Package ‘freqtables’

October 13, 2022

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 0.1.1

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 7.1.2

Imports dplyr, rlang, stringr

Suggests testthat (>= 2.1.0), knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Brad Cannell [aut, cre]

Repository CRAN

Date/Publication 2022-04-03 15:20:02 UTC

R topics documented:

freq_format *Format freq_table Output for Publication and Dissemination*

Description

The freq_format function is intended to make it quick and easy to format the output of the freq_table function for tables that may be used for publication. For example, a proportion and 95 could be formatted as "24.00 (21.00 - 27.00)."

Usage

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

Arguments

`.data`	A data frame of class "freq_table_one_way" or "freq_table_two_way".
`recipe`	A recipe used to create a new column from existing freq_table columns. The recipe must be in the form of a quoted string. It may contain any combination of column names, spaces, and characters. For example: "n (percent)" or "percent (lcl - ucl)".
`name`	An optional name to assign to the column created by the recipe. The default name is "formatted_stats"
`digits`	The number of decimal places to display.

Value

A tibble

Examples

```
library(dplyr)
library(freqtables)

data(mtcars)

# One-way frequency tables with defaults

mtcars %>%
  freq_table(am) %>%
  freq_format(
    recipe = "percent (lcl - ucl)",
    name = "percent_95",
    digits = 2
  ) %>%
  select(var, cat, percent_95)
#> # A tibble: 2 x 3
#>   var   cat   percent_95
#>   <chr> <chr> <chr>
#> 1 am    0     59.38 (40.94 - 75.50)
#> 2 am    1     40.62 (24.50 - 59.06)

# Two-way frequency tables with defaults

mtcars %>%
  freq_table(am, cyl) %>%
    freq_format(
    recipe = ”percent_row (lcl_row - ucl_row)”,
    name = ”percent_95”,
    digits = 2
  ) %>%
  select(1:4, percent_95)
#> # A tibble: 6 x 5
#>   row_var row_cat col_var col_cat percent_95
#>   <chr>   <chr>   <chr>   <chr>   <chr>
#> 1 am      0       cyl     4       15.79 (4.78 - 41.20)
#> 2 am      0       cyl     6       21.05 (7.58 - 46.44)
#> 3 am      0       cyl     8       63.16 (38.76 - 82.28)
#> 4 am      1       cyl     4       61.54 (32.30 - 84.29)
#> 5 am      1       cyl     6       23.08 (6.91 - 54.82)
#> 6 am      1       cyl     8       15.38 (3.43 - 48.18)
```

`freq_table` *Estimate Counts, Percentages, and Confidence Intervals in dplyr Pipelines*

Description

The freq_table function produces one-way and two-way frequency tables for categorical variables. In addition to frequencies, the freq_table function displays percentages, and the standard errors and confidence intervals of the percentages. For two-way tables only, freq_table also displays row (subgroup) percentages, standard errors, and confidence intervals.

freq_table is intended to be used in a dplyr pipeline.

All standard errors are calculated as some version of: sqrt(proportion * (1 - proportion) / (n - 1))

For one-way tables, the default 95 percent confidence intervals displayed are logit transformed confidence intervals equivalent to those used by Stata. Additionally, freq_table will return Wald ("linear") confidence intervals if the argument to ci_type = "wald".

For two-way tables, freq_table returns logit transformed confidence intervals equivalent to those used by Stata.

Usage

```
freq_table(.data, ..., percent_ci = 95, ci_type = ”logit”, drop = FALSE)
```

Arguments

`.data` A data frame. If it is already grouped (i.e., class == "grouped_df") then freq_table will ungroup it to prevent unexpected results.

For two-way tables, the count for each level of the variable in the first argument to freq_table will be the denominator for row percentages and their confidence intervals. Said another way, the goal of the analysis is to compare percentages of some characteristic across two or more groups of interest, then the variable in the first argument to freq_table should contain the groups of interest, and the variable in the second argument to freq_table should contain the characteristic of interest.

`...` Categorical variables to be used in calculations. Currently, freq_table accepts one or two variables – not more.

By default, if ... includes a factor variable with a level (category) that is unobserved in the data, that level will still appear in the results with a count (n) equal to zero. This behavior can be changed using the drop parameter (see below). When n = 0, the confidence intervals will be NaN.

`percent_ci` sets the level, as a percentage, for confidence intervals. The default is percent_ci = 95 for 95 percentage value entered (e.g., 95) is converted to an alpha level as 1 - (percent_ci / 100). It is then converted to a two-sided probability as (1 - alpha / 2), which is used to calculate a critical value from Student's t distribution with n - 1 degrees of freedom.

`ci_type` Selects the method used to estimate 95 percent confidence intervals. The default for one-way and two-way tables is logit transformed ("log"). For one-way tables only, ci_type can optionally calculate Wald ("linear") confidence intervals using the "wald" argument.

`drop` If false (default) unobserved factor levels will be included in the returned frequency table with an n of 0. For example, if you have a factor variable, gender, but no males in your data then frequency table returned by freq_table(df, gender) would still contain a row for males with the variable n = 0. If drop is set to TRUE, then the resulting frequency table would not include a row for males at all.

Value

A tibble with class "freq_table_one_way" or "freq_table_two_way"

References

Agresti, A. (2012). Categorical Data Analysis (3rd ed.). Hoboken, NJ: Wiley.

SAS confidence limits for proportions documentation

Stata confidence limits for proportions documentation

Examples

```
library(dplyr)
library(freqtables)
```

```
data(mtcars)

# -----------------------------------------------------------------------------
# One-way frequency table with defaults
#   - The default confidence intervals are logit transformed - matching the
#     method used by Stata
# -----------------------------------------------------------------------------
mtcars %>%
  freq_table(am)

#   A tibble: 2 x 9
#   var   cat       n n_total percent    se t_crit   lcl   ucl
#   <chr> <chr> <int>   <int>   <dbl> <dbl>  <dbl> <dbl> <dbl>
# 1 am    0        19      32    59.4  8.82   2.04  40.9  75.5
# 2 am    1        13      32    40.6  8.82   2.04  24.5  59.1


# -----------------------------------------------------------------------------
# One-way frequency table with arbitrary cconfidence intervals
#   - The default behavior of freq_table is to return 95% confidence
#     intervals (two-sided). However, this behavior can be adjusted to return
#     any alpha level. For example, to return 99% confidence intervals just
#     pass 99 to the percent_ci parameter of freq_table as demonstrated below.
# -----------------------------------------------------------------------------
mtcars %>%
  freq_table(am, percent_ci = 99)

#   A tibble: 2 x 9
#   var   cat       n n_total percent    se t_crit   lcl   ucl
#   <chr> <chr> <int>   <int>   <dbl> <dbl>  <dbl> <dbl> <dbl>
# 1 am    0        19      32    59.4  8.82   2.74  34.9  79.9
# 2 am    1        13      32    40.6  8.82   2.74  20.1  65.1


# -----------------------------------------------------------------------------
# One-way frequency table with Wald confidence intervals
# Optionally, the ci_type = ”wald” argument can be used to calculate Wald
# confidence intervals that match those returned by SAS.
# -----------------------------------------------------------------------------
mtcars %>%
  freq_table(am, ci_type = ”wald”)

#   A tibble: 2 x 9
#   var   cat       n n_total percent    se t_crit   lcl   ucl
#   <chr> <chr> <int>   <int>   <dbl> <dbl>  <dbl> <dbl> <dbl>
# 1 am    0        19      32    59.4  8.82   2.04  41.4  77.4
# 2 am    1        13      32    40.6  8.82   2.04  22.6  58.6


# -----------------------------------------------------------------------------
# One-way frequency table with drop = FALSE (default)
# -----------------------------------------------------------------------------
df <- data.frame(
  id = c(1, 2, 3, 4),
  gender = factor(
    # All females
    c(1, 1, 1, 1),
    levels = c(1, 2),
    labels = c("female", "male"))
)

df %>%
  freq_table(gender)

#   A tibble: 2 x 9
#   var    cat        n n_total percent    se t_crit   lcl   ucl
#   <chr>  <chr>  <int>   <int>   <dbl> <dbl>  <dbl> <dbl> <dbl>
# 1 gender female     4       4     100     0   3.18   NaN   NaN
# 2 gender male       0       4       0     0   3.18   NaN   NaN


# -----------------------------------------------------------------------------
# One-way frequency table with drop = TRUE
# -----------------------------------------------------------------------------
df <- data.frame(
  id = factor(rep(1:3, each = 4)),
  period = factor(rep(1:4)),
  x = factor(c(0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1))
)

# Now, supppose we want to drop period 3 & 4 from our analysis.
# By default, this will give us 0s for period 3 & 4, but we want to drop them.

df <- df %>%
  filter(period %in% c(1, 2))

df %>%
  freq_table(period)

#   A tibble: 4 x 9
#   var    cat       n n_total percent    se t_crit    lcl   ucl
#   <chr>  <chr> <int>   <int>   <dbl> <dbl>  <dbl>  <dbl> <dbl>
# 1 period 1         3       6      50  22.4   2.57   9.12  90.9
# 2 period 2         3       6      50  22.4   2.57   9.12  90.9
# 3 period 3         0       6       0   0     2.57 NaN    NaN
# 4 period 4         0       6       0   0     2.57 NaN    NaN

# But, we don't want period 3 & 4 in our frequency table at all. That's
# when we should change drop to TRUE.

df %>%
  freq_table(period, drop = TRUE)

#   A tibble: 4 x 9
#   var    cat       n n_total percent    se t_crit    lcl   ucl
#   <chr>  <chr> <int>   <int>   <dbl> <dbl>  <dbl>  <dbl> <dbl>
# 1 period 1         3       6      50  22.4   2.57   9.12  90.9
# 2 period 2         3       6      50  22.4   2.57   9.12  90.9


# -----------------------------------------------------------------------------
# Two-way frequency table with defaults
# Output truncated to fit the screen
# -----------------------------------------------------------------------------
mtcars %>%
  freq_table(am, cyl)

#   A tibble: 6 x 17
#   row_var row_cat col_var col_cat     n n_row n_total percent_total se_total
#   <chr>   <chr>   <chr>   <chr>   <int> <int>   <int>         <dbl>    <dbl>
# 1 am      0       cyl     4           3    19      32          9.38     5.24
# 2 am      0       cyl     6           4    19      32         12.5      5.94
# 3 am      0       cyl     8          12    19      32         37.5      8.70
# 4 am      1       cyl     4           8    13      32         25        7.78
# 5 am      1       cyl     6           3    13      32          9.38     5.24
# 6 am      1       cyl     8           2    13      32          6.25     4.35
```

freq_test *Hypothesis Testing for Frequency Tables*

Description

The freq_test function is an S3 generic. It currently has methods for conducting hypothesis tests on one-way and two-way frequency tables. Further, it is made to work in a dplyr pipeline with the freq_table function.

For the freq_table_two_way class, the methods used are Pearson's chi-square test of independence Fisher's exact test. When cell counts are <= 5, Fisher's Exact Test is considered more reliable.

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, ...)
```

Arguments

`.data`	A tibble of class freq_table_one_way or freq_table_two_way.
`...`	Other parameters to be passed on.
`method`	Options for this parameter control the method used to calculate p-values.

Value

A tibble.

Examples

```
library(dplyr)
library(freqtables)

data(mtcars)

# Test equality of proportions

mtcars %>%
  freq_table(am) %>%
  freq_test() %>%
  select(var:percent, p_chi2_pearson)

#>  # A tibble: 2 x 6
#>      var   cat     n n_total percent p_chi2_pearson
#>    <chr> <dbl> <int>   <int>   <dbl>          <dbl>
#>  1    am     0    19      32   59.38      0.2888444
#>  2    am     1    13      32   40.62      0.2888444

# Chi-square test of independence

mtcars %>%
  freq_table(am, vs) %>%
  freq_test() %>%
  select(row_var:n, percent_row, p_chi2_pearson)

#> # A tibble: 4 x 7
#>   row_var row_cat col_var col_cat     n percent_row p_chi2_pearson
#>     <chr>   <dbl>   <chr>   <dbl> <int>       <dbl>          <dbl>
#> 1      am       0      vs       0    12       63.16      0.3409429
#> 2      am       0      vs       1     7       36.84      0.3409429
#> 3      am       1      vs       0     6       46.15      0.3409429
#> 4      am       1      vs       1     7       53.85      0.3409429
```

`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

`.data`	A data frame or tibble
`...`	A dplyr::filter expression. Used to select subgroup.

Value

A character string

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