

Package: freqtables (via r-universe)

September 19, 2026

Type Package

Title Make Quick Descriptive Tables for Categorical Variables

Description Quickly make tables of descriptive statistics (i.e., counts, percentages, confidence intervals) for categorical variables. This package is designed to work in a Tidyverse pipeline, and consideration has been given to get results from R to Microsoft Word ® with minimal pain.

Version 0.2.0.9000

Maintainer Brad Cannell <brad.cannell@gmail.com>

URL <https://github.com/brad-cannell/freqtables>

BugReports <https://github.com/brad-cannell/freqtables/issues>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Imports dplyr (>= 1.0.0), rlang, stringr

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 4.1.0)

Config/testthat/edition 3

LazyData true

Config/pak/sysreqs libicu-dev

Repository <https://brad-cannell.r-universe.dev>

Date/Publication 2026-09-12 05:19:59 UTC

RemoteUrl <https://github.com/brad-cannell/freqtables>

RemoteRef HEAD

RemoteSha f3f9954207da5d4047ff81e300290bcf1da92a1f

Contents

confidence_intervals	2
freq_format	3
freq_study	4
freq_table	5
freq_tbl	6
freq_test	7
freq_xtab	9
get_group_n	10
Index	11

confidence_intervals	<i>Add confidence intervals to frequency counts</i>
----------------------	---

Description

These helpers add intervals to `freq_tbl()` output, and are also used by `freq_table()`. Both retain the package’s Student-t convention: $SE = \sqrt{p * (1 - p) / (N - 1)}$, with $N - 1$ degrees of freedom. For within-group intervals, N is that group’s denominator; for overall intervals, N is the total input size. This is a compatibility convention, not the usual normal-z binomial Wald interval.

Usage

```
wald_ci(.data, percent_ci = 95, se = FALSE, critical_value = FALSE)

logit_ci(.data, percent_ci = 95, se = FALSE, critical_value = FALSE)
```

Arguments

.data	A frequency table from <code>freq_tbl()</code> or <code>freq_table()</code> . A plain data frame is also accepted if it contains <code>n</code> and <code>n_group</code> with <code>prop_group/percent_group</code> , or <code>n_total</code> with <code>prop/percent</code> . Optional overall columns use <code>n_total</code> with <code>prop_total/percent_total</code> . An ungrouped compact table needs its original <code>freqtables</code> metadata or an explicit <code>n_total</code> ; totals are never reconstructed by summing a possibly filtered table. Counts must be finite nonnegative integers no larger than their denominators. Displayed proportions must agree with those counts.
percent_ci	Confidence level as a percentage, strictly between 0 and 100; default 95. Independent of the table’s proportion/percentage scale.
se	Logical. Include standard errors? Default FALSE.
critical_value	Logical. Include Student-t critical values in <code>t_crit</code> , <code>t_crit_group</code> , and/or <code>t_crit_total</code> ? Default FALSE.

Details

wald_ci() returns $p \pm t \times SE$ without clipping to the unit interval. logit_ci() transforms $\log(p / (1 - p)) \pm t \times SE / (p \times (1 - p))$ back to the proportion scale with `stats::plogis()`. At $p = 0$ or 1 , logit bounds are missing (no continuity correction); Wald bounds equal p . With $N \leq 1$, SE, critical values, and both types of bounds are missing.

Value

The input table with `lcl/ucl`, `lcl_group/ucl_group`, and/or `lcl_total/ucl_total` for each proportion present. Bounds and standard errors use the input scale; critical values do not scale. Repeated calls replace interval/detail columns for the same proportions, including removal of unrequested optional detail columns.

Examples

```
freq_tbl(mtcars, am) |> wald_ci()
freq_tbl(mtcars, am) |> logit_ci(percent_ci = 99, se = TRUE)
mtcars |> dplyr::group_by(cyl) |>
  freq_tbl(am, percent = TRUE, overall = TRUE) |>
  logit_ci(critical_value = TRUE)
```

freq_format

Format statistics for presentation

Description

Append a character column composed from existing table columns and literal text. Works with compact proportions, percentages, grouped results, and `freq_test()` output. Calculations retain full precision; rounding happens only in the new display column.

Usage

```
freq_format(.data, recipe, name = NA, digits = NA)
```

Arguments

<code>.data</code>	A data frame, commonly from <code>freq_tbl()</code> or <code>freq_table()</code> .
<code>recipe</code>	One quoted string containing column-name tokens and literal text, e.g. <code>"n (percent%)"</code> or <code>"prop_group (lcl_group - ucl_group)"</code> . Tokens use word boundaries; names containing spaces/punctuation should be renamed first. Unmatched tokens remain literal text.
<code>name</code>	Name of the appended column. Default NA selects <code>"formatted_stats"</code> . An existing column with this name is replaced.
<code>digits</code>	Nonnegative integer number of decimal places. Default NA uses character conversion without explicit rounding. When supplied, doubles are rounded with trailing zeros and thousands separators; integer counts retain their integer representation. Character categories are not rounded.

Value

The input data frame with one character column added or replaced.

Examples

```
freq_table(mtcars, am, percent = TRUE) |>
  freq_format("percent (lcl - ucl)", name = "percent_ci", digits = 2)
mtcars |> dplyr::group_by(cyl) |> freq_table(am) |>
  freq_format("n: prop_group (lcl_group - ucl_group)", digits = 3)
```

freq_study

Simulated study data.

Description

This is the code to create the study data - a simulated dataset that can be used to demonstrate how to use the freqtables package.

Usage

```
freq_study
```

Format

A data frame with 100 rows and 6 variables:

id Participant's study identification number

age_group Participant's age at the time of enrollment categorized as "Younger than 30" or "30 and Older"

sex Biological sex of the participant assigned at birth, female/male

bmi_4cat Participant's bmi at date of enrollment categorized as "Underweight", "Normal", "Overweight", or "Obese"

exposure Participant was exposed to a hypothetical risk factor of interest, Yes/No

outcome Participant experienced a hypothetical outcome of interest, Yes/No

freq_table

*Frequency tables with confidence intervals***Description**

Combine `freq_tbl()` with `logit_ci()` or `wald_ci()`. Accept one outcome; supply groups using `dplyr::group_by()`. Defaults are compact proportions with confidence intervals. See `vignette("migration")` for the transition from the old percentage-scale, two-variable interface.

Usage

```
freq_table(
  .data,
  .freq_var,
  ...,
  percent_ci = 95,
  ci_type = "logit",
  drop = FALSE,
  percent = FALSE,
  overall = FALSE,
  se = FALSE,
  critical_value = FALSE,
  generic_col_names = FALSE
)
```

Arguments

<code>.data</code>	A local data frame or grouped data frame. Rowwise data is not supported; call <code>dplyr::ungroup()</code> first.
<code>.freq_var</code>	One unquoted column name (or a string naming a column). In a wrapper function, use <code>{{ variable }}</code> .
<code>...</code>	Reserved for detecting obsolete multiple-outcome calls and misspelled options. Must be empty; options must be named.
<code>percent_ci</code>	Confidence level as a percentage, strictly between 0 and 100; default 95. Independent of the table's proportion/percentage scale.
<code>ci_type</code>	Interval method: "logit" (default) or "wald". Both methods support ungrouped and grouped data. Supply a string.
<code>drop</code>	Logical. If FALSE (default), retain unobserved factor levels and empty factor groups with zero counts. If TRUE, retain observed combinations only. Does not remove missing values.
<code>percent</code>	Logical. If FALSE, return proportions in <code>prop</code> , <code>prop_group</code> , and (when requested) <code>prop_total</code> . If TRUE, use <code>percent</code> , <code>percent_group</code> , and <code>percent_total</code> on the 0–100 scale.

overall	Logical. For grouped data, append <code>n_total</code> and <code>prop_total</code> (or <code>percent_total</code>): the cell count divided by the total input rows, not the outcome's marginal proportion. For ungrouped data, append only <code>n_total</code> ; <code>prop</code> is already the overall proportion.
se	Logical. Include standard errors? Default FALSE.
critical_value	Logical. Include Student-t critical values in <code>t_crit</code> , <code>t_crit_group</code> , and/or <code>t_crit_total</code> ? Default FALSE.
generic_col_names	Logical. If TRUE, replace each group column with <code>group_01_col/group_01_cat</code> , etc., and the outcome with <code>col/cat</code> . Category columns become character for stacking tables.

Value

An ungrouped tibble with the `freq_tbl()` columns plus confidence limits. Classes are `freq_table_one_way` (no groups), `freq_table_two_way` (one group), or `freq_table_n_way` (multiple groups), together with `freq_tbl` and `tibble` classes. `freq_test()` supports the first two cases; multiple grouping variables do not imply a particular statistical test.

Examples

```
freq_table(mtcars, am)
mtcars |> dplyr::group_by(cyl) |> freq_table(am)
mtcars |> dplyr::group_by(cyl, vs) |>
  freq_table(am, ci_type = "wald", percent_ci = 90)
freq_table(mtcars, am, percent = TRUE, overall = TRUE,
  se = TRUE, critical_value = TRUE, generic_col_names = TRUE)
```

freq_tbl	<i>Compact counts and proportions</i>
----------	---------------------------------------

Description

Count one outcome within zero, one, or multiple groups supplied by `dplyr::group_by()`. Each observation contributes once, including missing outcomes and missing group values. Use `dplyr::filter()` before counting to exclude missing observations explicitly.

Usage

```
freq_tbl(
  .data,
  .freq_var,
  ...,
  drop = FALSE,
  percent = FALSE,
  overall = FALSE,
  generic_col_names = FALSE
)
```

Arguments

<code>.data</code>	A local data frame or grouped data frame. Rowwise data is not supported; call <code>dplyr::ungroup()</code> first.
<code>.freq_var</code>	One unquoted column name (or a string naming a column). In a wrapper function, use <code>{{ variable }}</code> .
<code>...</code>	Reserved for detecting obsolete multiple-outcome calls and misspelled options. Must be empty; options must be named.
<code>drop</code>	Logical. If FALSE (default), retain unobserved factor levels and empty factor groups with zero counts. If TRUE, retain observed combinations only. Does not remove missing values.
<code>percent</code>	Logical. If FALSE, return proportions in <code>prop</code> , <code>prop_group</code> , and (when requested) <code>prop_total</code> . If TRUE, use <code>percent</code> , <code>percent_group</code> , and <code>percent_total</code> on the 0–100 scale.
<code>overall</code>	Logical. For grouped data, append <code>n_total</code> and <code>prop_total</code> (or <code>percent_total</code>): the cell count divided by the total input rows, not the outcome's marginal proportion. For ungrouped data, append only <code>n_total</code> ; <code>prop</code> is already the overall proportion.
<code>generic_col_names</code>	Logical. If TRUE, replace each group column with <code>group_01_col/group_01_cat</code> , etc., and the outcome with <code>col/cat</code> . Category columns become character for stacking tables.

Value

An ungrouped tibble of class `freq_tbl`. Ungrouped defaults are `outcome`, `n`, `prop`; grouped defaults are `group columns`, `outcome`, `n`, `n_group`, `prop_group`. `n_group` is the total number of observations in that group, across all outcome categories. Zero denominators produce `NA_real_` proportions. Variable types are preserved unless generic headers are requested. Metadata retains the original total for subsequent confidence-interval calculations even after filtering rows. Analysis column names that collide with generated columns are rejected; rename them before calling.

Examples

```
freq_tbl(mtcars, am)
mtcars |> dplyr::group_by(cyl) |> freq_tbl(am)
mtcars |> dplyr::group_by(cyl, vs) |> freq_tbl(am, percent = TRUE)
freq_tbl(mtcars, am, generic_col_names = TRUE, overall = TRUE)
```

Description

Test equal category probabilities for an ungrouped table, or independence between one grouping variable and the outcome for a two-way table. Pearson's statistic is uncorrected (no Yates correction). For a two-way table with any expected count ≤ 5 , also return Fisher's exact p-value and emit a message. The table is constructed from category identities, with omitted cells filled with zero, so row order and non-binary groups work.

Usage

```
freq_test(.data, ...)

## S3 method for class 'freq_table_one_way'
freq_test(.data, ...)

## S3 method for class 'freq_table_two_way'
freq_test(.data, ...)

## Default S3 method:
freq_test(.data, ...)
```

Arguments

<code>.data</code>	A complete <code>freq_table_one_way</code> or <code>freq_table_two_way</code> result from <code>freq_table()</code> , with its metadata and category/count columns intact. Missing values count as categories unless filtered before making the table. Tables stacked from different outcomes are not supported. Two-way tests require at least two observed categories on each axis and no zero marginal totals; use <code>drop = TRUE</code> to omit empty factor levels.
<code>...</code>	Reserved; no extra arguments are currently supported by either method. Nonempty <code>...</code> produces an error rather than being ignored.

Value

The input with expected counts, per-cell chi-square contributions, the overall Pearson statistic, degrees of freedom, and `p_chi2_pearson`. Two-way output also has `n_col`, `r`, `c`, and (when expected counts are small) `p_fisher`. Omitted zero-count cells contribute to the statistic even though they are absent from the output. Fisher's exact calculation can fail for very large tables; the error from `stats::fisher.test()` is propagated.

Examples

```
freq_table(mtcars, am) |> freq_test()
mtcars |> dplyr::group_by(am) |> freq_table(cyl) |> freq_test()
mtcars |> dplyr::group_by(cyl) |>
  freq_table(am, generic_col_names = TRUE) |> freq_test()
```

freq_xtab

*Contingency count matrices and grouped arrays***Description**

Cross-tabulate two columns, with exposure in rows and outcome in columns. Groups supplied by `dplyr::group_by()` become additional array dimensions in the supplied order. Every input row contributes once, including missing values; filter observations explicitly before counting to exclude them.

Usage

```
freq_xtab(.data, exposure, outcome, ..., margins = TRUE, drop = FALSE)
```

Arguments

<code>.data</code>	A local data frame or grouped data frame. Rowwise data is not supported; call <code>dplyr::ungroup()</code> first.
<code>exposure, outcome</code>	Two distinct unquoted column names, or strings naming columns. Forward columns in wrappers with <code>{{ }}</code> . Neither column may also be a grouping variable. Columns must be one-dimensional atomic vectors (for example factors, character, numeric, logical, or Date vectors).
<code>...</code>	Must be empty. Options must be named; derived columns should be created with <code>dplyr::mutate()</code> before calling.
<code>margins</code>	Logical. If TRUE (default), append a total row and column to each matrix, labeled Sum. If that label already exists on an axis, use the first available Sum.1, Sum.2, etc. Group dimensions have no margins. Use FALSE to obtain only the cell counts for further analysis.
<code>drop</code>	Logical. If FALSE (default), retain unused factor levels on every axis. If TRUE, remove globally unused levels, overriding the input's grouping drop setting. Both settings fill absent combinations with zero so every group has the same matrix dimensions. Missing values are retained independently of this option.

Value

A numeric matrix for ungrouped input, or a numeric array with dimensions exposure, outcome, then each grouping variable. Dimension names are the original column names. Factor categories follow level order; other categories are sorted (character values in C-locale order), with missing values last. Labels are character representations of categories; ambiguous representations of distinct values are rejected. Missing labels are actual `NA_character_`, distinct from literal strings such as "NA". Select a missing category by position using `which(is.na(dimnames(x)[[i]]))`.

Unobserved combinations, including empty group combinations, have zero counts. Empty input retains factor levels when `drop = FALSE`; other axes have length zero before margins. Thus empty ungrouped character input is a 1-by-1 zero matrix with margins, or a 0-by-0 matrix without. A grouped result with a zero-length grouping axis has no matrix slices. Singleton dimensions are retained in the returned object; use `drop = FALSE` when subsetting if those dimensions should remain.

Examples

```
freq_xtab(mtcars, vs, am)
freq_xtab(mtcars, "vs", "am", margins = FALSE)
mtcars |> dplyr::group_by(cyl) |> freq_xtab(vs, am)
mtcars |> dplyr::group_by(cyl, gear) |> freq_xtab(vs, am, drop = TRUE)
```

get_group_n

Formatted Group Sample Size for Tables

Description

Given a tibble and a filter expression, `get_group_n` returns the group sample size formatted as "N = XXXX". Made to work in a dplyr pipeline, and used when creating tables for publications / reports.

Usage

```
get_group_n(.data, ...)
```

Arguments

<code>.data</code>	A local data frame or tibble, optionally grouped. Grouped input returns one label per retained group, in dplyr group order.
<code>...</code>	Zero or more <code>dplyr::filter</code> expressions, combined with AND. With no expressions, count all input rows. Missing logical results are excluded by <code>filter()</code> . Expressions may use dplyr data masking.

Value

A character vector of formatted counts. Ungrouped input returns one label, including "N = 0" when no observations remain.

Examples

```
library(dplyr)
library(freqtables)

data(mtcars)

# Get sample size for cars with 4 cylinders
mtcars %>% get_group_n(cyl == 4)

#> [1] "N = 11"
```

Index

* datasets

freq_study, 4

confidence_intervals, 2

dplyr::filter(), 6

dplyr::group_by(), 5, 6, 9

dplyr::mutate(), 9

dplyr::ungroup(), 5, 7, 9

freq_format, 3

freq_study, 4

freq_table, 5

freq_table(), 2, 3, 8

freq_tbl, 6

freq_tbl(), 2, 3, 5, 6

freq_test, 7

freq_test(), 3, 6

freq_xtab, 9

get_group_n, 10

logit_ci(confidence_intervals), 2

logit_ci(), 5

stats::fisher.test(), 8

stats::plogis(), 3

wald_ci(confidence_intervals), 2

wald_ci(), 5