Package ‘descsuppR’

October 1, 2025

Version 1.2

License GPL-3

Title Support Functions for (Reproducible) Descriptive Statistics

Description Contains functions to help with generating tables with descriptive statistics. In addition, the package can display results of statistical tests for group comparisons. A wide range of test procedures is supported, and user-defined test functions can be incorporated.

Depends foreach

Imports plyr, descutils, tibble, dplyr, rlang, DescTools, nparcomp, rankFD, circular, glue, purrr

Suggests roxygen2, survival, stringr

Encoding UTF-8

Collate 'descsuppR-package.r' 'tod.r'

RoxygenNote 7.2.3

NeedsCompilation no

Author Andreas Leha [aut],
Fabian Kück [aut, cre]

Maintainer Fabian Kück `<fabian.kueck@med.uni-goettingen.de>`

Repository CRAN

Date/Publication 2025-10-01 07:10:08 UTC

Contents

`as.tod` *Conversion to and from 'Time of Day'*

Description

Does not safeguard against "26:69".

Usage

```
as.tod(x)

## S3 method for class 'character'
as.tod(x)

## S3 method for class 'circular'
as.tod(x)

is.tod(x)

## S3 method for class 'tod'
as.double(x, ...)

circular(x, ...)
```

```
## Default S3 method:
circular(
  x,
  type = c("angles", "directions"),
  units = c("radians", "degrees", "hours"),
  template = c("none", "geographics", "clock12", "clock24"),
  modulo = c("asis", "2pi", "pi"),
  zero = 0,
  rotation = c("counter", "clock"),
  names = NULL,
  ...
)

## S3 method for class 'tod'
circular(x, ...)
```

Arguments

`x`	An object to be converted to or represented as a circular object.
`...`	Additional arguments passed to methods or lower-level functions.
`type`	Character string indicating if data represent "angles" or "directions".
`units`	Character string; the measurement units for the input data (`"radians"`, `"degrees"`, or `"hours"`).
`template`	Character string specifying a specific template for circular data.
`modulo`	Character string indicating the modulo arithmetic to be used (`"asis"`, `"2pi"`, `"pi"`).
`zero`	Numeric; direction assigned as zero (in the specified units).
`rotation`	Character string; direction of rotation for increasing values (`"counter"`, `"clock"`).
`names`	Optional character vector of names for the object.

Value

x converted to/from 'tod'

Author(s)

Dr. Andreas Leha

Examples

```
times <- c("8:53", NA, "22:30")

## some conversions
as.tod(times)
as.numeric(as.tod(times))

is.tod(times)
is.tod(as.tod(times))
```

`buildDescrTbl` *buildDescrTbl*

Description

Calculate and Present Descriptive Values in Pritable

Usage

```
buildDescrTbl(
  df,
  tests,
  prmnames,
  prmunits,
  addFactorLevelsToNames = TRUE,
  excel_style = TRUE,
  groupby,
  addungrouped = FALSE,
  dopvals = FALSE,
  ignore_test_errors = FALSE,
  p.adjust.method = "holm",
  orderedAsUnordered = FALSE,
  factorlevellimit = 14,
  show.minmax = TRUE,
  show.IQR = FALSE,
  report_tests = FALSE,
  report_testmessages = FALSE,
  pvals_formatting = TRUE,
  pvals_digits = 3,
  pvals_signiflev = 0.05,
  extraLevels = NULL,
  missingName = "missing",
  nonNAsName = "N",
  removeZeroNAs = TRUE,
  removeZeroExtraLevels = TRUE,
  includeNAs = FALSE,
  includeNonNAs = FALSE,
  printOrgAlignment = FALSE,
  useutf8 = "latex",
  verbose = 0,
  without_attrs = FALSE,
  sd_digits = "by_mean",
  descr_digits = 2,
  significant_digits = TRUE,
  percentages = "columnwise"
)
```

Arguments

`df`	data.frame containing the variables of which to calc the descriptive values
`tests`	character vector or list of characters or list of functions or list of lists. In each case the i-th element gives the test to perform on the ith variable in the df (excluding stratification variables). The test can either be given as character (name of test function) or as function or as list where the first element is again either character or function and the following elemenst are *named* additional arguments to that test function. The individual function has to accept (at least) the arguments 'values' and 'grouping' which are vectors of equal length. For convenience, this package shipes with some example functions; have a look at those if you want to supply your own. These convenience functions include w.chisq.test w.cor.test, w.fisher.test, w.kruskal.test, w.wilcox.test. the whole list/vector is recycled if too short.
`prmnames`	names of the variables in df (if needed to be overwritten)
`prmunits`	units of the variables in df
`addFactorLevelsToNames`	logical. if `TRUE` expand 'sex' to 'sex [m/w]'. Defaults to `TRUE`.
`excel_style`	logical. if TRUE remove subsequent duplicates from the `parameter` column (as common in Excel). Default: `TRUE`
`groupby`	column of df. do more columns - one for each group. If the df$column is an ordered factor, the order will be respected in the resulting table
`addungrouped`	logical. if `TRUE` add a column 'total' with the ungrouped summary statistics. Default: `FALSE`
`dopvals`	boolean. if TRUE an additional column containing the p-values comparing the two strata in `groubpy`. Only implemented for a two-level stratum until now.
`ignore_test_errors`	logical. If `TRUE` returns an empty test results (as list).
`p.adjust.method`	character. if not NULL include an additional column with adjusted p values. see `p.adjust.methods` for possible values and explanations. Defaults to "holm"
`orderedAsUnordered`	logical. treat ordered factors as unordered factors?
`factorlevellimit`	integer. for factors with more than `factorlevellimit` levels, not all levels are printed
`show.minmax`	logical. if TRUE show minimum and maximum for numeric variables. Defaults to `TRUE`.
`show.IQR`	logical. if TRUE show 25% and 75% quantiles for numeric variables. Defaults to `FALSE`.
`report_tests`	boolean. if TRUE one additional column in the result table will contain the test, that was performed to calculate the p value. Ignored if dopvals=FALSE
`report_testmessages`	boolean. if TRUE one additional column in the result table will contain any warnings that appeared while the test was performed. Ignored if dopvals=FALSE

`pvals_formatting`	boolean. If FALSE report numbers, else report formatted strings (via prettyPvalue)
`pvals_digits`	integer. Number of digits for p value formatting. Ignored when pvals_formatting==FALSE. Defaults to 2
`pvals_signiflev`	double. The significance level for bold p value formatting. Ignored when pvals_formatting==FALSE. Defaults to 0.05
`extraLevels`	named list of lists. Names have to be variable names. Elements have to have to be named list of this form: `` `some label` = list(idxvec = idxvec, display = logical) ``. Here `idxvec` needs to be a logical vector of length `nrow(df)` that specifies the affected rows. If `display` is `TRUE` the number of affected rows will be shown under `some label`.
`missingName`	character. name of missing values (default: `missing`)
`nonNAsName`	character. name of not missing values (default: `N`)
`removeZeroNAs`	boolean. if TRUE, rows for missing values containing only 0s are removed from the result.
`removeZeroExtraLevels`	boolean. if TRUE, rows for ExtraLevels containing only 0s are removed from the result.
`includeNAs`	boolean. Include number of NAs in the output? Currently only one of either `includeNonNAs` or `includeNAs` can be set to `TRUE`
`includeNonNAs`	boolean. Include number of not missing values (Non-NAs) in the output? Currently only one of either `includeNonNAs` or `includeNAs` can be set to `TRUE`
`printOrgAlignment`	boolean. If TRUE, than a row like "<l> <r> <r>" will be included in the result df
`useutf8`	character. one of c("latex", "utf8", "replace"). if 'latex' (the default) use \pm in the output; if 'replace' use +- in the output, if 'utf8' use the unicode character
`verbose`	numeric. level of verbosity (0 : silent)
`without_attrs`	logical. If `TRUE` return the descriptive table with attrs. Otherwise add `df`, `groupby`, and a 'full' (closer to tidy) version of the table as attributes. Defaults to `TRUE`.
`sd_digits`	character. one of c("by_mean", "fixed"). If 'by_mean', the number of decimal places of the standard deviation is limited by the number of decimal places of the mean.
`descr_digits`	integer. Number of digits for formatting of descriptive values. Defaults to 2.
`significant_digits`	boolean. if TRUE, the number of significant digits of is given by descr_digits. Otherwise the number of decimal places is fixed.
`percentages`	character. one of c("columnwise", "rowwise"). If 'rowwise', percentages are computed by row. Defaults to "columnwise"

Details

Do a Table containing descriptiva values

Value

formatted `data.frame` with descriptive values

Author(s)

Andreas Leha

Examples

```
ttt <- data.frame(data="training set",
                  age=runif(100, 0, 100),
                  sex=sample(c("m","f"), 100, replace=TRUE, prob=c(0.3, 0.7)),
                  score=factor(sample(1:5, 100, replace=TRUE),
                    ordered=TRUE,
                    levels=1:5))
ttt2 <- data.frame(data="test set",
                   age=runif(100, 0, 100),
                   sex=sample(c("m","f"), 100, replace=TRUE, prob=c(0.5,0.5)),
                   score=factor(sample(1:5, 100, replace=TRUE),
                     ordered=TRUE,
                     levels=1:5))

units <- c("years", "", "")
buildDescrTbl(rbind(ttt, ttt2),
              prmunits=units,
              groupby="data",
              includeNAs=TRUE)
```

`buildDescrTbl.intern` *buildDescrTbl.intern*

Description

Internal wrapper around calc_descr_matrix

Usage

```
buildDescrTbl.intern(
  df,
  prmnames,
  prmunits,
  addFactorLevelsToNames = TRUE,
  extraLevels = NULL,
  includeNAs = FALSE,
  includeNonNAs = FALSE,
  orderedAsUnordered = FALSE,
  factorlevellimit = 14,
  show.minmax = TRUE,
```

```
  show.IQR = FALSE,
  sd_digits = "by_mean",
  descr_digits = 2,
  significant_digits = TRUE
)
```

Arguments

`df`	data.frame containing the variables of which to calc the descriptive values
`prmnames`	names of the variables in df (if needed to be overwritten)
`prmunits`	units of the variables in df
`addFactorLevelsToNames`	logical. if `TRUE` expand ’sex’ to ’sex [m/w]’. Defaults to `TRUE`.
`extraLevels`	named list of lists. Names have to be variable names. Elements have to have to be named list of this form: `` `some label` = list(idxvec = idxvec, display = logical) ``. Here `idxvec` needs to be a logical vector of length `nrow(df)` that specifies the affected rows. If `display` is `TRUE` the number of affected rows will be shown under `some label`.
`includeNAs`	boolean. Include number of NAs in the output? Currently only one of either `includeNonNAs` or `includeNAs` can be set to `TRUE`
`includeNonNAs`	boolean. Include number of not missing values (Non-NAs) in the output? Currently only one of either `includeNonNAs` or `includeNAs` can be set to `TRUE`
`orderedAsUnordered`	logical. treat ordered factors as unordered factors?
`factorlevellimit`	integer. for factors with more than `factorlevellimit` levels, not all levels are printed
`show.minmax`	logical. if TRUE show minimum and maximum for numeric variables. Defaults to `TRUE`.
`show.IQR`	logical. if TRUE show 25% and 75% quantiles for numeric variables. Defaults to `FALSE`.
`sd_digits`	character. one of c("by_mean", "fixed"). If ’by_mean’, the number of decimal places of the standard deviation is limited by the number of decimal places of the mean.
`descr_digits`	integer. Number of digits for formatting of descriptive values. Defaults to 2.
`significant_digits`	boolean. if TRUE, the number of significant digits of is given by descr_digits. Otherwise the number of decimal places is fixed.

Value

matrix with descriptive values

Author(s)

Andreas Leha

`calc_descr_matrix` *calc_descr_matrix*

Description

Wrapper for single vectors

Usage

```
calc_descr_matrix(
  ttt,
  format = "long",
  extraLevels = NULL,
  includeNAs = FALSE,
  includeNonNAs = FALSE,
  orderedAsUnordered = FALSE,
  factorlevellimit = 14,
  show.minmax = TRUE,
  show.IQR = FALSE,
  sd_digits = "by_mean",
  descr_digits = 2,
  significant_digits = TRUE
)
```

Arguments

`ttt`	the data.frame
`format`	in c("long", "wide")
`extraLevels`	named list of lists. Names have to be variable names. Elements have to have to be named list of this form: `` `some label` = list(idxvec = idxvec, display = logical) ``. Here `idxvec` needs to be a logical vector of length `nrow(df)` that specifies the affected rows. If `display` is `TRUE` the number of affected rows will be shown under `some label`.
`includeNAs`	boolean. include number of NAs in the output?
`includeNonNAs`	boolean. include number of non-NAs in the output?
`orderedAsUnordered`	logical. treat ordered factors as unordered factors?
`factorlevellimit`	integer. for factors with more than `factorlevellimit` levels, not all levels are printed
`show.minmax`	logical. if TRUE show minimum and maximum for numeric variables. Defaults to `TRUE`.
`show.IQR`	logical. if TRUE show 25% and 75% quantiles for numeric variables. Defaults to `FALSE`.

`sd_digits`	character. one of c("by_mean", "fixed"). If 'by_mean', the number of decimal places of the standard deviation is limited by the number of decimal places of the mean.
`descr_digits`	integer. Number of digits for formatting of descriptive values. Defaults to 2.
`significant_digits`	boolean. if TRUE, the number of significant digits of is given by descr_digits. Otherwise the number of decimal places is fixed.

Details

calls the one dimensional functions

Value

matrix containing the descriptive values

Author(s)

Andreas Leha

`convertColumnHeading` *Convert Grouping Column Values into Headings with ddply*

Description

For each unique value in the specified column of a data frame, creates a "subtable" where that value is moved as a group heading (first row) and the column is blanked for the remaining rows. Optionally, can "move along" specified metadata columns, shifting their first element to the last row.

Usage

```
convertColumnHeading(df, col, movealong = NULL)
```

Arguments

`df`	A data frame to be processed.
`col`	Character. Name of the column in `df` to use for grouping and converting to headings.
`movealong`	Optional character vector of column names. For each group, these columns are rotated, placing their first value after the last row.

Details

The function uses `plyr::ddply` to split the data frame by the levels of `col`. For each group, - Moves the grouping value to a heading (top row) - Adds an empty row for formatting - Optionally rotates metadata columns specified in `movealong` This can be useful for preparing tables for presentation or reporting.

Value

A data frame formatted with expanded headings per group.

Author(s)

Andreas Leha

Examples

```
df <- data.frame(Group = c("A", "A", "B", "B"), Data = 1:4, Meta = c("x", "y", "z", "w"))
convertColumnHeading(df, "Group")
convertColumnHeading(df, "Group", movealong = "Meta")
```

`descrSurvEstimate` *descrSurvEstimate*

Description

build a description table for survival estimates

Usage

```
descrSurvEstimate(
  S,
  strata,
  stratorder,
  survname = "survival time",
  evaltimes = c(3, 5),
  evaltimeunits = "years",
  digits = 2,
  includeNAs = TRUE,
  includeNonNAs = TRUE,
  missingName = "missing",
  nonNAsName = "N",
  stratheader = TRUE,
  pval = FALSE,
  pvals_formatting = TRUE,
  pvals_digits = 2,
  pvals_siglev = 0.05,
  hr = FALSE
)
```

Arguments

`S`	survival objects from `Surv`
`strata`	a list of vectors containing strata. If the vectors are ordered factors the columns will be used in the given order.

`stratorder`	(list of) character vector for the order of the reported columns. Overrides any order of strata
`survname`	the name of the survival time, e.g. 'DFS'
`evaltimes`	numeric vector. for which times to calculate the survival estimate
`evaltimeunits`	the unit of the survival times (years, months, ...)
`digits`	round to
`includeNAs`	boolean. Include number of NAs in the output?
`includeNonNAs`	Logical; if `TRUE`, include a column reporting the number of non-missing (non-NA) observations in the output table. Default is `TRUE`.
`missingName`	character. name of the rows with missing numbers. Defaults to "missing".
`nonNAsName`	Character string; the name to use as the column heading for the number of non-missing (non-NA) values. Default is `"N"`.
`stratheader`	boolean. print the stratheader? Turn off for inclusion into a bigger table
`pval`	boolean. if TRUE, the p value from a cox model is printed in a separate column
`pvals_formatting`	boolean. If FALSE report numbers, else report formatted strings (via prettyPvalue)
`pvals_digits`	integer. Number of digits for p value formatting. Ignored when pvals_formatting==FALSE. Defaults to 2
`pvals_signiflev`	double. The significance level for bold p value formatting. Ignored when pvals_formatting==FALSE. Defaults to 0.05
`hr`	boolean. if TRUE, the hazard ratio (with confidence interval) is printed as well. (only has an effect if pval==TRUE)

Details

calculate the survival estimates at the specified times and return a nicely formatted table

Value

a character matrix

Author(s)

Andreas Leha

Examples

```
if (require("survival")) {

  S <- Surv(aml$time, aml$status)

  descrSurvEstimate(S,
                    evaltimes=c(19, 24),
                    evaltimeunits="months")
}
```

`hours.tod` *Extract the hours/minutes Component of time-of-day*

Description

Extract the hours/minutes Component of time-of-day

Usage

```
hours.tod(x)

minutes.tod(x)
```

Arguments

`x`	vector of class tod

Value

numeric vector of length `length(x)` with hours/minutes

Author(s)

Dr. Andreas Leha

Examples

```
times <- as.tod(c("8:53", NA, "22:30"))
hours.tod(times)
minutes.tod(times)
```

`length.tod` *Basic Functions on time-of-day Vectors*

Description

These shoud just do whatever the same function does for character vectors

Usage

```
## S3 method for class 'tod'
length(x)

## S3 method for class 'tod'
x[i, ...]

## S3 replacement method for class 'tod'
x[i, ...] <- value

## S3 method for class 'tod'
x[[i, ...]]

## S3 replacement method for class 'tod'
x[[i, ...]] <- value

## S3 method for class 'tod'
c(...)
```

Arguments

`x`	vector of class `tod`
`i`	integer of indices
`...`	to match the generics' arguments
`value`	of type `tod`. to be inserted

Value

as expected

Author(s)

Dr. Andreas Leha

Examples

```
times <- c("8:53", NA, "22:30")
times <- as.tod(times)

length(times)
times[1]
times[1] <- "07:00"
c(times, times)
```

`mean.tod` *Descriptive Statistics for time-of-day Vectors*

Description

Functions to calculate descriptve values for timo-of-day vectors. The heavy lifting is done by the `circular` package.

Usage

```
## S3 method for class 'tod'
mean(x, ...)

sd(x, ...)

## Default S3 method:
sd(x, na.rm = FALSE, ...)

## S3 method for class 'tod'
sd(x, na.rm = FALSE, ...)

## S3 method for class 'tod'
median(x, na.rm = FALSE, ...)

## S3 method for class 'tod'
min(..., na.rm = FALSE)

## S3 method for class 'tod'
max(..., na.rm = FALSE)

## S3 method for class 'tod'
quantile(x, ...)
```

Arguments

`x`	vector of class `tod`
`...`	additional args passed on to base stats or circular stats functions
`na.rm`	often passed through to correponding functions in the circular packages. Otherwise logitcal. Defaults to `FALSE`.

Details

These functions are meant to provide the least surprising descriptive values.

Value

descriptive values as would be returned for non-circular numeric vectors

Author(s)

Dr. Andreas Leha

Examples

```
times <- as.tod(c("21:53", NA, "22:30", "23:10", "23:58", "00:15", "01:01"))

mean(times)
mean(times, na.rm = TRUE)
```

`pred.survfit` *pred.survfit*

Description

X-years prediction

Usage

```
pred.survfit(Sfit, time)
```

Arguments

`Sfit`	a survfit object
`time`	the time to calculate the estimate

Details

Calculate the X-years estimate of a survfit

Value

the estimate

Author(s)

Andreas Leha

`replaceGermanUmlauts` *Replace German Umlaute*

Description

Replace German Umlaute

Usage

```
replaceGermanUmlauts(txt)
```

Arguments

`txt`	character. within this txt the German umlauts will be replaced

Value

character. version of `txt` with all 'Umlaute' and 'scharfes s' replaced.

Author(s)

Andreas Leha

Examples

```
replaceGermanUmlauts("gefräßig")
```

`testWrapper` *Collect Warnings From Runnning testfun*

Description

wrap this around 'correlation' tests to get output formatted for buildDescrTbl

Usage

```
testWrapper(testfun, values, grouping, ignore_test_errors = FALSE, ...)
```

Arguments

`testfun`	character or function. Which function to call.
`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`ignore_test_errors`	logical. If `TRUE` returns an empty test results (as list).
`...`	additional parameters. are passed on to the testfun

Details

This function is called by buildDescrTbl in order to generate the comparison p values. Basically it just calls the provided testfun. Main purpose is, that it collects warnings and returns them as well.

Value

list. the results from testfun plus the element 'warnings' containing all warnings collected from the run of testfun. the results from testfun are assumed to be of type list and are additionally assumed to contain at least the elements 'p.value' and 'method'.

Author(s)

Andreas Leha

`w.anova.test` *ANOVA with unified interface*

Description

One-way ANOVA that unifies the api to other tests

Usage

```
w.anova.test(values, grouping, na.rm = TRUE, ...)
```

Arguments

`values`	vector. The values to compare (age, gene expression, ...)
`grouping`	vector ef the same length as `values`. treated as factor giving the group membership
`na.rm`	logcical. if TRUE (default) the `values` are subset to only non-missing and finite values
`...`	additional parameters. are passed on to `lm()`

Value

the value of `anova` augmented by 'p.value' and 'method'

Author(s)

Dr. Andreas Leha

`w.chisq.test` *chisq.test with unified interface*

Description

chisq.test with unified interface

Usage

```
w.chisq.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the chisq.test

Details

just a call to chisq.test that unifies the api to other tests

Value

the value chisq.test

Author(s)

Andreas Leha

`w.CochraneArmitageTrend.test` *Cochran-Armitage Test for Trend*

Description

Wrapper for `CochranArmitageTest` which is a test for trend in binomial proportions across the levels of a single variable

Usage

```
w.CochraneArmitageTrend.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare. Will be passed to `circular` to be converted to circular format
`grouping`	of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the `CochranArmitageTest`

Value

the value of `CochranArmitageTest`

Author(s)

Dr. Andreas Leha

`w.cor.test` *cor.test with unified interface*

Description

cor.test with unified interface

Usage

```
w.cor.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the cor.test

Details

just a call to cor.test that unifies the api to other tests

Value

the value cor.test

Author(s)

Andreas Leha

`w.fisher.test` *fisher.test with unified interface*

Description

fisher.test with unified interface

Usage

```
w.fisher.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the fisher.test

Details

just a call to fisher.test that unifies the api to other tests computes the exact p-value if possible and simulates the p-value otherwise

Value

the value fisher.test

Author(s)

Andreas Leha

`w.JonckheereTerpstraTest` *JonckheereTerpstraTest with unified interface*

Description

JonckheereTerpstraTest with unified interface

Usage

```
w.JonckheereTerpstraTest(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the JonckheereTerpstraTest

Details

just a call to JonckheereTerpstraTest that unifies the api to other tests

Value

the value JonckheereTerpstraTest

Author(s)

Fabian Kück

`w.kruskal.test` *kruskal.test with unified interface*

Description

kruskal.test with unified interface

Usage

```
w.kruskal.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the kruskal.test

Details

just a call to kruskal.test that unifies the api to other tests

Value

the value kruskal.test

Author(s)

Andreas Leha

`w.no.test` *no test*

Description

no test but unified interface

Usage

```
w.no.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. unused

Details

just returns NA. Included as pass through for non-testable variables

Value

NA

Author(s)

Andreas Leha

`w.npar.t.test` *npar.t.test with unified interface*

Description

npar.t.test with unified interface

Usage

```
w.npar.t.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the npar.t.test

Details

just a call to npar.t.test that unifies the api to other tests computes the ordinary Brunner-Munzel test for group sizes > 9 and the studentized permutation test version otherwise

Value

the value npar.t.test

Author(s)

Fabian Kück

`w.npar.t.test.permu` *npar.t.test with unified interface*

Description

npar.t.test with unified interface

Usage

```
w.npar.t.test.permu(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the npar.t.test

Details

just a call to npar.t.test that unifies the api to other tests computes the the studentized permutation test version of the Brunner-Munzel test

Value

the value npar.t.test

Author(s)

Fabian Kück

`w.prop.trend.test` *prop.trend.test with unified interface*

Description

prop.trend.test with unified interface

Usage

```
w.prop.trend.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the prop.trend.test

Details

just a call to prop.trend.test that unifies the api to other tests

Value

the value prop.trend.test

Author(s)

Fabian Kück

`w.rankFD.mid.ranks` *rankFD with unified interface*

Description

rankFD with unified interface

Usage

```
w.rankFD.mid.ranks(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the rankFD

Details

just a call to rankFD with effect="weighted" that unifies the api to other tests

Value

the value rankFD

Author(s)

Fabian Kück

`w.rankFD.pseudo.ranks` *rankFD with unified interface*

Description

rankFD with unified interface

Usage

```
w.rankFD.pseudo.ranks(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the rankFD

Details

just a call to rankFD that unifies the api to other tests

Value

the value rankFD

Author(s)

Fabian Kück

`w.t.test` *t.test with unified interface*

Description

just a call to t.test that unifies the api to other tests

Usage

```
w.t.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the t.test

Value

the value of t.test

Author(s)

Andreas Leha

`w.watson.williams.test`
Watson-Williams Test of Homogeneity of Means

Description

just a call to `watson.williams.test` that unifies the api to other tests

Usage

```
w.watson.williams.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare. Will be passed to `circular` to be converted to circular format
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the `watson.williams.test`

Value

the value of `watson.williams.test`

Author(s)

Dr. Andreas Leha

`w.wilcox.test` *wilcox.test with unified interface*

Description

wilcox.test with unified interface

Usage

```
w.wilcox.test(values, grouping, ...)
```

Arguments

`values`	vector. The values to compare (age, toxicity score, gene expression, ...)
`grouping`	vector of the same length as values. treated as factor giving the group membership
`...`	additional parameters. are passed on to the wilcox.test

Details

just a call to wilcox.test that unifies the api to other tests

Value

the value wilcox.test

Author(s)

Andreas Leha

Index