

# Package ‘DescrTab2’

October 12, 2022

**Type** Package

**Title** Publication Quality Descriptive Statistics Tables

**Version** 2.1.16

**Date** 2022-09-06

## Description

Provides functions to create descriptive statistics tables for continuous and categorical variables. By default, summary statistics such as mean, standard deviation, quantiles, minimum and maximum for continuous variables and relative and absolute frequencies for categorical variables are calculated. 'DescrTab2' features a sophisticated algorithm to choose appropriate test statistics for your data and provides p-values. On top of this, confidence intervals for group differences of appropriated summary measures are automatically produces for two-group comparison. Tables generated by 'DescrTab2' can be integrated in a variety of document formats, including .html, .tex and .docx documents. 'DescrTab2' also allows printing tables to console and saving table objects for later use.

**Depends** R (>= 4.0.0)

**Suggests** testthat, covr, tidyverse, here, shiny, Exact

**Imports** stats, utils, nlme, exact2x2, DescTools, dplyr, rlang, tibble, stringr, forcats, magrittr, scales, cli, kableExtra, flextable (>= 0.6.6), officer, knitr, rmarkdown, haven, Hmisc

**VignetteBuilder** knitr

**License** GPL (>= 3)

**Copyright** This package contains codes copyrighted by third parties.  
See file COPYRIGHTS for details.

**LazyLoad** true

**Encoding** UTF-8

**URL** <https://imbi-heidelberg.github.io/DscrTab2/>

**BugReports** <https://github.com/imbi-heidelberg/DscrTab2/issues>

**Config/testthat/edition** 3

**RoxygenNote** 7.2.0

**NeedsCompilation** no

**Author** Jan Meis [aut, cre] (<<https://orcid.org/0000-0001-5407-7220>>),  
 Lukas Baumann [aut] (<<https://orcid.org/0000-0001-7931-7470>>),  
 Maximilian Pilz [aut] (<<https://orcid.org/0000-0002-9685-1613>>),  
 Lukas Sauer [aut] (<<https://orcid.org/0000-0002-1340-9994>>),  
 Lorenz Uhlmann [ctb],  
 Csilla van Lunteren [ctb]

**Maintainer** Jan Meis <meis@imbi.uni-heidelberg.de>

**Repository** CRAN

**Date/Publication** 2022-09-06 08:50:02 UTC

## R topics documented:

.onLoad . . . . .	3
can.be.numeric . . . . .	3
codegen_load_all_sas_data . . . . .	4
create_character_subtable . . . . .	4
create_numeric_subtable . . . . .	5
descr . . . . .	6
DescrTab2 . . . . .	11
escape_latex_symbols . . . . .	12
extract_labels . . . . .	12
farrington.manning . . . . .	13
format_freqs . . . . .	14
guess_ID_variable . . . . .	15
ignore_unused_args . . . . .	16
in_minipage . . . . .	17
knit_print.DescrList . . . . .	17
knit_print.DescrPrint . . . . .	18
lapply_descr . . . . .	18
list_freetext_markdown . . . . .	19
n_int_digits . . . . .	20
parse_formats . . . . .	20
print.DescrList . . . . .	21
print_test_names . . . . .	22
read_redcap_formatted . . . . .	22
read_sas_formatted . . . . .	23
sigfig . . . . .	24
sigfig_gen . . . . .	24
sig_test . . . . .	25
split_redcap_dataset . . . . .	26
unlabel . . . . .	26
write_in_tmpfile_for_cran . . . . .	27

**Index**

**28**

---

.onLoad                      *Load LaTeX packages*

---

### **Description**

Load LaTeX packages

### **Usage**

```
.onLoad(libname = find.package("kableExtra"), pkgname = "kableExtra")
```

### **Arguments**

libname	library name
pkgname	package name

### **Details**

Thanks to Hao Zhu and his package [kableExtra](#).

### **Author(s)**

Hao Zhu

---

can.be.numeric              *Check whether a vector (usually a factor) can be cleanly converted to a numeric*

---

### **Description**

From <https://stackoverflow.com/a/47677916>

### **Usage**

```
can.be.numeric(x)
```

### **Arguments**

x	a vector
---	----------

```
codegen_load_all_sas_data
```

*Create code to load all SAS datasets in a folder.*

---

**Description**

This is useful if you work with lots of separate SAS datasets spread out in the same folder.

**Usage**

```
codegen_load_all_sas_data(dir, format = NULL)
```

**Arguments**

dir	path to dataset folder
format	path to format file

**Value**

NULL. Relevant code is printed to the console.

**Examples**

```
codegen_load_all_sas_data(system.file("examples", package = "DescrTab2"))
```

---

```
create_character_subtable
```

*Function to create (a part of a) nicely formatted table*

---

**Description**

Function to create (a part of a) nicely formatted table

**Usage**

```
create_character_subtable(  
  DescrVarObj,  
  var_name,  
  format_options,  
  format_summary_stats,  
  format_p,  
  reshape_rows  
)
```

**Arguments**

DescrVarObj     Variable object to be formatted  
var\_name        (character) Name of the variable  
format\_options   named list of options for formatting  
format\_summary\_stats  
                  named list of summary statistics  
format\_p        formatting function for p-values  
reshape\_rows    named list of row reshaping functions

---

create\_numeric\_subtable

*Function to create (a part of a) nicely formatted table*

---

**Description**

Function to create (a part of a) nicely formatted table

**Usage**

```
create_numeric_subtable(  
  DescrVarObj,  
  var_name,  
  format_options,  
  format_summary_stats,  
  format_p,  
  reshape_rows  
)
```

**Arguments**

DescrVarObj     Variable object to be formatted  
var\_name        (character) Name of the variable  
format\_options   named list of options for formatting  
format\_summary\_stats  
                  named list of summary statistics  
format\_p        formatting function for p-values  
reshape\_rows    named list of row reshaping functions

---

 descr

*Calculate descriptive statistics*


---

## Description

Generate a list of descriptive statistics. By default, the function calculates summary statistics such as mean, standard deviation, quantiles, minimum and maximum for continuous variables and relative and absolute frequencies for categorical variables. Also calculates p-values for an appropriately chosen statistical test. For two-group comparisons, confidence intervals for appropriate summary measures of group differences are calculated aswell. In particular, Wald confidence intervals from [prop.test](#) are used for categorical variables with 2 levels, confidence intervals from [t.test](#) are used for continuous variables and confidence intervals for the Hodges-Lehman estimator [1] from [wilcox.test](#) are used for ordinal variables.

## Usage

```
descr(
  dat,
  group = NULL,
  group_labels = list(),
  var_labels = list(),
  var_options = list(),
  summary_stats_cont = list(N = DescrTab2:::.N, Nmiss = DescrTab2:::.Nmiss, mean =
    DescrTab2:::.mean, sd = DescrTab2:::.sd, median = DescrTab2:::.median, Q1 =
    DescrTab2:::.Q1, Q3 = DescrTab2:::.Q3, min = DescrTab2:::.min, max =
    DescrTab2:::.max),
  summary_stats_numeric_ord = list(N = DescrTab2:::.factorN, Nmiss =
    DescrTab2:::.factorNmiss, mean = DescrTab2:::.factormean, sd = DescrTab2:::.factorsd,
    median = DescrTab2:::.factormedian, Q1 = DescrTab2:::.factorQ1, Q3 =
    DescrTab2:::.factorQ3, min = DescrTab2:::.factormin, max = DescrTab2:::.factormax),
  summary_stats_cat = list(),
  format_summary_stats = list(N = function(x) {
    format(x, digits = 2, scientific =
      3)
  }, mean = function(x) {
    format(x, digits = 2, scientific = 3)
  }, sd =
    function(x) {
    format(x, digits = 2, scientific = 3)
  }, median = function(x) {
    format(x, digits = 2, scientific = 3)
  }, Q1 = function(x) {
    format(x, digits
    = 2, scientific = 3)
  }, Q3 = function(x) {
    format(x, digits = 2, scientific =
```

```

    3)
  }, min = function(x) {
    format(x, digits = 2, scientific = 3)
  }, max =
    function(x) {
    format(x, digits = 2, scientific = 3)
  }, CI = function(x) {

    format(x, digits = 2, scientific = 3)
  }),
format_p = scales::pvalue_format(),
format_options = list(print_Total = NULL, print_p = TRUE, print_CI = TRUE,
  combine_mean_sd = FALSE, combine_median_Q1_Q3 = FALSE, omit_factor_level = "none",
  omit_Nmiss_if_0 = TRUE, omit_missings_in_group = TRUE, percent_accuracy = NULL,
  percent_suffix = "%", row_percent = FALSE, Nmiss_row_percent = FALSE,
  absolute_relative_frequency_mode = c("both", "only_absolute", "only_relative"),
  omit_missings_in_categorical_var = FALSE, categorical_missing_percent_mode =
  c("no_missing_percent", "missing_as_regular_category",
  "missing_as_separate_category"), caption = NULL, replace_empty_string_with_NA = TRUE,
  categories_first_summary_stats_second = FALSE, max_first_col_width = 7.5),
test_options = list(paired = FALSE, nonparametric = FALSE, exact = FALSE, var_equal =
  FALSE, indices = c(), guess_id = FALSE, include_group_missings_in_test = FALSE,
  include_categorical_missings_in_test = FALSE, test_override = NULL,
  additional_test_args = list(), boschloo_max_n = 200),
reshape_rows = list(`Q1 - Q3` = list(args = c("Q1", "Q3"), fun = function(Q1, Q3) {

  paste0(Q1, " -- ", Q3)
}), `min - max` = list(args = c("min", "max"), fun =
  function(min, max) {
  paste0(min, " -- ", max)
})),
  ...
)

```

## Arguments

<code>dat</code>	Data frame or tibble. The data set to be analyzed. Can contain continuous or factor (also ordered) variables.
<code>group</code>	name (as character) of the group variable in <code>dat</code> .
<code>group_labels</code>	named list of labels for the levels of the group variable in <code>dat</code> .
<code>var_labels</code>	named list of variable labels.
<code>var_options</code>	named list of lists. For each variable, you can have special options that apply only to that variable. These options are specified in this argument. See the details and examples for more explanation.
<code>summary_stats_cont</code>	named list of summary statistic functions to be used for numeric variables.

summary\_stats\_numeric\_ord  
                   named list of summary statistic function to be used for ordered factor variables  
                   which can be converted to numeric.

summary\_stats\_cat  
                   named list of summary statistic function to be used for categorical variables.

format\_summary\_stats  
                   named list of formatting functions for summary statistics.

format\_p          formatting function for p-values.

format\_options   named list of formatting options.

test\_options     named list of test options.

reshape\_rows     named list of lists. Describes how to combine different summary statistics into  
                   the same row.

...              further argument to be passed along

### Value

Returns a `A DescrList` object, which is a named list of descriptive statistics which can be passed along to the print function to create pretty summary tables.

### Labels

`group_labels` and `var_labels` need to be named lists of character elements. The names of the list elements have to match the variable names in your dataset. The values of the list elements are the labels that will be assigned to these variables when printing.

### Custom summary statistics

`summary_stats_cont` and `summary_stats_cat` are both named lists of functions. The names of the list elements are what will be displayed in the leftmost column of the descriptive table. These functions should take a vector and return a value.

Each summary statistic has to have an associated formatting function in the `format_summary_stats` list. The functions in `format_summary_stats` take a numeric value and convert it to a character string, e.g. 0.2531235 -> "0.2".

The `format_p` function converts p-values to character strings, e.g. 0.05 -> "0.05" or 0.000001 -> "<0.001".

### Formatting options

Further formatting options can be specified in the `format_options` list. It contains the following members:

- `print_Total` (logical) controls whether to print the "Total" column. If `print_Total = NULL`, `print_Total` will be set to `TRUE` if `test_options$paired == FALSE`, else it will be set to `FALSE`.
- `print_p` (logical) controls whether to print the p-value column.
- `print_CI` (logical) controls whether to print the confidence intervals for group-differences.
- `combine_mean_sd` (logical) controls whether to combine the mean and sd row into one mean  $\pm$  sd row. This is a shortcut argument for the specification of an appropriate entry in the `reshape_rows` argument.



- `combine_median_Q1_Q3` (logical) controls whether to combine the median, Q1 and Q3 row into one median (Q1, Q3) row. This is a shortcut argument for the specification of an appropriate entry in the `reshape_rows` argument.
- `omit_Nmiss_if_0` (logical) controls whether to omit the Nmiss row in continuous variables there are no missings in the variable.
- `omit_missings_in_group` (logical) controls whether to omit all observations where the group variable is missing.
- `percent_accuracy` (numeric) A number to round to. Use (e.g.) 0.01 to show 2 decimal places of precision. If NULL, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. See documentation of `scales::label_percent`
- `percent_suffix` (character) the symbol to be used where "%" is appropriate, sensible choices are usually "%" (default) or "" (i.e., empty string)
- `row_percent` (logical) controls whether percentages of regular categorical variables should be calculated column-wise (default) or row-wise
- `Nmiss_row_percent` (logical) controls whether percentages of the "Nmiss"-statistic (number of missing values) should be calculated column-wise (default) or row-wise
- `absolute_relative_frequency_mode` (character) controls how to display frequencies. It may be set to one of the following options:
  - "both" will display absolute and relative frequencies.
  - "only\_absolute" will only display absolute frequencies.
  - "only\_relative" will only display relative frequencies.
- `omit_missings_in_categorical_var` (logical) controls whether to omit missing values in categorical variables completely.
- `categorical_missing_percent_mode` (character) controls how to display percentages in categorical variables with a (Missing) category. It may be set to one of the following options:
  - "no\_missing\_percent" omits a percentage in the missing category entirely.
  - "missing\_as\_regular\_category" treats (Missing) as a regular category for %-calculation. This means that if You have three categories: "A" with 10 counts, "B" with 10 counts and "(Missing)" with 10 counts, they will become "A": 10 (33%), "B": 10 (33%), "(Missing)": 10 (33% purposes.)
  - "missing\_as\_separat\_category" calculates (Missing) percentages with respect to all observations (i.e.  $\#(\text{Missing}) / N$ ), but calculates all other category percentages with respect to the non-missing observations (e.g.  $\#A / N_{\text{nonmissing}}$ ). This means that if You have three categories: "A" with 10 counts, "B" with 10 counts and "(Missing)" with 10 counts, they will become "A": 10 (50%), "B": 10 (50%), "(Missing)": 10 (33%)
- "caption" adds a table caption to the LaTeX, Word or Pdf document
- `replace_empty_string_with_NA` (logical) controls whether empty strings ("" ) should be replaced with missing value (`NA_character_`).
- `categories_first_summary_stats_second` (logical) controls whether the categories should be printed first in the summary statistics table.
- `max_first_col_width` (numeric) controls the maximum width of the first column in LaTeX tables.

## Test options

`test_options` is a named list with test options. It's members `paired`, `nonparametric`, and `exact` (logicals) control which test in the corresponding situation. For details, check out the vignette: [https://imbi-heidelberg.github.io/DescrTab2/articles/b\\_test\\_choice\\_tree\\_pdf.pdf](https://imbi-heidelberg.github.io/DescrTab2/articles/b_test_choice_tree_pdf.pdf). The `test_options = list(test_override="<some test name>")` option can be specified to force usage of a specific test. This will produce errors if the data does not allow calculation of that specific test, so be wary. Use `print_test_names()` to see a list of all available test names. If `paired = TRUE` is specified, you need to supply an index variable `indices` that specifies which datapoints in your dataset are paired. `indices` may either be a length one character vector that describes the name of the index variable in your dataset, or a vector containing the respective indices. If you have `guess_id` set to `TRUE` (the default), `DescrTab2` will try to guess the ID variable from your dataset and report a warning if it succeeds. See [https://imbi-heidelberg.github.io/DescrTab2/articles/a\\_usage\\_guide.html#Paired-observations-1](https://imbi-heidelberg.github.io/DescrTab2/articles/a_usage_guide.html#Paired-observations-1) for a bit more explanation. The optional list `additional_test_args` can be used to pass arguments along to test functions, e.g. `additional_test_args=list(correct=TRUE)` will request continuity correction if available.

## Customization for single variables

The `var_options` list can be used to conduct customizations that should only apply to a single variable and leave the rest of the table unchanged.

`var_options` is a list of named lists. This means that each member of `var_options` is itself a list again. The names of the list elements of `var_options` determine the variables to which the options will apply. Let's say you have an `age` variable in your dataset. To change 'descr' options only for `age`, you will need to pass a list of the form `var_options = list(age = list(<Your options here>))`.

You can replace `<Your options here>` with the following options:

- `label` a character string containing the label for the variable
- `summary_stats` a list of summary statistics. See section "Custom summary statistics"
- `format_summary_stats` a list of formatting functions for summary statistics. See section "Custom summary statistics"
- `format_p` a function to format p-values. See section "Custom summary statistics"
- `format_options` a list of formatting options. See section "Formatting options"
- `test_options` a list of test options. See section "Test options"
- `test_override` manually specify the name of the test you want to apply. You can see a list of choices by typing `print_test_names()`. Possible choices are:
  - "Cochran's Q test"
  - "McNemar's test"
  - "Chi-squared goodness-of-fit test"
  - "Pearson's chi-squared test"
  - "Exact McNemar's test"
  - "Boschloo's test"
  - "Wilcoxon's one-sample signed-rank test"
  - "Mann-Whitney's U test"

- "Kruskal-Wallis's one-way ANOVA"
- "Student's paired t-test"
- "Mixed model ANOVA"
- "Student's one-sample t-test"
- "Student's two-sample t-test"
- "Welch's two-sample t-test"
- "F-test (ANOVA)"

### Combining rows

The `reshape_rows` argument offers a framework for combining multiple rows of the output table into a single one. `reshape_rows` is a named list of lists. The names of its member-lists determine the name that will be displayed as the name of the combined summary stats in the table (e.g. "mean  $\pm$  sd "). The member lists need to contain two elements: `args`, contains the names of the summary statistics to be combined as characters, and `fun` which contains a function to combine these summary stats. The argument names of this function need to match the character strings specified in `args`. Check out the default options for an exemplary definition.

### References

[1] Hodges, J. L.; Lehmann, E. L. (1963). "Estimation of location based on ranks". *Annals of Mathematical Statistics*. 34 (2): 598-611. doi:10.1214/aoms/1177704172. JSTOR 2238406. MR 0152070. Zbl 0203.21105. PE euclid.aoms/1177704172

### Examples

```
descr(iris)
DescrList <- descr(iris)
DescrList$variables$results$Sepal.Length$Total$mean
print(DescrList)
descr(iris, "Species")
```

---

DescrTab2

*DescrTab2*

---

### Description

Publication quality descriptive statistics tables with R

### Details

Provides functions to create descriptive statistics tables for continuous and categorical variables. By default, summary statistics such as mean, standard deviation, quantiles, minimum and maximum for continuous variables and relative and absolute frequencies for categorical variables are calculated. 'DescrTab2' features a sophisticated algorithm to choose appropriate test statistics for your data and provides p-values. On top of this, confidence intervals for group differences of appropriated summary measures are automatically produces for two-group comparison. Tables generated by

'DescrTab2' can be integrated in a variety of document formats, including .html, .tex and .docx documents. 'DescrTab2' also allows printing tables to console and saving table objects for later use.

Check out our documentation online: <https://imbi-heidelberg.github.io/DescrTab2/> or browse the help files in the Rstudio viewer. You can access the vignettes by typing: `browseVignettes("DescrTab2")`

The most important function you probably want to check out is called [descr](#).

escape\_latex\_symbols *Escape LaTeX Symbols*

### Description

Escape LaTeX Symbols

### Usage

```
escape_latex_symbols(tibl, numEscapes = 1)
```

### Arguments

tibl	A tibble filled with characters
numEscapes	(logical) chooses between "\" and "\\\"#

### Value

a tibble with appropriately escape LaTeX code

extract\_labels *Extract the label attribute from data*

### Description

Extract the label attribute from data

### Usage

```
extract_labels(dat)
```

### Arguments

dat	data in the form of a <a href="#">list</a> , <a href="#">data.frame</a> or <a href="#">tibble</a> , or a vector
-----	---

### Value

list of labels

**Examples**

```
a <- c(1, 2)
attr(a, "label") <- "b"
identical(extract_labels(a), list(a = attr(a, "label")))
```

---

farrington.manning      *Farrington-Manning test for rate difference*

---

**Description**

The Farrington-Manning test for rate differences can be used to compare the rate difference of successes between two groups to a preset value. It uses an explicit formula for the standard deviation of the test statistic under the null hypothesis [1].

**Usage**

```
farrington.manning(
  group1,
  group2,
  delta = 0,
  alternative = "greater",
  alpha = 0.025
)
```

**Arguments**

group1	a logical vector of data from group 1, where TRUE indicates a success
group2	a logical vector of data from group 2, where TRUE indicates a success
delta	the rate difference under the null hypothesis
alternative	character string indicating the alternative to use, either of "two.sided", "less", "greater"
alpha	the significance level (acceptable error of the first kind), a two-sided confidence interval is returned with confidence level $1 - 2*\alpha$ , such that the lower bound is a valid one sided confidence interval at the confidence level $1 - \alpha$ .

**Details**

The Farrington-Manning test for rate differences test the null hypothesis of

$$H_0 : p_1 - p_2 = \delta$$

for the "two.sided" alternative (or  $\geq$  for the "greater" respectively  $\leq$  for the "less" alternative). This formulation allows to specify non-inferiority and superiority test in a consistent manner:

**non-inferiority** for  $\delta < 0$  and `alternative == "greater"` the null hypothesis reads  $H_0 : p_1 - p_2 \geq \delta$  and consequently rejection allows concluding that  $p_1 \geq p_2 + \delta$  i.e. that the rate of success in group one is at least the success rate in group two plus delta - as delta is negative this is equivalent to the success rate of group 1 being at worst delta smaller than that of group 2.

**superiority** for  $\delta \geq 0$  and `alternative == "greater"` the null hypothesis reads  $H_0 : p_1 - p_2 \geq \delta$  and consequently rejection allows concluding that  $p_1 \geq p_2 + \delta$  i.e. that the rate of success in group one is at least  $\delta$  greater than the success rate in group two.

The confidence interval is always computed as two-sided, but with  $1-2\alpha$  confidence level in case of a one-sided hypothesis. This means that the lower or upper bound are valid one-sided confidence bounds at level  $\alpha$  in this case. The confidence interval is constructed by inverting the two-sided test directly.

## Value

A list of class "htest" containing the following components:

<code>statistic:</code>	the value of the Z-statistic
<code>parameter:</code>	delta, rate difference (group 1 - group 2) under the null hypothesis
<code>p.value:</code>	the p-value for the Farrington-Manning test
<code>null.value:</code>	rate difference (group 1 - group 2) under the null
<code>alternative:</code>	a character string indicating the alternative hypothesis
<code>method:</code>	a character string indicating the exact method employed
<code>data.name:</code>	a character string giving the names of the data used
<code>estimate:</code>	the estimated rate difference (maximum likelihood)
<code>conf.int:</code>	a confidence interval for the rate difference
<code>sample.size:</code>	the total sample size used for the test

## Author(s)

Kevin Kunzmann

## References

[1] Farrington, Conor P., and Godfrey Manning. "Test statistics and sample size formulae for comparative binomial trials with null hypothesis of non-zero risk difference or non-unity relative risk." *Statistics in medicine* 9.12 (1990): 1447-1454.

## Examples

```
x <- c(rep(TRUE, 20), rep(FALSE, 15))
y <- c(rep(TRUE, 30), rep(FALSE, 25))

farrington.manning(x, y, -.3)
```

---

format\_freqs

*Formatting function for absolute and relative frequencies*

---

## Description

Formatting function for absolute and relative frequencies

**Usage**

```
format_freqs(
  numerator,
  denominator = 1,
  absolute_relative_frequency_mode = c("both", "only_absolute", "only_relative"),
  percent_accuracy = NULL,
  percent_suffix = "%"
)
```

**Arguments**

`numerator` (numeric) numerator for % calculations

`denominator` (numeric) denominator for % calculations

`absolute_relative_frequency_mode` one of `c("both", "only_absolute", "only_relative")`. "both" will print "Absolute Freq. (Relative Freq. %)", the other options work accordingly.

`percent_accuracy` NULL or numeric. Refer to the accuracy argument in [percent](#).

`percent_suffix` usually "%" or "". Refer to the suffix argument in [percent](#).

**Value**

string of formatted frequencies

---

<code>guess_ID_variable</code>	<i>Make an educated guess about the name of the ID variable from a dataset</i>
--------------------------------	--

---

**Description**

Make an educated guess about the name of the ID variable from a dataset

**Usage**

```
guess_ID_variable(dat, suppressWarnings = FALSE)
```

**Arguments**

`dat` a dataset with names (`list`, `data.frame`, `tibble`)

`suppressWarnings` (logical) suppress warning messages if you know what you are doing

**Value**

if exactly one possible

**Examples**

```
dat <- data.frame(ID = c(1,2,3,4,5),
                  other = c(1,2,3,4,5))
guess_ID_variable(dat)
```

---

ignore\_unused\_args      *do.call but without an error for unused arguments*

---

**Description**

do.call but without an error for unused arguments

**Usage**

```
ignore_unused_args(what, args)
```

**Arguments**

what	either a function or a non-empty character string naming the function to be called.
args	a list of arguments to the function call. The names attribute of args gives the argument names.

**Value**

The result of the (evaluated) function call.

**Examples**

```
# works:
DescrTab2:::ignore_unused_args(
  chisq.test,
  list(x = factor(c(1, 0, 1, 1, 1, 0)), y = factor(c(0, 1, 1, 0, 1, 0)), abc = 3)
)

# would produce error:
# do.call(chisq.test, list(x=factor(c(1,0,1,1,1,0)), y=factor(c(0,1,1,0,1,0)), abc=3 ) )
```



---

in_minipage	<i>Wrap cell text in minipage LaTeX environment with stretchy space</i>
-------------	---

---

**Description**

Wrap cell text in minipage LaTeX environment with stretchy space

**Usage**

```
in_minipage(text, width, numEscapes = 1, stretchSpace = FALSE)
```

**Arguments**

text	text to be placed in minipage
width	width adjustment
numEscapes	(logical) chooses between "\" and "\\\"#
stretchSpace	(logical) will add stretchy space

**Value**

appropriate LaTeX code

**References**

<https://stackoverflow.com/a/50892682>

---

knit_print.DescrList	<i>S3 override for knit_print function for DescrList objects.</i>
----------------------	---

---

**Description**

S3 override for knit\_print function for DescrList objects.

**Usage**

```
## S3 method for class 'DescrList'
knit_print(x, print_format = options("print_format")[[1]], silent = FALSE, ...)
```

**Arguments**

x	a
print_format	b
silent	c
...	abc

**Value**

outputs formatted table depending on the environment (.RMD) which it is called from

---

`knit_print.DescrPrint` *S3 override for knit\_print function for DescrPrint objects.*

---

**Description**

S3 override for knit\_print function for DescrPrint objects.

**Usage**

```
## S3 method for class 'DescrPrint'
knit_print(x, print_format = print_format, silent = silent, ...)
```

**Arguments**

<code>x</code>	<code>a</code>
<code>print_format</code>	<code>b</code>
<code>silent</code>	<code>c</code>
<code>...</code>	<code>abc</code>

**Value**

outputs formatted table depending on the environment (.RMD) which it is called from

---

<code>lapply_descr</code>	<i>Convenience function to apply descr to a list of datasets and print the results</i>
---------------------------	--

---

**Description**

Convenience function to apply descr to a list of datasets and print the results

**Usage**

```
lapply_descr(list, ...)
```

**Arguments**

<code>list</code>	a list of datasets (tibbles or data.frames)
<code>...</code>	arguments to be passed to the descr call

**Value**

something printable.

**Examples**

```
l <- list()
for (i in 1:2){
  l <- append(l, list(iris))
}
lapply_descr(l, group="Species")
```

---

list\_freetext\_markdown

*Create a markdown listing from a character dataset*

---

**Description**

Create a markdown listing from a character dataset

**Usage**

```
list_freetext_markdown(dat)
```

**Arguments**

dat                    a character data.frame or tibble.

**Value**

string containing markdown code listing all nonempty free text in the dataset

**Examples**

```
dat <- data.frame(Freetext = c("Some text", "More text"))
list_freetext_markdown(dat)
# use inside a .Rmd document like this:
# `r list_freetext_markdown(dat)`
```

---

n_int_digits	<i>Digits before decimal -1</i>
--------------	---------------------------------

---

**Description**

Digits before decimal -1

**Usage**

```
n_int_digits(x)
```

**Arguments**

x	a
---	---

**Details**

<https://stackoverflow.com/questions/47190693/count-the-number-of-integer-digits>

**Value**

a

---

parse_formats	<i>Parse a text file containing format information</i>
---------------	--

---

**Description**

Useful to extract factor formatting information contained in a proc format SAS statement.

**Usage**

```
parse_formats(
  path_to_format_definition,
  ignore_keywords = c("value"),
  encoding = "ISO-8859-1"
)
```

**Arguments**

path_to_format_definition	(string) Path to the text file to be parsed
ignore_keywords	A vector of keywords to be ignored when searching for the name of the variable to be formatted
encoding	Encoding for the text file

**Value**

A named list with format definitions

**Examples**

```
tmpfile <- tempfile()
write(  "proc format;
        value yn 1=\\"yes\\"
          0=\\"no\\";
        value sex 1=\\"female\\"
          0=\\"male\\";
        run;", tmpfile)
parse_formats(tmpfile)
```

---

print.DescrList      *S3 override for print function for DescrList objects.*

---

**Description**

This function takes a DescrList object and converts it to either a DescrPrintCharacter or DescrPrintNumeric object, depending on the print\_format option. This object is then printed in an appropriate format.

**Usage**

```
## S3 method for class 'DescrList'
print(x, print_format = options("print_format")[[1]], silent = FALSE, ...)
```

**Arguments**

x	A DescrList object returned from <a href="#">descr</a> .
print_format	Possible values: "console" (default), "tex", "html", "word", "numeric"
silent	I TRUE, suppresses output to stdout.
...	further arguments to be passed along to print method

**Details**

There is no way to convert between DescrPrintCharacter and DescrPrintNumeric objects. The first type is for what you would usually want, the second type is mostly for debugging purposes. A DescrPrintCharacter object can be printed as html, tex code, as a flextable object or simply to the console.

**Value**

A DescrPrint object which can be printed in various formats.

You can use the print\_format option to control the output type. If you use 'DescrTab2' inside an .Rmd document, you can set the global option option(print\_format="tex") or option(print\_format="html") or option(print\_format="word") depending on your document type. This way, all your tables will be printed in the right format by default inside this document.

**Examples**

```

print(descr(iris), print_format = "console")
print(descr(iris), print_format = "tex")
print(descr(iris), print_format = "html")
print(descr(iris), print_format = "word")
print(descr(iris), print_format = "numeric")
options(print_format = "tex")
descr(iris)
options(print_format = "console")
descr(iris)
DescrPrint <- print(descr(iris))
DescrPrint$variables$results$Sepal.Length$Total$mean
print(DescrPrint)

```

---

```

print_test_names      Prints all possible tests names

```

---

**Description**

Prints all possible tests names

**Usage**

```
print_test_names()
```

**Value**

Returns the names of all possible test names you can specify.

**Examples**

```
print_test_names()
```

---

```

read_redcap_formatted Convencience function to load datasets downloaded from a Redcap database

```

---

**Description**

This function is specifically tailored to the way the default import script provided by a Redcap database functions. First, the Hmisc package is loaded. The .csv file containing the data is assumed to be located in the current working directory. Labels are assigned to all variables. Variables which are supposed to be factors are twice, once as a factor and once in an unformatted way.

**Usage**

```
read_redcap_formatted(path_to_redcap_script = NULL)
```

**Arguments**

path\_to\_redcap\_script  
(character) Path to the (automatically generated) redcap script for data import

**Details**

This script removes the "unformatted factor" variables and properly assigns labels.

**Value**

tibble with data

**Examples**

```
path_to_redcap_script <- system.file("examples", "testredcap.r", package = "DescrTab2")
read_redcap_formatted(path_to_redcap_script)
```

---

read\_sas\_formatted      *Convenience function to load SAS datasets*

---

**Description**

Convenience function to load SAS datasets

**Usage**

```
read_sas_formatted(path_to_data = NULL, path_to_format = NULL)
```

**Arguments**

path\_to\_data    path to .sas7bdat file  
path\_to\_format   path to .sas7bcat file

**Value**

tibble with data

**Examples**

```
path_to_data <- system.file("examples", "testsas.sas7bdat", package = "DescrTab2")
pat_to_format <- system.file("examples", "formats.sas7bcat", package = "DescrTab2")
read_sas_formatted(path_to_data, pat_to_format)
```

---

sigfig	<i>Format number to a specified number of digits, considering threshold for usage of scientific notation</i>
--------	--

---

**Description**

Format number to a specified number of digits, considering threshold for usage of scientific notation

**Usage**

```
sigfig(
  x,
  digits = 3,
  scientific_high_threshold = 6,
  scientific_low_threshold = -6,
  force_0_behind_0 = FALSE
)
```

**Arguments**

x	a
digits	a
scientific_high_threshold	a
scientific_low_threshold	a
force_0_behind_0	a

**Value**

a

---

sigfig_gen	<i>Generator function for nice formatting functions</i>
------------	---

---

**Description**

Generator function for nice formatting functions



**Usage**

```
sigfig_gen(
  digits = 3,
  scientific_high_threshold = 6,
  scientific_low_threshold = -6,
  force_0_behind_0 = FALSE
)
```

**Arguments**

```
digits          a
scientific_high_threshold
                a
scientific_low_threshold
                a
force_0_behind_0
                a
```

---

sig\_test

*Calculates a statistical significance test*


---

**Description**

Calculates a statistical significance test

**Usage**

```
sig_test(
  var,
  group = NULL,
  test_options = list(),
  test = NULL,
  var_name = NULL
)
```

**Arguments**

```
var          A variable (a vector).
group        A variable containing the grouping information.
test_options Named list containing test options.
test         Name of a statistical test.
var_name     Name of variable to be tested (only used in warning messages).
```

**Value**

A list of test test results.

**Examples**

```
cont_var <- c(1, 2, 3)
sig_test(cont_var)
```

---

`split_redcap_dataset` *Split a dataset imported from Redcap into convenient subsets*

---

**Description**

This function separates a datasets into three parts: "Singular" data, which is the data from non-repeating instruments. "missings\_everywhere", which is data which is missing for each row. The last parts are all the repeating instruments, which are referred to by their name as recorded in `dat$redcap_repeat_instrument`.

**Usage**

```
split_redcap_dataset(dat, id_name = "patid")
```

**Arguments**

`dat` a tibble produced by [read\\_redcap\\_formatted](#).  
`id_name` (character) the name of the subject ID variable.

**Value**

a list of datasets separated into the categories as described

**Examples**

```
path_to_redcap_script <- system.file("examples", "testredcap.r", package = "DescrTab2")
dat <- read_redcap_formatted(path_to_redcap_script)
d <- split_redcap_dataset(dat, guess_ID_variable(dat, TRUE))
```

---

`unlabel` *Remove the label attribute from data*

---

**Description**

Remove the label attribute from data

**Usage**

```
unlabel(dat)
```

### Arguments

`dat` data in the form of a [list](#), [data.frame](#) or [tibble](#), or a vector

### Value

data with the labels removed

### Examples

```
a <- c(1, 2)
attr(a, "label") <- "b"
identical(unlabel(a), c(1, 2))
```

---

`write_in_tmpfile_for_cran`

*Function that returns true in CRAN submission*

---

### Description

Function that returns true in CRAN submission

### Usage

```
write_in_tmpfile_for_cran()
```

### Value

TRUE for CRAN submission, FALSE otherwise

# Index

- \* **Farrington-Manning**
  - farrington.manning, 13
- \* **rates**
  - farrington.manning, 13
- \* **test**
  - farrington.manning, 13
  - .onLoad, 3
- can.be.numeric, 3
- codegen\_load\_all\_sas\_data, 4
- create\_character\_subtable, 4
- create\_numeric\_subtable, 5
- data.frame, 12, 27
- descr, 6, 12, 21
- DescrTab2, 11
- escape\_latex\_symbols, 12
- extract\_labels, 12
- farrington.manning, 13
- format\_freqs, 14
- guess\_ID\_variable, 15
- ignore\_unused\_args, 16
- in\_minipage, 17
- kableExtra, 3
- knit\_print.DescrList, 17
- knit\_print.DescrPrint, 18
- lapply\_descr, 18
- list, 12, 27
- list\_freetext\_markdown, 19
- n\_int\_digits, 20
- parse\_formats, 20
- percent, 15
- print.DescrList, 21
- print\_test\_names, 22
- prop.test, 6
- read\_redcap\_formatted, 22, 26
- read\_sas\_formatted, 23
- sig\_test, 25
- sigfig, 24
- sigfig\_gen, 24
- split\_redcap\_dataset, 26
- t.test, 6
- tibble, 12, 27
- unlabel, 26
- wilcox.test, 6
- write\_in\_tmpfile\_for\_cran, 27