Package 'spicy'

May 9, 2026

Title Descriptive Statistics, Summary Tables, and Data Management Tools

Version 0.11.0

Description Provides tools for descriptive data analysis, variable inspection, data management, and tabulation workflows in 'R'. Summarizes variable metadata, labels, classes, missing values, and representative values, with support for readable frequency tables, cross-tabulations, association measures for contingency tables (Cramer's V, Phi, Goodman-Kruskal Gamma, Kendall's Tau-b, Somers' D, and others), categorical and continuous summary tables, and model-based bivariate tables for continuous outcomes, including APA-style reporting outputs. Includes helpers for interactive codebooks, variable label extraction, clipboard export, and row-wise descriptive summaries. Designed to make descriptive analysis and reporting faster, clearer, and easier to work with in practice.

License MIT + file LICENSE

URL https://github.com/amaltawfik/spicy/, https://amaltawfik.github.io/spicy/

BugReports https://github.com/amaltawfik/spicy/issues

Encoding UTF-8

Language en-US

Imports crayon, dplyr, labelled, rlang (>= 1.1.0), sandwich, stats, stringr, tibble, tidyselect, utils

Suggests broom, clipr, clubSandwich, DT, effectsize, flextable, gt, haven, knitr, officer, openxlsx2, rmarkdown, testthat (>= 3.0.0), tinytable, withr

VignetteBuilder knitr

Depends R (>= 4.1.0)

Config/testthat/edition 3

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Amal Tawfik [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0006-2422-1555>, ROR:
<https://ror.org/04j47fz63>)

Maintainer Amal Tawfik <amal.tawfik@hesav.ch>

Repository CRAN

Date/Publication 2026-05-04 07:00:02 UTC

Contents

`assoc_measures` *Association measures summary table*

Description

`assoc_measures()` computes a range of association measures for a two-way contingency table and returns them in a tidy data frame.

Usage

```
assoc_measures(
  x,
  type = c("all", "nominal", "ordinal"),
  conf_level = 0.95,
  digits = 3L
)
```

Arguments

`x`	A contingency table (of class `table`).
`type`	Which family of measures to compute: `"all"` (default), `"nominal"`, or `"ordinal"`.
`conf_level`	A number between 0 and 1 giving the confidence level (default `0.95`). Set to `NULL` to omit the confidence interval.
`digits`	Number of decimal places used when printing the result (default `3`).

Details

`type = "all"` (the default) returns all nominal and ordinal measures. Use `type = "nominal"` or `type = "ordinal"` to restrict the output to a single family.

The nominal family includes `cramer_v()`, `contingency_coef()`, `lambda_gk()`, `goodman_kruskal_tau()`, `uncertainty_coef()`, and (for 2x2 tables) `phi()` and `yule_q()`.

The ordinal family includes `gamma_gk()`, `kendall_tau_b()`, `kendall_tau_c()`, and `somers_d()`.

Standard error formulas follow the DescTools implementations (Signorell et al., 2024).

Value

A data frame with columns `measure`, `estimate`, `se`, `ci_lower`, `ci_upper`, and `p_value`. For nominal measures (Cramer's V, Phi, Contingency Coef.), the p-value comes from the Pearson chi-squared test of independence. For all other measures, it is a Wald z-test of H0: measure = 0.

References

Agresti, A. (2002). *Categorical Data Analysis* (2nd ed.). Wiley.

Liebetrau, A. M. (1983). *Measures of Association*. Sage.

Signorell, A. et al. (2024). *DescTools: Tools for Descriptive Statistics*. R package.

See Also

cramer_v(), gamma_gk(), kendall_tau_b()

Other association measures: contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
assoc_measures(tab)
assoc_measures(tab, type = "nominal")
assoc_measures(tab, type = "ordinal")
```

---

code_book *Generate an interactive variable codebook*

Description

code_book() creates an interactive and exportable codebook summarizing selected variables of a data frame. It builds upon varlist() to provide an overview of variable names, labels, classes, and representative values in a sortable, searchable table.

The output is displayed as an interactive DT::datatable() in the Viewer pane (for example in RStudio or Positron), allowing searching, sorting, and export (copy, print, CSV, Excel, PDF) directly.

Usage

```
code_book(
  x,
  ...,
  values = FALSE,
  include_na = FALSE,
  title = "Codebook",
  filename = NULL,
  factor_levels = c("all", "observed")
)
```

Arguments

x	A data frame or tibble.
...	Optional tidyselect-style column selectors (e.g. starts_with("var"), where(is.numeric), etc.). Columns can be selected or reordered, but renaming selections is not supported.

values	Logical. If FALSE (the default), displays a compact summary of the variable's values. For numeric, character, date/time, labelled, and factor variables, all unique non-missing values are shown when there are at most four; otherwise the first three values, an ellipsis (...), and the last value are shown. Values are sorted when appropriate (e.g., numeric, character, date). For factors, factor_levels controls whether observed or all declared levels are shown; level order is preserved. For labelled variables, prefixed labels are displayed via labelled::to_factor(levels = "prefixed"). If TRUE, all unique non-missing values are displayed.
include_na	Logical. If TRUE, unique missing value markers (<NA>, <NaN>) are appended at the end of the Values summary when present in the variable. This applies to all variable types. Literal strings "NA", "NaN", and "" are quoted to distinguish them from missing markers. If FALSE (the default), missing values are omitted from Values but still counted in the NAs column.
title	Optional character string displayed as the table caption. Defaults to "Codebook". Set to NULL to remove the title completely. When filename = NULL, the title is also used as the base for export filenames after conversion to a portable ASCII name.
filename	Optional character string used as the base for exported CSV, Excel, and PDF filenames. If NULL (the default), a portable filename is derived from title, falling back to "Codebook" when needed. File extensions are added by the browser/export engine.
factor_levels	Character. Controls how factor values are displayed in Values. "all" (the default; varlist() uses "observed") shows all declared levels, including unused levels. "observed" shows only levels present in the data, preserving factor level order.

Details

- The interactive datatable supports column sorting, global searching, and client-side export to various formats.
- Variable selection uses the same tidyselect interface as varlist().
- By default, factor variables document all declared levels, including unused levels — appropriate for a schema-oriented codebook. This differs from varlist(), which defaults to "observed" to summarize observed data only. Pass factor_levels = "observed" to mirror varlist()'s default.
- All exports occur client-side through the Viewer or Tab.

Value

A DT::datatable object.

Dependencies

Requires the following package:

- **DT**

See Also

`varlist()` for generating the underlying variable summaries.

Other variable inspection: `label_from_names()`, `varlist()`

Examples

```
## Not run:
if (requireNamespace("DT", quietly = TRUE)) {
  code_book(sochealth)
  code_book(sochealth, starts_with("bmi"))
  code_book(sochealth, starts_with("bmi"), values = TRUE, include_na = TRUE)

  factors <- data.frame(
    group = factor(c("A", "B", NA), levels = c("A", "B", "C"))
  )
  code_book(
    factors,
    values = TRUE,
    include_na = TRUE,
    factor_levels = "observed"
  )

  code_book(
    sochealth,
    starts_with("bmi"),
    title = "BMI codebook",
    filename = "bmi_codebook"
  )
}

## End(Not run)
```

---

contingency_coef *Pearson's contingency coefficient*

Description

`contingency_coef()` computes Pearson's contingency coefficient C for a two-way contingency table.

Usage

```
contingency_coef(
  x,
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

The contingency coefficient is $C = \sqrt{\chi^2/(\chi^2 + n)}$. It ranges from 0 (independence) to a maximum that depends on the table dimensions. No standard asymptotic standard error exists, so the confidence interval is not computed.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests the null hypothesis of no association (Pearson chi-squared test). CI values are NA because no standard asymptotic SE exists for C.

See Also

cramer_v(), assoc_measures()

Other association measures: assoc_measures(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
contingency_coef(tab)
```

---

copy_clipboard *Copy data to the clipboard*

Description

copy_clipboard() copies a data frame, matrix, array (2D or higher), table or vector to the clipboard. You can paste the result into a text editor (e.g. Notepad++, Sublime Text), a spreadsheet (e.g. Excel, LibreOffice Calc), or a word processor (e.g. Word).

Usage

```
copy_clipboard(
  x,
  row.names.as.col = FALSE,
  row.names = TRUE,
  col.names = TRUE,
  show_message = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

x	A data frame, matrix, 2D array, 3D array, table, or atomic vector to be copied.
row.names.as.col	Logical or character. If FALSE (the default), row names are not added as a column. If TRUE, a column named "rownames" is prepended. If a character string is supplied, it is used as the column name for row names.
row.names	Logical. If TRUE (the default), includes row names in the clipboard output. If FALSE, row names are omitted.
col.names	Logical. If TRUE (the default), includes column names in the clipboard output. If FALSE, column names are omitted.
show_message	Logical. If TRUE (the default), displays a success message after copying. If FALSE, no success message is printed.
quiet	Logical. If FALSE (the default), messages are shown. If TRUE, suppresses all messages, including success, coercion notices, and warnings.
...	Additional arguments passed to clipr::write_clip().

Details

Note: Objects that are not data frames or 2D matrices (e.g. atomic vectors, arrays, tables) are automatically converted to character when copied to the clipboard, as required by clipr::write_clip(). The original object in R remains unchanged.

For multidimensional arrays (e.g. 3D arrays), the entire array is flattened into a 1D character vector, with each element on a new line. To preserve a tabular structure, you should extract a 2D slice before copying. For example: copy_clipboard(my_array[, , 1]).

Value

Invisibly returns the object x. The main purpose is the side effect of copying data to the clipboard.

Examples

```
if (clipr::clipr_available()) {
  # Data frame
  copy_clipboard(sochealth)
```

```
  # Data frame with row names as column
  copy_clipboard(head(sochealth), row.names.as.col = "id")

  # Matrix
  mat <- matrix(1:6, nrow = 2)
  copy_clipboard(mat)

  # Table
  tbl <- table(sochealth$education)
  copy_clipboard(tbl)

  # Array (3D) — flattened to character
  arr <- array(1:8, dim = c(2, 2, 2))
  copy_clipboard(arr)

  # Recommended: copy 2D slice for tabular layout
  copy_clipboard(arr[, , 1])

  # Numeric vector
  copy_clipboard(c(3.14, 2.71, 1.618))

  # Character vector
  copy_clipboard(c("apple", "banana", "cherry"))

  # Quiet mode (no messages shown)
  copy_clipboard(sochealth, quiet = TRUE)
}
```

`count_n` *Row-wise Count of Specific or Special Values*

Description

`count_n()` counts, for each row of a data frame or matrix, how many times one or more values appear across selected columns. It supports type-safe comparison, case-insensitive string matching, and detection of special values such as `NA`, `NaN`, `Inf`, and `-Inf`.

Usage

```
count_n(
  data = NULL,
  select = tidyselect::everything(),
  exclude = NULL,
  count = NULL,
  special = NULL,
  allow_coercion = TRUE,
  ignore_case = FALSE,
  regex = FALSE,
```

```
  verbose = FALSE
)
```

Arguments

data	A data frame or matrix. Optional inside `mutate()`.
select	Columns to include. Defaults to `tidyselect::everything()`. Uses tidyselect helpers like `tidyselect::starts_with()`, etc. If `regex = TRUE`, `select` is treated as a regex string.
exclude	Character vector of column names to exclude after selection. Defaults to `NULL` (no exclusion).
count	Value(s) to count. Defaults to `NULL`. Ignored if `special` is used. Multiple values are allowed (e.g., `count = c(1, 2, 3)` or `count = c("yes", "no")`). R automatically coerces all values in `count` to a common type (e.g., `c(2, "2")` becomes `c("2", "2")`), so all values are expected to be of the same final type. If `allow_coercion = FALSE`, matching is type-safe using `identical()`, and the type of `count` must match that of the values in the data.
special	Character vector of special values to count: `"NA"`, `"NaN"`, `"Inf"`, `"-Inf"`, or `"all"`. Defaults to `NULL`. `"NA"` uses `is.na()`, and therefore includes both `NA` and `NaN` values. `"NaN"` uses `is.nan()` to match only actual NaN values.
allow_coercion	Logical. If `TRUE` (the default), values are compared after coercion. If `FALSE`, uses strict matching via `identical()`.
ignore_case	Logical. If `FALSE` (the default), comparisons are case-sensitive. If `TRUE`, performs case-insensitive string comparisons.
regex	Logical. If `FALSE` (the default), uses tidyselect helpers. If `TRUE`, interprets `select` as a regular expression pattern.
verbose	Logical. If `FALSE` (the default), messages are suppressed. If `TRUE`, prints processing messages.

Details

This function is particularly useful for summarizing data quality or patterns in row-wise structures, and is designed to work fluently inside `dplyr::mutate()` pipelines.

Internally, `count_n()` wraps the stable and dependency-free base function `base_count_n()`, allowing high flexibility and testability.

Value

A numeric vector of row-wise counts (unnamed).

Note

This function is inspired by `datawizard::row_count()`, but provides additional flexibility:

- **Element-wise type-safe matching** using `identical()` when `allow_coercion = FALSE`. This ensures that both the value and its type match exactly, enabling precise comparisons in mixed-type columns.

- **Support for multiple values in** `count`, allowing queries like `count = c(2, 3)` or `count = c("yes", "no")` to count any of several values per row.
- **Detection of special values** such as `NA`, `NaN`, `Inf`, and `-Inf` through the `special` argument — a feature not available in `row_count()`.
- **Tidyverse-native behavior**: can be used inside `mutate()` without explicitly passing a `data` argument.

Value coercion behavior:

R automatically coerces mixed-type vectors passed to `count` into a common type. For example, `count = c(2, "2")` becomes `c("2", "2")`, because R converts numeric and character values to a unified type. This means that mixed-type checks are not possible at runtime once `count` is passed to the function. To ensure accurate type-sensitive matching, users should avoid mixing types in `count` explicitly.

Strict matching mode (`allow_coercion = FALSE`):

When strict matching is enabled, each value in `count` must match the type of the target column exactly.

For factor columns, this means that `count` must also be a factor. Supplying `count = "b"` (a character string) will not match a factor value, even if the label appears identical.

A common and intuitive approach is to use `count = factor("b")`, which works in many cases. However, `identical()` — used internally for strict comparisons — also checks the internal structure of the factor, including the order and content of its levels. As a result, comparisons may still fail if the levels differ, even when the label is the same.

To ensure a perfect match (label **and** levels), you can reuse a value taken directly from the data (e.g., `df$x[2]`). This guarantees that both the class and the factor levels align. However, this approach only works reliably if all selected columns have the same factor structure.

Case-insensitive matching (`ignore_case = TRUE`):

When `ignore_case = TRUE`, all values involved in the comparison are converted to lowercase using `tolower()` before matching. This behavior applies to both character and factor columns. Factors are first converted to character internally.

Importantly, this case-insensitive mode takes precedence over strict type comparison: values are no longer compared using `identical()`, but rather using lowercase string equality. This enables more flexible matching — for example, `"b"` and `"B"` will match even when `allow_coercion = FALSE`.

Example: strict vs. case-insensitive matching with factors:

```
df <- tibble::tibble(
  x = factor(c("a", "b", "c")),
  y = factor(c("b", "B", "a"))
)

# Strict match fails with character input
count_n(df, count = "b", allow_coercion = FALSE)
#> [1] 0 0 0

# Match works only where factor levels match exactly
count_n(df, count = factor("b", levels = levels(df$x)), allow_coercion = FALSE)
```

```
#> [1] 0 1 0

# Case-insensitive match succeeds for both "b" and "B"
count_n(df, count = "b", ignore_case = TRUE)
#> [1] 1 2 0
```

Like `datawizard::row_count()`, this function also supports regex-based column selection, case-insensitive string comparison, and column exclusion.

See Also

Other row-wise summaries: `mean_n()`, `sum_n()`

Examples

```
library(dplyr)
library(tibble)
library(labelled)

# Basic usage
df <- tibble(
  x = c(1, 2, 2, 3, NA),
  y = c(2, 2, NA, 3, 2),
  z = c("2", "2", "2", "3", "2")
)
count_n(df, count = 2)
count_n(df, count = 2, allow_coercion = FALSE)
df |> mutate(num_twos = count_n(count = 2))

# Mixed types and special values
df <- tibble(
  num   = c(1, 2, NA, -Inf, NaN),
  char  = c("a", "B", "b", "a", NA),
  fact  = factor(c("a", "b", "b", "a", "c")),
  date  = as.Date(c("2023-01-01", "2023-01-01", NA, "2023-01-02", "2023-01-01")),
  lab   = labelled(c(1, 2, 1, 2, NA), labels = c(No = 1, Yes = 2)),
  logic = c(TRUE, FALSE, NA, TRUE, FALSE)
)
count_n(df, count = 2)
count_n(df, count = "b", ignore_case = TRUE)
count_n(df, count = "a", select = fact)
count_n(df, count = as.Date("2023-01-01"), select = date)

# Count special values
count_n(df, special = "NA")

# Column selection strategies
df <- tibble(
  score_math    = c(1, 2, 2, 3, NA),
  score_science = c(2, 2, NA, 3, 2),
  score_lang    = c("2", "2", "2", "3", "2"),
  name          = c("Jean", "Marie", "Ali", "Zoe", "Nina")
)
```

```
count_n(df, select = c(score_math, score_science), count = 2)
count_n(df, select = starts_with("score_"), exclude = "score_lang", count = 2)
count_n(df, select = "^score_", regex = TRUE, count = 2)
df |> mutate(nb_two = count_n(count = 2))

# Strict type-safe matching with factor columns
df <- tibble(
  x = factor(c("a", "b", "c")),
  y = factor(c("b", "B", "a"))
)

# Coercion: character "b" matches both x and y
count_n(df, count = "b")

# Strict match: fails because "b" is character, not factor (returns only 0s)
count_n(df, count = "b", allow_coercion = FALSE)

# Strict match with factor value: works only where levels match
count_n(df, count = factor("b", levels = levels(df$x)), allow_coercion = FALSE)
```

`cramer_v` *Cramer's V*

Description

`cramer_v()` computes Cramer's V for a two-way contingency table, measuring the strength of association between two categorical variables.

Usage

```
cramer_v(
  x,
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

`x`	A contingency table (of class `table`).
`detail`	Logical. If `FALSE` (default), return the estimate as a numeric scalar. If `TRUE`, return a named numeric vector including confidence interval and p-value.
`conf_level`	A number between 0 and 1 giving the confidence level (default `0.95`). Only used when `detail = TRUE`. Set to `NULL` to omit the confidence interval.
`digits`	Number of decimal places used when printing the result (default `3`). Only affects the `detail = TRUE` output.
`.include_se`	Internal parameter; do not use.

Details

Cramer's V is computed as $V = \sqrt{\chi^2/(n \cdot (k-1))}$, where χ^2 is the Pearson chi-squared statistic, n is the total count, and $k = \min(r, c)$. The point estimate matches the DescTools (Signorell et al., 2024) and SPSS implementations. The confidence interval uses the Fisher z-transformation on V ($\tanh(\text{atanh}(V) \pm z_{\alpha/2}/\sqrt{n-3})$), which differs from the noncentral chi-squared or bootstrap CIs reported by `DescTools::CramerV()`.

Value

When `detail = FALSE`: a single numeric value (the estimate). When `detail = TRUE` and `conf_level` is non-`NULL`: `c(estimate, ci_lower, ci_upper, p_value)`. When `detail = TRUE` and `conf_level = NULL`: `c(estimate, p_value)`. The p-value tests the null hypothesis of no association (Pearson chi-squared test).

References

Agresti, A. (2002). *Categorical Data Analysis* (2nd ed.). Wiley.

Liebetrau, A. M. (1983). *Measures of Association*. Sage.

Signorell, A. et al. (2024). *DescTools: Tools for Descriptive Statistics*. R package.

See Also

`phi()`, `contingency_coef()`, `assoc_measures()`

Other association measures: `assoc_measures()`, `contingency_coef()`, `gamma_gk()`, `goodman_kruskal_tau()`, `kendall_tau_b()`, `kendall_tau_c()`, `lambda_gk()`, `phi()`, `somers_d()`, `uncertainty_coef()`, `yule_q()`

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
cramer_v(tab)
cramer_v(tab, detail = TRUE)
cramer_v(tab, detail = TRUE, conf_level = NULL)
```

---

cross_tab *Cross-tabulation*

Description

Computes a two-way cross-tabulation with optional weights, grouping (including combinations of multiple variables), percentage displays, and inferential statistics.

`cross_tab()` produces weighted or unweighted contingency tables with row or column percentages, optional grouping via `by`, and associated Chi-squared tests with an association measure and diagnostic information.

Both `x` and `y` variables are required. For one-way frequency tables, use `freq()` instead.

Usage

```
cross_tab(
  data,
  x,
  y = NULL,
  by = NULL,
  weights = NULL,
  rescale = FALSE,
  percent = c("none", "column", "row"),
  include_stats = TRUE,
 assoc_measure = c("auto", "cramer_v", "phi", "gamma", "tau_b", "tau_c", "somers_d",
    "lambda", "none"),
  assoc_ci = FALSE,
  correct = FALSE,
  simulate_p = FALSE,
  simulate_B = 2000,
  digits = NULL,
  styled = TRUE,
  show_n = TRUE,
  decimal_mark = ".",
  p_digits = 3L
)
```

Arguments

data	A data frame. Alternatively, a vector when using the vector-based interface.
x	Row variable (unquoted).
y	Column variable (unquoted). Mandatory; for one-way tables, use freq().
by	Optional grouping variable or expression. Can be a single variable or a combination of multiple variables (e.g. interaction(vs, am)).
weights	Optional numeric weights.
rescale	Logical. If FALSE (the default), weights are used as-is. If TRUE, rescales weights so total weighted N matches raw N.
percent	One of "none" (the default), "row", "column". Unique abbreviations are accepted (e.g. "n", "r", "c").
include_stats	Logical. If TRUE (the default), computes Chi-squared and an association measure (see assoc_measure).
assoc_measure	Character. Which association measure to report. "auto" (default) selects Kendall's Tau-b when both variables are ordered factors and Cramer's V otherwise. Other choices: "cramer_v", "phi", "gamma", "tau_b", "tau_c", "somers_d", "lambda", "none".
assoc_ci	Logical. If TRUE, includes the 95 percent confidence interval of the association measure in the note. Defaults to FALSE.
correct	Logical. If FALSE (the default), no continuity correction is applied. If TRUE, applies Yates correction (only for 2x2 tables).

simulate_p	Logical. If FALSE (the default), uses asymptotic p-values. If TRUE, uses Monte Carlo simulation.
simulate_B	Integer. Number of replicates for Monte Carlo simulation. Defaults to 2000.
digits	Number of decimals for cell values. Defaults to 1 for percentages, 0 for counts.
styled	Logical. If TRUE (the default), returns a spicy_cross_table object (for formatted printing). If FALSE, returns a plain data.frame.
show_n	Logical. If TRUE (the default), adds marginal N totals when percent != "none".
decimal_mark	Character used as the decimal mark in printed numeric values (cells, chi-squared, association estimate, CI bounds, p-value). Defaults to ".". Set to "," for European formatting; matches the decimal_mark argument of the table_*() family.
p_digits	Integer number of decimals used to format the p-value (and to determine the small-p threshold below which < .001 notation is used). Defaults to 3 (the APA standard); matches the p_digits argument of the table_*() family.

Value

A data.frame, list of data.frames, or spicy_cross_table object. When by is used, returns a spicy_cross_table_list.

Global Options

The function recognizes the following global options that modify its default behavior:

- options(spicy.percent = "column") Sets the default percentage mode for all calls to cross_tab(). Valid values are "none", "row", and "column". Equivalent to setting percent = "column" (or another choice) in each call.
- options(spicy.simulate_p = TRUE) Enables Monte Carlo simulation for all Chi-squared tests by default. Equivalent to setting simulate_p = TRUE in every call.
- options(spicy.rescale = TRUE) Automatically rescales weights so that total weighted N equals the raw N. Equivalent to setting rescale = TRUE in each call.

These options are convenient for users who wish to enforce consistent behavior across multiple calls to cross_tab() and other spicy table functions. They can be disabled or reset by setting them to NULL: options(spicy.percent = NULL, spicy.simulate_p = NULL, spicy.rescale = NULL).

Example:

```
options(spicy.simulate_p = TRUE, spicy.rescale = TRUE)
cross_tab(sochealth, smoking, education, weights = weight)
```

Examples

```
# Basic crosstab
cross_tab(sochealth, smoking, education)

# Column percentages
cross_tab(sochealth, smoking, education, percent = "column")

# Weighted (rescaled)
```

```
cross_tab(sochealth, smoking, education, weights = weight, rescale = TRUE)

# Grouped by sex
cross_tab(sochealth, smoking, education, by = sex)

# Grouped by combination of variables
cross_tab(sochealth, smoking, education, by = interaction(sex, age_group))

# Ordinal variables: auto-selects Kendall's Tau-b
cross_tab(sochealth, education, self_rated_health)

# 2x2 table with Yates correction
cross_tab(sochealth, smoking, physical_activity, correct = TRUE)

# APA-style p-value precision and European decimal mark
cross_tab(sochealth, smoking, education, decimal_mark = ",", p_digits = 4)
```

`freq` *Frequency Table*

Description

Creates a frequency table for a vector or variable from a data frame, with options for weighting, sorting, handling *labelled* data, defining custom missing values, and displaying cumulative percentages.

When `styled = TRUE`, the function prints a spicy-formatted ASCII table using `print.spicy_freq_table()` and `spicy_print_table()`; otherwise, it returns a `data.frame` containing frequencies and proportions.

Usage

```
freq(
  data,
  x = NULL,
  weights = NULL,
  digits = 1L,
  valid = TRUE,
  cum = FALSE,
  sort = "",
  na_val = NULL,
  labelled_levels = c("prefixed", "labels", "values"),
  factor_levels = c("observed", "all"),
  rescale = TRUE,
  decimal_mark = ".",
  styled = TRUE,
  ...
)
```

Arguments

data	A data.frame, vector, or factor. If a data frame is provided, specify the target variable x. If both data and x are supplied as vectors, data is ignored with a warning.
x	A variable from data (unquoted).
weights	Optional numeric vector of weights (same length as x). The variable may be referenced as a bare name when it belongs to data, or as a qualified expression like other$w (evaluated in the calling environment), which always takes precedence over data lookup. Observations with NA weights are dropped from the table with a warning; see Details.
digits	Number of decimal digits to display for percentages (default: 1).
valid	Logical. If TRUE (default), display valid percentages (excluding missing values).
cum	Logical. If FALSE (the default), cumulative percentages are omitted. If TRUE, adds cumulative percentages.
sort	Sorting method for values: • "" - no sorting (default) • "+" - increasing frequency • "-" - decreasing frequency • "name+" - alphabetical A-Z • "name-" - alphabetical Z-A

na_val	Atomic vector of numeric or character values to be treated as missing (NA).
	For *labelled* variables (from **haven** or **labelled**), this argument must refer to the underlying coded values, not the visible labels.
	Example:
	`x <- labelled(c(1, 2, 3, 1, 2, 3), c("Low" = 1, "Medium" = 2, "High" = 3))`
	`freq(x, na_val = 1) # Treat all "Low" as missing`

labelled_levels	For labelled variables, defines how labels and values are displayed: • "prefixed" or "p" - show labels as [value] label (default) • "labels" or "l" - show only labels • "values" or "v" - show only numeric codes
factor_levels	Character. Controls how factor and labelled values are displayed in the frequency table. "observed" (the default; matches Stata's tab) shows only levels present in the data. "all" (matches SPSS FREQUENCIES and code_book()'s default) keeps every declared level, including unused ones, which appear with n = 0.
rescale	Logical. If TRUE (default), rescale weights so that their total equals the unweighted sample size (length(weights)). See Details for the interaction with NA weights.
decimal_mark	Character used as the decimal mark in printed percentages. Either "." (the default) or ",". Matches the decimal_mark argument of cross_tab() and the three table_*() helpers, so European-locale users get a consistent experience across the package.

styled	Logical. If TRUE (default), print the formatted spicy table. If FALSE, return a plain data.frame with frequency values.
...	Additional arguments passed to print.spicy_freq_table().

Details

This function is designed to mimic common frequency procedures from statistical software such as SPSS or Stata, while integrating the flexibility of R's data structures.

It automatically detects the type of input (vector, factor, or labelled) and applies appropriate transformations, including:

- Handling of labelled variables via **labelled** or **haven**
- Optional recoding of specific values as missing (na_val)
- Optional weighting with a rescaling mechanism
- Support for cumulative percentages (cum = TRUE)
- Multiple display modes for labels via labelled_levels
- Schema-vs-observed level display via factor_levels

For factor and labelled inputs, the factor_levels argument controls whether declared-but-unobserved levels appear in the output. The default "observed" drops them (Stata tab behavior); "all" keeps them with n = 0, matching SPSS FREQUENCIES and code_book()'s default. For schema-level inspection without computing frequencies, use varlist() or code_book() with factor_levels = "all".

When weighting is applied (weights), the frequencies and percentages are computed proportionally to the weights. The argument rescale = TRUE normalizes weights so their sum equals the unweighted sample size (length(weights)).

Missing values in weights cause those observations to be dropped from the table entirely (with a warning), matching the behaviour of cross_tab() in spicy 0.11.0+. With rescale = TRUE, the remaining (non-NA-weighted) weights are normalized so the total weighted N equals the count of non-NA-weighted rows. With rescale = FALSE, the total weighted N is the actual sum of non-NA weights.

Value

With styled = FALSE, a plain data.frame with no extra attributes and columns:

- value - unique values or factor levels
- n - frequency count (weighted if applicable)
- prop - proportion of total
- valid_prop - proportion of valid responses (if valid = TRUE)
- cum_prop, cum_valid_prop - cumulative percentages (if cum = TRUE)

With styled = TRUE (default), prints the formatted table to the console and invisibly returns a spicy_freq_table object: the same data.frame carrying rendering metadata as attributes (digits, data_name, var_name, var_label, class_name, n_total, n_valid, weighted, rescaled, weight_var) used by print.spicy_freq_table().

See Also

print.spicy_freq_table() for formatted printing. spicy_print_table() for the underlying ASCII rendering engine.

Examples

```
# Frequency table with labelled ordered factor
freq(sochealth, education)
freq(sochealth, self_rated_health, sort = "-")

library(labelled)

# Simple numeric vector
x <- c(1, 2, 2, 3, 3, 3, NA)
freq(x)

# Plain vector with a sentinel value recoded as missing
freq(c(1, 2, 3, 99, 99), na_val = 99)

# Labelled variable (haven-style)
x_lbl <- labelled(
  c(1, 2, 3, 1, 2, 3, 1, 2, NA),
  labels = c("Low" = 1, "Medium" = 2, "High" = 3)
)
var_label(x_lbl) <- "Satisfaction level"

# Treat value 1 ("Low") as missing
freq(x_lbl, na_val = 1)

# Display only labels, add cumulative %
freq(x_lbl, labelled_levels = "labels", cum = TRUE)

# Display values only, sorted descending
freq(x_lbl, labelled_levels = "values", sort = "-")

# Show all declared factor levels, including unused ones (SPSS-style).
# The default "observed" mirrors Stata's `tab` and drops unused levels.
f <- factor(c("Yes", "No", "Yes"), levels = c("Yes", "No", "Maybe"))
freq(f, factor_levels = "all")

# With weighting
df <- data.frame(
  sex = factor(c("Male", "Female", "Female", "Male", NA, "Female")),
  weight = c(12, 8, 10, 15, 7, 9)
)

# Weighted frequencies (normalized)
freq(df, sex, weights = weight, rescale = TRUE)

# Weighted frequencies (without rescaling)
freq(df, sex, weights = weight, rescale = FALSE)
```

```
# Base R style, with weights and cumulative percentages
freq(df$sex, weights = df$weight, cum = TRUE)

# Piped version (tidy syntax) and sort alphabetically descending ("name-")
df |> freq(sex, sort = "name-")

# European decimal mark (matches `cross_tab()` and the `table_*()` family)
freq(sochealth, education, decimal_mark = ",")

# Non-styled return (for programmatic use)
f <- freq(df, sex, styled = FALSE)
head(f)
```

---

gamma_gk *Goodman-Kruskal Gamma*

Description

gamma_gk() computes the Goodman-Kruskal Gamma statistic for a two-way contingency table of ordinal variables.

Usage

```
gamma_gk(
  x,
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Gamma is computed as $\gamma = (C - D)/(C + D)$, where C and D are the numbers of concordant and discordant pairs. It ignores tied pairs, making it appropriate for ordinal variables with many ties. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: gamma = 0 (Wald z-test).

See Also

kendall_tau_b(), kendall_tau_c(), somers_d(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$education, sochealth$self_rated_health)
gamma_gk(tab)
gamma_gk(tab, detail = TRUE)
```

---

goodman_kruskal_tau *Goodman-Kruskal's Tau*

Description

goodman_kruskal_tau() computes Goodman-Kruskal's Tau, a proportional reduction in error (PRE) measure for nominal variables.

Usage

```
goodman_kruskal_tau(
  x,
  direction = c("row", "column"),
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
direction	Direction of prediction: "row" (default, column predicts row) or "column" (row predicts column).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.

digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Unlike lambda_gk(), Goodman-Kruskal's Tau uses all cell frequencies rather than only the modal categories, making it more sensitive to association patterns where lambda may be zero. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: tau = 0 (Wald z-test).

See Also

lambda_gk(), uncertainty_coef(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
goodman_kruskal_tau(tab)
goodman_kruskal_tau(tab, direction = "column", detail = TRUE)
```

---

kendall_tau_b *Kendall's Tau-b*

Description

kendall_tau_b() computes Kendall's Tau-b for a two-way contingency table of ordinal variables.

Usage

```
kendall_tau_b(
  x,
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Kendall's Tau-b is computed as $\tau_b = (C - D)/\sqrt{(n_0 - n_1)(n_0 - n_2)}$, where $n_0 = n(n-1)/2$, n_1 is the number of pairs tied on the row variable, and n_2 is the number tied on the column variable. Tau-b corrects for ties and is appropriate for square tables. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: tau-b = 0 (Wald z-test).

See Also

kendall_tau_c(), gamma_gk(), somers_d(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$education, sochealth$self_rated_health)
kendall_tau_b(tab)
```

---

kendall_tau_c *Kendall's Tau-c (Stuart's Tau-c)*

Description

kendall_tau_c() computes Stuart's Tau-c (also known as Kendall's Tau-c) for a two-way contingency table of ordinal variables.

Usage

```
kendall_tau_c(
  x,
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Stuart's Tau-c is computed as $\tau_c = 2m(C-D)/(n^2(m-1))$, where $m = \min(r, c)$. It is appropriate for rectangular tables and is not restricted to the range $[-1, 1]$ only for square tables. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: tau-c = 0 (Wald z-test).

See Also

kendall_tau_b(), gamma_gk(), somers_d(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), lambda_gk(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$education, sochealth$self_rated_health)
kendall_tau_c(tab)
```

`label_from_names` *Derive variable labels from column names* `name<sep>label`

Description

Splits each column name at the **first** occurrence of `sep`, renames the column to the part before `sep` (the *name*, trimmed of surrounding whitespace), and assigns the part after `sep` as a `"label"` attribute on the column. The label attribute follows the haven convention also used by `labelled::var_label()`, so labelled-aware tooling (`labelled`, `haven`, `varlist()`, `code_book()`, ...) reads it transparently. Splitting at the first `sep` means the label itself may contain the separator.

Usage

```
label_from_names(df, sep = ". ")
```

Arguments

`df`	A `data.frame` or tibble with column names of the form `"name<sep>label"` (e.g. `"code. question text"`).
`sep`	Character string used as separator between name and label. Default `". "` (LimeSurvey's default); any literal string can be used. Matched as a fixed string, so regex metacharacters such as `.` or `|` carry no special meaning.

Details

This is especially useful for **LimeSurvey CSV exports** when using *Export results -> Export format: CSV -> Headings: Question code & question text*, where column names look like `"code. question text"`. The default separator is `". "` to match that export.

LimeSurvey question codes (the part *before* `sep`) are restricted to alphanumeric characters, must start with a letter, and cannot contain spaces or special characters. The column name therefore needs to encode both the code *and* the question text, separated by a literal string – there is no way to recover a label from a code alone. If your export uses *Headings: Question code* (codes only), re-export with *Headings: Question code & question text* (which inserts the default `". "` separator) before calling this function.

Value

An object of the **same class as** `df` – a base `data.frame` if `df` was a base `data.frame`, a `tbl_df` if `df` was a tibble. The output has column names equal to the trimmed names (before `sep`) and, for every column whose original name contained `sep`, a `"label"` attribute equal to the label (after `sep`). Columns whose name does not contain `sep` are passed through unchanged with no label attached.

Errors

The function raises an actionable error – rather than letting the downstream constructor raise a cryptic one – when the split produces:

- duplicate column names (two original names share the same prefix before sep); or
- an empty column name (the original name starts with sep and has nothing before it).

See Also

labelled::var_label() reads the "label" attribute set by this function; varlist() and code_book() surface it in their inspection outputs.

Other variable inspection: code_book(), varlist()

Examples

```
# LimeSurvey-style column names (default sep = ". ").
df <- data.frame(
  "age. Age of respondent" = c(25, 30),
  "score. Total score. Manually computed." = c(12, 14),
  check.names = FALSE
)
out <- label_from_names(df)
attr(out$age, "label")
attr(out$score, "label")

# Custom separator.
df2 <- data.frame(
  "id|Identifier" = 1:3,
  "score|Total score" = c(10, 20, 30),
  check.names = FALSE
)
out2 <- label_from_names(df2, sep = "|")
```

---

lambda_gk *Goodman-Kruskal's Lambda*

Description

lambda_gk() computes Goodman-Kruskal's Lambda, a proportional reduction in error (PRE) measure for nominal variables.

Usage

```
lambda_gk(
  x,
  direction = c("symmetric", "row", "column"),
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
direction	Direction of prediction: "symmetric" (default), "row" (column predicts row), or "column" (row predicts column).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Lambda measures how much prediction error is reduced when the independent variable is used to predict the dependent variable. It ranges from 0 (no reduction) to 1 (perfect prediction). Lambda can equal zero even when variables are associated if the modal category dominates in every column (or row). Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: lambda = 0 (Wald z-test).

See Also

goodman_kruskal_tau(), uncertainty_coef(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), phi(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
lambda_gk(tab)
lambda_gk(tab, direction = "row")
lambda_gk(tab, direction = "column", detail = TRUE)
```

mean_n *Row Means with Optional Minimum Valid Values*

Description

`mean_n()` computes row means from a `data.frame` or `matrix`, handling missing values (`NA`s) automatically. Row-wise means are calculated across selected numeric columns, with an optional condition on the minimum number (or proportion) of valid (non-missing) values required for a row to be included. Non-numeric columns are excluded automatically and reported.

Usage

```
mean_n(
  data = NULL,
  select = tidyselect::everything(),
  exclude = NULL,
  min_valid = NULL,
  digits = NULL,
  regex = FALSE,
  verbose = FALSE
)
```

Arguments

`data`	A `data.frame` or `matrix`.
`select`	Columns to include. If `regex = FALSE`, use tidyselect syntax (default: `tidyselect::everything()`). If `regex = TRUE`, provide a regular expression pattern (character string).
`exclude`	Columns to exclude (default: `NULL`).
`min_valid`	Minimum number of valid (non-`NA`) values required per row. Accepts: • `NULL` (the default) — every selected column must be valid. • a proportion in `(0, 1)` — `round(ncol(x) * min_valid)` valid columns required (e.g. `min_valid = 0.5` requires at least half of the selected columns to be non-`NA`). • a non-negative integer count up to the number of selected numeric columns. Non-integer values `>= 1` (e.g. `1.5`) and counts greater than `ncol(x)` raise an actionable error.
`digits`	Optional non-negative integer giving the number of decimal places to round the result to. Defaults to `NULL` (no rounding).
`regex`	Logical. If `FALSE` (the default), uses tidyselect helpers. If `TRUE`, the `select` argument is treated as a regular expression.
`verbose`	Logical. If `FALSE` (the default), messages are suppressed. If `TRUE`, prints a message about non-numeric columns excluded.

Value

A numeric vector of row-wise means.

See Also

Other row-wise summaries: count_n(), sum_n()

Examples

```
library(dplyr)

# Create a simple numeric data frame
df <- tibble(
  var1 = c(10, NA, 30, 40, 50),
  var2 = c(5, NA, 15, NA, 25),
  var3 = c(NA, 30, 20, 50, 10)
)

# Compute row-wise mean (all values must be valid by default)
mean_n(df)

# Require at least 2 valid (non-NA) values per row
mean_n(df, min_valid = 2)

# Require at least 50% valid (non-NA) values per row
mean_n(df, min_valid = 0.5)

# Round the result to 1 decimal
mean_n(df, digits = 1)

# Select specific columns
mean_n(df, select = c(var1, var2))

# Select specific columns using a pipe
df |>
  select(var1, var2) |>
  mean_n()

# Exclude a column
mean_n(df, exclude = "var3")

# Select columns ending with "1"
mean_n(df, select = ends_with("1"))

# Use with native pipe
df |> mean_n(select = starts_with("var"))

# Use inside dplyr::mutate()
df |> mutate(mean_score = mean_n(min_valid = 2))

# Select columns directly inside mutate()
df |> mutate(mean_score = mean_n(select = c(var1, var2), min_valid = 1))
```

```
# Select columns before mutate
df |>
  select(var1, var2) |>
  mutate(mean_score = mean_n(min_valid = 1))

# Show verbose processing info
df |> mutate(mean_score = mean_n(min_valid = 2, digits = 1, verbose = TRUE))

# Add character and grouping columns
df_mixed <- mutate(df,
  name = letters[1:5],
  group = c("A", "A", "B", "B", "A")
)
df_mixed

# Non-numeric columns are ignored
mean_n(df_mixed)

# Use within mutate() on mixed data
df_mixed |> mutate(mean_score = mean_n(select = starts_with("var")))

# Use everything() but exclude non-numeric columns manually
mean_n(df_mixed, select = everything(), exclude = "group")

# Select columns using regex
mean_n(df_mixed, select = "^var", regex = TRUE)
mean_n(df_mixed, select = "ar", regex = TRUE)

# Apply to a subset of rows (first 3)
df_mixed[1:3, ] |> mean_n(select = starts_with("var"))

# Store the result in a new column
df_mixed$mean_score <- mean_n(df_mixed, select = starts_with("var"))
df_mixed

# With a numeric matrix
mat <- matrix(c(1, 2, NA, 4, 5, NA, 7, 8, 9), nrow = 3, byrow = TRUE)
mat
mat |> mean_n(min_valid = 2)
```

---

phi *Phi coefficient*

Description

`phi()` computes the phi coefficient for a 2x2 contingency table.

Usage

```
phi(x, detail = FALSE, conf_level = 0.95, digits = 3L, .include_se = FALSE)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

The phi coefficient is $\phi = \sqrt{\chi^2/n}$. It is equivalent to Cramer's V for 2x2 tables and equals the Pearson correlation between the two binary variables. The point estimate matches the DescTools (Signorell et al., 2024) and SPSS implementations. The confidence interval uses the Fisher z-transformation on ϕ; see cramer_v() for the formula and full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests the null hypothesis of no association (Pearson chi-squared test).

See Also

cramer_v(), yule_q(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), somers_d(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$sex)
phi(tab)
phi(tab, detail = TRUE)
```

`sochealth` *Simulated social-health survey*

Description

A simulated dataset of 1200 respondents from a fictional social-health survey, designed to illustrate the main features of the spicy package: variable labels, ordered factors, survey weights, association measures, and APA-style reporting.

Usage

```
sochealth
```

Format

A tibble with 1200 rows and 24 variables:

sex Factor. Sex of the respondent.

age Numeric. Age in years (25–75).

age_group Ordered factor. Age group (25–34, 35–49, 50–64, 65–75).

education Ordered factor. Highest education level (Lower secondary, Upper secondary, Tertiary).

social_class Ordered factor. Subjective social class (Lower, Working, Lower middle, Middle, Upper middle).

region Factor. Region of residence (6 regions).

employment_status Factor. Employment status (Employed, Student, Unemployed, Inactive).

income_group Ordered factor. Household income group (Low, Lower middle, Upper middle, High). Contains missing values.

income Numeric. Monthly household income in CHF.

smoking Factor. Current smoker (No, Yes). Contains missing values.

physical_activity Factor. Regular physical activity (No, Yes).

dentist_12m Factor. Dentist visit in the last 12 months (No, Yes).

self_rated_health Ordered factor. Self-rated health (Poor, Fair, Good, Very good). Contains missing values.

wellbeing_score Numeric. WHO-5 wellbeing index (0–100).

bmi Numeric. Body mass index. Contains missing values.

bmi_category Ordered factor. BMI category (Normal weight, Overweight, Obesity). Contains missing values.

institutional_trust Ordered factor. Trust in institutions (Very low, Low, High, Very high).

political_position Numeric. Political position on a 0 (left) to 10 (right) scale. Contains missing values.

life_sat_health Integer. Satisfaction with own health (1–5 Likert scale). Contains missing values.

life_sat_work Integer. Satisfaction with work or main activity (1–5 Likert scale). Contains missing values.

life_sat_relationships Integer. Satisfaction with personal relationships (1–5 Likert scale). Contains missing values.

life_sat_standard Integer. Satisfaction with standard of living (1–5 Likert scale). Contains missing values.

response_date POSIXct. Date and time of survey response (September–November 2024).

weight Numeric. Survey design weight.

Details

All variables carry labels (accessible via `labelled::var_label()` and displayed by `varlist()`). Several ordered factors are included so that `cross_tab()` can demonstrate automatic ordinal measure selection.

Source

Simulated data for illustration purposes.

Examples

```
data(sochealth)
varlist(sochealth)
freq(sochealth, education)
cross_tab(sochealth, education, self_rated_health)
```

`somers_d` *Somers' D*

Description

`somers_d()` computes Somers' D for a two-way contingency table of ordinal variables.

Usage

```
somers_d(
  x,
  direction = c("row", "column", "symmetric"),
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
direction	Direction of prediction: "row" (default, column predicts row), "column" (row predicts column), or "symmetric" (average of both directions).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

Somers' D is an asymmetric ordinal measure defined as $d = (C - D)/(C + D + T)$, where T is the number of pairs tied on the independent variable. The symmetric version is the harmonic mean of the two asymmetric values. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: D = 0 (Wald z-test).

See Also

kendall_tau_b(), gamma_gk(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), uncertainty_coef(), yule_q()

Examples

```
tab <- table(sochealth$education, sochealth$self_rated_health)
somers_d(tab, direction = "row")
somers_d(tab, direction = "column", detail = TRUE)
```

`spicy_print_table` *Print a spicy-formatted ASCII table*

Description

User-facing helper that prints a visually aligned, spicy-styled ASCII table created by functions such as `freq()` or `cross_tab()`. It automatically adjusts column alignment, spacing, and separators for improved readability in console outputs.

This function wraps the internal renderer `build_ascii_table()`, adding optional titles, notes, and automatic alignment rules depending on the type of table.

Usage

```
spicy_print_table(
  x,
  title = attr(x, "title"),
  note = attr(x, "note"),
  padding = 2L,
  first_column_line = TRUE,
  row_total_line = TRUE,
  column_total_line = TRUE,
  bottom_line = FALSE,
  lines_color = "darkgrey",
  align_left_cols = NULL,
  align_center_cols = integer(0),
  group_sep_rows = integer(0),
  total_row_idx = attr(x, "total_row_idx"),
  ...
)
```

Arguments

`x`	A `spicy_table` or `data.frame` to be printed.
`title`	Optional title displayed above the table. Defaults to the `"title"` attribute of `x` if present.
`note`	Optional note displayed below the table. Defaults to the `"note"` attribute of `x` if present.
`padding`	Non-negative integer giving the number of extra characters added to each column's auto-computed width (max of cell-content width and header width). Defaults to `2L`. See `build_ascii_table()` for the precise formula and the migration note from the pre-0.11.0 string enum.
`first_column_line`	Logical. If `TRUE` (the default), adds a vertical separator after the first column.

`row_total_line`, `column_total_line`, `bottom_line`
: Logical flags controlling the presence of horizontal lines before total rows/columns or at the bottom of the table. Both `row_total_line` and `column_total_line` default to `TRUE`; `bottom_line` defaults to `FALSE`.

`lines_color`
: Character. Color for table separators. Defaults to `"darkgrey"`. Only applied if the output supports ANSI colors (see `crayon::has_color()`).

`align_left_cols`
: Integer vector of column indices to left-align. If `NULL` (the default), alignment is auto-detected based on `x`:
  - For `freq` tables -> `c(1, 2)`
  - For `cross` tables -> `1`

`align_center_cols`
: Integer vector of column indices to center-align. Defaults to `integer(0)`.

`group_sep_rows`
: Integer vector of row indices before which a light dashed separator line is drawn. Defaults to `integer(0)`.

`total_row_idx`
: Optional integer vector of 1-based row indices identifying the totals rows; defaults to the `"total_row_idx"` attribute of `x` (set by `cross_tab()`). See `build_ascii_table()`.

`...`
: Additional arguments passed to `build_ascii_table()`.

Details

`spicy_print_table()` detects whether the table represents frequencies (`freq`-style) or cross-tabulations (`cross`-style) and adjusts formatting accordingly:

- For **frequency tables**, the first two columns (*Category* and *Values*) are left-aligned.
- For **cross tables**, only the first column (row variable) is left-aligned.

The function supports Unicode line-drawing characters and colored separators using the **crayon** package, with graceful fallback to monochrome output when color is not supported. If the table exceeds the console width, it is split into stacked horizontal panels while repeating the left-most identifier columns.

Value

Invisibly returns `x`, after printing the formatted ASCII table to the console.

See Also

`build_ascii_table()` for the underlying text rendering engine. `print.spicy_freq_table()` for the specialized printing method used by `freq()`.

Examples

```
# Simple demonstration
df <- data.frame(
  Category = c("Valid", "", "Missing", "Total"),
  Values = c("Yes", "No", "NA", ""),
  Freq. = c(12, 8, 1, 21),
```

```
  Percent = c(57.1, 38.1, 4.8, 100.0)
)

spicy_print_table(df,
  title = "Frequency table: Example",
  note = "Class: data.frame\nData: demo"
)
```

---

sum_n *Row Sums with Optional Minimum Valid Values*

Description

sum_n() computes row sums from a data.frame or matrix, handling missing values (NAs) automatically. Row-wise sums are calculated across selected numeric columns, with an optional condition on the minimum number (or proportion) of valid (non-missing) values required for a row to be included. Non-numeric columns are excluded automatically and reported.

Usage

```
sum_n(
  data = NULL,
  select = tidyselect::everything(),
  exclude = NULL,
  min_valid = NULL,
  digits = NULL,
  regex = FALSE,
  verbose = FALSE
)
```

Arguments

data	A data.frame or matrix.
select	Columns to include. If regex = FALSE, use tidyselect syntax (default: tidyselect::everything()). If regex = TRUE, provide a regular expression pattern (character string).
exclude	Columns to exclude (default: NULL).
min_valid	Minimum number of valid (non-NA) values required per row. Accepts: • NULL (the default) — every selected column must be valid. • a proportion in (0, 1) — round(ncol(x) * min_valid) valid columns required (e.g. min_valid = 0.5 requires at least half of the selected columns to be non-NA). • a non-negative integer count up to the number of selected numeric columns. Non-integer values >= 1 (e.g. 1.5) and counts greater than ncol(x) raise an actionable error.

digits	Optional non-negative integer giving the number of decimal places to round the result to. Defaults to NULL (no rounding).
regex	Logical. If FALSE (the default), uses tidyselect helpers. If TRUE, the select argument is treated as a regular expression.
verbose	Logical. If FALSE (the default), messages are suppressed. If TRUE, prints a message about non-numeric columns excluded.

Value

A numeric vector of row-wise sums.

See Also

Other row-wise summaries: count_n(), mean_n()

Examples

```
library(dplyr)

# Create a simple numeric data frame
df <- tibble(
  var1 = c(10, NA, 30, 40, 50),
  var2 = c(5, NA, 15, NA, 25),
  var3 = c(NA, 30, 20, 50, 10)
)

# Compute row-wise sums (all values must be valid by default)
sum_n(df)

# Require at least 2 valid (non-NA) values per row
sum_n(df, min_valid = 2)

# Require at least 50% valid (non-NA) values per row
sum_n(df, min_valid = 0.5)

# Round the results to 1 decimal
sum_n(df, digits = 1)

# Select specific columns
sum_n(df, select = c(var1, var2))

# Select specific columns using a pipe
df |>
  select(var1, var2) |>
  sum_n()

# Exclude a column
sum_n(df, exclude = "var3")

# Select columns ending with "1"
sum_n(df, select = ends_with("1"))
```

```
# Use with native pipe
df |> sum_n(select = starts_with("var"))

# Use inside dplyr::mutate()
df |> mutate(sum_score = sum_n(min_valid = 2))

# Select columns directly inside mutate()
df |> mutate(sum_score = sum_n(select = c(var1, var2), min_valid = 1))

# Select columns before mutate
df |>
  select(var1, var2) |>
  mutate(sum_score = sum_n(min_valid = 1))

# Show verbose message
df |> mutate(sum_score = sum_n(min_valid = 2, digits = 1, verbose = TRUE))

# Add character and grouping columns
df_mixed <- mutate(df,
  name = letters[1:5],
  group = c("A", "A", "B", "B", "A")
)
df_mixed

# Non-numeric columns are ignored
sum_n(df_mixed)

# Use inside mutate with mixed data
df_mixed |> mutate(sum_score = sum_n(select = starts_with("var")))

# Use everything(), but exclude known non-numeric
sum_n(df_mixed, select = everything(), exclude = "group")

# Select columns using regex
sum_n(df_mixed, select = "^var", regex = TRUE)
sum_n(df_mixed, select = "ar", regex = TRUE)

# Apply to a subset of rows
df_mixed[1:3, ] |> sum_n(select = starts_with("var"))

# Store the result in a new column
df_mixed$sum_score <- sum_n(df_mixed, select = starts_with("var"))
df_mixed

# With a numeric matrix
mat <- matrix(c(1, 2, NA, 4, 5, NA, 7, 8, 9), nrow = 3, byrow = TRUE)
mat
mat |> sum_n(min_valid = 2)
```

`table_categorical` *Categorical summary table*

Description

Builds a publication-ready frequency or cross-tabulation table for one or many categorical variables selected with tidyselect syntax.

With by, produces grouped cross-tabulation summaries (using `cross_tab()` internally) with Chi-squared *p*-values and optional association measures. Without by, produces one-way frequency-style summaries.

Multiple output formats are available via `output`: a printed ASCII table (`"default"`), a wide or long numeric `data.frame` (`"data.frame"`, `"long"`), or publication-ready tables (`"tinytable"`, `"gt"`, `"flextable"`, `"excel"`, `"clipboard"`, `"word"`).

Usage

```
table_categorical(
  data,
  select,
  by = NULL,
  labels = NULL,
  levels_keep = NULL,
  include_total = TRUE,
  drop_na = TRUE,
  weights = NULL,
  rescale = FALSE,
  correct = FALSE,
  simulate_p = FALSE,
  simulate_B = 2000,
  percent_digits = 1,
  p_digits = 3,
  v_digits = 2,
  assoc_measure = "auto",
  assoc_ci = FALSE,
  decimal_mark = ".",
  align = c("decimal", "auto", "center", "right"),
 output = c("default", "data.frame", "long", "tinytable", "gt", "flextable", "excel",
    "clipboard", "word"),
  indent_text = "  ",
  indent_text_excel_clipboard = strrep(" ", 6),
  add_multilevel_header = TRUE,
  blank_na_wide = FALSE,
  excel_path = NULL,
  excel_sheet = "Categorical",
  clipboard_delim = "\t",
  word_path = NULL
)
```

Arguments

data	A data frame.
select	Columns to include as row variables. Supports tidyselect syntax and character vectors of column names.
by	Optional grouping column used for columns/groups. Accepts an unquoted column name or a single character column name.
labels	Optional display labels for the variables. Two forms are accepted (matching table_continuous() and table_continuous_lm()): • A **named character vector** whose names match column names in data (e.g. c(bmi = "Body mass index")); only listed columns are relabelled, others fall back to attribute-based labels or the column name. **Recommended form**. • A **positional character vector** of the same length as select, in the same order. Backward-compatible with the spicy < 0.11.0 API. When NULL (the default), column names are used as-is. If a variable label attribute is present (e.g. from haven), it is *not* picked up here – pass labels = c(...) explicitly. (The continuous companions auto-detect attribute labels; the categorical function is conservative because the indented row labels expect predictable text.)
levels_keep	Optional character vector of levels to keep/order for row modalities. If NULL, all observed levels are kept.
include_total	Logical. If TRUE (the default), includes a Total group when available.
drop_na	Logical. If TRUE (the default), removes rows with NA in the row/group variable before each cross-tabulation. If FALSE, missing values are displayed as a dedicated "(Missing)" level.
weights	Optional weights. Either NULL (the default), a numeric vector of length nrow(data), or a single column in data supplied as an unquoted name or a character string.
rescale	Logical. If FALSE (the default), weights are used as-is. If TRUE, rescales weights so total weighted N matches raw N. Passed to spicy::cross_tab().
correct	Logical. If FALSE (the default), no continuity correction is applied. If TRUE, applies Yates correction in 2x2 chi-squared contexts. Passed to spicy::cross_tab().
simulate_p	Logical. If FALSE (the default), uses asymptotic p-values. If TRUE, uses Monte Carlo simulation. Passed to spicy::cross_tab().
simulate_B	Integer. Number of Monte Carlo replicates when simulate_p = TRUE. Defaults to 2000.
percent_digits	Number of digits for percentages in report outputs. Defaults to 1.
p_digits	Number of digits for p-values (except < .001). Defaults to 3.
v_digits	Number of digits for the association measure. Defaults to 2.
assoc_measure	Which association measure to report alongside the chi-squared *p*-value. Accepts four input shapes: • "none" — drop the column entirely.

- "auto" (the default) — pick a measure per row variable based on the variable type: a 2x2 table (binary row variable vs. binary by) uses phi, a pair of ordered factors uses tau_b, every other case uses cramer_v.
- a single string from c("cramer_v", "phi", "gamma", "tau_b", "tau_c", "somers_d", "lambda") — applied uniformly to every row variable.
- a character vector with one entry per row variable. Both **named** (c(smoking = "phi", health = "tau_b"), recommended; unnamed variables fall back to "auto") and **unnamed** positional (c("phi", "tau_b", "auto"), paired up with select) are accepted. Named is more robust to reordering of select.

When a single measure is used for every row, the column header is that measure's name (e.g. "Cramer's V"). When multiple measures are used (typically with "auto" on a heterogeneous select), the header collapses to "Effect size" and an APA-style Note. line is appended documenting which measure was used for which variable.

phi requires a 2x2 table; if explicitly requested for a non-2x2 variable, an error is raised so the user can choose another measure or fall back to "auto".

assoc_ci — Passed to cross_tab(). If TRUE, includes the confidence interval of the association measure. In wide raw outputs ("data.frame", "excel", "clipboard"), two extra columns CI lower / CI upper are added; in the long raw output ("long") the bounds appear as ci_lower / ci_upper. In rendered formats ("gt", "tinytable", "flextable", "word"), the CI is shown inline (e.g., .14 [.08, .19]). Defaults to FALSE.

decimal_mark — Decimal separator (".", ","). Defaults to ".".

align — Horizontal alignment of numeric columns in the printed ASCII table and in the tinytable, gt, flextable, word, and clipboard outputs. The first column (Variable) is always left-aligned. One of:

- "decimal" (default): align numeric columns on the decimal mark, the standard scientific-publication convention used by SPSS, SAS, LaTeX siunitx, and the native primitives of gt::cols_align_decimal() and tinytable::style_tt(align = "d"). For engines without a native primitive (flextable, word, clipboard, ASCII print), numeric cells are pre-padded with leading and trailing spaces so the dots line up vertically; the body of the flextable/word output additionally uses a monospace font (Consolas) to make character widths uniform.
- "center": center-align all numeric columns.
- "right": right-align all numeric columns.
- "auto": legacy uniform right-alignment used in spicy < 0.11.0.

The excel output uses the engine's default alignment in any case: cell-string padding does not align decimals under proportional fonts, and Excel's native right-alignment combined with the per-column numfmt already produces dot-aligned columns. Same default and semantics as table_continuous() / table_continuous_lm().

output — Output format. One of:

- "default" (a printed ASCII table, returned invisibly)
- "data.frame" (a wide numeric data.frame)

- "long" (a long numeric data.frame)
- "tinytable" (requires tinytable)
- "gt" (requires gt)
- "flextable" (requires flextable)
- "excel" (requires openxlsx2)
- "clipboard" (requires clipr)
- "word" (requires flextable and officer)

indent_text Prefix used for modality labels in report table building. Defaults to " " (two spaces).

indent_text_excel_clipboard
Stronger indentation used in Excel and clipboard exports. Defaults to six non-breaking spaces.

add_multilevel_header
Logical. If TRUE (the default), merges top headers in Excel export.

blank_na_wide Logical. If FALSE (the default), NA values are kept as-is in wide raw output. If TRUE, replaces them with empty strings.

excel_path Path for output = "excel". Defaults to NULL.

excel_sheet Sheet name for Excel export. Defaults to "Categorical".

clipboard_delim
Delimiter for clipboard text export. Defaults to "\t".

word_path Path for output = "word" or optional save path when output = "flextable". Defaults to NULL.

Value

Depends on output:

- "default": prints a styled ASCII table and returns the underlying data.frame invisibly (S3 class "spicy_categorical_table").
- "data.frame": a wide data.frame with one row per variable–level combination. When by is used, the columns are Variable, Level, and one pair of n / \% columns per group level (plus Total when include_total = TRUE), followed by Chi2, df, p, and the association measure column. When by = NULL, the columns are Variable, Level, n, \%.
- "long": a long data.frame with columns variable, level, group, n, percent (and chi2, df, p, association measure columns when by is used).
- "tinytable": a tinytable object.
- "gt": a gt_tbl object.
- "flextable": a flextable object.
- "excel" / "word": writes to disk and returns the file path invisibly.
- "clipboard": copies the table and returns the display data.frame invisibly.

Tests

When by is used, each selected variable is cross-tabulated against the grouping variable with cross_tab(). The omnibus chi-squared test (with optional Yates continuity correction or Monte Carlo *p*-value, see correct / simulate_p) is computed and reported in the p column. The chosen association measure (assoc_measure, with "auto" selecting Cramer's V for nominal variables and Kendall's Tau-b when both are ordered) is reported alongside, with optional CI via assoc_ci. Without by, the table reports the marginal frequency distribution of each variable with no inferential statistics.

For model-based comparisons (cluster-robust SE, weighted contrasts, fitted means) on continuous outcomes, see table_continuous_lm(). For descriptive (empirical) comparisons on continuous outcomes, see table_continuous().

Display conventions

By default (align = "decimal") numeric columns are aligned on the decimal mark, the standard scientific-publication convention used by SPSS, SAS, LaTeX siunitx, and the native primitives of gt::cols_align_decimal() / tinytable::style_tt(align = "d"). For the printed ASCII table the alignment is achieved by padding numeric cells with leading and trailing spaces so dots line up vertically. Pass align = "auto" to revert to the legacy uniform right-alignment used in spicy < 0.11.0.

p-values are formatted with p_digits decimal places (default 3, the APA standard). Leading zeros on *p* are always stripped (.045, not 0.045).

Optional output engines require suggested packages:

- **tinytable** for output = "tinytable"
- **gt** for output = "gt"
- **flextable** for output = "flextable"
- **flextable** + **officer** for output = "word"
- **openxlsx2** for output = "excel"
- **clipr** for output = "clipboard"

See Also

table_continuous() for empirical comparisons on continuous outcomes; table_continuous_lm() for the model-based companion (heteroskedasticity-consistent / cluster-robust / bootstrap / jackknife SE, fitted means, weighted contrasts); cross_tab() for two-way cross-tabulations; freq() for one-way frequency tables.

Other spicy tables: spicy_tables, table_continuous(), table_continuous_lm()

Examples

```
# --- Basic usage ------------------------------------------------------------

# Default: ASCII console table grouped by sex.
table_categorical(
  sochealth,
  select = c(smoking, physical_activity),
  by = sex
```

```
)

# One-way frequency-style table (no `by`).
table_categorical(
  sochealth,
  select = c(smoking, physical_activity)
)

# Pretty labels keyed by column name.
table_categorical(
  sochealth,
  select = c(smoking, physical_activity),
  by = education,
  labels = c(
    smoking           = "Current smoker",
    physical_activity = "Physical activity"
  )
)

# Survey weights with rescaling.
table_categorical(
  sochealth,
  select = c(smoking, physical_activity),
  by = education,
  weights = "weight",
  rescale = TRUE
)

# Confidence interval for the association measure.
table_categorical(
  sochealth,
  select = smoking,
  by = education,
  assoc_ci = TRUE
)

# --- Per-variable association measure ---------------------------------

# Default (`assoc_measure = "auto"`): one measure per row variable based on
# the variable type (2x2 -> Phi, both ordered factors -> Kendall's Tau-b,
# otherwise Cramer's V). When the chosen measures differ across rows, the
# column header collapses to `"Effect size"` and an APA-style `Note.` line
# documents which measure was used for which variable.
table_categorical(
  sochealth,
  select = c(smoking, education),
  by = sex
)

# Force a uniform measure across all row variables.
table_categorical(
  sochealth,
  select = c(smoking, education),
```

```
  by = sex,
  assoc_measure = "cramer_v"
)

# Per-variable override (recommended named form).
table_categorical(
  sochealth,
  select = c(smoking, education, self_rated_health),
  by = sex,
  assoc_measure = c(
    smoking           = "phi",        # binary x binary
    education         = "cramer_v",   # multi-category nominal
    self_rated_health = "tau_b"       # ordinal x binary, Tau-b
  )
)

# --- Output formats ------------------------------------------------

# The rendered outputs below all wrap the same call:
#   table_categorical(sochealth,
#                     select = c(smoking, physical_activity),
#                     by = sex)
# only `output` changes. Assign to a variable to avoid the
# console-friendly text fallback that some engines fall back to
# when printed directly in `?` help.

# Wide data.frame (one row per modality).
table_categorical(
  sochealth,
  select = c(smoking, physical_activity),
  by = sex,
  output = "data.frame"
)

# Long data.frame (one row per (modality x group)).
table_categorical(
  sochealth,
  select = c(smoking, physical_activity),
  by = sex,
  output = "long"
)


# Rendered HTML / docx objects -- best viewed inside a
# Quarto / R Markdown document or a pkgdown article.
if (requireNamespace("tinytable", quietly = TRUE)) {
  tt <- table_categorical(
    sochealth, select = c(smoking, physical_activity), by = sex,
    output = "tinytable"
  )
}
if (requireNamespace("gt", quietly = TRUE)) {
  tbl <- table_categorical(
```

```
    sochealth, select = c(smoking, physical_activity), by = sex,
    output = "gt"
  )
}
if (requireNamespace("flextable", quietly = TRUE)) {
  ft <- table_categorical(
    sochealth, select = c(smoking, physical_activity), by = sex,
    output = "flextable"
  )
}

# Excel and Word: write to a temporary file.
if (requireNamespace("openxlsx2", quietly = TRUE)) {
  tmp <- tempfile(fileext = ".xlsx")
  table_categorical(
    sochealth, select = c(smoking, physical_activity), by = sex,
    output = "excel", excel_path = tmp
  )
  unlink(tmp)
}
if (
  requireNamespace("flextable", quietly = TRUE) &&
    requireNamespace("officer", quietly = TRUE)
) {
  tmp <- tempfile(fileext = ".docx")
  table_categorical(
    sochealth, select = c(smoking, physical_activity), by = sex,
    output = "word", word_path = tmp
  )
  unlink(tmp)
}


## Not run:
# Clipboard: writes to the system clipboard.
table_categorical(
  sochealth, select = c(smoking, physical_activity), by = sex,
  output = "clipboard"
)

## End(Not run)
```

---

table_continuous *Continuous summary table*

Description

Computes descriptive statistics (mean, SD, min, max, confidence interval of the mean, *n*) for one or many continuous variables selected with tidyselect syntax.

With by, produces grouped summaries and reports a group-comparison *p*-value by default (Welch test; change via test). Additional inferential output is opt-in: test statistics (statistic) and effect sizes (effect_size / effect_size_ci). Set p_value = FALSE to suppress the *p*-value column. Without by, produces one-way descriptive summaries.

Multiple output formats are available via output: a printed ASCII table ("default"), a plain data.frame ("data.frame" or "long" – synonyms for the underlying long-format data, see Details), or publication-ready tables ("tinytable", "gt", "flextable", "excel", "clipboard", "word").

This is the descriptive companion to table_continuous_lm(). The two functions share their argument vocabulary (select, by, weights / vcov exclusively in the model variant, effect_size, ci_level, digits, p_digits, decimal_mark, align, ...) so a descriptive analysis and a model-based analysis of the same data use the same table layout, decimal mark, and reporting precision.

Usage

```
table_continuous(
  data,
  select = tidyselect::everything(),
  by = NULL,
  exclude = NULL,
  regex = FALSE,
  test = c("welch", "student", "nonparametric"),
  p_value = NULL,
  statistic = FALSE,
  show_n = TRUE,
  effect_size = c("none", "auto", "hedges_g", "eta_sq", "r_rb", "epsilon_sq"),
  effect_size_ci = FALSE,
  ci = TRUE,
  labels = NULL,
  ci_level = 0.95,
  digits = 2,
  effect_size_digits = 2,
  p_digits = 3,
  decimal_mark = ".",
  align = c("decimal", "auto", "center", "right"),
 output = c("default", "data.frame", "long", "tinytable", "gt", "flextable", "excel",
    "clipboard", "word"),
  excel_path = NULL,
  excel_sheet = "Descriptives",
  clipboard_delim = "\t",
  word_path = NULL,
  verbose = FALSE
)
```

Arguments

data	A data.frame.

`select` Columns to include. If `regex = FALSE`, use tidyselect syntax or a character vector of column names (default: `tidyselect::everything()`). If `regex = TRUE`, provide a regular expression pattern (character string).

`by` Optional grouping column. Accepts an unquoted column name or a single character column name. The column does not need to be numeric.

`exclude` Columns to exclude. Supports tidyselect syntax and character vectors of column names.

`regex` Logical. If `FALSE` (the default), uses tidyselect helpers. If `TRUE`, the `select` argument is treated as a regular expression.

`test` Character. Statistical test to use when comparing groups. One of `"welch"` (default), `"student"`, or `"nonparametric"`.

- `"welch"`: Welch *t*-test (2 groups) or Welch one-way ANOVA (3+ groups). Does not assume equal variances.
- `"student"`: Student *t*-test (2 groups) or classic one-way ANOVA (3+ groups). Assumes equal variances.
- `"nonparametric"`: Wilcoxon rank-sum / Mann–Whitney U (2 groups) or Kruskal–Wallis H (3+ groups).

Used whenever `by` is supplied (since `p_value` defaults to `TRUE` in that case) or when `statistic = TRUE` / `effect_size = TRUE`. Ignored when `by` is not used, or when all three display toggles are turned off.

`p_value` Logical or `NULL`. If `TRUE` and `by` is used, adds a *p*-value column from the test specified by `test`. When `NULL` (the default), the *p*-value is shown automatically whenever `by` is supplied, and hidden otherwise. Pass `p_value = FALSE` to suppress the column explicitly. Ignored when `by` is not used.

`statistic` Logical. If `TRUE` and `by` is used, the test statistic is shown in an additional column (e.g., `t(df) = ...`, `F(df1, df2) = ...`, `W = ...`, or `H(df) = ...`). Both `p_value` and `statistic` are independent; either or both can be enabled. Defaults to `FALSE`. Ignored when `by` is not used.

`show_n` Logical. If `TRUE`, includes an unweighted `n` column in the printed ASCII table and in every rendered output (`tinytable`, `gt`, `flextable`, `word`, `excel`, `clipboard`). Set to `FALSE` to drop the `n` column structurally from those outputs (no empty placeholder, no spanner). The `n` column is always present in the raw `output = "data.frame"` / `"long"` for downstream programmatic access. Defaults to `TRUE`.

`effect_size` Effect-size measure to include in the rendered outputs. One of:

- `"none"` (default): no effect-size column.
- `"auto"`: auto-select the canonical measure for the chosen `test` and group count – Hedges' g (parametric, 2 groups), eta-squared (parametric, 3+ groups), rank-biserial r (nonparametric, 2 groups), epsilon-squared (nonparametric, 3+ groups). This is the historical behaviour of `effect_size = TRUE`.
- `"hedges_g"`: Hedges' g (bias-corrected standardised mean difference, 2 groups, parametric). CI via the Hedges & Olkin normal approximation.
- `"eta_sq"`: Eta-squared (η^2, parametric ANOVA-style `SS_between / SS_total`). CI via inversion of the noncentral F distribution.

- "r_rb": Rank-biserial r from the Wilcoxon / Mann-Whitney statistic (2 groups, nonparametric). CI via Fisher z-transform.
- "epsilon_sq": Epsilon-squared (ε^2) from the Kruskal-Wallis statistic (3+ groups, nonparametric). CI via percentile bootstrap (2 000 replicates).

For backward compatibility, effect_size = TRUE is silently coerced to "auto" and effect_size = FALSE to "none". Explicit choices are validated against the active test and the number of groups; an incompatible request (e.g. "eta_sq" with two groups, or "hedges_g" with test = "nonparametric") triggers an actionable error. Ignored when by is not used.

effect_size_ci
: Logical. If TRUE, appends the confidence interval of the effect size in brackets (e.g., g = 0.45 [0.22, 0.68]). Implies a non-"none" effect size; if effect_size = "none" is left unchanged, this argument is ignored with a warning, and the function falls back to effect_size = "auto". Defaults to FALSE.

ci
: Logical. If TRUE, includes the mean confidence interval columns (<level>% CI LL / <level>% CI UL) and their spanner in the printed ASCII table and in every rendered output (tinytable, gt, flextable, word, excel, clipboard). Set to FALSE to drop both columns and the CI spanner structurally from those outputs (no empty placeholders, no border lines under an empty header). The CI bounds are always present as ci_lower / ci_upper in the raw output = "data.frame" / "long" for downstream programmatic access. Defaults to TRUE. The CI level is taken from ci_level.

labels
: An optional named character vector of variable labels. Names must match column names in data. When NULL (the default), labels are auto-detected from variable attributes (e.g., haven labels); if none are found, the column name is used.

ci_level
: Confidence level for the mean confidence interval (default: 0.95). Must be between 0 and 1 exclusive.

digits
: Number of decimal places for descriptive values and test statistics (default: 2).

effect_size_digits
: Number of decimal places for effect-size values in formatted displays (default: 2).

p_digits
: Integer >= 1. Number of decimal places used to render p-values in the p column (default: 3, the APA Publication Manual standard). Both the displayed precision and the small-p threshold derive from this argument: p_digits = 3 prints .045 and <.001; p_digits = 4 prints .0451 and <.0001; p_digits = 2 prints .05 and <.01. Useful for genomics / GWAS contexts with very small p-values, or for journals using a coarser convention. Leading zeros are always stripped, following APA convention.

decimal_mark
: Character used as decimal separator. Either "." (default) or ",".

align
: Horizontal alignment of numeric columns in the printed ASCII table and in the tinytable, gt, flextable, word, and clipboard outputs. The first column (Variable) and Group (when present) are always left-aligned. One of:

- "decimal" (default): align numeric columns on the decimal mark, the standard scientific-publication convention used by SPSS, SAS, LaTeX siunitx, gt::cols_align_decimal() and tinytable::style_tt(align = "d").

For engines without a native decimal-alignment primitive (`flextable`, `word`, `clipboard`, ASCII print), values are pre-padded with leading and trailing spaces so the dots line up vertically; the body of the `flextable`/`word` output additionally uses a monospace font to make character widths uniform.

- `"center"`: center-align all numeric columns.
- `"right"`: right-align all numeric columns.
- `"auto"`: legacy per-column rule (center for the descriptive columns, right for `n` and `p`).

The `excel` output uses the engine's default alignment in any case: cell-string padding does not align decimals under proportional fonts. Same default and semantics as `table_continuous_lm()`.

`output` Output format. One of:

- `"default"`: a printed ASCII table, returned invisibly.
- `"data.frame"` / `"long"`: a plain `data.frame` with one row per (`variable` x `group`) (or one row per `variable` when `by` is not used). The two names are synonyms; pick whichever reads better in your pipeline (`"long"` matches `table_continuous_lm()`'s naming).
- `"tinytable"` (requires `tinytable`)
- `"gt"` (requires `gt`)
- `"flextable"` (requires `flextable`)
- `"excel"` (requires `openxlsx2`)
- `"clipboard"` (requires `clipr`)
- `"word"` (requires `flextable` and `officer`)

`excel_path` File path for `output = "excel"`.

`excel_sheet` Sheet name for `output = "excel"` (default: `"Descriptives"`).

`clipboard_delim` Delimiter for `output = "clipboard"` (default: `"\t"`).

`word_path` File path for `output = "word"`.

`verbose` Logical. If `TRUE`, prints messages about excluded non-numeric columns (default: `FALSE`).

Value

Depends on `output`:

- `"default"`: prints a styled ASCII table and returns the underlying `data.frame` invisibly (S3 class `"spicy_continuous_table"` / `"spicy_table"`). The object can be re-coerced via `as.data.frame.spicy_continuous_table()` or piped into `broom::tidy()` / `broom::glance()`.
- `"data.frame"` / `"long"`: a plain `data.frame` with columns `variable`, `label`, `group` (when `by` is used), `mean`, `sd`, `min`, `max`, `ci_lower`, `ci_upper`, `n`. When `by` is used together with `p_value = TRUE`, `statistic = TRUE`, or `effect_size != "none"`, additional columns are appended (populated on the first row of each variable block only):
  - `test_type` – test identifier (e.g., `"welch_t"`, `"welch_anova"`, `"student_t"`, `"anova"`, `"wilcoxon"`, `"kruskal"`).

  - `statistic`, `df1`, `df2`, `p.value` – test results.
  - `es_type` – effect-size identifier (`"hedges_g"`, `"eta_sq"`, `"r_rb"`, or `"epsilon_sq"`), when `effect_size != "none"`.
  - `es_value`, `es_ci_lower`, `es_ci_upper` – effect-size estimate and confidence interval bounds.

  The two names `"data.frame"` and `"long"` are synonyms (the descriptive output is naturally already long). Pick whichever reads better in your code.

- `"tinytable"`: a `tinytable` object.
- `"gt"`: a `gt_tbl` object.
- `"flextable"`: a `flextable` object.
- `"excel"` / `"word"`: writes to disk and returns the file path invisibly.
- `"clipboard"`: copies the table and returns the display `data.frame` invisibly.

Tests

The omnibus test is computed only when `by` is supplied and at least two groups have two or more observations. Choice driven by `test`:

- `"welch"` (default): Welch *t*-test for two groups (`stats::t.test(var.equal = FALSE)`); Welch one-way ANOVA for three or more (`stats::oneway.test(var.equal = FALSE)`). Does not assume equal variances.
- `"student"`: Student *t*-test (`var.equal = TRUE`) / classical ANOVA (`stats::oneway.test(var.equal = TRUE)`).
- `"nonparametric"`: Wilcoxon rank-sum / Mann-Whitney *U* for two groups (`stats::wilcox.test`); Kruskal-Wallis *H* for three or more (`stats::kruskal.test`).

For model-based contrasts (heteroskedasticity-consistent SE, cluster-robust SE, weighted contrasts, fitted means, etc.), use `table_continuous_lm()`.

Effect sizes

Effect size is selected via `effect_size`. The default is `"none"` (no column). `"auto"` mirrors the historical `effect_size = TRUE` behaviour and chooses the canonical measure for the active (`test`, `n_groups`) combination:

- Parametric, 2 groups -> Hedges' g (Hedges & Olkin 1985).
- Parametric, 3+ groups -> Eta-squared (η^2).
- Nonparametric, 2 groups -> Rank-biserial r.
- Nonparametric, 3+ groups -> Epsilon-squared (ε^2).

Explicit choices (`"hedges_g"`, `"eta_sq"`, `"r_rb"`, `"epsilon_sq"`) are validated against (`test`, `n_groups`); an incompatible request triggers a clear error rather than a silent fallback. The model-based companion `table_continuous_lm()` adds Cohen's d, Hays' ω^2, and Cohen's f^2, all derived from the fitted (possibly weighted) `lm()`. CIs are available via `effect_size_ci = TRUE`: noncentral F inversion for η^2, Hedges-Olkin normal approximation for g, Fisher z-transform for r, and percentile bootstrap (2 000 replicates) for ε^2.

Display conventions

By default (`align = "decimal"`) numeric columns are aligned on the decimal mark, the standard scientific-publication convention used by SPSS, SAS, LaTeX `siunitx`, and the native primitives of `gt::cols_align_decimal()` / `tinytable::style_tt(align = "d")`. For engines without a native primitive (`flextable`, `word`, `clipboard`, ASCII print), values are pre-padded with leading and trailing spaces so dots line up vertically; `flextable`/`word` additionally use a monospace font in the body. Pass `align = "auto"` to revert to the legacy per-column rule (centre for the descriptive columns, right for `n` and `p`).

p-values are formatted with `p_digits` decimal places (default 3, the APA standard). The threshold below which the column shows `<.001` is `10^{-p_digits}`; setting `p_digits = 4` shifts both the displayed precision and the threshold accordingly. Leading zeros on *p* are always stripped (`.045`, not `0.045`).

Non-numeric columns are silently dropped (set `verbose = TRUE` to see which columns were excluded). When a single constant column is passed, SD and CI are shown as `"--"` in the ASCII table.

Optional output engines require suggested packages:

- **tinytable** for `output = "tinytable"`
- **gt** for `output = "gt"`
- **flextable** for `output = "flextable"`
- **flextable** + **officer** for `output = "word"`
- **openxlsx2** for `output = "excel"`
- **clipr** for `output = "clipboard"`

See Also

`table_continuous_lm()` for the model-based companion (heteroskedasticity-consistent SE, cluster-robust SE, weighted contrasts, fitted means); `table_categorical()` for categorical variables; `freq()` for one-way frequency tables; `cross_tab()` for two-way cross-tabulations.

Other spicy tables: `spicy_tables`, `table_categorical()`, `table_continuous_lm()`

Examples

```
# --- Basic usage ------------------------------------------------------------

# Default: ASCII console table.
table_continuous(
  sochealth,
  select = c(bmi, wellbeing_score)
)

# Grouped by education (Welch p-value added by default).
table_continuous(
  sochealth,
  select = c(bmi, wellbeing_score),
  by = education
)
```

```
# Test statistic alongside the p-value.
table_continuous(
  sochealth,
  select = c(bmi, wellbeing_score),
  by = education,
  statistic = TRUE
)

# --- Effect sizes ------------------------------------------------------

# Auto-selected effect size with confidence interval (Hedges' g for
# binary `by`, eta-squared for k > 2).
table_continuous(
  sochealth,
  select = wellbeing_score,
  by = sex,
  effect_size = "auto",
  effect_size_ci = TRUE
)

# Explicit effect-size measure.
table_continuous(
  sochealth,
  select = wellbeing_score,
  by = education,
  effect_size = "eta_sq",
  effect_size_ci = TRUE,
  effect_size_digits = 3
)

# --- Selection helpers -------------------------------------------------

# Regex selection.
table_continuous(
  sochealth,
  select = "^life_sat",
  regex = TRUE
)

# Pretty labels keyed by column name.
table_continuous(
  sochealth,
  select = c(bmi, life_sat_health),
  labels = c(
    bmi = "Body mass index",
    life_sat_health = "Satisfaction with health"
  )
)

# --- Output formats ----------------------------------------------------

# The rendered outputs below all wrap the same call:
```

```
#   table_continuous(sochealth,
#                    select = c(bmi, wellbeing_score),
#                    by = sex)
# only `output` changes. Assign to a variable to avoid the
# console-friendly text fallback that some engines fall back to
# when printed directly in `?` help.

# Wide / long data.frame (synonyms): one row per (variable x group).
table_continuous(
  sochealth,
  select = c(bmi, wellbeing_score),
  by = sex,
  output = "data.frame"
)


# Rendered HTML / docx objects -- best viewed inside a
# Quarto / R Markdown document or a pkgdown article.
if (requireNamespace("tinytable", quietly = TRUE)) {
  tt <- table_continuous(
    sochealth, select = c(bmi, wellbeing_score), by = sex,
    output = "tinytable"
  )
}
if (requireNamespace("gt", quietly = TRUE)) {
  tbl <- table_continuous(
    sochealth, select = c(bmi, wellbeing_score), by = sex,
    output = "gt"
  )
}
if (requireNamespace("flextable", quietly = TRUE)) {
  ft <- table_continuous(
    sochealth, select = c(bmi, wellbeing_score), by = sex,
    output = "flextable"
  )
}

# Excel and Word: write to a temporary file.
if (requireNamespace("openxlsx2", quietly = TRUE)) {
  tmp <- tempfile(fileext = ".xlsx")
  table_continuous(
    sochealth, select = c(bmi, wellbeing_score), by = sex,
    output = "excel", excel_path = tmp
  )
  unlink(tmp)
}
if (
  requireNamespace("flextable", quietly = TRUE) &&
    requireNamespace("officer", quietly = TRUE)
) {
  tmp <- tempfile(fileext = ".docx")
  table_continuous(
    sochealth, select = c(bmi, wellbeing_score), by = sex,
```

```
    output = "word", word_path = tmp
  )
  unlink(tmp)
}


## Not run:
# Clipboard: writes to the system clipboard.
table_continuous(
  sochealth, select = c(bmi, wellbeing_score), by = sex,
  output = "clipboard"
)

## End(Not run)
```

`table_continuous_lm` *Continuous-outcome linear-model table*

Description

Builds APA-style summary tables from a series of simple linear models for one or many continuous outcomes selected with tidyselect syntax.

A single predictor is supplied with `by`, and each selected numeric outcome is fit as `lm(outcome ~ by, ...)`. When `by` is categorical, the function returns a model-based mean-comparison table with fitted means by level derived from the linear model, plus an optional single difference for dichotomous predictors. When `by` is numeric, the table reports the slope and its confidence interval.

Multiple output formats are available via `output`: a printed ASCII table (`"default"`), a plain wide `data.frame` (`"data.frame"`), a raw long `data.frame` (`"long"`), or rendered outputs (`"tinytable"`, `"gt"`, `"flextable"`, `"excel"`, `"clipboard"`, `"word"`).

Usage

```
table_continuous_lm(
  data,
  select = tidyselect::everything(),
  by,
  exclude = NULL,
  regex = FALSE,
  weights = NULL,
  vcov = c("classical", "HC0", "HC1", "HC2", "HC3", "HC4", "HC4m", "HC5", "CR0", "CR1",
    "CR2", "CR3", "bootstrap", "jackknife"),
  cluster = NULL,
  boot_n = 1000,
  contrast = c("auto", "none"),
  statistic = FALSE,
  p_value = TRUE,
  show_n = TRUE,
```

```
  show_weighted_n = FALSE,
  effect_size = c("none", "f2", "d", "g", "omega2"),
  effect_size_ci = FALSE,
  r2 = c("r2", "adj_r2", "none"),
  ci = TRUE,
  labels = NULL,
  ci_level = 0.95,
  digits = 2,
  fit_digits = 2,
  effect_size_digits = 2,
  p_digits = 3,
  decimal_mark = ".",
  align = c("decimal", "auto", "center", "right"),
 output = c("default", "data.frame", "long", "tinytable", "gt", "flextable", "excel",
    "clipboard", "word"),
  excel_path = NULL,
  excel_sheet = "Linear models",
  clipboard_delim = "\t",
  word_path = NULL,
  verbose = FALSE
)
```

Arguments

data	A data.frame.
select	Outcome columns to include. If regex = FALSE, use tidyselect syntax or a character vector of column names (default: tidyselect::everything()). If regex = TRUE, provide a regular expression pattern (character string).
by	A single predictor column. Accepts an unquoted column name or a single character column name. The predictor can be: • **numeric** (continuous): treated as a covariate. The table reports the slope and its CI from lm(y ~ by). • **factor** or **ordered factor**: treated as categorical. Level order is preserved as declared; the **first level** is the reference for the displayed contrast (R's default treatment-contrast convention). • **character**: coerced to factor with factor(by), which orders the levels alphabetically. To control the reference level, supply by as an explicit factor with the desired level ordering (e.g. via forcats::fct_relevel() or factor(..., levels = ...)). • **logical**: coerced to factor with levels "FALSE", "TRUE" (in that order, since FALSE < TRUE). The reference level is "FALSE", so a binary contrast displays as Delta (TRUE - FALSE). Rows with NA in by are excluded from the analytic sample for each outcome (NAs in y and weights are also excluded; see Details).
exclude	Columns to exclude from select. Supports tidyselect syntax and character vectors of column names.

regex — Logical. If FALSE (the default), uses tidyselect helpers. If TRUE, the select argument is treated as a regular expression.

weights — Optional case weights. Accepts:

- NULL (default): an ordinary unweighted lm() is fit.
- an **unquoted numeric column name** present in data.
- a **single character column name** present in data.
- a **numeric vector of length** nrow(data) evaluated in the calling environment.

Validation: weights must be finite, non-negative, and contain at least one positive value (otherwise the function errors). Rows with NA in weights are excluded from the analytic sample for each outcome, alongside rows with NA in y or by. When supplied, weights are passed to lm(..., weights = ...), so coefficients become weighted least-squares estimates and R^2, adjusted R^2, and the four effect sizes are computed from the corresponding weighted sums of squares (see the *Weights* section in Details).

vcov — Variance estimator used for standard errors, confidence intervals, and Wald test statistics. One of:

- "classical" (default): the ordinary OLS/WLS variance from vcov(lm), which assumes homoscedastic errors.
- "HC0": the original Eicker–White heteroskedasticity-consistent sandwich estimator (White 1980), with no finite-sample correction.
- "HC1": HC0 multiplied by n / (n - p) (MacKinnon and White 1985). Matches Stata's , robust default.
- "HC2": residuals divided by sqrt(1 - h_ii) (MacKinnon and White 1985).
- "HC3": residuals divided by (1 - h_ii) (MacKinnon and White 1985). A common default for small to moderate samples (Long and Ervin 2000).
- "HC4": leverage-adaptive variant designed for influential observations (Cribari-Neto 2004).
- "HC4m": refinement of HC4 with a modified leverage exponent (Cribari-Neto and da Silva 2011).
- "HC5": alternative leverage-adaptive variant designed for leveraged data (Cribari-Neto, Souza and Vasconcellos 2007).
- "CR0", "CR1", "CR2", "CR3": cluster-robust sandwich estimators for non-independent observations (Liang and Zeger 1986); requires cluster. "CR2" is the modern default (Bell and McCaffrey 2002; Pustejovsky and Tipton 2018), with Satterthwaite degrees of freedom for inference; the fractional df is reported in the df2 column and in the t(df) / F(df1, df2) test header. "CR1" corresponds to Stata's , vce(cluster id) default. Cluster-robust variants are dispatched to clubSandwich::vcovCR() and inference uses clubSandwich::coef_test() / clubSandwich::Wald_test(); install clubSandwich to use them.
- "bootstrap": nonparametric (resampling cases) or cluster bootstrap variance, depending on whether cluster is supplied (Davison and Hinkley 1997; Cameron, Gelbach and Miller 2008). The number of replicates is set by boot_n. Inference is asymptotic (z for single contrasts, chi^2(q) for the global Wald test); CIs are Wald-type around the point estimate.

- "jackknife": leave-one-out variance, or leave-one-cluster-out when cluster is supplied (Quenouille 1956; MacKinnon and White 1985). Inference is asymptotic (z / chi^2(q)).

The HC* variants are computed via sandwich::vcovHC(). Coefficients (means, contrasts, slopes), R^2, and the standardized effect sizes (f2, d, g, omega2) are point estimates from the OLS/WLS fit and are not affected by vcov; only their standard errors, CIs, and the test statistic of the contrast change.

cluster — Cluster identifier for cluster-aware variance estimators. Required when vcov is one of the CR* variants; optional and triggers a cluster bootstrap or leave-one-cluster-out jackknife when vcov is "bootstrap" / "jackknife"; forbidden for the other (independent-observation) variants. Accepts:

- NULL (default): no cluster structure.
- an unquoted column name in data.
- a single character column name in data.
- an atomic vector of length nrow(data) evaluated in the calling environment (factor, character, integer, etc.).

Rows with NA in cluster are excluded from the analytic sample for each outcome (alongside rows with NA in y, by, or weights). At least two distinct non-missing cluster values are required. Multi-way clustering (a list / data.frame of multiple cluster vectors) is not supported; use sandwich::vcovCL() or clubSandwich::vcovCR() directly on the fitted model for that case.

boot_n — Integer. Number of bootstrap replicates used when vcov = "bootstrap". Defaults to 1000. Ignored otherwise. Larger values reduce Monte-Carlo error in the bootstrap variance; typical values for inference are 500-2000.

contrast — Contrast display for categorical predictors. One of:

- "auto" (default): show a single reference contrast Delta (level2 - level1) only when by has exactly two non-empty levels. The reference level is the **first level of the factor** (R's default treatment-contrast convention, getOption("contrasts")[1]). To change which level acts as the reference, re-level by upstream (for example with forcats::fct_relevel() or stats::relevel()).
- "none": suppress the contrast column for categorical predictors. Level-specific means are still displayed.

statistic — Logical. If TRUE, includes a test-statistic column in the wide and rendered outputs. Defaults to FALSE.

p_value — Logical. If TRUE, includes a p column in the wide and rendered outputs. Defaults to TRUE.

show_n — Logical. If TRUE, includes an unweighted n column in the wide and rendered outputs. Defaults to TRUE.

show_weighted_n — Logical. If TRUE and weights is supplied, includes a Weighted n column equal to the sum of case weights in the analytic sample. Defaults to FALSE.

effect_size — Character. Effect-size column to include in the wide and rendered outputs. One of:

- "none" (the default): no effect-size column.

- "f2": Cohen's f² = R² / (1 - R²). Defined for any predictor type. Familiar from Cohen (1988); standard input for a-priori power analysis. Note that for a single-predictor model, f² is a monotone transform of R² and adds no information beyond it.
- "d": Cohen's d = beta_hat / sigma_hat, where beta_hat is the model coefficient (the displayed difference) and sigma_hat is the residual standard deviation from the fitted model. Defined only when by has exactly two non-empty levels; otherwise the function errors. The sign matches the displayed Delta (level2 - level1).
- "g": Hedges' g = J * d with the small-sample correction J = 1 - 3 / (4 * df_resid - 1). Same domain as "d".
- "omega2": Hays' omega-squared, a bias-corrected estimator of the population variance explained, less optimistic than R² for small samples. Defined for any predictor type and truncated at 0.

When weights is supplied, "d", "g", and "omega2" are derived from the weighted least-squares fit (using weighted sums of squares and the model's weighted residual standard deviation), keeping them consistent with the weighted contrast and its CI shown in the table. All effect sizes are point estimates derived from the OLS/WLS fit and are **not** affected by vcov.

effect_size_ci Logical. If TRUE and effect_size != "none", adds a confidence interval for the effect size derived from inversion of the appropriate noncentral distribution (noncentral t for "d" / "g"; noncentral F for "omega2" / "f2"). The CI level is taken from ci_level. In the long output (output = "long"), the bounds are always present in es_ci_lower / es_ci_upper (numeric). In the wide raw output (output = "data.frame"), the bounds appear as numeric columns effect_size_ci_lower / effect_size_ci_upper. In the printed ASCII table and rendered outputs ("tinytable", "gt", "flextable", "word", "excel", "clipboard"), the effect-size column shows the value followed by the CI in brackets (e.g. 0.18 [0.07, 0.30]). Defaults to FALSE. When effect_size = "none", this argument is ignored with a warning.

r2 Character. Fit statistic to include in the wide and rendered outputs. One of:

- "r2" (default): the model R² (summary(lm)$r.squared).
- "adj_r2": adjusted R², penalising for df_effect relative to the residual degrees of freedom.
- "none": omit the fit-statistic column.

When weights is supplied, R² and adjusted R² are the weighted least-squares versions reported by summary(lm(..., weights = ...)).

ci Logical. If TRUE, includes contrast confidence-interval columns in the wide and rendered outputs when a single contrast is shown. Defaults to TRUE.

labels An optional named character vector of outcome labels. Names must match column names in data. When NULL (the default), labels are auto-detected from variable attributes; if none are found, the column name is used.

ci_level Confidence level for coefficient and model-based mean intervals (default: 0.95). Must be between 0 and 1 exclusive.

digits Number of decimal places for descriptive values, regression coefficients, and test statistics (default: 2).

`fit_digits` Number of decimal places for model-fit columns (R^2 or adjusted R^2) in wide and rendered outputs (default: 2).

`effect_size_digits` Number of decimal places for the effect-size column (`f2`, `d`, `g`, or `omega2`) in wide and rendered outputs (default: 2).

`p_digits` Integer >= 1. Number of decimal places used to render *p*-values in the `p` column (default: 3, the APA Publication Manual standard). Both the displayed precision and the small-*p* threshold derive from this argument: `p_digits = 3` prints `.045` and `<.001`; `p_digits = 4` prints `.0451` and `<.0001`; `p_digits = 2` prints `.05` and `<.01`. Useful for genomics / GWAS contexts where adjusted *p*-values can be very small, or for journals using a coarser convention. Leading zeros are always stripped, following APA convention.

`decimal_mark` Character used as decimal separator. Either `"."` (default) or `","`.

`align` Horizontal alignment of numeric columns in the printed ASCII table and in the `tinytable`, `gt`, `flextable`, `word`, and `clipboard` outputs. The first column (`Variable`) is always left-aligned. One of:

- `"decimal"` (default): align numeric columns on the decimal mark, the standard scientific-publication convention used by SPSS, SAS, LaTeX `siunitx`, `gt::cols_align_decimal()` and `tinytable::style_tt(align = "d")`. For engines without a native decimal-alignment primitive (`flextable`, `word`, `clipboard`, ASCII print), values are pre-padded with leading and trailing spaces so the dots line up vertically; the body of the `flextable`/`word` output additionally uses a monospace font to make character widths uniform.
- `"center"`: center-align all numeric columns.
- `"right"`: right-align all numeric columns.
- `"auto"`: legacy per-column rule used in spicy < 0.11.0 (center for the descriptive / inferential columns; right for `n`, `Weighted n`, and `p`).

The `excel` output uses the engine's default alignment in any case: cell-string padding does not align decimals under proportional fonts, and writing raw numbers with a numeric format would require a separate refactor.

`output` Output format. One of:

- `"default"`: a printed ASCII table, returned invisibly
- `"data.frame"`: a plain wide `data.frame`
- `"long"`: a raw long `data.frame`
- `"tinytable"` (requires `tinytable`)
- `"gt"` (requires `gt`)
- `"flextable"` (requires `flextable`)
- `"excel"` (requires `openxlsx2`)
- `"clipboard"` (requires `clipr`)
- `"word"` (requires `flextable` and `officer`)

`excel_path` File path for `output = "excel"`.

`excel_sheet` Sheet name for `output = "excel"` (default: `"Linear models"`).

`clipboard_delim` Delimiter for `output = "clipboard"` (default: `"\t"`).

word_path	File path for output = "word".
verbose	Logical. If TRUE, prints messages about ignored non-numeric selected outcomes (default: FALSE).

Value

Depends on output:

- "default": prints a styled ASCII table and invisibly returns the underlying long data.frame with class "spicy_continuous_lm_table" / "spicy_table".
- "data.frame": a plain wide data.frame with one row per outcome and numeric columns for means (categorical by) or slope (numeric by), optional contrast and CI, optional test statistic, p, fit statistic (R^2 or adjusted R^2), effect size, optional effect_size_ci_lower / effect_size_ci_upper (when effect_size_ci = TRUE), n, and Weighted n.
- "long": a raw data.frame with one block per outcome and 28 columns covering identification (variable, label, predictor_type, predictor_label, level, reference), fitted means and their CI (emmean, emmean_se, emmean_ci_lower, emmean_ci_upper), contrast or slope estimates and CI (estimate_type, estimate, estimate_se, estimate_ci_lower, estimate_ci_upper), inferential output (test_type, statistic, df1, df2, p.value), effect size with its CI (es_type, es_value, es_ci_lower, es_ci_upper), fit (r2, adj_r2), and sample size (n, weighted_n).
- "tinytable": a tinytable object.
- "gt": a gt_tbl object.
- "flextable": a flextable object.
- "excel" / "word": writes to disk and returns the file path.
- "clipboard": copies the wide table and returns it invisibly.

If no numeric outcome columns remain after applying select, exclude, and regex, the function emits a warning and returns an empty data.frame() regardless of output.

Model and outputs

table_continuous_lm() is designed for article-style bivariate reporting: a single predictor supplied with by, and one simple model per selected continuous outcome. The model fit is always lm(outcome ~ by, ...), optionally with weights. For categorical predictors, the reported means are model-based fitted means for each level of by, and contrasts are derived from the same fitted linear model. For an unweighted lm(y ~ factor) with classical variance, the fitted means coincide numerically with empirical subgroup means; the *model-based* qualifier matters because (a) under weights the means become weighted least-squares estimates, (b) their CIs are derived from the model vcov (classical or HC*), and (c) tests, *p*-values, and effect sizes all come from the same fitted model, keeping the table internally consistent.

Compared with table_continuous(), this function is the model-based companion: choose it when you want heteroskedasticity-consistent standard errors (vcov = "HC*"), model fit statistics, or case weights via lm(..., weights = ...). Because the function exists to report a fitted model, its inferential output is on by default: p_value = TRUE and r2 = "r2" are the defaults; set p_value = FALSE or r2 = "none" to suppress them.

Effect sizes

Effect size is selected explicitly via `effect_size` (defaults to `"none"`). All variants are derived from the same fitted model as the displayed coefficients, `R²`, and CIs, so the effect size stays internally consistent with the rest of the table.

- `"f2"`: Cohen's `f² = R² / (1 - R²)` (Cohen 1988). Defined for any predictor type. For a single-predictor model, `f²` is a monotone transform of `R²` and adds no information beyond it; its primary use is in *a priori* power analysis (e.g. G*Power).
- `"d"`, `"g"`: standardized mean difference (Cohen's d or Hedges' g), defined only when `by` has exactly two non-empty levels. `d = beta_hat / sigma_hat` with `sigma_hat = summary(fit)$sigma` (the pooled within-group SD for the unweighted two-group case); `g = J * d` with `J = 1 - 3 / (4 * df_resid - 1)` (Hedges and Olkin 1985). The sign matches the displayed `Delta (level2 - level1)`. For published reports of two-group comparisons, g is the convention recommended by Hedges and Olkin (1985).
- `"omega2"`: Hays' ω^2, computed from weighted sums of squares as `(SS_effect - df_effect * MSE) / (SS_total + MSE)` and truncated at 0 for small or null effects (Hays 1963; Olejnik and Algina 2003). Less biased than η^2 (which equals `R^2` in this single-predictor design) and recommended for reporting variance explained in ANOVA-style designs (Olejnik and Algina 2003).

All four effect sizes are point estimates derived from the OLS/WLS fit and are **invariant to** `vcov`: choosing `HC*` changes the SE, CI, and test statistic of the contrast but not the standardized magnitude itself.

Confidence intervals for the effect size are available via `effect_size_ci = TRUE` and use the modern noncentral-distribution inversion approach, the consensus standard in commercial statistical software (Stata `esize` / `estat esize`, SAS `PROC TTEST` and `PROC GLM EFFECTSIZE` 14.2+) and in mainstream R packages (`effectsize`, `MOTE`, `TOSTER`, `effsize`):

- `"d"`, `"g"`: noncentral t inversion (Steiger and Fouladi 1997; Goulet-Pelletier and Cousineau 2018). Empirical coverage is nominal across sample sizes (Cousineau and Goulet-Pelletier 2021), unlike the older Hedges-Olkin normal approximation which is biased for small samples. For Hedges' g the bounds inherit the J small-sample correction.
- `"omega2"`, `"f2"`: noncentral F inversion (Steiger 2004). Bounds are converted from the noncentrality parameter using `omega² = ncp / (ncp + N)` and `f² = ncp / N` respectively, with `N = df1 + df2 + 1` (total sample size).

For the weighted case, the CI uses raw (unweighted) group counts and `df.residual(fit) = n - p`, consistent with the WLS reporting convention (DuMouchel and Duncan 1983). For propensity-score balance assessment or complex-survey designs, dedicated packages (`cobalt::bal.tab()` for the Austin and Stuart 2015 formulation; `survey` for design-based effect sizes) are more appropriate.

Robust standard errors

When `vcov` is one of the `HC*` variants, the standard errors, CIs, and Wald test statistics use a heteroskedasticity-consistent sandwich estimator computed via `sandwich::vcovHC()` (Zeileis 2004), the canonical R implementation. For a brief guide:

- `"HC0"` is the original White (1980) form; `"HC1"` adds the `n / (n - p)` correction (MacKinnon and White 1985), Stata's `, robust` default.

- "HC2" and "HC3" use leverage-based residual rescalings (MacKinnon and White 1985); "HC3" is the sandwich::vcovHC() default for small to moderate samples (Long and Ervin 2000).
- "HC4" adapts the leverage exponent for influential observations (Cribari-Neto 2004); "HC4m" is a modified-exponent refinement (Cribari-Neto and da Silva 2011); "HC5" is an alternative leverage-adaptive variant (Cribari-Neto, Souza and Vasconcellos 2007).

When observations are not independent (repeated measurements per individual, students nested in classes, patients in hospitals, country-year panels), classical and HC* standard errors are biased downward. Use the CR* variants together with cluster = id_var to get cluster-robust inference (Liang and Zeger 1986). The implementation dispatches to clubSandwich::vcovCR() for the variance and to clubSandwich::coef_test() (single-coefficient, Satterthwaite *t*) and clubSandwich::Wald_test() (multi-coefficient Hotelling-T-squared with Satterthwaite df, "HTZ") for inference. "CR2" (Bell and McCaffrey 2002; Pustejovsky and Tipton 2018) is the modern recommended default; it generally produces fractional Satterthwaite degrees of freedom in df2, which the displayed t(df) / F(df1, df2) header renders to one decimal. "CR1" matches Stata's , vce(cluster id). Effect sizes remain invariant to vcov (including CR*); only the SE, CI, test statistic, and df2 of the contrast change.

Two resampling-based estimators are also available without adding any dependency: vcov = "bootstrap" (nonparametric resampling-cases bootstrap; Davison and Hinkley 1997) and vcov = "jackknife" (leave-one-out delete-1; Quenouille 1956; MacKinnon and White 1985). Supplying cluster switches both to their cluster-aware variants (cluster bootstrap, Cameron, Gelbach and Miller 2008; leave-one-cluster-out jackknife). The number of bootstrap replicates is controlled by boot_n (default 1000); replicates that fail to fit on rank-deficient resamples are dropped, with an explicit warning if more than half fail and a fallback to the classical OLS variance below 10 valid replicates. Inference for both estimators is asymptotic (z for single-coefficient contrasts, chi^2(q) for the multi-coefficient global Wald test on k > 2 categorical predictors), reflected in the displayed test header. Use the bootstrap when the residual distribution is non-standard or the sample is small; use the jackknife as a closed-form, deterministic alternative.
R^2, adjusted R^2, and the effect sizes remain ordinary least-squares (or weighted least-squares) statistics regardless of vcov.

Weights

When weights is supplied, table_continuous_lm() fits weighted linear models via lm(..., weights = ...). Means become weighted least-squares estimates and contrasts and slopes are weighted. The fit statistics R^2 and adjusted R^2, as well as Hays' omega^2 and Cohen's f^2, use the corresponding **weighted sums of squares** from the WLS fit. Cohen's d and Hedges' g use the **WLS coefficient and the model's weighted residual standard deviation** (summary(fit)$sigma), which is the standard convention for case-weighted regression-style reporting (DuMouchel and Duncan 1983); the noncentral *t* CI for d / g uses the raw (unweighted) group counts and the residual degrees of freedom of the WLS fit (n − p). This case-weighted workflow is appropriate for weighted article tables, but is **not** a substitute for a full complex-survey design (see e.g. the survey package), nor for propensity-score balance assessment under the Austin and Stuart (2015) convention (see e.g. cobalt::bal.tab()).

The n column always reports the unweighted analytic sample size for each outcome. When show_weighted_n = TRUE, an additional Weighted n column reports the sum of case weights in the same analytic sample.

Display conventions

For dichotomous categorical predictors, the wide outputs report fitted means in reference-level order and label the contrast column explicitly as `Delta (level2 - level1)`. For categorical predictors with more than two levels, no single contrast or contrast CI is shown in the wide outputs; instead, the table reports level-specific means plus the overall `F` test when `statistic = TRUE` (or `F(df1, df2)` when the degrees of freedom are constant across outcomes).

Optional output engines require the corresponding suggested packages:

- **tinytable** for `output = "tinytable"`
- **gt** for `output = "gt"`
- **flextable** for `output = "flextable"`
- **flextable** + **officer** for `output = "word"`
- **openxlsx2** for `output = "excel"`
- **clipr** for `output = "clipboard"`

References

Austin, P. C., & Stuart, E. A. (2015). Moving towards best practice when using inverse probability of treatment weighting (IPTW) using the propensity score to estimate causal treatment effects in observational studies. *Statistics in Medicine*, **34**(28), 3661–3679. doi:10.1002/sim.6607

Bell, R. M., & McCaffrey, D. F. (2002). Bias reduction in standard errors for linear regression with multi-stage samples. *Survey Methodology*, **28**(2), 169–181.

Cameron, A. C., Gelbach, J. B., & Miller, D. L. (2008). Bootstrap-based improvements for inference with clustered errors. *Review of Economics and Statistics*, **90**(3), 414–427. doi:10.1162/rest.90.3.414

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum.

Cousineau, D., & Goulet-Pelletier, J.-C. (2021). Expected and empirical coverages of different methods for generating noncentral *t* confidence intervals for a standardized mean difference. *Behavior Research Methods*, **53**, 2376–2394. doi:10.3758/s13428021015504

Cribari-Neto, F. (2004). Asymptotic inference under heteroskedasticity of unknown form. *Computational Statistics & Data Analysis*, **45**(2), 215–233. doi:10.1016/S01679473(02)003663

Cribari-Neto, F., Souza, T. C., & Vasconcellos, K. L. P. (2007). Inference under heteroskedasticity and leveraged data. *Communications in Statistics – Theory and Methods*, **36**(10), 1877–1888. doi:10.1080/03610920601126589

Cribari-Neto, F., & da Silva, W. B. (2011). A new heteroskedasticity-consistent covariance matrix estimator for the linear regression model. *AStA Advances in Statistical Analysis*, **95**(2), 129–146. doi:10.1007/s1018201001412

Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511802843

DuMouchel, W. H., & Duncan, G. J. (1983). Using sample survey weights in multiple regression analyses of stratified samples. *Journal of the American Statistical Association*, **78**(383), 535–543. doi:10.1080/01621459.1983.10478006

Goulet-Pelletier, J.-C., & Cousineau, D. (2018). A review of effect sizes and their confidence intervals, Part I: The Cohen's *d* family. *The Quantitative Methods for Psychology*, **14**(4), 242–265. doi:10.20982/tqmp.14.4.p242

Hays, W. L. (1963). *Statistics for Psychologists*. New York: Holt, Rinehart and Winston.

Hedges, L. V., & Olkin, I. (1985). *Statistical Methods for Meta-Analysis*. Orlando, FL: Academic Press.

Long, J. S., & Ervin, L. H. (2000). Using heteroscedasticity consistent standard errors in the linear regression model. *The American Statistician*, **54**(3), 217–224. doi:10.1080/00031305.2000.10474549

Liang, K.-Y., & Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, **73**(1), 13–22. doi:10.1093/biomet/73.1.13

MacKinnon, J. G., & White, H. (1985). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, **29**(3), 305–325. doi:10.1016/03044076(85)901587

Olejnik, S., & Algina, J. (2003). Generalized eta and omega squared statistics: Measures of effect size for some common research designs. *Psychological Methods*, **8**(4), 434–447. doi:10.1037/1082989X.8.4.434

Pustejovsky, J. E., & Tipton, E. (2018). Small-sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business & Economic Statistics*, **36**(4), 672–683. doi:10.1080/07350015.2016.1247004

Quenouille, M. H. (1956). Notes on bias in estimation. *Biometrika*, **43**(3/4), 353–360. doi:10.1093/biomet/43.34.353

Steiger, J. H. (2004). Beyond the *F* test: Effect size confidence intervals and tests of close fit in the analysis of variance and contrast analysis. *Psychological Methods*, **9**(2), 164–182. doi:10.1037/1082989X.9.2.164

Steiger, J. H., & Fouladi, R. T. (1997). Noncentrality interval estimation and the evaluation of statistical models. In L. L. Harlow, S. A. Mulaik, & J. H. Steiger (Eds.), *What if there were no significance tests?* (pp. 221–257). Mahwah, NJ: Lawrence Erlbaum.

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, **48**(4), 817–838. doi:10.2307/1912934

Zeileis, A. (2004). Econometric computing with HC and HAC covariance matrix estimators. *Journal of Statistical Software*, **11**(10), 1–17. doi:10.18637/jss.v011.i10

See Also

`table_continuous()`, `table_categorical()`. For broader workflows on the same statistical building blocks: `sandwich::vcovHC()` (the canonical R implementation of the `HC*` sandwich estimators, used internally for `vcov = "HC*"`); `clubSandwich::vcovCR()`, `clubSandwich::coef_test()` and `clubSandwich::Wald_test()` (the canonical R implementation of cluster-robust variance and Satterthwaite-style inference, used internally for `vcov = "CR*"`); `effectsize::cohens_d()`, `effectsize::hedges_g()`, and `effectsize::omega_squared()` (alternative effect-size computations and CIs); `cobalt::bal.tab()` for propensity-score covariate balance with weighted standardized mean differences (Austin and Stuart 2015); the `survey` package for design-based inference on complex-survey samples.

Other spicy tables: `spicy_tables`, `table_categorical()`, `table_continuous()`

Examples

```
# --- Basic usage -----------------------------------------------------------

# Default: ASCII table with model-based means, p, and R².
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex
)

# --- Effect sizes ----------------------------------------------------------

# Cohen's d (binary by required).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  effect_size = "d"
)

# Hedges' g with weighted analysis and weighted n column.
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  weights = weight,
  statistic = TRUE,
  effect_size = "g",
  show_weighted_n = TRUE
)

# Hedges' g with noncentral t confidence interval (bracket notation).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  effect_size = "g",
  effect_size_ci = TRUE
)

# Cohen's f² alongside R² (familiar power-analysis effect size).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  effect_size = "f2"
)

# Hays' omega-squared for a 3-level predictor (d / g would error here).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
```

```
  by = education,
  effect_size = "omega2"
)

# --- Robust SE for a numeric predictor ---------------------------------

# HC3 standard errors for the slope of a continuous predictor.
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = age,
  vcov = "HC3",
  ci = FALSE
)

# Cluster-robust SE for repeated-measures data: the `sleep` dataset
# has 10 subjects measured twice (one observation per group).
table_continuous_lm(
  sleep,
  select = extra,
  by = group,
  cluster = ID,
  vcov = "CR2"
)

# --- Article-style polish ----------------------------------------------

# Pretty outcome labels and adjusted R².
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  labels = c(
    wellbeing_score = "WHO-5 wellbeing (0-100)",
    bmi = "Body-mass index (kg/m²)"
  ),
  r2 = "adj_r2"
)

# European decimal comma.
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  decimal_mark = ","
)

# Regex selection of all columns starting with "life_sat".
table_continuous_lm(
  sochealth,
  select = "^life_sat",
  by = sex,
  regex = TRUE
```

```
)

# --- Output formats ------------------------------------------------------

# The rendered outputs below all wrap the same call:
#   table_continuous_lm(sochealth,
#                       select = c(wellbeing_score, bmi),
#                       by = sex)
# only `output` changes. Assign to a variable to avoid the
# console-friendly text fallback that some engines fall back to
# when printed directly in `?` help.

# Wide data.frame (one row per outcome).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  output = "data.frame"
)

# Raw long data.frame (one block per outcome).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  output = "long"
)


# Rendered HTML / docx objects -- best viewed inside a
# Quarto / R Markdown document or a pkgdown article.
if (requireNamespace("tinytable", quietly = TRUE)) {
  tt <- table_continuous_lm(
    sochealth, select = c(wellbeing_score, bmi), by = sex,
    output = "tinytable"
  )
}
if (requireNamespace("gt", quietly = TRUE)) {
  tbl <- table_continuous_lm(
    sochealth, select = c(wellbeing_score, bmi), by = sex,
    output = "gt"
  )
}
if (requireNamespace("flextable", quietly = TRUE)) {
  ft <- table_continuous_lm(
    sochealth, select = c(wellbeing_score, bmi), by = sex,
    output = "flextable"
  )
}

# Excel and Word: write to a temporary file.
if (requireNamespace("openxlsx2", quietly = TRUE)) {
  tmp <- tempfile(fileext = ".xlsx")
```

```
  table_continuous_lm(
    sochealth, select = c(wellbeing_score, bmi), by = sex,
    output = "excel", excel_path = tmp
  )
  unlink(tmp)
}
if (
  requireNamespace("flextable", quietly = TRUE) &&
    requireNamespace("officer", quietly = TRUE)
) {
  tmp <- tempfile(fileext = ".docx")
  table_continuous_lm(
    sochealth, select = c(wellbeing_score, bmi), by = sex,
    output = "word", word_path = tmp
  )
  unlink(tmp)
}


## Not run:
# Clipboard: writes to the system clipboard.
table_continuous_lm(
  sochealth, select = c(wellbeing_score, bmi), by = sex,
  output = "clipboard"
)

## End(Not run)
```

---

uncertainty_coef *Uncertainty Coefficient*

Description

uncertainty_coef() computes the Uncertainty Coefficient (Theil's U) for a two-way contingency table, based on information entropy.

Usage

```
uncertainty_coef(
  x,
  direction = c("symmetric", "row", "column"),
  detail = FALSE,
  conf_level = 0.95,
  digits = 3L,
  .include_se = FALSE
)
```

Arguments

x	A contingency table (of class table).
direction	Direction of prediction: "symmetric" (default), "row" (column predicts row), or "column" (row predicts column).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

The uncertainty coefficient measures association using Shannon entropy. For direction = "row": $U = (H_X + H_Y - H_{XY})/H_X$, where H_X, H_Y are the marginal entropies and H_{XY} is the joint entropy. The symmetric version is $U = 2(H_X + H_Y - H_{XY})/(H_X + H_Y)$.

The entropy terms use the standard mathematical convention $0 \log 0 = 0$, matching SPSS / PSPP CROSSTABS and the definition in Cover & Thomas (2006). Note that DescTools::UncertCoef() applies an additional Laplace correction (replacing zero cells with $1/n^2$) before the entropy computation, which produces slightly different point estimates on tables with empty cells; that correction is uncommon in the information-theory literature and is not used here. The asymptotic standard errors follow the DescTools delta method; see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: U = 0 (Wald z-test).

See Also

lambda_gk(), goodman_kruskal_tau(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), yule_q()

Examples

```
tab <- table(sochealth$smoking, sochealth$education)
uncertainty_coef(tab)
uncertainty_coef(tab, direction = "row", detail = TRUE)
```

varlist *Generate a comprehensive summary of the variables*

Description

`varlist()` lists the variables of a data frame and extracts essential metadata, including variable names, labels, summary values, classes, number of distinct values, number of valid (non-missing) observations, and number of missing values.

`vl()` is a convenient shorthand for `varlist()` that offers identical functionality with a shorter name.

Usage

```
varlist(
  x,
  ...,
  values = FALSE,
  tbl = FALSE,
  include_na = FALSE,
  factor_levels = c("observed", "all")
)

vl(
  x,
  ...,
  values = FALSE,
  tbl = FALSE,
  include_na = FALSE,
  factor_levels = c("observed", "all")
)
```

Arguments

`x`	A data frame, or a transformation of one.
`...`	Optional tidyselect-style column selectors (e.g. `starts_with("var")`, `where(is.numeric)`, etc.). Columns can be selected or reordered, but renaming selections is not supported.
`values`	Logical. If `FALSE` (the default), displays a compact summary of the variable's values. For numeric, character, date/time, labelled, and factor variables, all unique non-missing values are shown when there are at most four; otherwise the first three values, an ellipsis (`...`), and the last value are shown. Values are sorted when appropriate (e.g., numeric, character, date). For factors, `factor_levels` controls whether observed or all declared levels are shown; level order is preserved. For labelled variables, prefixed labels are displayed via `labelled::to_factor(levels = "prefixed")`. If `TRUE`, all unique non-missing values are displayed.

tbl	Logical. If FALSE (the default), opens the summary in the Viewer if the session is interactive. If TRUE, returns a tibble.
include_na	Logical. If TRUE, unique missing value markers (<NA>, <NaN>) are explicitly appended at the end of the Values summary when present in the variable. This applies to all variable types. Literal strings "NA", "NaN", and "" are quoted to distinguish them from missing markers. If FALSE (the default), missing values are omitted from Values but still counted in the NAs column.
factor_levels	Character. Controls how factor values are displayed in Values. "observed" (the default; code_book() uses "all") shows only levels present in the data, preserving factor level order. "all" shows all declared levels, including unused levels.

Details

The function can also apply tidyselect-style variable selectors to select or reorder columns dynamically.

If used interactively (e.g. in RStudio or Positron), the summary is displayed in the Viewer pane with a contextual title like vl: sochealth. If the data frame has been transformed or subsetted, the title will display an asterisk (*), e.g. vl: sochealth*. Anonymous or ambiguous calls use vl: <data>.

For factor variables, varlist() defaults to displaying only the levels observed in the data (factor_levels = "observed") — a reflection of what is actually present. By contrast, code_book() defaults to "all" to document the declared schema, including unused levels. Pass factor_levels explicitly to override either default.

Value

A tibble with one row per selected variable, containing the following columns:

- Variable: variable names
- Label: variable labels (if available via the label attribute)
- Values: a summary of the variable's values, depending on the values and include_na arguments. If values = FALSE, a compact summary is shown: all unique values when there are at most four, otherwise 3 + ... + last. If values = TRUE, all unique non-missing values are displayed. For labelled variables, **prefixed labels** are displayed using labelled::to_factor(levels = "prefixed"). For factors, levels are displayed according to factor_levels. Matrix and array columns are summarized by their dimensions. Missing value markers (<NA>, <NaN>) are optionally appended at the end (controlled via include_na). Literal strings "NA", "NaN", and "" are quoted to distinguish them from missing markers.
- Class: the class of each variable (possibly multiple, e.g. "labelled", "numeric")
- N_distinct: number of distinct non-missing values
- N_valid: number of non-missing observations
- NAs: number of missing observations

For matrix and array columns, observations are counted per **row**: a row is treated as missing if any of its cells is NA. N_valid / NAs therefore count complete vs. incomplete rows, not individual cells.

If `tbl = TRUE`, the tibble is returned. If `tbl = FALSE` and the session is interactive, the summary is displayed in the Viewer pane and the function returns invisibly. In non-interactive sessions, a message is displayed and the function returns invisibly.

See Also

Other variable inspection: `code_book()`, `label_from_names()`

Examples

```
varlist(sochealth, tbl = TRUE)
sochealth |> varlist(tbl = TRUE)
varlist(sochealth, where(is.numeric), values = TRUE, tbl = TRUE)
varlist(
  sochealth,
  starts_with("bmi"),
  values = TRUE,
  include_na = TRUE,
  tbl = TRUE
)

df <- data.frame(
  group = factor(c("A", "B", NA), levels = c("A", "B", "C"))
)
varlist(
  df,
  values = TRUE,
  include_na = TRUE,
  factor_levels = "all",
  tbl = TRUE
)

vl(sochealth, tbl = TRUE)
sochealth |> vl(tbl = TRUE)
vl(sochealth, starts_with("bmi"), tbl = TRUE)
vl(sochealth, where(is.numeric), values = TRUE, tbl = TRUE)
```

---

yule_q *Yule's Q*

Description

`yule_q()` computes Yule's Q coefficient of association for a 2x2 contingency table.

Usage

```
yule_q(x, detail = FALSE, conf_level = 0.95, digits = 3L, .include_se = FALSE)
```

Arguments

x	A contingency table (of class table).
detail	Logical. If FALSE (default), return the estimate as a numeric scalar. If TRUE, return a named numeric vector including confidence interval and p-value.
conf_level	A number between 0 and 1 giving the confidence level (default 0.95). Only used when detail = TRUE. Set to NULL to omit the confidence interval.
digits	Number of decimal places used when printing the result (default 3). Only affects the detail = TRUE output.
.include_se	Internal parameter; do not use.

Details

For a 2x2 table with cells a, b, c, d, Yule's Q is $Q = (ad - bc)/(ad + bc)$. It is equivalent to the Goodman-Kruskal Gamma for 2x2 tables. The asymptotic standard error is $SE = 0.5(1 - Q^2)\sqrt{1/a + 1/b + 1/c + 1/d}$. Standard error formulas follow the DescTools implementations (Signorell et al., 2024); see cramer_v() for full references.

Value

Same structure as cramer_v(): a scalar when detail = FALSE, a named vector when detail = TRUE. The p-value tests H0: Q = 0 (Wald z-test).

See Also

phi(), gamma_gk(), assoc_measures()

Other association measures: assoc_measures(), contingency_coef(), cramer_v(), gamma_gk(), goodman_kruskal_tau(), kendall_tau_b(), kendall_tau_c(), lambda_gk(), phi(), somers_d(), uncertainty_coef()

Examples

```
tab <- table(sochealth$smoking, sochealth$sex)
yule_q(tab)
```

Index