Package: spicy (via r-universe)

May 20, 2026

Title Descriptive Statistics, Summary Tables, and Data Management Tools

Version 0.12.0

Description Provides tabulation, descriptive-summary, and variable-inspection tools for applied data analysis. Frequency tables and cross-tabulations with contingency-table association measures (Cramer's V, Phi, Goodman-Kruskal Gamma, Kendall's Tau-b, Somers' D, and others); categorical and continuous summary tables; regression coefficient tables for one or more 'lm' or 'glm' fits side by side; and outcome-by-group comparison tables from linear models with optional additive covariate adjustment. All table outputs follow APA conventions and expose 'broom'-compatible 'tidy()' / 'glance()' methods for downstream pipelines. Helpers cover interactive codebooks, variable-label extraction, clipboard export, and row-wise descriptive summaries.

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, emmeans, flextable, gt, haven, htmltools, knitr, marginaleffects, officer, openxlsx2, parameters, performance, rmarkdown, spelling, 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 https://cran.r-universe.dev

Date/Publication 2026-05-19 21:40:02 UTC

RemoteUrl https://github.com/cran/spicy

RemoteRef HEAD

RemoteSha d1c72c4fe78f8b649390a5ad5489b9a7aee27b7b

Contents

as_structured *Extract the typed (structured) view of a* spicy_regression_table

Description

table_regression() returns a *display* representation by default – a character data.frame with stars suffixes, em-dash for reference rows, bracketed "[L, U]" confidence intervals, and APA padding on p-values. This accessor returns the *typed* view that the output engines (Excel, gt, tinytable, flextable, clipboard) consume internally: a fully numeric body with CI pre-split into LL / UL columns, NAs for non-applicable / reference cells, plus per-cell markers and a format specification.

Usage

```
as_structured(x)
```

Arguments

x	A spicy_regression_table object returned by table_regression().

Details

This is the right entry point for users who want to:

- **Filter coefficients programmatically**, e.g. as_structured(tbl)$body[as_structured(tbl)$body$p < 0.05,].
- **Aggregate raw values across rows**, e.g. mean(as_structured(tbl)$body[["B"]], na.rm = TRUE).
- **Build a custom downstream renderer** that consumes the same structured contract as spicy's built-in engines.

Value

A list with the structured view (see Details for the schema).

Schema

- body – data.frame with a Variable character column and one or more numeric columns. Confidence intervals are split into LL / UL columns named like "95% CI: LL" / "95% CI: UL" (or prefixed with the model label in multi-model output). Cells that have no value (reference levels, non-applicable rows in multi-model output, factor headers) are NA.
- reference_rows, factor_header_rows, fit_stat_rows, level_rows, outcome_row – integer row indices.
- col_meta – per-column metadata keyed by structured column name (token, model_id, precision, p-style, below-threshold, CI pair / role / label).
- spanners – named list mapping model labels to their column indices in body (multi-model only).

- `ci_pairs` – list of `(label, cols)` entries describing each CI pair in body.
- `format_spec` – global format defaults (decimal mark, digits, p-style, CI level, etc.).

See Also

`table_regression()` for the user-facing entry point.

Examples

```
fit <- lm(mpg ~ wt + factor(cyl), data = mtcars)
tbl <- table_regression(fit)
s <- as_structured(tbl)
s$body                                # raw numeric body
s$body[s$body$p < 0.05, ]             # filter significant rows
s$col_meta$B                          # column metadata for B
```

---

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`. The `p_value` comes from two test families:

- **Pearson chi-squared test of independence** for Cramer's V, Phi, and the Contingency Coefficient (the three chi-squared-derived nominal measures). All three carry the same chi-squared *p*-value on a given table.
- **Wald z-test of H0: measure = 0** for every other measure: Yule's Q, Lambda, Goodman-Kruskal's Tau, the Uncertainty Coefficient, and all ordinal measures (Gamma, Tau-b, Tau-c, Somers' D).

Direction-dependent measures (`lambda_gk()`, `goodman_kruskal_tau()`, `uncertainty_coef()`, `somers_d()`) contribute one row per direction (`symmetric` / `R|C` / `C|R` where applicable), so the output has more rows than the number of helper functions.

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 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.

`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 (copy, print, CSV, Excel, PDF) directly from the Viewer.
- Variable selection uses the same tidyselect interface as `varlist()`; the underlying summary tibble is built by `varlist()` with `tbl = TRUE`.

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

Copies a data.frame, matrix, 2D or higher array, table, or atomic vector to the system clipboard, ready to paste into a text editor, spreadsheet, or word processor. Wraps clipr::write_clip() (a Suggests dependency); requires clipr to be installed and a clipboard backend to be available on the platform.

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, it is used as the column name for the promoted row names. Ignored (with a warning) when x is neither a data.frame nor a strict matrix.

`row.names`	Logical. If `TRUE` (the default), row names are included in the clipboard output; `FALSE` omits them.
`col.names`	Logical. If `TRUE` (the default), column names are included in the clipboard output; `FALSE` omits them.
`show_message`	Logical. If `TRUE` (the default), prints a success message after copying.
`quiet`	Logical. If `FALSE` (the default), messages are shown. If `TRUE`, suppresses all messages, including the success message, coercion notices, and warnings.
`...`	Additional arguments passed to `clipr::write_clip()`.

Details

Objects that are not `data.frames` or 2D matrices (atomic vectors, arrays, tables) are automatically coerced to character on the way to the clipboard, as required by `clipr::write_clip()`. The R-side object passed to `x` is never mutated.

Multidimensional arrays (3D and higher) are flattened to a 1D character vector with one element per line. To preserve a tabular layout, extract a 2D slice first, e.g. `copy_clipboard(my_array[, , 1])`.

Value

Invisibly returns `x`; the function is called for its clipboard side effect.

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

Counts, for each row of a `data.frame` or `matrix`, how many times one or more values appear across selected columns. Supports type-safe comparison (`allow_coercion = FALSE`), case-insensitive string matching (`ignore_case = TRUE`), and detection of special values (`NA`, `NaN`, `Inf`, `-Inf`) via `special`. Designed to flow inside `dplyr::mutate()` pipelines.

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 `dplyr::mutate()`, where the current data context is used automatically.
`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.

Value

A numeric vector of row-wise counts (unnamed), of length nrow(data).

Strict matching (allow_coercion = FALSE)

Comparison falls back to identical() when types differ, which also inspects factor levels. Two consequences:

- count = "b" does not match a factor "b" value: pass a factor, e.g. count = factor("b", levels = levels(df$x)).
- Even with a factor count, comparisons against columns whose level set differs will return 0. To guarantee a perfect match (label *and* levels), reuse a value taken from the data itself (e.g. df$x[2]).

Case-insensitive matching (ignore_case = TRUE)

All values are converted to lowercase via tolower() before matching; factor columns are first coerced to character. This mode takes precedence over allow_coercion: equality becomes lowercase string equality, so "b" and "B" match even when allow_coercion = FALSE.

Coercion of count itself

R coerces mixed-type vectors at construction time: count = c(2, "2") becomes c("2", "2") before the function ever sees it. To get type-sensitive matching, keep count homogeneous.

See Also

datawizard::row_count() for a closely related row-wise counter; count_n() adds element-wise type-safe matching, multi-value count, and special-value detection.

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 ($\tanh(\operatorname{atanh}(V) \pm z_{\alpha/2}/\sqrt{n-3})$), 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 via `interaction()`), row / column percentages, and inferential statistics (Chi-squared test with an APA-style association measure).

Both `x` and `y` are required; for one-way frequency tables, use `freq()`.

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). Required; the NULL default in the signature is a placeholder and triggers an error if left unset (use freq() for one-way tables).
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), "column", or "row". Unique abbreviations are accepted (e.g. "n", "c", "r").
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 NULL, which is resolved to 1 when percent != "none" and 0 when percent = "none" (counts are always integers).

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

Depends on styled and by:

- styled = TRUE, no by: a spicy_cross_table object (a data.frame carrying rendering metadata as attributes: title, digits, decimal_mark, n_row_idx, n_col_name, and the inferential block when include_stats = TRUE). Printing dispatches to print.spicy_cross_table().
- styled = TRUE, by supplied: a spicy_cross_table_list, i.e. a named list of spicy_cross_table objects (one element per group level, named by that level). Printing dispatches to print.spicy_cross_table_list() which renders each table in turn separated by a blank line.
- styled = FALSE: the same payload returned as a plain data.frame (or named list of data.frames with by), stripped of the spicy_* classes for downstream programmatic use.

Cell columns are the levels of y; rows are the levels of x. When percent != "none", the N column (or N row) is added according to show_n. When include_stats = TRUE, the result carries a Chi-squared row (statistic, df, *p*) and an association-measure row (estimate, optional CI via assoc_ci).

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

Designed to mimic common frequency procedures from SPSS or Stata while integrating the flexibility of R's data structures. The input type (vector, factor, labelled) is auto-detected; see @param labelled_levels and @param factor_levels for the schema-vs-observed level controls, and @param na_val for optional sentinel-value recoding.

Weighting (weights): frequencies and percentages are computed proportionally to the 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 normalised 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.

For schema-level inspection without computing frequencies, use varlist() or code_book().

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

cross_tab() for two-way cross-tabulations; table_categorical() for multi-variable categorical summary tables; varlist() / code_book() for variable inspection; 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. Goodman-Kruskal's Tau is intrinsically directional and has no canonical symmetric form (unlike lambda_gk() or uncertainty_coef()); only "row" and "column" are supported.

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 designed for rectangular tables; the estimate is bounded by $[-1, 1]$ only when the table is square, and may fall outside that range otherwise. 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

Designed primarily for **LimeSurvey CSV exports** with *Headings: Question code & question text*, which produce column names like "code. question text". The default separator ". " matches that export.

LimeSurvey question codes (the part *before* sep) are restricted to alphanumerics, must start with a letter, and contain no spaces – so the column name has to carry both the code and the question text. If your export uses *Headings: Question code* (codes only), re-export with *Question code & question text* before calling this function; there is no way to recover a label from a code alone.

Whitespace handling: the **name** (left of sep) is trimmed of surrounding whitespace, because R column names are intended to be referenced bare (without backticks) and leading / trailing whitespace would force quoting throughout the user's downstream code. The **label** (right of sep) is

preserved verbatim, following the Stata / SPSS convention that variable labels are faithful user content – spicy does not silently mutate label strings. To trim labels yourself, post-process with `labelled::var_label(df) <- lapply(labelled::var_label(df), trimws)`.

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 an optional minimum-valid-values rule*

Description

Computes row-wise means across selected numeric columns of a data.frame or matrix. Missing values are handled per row via min_valid (an integer count or proportion of non-NA values required); rows that fail the rule return NA. Non-numeric columns are dropped silently (set verbose = TRUE to see which). Designed to flow inside dplyr::mutate(): when called without an explicit data argument, the current data context is used.

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. Optional inside dplyr::mutate(), where the current grouping/data context is used automatically.
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 absolute value of the Pearson correlation between the two binary variables – spicy returns only the magnitude (always non-negative), matching the DescTools (Signorell et al., 2024) and SPSS conventions. To recover the signed direction of the 2x2 association, compute the Pearson correlation directly (e.g. cor(x, y) after coding both variables 0/1).

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 (1000–7400).

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 in kg/m^2 (16–39). 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 (range 0.29–3.45); calibrated so that `sum(weight)` matches the unweighted N and `mean(weight)` is approximately 1. See `Details`.

Details

Every variable carries a `"label"` attribute (read by `labelled::var_label()` and surfaced by `varlist()` / `code_book()`). The mix of factor types is deliberate: nominal factors (`sex`, `region`, ...) and ordered factors (`education`, `self_rated_health`, ...) live side by side so that `cross_tab()` and `table_categorical()` can demonstrate the automatic ordinal-vs-nominal dispatch (Cramer's V, Phi, Kendall's Tau-b, Goodman-Kruskal Gamma) on the same dataset.

Survey weights (`weight`) are calibrated: `sum(weight)` matches the unweighted N to within rounding (≈ 1200) and `mean(weight)` is ≈ 1. Weighted means therefore agree with unweighted means up to sampling noise without further rescaling.

Source

Simulated data for illustration purposes; reproducible by sourcing `data-raw/sochealth.R`. The script seeds the main generation block with `set.seed(2025)`, and the two missing-value injection blocks with `set.seed(2027)` (the four `life_sat_*` items) and `set.seed(2026)` (`smoking`, `self_rated_health`, `income_group`, `political_position`, `bmi`).

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 (direction = "symmetric") is the *harmonic* mean of the two asymmetric values, matching the SPSS / PSPP convention; this is **not** identical to Kendall's Tau-b (which is the *geometric* mean of the same two quantities), although the two often agree to two decimals. No analytic SE / CI is reported for the symmetric form (DescTools follows the same convention). Standard error formulas for the asymmetric directions 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 spicy-styled ASCII table to the console with optional title and note, table-type-aware alignment defaults, and automatic horizontal panelling when the table is wider than the console. Wraps the internal renderer build_ascii_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),
  center_headers = FALSE,
  spanners = NULL,
  group_sep_rows = integer(0),
  total_row_idx = attr(x, "total_row_idx"),
  display_labels = NULL,
  ...
)
```

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).
center_headers	Logical. When TRUE, column headers are centered above their column content even when the data itself is right-aligned. Passed through to build_ascii_table(). Defaults to FALSE.
spanners	Optional named list of column-group labels (label -> integer column indices). Passed through to build_ascii_table(); when the table is split into horizontal panels each panel keeps only the spanners whose columns are fully contained in it. Defaults to NULL (no spanner row).
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().
display_labels	Optional character vector of length ncol(x) used to override colnames(x) for the printed header text only. Sliced per panel when the table is split across stacked panels. Forwarded to build_ascii_table(); see that function for full semantics. Defaults to NULL.
...	Additional arguments passed to build_ascii_table().

Details

Table type is auto-detected from x and drives the default alignment when align_left_cols = NULL:

- **frequency table** (a Category column is present): the first two columns (Category, Values) are left-aligned.
- **cross table** (otherwise): only the first column (row variable) is left-aligned.

If the table is wider than the console, it is split into stacked horizontal panels with the left-most identifier columns repeated on each panel. Unicode line-drawing characters are used by default; coloured separators are drawn when the terminal supports ANSI colour (crayon::has_color()) and fall back to monochrome otherwise.

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 an optional minimum-valid-values rule*

Description

Computes row-wise sums across selected numeric columns of a data.frame or matrix. Missing values are handled per row via min_valid (an integer count or proportion of non-NA values required); rows that fail the rule return NA. Non-numeric columns are dropped silently (set verbose = TRUE to see which). Designed to flow inside dplyr::mutate(): when called without an explicit data argument, the current data context is used.

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. Optional inside dplyr::mutate(), where the current grouping/data context is used automatically.
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` 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`. Leading zeros are always stripped, following APA convention.

`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 (". " or ", "). 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()` and the omnibus chi-squared *p*-value is reported in the `p` column. See `@param correct` / `simulate_p` to switch on Yates' continuity correction or Monte Carlo *p*-values, and `@param assoc_measure` for the per-row dispatch table used by `"auto"` (2x2 -> Phi, both ordered -> Kendall's Tau-b, otherwise Cramer's V). 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

Decimal alignment, *p*-value formatting, and required suggested packages per output engine are documented under `@param align`, `@param p_digits`, and `@param output` respectively.

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: 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 each result to a variable -- some
# engines auto-print as a console-friendly text fallback inside
# the `?` help viewer.

# 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 layout, alignment, and reporting precision so descriptive and model-based analyses of the same data look uniform side by side. Use `table_continuous_lm()` when you need robust SE, weighted contrasts, fitted means, or covariate adjustment.

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. Coerced to factor for grouping; non-numeric grouping columns (factor, character, logical) are supported as-is.
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).
- "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 left at the default effect_size = "none", the function warns and promotes effect_size to "auto" so the requested CI can be shown. 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 remain after dropping NAs, with every group contributing at least two observations. Choice of test family is driven by test (see the @param entry for the full dispatch and the underlying stats:: functions called).

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

Effect sizes

See @param effect_size for the dispatch table (canonical measure for each (test, n_groups) combination) and the validation rules applied to explicit requests.

Confidence intervals (enabled with effect_size_ci = TRUE) use noncentral F inversion for η^2, the Hedges-Olkin normal approximation for g, the Fisher z-transform for r, and percentile bootstrap (2,000 replicates) for ε^2.

For Cohen's d, Hays' ω^2, and Cohen's f^2 (derived from a fitted, possibly weighted lm()), use the model-based companion table_continuous_lm().

Display conventions

Decimal alignment, *p*-value formatting, and required suggested packages per output engine are documented under `@param align`, `@param p_digits`, and `@param output` respectively.

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

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: `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 each result to a variable -- some
# engines auto-print as a console-friendly text fallback inside
# the `?` help viewer.

# 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 linear models for one or many continuous outcomes selected with tidyselect syntax.

A single focal predictor is supplied with by; each selected numeric outcome is fit as lm(outcome ~ by, ...), optionally extended with additive covariates via covariates and case weights via weights. Categorical by produces model-based estimated marginal means by level (covariate-adjusted via adjustment when covariates are present), plus an optional single difference for dichotomous predictors. Numeric by produces the slope and its confidence interval.

Inference adapts via vcov: classical OLS, "HC0"-"HC5" (heteroscedasticity-consistent), "CR0"-"CR3" (cluster-robust, requires cluster), or "bootstrap" / "jackknife" resampling. Effect sizes (Cohen's "d", Hedges' "g", Hays' "omega2", Cohen's "f2") are reported with optional noncentral t / F confidence intervals via effect_size_ci, and adapt under covariate adjustment (see effect_size).

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,
  covariates = NULL,
  adjustment = c("proportional", "balanced"),
  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 continuous regressor. The table reports the slope of by 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).
covariates	Optional additive covariates to adjust each per-outcome linear model for. Accepts a tidyselect expression (e.g. covariates = c(age, sex), covariates = tidyselect::all_of(cov_vec), covariates = tidyselect::starts_with("control_")) or a literal character vector of column names. Each covariate must be numeric, integer, logical, factor, or character; covariates that also appear in select are silently auto-excluded from the outcome list (a variable cannot be both outcome and adjustment), and a covariate that equals by raises an error (a variable cannot be both predictor and adjustment). When non-empty, each model is fitted as lm(y ~ by + cov1 + cov2 + ...) and the reported estimate / SE / p-value / CI on by are covariate-adjusted via the focal coefficient. For categorical by, the displayed emmean is the covariate-adjusted estimated marginal mean – see adjustment for the choice of estimand (G-computation by default vs. equal-weight averaging). The omnibus test of by is the Wald F restricted to the focal coefficients (computed via sandwich /

clubSandwich for HC* / CR* mode), so adding covariates does not contaminate the omnibus statistic with covariate contributions. Effect sizes adapt automatically – see effect_size.

v1 supports additive covariates only. Formula syntax with interactions or transforms (covariates = ~ age * sex, covariates = ~ I(age^2)) is reserved for a future release; passing a formula raises a spicy_unsupported error with a migration hint.

Rows with NA in any covariate are dropped from the analytic sample for each outcome (complete-cases per outcome, matching the existing by / weights NA handling).

adjustment — How the covariate-adjusted estimated marginal means (the emmean / emmean_se / emmean_ci_* columns) are computed when covariates is non-empty. One of:

- "proportional" (the default; matches Stata margins and marginaleffects::avg_predictions() G-computation on the observed sample. For each focal level of by, the model predicts at every observation with by set to that level (covariates kept at their observed values), and the predictions are averaged. Population-weighted by construction – the empirical joint distribution of covariates is the reference. Best when the goal is "what is the predicted mean in *this* population if everyone had by = lvl".
- "balanced" (matches emmeans::emmeans() default and the SPSS UNIANOVA EMMEANS / SAS LSMEANS conventions): synthetic grid of factor-covariate level combinations x numeric covariates fixed at their sample mean, with each grid cell weighted equally. Treats the design as if covariates were balanced – the "marginal mean assuming a balanced design" estimand. Best when the goal is to report a covariate-purified comparison independent of the empirical covariate distribution.

Both methods reduce to the same linear-contrast formula emmean = avg_row %*% beta and inherit the spicy variance pipeline (HC* / CR* / bootstrap / jackknife). They give the same answer when there are no covariates, and also when all covariates are numeric / logical (no factor levels to expand over). The two estimands diverge only when at least one factor / character covariate has non-uniform observed proportions.

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 \eqn{R^2}{R^2}, adjusted \eqn{R^2}{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), \eqn{R^2}{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 \eqn{f^2}{f^2} = \eqn{R^2}{R^2} / (1 - \eqn{R^2}{R^2}). Defined for any predictor type. Familiar from Cohen (1988); standard input for a-priori power analysis. Note that for a single-predictor model, \eqn{f^2}{f^2} is a monotone transform of \eqn{R^2}{R^2} 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 \eqn{R^2}{R^2} 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.

Under covariate adjustment (covariates non-empty):

- "f2" and "omega2" become the **partial** f^2 / partial ω^2, derived from the partial F of by via stats::drop1() – the correctly-defined effect size when the model is adjusted. For numeric by, partial f^2 equals the squared partial correlation of by with the outcome, divided by (1 - r^2_partial).
- "d" and "g" raise a spicy_unsupported error: Cohen's d and Hedges' g have no canonical extension to adjusted models (the pooled SD is undefined under adjustment). Use "f2" or "omega2" instead – both generalise via partial F.

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 \eqn{R^2}{R^2} (summary(lm)$r.squared).
- "adj_r2": adjusted \eqn{R^2}{R^2}, penalising for df_effect relative to the residual degrees of freedom.
- "none": omit the fit-statistic column.

When weights is supplied, \eqn{R^2}{R^2} and adjusted \eqn{R^2}{R^2} 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 (`\eqn{R^2}{R^2}` or adjusted `\eqn{R^2}{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 (\eqn{R^2}{R^2} or adjusted \eqn{R^2}{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 reporting around a single focal predictor: one model per selected continuous outcome, fitted as lm(outcome ~ by, ...) and optionally extended with case weights and additive covariates (lm(outcome ~ by + cov1 + ...)). For categorical by, the reported means are model-based fitted means (or covariate-adjusted estimated marginal means; see adjustment) for each level, and contrasts come from the same fitted linear model. For an unweighted lm(y ~ factor) with classical variance and no covariates, 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 derive from the model vcov (classical, HC*, CR*, bootstrap or jackknife), (c) under covariates they become adjusted marginal means, and (d) 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, `\eqn{R^2}{R^2}`, and CIs, so the effect size stays internally consistent with the rest of the table.

- `"f2"`: Cohen's `\eqn{f^2}{f^2} = \eqn{R^2}{R^2} / (1 - \eqn{R^2}{R^2})` (Cohen 1988). Defined for any predictor type. For a single-predictor model, `\eqn{f^2}{f^2}` is a monotone transform of `\eqn{R^2}{R^2}` 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.

Under covariate adjustment (`covariates` non-empty), `"f2"` and `"omega2"` become the partial f^2 / partial ω^2 of `by`, derived from the partial F via `stats::drop1()` restricted to the focal term. `"d"` and `"g"` raise a `spicy_unsupported` error: the pooled standard deviation has no canonical extension under adjustment, so Cohen's d and Hedges' g are undefined for adjusted models. See `effect_size` for the full dispatch.

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 `omega^2 = ncp / (ncp + N)` and `\eqn{f^2}{f^2} = 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.
\eqn{R^2}{R^2}, adjusted \eqn{R^2}{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 \eqn{R^2}{R^2} and adjusted \eqn{R^2}{R^2}, as well as Hays' omega^2 and Cohen's \eqn{f^2}{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).

When `covariates` is non-empty, the printed ASCII table appends an APA-style footer naming the covariates and the chosen estimand, e.g. `Note. Adjusted for age, education (proportional).`

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: table_categorical(), table_continuous()

Examples

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

# Default: ASCII table with model-based means, p, and \eqn{R^2}{R^2}.
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 \eqn{f^2}{f^2} alongside \eqn{R^2}{R^2} (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"
)

# --- Covariate adjustment ----------------------------------------------

# Adjust the comparison of `wellbeing_score` and `bmi` by `sex` for `age`
# and `education`. The footer surfaces the adjustment estimand
# ("proportional" by default = G-computation, matching Stata `margins`).
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  covariates = c(age, education),
  vcov = "HC3"
)

# Same model with the emmeans / SPSS UNIANOVA convention (equal-weight
# marginal means on a synthetic covariate grid).
```

```
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  covariates = c(age, education),
  adjustment = "balanced",
  vcov = "HC3"
)

# Effect sizes adjust automatically: f2 / omega2 become partial
# effect sizes via partial F (drop1) restricted to the focal `by`.
# d / g are undefined under adjustment and raise spicy_unsupported.
table_continuous_lm(
  sochealth,
  select = c(wellbeing_score, bmi),
  by = sex,
  covariates = c(age, education),
  effect_size = "f2",
  effect_size_ci = TRUE
)

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

# Pretty outcome labels and adjusted \eqn{R^2}{R^2}.
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/m^2)"
  ),
  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)
```

---

table_regression *Regression coefficient summary table*

Description

Publication-ready coefficient table from one or more fitted `lm` / `glm` models. Supports standardised coefficients (β), average marginal effects (AME), partial effect sizes (f^2 / η^2 / ω^2 for `lm`; partial χ^2 for `glm`), pseudo-R^2 (`glm`), and a full vocabulary of variance estimators (classical / HC* / cluster-robust with Satterthwaite-corrected df / bootstrap / jackknife). `glm` covers binomial / poisson / Gamma / inverse.gaussian / quasi families with any link.

Usage

```
table_regression(
  models,
  vcov = "classical",
  cluster = NULL,
  ci_level = 0.95,
  ci_method = c("wald", "profile"),
  boot_n = 1000L,
  standardized = c("none", "refit", "posthoc", "basic", "smart", "pseudo"),
  exponentiate = FALSE,
  p_adjust = "none",
  show_columns = NULL,
```

```
  keep = NULL,
  drop = NULL,
  show_intercept = TRUE,
  intercept_position = c("first", "last"),
  factor_layout = c("grouped", "flat"),
  reference_style = c("row", "annotation", "footer", "none"),
  reference_label = "(ref.)",
  show_fit_stats = NULL,
  fit_stats_layout = c("first_col", "merged"),
  model_labels = NULL,
  outcome_labels = NULL,
  stars = FALSE,
  nested = FALSE,
  digits = 2L,
  p_digits = 3L,
  effect_size_digits = 2L,
  fit_digits = 2L,
  ic_digits = 1L,
  decimal_mark = ".",
  align = c("decimal", "center", "right", "auto"),
  padding = 0L,
  labels = NULL,
  title = NULL,
  note = NULL,
 output = c("default", "data.frame", "long", "gt", "flextable", "tinytable", "excel",
    "clipboard", "word"),
  excel_path = NULL,
  excel_sheet = "Regression",
  clipboard_delim = "\t",
  word_path = NULL,
  word_template = NULL
)
```

Arguments

models	An `lm` or `glm` fitted model, or a list of such fits (named or unnamed; `lm` and `glm` may be mixed). Single fits are auto-promoted to a 1-element list. `merMod` and other classes raise `spicy_unsupported`. Raw data + formula is not accepted – fit-only API.
vcov	Variance-covariance estimator: `"classical"`, `"HC0"`-`"HC5"`, `"CR0"`-`"CR3"`, `"bootstrap"`, or `"jackknife"`. A scalar is recycled to all models; a list (one string per model) allows mixed estimators. Default `"classical"`. See *Inference and standard errors*.
cluster	Cluster identifier for cluster-robust variance (used when `vcov` is `"CR0"`-`"CR3"` or a cluster-bootstrap / cluster-jackknife). Three accepted forms (see *How to specify* `cluster` in the details): • Formula: `~region`, `~region:year` (recommended). • String column name: `"region"`.

- Atomic vector of length `nobs(fit)`: `df$region`, `interaction(df$region, df$year)`, ... (for keys derived on the fly).

For multi-model use, pass a list of one form per model (mix-and-match allowed). Bare unquoted names (`cluster = region`) are NOT accepted – use `~region` or `"region"`. Default `NULL` (no clustering).

`ci_level` Confidence level for all reported CIs (B, β, AME, partial effect sizes). Default `0.95`.

`ci_method` CI construction. `"wald"` (default) uses `estimate +/- z x SE` (`t x SE` for `lm`). `"profile"` (*glm only*) uses the profile-likelihood CI from `MASS::confint.glm()` – asymmetric, exact for likelihood-based inference (Venables & Ripley *MASS* Section 7.2). Only the CI bounds change; estimate, SE, statistic and p-value remain Wald. `"profile"` with `lm` raises `spicy_invalid_input`.

`boot_n` Number of bootstrap replicates when `vcov = "bootstrap"`. Single positive integer. Default `1000L`.

`standardized` Standardisation method for the `"beta"` column. One of `"none"` (default), `"refit"`, `"posthoc"`, `"basic"`, `"smart"`, `"pseudo"`. `"pseudo"` is *glm only* (Menard 2011 fully-standardised); using it with `lm()` raises `spicy_invalid_input`. See the *Standardised coefficients* section.

`exponentiate` Logical. When `TRUE` and the model is a `glm` with a non-identity link, `B`, the CI bounds, and the SE are transformed via `exp()` (delta method: `SE_OR = OR x SE_log-odds`). The column header is rebranded per family / link: `OR` (binomial logit), `IRR` (poisson log), `HR` (binomial cloglog), `RR` (binomial log), `MR` (Gamma log), else `exp(B)`. The statistic and p-value stay on the link scale (invariant under monotone transformation). Default `FALSE`. No effect on `lm` or identity-link glm; emits a `spicy_ignored_arg` warning in those cases.

`p_adjust` Multiple-comparison adjustment method applied to the family of estimated coefficient p-values within each model (intercept and reference rows excluded). One of `"none"` (default), `"holm"`, `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"` / `"fdr"`, or `"BY"`. Delegated to `stats::p.adjust()`; applied per-model and per `estimate_type` (B and AME p-values are adjusted independently within their own families). Active adjustments are documented in the footer (method + family size). See the *Multiple-comparison adjustment* section for when this is and is not appropriate.

`show_columns` Character vector of tokens selecting the per-coefficient columns and their display order. Accepts **atomic tokens** (`"b"`, `"se"`, `"ci"`, `"t"`, `"p"`, `"beta"`, `"ame"`, `"ame_se"`, `"ame_ci"`, `"ame_p"`, `"partial_f2"` + `"partial_f2_ci"`, `"partial_eta2"` + `"partial_eta2_ci"`, `"partial_omega2"` + `"partial_omega2_ci"`, `"partial_chi2"`) and **group tokens** (`"all_b"`, `"all_b_compact"`, `"all_b_full"`, `"all_beta"`, `"all_ame"`, `"all_ame_compact"`, `"all_f2"`, `"all_eta2"`, `"all_omega2"`). See *Vocabulary tokens* in the details for the full enumeration. Default `NULL` selects a context-aware layout: `"all_b"` (single model) or `"all_b_compact"` (multi-model). The `"p"` token is always the B / beta p-value; for the AME-specific p-value use `"ame_p"`.

`keep` Character vector of regexes. Only coefficient rows whose term name (as in `stats::coef()` – e.g. `"wt"`, `"cyl6"`, `"factor(cyl)8"`) matches at least one pattern are kept. Mutually exclusive with `drop`. Filtering is a display choice;

p_adjust runs against the full coefficient family before filtering. Default NULL (no filter).

drop Character vector of regexes. Coefficient rows matching any pattern are removed. Mutually exclusive with keep. Default NULL.

show_intercept Whether to display the intercept row. Default TRUE (APA convention). Hide via FALSE.

intercept_position
Where to place the intercept when shown. "first" (default, APA) or "last" (Stata-style, intercept just above the fit-stats footer). Ignored when show_intercept = FALSE (with spicy_ignored_arg warning).

factor_layout Layout of factor predictors. Applies to **any categorical predictor** – factor, ordered, character, or logical (R coerces the latter two to factors at fit time). Two options:

- "grouped" (default): the variable name on its own header row ending with : (e.g., education:); each level follows as an indented sub-row with the bare level name. APA convention.
- "flat": each non-reference dummy is one row with the <variable><level> form (e.g., educationUpper); no header, no indent. Econometrics convention.

reference_style
Rendering of factor reference levels. Four modes, distinguishing WHERE the reference information is exposed (in a row, inline, in the footer, or nowhere):

- "row" (default): explicit row Female (ref.) with em-dashes in all stat columns (NEJM / BMJ clinical convention). reference_label controls the suffix.
- "annotation": the row is dropped and the reference is shown inline. Under factor_layout = "grouped" the factor header reads education: [ref: Lower]; under factor_layout = "flat" the marker [vs Lower] is attached to the **first non-reference dummy** of each factor (subsequent dummies inherit the same reference).
- "footer": the row is dropped and a single line Reference categories: education = Lower; sex is added to the footer note. SAS PROC LOGISTIC / SPSS "Categorical Variables Codings" convention. Best for publication-grade dense multi-factor tables.
- "none": the row is dropped and no reference information is displayed anywhere. The user is responsible for stating the reference convention elsewhere (article text, table caption). Under factor_layout = "flat", an informational message is emitted to flag the silent omission.

reference_label
Suffix shown after the reference level in reference_style = "row" mode. Default "(ref.)". Ignored by the other three modes (which use structural English wording – "ref:", "vs", "Reference categories:").

show_fit_stats Character vector of tokens for the model-level rows below the coefficients; row order follows token order. NULL (default) resolves class-aware:

- lm: c("nobs", "r2", "adj_r2").

- glm: c("nobs", "pseudo_r2_mcfadden", "pseudo_r2_nagelkerke", "AIC").
- mixed lm + glm: the union of the two (the renderer em-dashes per cell the stat not defined for a given model class).

Under nested = TRUE the default is extended with the class-appropriate change-stat tokens (e.g. "r2_change", "f_change" for lm). See *Vocabulary tokens* (show_fit_stats subsection) and *Hierarchical (nested) model comparison* in the details for the full vocabulary.

fit_stats_layout

Layout of the fit-stat values (n, R^2, AIC, ...) within each model's column group. Two options:

- "first_col" (default): the value is placed in the FIRST numeric sub-column of each model (typically B); the model's remaining sub-columns (SE, LL, UL, p, ...) are left empty for that row. The APA Manual 7 Table 7.13 layout.
- "merged": the model's numeric sub-columns are merged into a single wide cell containing the fit-stat value, centred under the model spanner. Stata esttab layout / *Econometrica* and *AER* journal convention. Resolves the mixed-precision look of "first_col" (an integer n row sharing the B column with two-decimal coefficients).

Cell merging is supported by excel, flextable, and word (via flextable). gt, tinytable, clipboard, and default (console) always render in "first_col" mode regardless of this setting:

- gt lacks a native row-spanning cell-merge API (tab_spanner covers columns, not row-cell ranges).
- tinytable's style_tt(colspan = N) emits HTML colspan only on header rows, not on body cells.
- clipboard ships TSV plaintext.
- default ships fixed-width ASCII.

Decimal alignment of every numeric column is preserved in both modes: the B column decimal-aligns its coefficient values plus any fit-stat value(s) in "first_col" mode (native primitives handle the mixed-precision case), and trivially decimal-aligns in "merged" mode (the fit-stat values move out of the B column into the merged cell).

model_labels Per-model labels used as the **column-group spanner** above each model's sub-columns (console + gt / flextable / tinytable / Excel / Word renderers). NULL (default) resolves automatically; see *Multi-model semantics* for the full rule. A character vector of length length(models) overrides.

outcome_labels Optional **Outcome body row** override. NULL (default) hides the row entirely – under the multi-model spanner the DV is already visible above the data. A character vector of length length(models) forces an explicit Outcome row with those values (the spanner stays as "Model 1, ..." unless model_labels is also supplied). FALSE also suppresses the row.

stars Significance asterisks. FALSE (default, APA 7 Section 6.46) – no stars. TRUE – APA cutoffs c("*" = 0.05, "**" = 0.01, "***" = 0.001). A named numeric vector specifies custom thresholds, e.g. c("+" = 0.10, "*" = 0.05, "**" = 0.01, "***" = 0.001).

`nested` Whether to inject pairwise change-statistic rows for adjacent models (M2 vs M1, M3 vs M2, ...). `FALSE` (default) – pure side-by-side display. `TRUE` – requires identical `nobs` and identical response variable across all models. See *Hierarchical (nested) model comparison*.

`digits` Decimal places for general numeric tokens (`b`, `beta`, `se`, `ci`, `t`, `f_change`, `lrt_change`, `deviance`, `deviance_change`, `ame`, `ame_se`, `weighted_nobs`). Default `2L`.

`p_digits` Decimal places for p-values (`p`, `ame_p`, `p_change`). APA-strict: leading zero stripped, `<.001` (or `<.0001` etc. depending on `p_digits`) for small values. Default `3L`.

`effect_size_digits` Decimals for per-coefficient effect sizes (`partial_f2`, `partial_eta2`, `partial_omega2`). Default `2L`.

`fit_digits` Decimals for variance-explained / model-level effect-size fit stats (`r2`, `adj_r2`, `r2_change`, `adj_r2_change`, `omega2`, `f2`, `f2_change`, `sigma`, `rmse`). Default `2L`.

`ic_digits` Decimals for information criteria (`AIC`, `AICc`, `BIC`, and their `_change` form). Default `1L`.

`decimal_mark` Decimal mark used in numeric display. `"."` (default) or `","` (European convention). When `","` is used, the CI bracket separator switches to `"; "` automatically to avoid `"0,18 [0,07, 0,30]"` ambiguity.

`align` Numeric column alignment. `"decimal"` (default) – pre-pad cells so decimal marks line up vertically (publication-style). For CI cells (`[LL, UL]`) the left bracket, the LL decimal point, the comma separator, the UL decimal point, and the right bracket are independently aligned across rows. `"center"`, `"right"`, or `"auto"` for legacy per-column alignment.

`padding` Non-negative integer giving the extra characters added to each data column's auto-computed width when the default `print` method renders the table. Default `0L` (compact – fits more models in the same console / page width). Use `2L` (Stata-like) or `4L` for a more spacious look. Headers stay centered above the data region regardless of padding.

`labels` Named character vector overriding per-coefficient row labels. Names are coefficient term names (from `stats::terms()`); values are the displayed labels. E.g. `c("age" = "Age (years)", "sexM" = "Male (vs Female)")`. Default `NULL` (use raw term names).

`title`, `note` Override or suppress the auto-built caption / methodological footer. Three modes per argument:

- `NULL` (default): the package builds the standard caption ("Linear regression on <DV>" / "Hierarchical linear regression on <DV>" / ...) and a methodological note (VCV type, p-adjust method, reference categories, ...).
- `FALSE`: the corresponding banner row is omitted from every output engine. Use when the surrounding manuscript provides its own caption / note.
- character string (length 1): replaces the auto-built text verbatim. The renderer applies no APA formatting on top – supply the exact string you want displayed (multi- line notes accepted via embedded `"\n"`).

Validation messages, the spanner row, and the in-body change- stat rows are *not* affected – they belong to the table structure, not to the banner.

`output` Output type. `"default"` (a printable `spicy_regression_table`); `"data.frame"` / `"long"` (raw data); `"gt"` / `"flextable"` / `"tinytable"` (rich-format tables); `"excel"` (writes to `excel_path`); `"clipboard"` (copies to system clipboard); `"word"` (writes flextable to `word_path`).

`excel_path` File path for `output = "excel"`. Default `NULL` (required when `output = "excel"`).

`excel_sheet` Sheet name when writing to Excel. Default `"Regression"`.

`clipboard_delim`

Field delimiter for `output = "clipboard"`. Default `"\t"` (tab-separated, pastes cleanly into Excel / Google Sheets / Word). The clipboard payload mirrors the Excel layout (title row, spanner row, header, body, footer note) but is plain text – horizontal rules, cell merging, decimal alignment, monospace font, and factor-level indentation cannot be encoded in TSV and are therefore absent from the paste.

Paste behaviour by target:

- **Excel / Google Sheets:** numerics are auto-detected and right-aligned; text cells stay left-aligned. (P-values such as `.005` get re-parsed as `0.005` by Excel's auto-format – to preserve the APA leading-zero-dropped display, prefer `output = "excel"`.)
- **Word:** the paste is converted to a Word table; all cells start left-aligned. Apply a Table Style (Insert > Table > Design) for APA-style borders, and set right-alignment on numeric columns (Layout > Align Right). For a self-contained Word file with borders and alignment pre-applied, use `output = "word"` instead.

`word_path` File path for `output = "word"`. Default `NULL` (required when `output = "word"`). The Word table inherits the flextable styling (Calibri font, APA borders, decimal-aligned numerics) and adds Word-specific features: an auto-numbered caption ("Table 1: ...", "Table 2: ...") via Word's `SEQ` field so multiple `table_regression()` calls in one document number consecutively; a re-printed header row on each page break; row split prevention so a single coefficient row never wraps across two pages; and an APA-styled note line (`*Note.*` italic prefix per APA Manual 7 §7.14).

R Markdown / Quarto: for embedded use, prefer `output = "flextable"` (returns the flextable object that knits to docx/HTML/PDF natively). `output = "word"` writes a standalone .docx file, suited to scripted exports rather than chunk-level rendering.

`word_template` Optional path to a custom .docx file used as the template for `output = "word"`. The template's header, footer, page size, margins, and named styles ("Table Caption" in particular) are honoured; the table is appended to the template body. Useful for institutional templates with pre-set headers ("APA Style", "Manuscript Submission Template", etc.). Default `NULL` (uses flextable's stock template).

Customising the caption appearance: the table caption is tagged with the Word named style `"Table Caption"`. The visual rendering (italic / bold / colour / font) follows whatever that style is set to in the docx template. The stock

Word template renders "Table Caption" in italic — the APA Manual 7 §7.10 condensed convention. For a different appearance (Nature-style bold non-italic, APA-strict 2-line bold-number / italic-title, etc.), edit the "Table Caption" style in a docx template and pass it via word_template = "your_template.docx". Style-based delegation keeps the rendered caption consistent with the surrounding document and lets editorial conventions (Nature, APA-strict, journal-specific) be applied without modifying the call site.

Value

A spicy_regression_table object (a data.frame subclass with classes c("spicy_regression_table", "spicy_table", "data.frame")) when output = "default". The result carries rendering attributes (title, note, align, padding) and provenance attributes (outcome, model_ids) consumed by the print method and the broom methods. For other output values, returns the format-specific object (gt_tbl, flextable, tinytable, data.frame, tbl_df, or invisible(x) for side-effect outputs).

Vocabulary tokens

Two vector arguments – show_columns and show_fit_stats – accept named tokens that select **what** to display and in **what order**. All tokens are lowercase (snake_case for compound tokens). Group tokens ("all_b", "all_ame", ...) expand to a fixed vector of atomic tokens; see show_columns below.

show_columns **– per-coefficient columns:**

Each token = one displayed column.

- Coefficient family: "b", "beta" (standardised), "se", "ci", "t", "p".
- Marginal effects: "ame", "ame_se", "ame_ci", "ame_p". "p" always refers to the B-coefficient p-value; for the AME-specific p-value use "ame_p".
- Partial effect sizes – lm only: "partial_f2", "partial_eta2", "partial_omega2", each with a paired _ci companion ("partial_f2_ci", ...).
- Partial effect size – glm only: "partial_chi2" (likelihood-ratio chi-square via drop1(test = "LRT"); SAS PROC LOGISTIC TYPE3; Long & Freese 2014 Section 3.5). Rendered as value (df) to disambiguate factor terms (k-1 df) from numeric terms (1 df).

Group tokens (presets) expand to a fixed atomic vector before validation:

- "all_b" -> c("b", "se", "ci", "p")
- "all_b_compact" -> c("b", "se", "p")
- "all_b_full" -> c("b", "se", "ci", "t", "p")
- "all_beta" -> c("b", "beta", "se", "ci", "p")
- "all_ame" -> c("ame", "ame_se", "ame_ci", "ame_p")
- "all_ame_compact" -> c("ame", "ame_p")
- "all_f2" / "all_eta2" / "all_omega2" -> partial_* + its _ci companion.

Mix groups and atomic tokens: show_columns = c("all_b", "ame", "ame_p"). Duplicates after expansion are deduplicated; the order of tokens controls the order of the displayed columns. If standardized != "none" and "beta" is not already requested, it is auto-injected after "b". Asking for "beta" while standardized = "none" raises spicy_invalid_input.

Default (show_columns = NULL) is context-aware: "all_b" for a single model (APA-7 Section 6.46 publication layout), "all_b_compact" for two or more models (CI dropped to fit the side-by-side layout; restore it explicitly when needed).

show_fit_stats **– model-level rows below the coefficients:**

- Counts: "nobs", "weighted_nobs".
- Variance explained (lm only): "r2", "adj_r2", "omega2".
- Pseudo-R^2 (glm only): "pseudo_r2_mcfadden" (McFadden 1974), "pseudo_r2_nagelkerke" (Nagelkerke 1991), "pseudo_r2_tjur" (Tjur 2009; binomial only).
- Residual scale: "sigma" (lm $\hat{\sigma}$ / glm dispersion), "rmse".
- Effect size: "f2".
- Information criteria: "AIC", "AICc", "BIC", "deviance".
- Change-stats for hierarchical comparison (active under nested = TRUE; see *Hierarchical comparison* below): "r2_change", "adj_r2_change", "f_change", "f2_change", "lrt_change", "aic_change", "aicc_change", "bic_change", "deviance_change", "p_change".

Default (resolved when NULL) is class-aware: lm fits get c("nobs", "r2", "adj_r2"); glm fits get c("nobs", "pseudo_r2_mcfadden", "pseudo_r2_nagelkerke", "AIC"); mixed lm + glm sets union both groups (the renderer per-row em-dashes the inappropriate cell). When nested = TRUE, the class-aware default is extended with change tokens (c("r2_change", "f_change", "p_change") for lm, c("lrt_change", "p_change") for glm). The order of tokens in show_fit_stats controls the order of the rows.

Multi-model semantics

Pass a single fit or a list() of fits. Multi-model layout draws a centred **spanner label** above each model's sub-columns:

- list("Naive" = m1, "Adjusted" = m2) -> spanner labels "Naive" / "Adjusted". Partial naming (list("Naive" = m1, m2)) auto-fills missing slots as "Model <position>".
- list(m1, m2) (unnamed) -> if all response variables differ, the bare DV name (from formula(fit)[[2]]) becomes the spanner label and the redundant Outcome body row is suppressed. If DVs match, the labels default to "Model 1, 2, ...".
- model_labels = c("A", "B") overrides everything.

Duplicate explicit names in the list are rejected (spicy_invalid_input) – they would silently collide in the internal model_id key.

Inference and standard errors

vcov selects the variance-covariance estimator:

- "classical" – OLS (lm) / MLE inverse Hessian (glm).
- "HC0" to "HC5" – heteroskedasticity-consistent (via sandwich::vcovHC()).
- "CR0" to "CR3" – cluster-robust with Satterthwaite-corrected df (via clubSandwich::vcovCR()). Requires cluster.
- "bootstrap" – nonparametric or cluster bootstrap (boot_n replicates).

- `"jackknife"` – leave-one-out / leave-one-cluster-out.

For multi-model use, both `vcov` and `cluster` accept a single value (recycled to all models) or a list (one per model). The same fit can appear several times with different estimators to compare standard errors side-by-side.

Inferential regimes (B and AME share the same regime):

- `classical`, `HC*` -> t with `df.residual`.
- `bootstrap`, `jackknife` -> z asymptotic.
- `CR0-CR3` -> t with **Satterthwaite-corrected df** (B via `clubSandwich::coef_test()`; AME via `clubSandwich::linear_contrast()`; Pustejovsky & Tipton 2018). Under non-linear terms (`poly()`, `I()`, `log()`, `splines::ns()`), AME falls back to z-asymptotic with a `spicy_fallback` warning.

How to specify `cluster`:

Three accepted forms, in order of preference:

1. **Formula** – `cluster = ~region` (or `cluster = ~region:year` for the interaction of two variables). The variables are looked up in `model.frame(fit)` first, then in the original `data` argument captured by the fit. **Recommended**: independent of the dataset's name, composable for multi-way clustering, consistent with `sandwich::vcovCL()` / `clubSandwich::vcovCR()`.
2. **String** – `cluster = "region"`. A single column name resolved the same way as the formula. Convenient but cannot express interactions.
3. **Vector** – `cluster = df$region`. An atomic vector of length `nobs(fit)`. Use this when the cluster key is **derived on the fly** (`cluster = interaction(df$region, df$year)`, `cluster = as.integer(format(df$date, "%Y"))`), comes from a **different dataset** with matching row order, or is otherwise not a column of the model's `data`.

Bare unquoted names (`cluster = region`) are **not** accepted – they would require non-standard evaluation magic that breaks under programmatic use (function wrapping, dynamic column choice, loops). Use `~region` or `"region"` instead.

For multi-model use, mix forms freely: `cluster = list(~region, "region", df$region)`.

Hierarchical (nested) model comparison

`nested = TRUE` adds **per-pair change statistics as in-table rows** (APA Table 7.13 / Stata `esttab` / SPSS Model Summary convention). Each adjacent pair (M2 vs M1, M3 vs M2, ...) contributes one column of change stats; the FIRST model column gets em-dashes (no previous model to compare to). Validation requires identical `nobs` and identical response variable across all models.

Default change tokens auto-injected when `show_fit_stats` is `NULL`:

- All-lm: `c("r2_change", "f_change", "p_change")` – APA hierarchical regression standard.
- All-glm: `c("lrt_change", "p_change")` – Hosmer & Lemeshow Section 3.5; Long & Freese 2014 Section 3.6.

To customise, pass the change tokens directly to `show_fit_stats`. Variance-explained change tokens on an all-glm hierarchy raise `spicy_invalid_input` (the residual-sum-of-squares partition does not apply outside the least-squares framework – the renderer points the user at `lrt_change`).

Standardised coefficients

`standardized` controls the method when `"beta"` is in `show_columns`:

- `"refit"` – refit on z-scored data. For `lm` both X and Y are z-scored (Cohen et al. 2003 gold standard); for `glm` only numeric X (Long & Freese 2014 Section 4.3.4 "x-standardization").
- `"posthoc"` – post-hoc scaling. lm: $\beta = B \times SD(X)/SD(Y)$; glm: X-only $\beta = B \times SD(X)$ (Y is undefined on the link scale).
- `"basic"` – like `"posthoc"` but factor dummies are scaled by their column SD.
- `"smart"` – Gelman (2008): divide binary predictors by `2 * SD` instead of `SD`.
- `"pseudo"` – *glm only*. Menard (2004, 2011) fully-standardised $\beta = B \times SD(X)/SD(Y^*)$, with Y^* the latent variable on the link scale and $SD(Y^*) = \sqrt{Var(\hat{\eta}) + Var_{link}}$ ($\pi^2/3$ logit, 1 probit, $\pi^2/6$ cloglog). Binomial families only; non-binomial returns NA with a `spicy_caveat`.
- `"none"` (default) – no β computed.

Under interactions or transformed predictors (`I()`, `poly()`, `log()`, `splines::ns()`), a `spicy_caveat` warns that standardised coefficients on such terms are subtle to interpret (Cohen et al. 2003 Section 7.7; Aiken & West 1991). The caveat is auto-documented in the footer.

Multiple-comparison adjustment

Adjusting the p-values of all coefficients of a single regression model is **not** the standard convention. Each coefficient tests a distinct hypothesis on a distinct predictor – not the situation multiple-testing procedures were designed for (Rothman 1990; Greenland 2017; APA Manual 7 Section 6.46; Harrell *Regression Modeling Strategies* Section 5.4; Gelman, Hill & Yajima 2012). Hence the default `p_adjust = "none"`.

Adjustment is appropriate for: mass screening with no prior hypothesis (typically `"BH"` / FDR), pre-registered multi-endpoint confirmatory designs (typically `"holm"`), or when a journal / SAP explicitly requests it.

The adjustment runs **before** any `keep` / `drop` filtering, so the family is the model's full coefficient set (intercept and reference rows excluded), not the displayed subset – filtering is a display choice and must not change the inferential family.

Output formats and broom integration

`output` selects the return type:

- `"default"` – a `spicy_regression_table` (`data.frame` subclass) printed via `spicy_print_table()`.
- `"data.frame"` / `"long"` – raw data.frame / long-format tibble.
- `"gt"` / `"flextable"` / `"tinytable"` – rich-format HTML / Word / PDF tables (require the corresponding Suggests package).
- `"excel"` – writes to `excel_path` via `openxlsx2::write_xlsx()`.
- `"word"` – writes to `word_path` via `flextable::save_as_docx()`.
- `"clipboard"` – copies to the system clipboard via `clipr::write_clip()`.

broom::tidy() returns a long tibble with one row per (model_id, term, estimate_type) and broom-canonical column names (estimate, std.error, conf.low, conf.high, statistic, p.value). broom::glance() returns one row per model with the model-level statistics; df.residual is kept numeric so cluster-robust Satterthwaite df is preserved.

Weights

No weights argument: weights are a property of the fit (extracted via stats::weights()). Pass them when fitting: lm(y ~ x, data = df, weights = w). All downstream computations (vcov, AME, standardisation, weighted_nobs) extract them automatically.

Internationalisation

Output is in English. Override user-facing strings via reference_label, model_labels, outcome_labels, and labels. The title and footer are post-processable via attr(result, "title") and attr(result, "note").

Classed conditions

Every error and warning emitted by table_regression() carries a classed condition for programmatic dispatch via tryCatch() or withCallingHandlers(). Errors inherit from spicy_error (root); warnings from spicy_warning. Specific leaves used by this function include spicy_invalid_input, spicy_invalid_data, spicy_unsupported, spicy_missing_pkg, spicy_missing_column, spicy_ignored_arg, spicy_caveat, spicy_fallback. See spicy for the full taxonomy.

References

APA Manual 7 (American Psychological Association, 2020), Tables 7.13-7.15.

Aiken, L.S. & West, S.G. (1991). *Multiple regression: Testing and interpreting interactions*.

Cohen, J., Cohen, P., West, S.G., & Aiken, L.S. (2003). *Applied multiple regression / correlation analysis for the behavioral sciences* (3rd ed.). Lawrence Erlbaum.

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.

Wasserstein, R.L., Schirm, A.L., & Lazar, N.A. (2019). Moving to a world beyond "p < 0.05". *The American Statistician*, 73(sup1), 1-19.

See Also

Other regression-table functions: table_continuous_lm() for one-predictor-by-many-outcomes descriptive tables. Other spicy table functions: freq(), cross_tab(), table_categorical(), table_continuous(). Underlying machinery: spicy_print_table() for ASCII rendering; build_ascii_table() for the low-level renderer. Inferential infrastructure (internal): compute_lm_vcov(), compute_lm_coef_inference(), compute_lm_wald_test(). broom integration: broom::tidy(), broom::glance().

Examples

```
# ---- Single-model usage ------------------------------------------
fit <- lm(wellbeing_score ~ age + sex + smoking, data = sochealth)

# Default APA layout: B / SE / 95% CI / p plus the n / R^2 /
# Adj.R^2 fit-stats footer. Factor reference level is annotated
# with `(ref.)` and shows an em-dash in the statistic columns.
table_regression(fit)

# Standardised coefficients (beta) injected next to B. Four
# methods available; "refit" is the SPSS / Stata regress, beta
# gold standard.
table_regression(fit, standardized = "refit")

# Custom column set: B + AME + AME-specific p-value. Note that
# the `p` token always belongs to B, never to AME -- use the
# explicit `ame_p` token for AME inference.
table_regression(
  fit,
  show_columns = c("b", "p", "ame", "ame_ci", "ame_p")
)

# Group-token shortcut: "all_b" + "all_ame" expands to the full
# B / AME column families side by side.
table_regression(fit, show_columns = c("all_b", "all_ame"))

# ---- Cluster-robust variance -------------------------------------
# CR2 (Bell-McCaffrey) with Satterthwaite-corrected df is the
# recommended default under few clusters. Three forms are accepted
# for `cluster`; the formula is preferred for composability with
# multi-way clustering and for programmatic robustness.
table_regression(fit, vcov = "CR2", cluster = ~region)
table_regression(fit, vcov = "CR2", cluster = "region")
table_regression(fit, vcov = "CR2", cluster = ~region:age_group)

# ---- Hierarchical (nested) regression ----------------------------
# Adds in-table change-statistic rows (Delta R^2 / F-change /
# p-change for lm; LRT / p-change for glm) below the fit-stats.
# Note: hierarchical comparison requires identical observations
# across all models -- prepare a complete-case subset first so
# R's listwise deletion does not produce different `nobs` per
# model (which the function rejects).
sochealth_cc <- na.omit(
  sochealth[, c("wellbeing_score", "age", "sex", "smoking")]
)
m1 <- lm(wellbeing_score ~ age,                 data = sochealth_cc)
m2 <- lm(wellbeing_score ~ age + sex,           data = sochealth_cc)
m3 <- lm(wellbeing_score ~ age + sex + smoking, data = sochealth_cc)
table_regression(
  list("Step 1" = m1, "Step 2" = m2, "Step 3" = m3),
  nested = TRUE
)
```

```
# ---- Side-by-side variance comparison ----------------------------
# Same fit, three vcovs in one wide table. Useful for showing the
# sensitivity of inference to the variance assumption.
table_regression(
  list("Classical" = fit, "HC3" = fit, "CR2" = fit),
  vcov    = list("classical", "HC3", "CR2"),
  cluster = list(NULL, NULL, ~region)
)

# ---- Tidy long format for downstream pipelines -------------------
broom::tidy(table_regression(fit))

## Not run:
# ---- Rich-format outputs (require optional Suggests packages) ----
table_regression(fit, output = "gt")
table_regression(fit, output = "flextable")
table_regression(fit, output = "tinytable")

# ---- File outputs ------------------------------------------------
table_regression(fit, output = "excel",
                 excel_path = tempfile(fileext = ".xlsx"))
table_regression(fit, output = "word",
                 word_path  = tempfile(fileext = ".docx"))

# ---- System clipboard (interactive use) --------------------------
table_regression(fit, 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. Let H_X and H_Y be the marginal entropies of the **row** and **column** variables respectively, and H_{XY} the joint entropy.

- direction = "row" (column predicts row): $U = (H_X + H_Y - H_{XY})/H_X$.
- direction = "column" (row predicts column): $U = (H_X + H_Y - H_{XY})/H_Y$.
- direction = "symmetric": $U = 2(H_X + H_Y - H_{XY})/(H_X + H_Y)$.

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: variable names, labels, summary values, classes, number of distinct values, number of valid (non-missing) observations, and number of missing values. Tidyselect-style selectors can be supplied to pick or reorder columns dynamically.

`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

In an interactive session (RStudio, Positron, ...), the summary opens in the Viewer pane with a contextual title like vl: sochealth. If the data frame has been transformed or subsetted, the title is suffixed with * (e.g. vl: sochealth*); anonymous or ambiguous calls fall back to vl: <data>. Pass tbl = TRUE to return a tibble instead.

The default factor_levels = "observed" mirrors what is actually in the data; code_book() defaults to "all" to document the declared schema. See @param factor_levels 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.

With tbl = FALSE (the default) the tibble is sent to the Viewer (interactive) or surfaced via a message (non-interactive) and the function returns invisibly NULL. Set tbl = TRUE to return the tibble directly for downstream use.

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 $SE = 0.5(1 - Q^2)\sqrt{1/a + 1/b + 1/c + 1/d}$.

Edge cases: when ad + bc = 0, Q itself is undefined and the function returns NA with a spicy_undefined_stat warning. When any cell is zero (and ad + bc > 0), Q is well-defined but the SE formula divides by zero – the point estimate is returned, and se, ci_lower, ci_upper, and p_value are all NA.

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