Package 'atable'

September 17, 2023

Type Package

Title Create Tables for Reporting Clinical Trials

Version 0.1.14

Author Armin Ströbel [aut, cre] (<https://orcid.org/0000-0002-6873-5332>), Alan Haynes [aut] (<https://orcid.org/0000-0003-1374-081X>)

Maintainer Armin Ströbel <arminstroebel@web.de>

Description Create Tables for Reporting Clinical Trials.
Calculates descriptive statistics and hypothesis tests,
arranges the results in a table ready for reporting with LaTeX, HTML or Word.

License GPL-3

Depends R (>= 3.5)

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

Imports stats (>= 3.4), doBy (>= 4.6), plyr (>= 1.8.4), reshape2 (>= 1.4.3), Hmisc (>= 4.1), settings (>= 0.2.4), DescTools (>= 0.99.24), effsize (>= 0.7.1)

Suggests testthat, knitr, survival, rmarkdown

URL https://github.com/arminstroebel/atable

BugReports https://github.com/arminstroebel/atable/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-17 10:20:02 UTC

R topics documented:

`add_name_to_statistics` *add_name_to_statistics*

Description

add a column to a data.frame x with value name as character. Helper Function. Not intended to be called by the user.

Usage

```
add_name_to_statistics(x, name, ...)

## S3 method for class 'list'
add_name_to_statistics(x, name, ...)

## S3 method for class 'data.frame'
add_name_to_statistics(
  x,
```

```
  name,
  colname_for_variable = atable_options("colname_for_variable"),
  ...
)
```

Arguments

x	an object
name	a value
...	passed to methods
colname_for_variable	a character length 1. Default is defined in atable_options

Details

checks if the new field already exists

Value

x now with new field colname_for_variable

Methods (by class)

- `add_name_to_statistics(list)`: apply add_name_to_statistics to all field of the list
- `add_name_to_statistics(data.frame)`: add field colname_for_variable to the data.frame. chekc for a name clash as this field as there are many user-defined fields

---

add_name_to_tests *add_name_to_tests*

Description

Helper-function to add a field to a list or data.frame

Usage

```
add_name_to_tests(x, name, ...)

## S3 method for class 'list'
add_name_to_tests(x, name, ...)

## S3 method for class 'data.frame'
add_name_to_tests(
  x,
  name,
  colname_for_variable = atable_options("colname_for_variable"),
  ...
)
```

Arguments

x	an object
name	a value
...	passed to methods
colname_for_variable	a character length 1. Default is defined in atable_options

Details

Not intended to be called by the user.

checks if the new field already exists

Value

x now with new field colname_for_variable

Methods (by class)

- `add_name_to_tests(list)`: apply add_name_to_statistics to all field of the list
- `add_name_to_tests(data.frame)`: add field colname_for_variable to the data.frame. chekc for a name clash as this field as there are many user-defined fields

---

add_observation_column

Adds a column to a data.frame

Description

The new column has name `atable_options('colname_for_observations')` and class `'count_me'`.

Usage

```
add_observation_column(DD)
```

Arguments

DD	A data.frame.

Details

Throws an error if a column of that name is already present in `DD`.

Value

As DD now with one more column.

atable *Create Tables for Reporting of Clinical Trials*

Description

Applies descriptive statistics and hypothesis tests to data, and arranges the results for printing.

Usage

```
atable(x, ...)

## S3 method for class 'data.frame'
atable(
  x,
  target_cols,
  group_col = NULL,
  split_cols = NULL,
  format_to = atable_options("format_to"),
  drop_levels = TRUE,
  add_levels_for_NA = FALSE,
  blocks = NULL,
  add_margins = atable_options("add_margins"),
  indent_character = NULL,
  indent = atable_options("indent"),
  ...
)

## S3 method for class 'formula'
atable(formula, data, ...)
```

Arguments

x	An object. If x is a data.frame, it must have unique and syntactically valid colnames, see is_syntactically_valid_name. If x is a formula, then its format must be target_cols ~ group_col | split_cols. See other arguments for more details.
...	Passed from and to other methods. You can use the ellipsis ... to modify atable: For example the default-statistics for numeric variables are mean and sd. To change these statistics pass a function to argument statistics.numeric, that calculates the statistics you prefer for your data. See examples below how to modify atable by ... . Actually statistics.numeric is passed to statistics and thus documented there, but for convenience it also documented here. Here is a list of the statistics and hypothesis tests that can be modified by ... : • statistics.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::statistics.numeric when atable is called. The function must mimic statistics: see the help there.

- statistics.factor: Analog to argument statistics.numeric.
- statistics.ordered: Analog to argument statistics.numeric.
- two_sample_htest.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::two_sample_htest.numeric when atable is called. The function must mimic two_sample_htest: see the help there.
- two_sample_htest.factor: Analog to argument two_sample_htest.numeric
- two_sample_htest.ordered: Analog to argument two_sample_htest.numeric
- multi_sample_htest.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::multi_sample_htest.numeric when atable is called. The function must mimic multi_sample_htest: see the help there.
- multi_sample_htest.factor: Analog to argument multi_sample_htest.numeric
- multi_sample_htest.ordered: Analog to argument multi_sample_htest.numeric
- format_statistics.statistics_numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::format_statistics.statistics_nume The function must mimic format_statistics: see the help there.
- format_statistics.statistics_factor: Analog to argument format_statistics.statistics_numeric
- format_tests.htest: Either NULL or a function. Default is NULL. If a function, then it will replace format_tests.htest. The function must mimic format_tests: see the help there.
- format_tests.htest_with_effect_size: Analog to argument format_tests.htest

target_cols
: A character vector containing some column names of x.
Descriptive statistics and hypothesis test are applied to these columns depending on their class. The descriptive statistics are defined by statistics; their representation and format by format_statistics.
Hypothesis test are defined by two_sample_htest or multi_sample_htest (depending on the number of levels of group_col); their representation and format by format_tests. Note that atable always adds one name to target_cols to count the number of obsservations. This name is stored in atable_options('colname_for_observat

group_col
: A character of length 1 containing a column of x or NULL. This column defines the groups that are compared by the hypothesis tests. as.factor is applied to this column before further processing. Default is NULL, meaning that no hypothesis tests are applied.

split_cols
: A character vector containing some of colnames(x) or NULL. x is splitted by these columns before descriptive statistics and hypothesis test are applied. as.factor is applied to this column before further processing. Default is NULL, meaning that no splitting is done.

format_to
: A character vector of length 1. Specifies the format of the output of atable. Possible values are 'Latex', 'Word', 'Raw', 'HTML', 'Console', 'markdown', 'md'. Default is defined in atable_options.

drop_levels
: A logical. If TRUE then droplevels is called on group_col and split_cols before further processing. Default is TRUE.

add_levels_for_NA
: If TRUE then addNA is called on group_col and split_cols before further processing. Default is FALSE.

`blocks` NULL or a list. If blocks is a list, then the names of the list must be non-NA characters. The elements of the list must be some of `target_cols`, retaining the order of `target_cols`. Also in this case `split_cols` must be NULL as simultaneous blocking and splitting is not supported. Default is NULL, meaning that no blocking is done. Variables of a block are additionally indented. Blocking has no effect on the statistics, it only affects the indentation of the resulting table. See Examples.

`add_margins` A logical with length one, TRUE or FALSE. Default is defined in `atable_options` as FALSE. When `add_margins` is TRUE and `group_col` is not NULL, a column containing the results of an ungrouped `atable`-call is added to the results. See Examples.

`indent_character` A character with length 1 or NULL (default). This character is used for indentation in the resulting table. If NULL, then the value stored in `atable_options` is taken instead, depending on `format_to`. `indent_data_frame` does the indentation. See help there.

`indent` A logical with length one, TRUE or FALSE. Default is defined in `atable_options`. Decides if indentation is done or not. The resulting table will have a different layout. If FALSE, then `blocks` is ignored.

`formula` A formula of the form `target_cols ~ group_col | split_cols`. The `|` separates the `group_col` from the `split_cols`. Read the `|` as 'given' as in a conditional probability `P(target_cols | split_cols)`. `target_cols` and `split_cols` may contain multiple names separated by `+`. `group_col` must be a single name if given. `group_col` and `split_cols` may be omitted and can be replaced by `1` in this case. The `|` may also be omitted if no `split_cols` are given.

`data` Passed to `atable(x = data, ...)`.

Value

Results depend on `format_to`:

- `'Raw'`: A list with two elemtents called `'statistics_result'` and `'tests_result'`, that contain all results of the descriptve statistics and the hypothesis tests. This format useful, when extracting a specific result unformated (when `format_to` is not `'Raw'` all numbers are also returned, but as rounded characters for printing and squeezed into a data.frame).
  - `'statistics_result'`: contains a data.frame with colnames `c(split_cols, group_col, target_cols`. `split_cols` and `group_col` retain their original values (now as factor). `target_cols` contain lists with the results of function `statistics`. As the result of function `statistics` is also a list, `target_cols` contain lists of lists.
  - `'tests_result'`: has the same structure as `'statistics_result'`, but contains the results of `two_sample_htest` and `multi_sample_htest`. Note that `tests_result` only exists if `split_cols` is not NULL.
- `'Word'`: A data.frame. Column `atable_options('colname_for_group')` contains all combinations of the levels of `split_cols` and the names of the results of function `format_statistics`.

  Further columns are the levels of `group_col` the names of the results of `format_tests`.

  The levels of `split_cols` and the statistics are arranged vertically. The hypothesis test are arranged horizontally.

- 'HTML': Same as for format_to = 'Word' but a different character indents the first column. #'
- 'Console': Meant for printing in the R console for interactive analysis. Same as for format_to = 'Word' but a different character indents the first column.
- 'Latex': Same as for format_to = 'Word' but a different character indents the first column and with translate_to_LaTeX applied afterwards.

Methods (by class)

- atable(data.frame): applies descriptive statistics and hypothesis tests, arranges the results for printing.
- atable(formula): parses the formula and passes its parts to atable.

Examples

```
# See vignette for more examples:
# utils::vignette('atable_usage', package = 'atable')

# Analyse datasets::ToothGrowth:
# Length of tooth for each dose level and delivery method:
atable::atable(datasets::ToothGrowth,
  target_cols = 'len',
  group_col = 'supp',
  split_cols = 'dose',
  format_to = 'Word')
# Print in .docx with e.g. flextable::regulartable and officer::body_add_table

# Analyse datasets::ChickWeight:
# Weight of chickens for each time point and diet:
atable(weight ~ Diet | Time, datasets::ChickWeight, format_to = 'Latex')
# Print as .pdf with e.g. Hmisc::latex

# Analyse atable::test_data:
atable(Numeric + Logical + Factor + Ordered ~ Group | Split1 + Split2,
  atable::test_data, format_to = 'HTML')
# Print as .html with e.g. knitr::kable and options(knitr.kable.NA = '')

# Modify atable: calculate median and MAD for numeric variables
new_stats  <- function(x, ...){list(Median = median(x, na.rm = TRUE),
                                    MAD = mad(x, na.rm = TRUE))}
atable(atable::test_data,
       target_cols = c('Numeric', 'Numeric2'),
       statistics.numeric = new_stats,
       format_to = 'Console')
# Print in Console with format_to = 'Console'.

# Analyse mtcars and add labels and units of via package Hmisc
mtcars <- within(datasets::mtcars, {gear <- factor(gear)})
# Add labels and units.
attr(mtcars$mpg, 'alias') = 'Consumption [Miles (US)/ gallon]'
Hmisc::label(mtcars$qsec) = 'Quarter Mile Time'
```

```
units(mtcars$qsec) = 's'

# apply atable
atable::atable(mpg + hp + gear + qsec ~ cyl | vs,
               mtcars,
               format_to = 'Console')

# Blocks
# In datasets::mtcars the variables cyl, disp and mpg are related to the engine and am and gear are
# related to the gearbox. So grouping them together is desireable.
atable::atable(datasets::mtcars,
               target_cols = c("cyl", "disp", "hp", "am", "gear", "qsec") ,
               blocks = list("Engine" = c("cyl", "disp", "hp"),
                             "Gearbox" = c("am", "gear")),
               format_to = "Console")
# Note that Variable qsec is not blocked and thus not indented.



# add_margins
atable::atable(atable::test_data,
               target_cols = "Numeric",
               group_col = "Group",
               split_cols = "Split1",
               add_margins = TRUE,
               format_to = "Console")
# The column 'Total' contains the results of the ungrouped atable-call:
# The number of observations is the sum of observations of the groups.
# The default of add_margins can be changed via atable_options.
```

`atable_compact` *More compact formatting than atable()*

Description

This is a wrapper for atable(), calculating the same statistics, but with different format.

Usage

```
atable_compact(x, ...)

## S3 method for class 'data.frame'
atable_compact(
  x,
  target_cols,
  group_col = NULL,
  indent_character = atable_options("indent_character_compact"),
  blocks = NULL,
```

```
  format_factor = atable_options("format_statistics_compact.statistics_factor"),
  format_numeric = atable_options("format_statistics_compact.statistics_numeric"),
  ...
)
```

Arguments

x	object passed to atable.
...	Passed to atable.
target_cols	character. Some of colnames(x).
group_col	character or NULL. If character then, one of colnames(x).
indent_character	character length 1. Default is defined in table_options("indent_character_compact"). For Latex-Format use e.g. indent_character="\quad". For Word-Format use e.g. indent_character=paste0(rep(intToUtf8(160), 5), collapse = "") and e.g. Package officer and its functions officer::read_docx(), officer::body_add_table and print-methods.
blocks	NULL or a list, passed to atable, see help there.
format_factor	a function that defines the format of factor variables. Default is defined in atable_options. See check_format_statistics for the return-value of this function.
format_numeric	a function that defines the format of numeric variables. Analog to format_factor.

Details

The compact formatting is:

Numeric target_cols get one line in the table; the line contains the mean and SD of the variable.

Factor target_cols also get one line in the table, when they have only two levels and only the first level is displayed in the table and the name of the variable is omitted. This is intended for item like "Sex at birth: Female/Male". Knowing the percentage of Female is sufficient in this case (when NAs are not counted). Be careful with items like "Pregnant: Yes/No". Here only the level "Yes" will be printed and the name of the item (Pregnant) is omitted, making the table uninformative. Factors with three or more levels get one line per level, the levels are intended and a header line containing the name of the variable is added.

Arguments in ... are passed to atable. See the help there. atable_compact is not designed for splitted atables, so argument split_cols must be omitted or NULL. Also argument format_to is ignored. Other features of atable (blocking, add_margins, alias) are available, see examples.

Value

data.frame

Methods (by class)

- atable_compact(data.frame): a compact version of atable.

Examples

```
# For Console:
atable_compact(
  atable::test_data,
  target_cols = c("Numeric", "Numeric2", "Split2", "Factor", "Ordered"),
  group_col = "Group2",
  blocks = list("Primary Endpoint" = "Numeric",
                "Secondary Endpoints" = c("Numeric2", "Split2", "Factor")),
  add_margins = TRUE)

# The target_cols are "Numeric", "Numeric2", "Split2", "Factor", "Ordered".
# The group_col is "Group2".
# The data.frame is grouped by group_col and the summary statistcs of the target_cols are
# calculated: mean, sd for numeric, counts and percentages for factors.
# Some target_cols are blocked: the first block 'Primary Endpoint' contains the variable Numeric.
# The second block 'Secondary  Endpoint' contains the variables "Numeric2", "Split2", "Factor".
# The blocks are intended.
# For variable Split2 only the first level is reported, as the variable has only two levels and
# the name 'Split2' does not appear in the table.
# The variable Factor has more than two levels, so all of them are
# reported and appropriately intended.
# The variable Ordered is not part of a block and thus not intended.

# For Latex:
# Same as for Console, but with different indent_character:

tab = atable_compact(atable::test_data,
                  target_cols = c("Numeric", "Numeric2", "Logical", "Factor", "Ordered"),
                     group_col = "Group2",
                     indent_character = "\\quad")

tab = atable::translate_to_LaTeX(tab)

# Then call e.g. Hmisc::latex(tab, ...)


# Example for Word format:
## Not run:
tab = atable_compact(
  atable::test_data,
  target_cols = c("Numeric", "Numeric2", "Split2", "Factor", "Ordered", "Character"),
  group_col = "Group2",
  blocks = list("Primary Endpoint" = "Numeric",
                "Secondary Endpoints" = c("Numeric2", "Split2", "Factor")),
  add_margins = TRUE,
  indent_character = paste0(rep(intToUtf8(160), 5), collapse = ""))

# The argument indent_character has the value intToUtf8(160) (non breakable space).
# This is the important part:
# Spaces at the beginning of a cell of a data.frame are somehow lost on the way to the docx.
# Other indent_characters may also do the job.
```

```
# doc = officer::read_docx()
# doc = officer::body_add_table(doc,tab)

# print(doc, target = "atable_Word.docx")

# Other packages may exist for Word-export.

## End(Not run)
```

`atable_longitudinal` *A longitudinal version of atable*

Description

This is a wrapper for atable(), calculating the same statistics, but with different format.

Usage

```
atable_longitudinal(x, ...)

## S3 method for class 'data.frame'
atable_longitudinal(
  x,
  target_cols,
  split_cols,
  group_col = NULL,
 format_numeric = atable_options("format_statistics_longitudinal.statistics_numeric"),
 format_factor = atable_options("format_statistics_longitudinal.statistics_factor"),
  ...
)
```

Arguments

x	object passed to atable. Currently x must be a data.frame.
...	Passed to atable.
target_cols	character. Exactly one of colnames(x).
split_cols	character. Exactly one of colnames(x).
group_col	character or NULL. If character then, one of colnames(x).
format_numeric	a function that defines the format of numeric variables. Analog to format_factor.
format_factor	a function that defines the format of factor variables. Default is defined in atable_options. See check_format_statistics for the return-value of this function.

Details

The intention is to report longitudinal data, i.e. data measured on the same objects on multiple times points.

This function allows only one target_col and only one split_col (the time point of the measurement).

The longitudinal formatting is:

The names of the target_col and split_col do not show up in the table. The names should thus be written in the caption of the table.

Numeric target_cols get one line in the table; the format of the statistics is: mean (sd), N, missing.

Factor target_cols also get one line in the table, when it has only two levels and only the first level is displayed in the table and the name of the variable is omitted. This is intended for item like "Sex at birth: Female/Male". Knowing the percentage of Female is sufficient in this case (when NAs are not counted). The name of the target_cols and its first level should be stated in the caption of the table, otherwise the table is uninformative. The format of the statistics is: percent

Factors with three or more levels get one line per level and the name of the variable is omitted. The format of the statistics is: percent

Argument block must omitted, as there is only one target_col and nothing to block.

See examples.

Value

data.frame

Methods (by class)

- `atable_longitudinal(data.frame)`: a longitudinal version of atable.

Examples

```
# create data with a time-variable
x = atable::test_data
set.seed(42)
x = within(x, {time = sample(paste0("time_", 1:5), size=nrow(x), replace = TRUE)})
split_cols = "time"
group_col = "Group2"

# table for a factor with two levels
atable_longitudinal(x,
  target_cols = "Split2",
  group_col = group_col,
  split_cols = split_cols,
  add_margins = TRUE)


# table for a factor with three levels
atable_longitudinal(x,
  target_cols = "Split1",
  group_col = group_col,
  split_cols = split_cols,
```

```
  add_margins = TRUE)


# table for a numeric variable
atable_longitudinal(x,
  target_cols = "Numeric",
  group_col = group_col,
  split_cols = split_cols,
  add_margins = TRUE)

# To print the table in Word or with Latex, use
# e.g. \link[Hmisc]{latex} or \link[officer]{body_add_table}.
# No further modification of the table is needed.
# See \code{\link{atable_compact}} for examples.
```

`atable_options` *Set or get options*

Description

Set or get options for the atable-package via the `settings` package.

Usage

```
atable_options(...)
```

Arguments

`...`	Option names to retrieve option values or `[key]=[value]` pairs to set options.

Details

These options control some aspects of the atable package.

For restoring the default values see `atable_options_reset`.

Supported options

The following options are supported:

- `add_margins`: A logical with length 1, TRUE of FALSE. This is the default-value of atable's argument `add_margins`. See the help there.
- `colname_for_total`: A character with length 1. Default is `'Total'`. This character will show up in the results of `atable` when `add_margins` is `TRUE` and `group_col` is not `NULL`.
- `replace_NA_by`: A character with length 1, or `NULL`. Default is `'missing'`. Used in function `replace_NA`. This character will show up in the results of `atable`, so it can be modified.

- colname_for_variable: A character with length 1. Default is 'variable___'. Used in function add_name_to_tests and add_name_to_statistics. This character will not show up in the results and is only used internally for intermediate data.frames. There may be name clashes with user-supplied data.frames; so modification may be necessary.
- colname_for_observations: A character with length 1. Default is 'Observations'. Used in function add_observation_column. This character will show up in the results of atable, so it can be modified. There may be name clashes with user-supplied data.frames; so modification may be necessary.
- colname_for_blocks: A character with length 1. Default is 'block_name___'. Used in function indent_data_frame_with_blocks. This character will not show up in the results and is only used internally for intermediate data.frames. There may be name clashes with user-supplied data.frames; so modification may be necessary.
- labels_TRUE_FALSE: A character of length 2. Default is c('yes', 'no'). Currently used in function statistics.logical (see statistics) to cast logical to factor. TRUE is mapped to labels_TRUE_FALSE[1] and FALSE to labels_TRUE_FALSE[2]. This characters may show up in the results of atable, so it can be modified.
- labels_Mean_SD: A character length 1. Default is 'Mean (SD)'. Currently used in function format_statistics as a name for the mean and standard deviation of numeric variables. This character may show up in the results of atable, so it can be modified.
- labels_valid_missing: A character length 1. Default is 'valid (missing)'. Currently used in function format_statistics as a name for the number of valid and missing values of numeric variables. This character may show up in the results of atable, so it can be modified.
- format_to: A character length 1. Default is 'Latex'. Currently used in function atable.
- colname_for_group: A character of length 1. Default is 'Group'. This character will show up in the results of atable. This column will contain all values of DD[split_cols] and DD[target_cols].
- colname_for_value: A character of length 1. Default is 'value'. This character shows up in the results of atable when group_col is NULL. The column will contain the results of the statistics.
- colname_for_variable_compact: A character of length 1. Default is intToUtf8(160), a non-breaking space. This character will show up in the results of atable_compact as name of the first column.
- statistics.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::statistics.numeric when atable is called. The function must mimic statistics: see the help there.
- statistics.factor: Analog to argument statistics.numeric.
- statistics.ordered: Analog to argument statistics.numeric.
- two_sample_htest.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::two_sample_htest.numeric when atable is called. The function must mimic two_sample_htest: see the help there.
- two_sample_htest.factor: Analog to argument two_sample_htest.numeric
- two_sample_htest.ordered: Analog to argument two_sample_htest.numeric

- multi_sample_htest.numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::multi_sample_htest.numeric when atable is called. The function must mimic multi_sample_htest: see the help there.
- multi_sample_htest.factor: Analog to argument multi_sample_htest.numeric
- multi_sample_htest.ordered: Analog to argument multi_sample_htest.numeric
- format_statistics.statistics_numeric: Either NULL or a function. Default is NULL. If a function, then it will replace atable:::format_statistics.statistics_numeric. The function must mimic format_statistics: see the help there.
- format_statistics.statistics_factor: Analog to argument format_statistics.statistics_numeric
- format_tests.htest: Either NULL or a function. Default is NULL. If a function, then it will replace format_tests.htest. The function must mimic format_tests: arguments are x and the ellipsis ... . Result is a data.frame with 1 rows and unique colnames.
- format_tests.htest_with_effect_size: Analog to argument format_tests.htest
- format_p_values: A function with one argument returning a character with same length as the argument. This functions is called by format_tests to produce printable p-values.
- format_percent: A function with one argument returning a character with same length as the argument. This functions is called by format_statistics for factors to produce printable percentages.
- format_numbers: A function with one argument returning a character with same length as the argument. This functions is called by format_statistics and format_tests for number, that are not p-values or percentages.
- digits: 2. How many digits a number should have in the table. Used by format_percent and format_percent and passed to format.
- get_alias.default: A function with one argument x and ... returning a character or NULL. This functions is called by get_alias and create_alias_mapping to retrieve alternative Variable names to print in the table.
- get_alias.labelled: A function with one argument x and ..., that must return a character. This functions is called by get_alias on the columns that have class labelled.
- modifiy_colnames_without_alias: A function with one argument x and ... returning a character. This functions is called by create_alias_mapping on the columns that have is.NULL(get_alias(x)). Replaces underscores by blanks and then calls trimws.
- indent_character: A Character with length 1. Passed to indent_data_frame. Every option of format_to has a corresponding indent_character. See the help of atable for these options.
- indent_character_compact: A Character with length 1. Passed to atable_compact. Value is " " for viewing in the console. Use "\quad" for Latex and intToUtf8(160) for Word.
- indent: A logical with length 1. Passed to atable. Controls, if indent_data_frame is called.
- format_statistics_compact.statistics_factor: A function with the same Properties as format_statistics. Used as a default value for atable_compact
- format_statistics_compact.statistics_numeric : A function with the same Properties as format_statistics. Used as a default value for atable_compact
- format_statistics_longitudinal.statistics_factor: A function with the same Properties as format_statistics. Used as a default value for atable_longitudinal
- format_statistics_longitudinal.statistics_numeric : A function with the same Properties as format_statistics. Used as a default value for atable_longitudinal

Examples

```
atable_options() # show all options
atable_options('replace_NA_by' = 'no value') # set a new value
atable_options('replace_NA_by') # return the new value
```

---

atable_options_reset *Reset atable_options to default*

Description

Does as the name implies. See also atable_options.

Usage

```
atable_options_reset()
```

Examples

```
atable_options('replace_NA_by') # show options
atable_options('replace_NA_by' = 'foo bar') # set a new value
atable_options('replace_NA_by') # show options
atable_options_reset() # restore all defaults
atable_options('replace_NA_by') # as before
```

---

atable_package *atable: Create Tables for Reporting Clinical Trials*

Description

The packages provides functions for descriptive statistics and hypothesis tests, and arranging the results for printing.

Details

The main function is atable. See documentation there.

check_alias_mapping *Checks the output of function create_alias_mapping*

Description

Checks the output of function `create_alias_mapping`.

Usage

```
check_alias_mapping(Alias_mapping)
```

Arguments

`Alias_mapping`	Result of function `create_alias_mapping`.

Value

`TRUE` if `x` has the following properties: `Alias_mapping` is a non-empty data.frame with character columns `'old'` and `'new'`, without NA and "". Column `'new'` has no duplicates. Else throws an error. Prints the duplicates of column `'new'`, if available.

check_format_statistics *Checks the output of function format_statistics*

Description

Checks the output of function `format_statistics`.

Usage

```
check_format_statistics(x)
```

Arguments

`x`	Result of function `format_statistics`.

Value

`TRUE` if `x` has the following properties: `x` is a non-empty data.frame with 2 columns called `'tag'` and `'value'`. Column `'tag'` has class factor and no duplicates. Column `'value'` is a character. Else throws an error.

check_format_tests *Checks the output of functions format_test*

Description

Checks the output of function format_tests.

Usage

```
check_format_tests(x)
```

Arguments

x	Result of function format_tests.

Value

TRUE if x has the following properties: x is a data.frame with exactly one row and with unique colnames. Else throws an error.

check_statistics *Checks the output of function statistics*

Description

Checks the output of function statistics.

Usage

```
check_statistics(x)
```

Arguments

x	Result of function statistics.

Value

TRUE if x has the following properties: x is a named list with length > 0. The names of the list must not have duplicates. The names may contain NA. Else an error.

check_tests *Checks the output of functions two_sample_htest and multi_sample_htest*

Description

Checks the output of function `two_sample_htest` and `multi_sample_htest`.

Usage

```
check_tests(x)
```

Arguments

x	Result of function `two_sample_htest` or `multi_sample_htest`.

Value

`TRUE` if `x` has the following properties: `x` is a named list with length > 0. The names of the list must not have duplicates. The names may contain NA. Else an error.

Most hypothesis-test-functions in R like `t.test` or `chisq.test` return an object of class htest. This object passes this checks. Additional fields can be added to these objects and they will still pass this check.

create_alias_mapping *Get Aliases of column names*

Description

Column names of data.frame in atable must have syntactically valid colnames, see `is_syntactically_valid_name`. So no blanks or special characters allowed. But Reporting in human readable language needs special characters. These functions here allow atable to handle arbitrary character for pretty printing.

Usage

```
create_alias_mapping(DD, ...)
```

Arguments

DD	A data.frame
...	Passed from and to other methods.

Details

We use attributes here, to assign alternative names to columns. Also class labelled created by Hmisc's label is supported.

See create_alias_mapping for the function that does the actual work.

If no aliases are found, then underscores in the column names of DD will be replaced by blanks. See Examples in ?atable.

Value

create_alias_mapping returns a data.frame with two columns old and new and as many rows as DD has columns. Column old contains the original column names of DD and column new their aliases.

---

format_statistics *Format statistics*

Description

The results of function statistics must be formated before printing. format_statistics does this.

Usage

```
format_statistics(x, ...)

## S3 method for class 'statistics_numeric'
format_statistics(x, format_statistics.statistics_numeric = NULL, ...)

## S3 method for class 'statistics_factor'
format_statistics(x, format_statistics.statistics_factor = NULL, ...)

## S3 method for class 'statistics_count_me'
format_statistics(x, ...)

## Default S3 method:
format_statistics(x, ...)
```

Arguments

x	An object.
...	Passed from and to other methods.
format_statistics.statistics_numeric	Either NULL or a function. Default is NULL. If a function, then it will replace atable:::format_statistics.statistics_numeric. The function must mimic format_statistics: arguments are x and the ellipsis ... . Result is a non-empty data.frame with 2 columns called 'tag' and 'value'.
format_statistics.statistics_factor	Analog to argument format_statistics.statistics_numeric

Details

This function defines which statistics are printed in the final table and how they are formated.

The format depends on the class `x`. See section methods.

If you are not pleased with the current format you may alter these functions. But you must keep the original output-format, see section Value. Function `check_format_statistics` checks if the output of statistics is suitable for further processing.

Value

A non-empty data.frame with 2 columns called `'tag'` and `'value'`. Column `'tag'` has class factor and no duplicates. Column `'value'` is a character. See also function `check_format_statistics`.

Methods (by class)

- `format_statistics(statistics_numeric)`: Defines how to format class `statistics_numeric`. Returns a data.frame with 2 rows. Column `'tag'` contains `'Mean_SD'` and `'valid_missing'`. Column `'value'` contains two values: first value is the rounded mean and standard deviation, pasted them together. The standard deviation is bracketed. Second value is the number of non-missing and missing values pasted together. The number of missing values is bracketed.
- `format_statistics(statistics_factor)`: Defines how to format class `statistics_factor`. Returns a data.frame. Column `'tag'` contains all names of `x`. Column `'value'` contains the percentages and the total number of values in brackets.
- `format_statistics(statistics_count_me)`: Defines how to format class `statistics_count_me`. Returns a data.frame. Column `'tag'` contains the empty character `''`. The empty character is choosen because `colname_for_observations` already appears in the final table. Column `'value'` contains the number of observations. See also `'colname_for_observations'` in `atable_options`.
- `format_statistics(default)`: Returns a data.frame. Column `'tag'` contains all names of `x`. Column `'value'` contains all elements of `x`, rounded by `format`.

`format_tests` *Formats hypothesis test results*

Description

The results of function `two_sample_htest` and `multi_sample_htest` must be formated before printing. `format_tests` does this.

Usage

```
format_tests(x, ...)

## S3 method for class 'htest'
format_tests(x, format_tests.htest = NULL, ...)
```

```
## S3 method for class 'htest_with_effect_size'
format_tests(x, format_tests.htest_with_effect_size = NULL, ...)

## Default S3 method:
format_tests(x, ...)
```

Arguments

x	An object.
...	Passed from and to other methods.
format_tests.htest	Either NULL or a function. Default is NULL. If a function, then it will replace format_tests.htest. The function must mimic format_tests: arguments are x and the ellipsis ... . Result is a data.frame with 1 rows and unique colnames.
format_tests.htest_with_effect_size	Analog to argument format_tests.htest

Details

This function defines which test results are printed in the final table and how they are formated.

The format depends on the class x. See section methods.

If you are not pleased with the current format you may alter these functions. But you must keep the original output-format, see section Value. Function check_format_tests checks if the output of format_tests is suitable for further processing.

Value

A non-empty data.frame with one row. See also function check_format_tests.

Methods (by class)

- format_tests(htest): Defines how to format class htest. Returns a data.frame with 1 rows. Column p contains the p-value of the x.
- format_tests(htest_with_effect_size): Defines how to format class htest_with_effect_size. Returns a data.frame with 1 rows. Column p contains the p-value of the x. Column stat contains the teststatistic. Column Effect Size (CI) contains a effect size and its 95% Confidence interval.
- format_tests(default): Tries to cast to data.frame with one row. Uses the names of the list as colnames.

get_alias *Get Aliases of column names*

Description

Retrieves attributes `label` and `units` of class labelled and attribute `alias` otherwise.

Usage

```
get_alias(x, ...)

## S3 method for class 'labelled'
get_alias(x, ...)

## Default S3 method:
get_alias(x, ...)

## S3 method for class 'data.frame'
get_alias(x, ...)

## S3 method for class 'list'
get_alias(x, ...)
```

Arguments

`x`	An object. Aliases will be retrieved of `x`.
`...`	Passed from and to other methods.

Details

We use `attributes` here, to assign alternative names to columns. Also class `labelled` created by Hmisc's `label` is supported.

This is a workhorse function, see `create_alias_mapping` for the high level function

Value

For atomic vectors a character of `NULL`; for non-atomic vectors the results of `get_alias` applied to its elements.

Methods (by class)

- `get_alias(labelled)`: Retrieve attributes `label` and `units`, if available. Units are bracketed by '[]'. See also `label` and `units`. The user may alter this method via `atable_options`, see help there.
- `get_alias(default)`: Retrieve attribute `alias` via `attr`. This attribute may be an arbitrary character. If there is no attribute `alias`, then `get_alias.default` returns `NULL`.

- `get_alias(data.frame)`: Calls `get_alias` on every column.
- `get_alias(list)`: Calls `get_alias` on every element of the list.

`indent_data_frame` *Indents data.frames*

Description

Indents data.frames for printing them as tables.

Usage

```
indent_data_frame(
  DD,
  keys,
  values = setdiff(colnames(DD), keys),
  character_empty = "",
  numeric_empty = NA,
  indent_character = "\\quad",
  colname_indent = "Group"
)
```

Arguments

`DD`	A data.frame. Should be sorted by `keys` with `keys[1]` varying slowest and `keys[length(keys)]` varying fastest.
`keys`	A character. Subset of `colnames(DD)` with `length(keys)>=2`. The combination of keys must be unique. `DD[keys]` must be class character or factor.
`values`	A character. Subset of colnames(DD). DD[keys] must be class character, factor or numeric.
`character_empty`	A character. Default ''. This character will be put in the new lines in class character columns.
`numeric_empty`	A numeric. Default NA. This character will be put in the new lines in class numeric columns.
`indent_character`	A character. character for one indent. Default is '\quad' (meant for latex). Can also be ' ' for Word.
`colname_indent`	A character. Default 'Group'. Name of the new column with the indented keys.

Details

Squeeze multiple key-columns into one column and indents the values accordingly. Adds new lines with the indented keys to the data.frame. Meant for wide tables that need to be narrower and more 'readable' Meant for plotting with e.g. xtable::xtable or Hmisc::latex or officer::body_add_table. Look at the examples for a more precise description. Meant for left-aligned columns. Thats why the `indent_character` is inserted to the left of the original values.

Value

A data.frame. Columns: `c(colname_indent, values)`. Column `colname_indent` contains all combination of `DD[keys]`, but now indented and squeezed in this column and casted to character. Columns `'values'` contain all values of `DD[values]` unchanged. Number of rows is `sum(cumprod(nlevels(DD[keys])))`.

Examples

```
DD <- expand.grid(Arm = paste0('Arm ', c(1,2,4)),
                  Gender = c('Male', 'Female'),
                  Haircolor = c('Red', 'Green', 'Blue'),
                  Income = c('Low', 'Med', 'High'), stringsAsFactors = TRUE)

DD <- doBy::orderBy(~ Arm + Gender + Haircolor + Income, DD)

DD$values1 <- runif(dim(DD)[1])
DD$values2 <- 1
DD$values3 <- sample(letters[1:4], size = nrow(DD), replace = TRUE)

keys = c('Arm', 'Gender', 'Haircolor', 'Income')
values = c('values1', 'values2', 'values3')
## Not run:
DDD <- indent_data_frame(DD, keys, indent_character = '   ')

# print both:

Hmisc::latex(DD,
      file = '',
      longtable = TRUE,
      caption = 'Original table',
      rowname = NULL)

Hmisc::latex(DDD,
      file = '',
      longtable = TRUE,
      caption = 'Indented table',
      rowname = NULL)

## End(Not run)
```

`is_syntactically_valid_name` *Checks if valid name*

Description

Checks for valid names by `make.names`, i.e. `x` is valid iff `make.names` does nothing with `x`.

Usage

```
is_syntactically_valid_name(x)
```

Arguments

x	An object.

Value

A logical with length 1. TRUE when x is a character with length > 0 without duplicates and is valid. Else FALSE and a warning what's wrong.

Examples

```
x <- c('asdf', NA,'.na', '<y', 'asdf', 'asdf.1')
is_syntactically_valid_name(x)
is_syntactically_valid_name(x[FALSE]) # FALSE because empty
is_syntactically_valid_name(NA) # FALSE because not character
is_syntactically_valid_name(as.character(NA)) # FALSE because NA
is_syntactically_valid_name('NA') # FALSE. make.names changes 'NA' to 'NA.'
is_syntactically_valid_name(letters) # TRUE
```

---

multi_sample_htest *Calculates multi sample hypothesis tests*

Description

Calculates multi sample hypothesis tests depending on the class of its input.

Usage

```
multi_sample_htest(value, group, ...)

## S3 method for class 'logical'
multi_sample_htest(value, group, ...)

## S3 method for class 'factor'
multi_sample_htest(value, group, multi_sample_htest.factor = NULL, ...)

## S3 method for class 'character'
multi_sample_htest(value, group, ...)

## S3 method for class 'ordered'
multi_sample_htest(value, group, multi_sample_htest.ordered = NULL, ...)

## S3 method for class 'numeric'
multi_sample_htest(value, group, multi_sample_htest.numeric = NULL, ...)
```

Arguments

value	An atomic vector.
group	A factor, same length as value.
...	Passed to methods.
multi_sample_htest.factor	Analog to argument two_sample_htest.numeric
multi_sample_htest.ordered	Analog to argument two_sample_htest.numeric
multi_sample_htest.numeric	Either NULL or a function. Default is NULL. If a function, then it will replace atable:::multi_sample_htest.numeric. The function must mimic multi_sample_htest.numeric: arguments are value, group and the ellipsis ... . Result is a named list with length > 0 with unique names.

Details

Calculates multi sample hypothesis tests depending on the class of its input.

Results are passed to function format_tests for the final table.

If you are not pleased with the current hypothesis tests you may alter these functions. But you must keep the original output-format, see section Value. Function check_tests checks if the output of statistics is suitable for further processing.

The function multi_sample_htest is essentially a wrapper to standardize the arguments of various hypothesis test functions.

Value

A named list with length > 0.

Most hypothesis-test-functions in R like t.test or chisq.test return an object of class 'htest'. 'htest'-objects are a suitable output for function two_sample_htest. Function check_tests checks if the output is suitable for further processing.

Methods (by class)

- multi_sample_htest(logical): Casts to factor and then calls method multi_sample_htest again.
- multi_sample_htest(factor): Calls chisq.test.
- multi_sample_htest(character): Casts value to factor and then calls method multi_sample_htest again.
- multi_sample_htest(ordered): Calls kruskal.test.
- multi_sample_htest(numeric): Calls multi_sample_htest's method on ordered(value).

`replace_consecutive` *Replaces consecutive elements*

Description

If `x[i+1]=x[i]` then `x[i+1]` is replaced by by for `i=1,...length(x)-1`.

Usage

```
replace_consecutive(x, by = "", fun_for_identical = base::identical)
```

Arguments

`x`	A character or factor.
`by`	A character with length 1.
`fun_for_identical`	A function with two arguments called `x` and `y`.

Details

The = is defined by function `identical` by default. This function can be changed by argument fun_for_identical

Value

A character, same length as `x`, now with consecutives replaced by `by`. If `length(x) < 2`, x is returned unchanged.

Examples

```
x <- rep(c('a','b','c','d'), times=c(2,4,1,3))
x
## Not run: replace_consecutive(x)
# NA should not be identical. So change fun_for_identical
fun_for_identical <- function(x,y) !is.na(x) && !is.na(y) && identical(x,y)
x <- c(1,1,3,3,NA,NA, 4)
x
## Not run: replace_consecutive(x, by="99")
## Not run: replace_consecutive(x, by="99", fun_for_identical = fun_for_identical)
```

replace_NA *Replaces NA*

Description

Replaces NA in characters, factors and data.frames.

Usage

```
replace_NA(x, ...)

## S3 method for class 'character'
replace_NA(x, replacement = atable_options("replace_NA_by"), ...)

## S3 method for class 'factor'
replace_NA(x, ...)

## S3 method for class 'ordered'
replace_NA(x, ...)

## S3 method for class 'data.frame'
replace_NA(x, ...)

## S3 method for class 'list'
replace_NA(x, ...)

## Default S3 method:
replace_NA(x, ...)
```

Arguments

x	An object.
...	Passed to methods.
replacement	A character of length 1. Default value is defined in atable_options('replace_NA_by'), see atable_options.

Details

The atable package aims to create readable tables. For non-computer-affine readers NA has no meaning. So replace_NA exists.

Methods for character, factor, ordered, list and data.frame available. Default method returns x unchanged.

Gives a warning when replacement is already present in x and does the replacement.

Silently returns x unchanged when there are no NA in x.

Silently returns x unchanged when replacement is not a character of length 1 or when replacement is NA.

Value

Same class as x, now with NA replaced by replacement.

Methods (by class)

- replace_NA(character): replaces NA with replacement.
- replace_NA(factor): applies replace_NA to the levels of the factor. A factor with length > 0 without levels will get the level replacement.
- replace_NA(ordered): as factor.
- replace_NA(data.frame): applies replace_NA to all columns.
- replace_NA(list): applies replace_NA to all elements of the list.
- replace_NA(default): return x unchanged.

Examples

```
Character <- c(NA,letters[1:3], NA)
Factor <- factor(Character)
Ordered <- ordered(Factor)
Numeric <- rep(1, length(Factor))
Factor_without_NA <- factor(letters[1:length(Factor)])

DD <- data.frame(Character, Factor, Ordered,
                 Numeric, Factor_without_NA,
                 stringsAsFactors = FALSE)
## Not run:
DD2 <- replace_NA(DD, replacement = 'no value')

summary(DD)
summary(DD2) # now with 'no value' instead NA in column Character, Factor and Ordered

atable_options(replace_NA_by = 'not measured') # use atable_options to set replacement
DD3 <- replace_NA(DD)
summary(DD3) # now with 'not measured' instead NA

atable_options_reset() # set 'replace_NA_by' back to default

## End(Not run)
```

---

standardized_test_data

A data.frame with standardized random data of various classes

---

Description

A data.frame intended for testing the atable function with standardized random data and missing values in various classes.

Usage

```
standardized_test_data
```

Format

A data frame with 1080 rows and 7 variables:

Split1 A factor with 2 levels without NA. The two levels have the same frequency (540).

Split2 A factor with 2 levels with NA. The two levels and the NA have the same frequency (360).

Group A factor with 2 levels with NA. The two levels and the NA have the same frequency (360).

Logical A logical.

Factor A factor with 3 levels.

Ordered Class ordered with 4 levels.

Numeric Class numeric.

Details

For every subset defined by a triplet of the levels of Split1, Split2 and Group the variables have the following properties:

- 60 observations
- Logical has exactly the same number of TRUE and FALSE and NA (20).
- Factor has exactly the same number of levels taken and NA (15).
- Ordered has exactly the same number of levels taken and NA (12).
- Numeric is sampled from a normal distribution and then standardized to sd 1 and with 6 NA. Its mean is 12 when Group is 'Treatment' and 10 otherwise (up to 10^-17).

Examples

```
atable::atable(Logical+ Numeric + Factor + Ordered ~ Group | Split1 + Split2,
 atable::standardized_test_data, add_levels_for_NA = TRUE, format_to = 'Word')
```

---

statistics *Calculates descriptive statistics*

Description

Calculates descriptive statistics depending on the class of its input.

Usage

```
statistics(x, ...)

## S3 method for class 'numeric'
statistics(x, statistics.numeric = NULL, ...)

## S3 method for class 'factor'
statistics(x, statistics.factor = NULL, ...)

## S3 method for class 'logical'
statistics(x, labels_TRUE_FALSE = atable_options("labels_TRUE_FALSE"), ...)

## S3 method for class 'character'
statistics(x, ...)

## S3 method for class 'ordered'
statistics(x, statistics.ordered = NULL, ...)

## S3 method for class 'count_me'
statistics(x, ...)
```

Arguments

x	An object. Statistics will be calculated of x.
...	Passed from and to other methods.
statistics.numeric	Either NULL or a function. Default is NULL. If a function, then it will replace atable:::statistics.numeric. The function must mimic statistics: arguments are x and the ellipsis ... . Result is a named list with length > 0 with unique names.
statistics.factor	Analog to argument statistics.numeric
labels_TRUE_FALSE	For relabeling logicals. See also atable_options.
statistics.ordered	Analog to argument statistics.numeric

Details

Calculates descriptive statistics depending on the class of its input.

Results are passed to function format_statistics.

If you are not pleased with the current descriptive statistics you may alter these functions. But you must keep the original output-format, see section Value. Function check_statistics checks if the output of statistics is suitable for further processing.

Value

The results of `statistics` are passed to function `format_statistics`. So the results of `statistics` must have a class for which the generic `format_statistics` has a method.

`format_statistics` has a default method, which accepts lists. So the results of `statistics` can be a named list with length > 0. The names of the list must have no duplicates.

Function `check_statistics` checks if the output of statistics is suitable for further processing.

Methods (by class)

- `statistics(numeric)`: Descriptive statistics are: length, number of missing values, mean and standard deviation. Class of the result is `'statistics_numeric'` and there is a method `format_statistics_to_Latex.statistics_numeric`. This function is meant for interval scaled variables.
- `statistics(factor)`: Counts the numbers of occurrences of the levels of `x` with function `table`. This function is meant for nominal and ordinal scaled variables.
- `statistics(logical)`: Casts `x` to factor, then applies `statistics` again. The labels for `TRUE` and `FALSE` can also be modfied by setting `atable_options('labels_TRUE_FALSE')`.
- `statistics(character)`: Casts `x` to factor, then applies `statistics` again.
- `statistics(ordered)`: Casts `x` to factor, then applies `statistics` again.
- `statistics(count_me)`: Returns the `length` of `x`. For class `'count_me'` see `add_observation_column`.

`test_data` *A data.frame with random data of various classes*

Description

A data.frame intended for testing the atable function with random data and missing values in various classes.

Usage

```
test_data
```

Format

A data frame with 129 rows and 11 variables:

Split1 A factor with 2 levels, drawn uniformly.

Split2 A factor with 3 levels, drawn uniformly.

Group A factor with 2 levels, drawn uniformly.

Group2 A factor with 3 levels, drawn uniformly.

Numeric A sample from the standard normal distribution.

Numeric2 A sample from the normal distribution with mean 4 and sd 3.

Logical A Logical, drawn uniformly from `TRUE`, `FALSE` and `NA`.

Factor A factor with 4 level drawn with weigths `1:1:2:2`.

Ordered Class Ordered with 3 levels, drawn uniformly.

Character Class character drawn uniformly from `c('a', 'b', '')`.

Date Class Date, generated by adding `2001-05-25` to a sample of the Poisson distribution with lambda `42`.

6 Missing values were randomly added to each of Numeric, Numeric2, Factor, Ordered, Character and Date.

`translate_to_LaTeX` *A wrapper for latexTranslate*

Description

Translate_to_LaTeX calls `latexTranslate`.

Usage

```
translate_to_LaTeX(x, ...)

## S3 method for class 'data.frame'
translate_to_LaTeX(x, ...)

## S3 method for class 'list'
translate_to_LaTeX(x, ...)

## S3 method for class 'character'
translate_to_LaTeX(
  x,
  inn = NULL,
  out = NULL,
  pb = FALSE,
  greek = FALSE,
  na = "",
  ...
)

## S3 method for class 'numeric'
translate_to_LaTeX(x, ...)

## S3 method for class 'factor'
translate_to_LaTeX(x, ...)

## S3 method for class 'logical'
translate_to_LaTeX(x, ...)
```

Arguments

x	An object.
inn, out, pb, greek, na, ...	As in latex.

Details

Result is suitable for print with latex.

Translate_to_LaTeX uses S3 object system. See seection methods.

Value

Same length as x, now translated to latex.

Methods (by class)

- translate_to_LaTeX(data.frame): Applies latexTranslate to rownames(x), colnames(x) and all columns of x.
- translate_to_LaTeX(list): Translates all elements of x.
- translate_to_LaTeX(character): As latexTranslate.
- translate_to_LaTeX(numeric): Casts to character and then translates.
- translate_to_LaTeX(factor): Translates the levels of the factor.
- translate_to_LaTeX(logical): Casts to character and then translates.

---

two_sample_htest *Two sample hypothesis tests and effect size*

Description

Calculates two sample hypothesis tests and effect size depending on the class of its input.

Usage

```
two_sample_htest(value, group, ...)

## S3 method for class 'character'
two_sample_htest(value, group, ...)

## S3 method for class 'factor'
two_sample_htest(value, group, two_sample_htest.factor = NULL, ...)

## S3 method for class 'logical'
two_sample_htest(value, group, ...)

## S3 method for class 'numeric'
```

```
two_sample_htest(value, group, two_sample_htest.numeric = NULL, ...)

## S3 method for class 'ordered'
two_sample_htest(value, group, two_sample_htest.ordered = NULL, ...)
```

Arguments

value	An atomic vector. These values will be tested.
group	A factor with two levels and same length as value. Defines the two groups of value, that are compared by a two sample hypothesis tests.
...	Passed to methods.
two_sample_htest.factor	Analog to argument two_sample_htest.numeric
two_sample_htest.numeric	Either NULL or a function. Default is NULL. If a function, then it will replace atable:::two_sample_htest.numeric. The function must mimic two_sample_htest.numeric: arguments are value, group and the ellipsis ... . Result is a named list with length > 0 with unique names.
two_sample_htest.ordered	Analog to argument two_sample_htest.numeric

Details

Results are passed to function format_tests for the final table. So the results of two_sample_htest must have a class for which the generic format_tests has a method.

If you are not pleased with the current hypothesis tests you may alter these functions. But you must keep the original output-format, see section Value.

Note that the various statistical test functions in R have heterogeneous arguments: for example chisq.test and ks.test do not have formula/data as arguments, whereas wilcox.test and kruskal.test do. So the function two_sample_htest is essentially a wrapper to standardize the arguments of various hypothesis test functions.

As two_sample_htest is only intended to be applied to unpaired two sample data, the two arguments value and group are sufficient to describe the data.

Note that e.g. for class numeric the p-value is calculated by ks.test and the effects size 95% CI by cohen.d. As these are two different functions the results may be contradicting: the p-value of ks.test can be smaller than 0.05 and the CI of cohen.d contains 0 at the same time.

Value

A named list with length > 0, where all elements of the list are atomic and have the same length.

Most hypothesis-test-functions in R like t.test or chisq.test return an object of class 'htest'. 'htest'-objects are a suitable output for function two_sample_htest. Function check_tests checks if the output is suitable for further processing.

Methods (by class)

- `two_sample_htest(character)`: Casts `value` to factor and then calls method `two_sample_htest` again.
- `two_sample_htest(factor)`: Calls `chisq.test` on `value`. Effect size is the odds ratio calculated by `fisher.test` (if `value` has two levels), or Cramer's V by `CramerV`.
- `two_sample_htest(logical)`: Casts `value` to factor and then calls `two_sample_htest` again.
- `two_sample_htest(numeric)`: Calls `ks.test` on `value`. Effect size is Cohen's d calculated by `cohen.d`.
- `two_sample_htest(ordered)`: Calls `wilcox.test` on `value`. Effect size is Cliff's delta calculated by `cliff.delta`.

Index