

Package: Saqrmisc (via r-universe)

October 19, 2026

Type Package

Title Miscellaneous Functions for Data Processing and Analysis

Version 0.9.8

Description A comprehensive R package providing miscellaneous functions for data processing, statistical analysis, and visualization. Includes functions for latent profile clustering via the 'latents' package, comparison plots with statistical testing, and mosaic plot analysis for categorical variables. The package offers robust error handling, comprehensive documentation, and user-friendly interfaces for complex statistical tasks.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://github.com/mohsaqr/Saqrmisc>

BugReports <https://github.com/mohsaqr/Saqrmisc/issues>

Depends R (>= 4.1.0)

Imports latents (>= 0.8.8), dplyr, ggplot2, vcd, grid, tibble, rlang, gridExtra, gt, magrittr, tidyr, bootnet, qgraph, mgm, stats, grDevices, utils, BayesFactor, effectsize, effsize

Suggests testthat (>= 3.0.0), knitr, rmarkdown, devtools, roxygen2, ggstatsplot, kableExtra, TOSTER, httr2, clipr, scales, NetworkComparisonTest, shiny, DT

Remotes mohsaqr/latents

VignetteBuilder knitr

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev pari-gp libnode-dev libx11-dev zlib1g-dev

Repository <https://mohsaqr.r-universe.dev>

Date/Publication 2026-09-28 20:27:04 UTC

RemoteUrl <https://github.com/mohsaqr/Saqrmisc>

RemoteRef HEAD

RemoteSha 364f9cfe6e5e9a819b9f40f6f19472c1efa129a8

Contents

add_total	3
as.data.frame.mosaic_analysis	4
auto_describe	5
categorical_table	7
center	10
center_vec	11
clustering	12
clustering_plot_types	13
compare_groups	14
compare_networks	21
convert_sequences	22
correlation_matrix	24
correlations	26
counts	29
data_overview	30
descriptive_table	31
estimate_grouped_networks	34
estimate_single_network	36
fitted.saqr_clustering	37
is_outlier	38
launch_app	39
missing_analysis	40
mosaic_analysis	41
normality_check	45
outlier_check	46
pass	47
plot.mosaic_analysis	50
plot.saqr_clustering	51
plot_clustering	51
print.grouped_network_analysis	52
print.html_table	53
print.latex_table	53
print.markdown_table	54
print.mosaic_analysis	54
print.mosaic_stratified	55
print.network_analysis	55
print.saqr_clustering	56
print.saqr_plots	56
print.saqr_result	57
replace_missing	57
replace_outliers	58
reverse_code	60

reverse_code_vec	61
saqr_result	62
scale_vars	63
scale_vec	64
set_api_key	65
set_claude_key	66
set_gemini_key	66
set_openai_key	67
set_openrouter_key	68
standardize	68
standardize_vec	69
summary.mosaic_analysis	70
summary.mosaic_stratified	70
summary.network_analysis	71
summary.saqr_clustering	71
to_dataframe	72
to_gt	72
to_html	74
to_kable	75
to_latex	76
to_markdown	77
winsorize_vec	78

Index

80

add_total	<i>Add Row Totals to Data Frame</i>
-----------	-------------------------------------

Description

Computes row totals across specified columns and adds them as a new column.

Usage

```
add_total(data, cols = NULL, name = "total", na.rm = TRUE)
```

Arguments

data	A data frame.
cols	Column specification. Can be: • NULL (default): uses all numeric columns • Character vector: column names, e.g., 'c("a", "b", "c")' • Numeric vector: column indices, e.g., '2:5' or 'c(2,3,4)' • Single number: from that column to end, e.g., '3' means cols 3 to last
name	Character. Name for the total column. Default: "total".
na.rm	Logical. Remove NA values when summing? Default: 'TRUE'.

Value

The original data frame with an added total column.

Examples

```
## Not run:
df <- data.frame(id = 1:3, a = c(1,2,3), b = c(4,5,6), c = c(7,8,9))

# All numeric columns
add_total(df)

# Specific columns by name
add_total(df, cols = c("a", "b"))

# Columns by index range
add_total(df, cols = 2:4)

# From column 2 to end
add_total(df, cols = 2)

# Custom name
add_total(df, cols = c("a", "b"), name = "sum_ab")

## End(Not run)
```

```
as.data.frame.mosaic_analysis
```

Tidy accessor for a mosaic_analysis result

Description

Returns one of the result's tables as a plain `data.frame`, so callers never have to reach into the object. For a stratified fit (one produced with `by =`), `what = "strata"` gives one row per stratum with its own test, effect size and multiplicity-corrected p-value.

Usage

```
## S3 method for class 'mosaic_analysis'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  what = c("summary", "table", "residuals", "strata", "overall"),
  ...
)
```

Arguments

<code>x</code>	A <code>mosaic_analysis</code> object.
<code>row.names</code> , optional	Ignored; present for S3 consistency.
<code>what</code>	Which table to return: "summary" (the statistical summary, default), "table" (consolidated observed/expected counts), "residuals" (standardized Pearson residuals), "strata" (one row per stratum), or "overall" (the pooled test plus the Cochran-Mantel-Haenszel test). The last two require a stratified fit.
<code>...</code>	Ignored.

Value

A data.frame. For `what = "strata"`, one row per stratum with columns `stratum`, `n`, `test`, `statistic`, `df`, `p_value`, `p_adjusted`, `cramers_v`, `effect_size` and `n_cells_beyond_2`. For `what = "residuals"` on a stratified fit, one row per stratum x cell.

Examples

```
set.seed(1)
d <- data.frame(
  a = sample(c("X", "Y", "Z"), 400, replace = TRUE),
  b = sample(c("P", "Q"), 400, replace = TRUE),
  g = sample(c("g1", "g2"), 400, replace = TRUE)
)
fit <- mosaic_analysis(d, "a", "b", by = "g", min_count = 5, verbose = FALSE)
as.data.frame(fit, what = "strata")
as.data.frame(fit, what = "overall")
```

auto_describe

Automatically Generate Descriptive Statistics for All Variables

Description

Automatically detects numeric and categorical variables in a data frame and generates appropriate descriptive statistics tables for each type. Numeric variables get summary statistics (mean, SD, etc.) and categorical variables get frequency tables.

Usage

```
auto_describe(
  data,
  group_by = NULL,
  numeric_stats = c("n", "mean", "sd", "median", "min", "max"),
  digits = 2,
  exclude = NULL,
  force_categorical = NULL,
  title_numeric = "Numeric Variables",
```

```

    title_categorical = "Categorical Variables",
    theme = "default",
    format = "gt",
    print = TRUE
  )

```

Arguments

<code>data</code>	A data frame to describe.
<code>group_by</code>	Optional character. Name of a grouping variable for stratified statistics.
<code>numeric_stats</code>	Character vector of statistics for numeric variables. Default: <code>c("n", "mean", "sd", "median", "min", "max")</code> .
<code>digits</code>	Integer. Decimal places for numeric output. Default: <code>'2'</code> .
<code>exclude</code>	Character vector of column names to exclude. Default: <code>'NULL'</code> .
<code>force_categorical</code>	Character vector of numeric column names to treat as categorical instead. Default: <code>'NULL'</code> .
<code>title_numeric</code>	Title for numeric table. Default: <code>"Numeric Variables"</code> .
<code>title_categorical</code>	Title for categorical tables. Default: <code>"Categorical Variables"</code> .
<code>theme</code>	Visual theme: <code>"default"</code> , <code>"minimal"</code> , <code>"dark"</code> , <code>"colorful"</code> . Default: <code>"default"</code> .
<code>format</code>	Output format: <code>"gt"</code> or <code>"data.frame"</code> . Default: <code>"gt"</code> .
<code>print</code>	Logical. Print tables to console? Default: <code>TRUE</code> .

Value

A list with:

- `numeric`: Descriptive table for numeric variables (or `NULL` if none)
- `categorical`: List of frequency tables for categorical variables
- `variable_types`: Data frame showing detected variable types

Examples

```

## Not run:
# Automatic description of all variables
data <- data.frame(
  age = rnorm(100, 35, 10),
  score = rnorm(100, 75, 15),
  gender = sample(c("M", "F"), 100, replace = TRUE),
  education = sample(c("HS", "BA", "MA", "PhD"), 100, replace = TRUE)
)

# Describe all variables automatically
results <- auto_describe(data)

```

```
# With grouping
results <- auto_describe(data, group_by = "gender")

# Access individual tables
results$numeric
results$categorical$education

## End(Not run)
```

categorical_table	<i>Generate Publication-Ready Frequency Tables for Categorical Variables</i>
-------------------	--

Description

Creates comprehensive frequency tables for categorical variables with counts, percentages, and optional cross-tabulation. Supports chi-square tests for associations and outputs beautifully formatted gt tables.

Usage

```
categorical_table(
  data,
  var,
  by = NULL,
  group_by = NULL,
  percentages = NULL,
  show_n = TRUE,
  show_total = TRUE,
  show_missing = FALSE,
  show_header = TRUE,
  chi_square = TRUE,
  fisher = FALSE,
  cramers_v = TRUE,
  digits = 1,
  sort_by = "none",
  labels = NULL,
  title = NULL,
  subtitle = NULL,
  format = c("gt", "plain", "markdown", "latex", "kable"),
  theme = "default",
  combine = TRUE,
  interpret = FALSE,
  ...
)
```

Arguments

<code>data</code>	A data frame containing the variables to summarize.
<code>var</code>	Character. Name of the primary categorical variable.
<code>by</code>	Optional character. Name of a second categorical variable for cross-tabulation. When provided, creates a contingency table.
<code>group_by</code>	Optional character. Name of a stratification variable. Creates separate tables for each level of this variable.
<code>percentages</code>	Character vector specifying which percentages to include. Options: "col" (column), "row", "total", "none". Default: <code>c("col")</code> for cross-tabs, <code>c("total")</code> for single variable.
<code>show_n</code>	Logical. Show frequency counts? Default: TRUE.
<code>show_total</code>	Logical. Include row/column totals? Default: TRUE.
<code>show_missing</code>	Logical. Include missing values as a category? Default: FALSE.
<code>show_header</code>	Logical. Show title/subtitle header? Default: TRUE. Set to FALSE to hide the table header.
<code>chi_square</code>	Logical. Compute chi-square test for cross-tabulations? Default: TRUE.
<code>fisher</code>	Logical. Also compute Fisher's exact test (for small samples)? Default: FALSE.
<code>cramers_v</code>	Logical. Compute Cramer's V effect size? Default: TRUE.
<code>digits</code>	Integer. Decimal places for percentages. Default: 1.
<code>sort_by</code>	How to sort categories: "none" (original order), "frequency" (descending), "alphabetical". Default: "none".
<code>labels</code>	Optional named character vector for category labels.
<code>title</code>	Optional character string for table title.
<code>subtitle</code>	Optional character string for table subtitle.
<code>format</code>	Character. Output format: "gt" (default, publication-ready gt table), "plain" (data frame), "markdown", "latex", or "kable".
<code>theme</code>	Character. Visual theme: "default", "minimal", "dark", or "colorful". Default: "default".
<code>combine</code>	Logical. For single variables, combine n and percent in one column? Default: TRUE. Shows as "n (pct)".
<code>interpret</code>	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Requires API key setup (see set_api_key).
<code>...</code>	Additional arguments passed to <code>pass</code> when <code>interpret = TRUE</code> (e.g., provider, model, context).

Value

A gt table object (default) or data frame with frequencies and percentages. For cross-tabulations with `chi_square = TRUE`, includes test statistics.

See Also

[descriptive_table](#) for numeric variable summaries [mosaic_analysis](#) for mosaic plot visualization with chi-square tests

Examples

```
## Not run:
# =====
# EXAMPLE 1: Single Variable Frequency Table
# =====
data <- data.frame(
  gender = sample(c("Male", "Female", "Other"), 200, replace = TRUE,
    prob = c(0.45, 0.45, 0.10)),
  education = sample(c("High School", "Bachelor", "Master", "PhD"), 200,
    replace = TRUE, prob = c(0.3, 0.4, 0.2, 0.1)),
  country = sample(c("USA", "UK", "Germany"), 200, replace = TRUE)
)

# Basic frequency table
categorical_table(data, var = "gender")

# With sorting by frequency
categorical_table(data, var = "education", sort_by = "frequency")

# =====
# EXAMPLE 2: Cross-Tabulation (Two Variables)
# =====
categorical_table(
  data = data,
  var = "gender",
  by = "education",
  chi_square = TRUE,
  cramers_v = TRUE
)

# =====
# EXAMPLE 3: Stratified Analysis
# =====
categorical_table(
  data = data,
  var = "gender",
  by = "education",
  group_by = "country"
)

# =====
# EXAMPLE 4: Custom Labels and Formatting
# =====
categorical_table(
  data = data,
  var = education,
  labels = c(
```

```

    "High School" = "Secondary Education",
    "Bachelor" = "Bachelor's Degree",
    "Master" = "Master's Degree",
    "PhD" = "Doctorate"
  ),
  title = "Educational Attainment",
  subtitle = "Sample Distribution",
  theme = "colorful"
)

# =====
# EXAMPLE 5: Show Row and Column Percentages
# =====
categorical_table(
  data = data,
  var = gender,
  by = education,
  percentages = c("row", "col"),
  combine = FALSE
)

## End(Not run)

```

center

Center Variables (Mean-Centering)

Description

Centers numeric variables by subtracting the mean. Supports group-wise centering for multi-level/nested data. Works with dplyr mutate/across.

Usage

```
center(data, Vars, suffix = "_c", group_by = NULL)
```

Arguments

data	A data frame (or vector if used in mutate).
Vars	Character vector of numeric variable names to center. Not needed if used in mutate.
suffix	Character. Suffix for new column names. Default "_c".
group_by	Character. Name of grouping variable for group-wise centering. Optional.

Value

Data frame with centered variables added, or vector if input is vector.

Examples

```
## Not run:
# Simple mean centering
df <- center(mtcars, Vars = c("mpg", "hp"))

# Group-mean centering
df <- center(mtcars, Vars = "mpg", group_by = "cyl")

# With dplyr
library(dplyr)
mtcars %>% mutate(mpg_c = center_vec(mpg))
mtcars %>% mutate(across(c(mpg, hp), center_vec, .names = "{.col}_c"))

# Group-wise with dplyr
mtcars %>% group_by(cyl) %>% mutate(mpg_c = center_vec(mpg))

## End(Not run)
```

center_vec	<i>Center a Vector (for use in mutate/across)</i>
------------	---

Description

Vectorized centering function for use with dplyr mutate and across. For group-wise centering, use dplyr::group_by() before mutate().

Usage

```
center_vec(x, na.rm = TRUE)
```

Arguments

x	Numeric vector to center.
na.rm	Logical. Remove NA values when computing mean? Default TRUE.

Value

Centered numeric vector.

Examples

```
## Not run:
library(dplyr)

# Single variable
mtcars %>% mutate(mpg_c = center_vec(mpg))

# Multiple variables
```

```
mtcars %>% mutate(across(c(mpg, hp, wt), center_vec, .names = "{.col}_c"))

# Group-wise centering
mtcars %>% group_by(cyl) %>% mutate(mpg_c = center_vec(mpg))

## End(Not run)
```

clustering

Latent Profile Clustering

Description

Fits latent profile models with the **latents** package and keeps the candidate with the lowest BIC. Every combination of `n_profiles` and `models` is fitted by `latents::enumerate_lpa()`; a single combination is simply a grid of one.

The result is a **latents** fit, so its verbs apply directly: `as.data.frame(fit)` for profile means, `latents::diagnostics(fit)` for classification quality, `latents::get_results(fit, what)` for any table.

Usage

```
clustering(
  data,
  vars,
  n_profiles = NULL,
  n_clusters = NULL,
  models = "VVI",
  scaling = c("standardize", "center", "minmax", "none"),
  n_starts = 10L,
  seed = NULL,
  na_action = c("omit", "fail"),
  verbose = TRUE
)
```

Arguments

<code>data</code>	A data frame.
<code>vars</code>	Character vector of numeric columns to cluster on.
<code>n_profiles</code>	Number(s) of profiles, each at least 2: one value, or a range such as 2:5 to compare.
<code>n_clusters</code>	Alias for <code>n_profiles</code> .
<code>models</code>	Covariance structure(s) as three-letter mclust codes (e.g. "EEE", "VVI"), or "all" for all 14.
<code>scaling</code>	Transformation applied to <code>vars</code> before fitting: "standardize" (mean 0, SD 1), "center", "minmax" (0 to 1), or "none".

n_starts	Random starts per candidate.
seed	Random seed passed to latents .
na_action	"omit" drops rows with a missing value in vars; "fail" raises an error instead.
verbose	Report progress with <code>message()</code> .

Value

An object of class `c("saqr_clustering", "multilpa")`: the selected **latents** fit, fitted to the scaled data. Methods: `print()`, `summary()` (one row per candidate), `fitted()` (the input data with profile assignments), `plot()`.

Errors

Raises `saqrmisc_bad_input` for unknown models, non-numeric or constant vars, missing values under `na_action = "fail"`, or fewer than 10 complete rows; `saqrmisc_no_converged` when no candidate converged.

References

Vermunt, J. K. (2003). Multilevel latent class models. *Sociological Methodology*, 33, 213-239.

See Also

`plot_clustering()`, `latents::enumerate_lpa()`

Examples

```
set.seed(1)
vars <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")
fit <- clustering(iris, vars, n_profiles = 2:4, models = c("EEE", "VVI"))
fit
summary(fit)
as.data.frame(fit)
fitted(fit)
plot(fit)
```

`clustering_plot_types` *List the Plot Types of `plot_clustering()`*

Description

The figures `plot_clustering()` can draw. Any type, any group, or "all" can be passed as its type.

Usage

```
clustering_plot_types()
```

Value

A data.frame with one row per plot type and columns type, group ("clusters", "diagnostics" or "selection") and description.

Examples

```
clustering_plot_types()
```

compare_groups

Generate Group Comparison Plots and Statistics

Description

A comprehensive function for comparing groups using publication-ready visualizations and statistical tests. Creates 'ggbetweenstats' plots from the ggstatsplot package with automatic test selection based on the number of groups:

For 2 groups: t-test (parametric) or Mann-Whitney U (nonparametric). For 3+ groups: ANOVA (parametric) or Kruskal-Wallis (nonparametric) with post-hoc pairwise comparisons. Also supports Bayesian analysis, equivalence testing (TOST), and stratified analyses using repeat_category.

Usage

```
compare_groups(
  data,
  category,
  Vars = NULL,
  category_sep = " | ",
  repeat_category = NULL,
  compare_by = NULL,
  compare_mode = c("between", "within"),
  repeat_combine = TRUE,
  repeat_levels = NULL,
  plots = TRUE,
  plot_style = c("points", "boxplot", "bar", "violin", "ggstatsplot"),
  table = TRUE,
  type = "auto",
  bayesian = FALSE,
  equivalence = FALSE,
  equivalence_bounds = c(-0.5, 0.5),
  nonparametric = FALSE,
  p_adjust_method = "fdr",
  posthoc = TRUE,
  posthoc_method = "games-howell",
  posthoc_table = FALSE,
  posthoc_format = c("wide", "long", "both"),
```

```

pairwise_display = "significant",
pivot = FALSE,
pivot_by = c("category", "repeat_category"),
pivot_stat = c("mean", "mean_sd", "median", "n"),
pivot_stars = TRUE,
pivot_split_level = TRUE,
min_threshold = 0.05,
min_subcategory = 5,
colors = NULL,
verbose = TRUE,
combined_table = TRUE,
format = c("gt", "plain", "markdown", "latex", "kable"),
show_header = TRUE,
interpret = FALSE,
...
)

```

Arguments

<code>data</code>	A data frame containing the variables to analyze.
<code>category</code>	Character. Name(s) of the grouping variable(s). Can be a single variable (e.g., "gender") or multiple variables for interaction groups (e.g., c("gender", "treatment") creates "Male Treatment", etc.).
<code>Vars</code>	Column specification for variables to compare. Can be NULL (default, all numeric columns), a character vector of column names, a numeric vector of column indices, or a single number (from that column to end).
<code>category_sep</code>	Character. Separator for combined group labels when multiple variables are passed to category. Default: " ".
<code>repeat_category</code>	Optional character. Name of a stratification variable. When provided, separate analyses are performed for each level (e.g., analyze gender differences separately for each country). Used with <code>compare_mode = "between"</code> .
<code>compare_by</code>	Optional character vector. Factors to test within each category level. Used with <code>compare_mode = "within"</code> . For example, if <code>category = "llm"</code> and <code>compare_by = c("pronoun", "support")</code> , tests whether pronoun and support have effects within each LLM dataset separately.
<code>compare_mode</code>	Character. Analysis mode: "between" (default) - compare category levels (e.g., GPT vs Mistral vs Qwen), "within" - test <code>compare_by</code> factors within each category level (e.g., test pronoun effect within GPT, within Mistral, within Qwen separately).
<code>repeat_combine</code>	Logical. When multiple <code>repeat_category</code> variables are provided with <code>compare_mode = "between"</code> , should they be combined into one grouping (e.g., "he/him High")? Default TRUE.
<code>repeat_levels</code>	Optional character vector. Specific levels of <code>repeat_category</code> to include in the analysis. If NULL (default), all levels are used.
<code>plots</code>	Logical. Generate visualizations? Default TRUE.

plot_style	Character. Style of plots: "points" (default, jittered with CI), "boxplot", "bar", "violin", or "ggstatsplot" (requires ggstatsplot package).
table	Logical. Generate summary statistics table? Default TRUE.
type	Character. Type of statistical test: "auto" (default), "parametric"/"p", "nonparametric"/"np", "kw" (Kruskal-Wallis), "mw" (Mann-Whitney), or "bayes"/"bf".
bayesian	Logical. Compute Bayesian t-tests with Bayes Factors? Default: FALSE. Requires the BayesFactor package. Can also use 'type = "bayes"'.
equivalence	Logical. Perform equivalence testing (TOST)? Default: FALSE. Requires the TOSTER package.
equivalence_bounds	Numeric vector of length 2. Equivalence bounds in Cohen's d units for TOST. Default: c(-0.5, 0.5).
nonparametric	Logical. Use nonparametric tests? Default: FALSE. When TRUE, uses Mann-Whitney U (2 groups) or Kruskal-Wallis (3+ groups).
p_adjust_method	Method for multiple comparison correction. Options: "none", "bonferroni", "holm", "hochberg", "hommel", "BH", "BY", "fdr". Default: "fdr" (Benjamini-Hochberg False Discovery Rate). Applied to both tables and post-hoc tests.
posthoc	Logical. Compute post-hoc pairwise comparisons for 3+ groups? Default: TRUE. Results include a comparison table and text report.
posthoc_method	Method for post-hoc comparisons: "games-howell" (default, does not assume equal variances) or "tukey" (Tukey's HSD, assumes equal variances).
posthoc_table	Logical. Include detailed post-hoc pairwise comparison table in the output? Default FALSE.
posthoc_format	Character. Format for post-hoc table when pivot = TRUE: "wide" (default, comparisons as columns), "long" (one row per comparison), or "both" (return both formats).
pairwise_display	Which pairwise comparisons to show on plots? Options: "significant" (default), "all", "none".
pivot	Logical. Pivot to wide format? Default FALSE. When TRUE, creates a wide table based on pivot_by parameter.
pivot_by	Character. Which variable to pivot to columns: "category" (default) - category groups become columns (e.g., GPT, Mistral, Qwen), "repeat_category" - repeat_category levels become columns (e.g., he/him, she/her).
pivot_stat	Character. Statistic to display in pivoted cells: "mean" (default), "mean_sd" (mean with SD in parentheses), "median", or "n" (count).
pivot_stars	Logical. Show significance stars in pivot table? Default TRUE when pivot = TRUE.
pivot_split_level	Logical. When multiple repeat_category variables are combined (e.g., "he/him High"), split them back into separate columns? Default TRUE. When FALSE, keeps the combined "level" column.

min_threshold	Numeric. Minimum proportion (0-1) of total sample required to include a repeat_category level. Default: 0.05 (5 percent).
min_subcategory	Integer. Minimum observations required per group. Groups with fewer observations are excluded from analysis. Default: 5.
colors	Character vector of colors for groups. Default: NULL uses a built-in palette.
verbose	Logical. Print progress messages? Default: TRUE.
combined_table	Logical. When 'repeat_category' is used, combine all results into a single table? Default: TRUE. The combined table shows all repeat_category levels together with bold highest means and red significant p-values.
format	Character. Output format for tables: "gt" (default, publication-ready), "plain" (data frame), "markdown", "latex", or "kable".
show_header	Logical. Show title/subtitle header? Default TRUE. Set to FALSE to hide the table header.
interpret	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Requires API key setup (see set_api_key).
...	Additional arguments passed to pass when interpret = TRUE (e.g., provider, model, context).

Value

A list with class "comparison_results" containing: plots (named list of ggplot objects), grid_plot (combined plot grid if multiple variables), summary_table (formatted gt table), and summary_data (data frame with raw statistics). For stratified analysis (with repeat_category), returns a named list where each element corresponds to a level of repeat_category. When pivot = TRUE, additional elements are returned:

- pivot_table: gt table with groups as columns and factor levels as rows
- posthoc_pivot_wide: Post-hoc comparisons in wide format (if posthoc_table = TRUE)
- posthoc_pivot_long: Post-hoc comparisons in long format (if posthoc_format = "long" or "both")

See Also

[ggbetweenstats](#) for the underlying plot function

Examples

```
## Not run:
# =====
# EXAMPLE 1: Basic Two-Group Comparison (t-test)
# =====
data <- data.frame(
  gender = rep(c("Male", "Female"), each = 50),
  score = c(rnorm(50, mean = 100, sd = 15),
            rnorm(50, mean = 105, sd = 15))
)
```

```

)

results <- compare_groups(
  data = data,
  category = "gender",
  Vars = c("score")
)

# View the plot
results$plots$gender_vs_score

# View summary statistics
results$summary_data

# =====
# EXAMPLE 2: Three-Group Comparison (ANOVA with post-hoc)
# =====
data <- data.frame(
  country = rep(c("USA", "UK", "Germany"), each = 40),
  score1 = c(rnorm(40, 75, 10), rnorm(40, 70, 11), rnorm(40, 85, 9)),
  score2 = c(rnorm(40, 50, 8), rnorm(40, 55, 9), rnorm(40, 48, 7))
)

results <- compare_groups(
  data = data,
  category = "country",
  Vars = c("score1", "score2"),
  posthoc = TRUE,
  posthoc_method = "tukey",
  pairwise_display = "all"
)

# View post-hoc pairwise comparisons
results$summary_data$posthoc_results[[1]]

# Print full ANOVA report
cat(results$summary_data$anova_report[1])

# =====
# EXAMPLE 3: Bayesian Analysis
# =====
results <- compare_groups(
  data = data,
  category = "country",
  Vars = c("score1"),
  bayesian = TRUE
)

# View Bayes Factors
results$summary_data[, c("country", "mean", "bf10", "bf_interpretation")]

# =====
# EXAMPLE 4: Nonparametric Tests

```

```

# =====
results <- compare_groups(
  data = data,
  category = "country",
  Vars = c("score1"),
  nonparametric = TRUE # Uses Kruskal-Wallis + pairwise Wilcoxon
)

# =====
# EXAMPLE 5: Stratified Analysis (separate tests per subgroup)
# =====
data <- data.frame(
  country = rep(c("USA", "UK", "Germany"), each = 60),
  gender = rep(c("Male", "Female"), times = 90),
  score = rnorm(180, mean = 100, sd = 15)
)

# Compare gender WITHIN each country
results <- compare_groups(
  data = data,
  category = "gender",
  Vars = c("score"),
  repeat_category = "country"
)

# Access results by country
results$USA$summary_data
results$UK$plots$gender_vs_score
results$Germany$summary_table

# =====
# EXAMPLE 6: Equivalence Testing (TOST)
# =====
results <- compare_groups(
  data = data,
  category = "gender",
  Vars = c("score"),
  equivalence = TRUE,
  equivalence_bounds = c(-0.3, 0.3) # Small effect bounds
)

# Check equivalence conclusions
results$summary_data[, c("gender", "mean", "equivalence_conclusion")]

# =====
# EXAMPLE 7: Combined/Interaction Groups
# =====
data <- data.frame(
  strength = rep(c("Strong", "Weak"), each = 60),
  gender = rep(c("Male", "Female"), times = 60),
  score = rnorm(120, mean = 100, sd = 15)
)

```

```

# Pass multiple variables to category for interaction groups
results <- compare_groups(
  data = data,
  category = c("strength", "gender"), # Creates "Strong | Male", "Strong | Female", etc.
  Vars = c("score")
)

# Custom separator
results <- compare_groups(
  data = data,
  category = c("strength", "gender"),
  category_sep = " x ", # Creates "Strong x Male", etc.
  Vars = c("score")
)

# =====
# EXAMPLE 8: Pivoted Table (groups as columns)
# =====
# LLM comparison data
data <- data.frame(
  Support_level = c(rnorm(100, 4, 1), rnorm(100, 6.5, 1), rnorm(100, 4.8, 1)),
  pronoun = rep(c("he/him", "she/her", "they/them"), each = 100),
  LA_activity = rep(c("Negative", "Positive"), 150),
  llm = rep(c("GPT", "Mistral", "Qwen"), each = 100)
)

# Basic pivot - groups become columns
results <- compare_groups(
  data = data,
  category = "llm", # Becomes columns: GPT | Mistral | Qwen
  Vars = "Support_level",
  repeat_category = "pronoun", # Rows: he/him, she/her, they/them
  pivot = TRUE # Enable pivot mode
)

# Access the pivot table
results$pivot_table

# =====
# EXAMPLE 9: Pivot with Mean and SD
# =====
results <- compare_groups(
  data = data,
  category = "llm",
  Vars = "Support_level",
  repeat_category = c("pronoun", "LA_activity"),
  pivot = TRUE,
  pivot_stat = "mean_sd" # Shows "4.32 (1.21)"
)

# =====
# EXAMPLE 10: Pivot with Post-hoc Comparisons Table
# =====

```

```

results <- compare_groups(
  data = data,
  category = "llm",
  Vars = "Support_level",
  repeat_category = "pronoun",
  pivot = TRUE,
  posthoc = TRUE,
  posthoc_table = TRUE,          # Include post-hoc table
  posthoc_format = "wide"       # "wide", "long", or "both"
)

# Access tables
results$pivot_table             # Main pivoted means table
results$posthoc_pivot_wide      # Post-hoc comparisons (wide format)

# =====
# EXAMPLE 11: Different P-value Adjustment Methods
# =====
# Available methods: "none", "bonferroni", "holm", "hochberg",
#                   "hommel", "BH", "BY", "fdr" (default)
results <- compare_groups(
  data = data,
  category = "llm",
  Vars = "Support_level",
  repeat_category = "pronoun",
  pivot = TRUE,
  p_adjust_method = "bonferroni"
)

## End(Not run)

```

compare_networks	<i>Compare Two Networks</i>
------------------	-----------------------------

Description

Performs Network Comparison Test (NCT) to compare network structure and global strength between two groups.

Usage

```
compare_networks(results, groups, it = 1000, test = "both", verbose = TRUE)
```

Arguments

results	Object from estimate_grouped_networks
groups	Character vector of length 2 specifying which groups to compare
it	Number of permutations for NCT. Defaults to 1000.

test	Which tests to perform: "network" (structure), "global" (strength), "both" (default).
verbose	Print progress. Defaults to TRUE.

Value

A list containing NCT results with p-values for network structure and global strength differences.

Examples

```
## Not run:
# First estimate grouped networks
grouped_results <- estimate_grouped_networks(df, Vars, GroupVar = "group")

# Then compare two groups
comparison <- compare_networks(
  grouped_results,
  groups = c("GroupA", "GroupB"),
  it = 500
)

## End(Not run)
```

convert_sequences	<i>Convert Sequence Data to Various Formats</i>
-------------------	---

Description

Converts long-form or wide-form sequence data into useful formats for analysis, such as frequency tables, one-hot encodings, or edge lists (for network analysis).

Usage

```
convert_sequences(
  data,
  seq_cols = NULL,
  id_col = NULL,
  format = c("frequency", "onehot", "edgelist", "follows"),
  na_values = NULL
)
```

Arguments

data	A data frame containing sequence data.
seq_cols	Character vector of sequence column names. If NULL (default), auto-detects all columns except the ID column.
id_col	Character. Name of the ID column. If NULL (default), uses the first column.

format	Character. Output format: "frequency" Counts of each action per sequence (default) "onehot" Binary (1/0) presence of each action per sequence "edgelist" From -> To transition pairs (for network analysis) "follows" Action -> Previous Action pairs
na_values	Character vector of values to treat as missing/NA. Default includes: NA, NaN, "", "*", " TraMineR typically uses "*" for missing states.

Value

A data frame structured according to the requested format:

- 'frequency': Wide format with action counts per ID
- 'onehot': Wide format with binary indicators per ID
- 'edgelist': Long format with columns: id, from, to
- 'follows': Long format with columns: id, act, follows

Examples

```
## Not run:
# Example sequence data
seq_data <- data.frame(
  id = 1:3,
  t1 = c("A", "B", "A"),
  t2 = c("B", "A", "C"),
  t3 = c("C", "C", "A"),
  t4 = c("A", "*", "B") # "*" is TraMineR missing
)

# Frequency table
convert_sequences(seq_data, format = "frequency")

# One-hot encoding
convert_sequences(seq_data, format = "onehot")

# Edge list for network analysis
convert_sequences(seq_data, format = "edgelist")

# What follows what
convert_sequences(seq_data, format = "follows")

## End(Not run)
```

correlation_matrix *Create a Publication-Ready Correlation Matrix*

Description

Creates a formatted correlation matrix with significance stars, optional confidence intervals, and heatmap visualization. Supports bivariate (zero-order), partial, and semi-partial correlations.

Usage

```
correlation_matrix(
  data,
  Vars = NULL,
  type = c("bivariate", "partial", "semi-partial"),
  method = c("pearson", "spearman", "kendall"),
  triangle = c("lower", "upper", "full"),
  diagonal = c("dash", "one", "names"),
  show_n = FALSE,
  show_ci = FALSE,
  show_p = FALSE,
  p_adjust = c("none", "bonferroni", "holm", "fdr"),
  stars = TRUE,
  heatmap = FALSE,
  digits = 2,
  title = NULL,
  use = c("pairwise", "complete"),
  format = c("gt", "plain", "markdown", "latex", "kable"),
  show_header = TRUE,
  interpret = FALSE,
  ...
)
```

Arguments

data	A data frame containing the variables to correlate.
Vars	Character vector of variable names to include in the correlation matrix. If NULL (default), all numeric variables in the data frame are used.
type	Type of correlation: "bivariate" (default, zero-order), "partial", or "semi-partial". For partial/semi-partial, each pair is controlled for all other variables in Vars.
method	Correlation method: "pearson" (default), "spearman", or "kendall".
triangle	Which triangle to display: "lower" (default), "upper", or "full".
diagonal	What to show on the diagonal: "dash" (default), "one", or "names".
show_n	Logical. Show pairwise sample sizes? Default FALSE.
show_ci	Logical. Show 95 percent confidence intervals? Default FALSE. Only available for bivariate Pearson.

show_p	Logical. Show p-values below correlations? Default FALSE.
p_adjust	Method for p-value adjustment: "none" (default), "bonferroni", "holm", "fdr".
stars	Logical. Show significance stars? Default TRUE.
heatmap	Logical. Create a heatmap visualization? Default FALSE.
digits	Number of decimal places. Default 2.
title	Optional title for the table.
use	Method for handling missing data: "pairwise" (default) or "complete". Note: partial and semi-partial correlations always use complete cases.
format	Character. Output format: "gt" (default, publication-ready gt table), "plain" (data frame), "markdown", "latex", or "kable".
show_header	Logical. Show title header? Default: 'TRUE'. Set to 'FALSE' to hide the table header.
interpret	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Requires API key setup (see set_api_key).
...	Additional arguments passed to pass when interpret = TRUE (e.g., provider, model, context, append_prompt).

Value

A list containing: table (gt table with formatted correlations), correlation_matrix (numeric matrix), p_matrix (matrix of p-values), n_matrix (pairwise sample sizes), and heatmap (ggplot if requested).

Examples

```
## Not run:
# Basic correlation matrix
correlation_matrix(
  data = mtcars,
  Vars = c("mpg", "cyl", "disp", "hp")
)

# Partial correlations (controlling for other variables)
correlation_matrix(
  data = mtcars,
  Vars = c("mpg", "cyl", "disp", "hp"),
  type = "partial",
  title = "Partial Correlations"
)

# Semi-partial correlations
correlation_matrix(
  data = mtcars,
  Vars = c("mpg", "cyl", "disp", "hp"),
  type = "semi-partial"
)

# With confidence intervals and heatmap
```

```

correlation_matrix(
  data = mtcars,
  Vars = c("mpg", "cyl", "disp", "hp"),
  show_ci = TRUE,
  heatmap = TRUE,
  title = "Motor Trend Car Correlations"
)

# Spearman correlations with Bonferroni correction
correlation_matrix(
  data = mtcars,
  Vars = c("mpg", "cyl", "disp"),
  method = "spearman",
  p_adjust = "bonferroni"
)

## End(Not run)

```

correlations

Full Pairwise Correlation Table

Description

Creates a comprehensive long-format correlation table showing all pairwise correlations with full statistics including r, confidence intervals, test statistics, degrees of freedom, p-values, and sample sizes. Supports multilevel/repeated measures correlations for nested data.

Usage

```

correlations(
  data,
  Vars = NULL,
  type = c("bivariate", "partial", "semi-partial"),
  method = c("pearson", "spearman", "kendall"),
  p_adjust = c("none", "bonferroni", "holm", "fdr"),
  ci_level = 0.95,
  min_r = NULL,
  sig_only = FALSE,
  multilevel = FALSE,
  id = NULL,
  between = FALSE,
  group_by = NULL,
  include = NULL,
  exclude = NULL,
  auto_consolidate = TRUE,
  digits = 3,
  title = NULL,
  format = c("gt", "plain", "markdown", "latex", "kable"),

```

```

    show_header = TRUE,
    interpret = FALSE,
    ...
)

```

Arguments

<code>data</code>	A data frame containing the variables to correlate.
<code>Vars</code>	Character vector of variable names to include. If NULL (default), all numeric variables in the data frame are used.
<code>type</code>	Type of correlation: "bivariate" (default), "partial", or "semi-partial".
<code>method</code>	Correlation method: "pearson" (default), "spearman", or "kendall".
<code>p_adjust</code>	Method for p-value adjustment: "none" (default), "bonferroni", "holm", "fdr".
<code>ci_level</code>	Confidence level for intervals. Default 0.95.
<code>min_r</code>	Numeric. Only show correlations with $ r \geq$ this value. Default NULL (show all).
<code>sig_only</code>	Logical. Only show significant correlations ($p < .05$)? Default FALSE.
<code>multilevel</code>	Logical. Calculate multilevel (within-cluster) correlations? Default FALSE. When TRUE, removes between-cluster variance to estimate within-cluster associations.
<code>id</code>	Character. Name of the clustering/ID variable for multilevel correlations. Required when <code>multilevel = TRUE</code> .
<code>between</code>	Logical. Also report between-cluster correlations? Default FALSE. Only used when <code>multilevel = TRUE</code> .
<code>group_by</code>	Character. Name of grouping variable to compute correlations separately per group. Results are combined into one table with a "Group" column.
<code>include</code>	Character vector. Statistics to include in the table. Options: "r", "ci", "stat" (t/S/z), "df", "p", "n", "sig". Default NULL includes all. Use this OR exclude, not both.
<code>exclude</code>	Character vector. Statistics to exclude from the table. Options: "r", "ci", "stat" (t/S/z), "df", "p", "n", "sig". Default NULL excludes none. Use this OR include, not both.
<code>auto_consolidate</code>	Logical. If TRUE (default), when df or n are constant across all pairs, they are moved to the subtitle instead of shown as columns.
<code>digits</code>	Number of decimal places. Default 3.
<code>title</code>	Optional title for the table.
<code>format</code>	Character. Output format: "gt" (default, publication-ready gt table), "plain" (data frame), "markdown", "latex", or "kable".
<code>show_header</code>	Logical. Show title/subtitle header? Default TRUE. Set to FALSE to hide the table header.
<code>interpret</code>	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Requires API key setup (see set_api_key).
<code>...</code>	Additional arguments passed to pass when <code>interpret = TRUE</code> (e.g., provider, model, context, <code>append_prompt</code>).

Details

When `multilevel = TRUE`, the function calculates within-cluster correlations by group-mean centering variables before computing correlations. This removes between-cluster variance and estimates the pooled within-cluster association, appropriate for repeated measures or hierarchically nested data.

The degrees of freedom for multilevel correlations are adjusted as: $df = n_{\text{observations}} - n_{\text{clusters}} - 1$

Value

A list containing: `table` (formatted output), `data` (raw statistics), `display` (formatted display data frame), `n_pairs`, `n_significant`, `consolidated` (values moved to subtitle), `between_data` (if multilevel).

Examples

```
## Not run:
# Correlate all numeric variables in data frame
correlations(mtcars)

# Full correlation table with specific variables
correlations(mtcars, Vars = c("mpg", "cyl", "disp", "hp", "wt"))

# Only strong correlations
correlations(mtcars, Vars = c("mpg", "cyl", "disp", "hp"), min_r = 0.5)

# Partial correlations with Bonferroni correction
correlations(mtcars, Vars = c("mpg", "cyl", "disp", "hp"),
             type = "partial", p_adjust = "bonferroni")

# Only significant correlations
correlations(mtcars, Vars = c("mpg", "cyl", "disp", "hp", "wt"),
             sig_only = TRUE)

# Multilevel correlations (within-person)
correlations(longitudinal_data, Vars = c("anxiety", "depression", "stress"),
             multilevel = TRUE, id = "participant_id")

# Multilevel with between-cluster correlations
correlations(longitudinal_data, Vars = c("anxiety", "depression"),
             multilevel = TRUE, id = "participant_id", between = TRUE)

# Correlations by group (stratified)
correlations(mtcars, Vars = c("mpg", "hp", "wt"), group_by = "cyl")

# Correlations by group with significance filter
correlations(mtcars, Vars = c("mpg", "hp", "wt", "disp"),
             group_by = "am", sig_only = TRUE)

# Include only specific statistics
correlations(mtcars, Vars = c("mpg", "hp", "wt"),
             include = c("r", "ci", "p", "sig"))
```

```
# Exclude statistics you don't need
correlations(mtcars, Vars = c("mpg", "hp", "wt"),
             exclude = c("stat", "df"))

# Disable auto-consolidation (always show df and n columns)
correlations(mtcars, Vars = c("mpg", "hp", "wt"),
             auto_consolidate = FALSE)

## End(Not run)
```

counts

Count Events per ID

Description

Converts long-format event data to a frequency table with counts per ID. Useful for transforming event logs, clickstream data, or any repeated measures into a wide format suitable for analysis.

Usage

```
counts(data, id = NULL, event = NULL, na_values = NULL)
```

Arguments

data	A data frame containing event data in long format.
id	Character vector. Name(s) of ID column(s). If multiple IDs are provided, they are combined. If NULL (default), uses the first column.
event	Character. Name of the event/action column to count. If NULL (default), uses the second column.
na_values	Character vector of values to treat as missing. Default includes common missing indicators.

Value

A data frame with one row per unique ID combination and columns for each unique event value containing counts.

Examples

```
## Not run:
# Event log data
events <- data.frame(
  student = c(1, 1, 1, 2, 2, 2, 3, 3),
  action = c("login", "view", "submit", "login", "view", "view", "login", "submit")
)
```

```
# Count events per student
counts(events, id = "student", event = "action")

# Multiple ID columns
events2 <- data.frame(
  student = c(1, 1, 1, 1, 2, 2),
  course = c("A", "A", "B", "B", "A", "A"),
  action = c("view", "submit", "view", "view", "view", "submit")
)

counts(events2, id = c("student", "course"), event = "action")

## End(Not run)
```

data_overview

Quick Dataset Overview

Description

Provides a comprehensive snapshot of a dataset including variable types, missing data percentages, unique values, and basic statistics.

Usage

```
data_overview(data, Vars = NULL, max_levels = 10, digits = 2)
```

Arguments

data	A data frame to summarize.
Vars	Character vector of variable names. If NULL (default), all variables are included.
max_levels	Integer. For categorical variables, show level counts if <= this value. Default 10.
digits	Integer. Number of decimal places for numeric summaries. Default 2.

Value

A gt table with dataset overview.

Examples

```
## Not run:
data_overview(mtcars)
data_overview(iris, Vars = c("Sepal.Length", "Species"))

## End(Not run)
```

descriptive_table	<i>Generate Publication-Ready Descriptive Statistics Table</i>
-------------------	--

Description

Creates a comprehensive, publication-ready table of descriptive statistics for numeric variables. Supports stratification by grouping variables and outputs beautifully formatted gt tables.

Usage

```
descriptive_table(
  data,
  Vars = NULL,
  group_by = NULL,
  stats = c("n", "mean", "sd", "median", "min", "max"),
  digits = 2,
  labels = NULL,
  title = "Descriptive Statistics",
  subtitle = NULL,
  overall = FALSE,
  transpose = FALSE,
  format = c("gt", "plain", "markdown", "latex", "kable"),
  show_header = TRUE,
  theme = "default",
  compare = FALSE,
  bold_highest = NULL,
  sig_color = "red",
  interpret = FALSE,
  ...
)
```

Arguments

<code>data</code>	A data frame containing the variables to summarize.
<code>Vars</code>	Column specification for variables to describe. Can be NULL (default, all numeric columns), a character vector of column names, a numeric vector of column indices, or a single number (from that column to end).
<code>group_by</code>	Optional character. Name of a grouping variable for stratified statistics. When provided, statistics are calculated separately for each group level.
<code>stats</code>	Character vector specifying which statistics to compute. Default: <code>c("n", "mean", "sd", "median", "min", "max")</code> . Available options: <code>"n", "missing", "missing_pct", "mean", "sd", "se", "var", "median", "min", "max", "range", "iqr", "q1", "q3", "skewness", "kurtosis", "cv"</code> .
<code>digits</code>	Integer. Number of decimal places for numeric output. Default: <code>'2'</code> .
<code>labels</code>	Optional named character vector for variable labels. Example: <code>'c(age = "Age (years)", score1 = "Test Score")'</code> . If NULL, variable names are used as-is.

title	Optional character string for table title. Default: "Descriptive Statistics".
subtitle	Optional character string for table subtitle.
overall	Logical. When 'group_by' is specified, also include overall (ungrouped) statistics? Default: FALSE.
transpose	Logical. Transpose the table so variables are columns and statistics are rows? Default: FALSE.
format	Character. Output format: "gt" (default, publication-ready gt table), "plain" (data frame), "markdown", "latex", or "kable".
show_header	Logical. Show title/subtitle header? Default TRUE. Set to FALSE to hide the table header.
theme	Character. Visual theme for gt table: "default" (clean scientific style), "fancy" (blue with striped rows), "minimal" (bottom border only), "dark" (dark background), or "colorful" (purple accents). Default: "default".
compare	Logical. When 'group_by' is specified, run statistical tests (t-test for 2 groups, ANOVA for 3+ groups) and report p-values and effect sizes? Default: FALSE.
bold_highest	Logical. When 'group_by' is specified, bold the highest mean for each variable? Default: TRUE when compare is 'TRUE'.
sig_color	Character. Color for significant p-values (< 0.05). Default: "red". Set to 'NULL' to disable coloring.
interpret	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Requires API key setup (see set_api_key).
...	Additional arguments passed to pass when interpret = TRUE (e.g., provider, model, context, append_prompt).

Value

A saqr_result object containing:

- table: The formatted gt table (or other format if specified)
- data: Raw data frame with computed statistics
- markdown: Markdown version of the table for AI interpretation
- type: "descriptive"
- variables: Variables analyzed
- statistics: Statistics computed
- n: Total sample size

When printed, displays the formatted table. Use `result$markdown` to get a text version suitable for [pass](#).

Exporting Tables

The gt output can be exported using `gtsave()`: HTML (.html), Word (.docx), PDF (.pdf), or PNG (.png). Example: `gtsave(table, "descriptives.html")`.

See Also

[gt](#) for gt table customization

Examples

```
## Not run:
# =====
# EXAMPLE 1: Basic Descriptive Statistics
# =====
data <- data.frame(
  age = rnorm(100, mean = 35, sd = 10),
  score = rnorm(100, mean = 75, sd = 15),
  income = rnorm(100, mean = 50000, sd = 15000)
)

# Default statistics (n, mean, sd, median, min, max)
descriptive_table(data, Vars = c("age", "score", "income"))

# =====
# EXAMPLE 2: Custom Statistics Selection
# =====
descriptive_table(
  data = data,
  Vars = c("age", "score"),
  stats = c("n", "mean", "sd", "se", "median", "iqr", "skewness", "kurtosis")
)

# =====
# EXAMPLE 3: Stratified by Group
# =====
data$gender <- sample(c("Male", "Female"), 100, replace = TRUE)

descriptive_table(
  data = data,
  Vars = c("age", "score"),
  group_by = gender,
  overall = TRUE # Include overall statistics
)

# =====
# EXAMPLE 4: Custom Labels and Title
# =====
descriptive_table(
  data = data,
  Vars = c("age", "score", "income"),
  labels = c(
    age = "Age (years)",
    score = "Test Score",
    income = "Annual Income ($)"
  ),
  title = "Sample Characteristics",
  subtitle = "N = 100 participants"
```

```

)

# =====
# EXAMPLE 5: Export to Data Frame
# =====
df <- descriptive_table(
  data = data,
  Vars = c("age", "score"),
  format = "data.frame"
)
print(df)

# =====
# EXAMPLE 6: Different Themes
# =====
descriptive_table(data, Vars = c("age", "score"), theme = "minimal")
descriptive_table(data, Vars = c("age", "score"), theme = "dark")
descriptive_table(data, Vars = c("age", "score"), theme = "colorful")

## End(Not run)

```

estimate_grouped_networks

Estimate Grouped Networks

Description

Estimates networks for different groups within a data frame, using ‘estimate_single_network’ internally. Produces separate network visualizations and metrics for each group.

Usage

```

estimate_grouped_networks(
  df,
  Vars = NULL,
  GroupVar,
  layout = "circle",
  color = DEFAULT_NODE_COLOR,
  default = "EBICglasso",
  var_types = NULL,
  qgraph_args = list(),
  network_args = list(),
  verbose = TRUE,
  min_group_size = 30
)

```

Arguments

df	A data frame containing the variables and group variable.
Vars	Character vector of variable names to include in the network. If NULL (default), all columns except GroupVar are used.
GroupVar	Character string specifying the grouping variable name.
layout	Layout algorithm for network plots. Defaults to "circle".
color	Node color(s). Defaults to "#EEEEEE".
default	Network estimation method. Defaults to "EBICglasso".
var_types	Variable types for MGM. If NULL, auto-detected.
qgraph_args	Additional arguments for qgraph.
network_args	Additional arguments for bootnet::estimateNetwork.
verbose	Print progress messages. Defaults to TRUE.
min_group_size	Minimum group size required. Groups with fewer observations are skipped. Defaults to 30.

Value

A list of class "grouped_network_analysis" containing:

- networks: List of network objects per group
- network_matrices: List of adjacency matrices per group
- predictions: List of prediction results per group
- centralities: List of centrality measures per group
- qgraphs: List of qgraph plot objects per group
- group_sizes: Sample sizes per group
- failed_groups: Groups that failed estimation

Examples

```
## Not run:
# Estimate networks by group
results <- estimate_grouped_networks(
  df = my_data,
  Vars = c("var1", "var2", "var3", "var4"),
  GroupVar = "group"
)

# Access results for specific group
results$networks$GroupA
results$predictions$GroupA

# Compare networks
compare_networks(results, groups = c("GroupA", "GroupB"))

## End(Not run)
```

estimate_single_network

Estimate a Single Network

Description

Estimates a single psychological network from a data frame using regularized partial correlations (via bootnet) and Mixed Graphical Models (via mgm). Provides network visualization with predictability metrics (R-squared pie charts).

Usage

```
estimate_single_network(
  df,
  Vars = NULL,
  layout = "circle",
  color = DEFAULT_NODE_COLOR,
  title = "Between-person network",
  default = "EBICglasso",
  var_types = NULL,
  qgraph_args = list(),
  network_args = list(),
  verbose = TRUE,
  compute_centrality = TRUE
)
```

Arguments

df	A data frame containing the variables to be analyzed.
Vars	A character vector specifying the names of variables to include. If NULL (default), all columns of the data frame are used.
layout	Character string specifying the layout algorithm for the network plot. Options: "circle", "spring", "groups", etc. Defaults to "circle".
color	Character string or vector specifying node colors. Defaults to "#EEEEEE".
title	Title for the network plot. Defaults to "Between-person network".
default	Network estimation method for bootnet::estimateNetwork. Options: "EBICglasso" (default), "ggmModSelect", "pcor", etc.
var_types	Character vector of variable types for MGM ("g" = Gaussian, "c" = categorical, "p" = Poisson). If NULL (default), types are auto-detected.
qgraph_args	List of additional arguments passed to qgraph::qgraph.
network_args	List of additional arguments passed to bootnet::estimateNetwork.
verbose	Logical. If TRUE, prints progress messages. Defaults to TRUE.
compute_centrality	Logical. If TRUE, computes centrality measures. Defaults to TRUE.

Value

A list of class "network_analysis" containing:

- network_object: Full network object from bootnet
- network_matrix: Adjacency/weight matrix
- prediction: MGM prediction results (R2, RMSE)
- centrality: Centrality measures (if compute_centrality = TRUE)
- qgraph: The qgraph plot object
- var_types: Variable types used for MGM

Examples

```
## Not run:
# Basic usage
result <- estimate_single_network(
  df = my_data,
  Vars = c("var1", "var2", "var3", "var4", "var5")
)

# Access network matrix
result$network_matrix

# View predictability
result$prediction

# View centrality
result$centrality

# Custom settings
result <- estimate_single_network(
  df = my_data,
  Vars = c("var1", "var2", "var3"),
  layout = "spring",
  default = "ggmModSelect",
  var_types = c("g", "g", "c"), # Manual type specification
  verbose = FALSE
)

## End(Not run)
```

fitted.sqr_clustering

Input Data with Profile Assignments

Description

Input Data with Profile Assignments

Usage

```
## S3 method for class 'saqr_clustering'
fitted(object, ...)
```

Arguments

object A 'saqr_clustering' object from [clustering()].

... Ignored.

Value

The data frame passed to [clustering()], one row per input row, with 'profile' (modal assignment), 'uncertainty' and one 'posterior_profile_<k>' column per profile added. Rows dropped for missing values carry 'NA' in the added columns.

is_outlier	<i>Check if Value is Outlier (vectorized for mutate/across)</i>
------------	---

Description

Vectorized function to detect outliers for use with dplyr. For group-wise detection, use dplyr::group_by() before mutate().

Usage

```
is_outlier(
  x,
  method = c("zscore", "iqr", "percentile"),
  threshold = NULL,
  na.rm = TRUE
)
```

Arguments

x Numeric vector.

method Detection method: "zscore", "iqr", or "percentile".

threshold Threshold value.

na.rm Logical. Remove NA values? Default TRUE.

Value

Logical vector indicating outliers.

Examples

```
## Not run:
library(dplyr)

# Flag outliers beyond 2 SD
mtcars %>% mutate(hp_outlier = is_outlier(hp, method = "zscore", threshold = 2))

# Flag outliers at 5th/95th percentile
mtcars %>% mutate(hp_outlier = is_outlier(hp, method = "percentile", threshold = 0.05))

# Group-wise outlier detection
mtcars %>% group_by(cyl) %>% mutate(hp_outlier = is_outlier(hp))

## End(Not run)
```

launch_app

Launch the Saqrmisc mosaic analysis Shiny app

Description

Opens an interactive Shiny application for [mosaic_analysis](#). Upload a CSV (or use the built-in demo), pick two categorical variables, and explore the chi-square / Fisher test, Cramer's V effect size, standardized residuals, and the shaded mosaic plot. Every `mosaic_analysis()` option is exposed in the sidebar.

Requires the **shiny** and **DT** packages.

Usage

```
launch_app(...)
```

Arguments

... Passed to [runApp](#) (e.g. `port`, `launch.browser`, `host`).

Value

Called for its side effect (launches the app). No return value.

Examples

```
if (interactive()) {
  launch_app()
}
```

missing_analysis	Analyze Missing Data Patterns
------------------	-------------------------------

Description

Provides comprehensive missing data analysis including patterns, Little's MCAR test, and visualizations.

Usage

```
missing_analysis(  
  data,  
  Vars = NULL,  
  pattern_plot = TRUE,  
  mcar_test = TRUE,  
  correlations = FALSE,  
  digits = 2  
)
```

Arguments

data	A data frame to analyze.
Vars	Character vector of variable names. If NULL (default), all variables are included.
pattern_plot	Logical. Create missing data pattern visualization? Default TRUE.
mcar_test	Logical. Perform Little's MCAR test? Default TRUE.
correlations	Logical. Show correlations between missingness indicators? Default FALSE.
digits	Integer. Number of decimal places. Default 2.

Value

A list containing:

- summary: gt table with missing data summary
- patterns: Data frame of missing patterns
- mcar: MCAR test results (if requested)
- plot: Missing pattern plot (if requested)

Examples

```
## Not run:  
# Create data with missing values  
df <- mtcars  
df$mpg[c(1, 5, 10)] <- NA  
df$hp[c(2, 5, 15)] <- NA
```

```
missing_analysis(df)

## End(Not run)
```

mosaic_analysis	<i>Perform a comprehensive mosaic plot analysis</i>
-----------------	---

Description

This function conducts a complete mosaic plot analysis for two categorical variables. It includes filtering by minimum count, chi-square testing (with optional Fisher's exact test), calculation of Cramer's V with effect size interpretation, and generation of a detailed summary table with observed/expected counts and residuals.

Usage

```
mosaic_analysis(
  data,
  var1,
  var2,
  by = NULL,
  min_count = 10,
  fontsize = 8,
  title = "",
  var1_label = NULL,
  var2_label = NULL,
  show_varnames = FALSE,
  plot_style = c("flat", "classic"),
  tile_label = c("count", "percent", "count_percent", "residual", "category", "none"),
  col_label_side = c("top", "bottom", "both", "none"),
  row_label_side = c("left", "right", "both", "none"),
  col_label_angle = 0,
  row_label_angle = 0,
  show_legend = TRUE,
  legend_position = "right",
  legend_size = 0.7,
  legend_title = "Std.\nresidual",
  label_size = 3.5,
  show_percentages = TRUE,
  percentage_base = "total",
  use_fisher = FALSE,
  by_label = NULL,
  min_stratum_n = 30,
  p_adjust = "BH",
  facet_ncol = NULL,
  facet_show_n = TRUE,
```

```

    seed = NULL,
    verbose = TRUE,
    save_plot = NULL,
    interpret = FALSE,
    ...
)

```

Arguments

<code>data</code>	A data frame containing the variables.
<code>var1</code>	Character. Name of the first categorical variable.
<code>var2</code>	Character. Name of the second categorical variable.
<code>by</code>	Character. Optional name of a third categorical variable to stratify (facet) on. When supplied, the <code>var1-var2</code> table is fitted separately within each level of <code>by</code> and the plot becomes a panel of mosaics, one per stratum. Category filtering (<code>min_count</code>) is applied to the pooled table <i>before</i> splitting, so every panel shows the same rows and columns; the residual colour scale is likewise shared across panels. Defaults to <code>NULL</code> (a single, un-stratified mosaic).
<code>min_count</code>	The minimum number of observations required for a category to be included in the analysis. Defaults to 10.
<code>fontsize</code>	The font size for the mosaic plot labels. Defaults to 8.
<code>title</code>	The title for the mosaic plot. Defaults to "".
<code>var1_label</code>	The label for the first variable in the plot. If <code>NULL</code> , uses the variable name.
<code>var2_label</code>	The label for the second variable in the plot. If <code>NULL</code> , uses the variable name.
<code>show_varnames</code>	Logical. If <code>TRUE</code> , draws the variable-name titles along the plot axes (e.g. the row and column variable names). Defaults to <code>FALSE</code> because these titles often overprint the category labels, especially when a thin category sits next to them. The category labels and column headers remain visible either way.
<code>plot_style</code>	The mosaic plot style: "flat" (a modern flat ggplot2 mosaic shaded by standardized residuals; the default) or "classic" (the vcd shaded mosaic).
<code>tile_label</code>	(flat style) What to print inside each tile: "count" (the default actual counts), "percent" (using <code>percentage_base</code>), "count_percent" (the count with its percentage on a second line), "residual" (standardized residual), "category" (the second-variable level name), or "none".
<code>col_label_side</code>	(flat style) Placement of the first-variable labels: "top" (default), "bottom", "both", or "none".
<code>row_label_side</code>	(flat style) Placement of the second-variable labels: "left" (default), "right", "both", or "none".
<code>col_label_angle, row_label_angle</code>	(flat style) Rotation in degrees for the column and row category labels (0 = horizontal, 90 = vertical).
<code>show_legend</code>	Logical. Show the residual colour legend. Defaults to <code>TRUE</code> .
<code>legend_position</code>	Legend placement for the flat style: one of "right" (default), "left", "top", "bottom", or "none".

legend_size	Numeric multiplier (> 0) scaling the legend key and text in the flat style. Smaller is more compact. Defaults to 0.7.
legend_title	Legend title for the flat style. Defaults to "Std.\nresidual".
label_size	Tile-label text size for the flat style. Defaults to 3.5.
show_percentages	Logical. If TRUE, includes percentages in the summary table. Defaults to TRUE.
percentage_base	The base for calculating percentages ("total", "row", or "column"). Defaults to "total".
use_fisher	Logical. If TRUE, uses Fisher's exact test instead of chi-square (recommended for small expected cell counts). Defaults to FALSE.
by_label	Label for the stratifying variable. If NULL, uses by.
min_stratum_n	Minimum number of observations for a stratum to be fitted. Strata below this, or no longer spanning at least two levels of each variable, are dropped with a warning naming them. Defaults to 30.
p_adjust	Multiplicity correction applied to the per-stratum p-values, passed to p.adjust . Defaults to "BH". Fitting one test per stratum is a multiple-testing problem, so the corrected value is reported alongside the raw one.
facet_ncol	Number of facet columns in a stratified plot. NULL (the default) lets ggplot2 choose.
facet_show_n	Logical. Append "(n = ...)" to each panel strip so equal panel widths never imply equal sample sizes. Defaults to TRUE.
seed	Optional integer. Seeds the Monte-Carlo Fisher p-value so a fit is reproducible; the caller's RNG stream is restored on exit. Defaults to NULL (no seeding).
verbose	Logical. If TRUE, prints results to console. Defaults to TRUE.
save_plot	Optional file path to save the mosaic plot. Supports .png, .pdf, .svg. Defaults to NULL (no saving).
interpret	Logical. Pass results to AI for automatic interpretation? Default FALSE. When TRUE, generates clean Methods and Results text using AI. Includes chi-square/Fisher test results and Cramer's V effect size. Requires API key setup (see set_api_key).
...	Additional arguments passed to pass when interpret = TRUE (e.g., provider, model, context, append_prompt).

Value

A list of class "mosaic_analysis" containing:

- plot: The mosaic plot object
- consolidated_table: Tibble with observed, expected counts and percentages
- residuals: Data frame of standardized Pearson residuals
- chi_test: Chi-square test results (or Fisher's test if use_fisher=TRUE)
- cramers_v: Cramer's V effect size value

- `cramers_v_interpretation`: Effect size interpretation (negligible/small/medium/large)
- `stats_summary`: Tibble summarizing all statistical results
- `filtered_data`: The filtered data used for analysis
- `original_n`: Original sample size before filtering
- `filtered_n`: Sample size after filtering
- `removed_categories`: List of categories removed due to `min_count`

When `by` is supplied the object additionally gains class `"mosaic_stratified"` and the fields `strata_summary` (one row per stratum: `n`, `test`, `statistic`, `df`, `raw` and `adjusted p`, Cramer's V, effect size, and the number of cells beyond `l2l`), `strata_residuals` (one row per stratum x cell), `strata_table`, and `overall_summary` (the pooled test beside the Cochran-Mantel-Haenszel test of association conditional on the stratifier). Reach these with `as.data.frame`, e.g. `as.data.frame(fit, what = "strata")`.

Examples

```
# Create example data
set.seed(123)
example_data <- data.frame(
  gender = sample(c("Male", "Female"), 200, replace = TRUE),
  education = sample(c("High School", "Bachelor", "Master", "PhD"), 200,
    replace = TRUE, prob = c(0.3, 0.4, 0.2, 0.1))
)

# Basic usage with quoted variable names
results <- mosaic_analysis(example_data, "gender", "education")

# Access results
results$cramers_v
results$cramers_v_interpretation
results$stats_summary

# With row percentages and custom labels
results <- mosaic_analysis(
  example_data, "gender", "education",
  min_count = 5,
  var1_label = "Gender",
  var2_label = "Education Level",
  percentage_base = "row"
)

# Using Fisher's exact test for small samples
results <- mosaic_analysis(
  example_data, "gender", "education",
  use_fisher = TRUE,
  verbose = FALSE
)

# Stratify (facet) on a third variable: one mosaic and one test per region,
# with p-values corrected across strata.
```

```

example_data$region <- sample(c("North", "South"), 200, replace = TRUE)
by_region <- mosaic_analysis(
  example_data, "gender", "education",
  by = "region", min_count = 5, min_stratum_n = 20, verbose = FALSE
)
as.data.frame(by_region, what = "strata")
as.data.frame(by_region, what = "overall")

```

normality_check	<i>Check Normality of Variables</i>
-----------------	-------------------------------------

Description

Tests and visualizes normality assumptions for numeric variables using Shapiro-Wilk test, skewness, kurtosis, and Q-Q plots.

Usage

```
normality_check(data, Vars, tests = TRUE, plots = TRUE, digits = 3)
```

Arguments

data	A data frame containing the variables.
Vars	Character vector of numeric variable names to check.
tests	Logical. Perform normality tests? Default TRUE.
plots	Logical. Create Q-Q plots? Default TRUE.
digits	Integer. Number of decimal places. Default 3.

Value

A list containing:

- table: A gt table with normality statistics
- plots: Q-Q plots (if requested)
- results: Data frame with all statistics

Examples

```

## Not run:
normality_check(mtcars, Vars = c("mpg", "hp", "wt"))

## End(Not run)

```

outlier_check	<i>Check for Outliers</i>
---------------	---------------------------

Description

Detects univariate and multivariate outliers using various methods including z-score (2SD, 3SD, etc.), IQR, percentile, or Mahalanobis distance.

Usage

```
outlier_check(
  data,
  Vars,
  method = c("zscore", "iqr", "percentile", "mahalanobis"),
  threshold = NULL,
  flag = TRUE,
  plot = TRUE
)
```

Arguments

data	A data frame containing the variables.
Vars	Character vector of numeric variable names to check.
method	Detection method: "zscore", "iqr", "percentile", or "mahalanobis".
threshold	Numeric threshold for outlier detection. For zscore: number of SDs (default 3). Common values: 2, 2.5, 3, 3.29. For iqr: IQR multiplier (default 1.5). Common values: 1.5, 3. For percentile: percentile cutoff (default 0.01 for 1st/99th). Values like 0.05 for 5th/95th. For mahalanobis: chi-sq p-value threshold (default 0.001).
flag	Logical. Add outlier flag column to returned data? Default TRUE.
plot	Logical. Create outlier visualization? Default TRUE.

Value

A list containing:

- summary: gt table with outlier summary
- data: Data frame with outlier flags (if flag = TRUE)
- outlier_indices: Row indices of outliers
- plot: Visualization (if requested)

Examples

```
## Not run:
# Z-score with 3 SD threshold
outlier_check(mtcars, Vars = c("mpg", "hp"), method = "zscore", threshold = 3)

# Z-score with 2 SD threshold (more conservative)
outlier_check(mtcars, Vars = c("mpg", "hp"), method = "zscore", threshold = 2)

# IQR method
outlier_check(mtcars, Vars = c("mpg", "hp"), method = "iqr", threshold = 1.5)

# Percentile method (1st and 99th percentile)
outlier_check(mtcars, Vars = c("mpg", "hp"), method = "percentile", threshold = 0.01)

# Percentile method (5th and 95th percentile)
outlier_check(mtcars, Vars = c("mpg", "hp"), method = "percentile", threshold = 0.05)

# Mahalanobis distance (multivariate)
outlier_check(mtcars, Vars = c("mpg", "hp", "wt"), method = "mahalanobis")

## End(Not run)
```

pass

Pass R Output to AI for Interpretation

Description

Pipes any R output (test results, model summaries, tables) to an AI model for scientific interpretation. Perfect for getting publication-ready interpretations of statistical results.

Usage

```
pass(
  x,
  prompt = NULL,
  append_prompt = NULL,
  action = c("write", "interpret", "explain", "summarize", "critique", "suggest"),
  style = c("scientific", "simple", "detailed", "brief"),
  output = c("text", "markdown", "md", "latex", "html"),
  provider = c("openai", "anthropic", "gemini", "openrouter"),
  model = NULL,
  base_url = NULL,
  api_key = NULL,
  context = NULL,
  system_message = NULL,
  auto_local = TRUE,
  copy = FALSE,
```

```

    quiet = FALSE
  )

```

Arguments

<code>x</code>	Any R object to interpret (test result, model, data frame, etc.)
<code>prompt</code>	Custom prompt to use. If NULL, builds from action and style. Note: This REPLACES the default prompt entirely.
<code>append_prompt</code>	Additional instructions to ADD to the default prompt. Unlike ‘prompt’, this appends to (not replaces) the auto-generated prompt.
<code>action</code>	What to do with the output: • “write” (default): Write publication-ready text (methods/results) • “interpret”: Interpret the statistical results • “explain”: Explain what the analysis does and means • “summarize”: Brief summary of key findings • “critique”: Critical evaluation with limitations • “suggest”: Suggest follow-up analyses
<code>style</code>	Interpretation style: • “scientific” (default): APA-style for academic papers • “simple”: Plain language, no jargon • “detailed”: Comprehensive with assumptions, limitations, caveats • “brief”: Just the key takeaway
<code>output</code>	Output format: • “text” (default): Plain text • “markdown” or “md”: Markdown formatted • “latex”: LaTeX formatted for papers • “html”: HTML formatted
<code>provider</code>	AI provider: “openai” (default), “anthropic”, “gemini”, or “openrouter”.
<code>model</code>	Model to use. Defaults: “gpt-4.1-nano” (OpenAI), “claude-sonnet-4-20250514” (Anthropic), “gemini-2.5-flash” (Gemini), “anthropic/claude-sonnet-4” (Open-Router).
<code>base_url</code>	Custom API base URL for OpenAI-compatible servers (e.g., LM Studio, Ollama, vLLM). Example: “http://127.0.0.1:1234” for LM Studio. When set, uses OpenAI-compatible format regardless of provider setting.
<code>api_key</code>	API key. If NULL, checks environment variables (‘OPENAI_API_KEY’, ‘ANTHROPIC_API_KEY’, ‘GEMINI_API_KEY’, or ‘OPENROUTER_API_KEY’), then prompts interactively. For local servers like LM Studio, use ‘api_key = “none”’ or any string.
<code>context</code>	Optional context about your study (e.g., “This is a study on student learning outcomes with N=500 participants”)
<code>system_message</code>	Optional custom instructions for the AI (e.g., “Focus on clinical implications”, “Be more concise”, “Emphasize effect sizes”)

auto_local	Logical. Automatically detect and use local AI servers (LM Studio on port 1234, Ollama on port 11434)? Default: TRUE. Set to FALSE to force using cloud providers.
copy	Logical. Copy result to clipboard? Default: FALSE
quiet	Logical. Suppress messages? Default: FALSE

Details

On first use, you'll be prompted to enter your API key. The key is stored in your R environment for the session. To persist it, add to your .Renviron: ““ ANTHROPIC_API_KEY=your-key-here # or OPENAI_API_KEY=your-key-here ““

Value

Character string with the AI interpretation (invisibly). Also prints the interpretation.

Examples

```
## Not run:
# Basic usage - pipe test results
t.test(mpg ~ am, data = mtcars) |> pass()

# With context
cor.test(mtcars$mpg, mtcars$hp) |>
  pass(context = "Studying fuel efficiency in 1974 automobiles")

# Different actions
lm(mpg ~ wt + hp, data = mtcars) |> summary() |> pass(action = "write")
chisq.test(mtcars$cyl, mtcars$am) |> pass(action = "explain", style = "simple")

# Get LaTeX output for paper
aov(mpg ~ factor(cyl), data = mtcars) |> summary() |>
  pass(action = "write", output = "latex")

# Critique an analysis
lm(mpg ~ ., data = mtcars) |> summary() |> pass(action = "critique")

# Custom prompt (REPLACES default - specific request to the AI)
my_results |> pass(prompt = "Focus only on the interaction effects")

# Add to default prompt (keeps "write methods/results" + your addition)
my_results |> pass(append_prompt = "Also mention limitations of the sample size")
my_results |> pass(action = "write", append_prompt = "Include a brief discussion section")

# Custom context (about your study)
t.test(score ~ group, data = mydata) |>
  pass(context = "RCT comparing drug vs placebo, N=200 patients with diabetes")

# Custom system message (instructions for the AI)
my_results |> pass(system_message = "Focus on clinical implications and effect sizes")
```

```
# Combine all customizations
lm(outcome ~ treatment * age, data = mydata) |> summary() |>
  pass(
    action = "write",
    context = "Phase 3 clinical trial for hypertension medication",
    system_message = "Emphasize clinical significance over statistical significance",
    prompt = "Pay special attention to the treatment-age interaction"
  )

## End(Not run)
```

plot.mosaic_analysis *Plot method for mosaic_analysis objects*

Description

Re-renders the mosaic plot from a fitted `mosaic_analysis` object using the stored contingency table and residuals, so styling can be changed without re-running the statistical test. Any styling argument accepted by `mosaic_analysis` (e.g. `plot_style`, `tile_label`, `col_label_side`, `legend_size`) can be overridden via `...`; unspecified arguments keep the values from the original call.

Usage

```
## S3 method for class 'mosaic_analysis'
plot(x, ...)
```

Arguments

`x` A `mosaic_analysis` object.

`...` Styling overrides (see `mosaic_analysis`).

Value

Invisibly, the plot object (a `ggplot` for the flat style). Called for its side effect of drawing the plot.

Examples

```
set.seed(1)
d <- data.frame(
  a = sample(c("X", "Y", "Z"), 200, replace = TRUE),
  b = sample(c("P", "Q"), 200, replace = TRUE)
)
res <- mosaic_analysis(d, "a", "b", min_count = 5, verbose = FALSE)

plot(res, tile_label = "percent", legend_size = 0.4)
plot(res, plot_style = "classic")
```

plot.saqr_clustering *Plot Method for Clustering Results*

Description

Plot Method for Clustering Results

Usage

```
## S3 method for class 'saqr_clustering'
plot(x, type = "clusters", ...)
```

Arguments

x	A 'saqr_clustering' object from [clustering()].
type	Plot types and/or groups; see [plot_clustering()].
...	Passed to [plot_clustering()].

Value

As [plot_clustering()]: a ggplot object, or a 'saqr_plots' list.

plot_clustering *Plot a Clustering Result*

Description

Builds figures of a [clustering\(\)](#) result with **latents**, as ggplot objects. type takes any mix of plot types and groups:

- "clusters": "profiles", "bars", "heatmap", "raincloud", "parallel", "pairs", "sizes".
- "diagnostics": "entropy", "posteriors", "avepp".
- "selection": "enumeration" (information criteria across the candidates) and "tree" (how profiles split as more are added); both need more than one candidate.
- "all": every type above.

Usage

```
plot_clustering(results, type = "clusters", ...)
```

Arguments

results	A saqr_clustering object from clustering() .
type	Plot types and/or groups; see clustering_plot_types() .
...	Passed to the latents plot method of the selected fit, e.g. scale = "standardized" or main.

Value

One type: a ggplot object. Several: a saqr_plots list of ggplot objects named by type, which draws every one when printed. Print, save with `ggplot2::ggsave()`, or restyle with `+ ggplot2::theme()`.

Errors

Raises `saqrmisc_bad_input` when results is not a `clustering()` result and `saqrmisc_bad_type` for an unknown type. A selection view that needs more candidates than were fitted is skipped with a message.

See Also

`clustering()`, `clustering_plot_types()`

Examples

```
set.seed(1)
vars <- c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")
fit <- clustering(iris, vars, n_profiles = 2:4, models = "EEE")
plot_clustering(fit, type = "profiles")
plot_clustering(fit, type = "diagnostics")
plot_clustering(fit, type = c("pairs", "tree"))
```

```
print.grouped_network_analysis
```

Print method for grouped_network_analysis objects

Description

Print method for grouped_network_analysis objects

Usage

```
## S3 method for class 'grouped_network_analysis'
print(x, ...)
```

Arguments

<code>x</code>	A grouped_network_analysis object
<code>...</code>	Additional arguments (ignored)

print.html_table	<i>Print method for html tables</i>
------------------	-------------------------------------

Description

Print method for html tables

Usage

```
## S3 method for class 'html_table'  
print(x, ...)
```

Arguments

x	An html_table object
...	Additional arguments (ignored)

print.latex_table	<i>Print method for latex tables</i>
-------------------	--------------------------------------

Description

Print method for latex tables

Usage

```
## S3 method for class 'latex_table'  
print(x, ...)
```

Arguments

x	A latex_table object
...	Additional arguments (ignored)

print.markdown_table *Print method for markdown tables*

Description

Print method for markdown tables

Usage

```
## S3 method for class 'markdown_table'  
print(x, ...)
```

Arguments

x	A markdown_table object
...	Additional arguments (ignored)

print.mosaic_analysis *Print method for mosaic_analysis objects*

Description

Print method for mosaic_analysis objects

Usage

```
## S3 method for class 'mosaic_analysis'  
print(x, ...)
```

Arguments

x	A mosaic_analysis object
...	Additional arguments (ignored)

```
print.mosaic_stratified
```

Print method for stratified mosaic_analysis objects

Description

Print method for stratified mosaic_analysis objects

Usage

```
## S3 method for class 'mosaic_stratified'  
print(x, ...)
```

Arguments

x	A mosaic_analysis object fitted with by =.
...	Additional arguments (ignored).

Value

Invisibly, x. Called for its side effect of printing.

```
print.network_analysis
```

Print method for network_analysis objects

Description

Print method for network_analysis objects

Usage

```
## S3 method for class 'network_analysis'  
print(x, ...)
```

Arguments

x	A network_analysis object
...	Additional arguments (ignored)

print.saqr_clustering *Print a Clustering Result*

Description

Print a Clustering Result

Usage

```
## S3 method for class 'saqr_clustering'  
print(x, ...)
```

Arguments

x	A 'saqr_clustering' object from [clustering()].
...	Ignored.

Value

'x', invisibly.

print.saqr_plots *Print Several Clustering Plots*

Description

Print Several Clustering Plots

Usage

```
## S3 method for class 'saqr_plots'  
print(x, ...)
```

Arguments

x	A 'saqr_plots' list from [plot_clustering()].
...	Ignored.

Value

'x', invisibly. Called for the side effect of drawing each plot.

print.saqr_result	<i>Print method for saqr_result</i>
-------------------	-------------------------------------

Description

Print method for saqr_result

Usage

```
## S3 method for class 'saqr_result'
print(x, ...)
```

Arguments

x	A saqr_result object
...	Additional arguments (ignored)

replace_missing	<i>Replace Missing Values</i>
-----------------	-------------------------------

Description

Imputes missing values using various methods including mean, median, mode, or custom values.

Usage

```
replace_missing(
  data,
  Vars,
  method = c("mean", "median", "mode", "zero", "min", "max", "value"),
  value = NULL,
  group_by = NULL,
  suffix = NULL
)
```

Arguments

data	A data frame.
Vars	Character vector of variable names to impute.
method	Imputation method: "mean", "median", "mode", "zero", "min", "max", or "value".
value	Custom value to use when method = "value".
group_by	Character. Name of grouping variable for group-wise imputation. Optional.
suffix	Character. Suffix for new columns. If NULL (default), replaces in place.

Value

Data frame with imputed values.

Examples

```
## Not run:
df <- mtcars
df$mpg[c(1, 5, 10)] <- NA

# Replace with mean
df_imputed <- replace_missing(df, Vars = "mpg", method = "mean")

# Replace with median, create new column
df_imputed <- replace_missing(df, Vars = "mpg", method = "median", suffix = "_imp")

# Group-wise mean imputation
df_imputed <- replace_missing(df, Vars = "mpg", method = "mean", group_by = "cyl")

## End(Not run)
```

replace_outliers	<i>Replace Outliers</i>
------------------	-------------------------

Description

Replaces detected outliers with specified values such as NA, mean, median, or winsorized values.
Supports multiple detection methods.

Usage

```
replace_outliers(
  data,
  Vars,
  detect = c("zscore", "iqr", "percentile"),
  threshold = NULL,
  replace_with = c("NA", "mean", "median", "winsorize", "boundary"),
  suffix = NULL
)
```

Arguments

data	A data frame.
Vars	Character vector of numeric variable names to process.
detect	Detection method: "zscore", "iqr", or "percentile".
threshold	Numeric threshold for detection. For zscore: number of SDs (default 3). Use 2 for 2SD, 2.5 for 2.5SD, etc. For iqr: IQR multiplier (default 1.5). For percentile: percentile cutoff (default 0.01 for 1st/99th).

replace_with	Replacement method: "NA", "mean", "median", "winsorize", or "boundary".
suffix	Character. Suffix for new columns. If NULL (default), replaces in place.

Details

Detection methods:

- "zscore": Values beyond threshold SDs from mean. Common thresholds: 2, 2.5, 3, 3.29
- "iqr": Values beyond Q1/Q3 +/- threshold*IQR. Common thresholds: 1.5, 3
- "percentile": Values below or above percentile cutoffs. E.g., 0.01 = 1st/99th, 0.05 = 5th/95th

Replacement methods:

- "NA": Set outliers to NA
- "mean": Replace with mean of non-outliers
- "median": Replace with median of non-outliers
- "winsorize": Cap at threshold boundary
- "boundary": Replace with nearest non-outlier value

Value

Data frame with outliers replaced.

Examples

```
## Not run:
df <- mtcars

# Replace outliers beyond 2 SD with NA
df_clean <- replace_outliers(df, Vars = "hp",
                             detect = "zscore", threshold = 2,
                             replace_with = "NA")

# Winsorize at 3 SD
df_clean <- replace_outliers(df, Vars = "hp",
                             detect = "zscore", threshold = 3,
                             replace_with = "winsorize")

# Replace outliers at 5th/95th percentile with median
df_clean <- replace_outliers(df, Vars = "hp",
                             detect = "percentile", threshold = 0.05,
                             replace_with = "median")

# IQR-based winsorization
df_clean <- replace_outliers(df, Vars = c("mpg", "hp"),
                             detect = "iqr", replace_with = "winsorize")

## End(Not run)
```

reverse_code	<i>Reverse Code Variables</i>
--------------	-------------------------------

Description

Reverse codes numeric variables, commonly used for Likert scales where some items are negatively worded. Uses the formula: $\text{reversed} = (\text{max} + \text{min}) - \text{original}$.

Usage

```
reverse_code(data, Vars, min = NULL, max = NULL, suffix = "_r")
```

Arguments

data	A data frame.
Vars	Character vector of variable names to reverse code.
min	Numeric. Minimum value of the scale. If NULL (default), detected from data.
max	Numeric. Maximum value of the scale. If NULL (default), detected from data.
suffix	Character. Suffix for new column names. Default "_r".

Details

The reverse coding formula is: $\text{reversed} = (\text{max} + \text{min}) - \text{original}$

For a 1-5 Likert scale:

- 1 becomes 5
- 2 becomes 4
- 3 stays 3
- 4 becomes 2
- 5 becomes 1

If min/max are not specified, they are detected from the data. For Likert scales, it's recommended to explicitly specify min and max to ensure correct reversal even if extreme values are not present in the data.

Value

Data frame with reverse coded variables added.

Examples

```
## Not run:
# Create sample Likert data
df <- data.frame(
  item1 = c(1, 2, 3, 4, 5),
  item2 = c(5, 4, 3, 2, 1), # reverse-worded
  item3 = c(2, 3, 4, 3, 2)
)

# Reverse code with automatic detection
df <- reverse_code(df, Vars = "item2")

# Reverse code with explicit scale (1-5 Likert)
df <- reverse_code(df, Vars = "item2", min = 1, max = 5)

# Reverse code multiple items
df <- reverse_code(df, Vars = c("item2", "item3"), min = 1, max = 5)

# 0-10 scale
df <- reverse_code(df, Vars = "item2", min = 0, max = 10)

## End(Not run)
```

reverse_code_vec	<i>Reverse Code a Vector (for use in mutate/across)</i>
------------------	---

Description

Vectorized reverse coding for use with dplyr mutate and across.

Usage

```
reverse_code_vec(x, min = NULL, max = NULL, na.rm = TRUE)
```

Arguments

x	Numeric vector to reverse code.
min	Numeric. Minimum value of the scale. If NULL (default), detected from data.
max	Numeric. Maximum value of the scale. If NULL (default), detected from data.
na.rm	Logical. Remove NA values when detecting min/max? Default TRUE.

Details

The reverse coding formula is: $\text{reversed} = (\text{max} + \text{min}) - \text{original}$

For Likert scales, always specify min and max explicitly to ensure correct reversal even when extreme values are not present in the data.

Value

Reverse coded numeric vector.

Examples

```
## Not run:
library(dplyr)

df <- data.frame(
  item1 = c(1, 2, 3, 4, 5),
  item2 = c(5, 4, 3, 2, 1),
  item3 = c(2, 3, 4, 3, 2)
)

# Reverse single variable (auto-detect scale)
df %>% mutate(item2_r = reverse_code_vec(item2))

# Reverse with explicit 1-5 scale
df %>% mutate(item2_r = reverse_code_vec(item2, min = 1, max = 5))

# Reverse multiple variables
df %>% mutate(across(c(item2, item3), ~reverse_code_vec(.x, min = 1, max = 5), .names = "{.col}_r"))

# 0-10 scale
df %>% mutate(item2_r = reverse_code_vec(item2, min = 0, max = 10))

# 1-7 Likert scale
df %>% mutate(item2_r = reverse_code_vec(item2, min = 1, max = 7))

## End(Not run)
```

saqr_result	Create a saqr_result object
-------------	-----------------------------

Description

Creates a standardized result object for Saqrmisc functions that includes both the original data and a clean markdown representation for AI interpretation.

Usage

```
saqr_result(data = NULL, table = NULL, type = "analysis", markdown = NULL, ...)
```

Arguments

- data The main data (data frame or tibble with results)
- table Optional gt table or formatted table

type	Character describing the analysis type (e.g., "group_comparison", "correlation", "descriptive")
markdown	Optional pre-built markdown string. If NULL, auto-generated from data.
...	Additional components to include in the result

Value

A list with class "saqr_result" containing data, table, markdown, and other components

scale_vars	<i>Scale Variables</i>
------------	------------------------

Description

Scales numeric variables by dividing by standard deviation or rescaling to a specified range. Supports group-wise scaling.

Usage

```
scale_vars(
  data,
  Vars,
  method = c("sd", "range"),
  range = c(0, 1),
  suffix = "_s",
  group_by = NULL
)
```

Arguments

data	A data frame.
Vars	Character vector of numeric variable names to scale.
method	Scaling method: "sd" (divide by SD) or "range" (min-max scaling).
range	Numeric vector of length 2 specifying target range for "range" method. Default c(0, 1). Use c(1, 10) for 1-10 scaling.
suffix	Character. Suffix for new column names. Default "_s".
group_by	Character. Name of grouping variable for group-wise scaling. Optional.

Value

Data frame with scaled variables added.

Examples

```
## Not run:
# Scale by SD
df <- scale_vars(mtcars, Vars = c("mpg", "hp"), method = "sd")

# Min-max scaling (0-1)
df <- scale_vars(mtcars, Vars = "mpg", method = "range")

# Scale to 1-10 range
df <- scale_vars(mtcars, Vars = "mpg", method = "range", range = c(1, 10))

# Group-wise scaling
df <- scale_vars(mtcars, Vars = "mpg", method = "sd", group_by = "cyl")

## End(Not run)
```

scale_vec

Scale a Vector (for use in mutate/across)

Description

Vectorized scaling function for use with dplyr mutate and across. For group-wise scaling, use dplyr::group_by() before mutate().

Usage

```
scale_vec(x, method = c("sd", "range"), range = c(0, 1), na.rm = TRUE)
```

Arguments

x	Numeric vector to scale.
method	Scaling method: "sd" or "range".
range	Target range for "range" method. Default c(0, 1).
na.rm	Logical. Remove NA values? Default TRUE.

Value

Scaled numeric vector.

Examples

```
## Not run:
library(dplyr)

# Scale by SD
mtcars %>% mutate(mpg_s = scale_vec(mpg, method = "sd"))
```

```
# Scale to 1-10
mtcars %>% mutate(mpg_s = scale_vec(mpg, method = "range", range = c(1, 10)))

# Multiple variables to 0-100
mtcars %>% mutate(across(c(mpg, hp), ~scale_vec(.x, method = "range", range = c(0, 100))))

# Group-wise scaling
mtcars %>% group_by(cyl) %>% mutate(mpg_s = scale_vec(mpg, method = "sd"))

## End(Not run)
```

set_api_key

Set API Key and Options for Session

Description

Set your API key and optional default settings for the current R session. Settings are stored as R options and used as defaults by `'pass()'`.

Usage

```
set_api_key(
  key,
  provider = c("openai", "anthropic", "gemini", "openrouter"),
  model = NULL,
  ...
)
```

Arguments

key	Your API key
provider	Provider name: "openai" (default), "anthropic", "gemini", or "openrouter"
model	Optional. Default model to use for this provider.
...	Additional options to store (e.g., <code>'base_url'</code> , <code>'timeout'</code>).

Value

Invisibly returns TRUE.

Examples

```
## Not run:
# Just set the key
set_api_key("sk-...", "openai")

# Set key and default model
set_api_key("sk-...", "openai", model = "gpt-4o")
```

```
# Set key with custom base URL (for Azure OpenAI, etc.)
set_api_key("sk-...", "openai", model = "gpt-4", base_url = "https://my-azure.openai.azure.com")

## End(Not run)
```

set_claude_key	<i>Set Anthropic (Claude) API Key</i>
----------------	---------------------------------------

Description

Convenience alias for ‘set_api_key(key, "anthropic", ...)’.

Usage

```
set_claude_key(key, model = NULL, ...)
```

Arguments

- key Your Anthropic API key
- model Optional. Default model (e.g., "claude-sonnet-4-20250514").
- ... Additional options.

Examples

```
## Not run:
set_claude_key("sk-ant-...")
set_claude_key("sk-ant-...", model = "claude-sonnet-4-20250514")

## End(Not run)
```

set_gemini_key	<i>Set Google Gemini API Key</i>
----------------	----------------------------------

Description

Convenience alias for ‘set_api_key(key, "gemini", ...)’.

Usage

```
set_gemini_key(key, model = NULL, ...)
```

Arguments

key	Your Gemini API key
model	Optional. Default model (e.g., "gemini-2.5-flash").
...	Additional options.

Examples

```
## Not run:
set_gemini_key("AIza...")
set_gemini_key("AIza...", model = "gemini-2.5-pro")

## End(Not run)
```

set_openai_key	<i>Set OpenAI API Key</i>
----------------	---------------------------

Description

Convenience alias for ‘set_api_key(key, "openai", ...)’.

Usage

```
set_openai_key(key, model = NULL, ...)
```

Arguments

key	Your OpenAI API key
model	Optional. Default model (e.g., "gpt-4o", "gpt-4.1-nano").
...	Additional options (e.g., ‘base_url’ for Azure OpenAI).

Examples

```
## Not run:
set_openai_key("sk-...")
set_openai_key("sk-...", model = "gpt-4o")
set_openai_key("sk-...", model = "gpt-4", base_url = "https://my-azure.openai.azure.com")

## End(Not run)
```

set_openrouter_key	<i>Set OpenRouter API Key</i>
--------------------	-------------------------------

Description

Convenience alias for ‘set_api_key(key, "openrouter", ...)’.

Usage

```
set_openrouter_key(key, model = NULL, ...)
```

Arguments

key	Your OpenRouter API key
model	Optional. Default model (e.g., "anthropic/claude-sonnet-4", "openai/gpt-4o").
...	Additional options.

Examples

```
## Not run:
set_openrouter_key("sk-or-...")
set_openrouter_key("sk-or-...", model = "openai/gpt-4o")

## End(Not run)
```

standardize	<i>Standardize Variables (Z-Score)</i>
-------------	--

Description

Standardizes numeric variables to z-scores (mean = 0, SD = 1). Supports group-wise standardization for multilevel/nested data.

Usage

```
standardize(data, Vars, suffix = "_z", group_by = NULL)
```

Arguments

data	A data frame.
Vars	Character vector of numeric variable names to standardize.
suffix	Character. Suffix for new column names. Default "_z".
group_by	Character. Name of grouping variable for group-wise standardization. Optional.

Value

Data frame with standardized variables added.

Examples

```
## Not run:
# Simple standardization
df <- standardize(mtcars, Vars = c("mpg", "hp"))

# Group-wise standardization
df <- standardize(mtcars, Vars = "mpg", group_by = "cyl")

## End(Not run)
```

standardize_vec

Standardize a Vector (for use in mutate/across)

Description

Vectorized standardization function for use with dplyr mutate and across. For group-wise standardization, use `dplyr::group_by()` before `mutate()`.

Usage

```
standardize_vec(x, na.rm = TRUE)
```

Arguments

x	Numeric vector to standardize.
na.rm	Logical. Remove NA values? Default TRUE.

Value

Standardized numeric vector (mean = 0, SD = 1).

Examples

```
## Not run:
library(dplyr)

# Single variable
mtcars %>% mutate(mpg_z = standardize_vec(mpg))

# Multiple variables
mtcars %>% mutate(across(c(mpg, hp, wt), standardize_vec, .names = "{.col}_z"))

# Group-wise standardization
mtcars %>% group_by(cyl) %>% mutate(mpg_z = standardize_vec(mpg))
```

```
## End(Not run)
```

```
summary.mosaic_analysis
```

Summary method for mosaic_analysis objects

Description

Summary method for mosaic_analysis objects

Usage

```
## S3 method for class 'mosaic_analysis'
summary(object, ...)
```

Arguments

object	A mosaic_analysis object
...	Additional arguments (ignored)

```
summary.mosaic_stratified
```

Summary method for stratified mosaic_analysis objects

Description

Summary method for stratified mosaic_analysis objects

Usage

```
## S3 method for class 'mosaic_stratified'
summary(object, ...)
```

Arguments

object	A mosaic_analysis object fitted with by =.
...	Additional arguments (ignored).

Value

Invisibly, the per-stratum summary data.frame.

`summary.network_analysis`*Summary method for network_analysis objects*

Description

Summary method for network_analysis objects

Usage

```
## S3 method for class 'network_analysis'  
summary(object, ...)
```

Arguments

<code>object</code>	A network_analysis object
<code>...</code>	Additional arguments (ignored)

`summary.sqr_clustering`*Compare the Candidate Models of a Clustering Result*

Description

Compare the Candidate Models of a Clustering Result

Usage

```
## S3 method for class 'sqr_clustering'  
summary(object, ...)
```

Arguments

<code>object</code>	A 'sqr_clustering' object from [clustering()].
<code>...</code>	Ignored.

Value

A data.frame with one row per candidate (profiles x covariance model): 'n_profiles', 'model', 'log_likelihood', 'n_parameters', 'aic', 'bic', 'icl', 'entropy' (relative), 'converged', 'boundary', 'delta_bic' (distance from the selected model) and 'selected'.

to_dataframe	<i>Convert Any Table to Data Frame</i>
--------------	--

Description

Extracts the underlying data frame from various table objects including gt tables, kable objects, and matrices.

Usage

```
to_dataframe(x, digits = NULL)
```

Arguments

x	A table object (gt, kable, matrix, or data frame).
digits	Number of decimal places for formatting numeric columns. Set to NULL to skip formatting. Default NULL.

Value

A data frame.

Examples

```
## Not run:
# Extract data from a gt table
library(gt)
gt_table <- gt(mtcars[1:5, 1:3])
df <- to_dataframe(gt_table)

# Works with matrices too
mat <- matrix(1:6, nrow = 2)
to_dataframe(mat)

## End(Not run)
```

to_gt	<i>Convert Table to GT Format</i>
-------	-----------------------------------

Description

Converts a data frame or matrix to a gt table object for publication-ready output. Provides a consistent interface matching other to_* functions.

Usage

```
to_gt(
  x,
  title = NULL,
  subtitle = NULL,
  digits = 2,
  rowname_col = NULL,
  theme = c("default", "fancy", "minimal", "dark", "none")
)
```

Arguments

x	A data frame or matrix to convert.
title	Optional title for the table.
subtitle	Optional subtitle for the table.
digits	Number of decimal places for numeric columns. Default 2.
rowname_col	Optional column name to use as row names.
theme	Character. Visual theme: "default" (clean scientific style), "fancy" (blue headers), "minimal" (simple borders), "dark" (dark mode), or "none" (no styling). Default "default".

Value

A gt table object.

Examples

```
## Not run:
# Convert a data frame to gt
df <- data.frame(
  Variable = c("Age", "Score"),
  Mean = c(35.2, 78.5),
  SD = c(10.1, 12.3)
)
to_gt(df)
to_gt(df, title = "Descriptive Statistics", theme = "fancy")

# With row names
to_gt(mtcars[1:5, 1:3], rowname_col = NULL)

## End(Not run)
```

to_html

*Convert Table to HTML Format***Description**

Converts a data frame or gt table to HTML table format. Works with any tabular data structure.

Usage

```
to_html(
  x,
  title = NULL,
  digits = 2,
  class = "table",
  id = NULL,
  border = 1,
  style = NULL,
  header_style = "font-weight: bold;",
  stripe = FALSE
)
```

Arguments

x	A data frame, matrix, or gt table to convert.
title	Optional title (added as caption).
digits	Number of decimal places for numeric columns. Default 2.
class	CSS class(es) for the table element. Default "table".
id	Optional HTML id attribute for the table.
border	Table border width. Default 1.
style	Additional inline CSS style for the table.
header_style	CSS style for header cells. Default bold.
stripe	Logical. Add striped rows? Default FALSE.

Value

A character string containing the HTML code with class "html_table" for proper printing.

Examples

```
## Not run:
# Convert a data frame to HTML
df <- data.frame(
  Name = c("Alice", "Bob"),
  Score = c(85, 92)
)
```

```
to_html(df)
to_html(df, title = "Scores", class = "table table-striped")

## End(Not run)
```

to_kable	<i>Convert Table to Kable Format</i>
----------	--------------------------------------

Description

Converts a data frame or gt table to knitr::kable format for use in R Markdown documents.

Usage

```
to_kable(
  x,
  format = c("pipe", "markdown", "html", "latex", "rst"),
  digits = 2,
  caption = NULL,
  col.names = NULL,
  row.names = FALSE,
  align = NULL,
  ...
)
```

Arguments

x	A data frame, matrix, or gt table to convert.
format	Output format: "markdown", "html", "latex", "rst", "pipe". Default "pipe" for GitHub-flavored markdown.
digits	Number of decimal places for numeric columns. Default 2.
caption	Optional table caption.
col.names	Optional column names (overrides data frame names).
row.names	Logical. Include row names? Default FALSE.
align	Column alignment: "l" (left), "c" (center), "r" (right), or a vector of alignments.
...	Additional arguments passed to knitr::kable.

Value

A knitr_kable object.

Examples

```
## Not run:
df <- data.frame(A = 1:3, B = c("x", "y", "z"))
to_kable(df)
to_kable(df, format = "html", caption = "My Table")

## End(Not run)
```

to_latex

Convert Table to LaTeX Format

Description

Converts a data frame or gt table to LaTeX tabular format. Works with any tabular data structure.

Usage

```
to_latex(
  x,
  title = NULL,
  label = NULL,
  digits = 2,
  align = "l",
  booktabs = TRUE,
  float = NULL,
  position = "htbp"
)
```

Arguments

x	A data frame, matrix, or gt table to convert.
title	Optional title (added as caption).
label	Optional LaTeX label for cross-referencing.
digits	Number of decimal places for numeric columns. Default 2.
align	Character vector of column alignments ("l", "c", "r") or single character to apply to all columns. Default "l" (left).
booktabs	Logical. Use booktabs style (toprule, midrule, bottomrule)? Default TRUE for cleaner tables.
float	Logical. Wrap in table environment with caption? Default TRUE if title is provided.
position	Table float position (e.g., "htbp"). Default "htbp".

Value

A character string containing the LaTeX code with class "latex_table" for proper printing.

Examples

```
## Not run:
# Convert a data frame to LaTeX
df <- data.frame(
  Variable = c("Age", "Score", "Income"),
  Mean = c(35.2, 78.5, 52000),
  SD = c(10.1, 12.3, 15000)
)
to_latex(df)
to_latex(df, title = "Descriptive Statistics", booktabs = TRUE)

# With custom alignment
to_latex(df, align = c("l", "r", "r"))

## End(Not run)
```

to_markdown

Convert Table to Markdown Format

Description

Converts a data frame or gt table to markdown format. Works with any tabular data structure.

Usage

```
to_markdown(
  x,
  title = NULL,
  subtitle = NULL,
  digits = 2,
  align = "l",
  caption_style = c("header", "bold", "plain")
)
```

Arguments

<code>x</code>	A data frame, matrix, or gt table to convert.
<code>title</code>	Optional title for the table.
<code>subtitle</code>	Optional subtitle for the table.
<code>digits</code>	Number of decimal places for numeric columns. Default 2.
<code>align</code>	Character vector of column alignments ("l", "c", "r") or single character to apply to all columns. Default "l" (left).
<code>caption_style</code>	Style for title: "header" (## Title), "bold" (**Title**), or "plain". Default "header".

Value

A character string containing the markdown table with class "markdown_table" for proper printing.

Examples

```
## Not run:
# Convert a data frame to markdown
df <- data.frame(
  Name = c("Alice", "Bob", "Carol"),
  Score = c(85.5, 92.3, 78.9),
  Grade = c("B", "A", "C")
)
to_markdown(df)
to_markdown(df, title = "Student Scores", digits = 1)

# Works with gt tables too
library(gt)
gt_table <- gt(df)
to_markdown(gt_table)

## End(Not run)
```

winsorize_vec

Winsorize a Vector (vectorized for mutate/across)

Description

Vectorized winsorization for use with dplyr. For group-wise winsorization, use `dplyr::group_by()` before `mutate()`.

Usage

```
winsorize_vec(
  x,
  method = c("zscore", "iqr", "percentile"),
  threshold = NULL,
  na.rm = TRUE
)
```

Arguments

<code>x</code>	Numeric vector.
<code>method</code>	Detection method: "zscore", "iqr", or "percentile".
<code>threshold</code>	Threshold value.
<code>na.rm</code>	Logical. Remove NA values? Default TRUE.

Value

Winsorized numeric vector.

Examples

```
## Not run:
library(dplyr)

# Winsorize at 3 SD
mtcars %>% mutate(hp_w = winsorize_vec(hp, method = "zscore", threshold = 3))

# Winsorize at 5th/95th percentile
mtcars %>% mutate(hp_w = winsorize_vec(hp, method = "percentile", threshold = 0.05))

# Group-wise winsorization
mtcars %>% group_by(cyl) %>% mutate(hp_w = winsorize_vec(hp))

## End(Not run)
```

Index

add_total, 3
as.data.frame, 44
as.data.frame.mosaic_analysis, 4
auto_describe, 5

categorical_table, 7
center, 10
center_vec, 11
clustering, 12
clustering(), 51, 52
clustering_plot_types, 13
clustering_plot_types(), 51, 52
compare_groups, 14
compare_networks, 21
convert_sequences, 22
correlation_matrix, 24
correlations, 26
counts, 29

data_overview, 30
descriptive_table, 9, 31

estimate_grouped_networks, 34
estimate_single_network, 36

fitted.saqr_clustering, 37

ggbetweenstats, 17
gt, 33

is_outlier, 38

latents::enumerate_lpa(), 12, 13
launch_app, 39

message(), 13
missing_analysis, 40
mosaic_analysis, 9, 39, 41, 50

normality_check, 45
outlier_check, 46

p.adjust, 43
pass, 8, 17, 25, 27, 32, 43, 47
plot.mosaic_analysis, 50
plot.saqr_clustering, 51
plot_clustering, 51
plot_clustering(), 13
print.grouped_network_analysis, 52
print.html_table, 53
print.latex_table, 53
print.markdown_table, 54
print.mosaic_analysis, 54
print.mosaic_stratified, 55
print.network_analysis, 55
print.saqr_clustering, 56
print.saqr_plots, 56
print.saqr_result, 57

replace_missing, 57
replace_outliers, 58
reverse_code, 60
reverse_code_vec, 61
runApp, 39

saqr_result, 62
scale_vars, 63
scale_vec, 64
set_api_key, 8, 17, 25, 27, 32, 43, 65
set_claude_key, 66
set_gemini_key, 66
set_openai_key, 67
set_openrouter_key, 68
standardize, 68
standardize_vec, 69
summary.mosaic_analysis, 70
summary.mosaic_stratified, 70
summary.network_analysis, 71
summary.saqr_clustering, 71

to_dataframe, 72
to_gt, 72

to_html, [74](#)

to_kable, [75](#)

to_latex, [76](#)

to_markdown, [77](#)

winsorize_vec, [78](#)