

Package ‘atable’

November 30, 2019

Type Package

Title Create Tables for Reporting Clinical Trials

Version 0.1.5

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 7.0.0

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, officer, flextable, survival, rmarkdown

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

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

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-30 01:20:10 UTC

R topics documented:

add_observation_column	2
atable	3

atable_options	6
atable_options_reset	9
atable_package	9
check_format_statistics	9
check_format_tests	10
check_statistics	10
check_tests	11
create_alias_mapping	11
format_statistics	12
format_tests	13
get_alias	14
indent_data_frame	16
is_syntactically_valid_name	17
multi_sample_hstest	18
replace_consecutive	20
replace_NA	20
standardized_test_data	22
statistics	23
test_data	25
translate_to_LaTeX	26
two_sample_hstest	27

Index **29**

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,
  ...
)

## 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_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](#): see the help there.
- `format_tests.htest_with_effect_size`: Analog to argument `format_tests.htest`

<code>target_cols</code>	<p>A character vector containing some column names of <code>x</code>.</p> <p>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.</p> <p>Hypothesis test are defined by two_sample_htest or multi_sample_htest (depending on the number of levels of <code>group_col</code>); their representation and format by format_tests. Note that <code>atable</code> always adds one name to <code>target_cols</code> to count the number of observations. This name is stored in <code>atable_options('colname_for_observat</code></p>
<code>group_col</code>	<p>A character of length 1 containing a column of <code>x</code> 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.</p>
<code>split_cols</code>	<p>A character vector containing some of <code>colnames(x)</code> or NULL. <code>x</code> 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.</p>
<code>format_to</code>	<p>A character vector of length 1. Specifies the format of the output of <code>atable</code>. Possible values are 'Latex', 'Word', 'Raw', 'HTML', 'Console'. Default is defined in atable_options.</p>
<code>drop_levels</code>	<p>A logical. If TRUE then droplevels is called on <code>group_col</code> and <code>split_cols</code> before further processing. Default is TRUE.</p>
<code>add_levels_for_NA</code>	<p>If TRUE then addNA is called on <code>group_col</code> and <code>split_cols</code> before further processing. Default is FALSE.</p>

formula	A formula of the form <code>target_cols ~ group_col split_cols</code> . The <code> </code> separates the <code>group_col</code> from the <code>split_cols</code> . Read the <code> </code> as 'given' as in a conditional probability $P(\text{target_cols} \mid \text{split_cols})$. <code>target_cols</code> and <code>split_cols</code> may contain multiple names separated by <code>+</code> . <code>group_col</code> must be a single name if given. <code>group_col</code> and <code>split_cols</code> may be omitted and can be replaced by <code>1</code> in this case. The <code> </code> may also be omitted if no <code>split_cols</code> are given.
data	Passed to <code>atable(x = data, ...)</code> .

Value

Results depend on `format_to`:

- 'Raw': A list with two elements called 'statistics_result' and 'tests_result', that contain all results of the descriptive statistics and the hypothesis tests. This format useful, when extracting a specific result unformatted (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)

- `data.frame`: applies descriptive statistics and hypothesis tests, arranges the results for printing.
- `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')

```

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:

- `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.
- `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](#).
- `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.
- `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`.
- `modify_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))`

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_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	<i>Checks the output of functions two_sample_hstest and multi_sample_hstest</i>
-------------	---

Description

Checks the output of function `two_sample_hstest` and `multi_sample_hstest`.

Usage

```
check_tests(x)
```

Arguments

`x` Result of function `two_sample_hstest` or `multi_sample_hstest`.

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 `hstest`. This object passes this checks. Additional fields can be added to these objects and they will still pass this check.

create_alias_mapping	<i>Get Aliases of column names</i>
----------------------	------------------------------------

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.

<code>format_statistics</code>	<i>Format statistics</i>
--------------------------------	--------------------------

Description

The results of function `statistics` must be formatted 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

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

Details

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

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)

- `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.
- `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.
- `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 chosen 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`.
- `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_hstest` and `multi_sample_hstest` must be formatted before printing. `format_tests` does this.

Usage

```
format_tests(x, ...)

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

## S3 method for class 'hstest_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 formatted.

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)

- `htest`: Defines how to format class `htest`. Returns a data.frame with 1 rows. Column `p` contains the p-value of the `x`.
- `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.
- `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)

- `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. xxx
- `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.
- `data.frame`: Calls `get_alias` on every column.
- `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. That's 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_hstest *Calculates multi sample hypothesis tests*

Description

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

Usage

```
multi_sample_hstest(value, group, ...)

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

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

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

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

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

Arguments

value	An atomic vector.
group	A factor, same length as value.
...	Passed to methods.
multi_sample_hstest.factor	Analog to argument two_sample_hstest.numeric
multi_sample_hstest.ordered	Analog to argument two_sample_hstest.numeric
multi_sample_hstest.numeric	Either NULL or a function. Default is NULL. If a function, then it will replace <code>atable::multi_sample_hstest.numeric</code> . The function must mimic <code>multi_sample_hstest.numeric</code> : arguments are value, group and the ellipsis Result is a named list with <code>length > 0</code> 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_hstest` 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 'hstest'. 'hstest'-objects are a suitable output for function `two_sample_hstest`. Function `check_tests` checks if the output is suitable for further processing.

Methods (by class)

- logical: Casts to factor and then calls method `multi_sample_hstest` again.
- factor: Calls `chisq.test`.
- character: Casts value to factor and then calls method `multi_sample_hstest` again.
- ordered: Calls `kruskal.test`.
- numeric: Calls `multi_sample_hstest`'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 $i=1, \dots, \text{length}(x)-1$.

Usage

```
replace_consecutive(x, by = "")
```

Arguments

x A character or factor.
 by A character with length 1.

Details

The $=$ is defined by function [identical](#).

Value

A character, same length as x , now with consecutives replaced by by . If $\text{length}(x) < 2$, x is returned unchanged.

Examples

```
x <- rep(c('a','b','c','d'), times=c(2,4,1,3))
x
## Not run: replace_consecutive(x)
```

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 <code>atable_options('replace_NA_by')</code> , 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)

- `character`: replaces NA with `replacement`.

- `factor`: applies `replace_NA` to the levels of the factor. A factor with length > 0 without levels will get the level replacement.
- `ordered`: as factor.
- `data.frame`: applies `replace_NA` to all columns.
- `list`: applies `replace_NA` to all elements of the list.
- `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)

- **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`. This function is meant for interval scaled variables.
- **factor**: Counts the numbers of occurrences of the levels of `x` with function `table`. This function is meant for nominal and ordinal scaled variables.
- **logical**: Casts `x` to factor, then applies `statistics` again. The labels for TRUE and FALSE can also be modified by setting `atable_options('labels_TRUE_FALSE')`.
- **character**: Casts `x` to factor, then applies `statistics` again.
- **ordered**: Casts `x` to factor, then applies `statistics` again.
- **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 weights 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 section methods.

Value

Same length as x, now translated to latex.

Methods (by class)

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

two_sample_hstest

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_hstest(value, group, ...)
```

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

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

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

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

```
## S3 method for class 'ordered'
two_sample_hstest(value, group, two_sample_hstest.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_hstest.factor`
 Analog to argument `two_sample_hstest.numeric`

`two_sample_hstest.numeric`
 Either NULL or a function. Default is NULL. If a function, then it will replace `atable:::two_sample_hstest.numeric`. The function must mimic `two_sample_hstest.numeric:` arguments are `value`, `group` and the ellipsis `...`. Result is a named list with `length > 0` with unique names.

`two_sample_hstest.ordered`
 Analog to argument `two_sample_hstest.numeric`

Details

Results are passed to function `format_tests` for the final table. So the results of `two_sample_hstest` 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_hstest` is essentially a wrapper to standardize the arguments of various hypothesis test functions.

As `two_sample_hstest` 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 `'hstest'`. `'hstest'`-objects are a suitable output for function `two_sample_hstest`. Function `check_tests` checks if the output is suitable for further processing.

Methods (by class)

- `character`: Casts `value` to `factor` and then calls method `two_sample_hstest` again.
- `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`.
- `logical`: Casts `value` to `factor` and then calls `two_sample_hstest` again.
- `numeric`: Calls `ks.test` on `value`. Effect size is Cohen's d calculated by `cohen.d`.
- `ordered`: Calls `wilcox.test` on `value`. Effect size is Cliff's delta calculated by `cliff.delta`.

Index

*Topic **datasets**

standardized_test_data, 22
test_data, 25

add_observation_column, 2, 25
addNA, 4
as.factor, 4
atable, 3, 7, 9
atable_options, 4, 6, 9, 13, 15, 21, 24
atable_options_reset, 7, 9
atable_package, 9
attr, 15
attributes, 12, 15

check_format_statistics, 9, 13
check_format_tests, 10, 14
check_statistics, 10, 24
check_tests, 11, 19, 28
chisq.test, 11, 19, 28
cliff.delta, 28
cohen.d, 28
CramerV, 28
create_alias_mapping, 11

droplevels, 4

fisher.test, 28
format, 13
format_statistics, 4, 5, 7-9, 12, 12, 24
format_tests, 4, 8, 10, 13, 14, 28

get_alias, 14

identical, 20
indent_data_frame, 16
is_syntactically_valid_name, 3, 11, 17

kruskal.test, 19, 28
ks.test, 28

label, 12, 15

latex, 26
latexTranslate, 26, 27
length, 25

make.names, 17
mean, 23
multi_sample_hstest, 4, 5, 8, 11, 13, 18
multi_sample_hstest.numeric, 19

replace_consecutive, 20
replace_NA, 7, 20

sd, 23
settings, 6
standardized_test_data, 22
statistics, 3-5, 7, 10, 23, 24

t.test, 11, 19, 28
table, 25
test_data, 25
translate_to_LaTeX, 5, 26
two_sample_hstest, 4, 5, 8, 11, 13, 27
two_sample_hstest.numeric, 28

units, 15

wilcox.test, 28