Package 'desctable'

July 11, 2018

Title Produce Descriptive and Comparative Tables Easily

Version 0.1.3

Description Easily create descriptive and comparative tables.
It makes use and integrates directly with the tidyverse family of packages, and pipes.
Tables are produced as data frames/lists of data frames for easy manipulation after creation, and ready to be saved as csv, or piped to DT::datatable() or pander::pander() to integrate into reports.

Depends R (>= 3.2.3)

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/maximewack/desctable

BugReports https://github.com/maximewack/desctable/issues

Imports dplyr, purrr, DT, htmltools, pander

Suggests knitr, rmarkdown, survival

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Maxime Wack [aut, cre]

Maintainer Maxime Wack <maximewack@free.fr>

Repository CRAN

Date/Publication 2018-07-11 14:20:02 UTC

R topics documented:

`ANOVA` *Wrapper for oneway.test(var.equal = T)*

Description

Wrapper for oneway.test(var.equal = T)

Usage

```
ANOVA(formula)
```

Arguments

`formula`	An anova formula (`variable ~ grouping variable`)

See Also

`oneway.test`

as.data.frame.desctable *As.data.frame method for desctable*

Description

As.data.frame method for desctable

Usage

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

Arguments

x	A desctable
...	Additional as.data.frame parameters

Value

A flat dataframe

chisq.test *Pearson's Chi-squared Test for Count Data*

Description

chisq.test performs chi-squared contingency table tests and goodness-of-fit tests, with an added method for formulas.

Usage

```
chisq.test(x, y, correct, p, rescale.p, simulate.p.value, B)

## Default S3 method:
chisq.test(x, y = NULL, correct = TRUE,
  p = rep(1/length(x), length(x)), rescale.p = FALSE,
  simulate.p.value = FALSE, B = 2000)

## S3 method for class 'formula'
chisq.test(x, y = NULL, correct = T,
  p = rep(1/length(x), length(x)), rescale.p = F, simulate.p.value = F,
  B = 2000)
```

Arguments

x	a numeric vector, or matrix, or formula of the form `lhs ~ rhs` where `lhs` and `rhs` are factors. `x` and `y` can also both be factors.
y	a numeric vector; ignored if `x` is a matrix or a formula. If `x` is a factor, `y` should be a factor of the same length.
correct	a logical indicating whether to apply continuity correction when computing the test statistic for 2 by 2 tables: one half is subtracted from all $\|O-E\|$ differences; however, the correction will not be bigger than the differences themselves. No correction is done if `simulate.p.value = TRUE`.
p	a vector of probabilities of the same length of `x`. An error is given if any entry of `p` is negative.
rescale.p	a logical scalar; if TRUE then `p` is rescaled (if necessary) to sum to 1. If `rescale.p` is FALSE, and `p` does not sum to 1, an error is given.
simulate.p.value	a logical indicating whether to compute p-values by Monte Carlo simulation.
B	an integer specifying the number of replicates used in the Monte Carlo test.

Details

If `x` is a matrix with one row or column, or if `x` is a vector and `y` is not given, then a _goodness-of-fit test_ is performed (`x` is treated as a one-dimensional contingency table). The entries of `x` must be non-negative integers. In this case, the hypothesis tested is whether the population probabilities equal those in `p`, or are all equal if `p` is not given.

If `x` is a matrix with at least two rows and columns, it is taken as a two-dimensional contingency table: the entries of `x` must be non-negative integers. Otherwise, `x` and `y` must be vectors or factors of the same length; cases with missing values are removed, the objects are coerced to factors, and the contingency table is computed from these. Then Pearson's chi-squared test is performed of the null hypothesis that the joint distribution of the cell counts in a 2-dimensional contingency table is the product of the row and column marginals.

If `simulate.p.value` is `FALSE`, the p-value is computed from the asymptotic chi-squared distribution of the test statistic; continuity correction is only used in the 2-by-2 case (if `correct` is `TRUE`, the default). Otherwise the p-value is computed for a Monte Carlo test (Hope, 1968) with `B` replicates.

In the contingency table case simulation is done by random sampling from the set of all contingency tables with given marginals, and works only if the marginals are strictly positive. Continuity correction is never used, and the statistic is quoted without it. Note that this is not the usual sampling situation assumed for the chi-squared test but rather that for Fisher's exact test.

In the goodness-of-fit case simulation is done by random sampling from the discrete distribution specified by `p`, each sample being of size `n = sum(x)`. This simulation is done in R and may be slow.

Value

A list with class `"htest"` containing the following components: statistic: the value the chi-squared test statistic.

parameter: the degrees of freedom of the approximate chi-squared distribution of the test statistic, NA if the p-value is computed by Monte Carlo simulation.

p.value: the p-value for the test.

method: a character string indicating the type of test performed, and whether Monte Carlo simulation or continuity correction was used.

data.name: a character string giving the name(s) of the data.

observed: the observed counts.

expected: the expected counts under the null hypothesis.

residuals: the Pearson residuals, '(observed - expected) / sqrt(expected)'.

stdres: standardized residuals, `(observed - expected) / sqrt(V)`, where V is the residual cell variance (Agresti, 2007, section 2.4.5 for the case where x is a matrix, 'n * p * (1 - p)' otherwise).

Source

The code for Monte Carlo simulation is a C translation of the Fortran algorithm of Patefield (1981).

References

Hope, A. C. A. (1968) A simplified Monte Carlo significance test procedure. _J. Roy, Statist. Soc. B_ *30*, 582-598.

Patefield, W. M. (1981) Algorithm AS159. An efficient method of generating r x c tables with given row and column totals. _Applied Statistics_ *30*, 91-97.

Agresti, A. (2007) _An Introduction to Categorical Data Analysis, 2nd ed._, New York: John Wiley & Sons. Page 38.

See Also

For goodness-of-fit testing, notably of continuous distributions, ks.test.

Examples

```
## Not run:
## From Agresti(2007) p.39
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(gender = c("F", "M"),
                    party = c("Democrat","Independent", "Republican"))
(Xsq <- chisq.test(M))  # Prints test summary
Xsq$observed   # observed counts (same as M)
Xsq$expected   # expected counts under the null
Xsq$residuals  # Pearson residuals
Xsq$stdres     # standardized residuals


## Effect of simulating p-values
x <- matrix(c(12, 5, 7, 7), ncol = 2)
chisq.test(x)$p.value           # 0.4233
chisq.test(x, simulate.p.value = TRUE, B = 10000)$p.value
                                # around 0.29!
```

```
## Testing for population probabilities
## Case A. Tabulated data
x <- c(A = 20, B = 15, C = 25)
chisq.test(x)
chisq.test(as.table(x))             # the same
x <- c(89,37,30,28,2)
p <- c(40,20,20,15,5)
try(
chisq.test(x, p = p)                # gives an error
)
chisq.test(x, p = p, rescale.p = TRUE)
                                    # works
p <- c(0.40,0.20,0.20,0.19,0.01)
                                    # Expected count in category 5
                                    # is 1.86 < 5 ==> chi square approx.
chisq.test(x, p = p)                #               maybe doubtful, but is ok!
chisq.test(x, p = p, simulate.p.value = TRUE)

## Case B. Raw data
x <- trunc(5 * runif(100))
chisq.test(table(x))            # NOT 'chisq.test(x)'!

###

## End(Not run)
```

`datatable` *Create an HTML table widget using the DataTables library*

Description

This function creates an HTML widget to display rectangular data (a matrix or data frame) using the JavaScript library DataTables, with a method for `desctable` objects.

Usage

```
datatable(data, ...)

## Default S3 method:
datatable(data, options = list(), class = "display",
  callback = DT::JS("return table;"), caption = NULL, filter = c("none",
  "bottom", "top"), escape = TRUE, style = "default", width = NULL,
  height = NULL, elementId = NULL,
  fillContainer = getOption("DT.fillContainer", NULL),
  autoHideNavigation = getOption("DT.autoHideNavigation", NULL),
  selection = c("multiple", "single", "none"), extensions = list(),
  plugins = NULL, ...)
```

```
## S3 method for class 'desctable'
datatable(data, options = list(paging = F, info = F,
  search = F, dom = "Brtip", fixedColumns = T, fixedHeader = T, buttons =
  c("copy", "excel")), class = "display",
  callback = DT::JS("return table;"), caption = NULL, filter = c("none",
  "bottom", "top"), escape = FALSE, style = "default", width = NULL,
  height = NULL, elementId = NULL,
  fillContainer = getOption("DT.fillContainer", NULL),
  autoHideNavigation = getOption("DT.autoHideNavigation", NULL),
  selection = c("multiple", "single", "none"), extensions = c("FixedHeader",
  "FixedColumns", "Buttons"), plugins = NULL, rownames = F, digits = 2,
  ...)
```

Arguments

data	a data object (either a matrix or a data frame)
...	arguments passed to format.
options	a list of initialization options (see http://datatables.net/reference/option/); the character options wrapped in JS() will be treated as literal JavaScript code instead of normal character strings; you can also set options globally via options(DT.options = list(. and global options will be merged into this options argument if set
class	the CSS class(es) of the table; see http://datatables.net/manual/styling/classes
callback	the body of a JavaScript callback function with the argument table to be applied to the DataTables instance (i.e. table)
caption	the table caption; a character vector or a tag object generated from htmltools::tags$caption()
filter	whether/where to use column filters; none: no filters; bottom/top: put column filters at the bottom/top of the table; range sliders are used to filter numeric/date/time columns, select lists are used for factor columns, and text input boxes are used for character columns; if you want more control over the styles of filters, you can provide a list to this argument of the form list(position = 'top', clear = TRUE, pla = FALSE), where clear indicates whether you want the clear buttons in the input boxes, and plain means if you want to use Bootstrap form styles or plain text input styles for the text input boxes
escape	whether to escape HTML entities in the table: TRUE means to escape the whole table, and FALSE means not to escape it; alternatively, you can specify numeric column indices or column names to indicate which columns to escape, e.g. 1:5 (the first 5 columns), c(1, 3, 4), or c(-1, -3) (all columns except the first and third), or c('Species', 'Sepal.Length')
style	the style name (http://datatables.net/manual/styling/); currently only 'default' and 'bootstrap' are supported
width	Width/Height in pixels (optional, defaults to automatic sizing)
height	Width/Height in pixels (optional, defaults to automatic sizing)
elementId	An id for the widget (a random string by default).

`fillContainer` TRUE to configure the table to automatically fill it's containing element. If the table can't fit fully into it's container then vertical and/or horizontal scrolling of the table cells will occur.

`autoHideNavigation` TRUE to automatically hide navigational UI when the number of total records is less than the page size.

`selection` the row/column selection mode (single or multiple selection or disable selection) when a table widget is rendered in a Shiny app; alternatively, you can use a list of the form `list(mode = 'multiple',selected = c(1, 3, 8), target = 'row')` to pre-select rows; the element `target` in the list can be `'column'` to enable column selection, or `'row+column'` to make it possible to select both rows and columns (click on the footer to select columns), or `'cell'` to select cells

`extensions` a character vector of the names of the DataTables extensions (https://datatables.net/extensions/index)

`plugins` a character vector of the names of DataTables plug-ins (https://rstudio.github.io/DT/plugins.html)

`rownames` TRUE (show row names) or FALSE (hide row names) or a character vector of row names; by default, the row names are displayed in the first column of the table if exist (not NULL)

`digits` the desired number of digits after the decimal point (`format = "f"`) or *significant* digits (`format = "g"`, `= "e"` or `= "fg"`).

Default: 2 for integer, 4 for real numbers. If less than 0, the C default of 6 digits is used. If specified as more than 50, 50 will be used with a warning unless `format = "f"` where it is limited to typically 324. (Not more than 15–21 digits need be accurate, depending on the OS and compiler used. This limit is just a precaution against segfaults in the underlying C runtime.)

Note

You are recommended to escape the table content for security reasons (e.g. XSS attacks) when using this function in Shiny or any other dynamic web applications.

References

See http://rstudio.github.io/DT for the full documentation.

Examples

```
library(DT)

# see the package vignette for examples and the link to website
vignette('DT', package = 'DT')

# some boring edge cases for testing purposes
m = matrix(nrow = 0, ncol = 5, dimnames = list(NULL, letters[1:5]))
datatable(m)  # zero rows
datatable(as.data.frame(m))
```

```
m = matrix(1, dimnames = list(NULL, 'a'))
datatable(m)  # one row and one column
datatable(as.data.frame(m))

m = data.frame(a = 1, b = 2, c = 3)
datatable(m)
datatable(as.matrix(m))

# dates
datatable(data.frame(
  date = seq(as.Date("2015-01-01"), by = "day", length.out = 5), x = 1:5
))
datatable(data.frame(x = Sys.Date()))
datatable(data.frame(x = Sys.time()))

###
```

`desctable` *Generate a statistics table*

Description

Generate a statistics table with the chosen statistical functions, and tests if given a `"grouped"` dataframe.

Usage

```
desctable(data, stats, tests, labels)

## Default S3 method:
desctable(data, stats = stats_auto, tests, labels = NULL)

## S3 method for class 'grouped_df'
desctable(data, stats = stats_auto, tests = tests_auto,
  labels = NULL)
```

Arguments

`data`	The dataframe to analyze
`stats`	A list of named statistics to apply to each element of the dataframe, or a function returning a list of named statistics
`tests`	A list of statistical tests to use when calling desctable with a grouped_df
`labels`	A named character vector of labels to use instead of variable names

Value

A desctable object, which prints to a table of statistics for all variables

Labels

labels is an option named character vector used to make the table prettier.

If given, the variable names for which there is a label will be replaced by their corresponding label.

Not all variables need to have a label, and labels for non-existing variables are ignored.

labels must be given in the form c(unquoted_variable_name = "label")

Stats

The stats can be a function which takes a dataframe and returns a list of statistical functions to use.

stats can also be a named list of statistical functions, or formulas.

The names will be used as column names in the resulting table. If an element of the list is a function, it will be used as-is for the stats. If an element of the list is a formula, it can be used to conditionally use stats depending on the variable.

The general form is `condition ~ T | F`, and can be nested, such as `is.factor ~ percent | (is.normal ~ mean | media` for example.

Tests

The tests can be a function which takes a variable and a grouping variable, and returns an appropriate statistical test to use in that case.

tests can also be a named list of statistical test functions, associating the name of a variable in the data, and a test to use specifically for that variable.

That test name must be expressed as a single-term formula (e.g. `~t.test`). You don't have to specify tests for all the variables: a default test for all other variables can be defined with the name `.default`, and an automatic test can be defined with the name `.auto`.

If data is a grouped dataframe (using `group_by`), subtables are created and statistic tests are performed over each sub-group.

Output

The output is a desctable object, which is a list of named dataframes that can be further manipulated. Methods for printing, using in **pander** and **DT** are present. Printing reduces the object to a dataframe.

See Also

`stats_auto`

`tests_auto`

`print.desctable`

`pander.desctable`

`datatable.desctable`

Examples

```
iris %>%
  desctable

# Does the same as stats_auto here
iris %>%
  desctable(stats = list("N"      = length,
                         "%/Mean" = is.factor ~ percent | (is.normal ~ mean),
                         "sd"     = is.normal ~ sd,
                         "Med"    = is.normal ~ NA | median,
                         "IQR"    = is.normal ~ NA | IQR))

# With labels
mtcars %>% desctable(labels = c(hp  = "Horse Power",
                                cyl = "Cylinders",
                                mpg = "Miles per gallon"))

# With grouping on a factor
iris %>%
  group_by(Species) %>%
  desctable(stats = stats_default)

# With nested grouping, on arbitrary variables
mtcars %>%
  group_by(vs, cyl) %>%
  desctable

# With grouping on a condition, and choice of tests
iris %>%
  group_by(Petal.Length > 5) %>%
  desctable(tests = list(.auto = tests_auto, Species = ~chisq.test))
```

`fisher.test` *Fisher's Exact Test for Count Data*

Description

Performs Fisher's exact test for testing the null of independence of rows and columns in a contingency table with fixed marginals, or with a formula expression.

Usage

```
fisher.test(x, y, workspace, hybrid, control, or, alternative, conf.int,
  conf.level, simulate.p.value, B)

## Default S3 method:
fisher.test(x, ...)

## S3 method for class 'formula'
```

```
fisher.test(x, y = NULL, workspace = 2e+05, hybrid = F,
  control = list(), or = 1, alternative = "two.sided", conf.int = T,
  conf.level = 0.95, simulate.p.value = F, B = 2000)
```

Arguments

x	either a two-dimensional contingency table in matrix form, a factor object, or a formula of the form `lhs ~ rhs` where `lhs` and `rhs` are factors.
y	a factor object; ignored if x is a matrix or a formula.
workspace	an integer specifying the size of the workspace used in the network algorithm. In units of 4 bytes. Only used for non-simulated p-values larger than 2×2 tables. Since R version 3.5.0, this also increases the internal stack size which allows larger problems to be solved, however sometimes needing hours. In such cases, `simulate.p.values=TRUE` may be more reasonable.
hybrid	a logical. Only used for larger than 2×2 tables, in which cases it indicates whether the exact probabilities (default) or a hybrid approximation thereof should be computed.
control	a list with named components for low level algorithm control. At present the only one used is `"mult"`, a positive integer ≥ 2 with default 30 used only for larger than 2×2 tables. This says how many times as much space should be allocated to paths as to keys: see file '`fexact.c`' in the sources of this package.
or	the hypothesized odds ratio. Only used in the 2×2 case.
alternative	indicates the alternative hypothesis and must be one of `"two.sided"`, `"greater"` or `"less"`. You can specify just the initial letter. Only used in the 2×2 case.
conf.int	logical indicating if a confidence interval for the odds ratio in a 2×2 table should be computed (and returned).
conf.level	confidence level for the returned confidence interval. Only used in the 2×2 case and if `conf.int = TRUE`.
simulate.p.value	a logical indicating whether to compute p-values by Monte Carlo simulation, in larger than 2×2 tables.
B	an integer specifying the number of replicates used in the Monte Carlo test.
...	additional params to feed to original fisher.test

Details

If x is a matrix, it is taken as a two-dimensional contingency table, and hence its entries should be nonnegative integers. Otherwise, both x and y must be vectors of the same length. Incomplete cases are removed, the vectors are coerced into factor objects, and the contingency table is computed from these.

For 2 by 2 cases, p-values are obtained directly using the (central or non-central) hypergeometric distribution. Otherwise, computations are based on a C version of the FORTRAN subroutine FEXACT which implements the network developed by Mehta and Patel (1986) and improved by Clarkson, Fan and Joe (1993). The FORTRAN code can be obtained from http://www.netlib.org/toms/643. Note this fails (with an error message) when the entries of the table are too large.

(It transposes the table if necessary so it has no more rows than columns. One constraint is that the product of the row marginals be less than 2^31 - 1.)

For 2 by 2 tables, the null of conditional independence is equivalent to the hypothesis that the odds ratio equals one. `Exact` inference can be based on observing that in general, given all marginal totals fixed, the first element of the contingency table has a non-central hypergeometric distribution with non-centrality parameter given by the odds ratio (Fisher, 1935). The alternative for a one-sided test is based on the odds ratio, so `alternative = "greater"` is a test of the odds ratio being bigger than `or`.

Two-sided tests are based on the probabilities of the tables, and take as `more extreme` all tables with probabilities less than or equal to that of the observed table, the p-value being the sum of such probabilities.

For larger than 2 by 2 tables and `hybrid = TRUE`, asymptotic chi-squared probabilities are only used if the 'Cochran conditions' are satisfied, that is if no cell has count zero, and more than 80 exact calculation is used.

Simulation is done conditional on the row and column marginals, and works only if the marginals are strictly positive. (A C translation of the algorithm of Patefield (1981) is used.)

Value

A list with class `"htest"` containing the following components:

p.value: the p-value of the test.

conf.int: a confidence interval for the odds ratio. Only present in the 2 by 2 case and if argument `conf.int = TRUE`.

estimate: an estimate of the odds ratio. Note that the _conditional_ Maximum Likelihood Estimate (MLE) rather than the unconditional MLE (the sample odds ratio) is used. Only present in the 2 by 2 case.

null.value: the odds ratio under the null, `or`. Only present in the 2 by 2 case.

alternative: a character string describing the alternative hypothesis.

method: the character string `"Fisher's Exact Test for Count Data"`.

data.name: a character string giving the names of the data.

References

Agresti, A. (1990) _Categorical data analysis_. New York: Wiley. Pages 59-66.

Agresti, A. (2002) _Categorical data analysis_. Second edition. New York: Wiley. Pages 91-101.

Fisher, R. A. (1935) The logic of inductive inference. _Journal of the Royal Statistical Society Series A_ *98*, 39-54.

Fisher, R. A. (1962) Confidence limits for a cross-product ratio. _Australian Journal of Statistics_ *4*, 41.

Fisher, R. A. (1970) _Statistical Methods for Research Workers._ Oliver & Boyd.

Mehta, C. R. and Patel, N. R. (1986) Algorithm 643. FEXACT: A Fortran subroutine for Fisher's exact test on unordered r*c contingency tables. _ACM Transactions on Mathematical Software_, *12*, 154-161.

Clarkson, D. B., Fan, Y. and Joe, H. (1993) A Remark on Algorithm 643: FEXACT: An Algorithm for Performing Fisher's Exact Test in r x c Contingency Tables. _ACM Transactions on Mathematical Software_, *19*, 484-488.

Patefield, W. M. (1981) Algorithm AS159. An efficient method of generating r x c tables with given row and column totals. _Applied Statistics_ *30*, 91-97.

See Also

chisq.test

`fisher.exact` in package **kexact2x2** for alternative interpretations of two-sided tests and confidence intervals for 2 by 2 tables.

Examples

```
## Not run:
## Agresti (1990, p. 61f; 2002, p. 91) Fisher's Tea Drinker
## A British woman claimed to be able to distinguish whether milk or
##  tea was added to the cup first.  To test, she was given 8 cups of
##  tea, in four of which milk was added first.  The null hypothesis
##  is that there is no association between the true order of pouring
##  and the woman's guess, the alternative that there is a positive
##  association (that the odds ratio is greater than 1).
TeaTasting <-
matrix(c(3, 1, 1, 3),
       nrow = 2,
       dimnames = list(Guess = c("Milk", "Tea"),
                       Truth = c("Milk", "Tea")))
fisher.test(TeaTasting, alternative = "greater")
## => p = 0.2429, association could not be established

## Fisher (1962, 1970), Criminal convictions of like-sex twins
Convictions <-
matrix(c(2, 10, 15, 3),
       nrow = 2,
       dimnames =
       list(c("Dizygotic", "Monozygotic"),
            c("Convicted", "Not convicted")))
Convictions
fisher.test(Convictions, alternative = "less")
fisher.test(Convictions, conf.int = FALSE)
fisher.test(Convictions, conf.level = 0.95)$conf.int
fisher.test(Convictions, conf.level = 0.99)$conf.int

## A r x c table  Agresti (2002, p. 57) Job Satisfaction
Job <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4,
dimnames = list(income = c("< 15k", "15-25k", "25-40k", "> 40k"),
                satisfaction = c("VeryD", "LittleD", "ModerateS", "VeryS")))
fisher.test(Job)
fisher.test(Job, simulate.p.value = TRUE, B = 1e5)

###
```

```
## End(Not run)
```

`flatten_desctable` *Flatten a desctable to a dataframe recursively*

Description

Flatten a desctable to a dataframe recursively

Usage

```
flatten_desctable(desctable)
```

Arguments

`desctable`	A desctable object

Value

A flat dataframe

`header` *Build header*

Description

Take a desctable object and create a suitable header for the mentionned output. Output can be one of "pander", "datatable", or "dataframe".

Usage

```
header(desctable, output = c("pander", "datatable", "dataframe"))
```

Arguments

`desctable`	A desctable object
`output`	An output format for the header

Value

A header object in the output format

headerList *Build a header list object*

Description

Build a header list object

Usage

```
headerList(desctable)
```

Arguments

desctable	A desctable

Value

A nested list of headers with colspans

head_dataframe *Build the header for dataframe*

Description

Build the header for dataframe

Usage

```
head_dataframe(head)
```

Arguments

head	A headerList object

Value

A names vector

head_datatable *Build the header for datatable*

Description

Build the header for datatable

Usage

```
head_datatable(head)
```

Arguments

head	A headerList object

Value

An htmltools$tags object containing the header

head_pander *Build the header for pander*

Description

Build the header for pander

Usage

```
head_pander(head)
```

Arguments

head	A headerList object

Value

A names vector

insert *Insert a vector y inside another vector x at position*

Description

Insert a vector y inside another vector x at position

Usage

```
insert(x, y, position)
```

Arguments

x	A vector
y	A vector or list of vectors
position	The position / vector of positions to insert vector(s) y in vector x

Value

The combined vector

IQR *Return the inter-quartile range*

Description

Safe version of IQR for statify

Usage

```
IQR(x)
```

Arguments

x	A vector

Value

The IQR

is.normal *Test if distribution is normal*

Description

Test if distribution is normal. The condition for normality is length > 30 and non-significant Shapiro-Wilks test with p > .1

Usage

```
is.normal(x)
```

Arguments

x	A numerical vector

Value

A boolean

no.test *No test*

Description

An empty test

Usage

```
no.test(formula)
```

Arguments

formula	A formula

`pander.desctable` *Pander method for desctable*

Description

Pander method to output a desctable

Usage

```
pander.desctable(x = NULL, digits = 2, justify = "left", missing = "",
  keep.line.breaks = T, split.tables = Inf, emphasize.rownames = F, ...)
```

Arguments

`x`	A desctable
`digits`	passed to `format`. Can be a vector specifying values for each column (has to be the same length as number of columns).
`justify`	defines alignment in cells passed to `format`. Can be `left`, `right` or `centre`, which latter can be also spelled as `center`. Defaults to `centre`. Can be abbreviated to a string consisting of the letters `l`, `c` and `r` (e.g. 'lcr' instead of c('left', 'centre', 'right').
`missing`	string to replace missing values
`keep.line.breaks`	(default: `FALSE`) if to keep or remove line breaks from cells in a table
`split.tables`	where to split wide tables to separate tables. The default value (`80`) suggests the conventional number of characters used in a line, feel free to change (e.g. to `Inf` to disable this feature) if you are not using a VT100 terminal any more :)
`emphasize.rownames`	boolean (default: `TRUE`) if row names should be highlighted
`...`	unsupported extra arguments directly placed into `/dev/null`

Details

Uses `pandoc.table`, with some default parameters (`digits = 2`, `justify = "left"`, `missing = ""`, `keep.line.breaks = T`, `split.tables = Inf`, and `emphasize.rownames = F`), that you can override if needed.

See Also

`pandoc.table`

parse_formula *Parse a formula*

Description

Parse a formula defining the conditions to pick a stat/test

Usage

```
parse_formula(x, f)
```

Arguments

x	The variable to test it on
f	A formula to parse

Details

Parse a formula defining the conditions to pick a stat/test and return the function to use. The formula is to be given in the form of conditional ~ T | F and conditions can be nested such as conditional1 ~ (conditional2 ~ T | F) | F The FALSE option can be omitted, and the TRUE can be replaced with NA

Value

A function to use as a stat/test

percent *Return the percentages for the levels of a factor*

Description

Return a compatible vector of length nlevels(x) + 1 to print the percentages of each level of a factor

Usage

```
percent(x)
```

Arguments

x	A factor

Value

A nlevels(x) + 1 length vector of percentages

print.desctable *Print method for desctable*

Description

Print method for desctable

Usage

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

Arguments

x	A desctable
...	Additional print parameters

Value

A flat dataframe

statColumn *Generate one statistic for all variables*

Description

Generate one statistic for all variables

Usage

```
statColumn(stat, data)
```

Arguments

stat	The statistic to use
data	The dataframe to apply the statistic to

Value

A vector for one statistic column

statify *Transform any function into a valid stat function for the table*

Description

Transform a function into a valid stat function for the table

Usage

```
statify(x, f)

## Default S3 method:
statify(x, f)

## S3 method for class 'formula'
statify(x, f)
```

Arguments

x	A vector
f	The function to try to apply, or a formula combining two functions

Details

NA values are removed from the data

Applying the function on a numerical vector should return one value

Applying the function on a factor should return nlevels + 1 value, or one value per factor level

See `parse_formula` for the usage for formulaes.

Value

The results for the function applied on the vector, compatible with the format of the result table

stats_default *Functions to create a list of statistics to use in desctable*

Description

These functions take a dataframe as argument and return a list of statistcs in the form accepted by desctable.

Usage

```
stats_default(data)

stats_normal(data)

stats_nonnormal(data)

stats_auto(data)
```

Arguments

data	The dataframe to apply the statistic to

Details

Already defined are

1. stats_default with length, %, mean, sd, med and IQR
2. stats_normal with length, %, mean and sd
3. stats_nonnormal with length,
4. stats_auto, which picks stats depending of the data

You can define your own automatic functions, as long as they take a dataframe as argument and return a list of functions or formulas defining conditions to use a stat function.

Value

A list of statistics to use, potentially assessed from the dataframe

---

statTable *Generate the table of all statistics for all variables*

Description

Generate the table of all statistics for all variables

Usage

```
statTable(data, stats)
```

Arguments

data	The dataframe to apply the statistic to
stats	A list of named statistics to use

Value

A dataframe of all statistics for all variables

subNames *Create the subtables names*

Description

Create the subtables names, as factor: level (n=sub-group length)

Usage

```
subNames(grp, df)
```

Arguments

grp	Grouping factor
df	Dataframe containing the grouping factor

Value

A character vector with the names for the subtables

subTable *Create a subtable in a grouped desctable*

Description

Create a subtable in a grouped desctable

Usage

```
subTable(df, stats, tests, grps)
```

Arguments

df	Dataframe to use
stats	Stats list/function to use
tests	Tests list/function to use
grps	List of symbols for grouping factors

Value

A nested list of statTables and testColumns

testColumn *Create the pvalues column*

Description

Create the pvalues column

Usage

```
testColumn(df, tests, grp)
```

Arguments

df	Dataframe to use for the tests
tests	Test function or list of functions
grp	Grouping factor

Value

A numeric vector of pvalues

testify *Transform any test function into a valid test function for the table*

Description

Transform a function into a valid test function for the table Applying the function on a numerical vector should return one value Applying the function on a factor should return nlevels + 1 value, or one value per factor level

Usage

```
testify(x, f, group)
```

Arguments

x	A vector
f	The function to try to apply, or a formula combining two functions
group	Grouping factor

Value

The results for the function applied on the vector, compatible with the format of the result table

tests_auto *Functions to choose a statistical test*

Description

These functions take a variable and a grouping variable as arguments, and return a statistcal test to use, expressed as a single-term formula.

Usage

```
tests_auto(var, grp)
```

Arguments

var	The variable to test
grp	The variable for the groups

Details

Currently, only `tests_auto` is defined, and picks between t test, wilcoxon, anova, kruskal-wallis and fisher depending on the number of groups, the type of the variable, the normality and homoskedasticity of the distributions.

Value

A statistical test function

varColumn *Generate the variable column to display as row names*

Description

Generates the variable column. Replaces the variable names by their label if given in the named character vector labels, and inserts levels for factors.

Usage

```
varColumn(data, labels = NULL)
```

Arguments

data	The dataframe to get the names from
labels	The optional named character vector containing the keypairs var = "Label"

Details

labels is an option named character vector used to make the table prettier. If given, the variable names for which there is a label will be replaced by their corresponding label. Not all variables need to have a label, and labels for non-existing variables are ignored.

Value

A dataframe with one variable named "Variables", a character vector of variable names/labels and levels

Index