROC, method, H0, stat)
```

Arguments

testROC	(list) object from pRPC::roc() function for test assay.
refROC	(list) object from pRPC::roc() function for reference/standard assay.
method	(character) method of hypothesis test.
H0	(numeric) margin of test.
stat	(list) list that contains the difference comparing results, such as the difference of AUC, standard error, confidence interval, Z statistic and P value.

Value

An object of class tpROC.

Slots

testROC testROC

refROC refROC

method method

stat stat

---

tukey_outlier *Detect Tukey Outlier*

Description

[Experimental]

Help function detects the potential outlier with Tukey method where the number is below Q1-1.5*IQR and above Q3+1.5*IQR.

Usage

```
tukey_outlier(x)
```

Arguments

x	(numeric) numeric input

Value

A list contains outliers and vector without outliers.

Examples

```
x <- c(13.6, 44.4, 45.9, 14.9, 41.9, 53.3, 44.7, 95.2, 44.1, 50.7, 45.2, 60.1, 89.1)
tukey_outlier(x)
```

`VCAinference` *Inferential Statistics for VCA-Results*

Description

[Experimental]

A copy from VCA::VCAinference in `VCA` package

Usage

```
VCAinference(...)
```

Arguments

`...` Arguments passed on to `VCA::VCAinference`

`obj` (object) of class 'VCA' or, alternatively, a list of 'VCA' objects, where all other arguments can be specified as vectors, where the i-th vector element applies to the i-th element of 'obj' (see examples)

`alpha` (numeric) value specifying the significance level for $100 * (1 - alpha)\%$ confidence intervals.

`total.claim` (numeric) value specifying the claim-value for the Chi-Squared test for the total variance (SD or CV, see `claim.type`).

`error.claim` (numeric) value specifying the claim-value for the Chi-Squared test for the error variance (SD or CV, see `claim.type`).

`claim.type` (character) one of "VC", "SD", "CV" specifying how claim-values have to be interpreted:
"VC" (Default) = claim-value(s) specified in terms of variance(s),
"SD" = claim-values specified in terms of standard deviations (SD),
"CV" = claim-values specified in terms of coefficient(s) of variation (CV) and are specified as percentages.
If set to "SD" or "CV", claim-values will be converted to variances before applying the Chi-Squared test (see examples).

`VarVC` (logical) TRUE = the covariance matrix of the estimated VCs will be computed (see `vcovVC`), where diagonal elements correspond to the variances of the individual VCs. This matrix is required for estimation of CIs for intermediate VCs if 'method.ci="sas"'. FALSE (Default) = computing covariance matrix of VCs is omitted, as well as CIs for intermediate VCs.

`excludeNeg` (logical) TRUE = confidence intervals of negative variance estimates will not be reported.
FALSE = confidence intervals for all VCs will be reported including those with negative VCs.
See the details section for a thorough explanation.

`constrainCI` (logical) TRUE = CI-limits for all variance components are constrained to be >= 0.
FALSE = unconstrained CIs with potentially negative CI-limits will be reported.
which will preserve the original width of CIs. See the details section for a thorough explanation.

`ci.method` (character) string or abbreviation specifying which approach to use for computing confidence intervals of variance components (VC). "sas" (default) uses Chi-Squared based CIs for total and error and normal approximation for all other VCs (Wald-limits, option "NOBOUND" in SAS PROC MIXED); "satterthwaite" will approximate DFs for each VC using the Satterthwaite approach (see `SattDF` for models fitted by ANOVA) and all Cis are based on the Chi-Squared distribution. This approach is conservative but avoids negative values for the lower bounds.

`quiet` (logical) TRUE = will suppress any warning, which will be issued otherwise

Value

object of `VCAinference` contains a series of statistics.

See Also

`VCA::VCAinference()`

Examples

```
data(glucose)
fit <- anovaVCA(value ~ day / run, glucose)
VCAinference(fit)
```

Index

∗ **datasets**
adsl_sub, 3
calcium, 14
glucose, 29
ldlroc, 35
nonparRanks, 39
PDL1RP, 40
platelet, 42
qualData, 43

adsl_sub, 3, *4*
anovaVCA, 4
aucTest, 5
autoplot, 7
autoplot,BAsummary-method *(*autoplot*)*, 7
autoplot,MCResult-method *(*autoplot*)*, 7

BAsummary, *12*
BAsummary *(*BAsummary-class*)*, 11
BAsummary-class, 11
blandAltman, 12, *28*
boot::boot(), *56*

calcBias, 13
calcium, *14*, 14
cat(), *14*
cat_with_newline, 14
cli::cat_line(), *15*
cor.test(), *41, 42*, *56*

Desc *(*Desc-class*)*, 15
Desc-class, 15
descfreq, 16
DescTools::BinomCI, *25*
DescTools::BinomCIn, *52*
descvar, 18
diagTab, 20, *25*
dixon_outlier, 22

esd.critical, 22
ESD_test, 23

factor(), *31*
formatC(), *33*
formatters::format_value, *32*
formula, *20*

getAccuracy, 24
getAccuracy,MCTab-method *(*getAccuracy*)*, 24
getCoefficients, 27
getOutlier, 28
getOutlier,BAsummary-method *(*getOutlier*)*, 28
glucose, *29*, 29

h_difference, 30
h_difference(), *10*, *12*
h_factor, 31
h_fmt_count_perc, 31
h_fmt_est, 32
h_fmt_num, 33
h_fmt_num(), *33, 34*
h_fmt_range, 34
h_summarize, 35

ldlroc, 35, *36*

mcr::calcBias, *13*
mcr::calcBias(), *13*
mcr::creatinine, *42*
mcr::getCoefficients, *27*
mcr::mcreg, *36*
mcr::mcreg(), *9, 10*, *38*
mcr::printSummary, *43*
mcradds *(*mcradds-package*)*, 3
mcradds-package, 3
mcreg, 36
MCTab, *24*
MCTab *(*MCTab-class*)*, 38
MCTab-class, 38

nonparRanks, *39*, 39

nonparRI, 39

PDL1RP, *40*, 40
pearsonTest, 41
platelet, *42*, 42, *44*
printSummary, 43
pROC::roc(), *6*

qualData, *43*, 43
quantile(), *19*

RefInt *(*RefInt-class*)*, 44
RefInt-class, 44
refInterval, 45
robustRI, 47

SampleSize *(*SampleSize-class*)*, 48
SampleSize-class, 48
SattDF, *59*
show *(*show,SampleSize-method*)*, 49
show,BAsummary-method
        *(*show,SampleSize-method*)*, 49
show,Desc-method
        *(*show,SampleSize-method*)*, 49
show,MCTab-method
        *(*show,SampleSize-method*)*, 49
show,RefInt-method
        *(*show,SampleSize-method*)*, 49
show,SampleSize-method, 49
show,tpROC-method
        *(*show,SampleSize-method*)*, 49
size_ci_corr, 50
size_ci_corr(), *53–55*
size_ci_one_prop, 52
size_ci_one_prop(), *51*, *54, 55*
size_corr, 53
size_corr(), *51*, *53*, *55*
size_one_prop, 54
size_one_prop(), *51*, *53, 54*
spearmanTest, 55
Summary(), *21*

table(), *38*
tpROC *(*tpROC-class*)*, 56
tpROC-class, 56
tukey_outlier, 57

VCA::anovaVCA, *4*
VCA::anovaVCA(), *5*
VCA::VCAinference, *58*
VCA::VCAinference(), *59*
VCAinference, 58
vcovVC, *58*