Package ‘SurveyStat’

January 22, 2026

Type Package

Title Survey Data Cleaning, Weighting and Analysis

Version 1.0.3

Description Provides utilities for cleaning survey data, computing weights, and performing descriptive statistical analysis. Methods follow Lohr (2019, ISBN:978-0367272454) ``Sampling: Design and Analysis'' and Lumley (2010) <doi:10.1002/9780470580066>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports dplyr, ggplot2, rlang, stats

Suggests knitr, rmarkdown, markdown, testthat

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Muhammad Ali [aut, cre]

Maintainer Muhammad Ali <aliawan1170@gmail.com>

Repository CRAN

Date/Publication 2026-01-22 21:20:02 UTC

Contents

`apply_weights` *Apply survey weights to data*

Description

This function applies survey weights by creating a weighted version of the dataset. The weights are normalized to sum to the sample size for computational stability.

Usage

```
apply_weights(data, weight_col)
```

Arguments

`data`	A data.frame containing survey data
`weight_col`	Character string specifying column name containing weights

Value

A data.frame with normalized weights

Examples

```
data <- data.frame(age = c(25, 30, 35), weight = c(1.2, 0.8, 1.0))
weighted_data <- apply_weights(data, "weight")
```

`clean_missing` *Clean missing values in specified column*

Description

This function handles missing values using specified imputation method. Supports mean, median, and mode imputation for numeric variables.

Usage

```
clean_missing(data, col, method = c("mean", "median", "mode"))
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name to clean
`method`	Character string specifying imputation method ("mean", "median", or "mode")

Value

A data.frame with missing values imputed

Examples

```
data <- data.frame(age = c(25, NA, 30, NA, 35))
clean_data <- clean_missing(data, "age", method = "mean")
```

`cross_tabulation` *Generate cross-tabulation table with chi-square test*

Description

This function creates a cross-tabulation between two categorical variables and performs a chi-square test of independence. Can incorporate survey weights.

Usage

```
cross_tabulation(data, col1, col2, weight_col = NULL)
```

Arguments

`data`	A data.frame containing survey data
`col1`	Character string specifying first categorical variable
`col2`	Character string specifying second categorical variable
`weight_col`	Character string specifying column name containing weights (optional)

Value

A list containing cross-tabulation and chi-square test results

Examples

```
data <- data.frame(gender = c("M", "F", "M", "F"),
                   education = c("HS", "College", "HS", "College"))
cross_tab <- cross_tabulation(data, "gender", "education")
```

describe_survey *Generate comprehensive survey description*

Description

This function provides a comprehensive description of survey data including sample size, variable types, missing value patterns, and basic statistics. Can incorporate survey weights if provided.

Usage

```
describe_survey(data, weight_col = NULL)
```

Arguments

`data`	A data.frame containing survey data
`weight_col`	Character string specifying column name containing weights (optional)

Value

A list containing descriptive statistics

Examples

```
data <- data.frame(
  age = c(25, 30, 35),
  gender = c("M", "F", "M"),
  weight = c(1.2, 0.8, 1.0)
)
desc <- describe_survey(data)
desc_weighted <- describe_survey(data, "weight")
```

example_survey *Example Survey Dataset*

Description

A small example dataset used to demonstrate SurveyStat functions.

Usage

```
example_survey
```

Format

A data frame with 10 rows and 5 variables:

Age Numeric age of respondent

Gender Gender of respondent (Male/Female)

Education Education level (High School/Bachelor/Graduate)

Income Numeric income value

Weight Survey weight

Source

Simulated data for demonstration purposes

`frequency_table` *Generate frequency table for categorical variable*

Description

This function creates a frequency table for a categorical variable, optionally incorporating survey weights.

Usage

```
frequency_table(data, col, weight_col = NULL)
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name for categorical variable
`weight_col`	Character string specifying column name containing weights (optional)

Value

A data.frame with frequency statistics

Examples

```
data <- data.frame(gender = c("M", "F", "M", "F"), weight = c(1, 1.2, 0.8, 1.1))
freq_table <- frequency_table(data, "gender")
weighted_freq <- frequency_table(data, "gender", "weight")
```

`plot_boxplot` *Create publication-quality box plot*

Description

This function creates a clean, publication-quality box plot for numeric variables, optionally grouped by a categorical variable.

Usage

```
plot_boxplot(data, col, group_col = NULL, add_points = TRUE)
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name for numeric variable
`group_col`	Character string specifying column name for grouping variable (optional)
`add_points`	Logical whether to add individual data points (default: TRUE)

Value

A ggplot object

Examples

```
data <- data.frame(age = c(25, 30, 35, 40, 45), gender = c("M", "F", "M", "F", "M"))
box_plot <- plot_boxplot(data, "age")
grouped_box <- plot_boxplot(data, "age", "gender")
```

`plot_histogram` *Create publication-quality histogram*

Description

This function creates a clean, publication-quality histogram for numeric variables using ggplot2 with minimal theme and appropriate statistical overlays.

Usage

```
plot_histogram(data, col, bins = 30, add_density = TRUE)
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name for numeric variable
`bins`	Number of bins for histogram (default: 30)
`add_density`	Logical whether to add density curve (default: TRUE)

Value

A ggplot object

Examples

```
data <- data.frame(age = rnorm(100, 35, 10))
hist_plot <- plot_histogram(data, "age")
print(hist_plot)
```

`plot_weighted_bar` *Create weighted bar plot for categorical variables*

Description

This function creates a bar plot for categorical variables, optionally using survey weights to show weighted frequencies.

Usage

```
plot_weighted_bar(data, col, weight_col = NULL, show_percentages = TRUE)
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name for categorical variable
`weight_col`	Character string specifying column name containing weights (optional)
`show_percentages`	Logical whether to show percentage labels (default: TRUE)

Value

A ggplot object

Examples

```
data <- data.frame(gender = c("M", "F", "M", "F"), weight = c(1, 1.2, 0.8, 1.1))
bar_plot <- plot_weighted_bar(data, "gender")
weighted_bar <- plot_weighted_bar(data, "gender", "weight")
```

`rake_weights` *Rake survey weights to match population targets*

Description

This function implements simple raking (iterative proportional fitting) to adjust survey weights to match known population marginal totals. Assumes two-dimensional raking for simplicity.

Usage

```
rake_weights(data, population_targets, weight_col = "weight")
```

Arguments

`data`	A data.frame containing survey data
`population_targets`	Named list with population totals for each variable
`weight_col`	Character string specifying initial weight column name

Value

A data.frame with raked weights

Examples

```
# Assuming we have gender and education population totals
targets <- list(
  gender = c(Male = 1000000, Female = 1050000),
  education = c(HighSchool = 800000, Bachelor = 900000, Graduate = 350000)
)
data <- data.frame(
  gender = c("Male", "Female", "Male", "Female", "Male"),
  education = c("HighSchool", "Bachelor", "Bachelor", "HighSchool", "Graduate"),
  weight = c(1, 1, 1, 1, 1)
)
raked_data <- rake_weights(data, targets, "weight")
```

`remove_duplicates` *Remove duplicate rows from survey data*

Description

This function identifies and removes duplicate rows based on all columns. Preserves the first occurrence of each duplicate.

Usage

```
remove_duplicates(data)
```

Arguments

`data`	A data.frame containing survey data

Value

A data.frame with duplicates removed

Examples

```
data <- data.frame(id = c(1, 2, 2, 3), age = c(25, 30, 30, 35))
clean_data <- remove_duplicates(data)
```

`standardize_categories` *Standardize categorical values*

Description

This function standardizes categorical variables by mapping values to standardized categories. Useful for consolidating different representations of the same category.

Usage

```
standardize_categories(data, col, mapping)
```

Arguments

`data`	A data.frame containing survey data
`col`	Character string specifying column name to standardize
`mapping`	Named list or vector mapping old values to new values

Value

A data.frame with standardized categories

Examples

```
data <- data.frame(gender = c("M", "Male", "F", "Female", "m"))
mapping <- list("M" = "Male", "Male" = "Male", "F" = "Female", "Female" = "Female", "m" = "Male")
clean_data <- standardize_categories(data, "gender", mapping)
```

`weighted_mean` *Calculate weighted mean*

Description

This function calculates the weighted mean of a numeric variable. Uses standard weighted mean formula: sum(x * w) / sum(w)

Usage

```
weighted_mean(data, target_col, weight_col)
```

Arguments

`data`	A data.frame containing survey data
`target_col`	Character string specifying column name for target variable
`weight_col`	Character string specifying column name containing weights

Value

Numeric weighted mean

Examples

```
data <- data.frame(income = c(50000, 75000, 100000), weight = c(1.2, 0.8, 1.0))
weighted_income <- weighted_mean(data, "income", "weight")
```

`weighted_total` *Calculate weighted total*

Description

This function calculates the weighted total of a numeric variable. Useful for estimating population totals from survey data.

Usage

```
weighted_total(data, target_col, weight_col)
```

Arguments

`data`	A data.frame containing survey data
`target_col`	Character string specifying column name for target variable
`weight_col`	Character string specifying column name containing weights

Value

Numeric weighted total

Examples

```
data <- data.frame(income = c(50000, 75000, 100000), weight = c(1000, 800, 1200))
total_income <- weighted_total(data, "income", "weight")
```

Index