Package: statelier (via r-universe)

November 24, 2026

Title Confidence Intervals and Hypothesis Tests with a Consistent Layout

Version 0.2.0

Description Computes confidence intervals and hypothesis tests for means and proportions. Every function prints its result in the same consistent three-section layout -- sample information, intermediary calculations, and the final answer -- so students always know where to look. Intended for classroom teaching, where each step of a calculation is shown, while every reported quantity is also returned in a standard R object for further use.

License MIT + file LICENSE

Encoding UTF-8

Config/testthat/edition 3

Suggests testthat (>= 3.0.0)

NeedsCompilation no

Config/roxygen2/version 8.1.1

Author Doru Cojoc [aut, cre]

Maintainer Doru Cojoc <dc3035@columbia.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-24 12:50:02 UTC

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

RemoteRef HEAD

RemoteSha f168e59753aeb5efbf1da2dc3c9dfab52cfe69a1

Contents

ci_mean_stats 2
ci_prop_stats 4
ht_mean_stats 6
ht_prop_stats 9

print.statelier_ci 12
print.statelier_ht 12

Index **13**

`ci_mean_stats` *Confidence interval for a mean or a difference in means*

Description

Computes a confidence interval for a single population mean or for the difference between two population means, and prints the result in the package's consistent three-section layout: sample information, intermediary calculations, and the final answer – the same layout used by every function in the package.

Usage

```
ci_mean_stats(
  data1,
  data2 = NULL,
  conf_level = 0.95,
  diff_direction = c("2-1", "1-2"),
  var_equal = FALSE,
  data = NULL,
  first = NULL
)

ci_mean(
  data1,
  data2 = NULL,
  conf_level = 0.95,
  diff_direction = c("2-1", "1-2"),
  var_equal = FALSE,
  data = NULL,
  first = NULL
)
```

Arguments

`data1`	A numeric vector with the sample data, or a formula of the form `outcome ~ group` together with `data`. Missing values in a vector are dropped and counted.
`data2`	An optional second numeric vector. If `NULL` (the default), a one-sample interval for the population mean is computed; otherwise a two-sample interval for the difference in population means is computed. Must not be supplied together with a formula.
`conf_level`	Confidence level, a single number strictly between 0 and 1. Defaults to 0.95.

`diff_direction` Direction of the estimated difference in the two-sample case: `"2-1"` (the default) estimates mean2 - mean1, `"1-2"` estimates mean1 - mean2. Ignored for one-sample intervals.

`var_equal` Should the two population variances be treated as equal? If `TRUE`, a pooled-variance (Student) interval is computed; if `FALSE` (the default), Welch's unequal-variance interval is computed. Ignored for one-sample intervals.

`data` A data frame, used only together with a formula in `data1` (e.g. `ci_mean_stats(weight ~ sex, data = df)`). With vectors, leave it `NULL` and subset the data yourself.

`first` What goes first in the difference (the difference is mean(first) - mean(other)). With a formula, `first` is the group value exactly as it appears in the data: `first = "male"`, `first = 1`, or a factor level's name as a string. With two vectors, `first` is a number, 1 or 2: which vector, counted in the order you typed them, goes first. See Details for the defaults when `first` is not supplied.

Details

`ci_mean()` is a short alias for `ci_mean_stats()`; both names are exported.

There are two ways to give the samples. With **vectors**, `ci_mean_stats(x)` is a one-sample interval and `ci_mean_stats(x, y)` a two-sample interval. With a **data frame**, `ci_mean_stats(y ~ group, data = df)` splits the column `y` by the two values of the column `group` – the tilde reads as "by", so this is "the mean of y, by group". Exactly one of these patterns may be used; mixing them (a formula and a second vector, or `data` without a formula) is an error. The grouping variable must have exactly two groups; rows with a missing outcome or group are dropped and counted.

The **direction** of the difference – which mean is subtracted from which – follows one rule: the difference is mean(first) - mean(other), where `first` is chosen by the `first` argument. When `first` is not supplied, the defaults are: with two vectors, the second vector minus the first (treatment - control); with a formula and numeric groups, the larger code minus the smaller (1 - 0, 2 - 1); with a factor, the factor level order; with character groups, alphabetical order. The printed output always states the direction with the actual group labels (e.g. "Difference in Means: male - female"), so it never has to be guessed. (`diff_direction` is the older way to set the direction for vector calls and is still honoured; do not supply both it and `first`.)

The one-sample interval is the usual mean +/- t * s/sqrt(n) interval. The two-sample interval uses either Welch's approximation (`var_equal = FALSE`) or the pooled-variance formula (`var_equal = TRUE`).

Value

An object of class `"statelier_ci"` (a list) returned invisibly when called interactively through its print method. The list carries every quantity shown in the printed output (sample statistics, standard error, degrees of freedom, interval limits, ...) so it can be used in further calculations. Assigning the result to a variable suppresses printing.

See Also

`ht_mean_stats()` for the corresponding hypothesis test, `ci_prop_stats()` for proportions.

Examples

```
set.seed(1)
x <- rnorm(30, mean = 100, sd = 15)
y <- rnorm(40, mean = 105, sd = 15)

# One-sample interval
ci_mean_stats(x)

# Two-sample interval, Welch
ci_mean_stats(x, y)

# Two-sample interval, equal variances
ci_mean_stats(x, y, var_equal = TRUE)

# Capture the numbers without printing
res <- ci_mean_stats(x, y)
res$ci_lower
res$ci_upper

# The same two-sample interval from a data frame:
# "weight, by sex" (default direction: female - male, alphabetical)
df <- data.frame(
  weight = c(70, 82, 65, 90, 77, 68, 74, 88, 61, 79),
  sex    = c("male", "male", "female", "male", "female",
             "female", "male", "male", "female", "female")
)
ci_mean_stats(weight ~ sex, data = df)
ci_mean_stats(weight ~ sex, data = df, first = "male")
```

`ci_prop_stats` *Confidence interval for a proportion or a difference in proportions*

Description

Computes a Wald confidence interval for a single population proportion or for the difference between two population proportions, and prints the result in the package's consistent three-section layout: sample information, intermediary calculations, and the final answer – the same layout used by every function in the package.

Usage

```
ci_prop_stats(
  data1,
  data2 = NULL,
  conf_level = 0.95,
  diff_direction = c("2-1", "1-2"),
  data = NULL,
  first = NULL
```

```
)

ci_prop(
  data1,
  data2 = NULL,
  conf_level = 0.95,
  diff_direction = c("2-1", "1-2"),
  data = NULL,
  first = NULL
)
```

Arguments

`data1`	A numeric vector of 0s and 1s with the sample data (logical vectors are coerced), or a formula of the form `outcome ~ group` together with `data`, where the outcome column holds 0s and 1s (or a logical column). Missing values are dropped and counted.
`data2`	An optional second numeric vector of 0s and 1s. If `NULL` (the default), a one-sample interval for the population proportion is computed; otherwise a two-sample interval for the difference in population proportions is computed. Must not be supplied together with a formula.
`conf_level`	Confidence level, a single number strictly between 0 and 1. Defaults to 0.95.
`diff_direction`	Direction of the estimated difference in the two-sample case: `"2-1"` (the default) estimates prop2 - prop1, `"1-2"` estimates prop1 - prop2. Ignored for one-sample intervals. Not used with a formula; use `first` there instead.
`data`	A data frame, used only together with a formula in `data1` (e.g. `ci_prop_stats(passed ~ section, data = df)`). With vectors, leave it `NULL` and subset the data yourself.
`first`	What goes first in the difference (the difference is prop(first) - prop(other)). With a formula, `first` is the group value exactly as it appears in the data: `first = "morning"`, `first = 1`, or a factor level's name as a string. With two vectors, `first` is a number, 1 or 2: which vector, counted in the order you typed them, goes first. See Details for the defaults when `first` is not supplied.

Details

`ci_prop()` is a short alias for `ci_prop_stats()`; both names are exported.

There are two ways to give the samples. With **vectors**, `ci_prop_stats(x)` is a one-sample interval and `ci_prop_stats(x, y)` a two-sample interval. With a **data frame**, `ci_prop_stats(y ~ group, data = df)` splits the 0/1 column `y` by the two values of the column `group` – the tilde reads as "by", so this is "the proportion of y, by group". Exactly one of these patterns may be used; mixing them (a formula and a second vector, or `data` without a formula) is an error. The grouping variable must have exactly two groups; rows with a missing outcome or group are dropped and counted.

The **direction** of the difference – which proportion is subtracted from which – follows one rule: the difference is prop(first) - prop(other), where `first` is chosen by the `first` argument. When `first`

is not supplied, the defaults are: with two vectors, the second vector minus the first (treatment - control); with a formula and numeric groups, the larger code minus the smaller (1 - 0, 2 - 1); with a factor, the factor level order; with character groups, alphabetical order. The printed output always states the direction with the actual group labels, so it never has to be guessed. (`diff_direction` is the older way to set the direction for vector calls and is still honoured; do not supply both it and `first`.)

Both intervals use the large-sample normal (Wald) approximation. The one-sample interval is truncated to $[0, 1]$. A warning is issued when the normal approximation looks unreliable (fewer than 5 expected successes or failures).

Value

An object of class `"statelier_ci"` (a list) with every quantity shown in the printed output. Assigning the result to a variable suppresses printing.

See Also

`ht_prop_stats()` for the corresponding hypothesis test, `ci_mean_stats()` for means.

Examples

```
set.seed(2)
x <- rbinom(200, 1, 0.35)
y <- rbinom(250, 1, 0.45)

# One-sample interval
ci_prop_stats(x)

# Two-sample interval for prop2 - prop1
ci_prop_stats(x, y)

# The same two-sample interval from a data frame:
# "passed, by section" (default direction is alphabetical)
df <- data.frame(
  passed  = c(1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
              1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1),
  section = rep(c("morning", "evening"), each = 20)
)
ci_prop_stats(passed ~ section, data = df)
ci_prop_stats(passed ~ section, data = df, first = "morning")
```

`ht_mean_stats` *Hypothesis test for a mean or a difference in means*

Description

Tests a hypothesis about a single population mean or about the difference between two population means, and prints the result in the package's consistent three-section layout: sample information, intermediary calculations, and the final answer – the same layout used by every function in the package. (The reject / fail-to-reject conclusion is shown only when `alpha` is supplied.)

Usage

```
ht_mean_stats(
  data1,
  data2 = NULL,
  null_value = 0,
  alpha = NULL,
  alternative = c("two.sided", "less", "greater"),
  diff_direction = c("2-1", "1-2"),
  var_equal = FALSE,
  data = NULL,
  first = NULL
)

ht_mean(
  data1,
  data2 = NULL,
  null_value = 0,
  alpha = NULL,
  alternative = c("two.sided", "less", "greater"),
  diff_direction = c("2-1", "1-2"),
  var_equal = FALSE,
  data = NULL,
  first = NULL
)
```

Arguments

`data1`	A numeric vector with the sample data, or a formula of the form `outcome ~ group` together with `data`. Missing values in a vector are dropped and counted.
`data2`	An optional second numeric vector. If `NULL` (the default), a one-sample t test is performed; otherwise a two-sample t test is performed. Must not be supplied together with a formula.
`null_value`	The hypothesized value under the null hypothesis: the population mean for a one-sample test, or the difference in population means (in the direction given by `diff_direction`) for a two-sample test. Defaults to 0.
`alpha`	Significance level, a single number strictly between 0 and 1. If `NULL` (the default), no significance level is assumed: the significance level and the reject / fail-to-reject conclusion are omitted from the printed output, so the student must compare the p-value to a significance level of their own choosing.
`alternative`	The alternative hypothesis: `"two.sided"` (the default), `"less"`, or `"greater"`.

`diff_direction` Direction of the estimated difference in the two-sample case: `"2-1"` (the default) estimates mean2 - mean1, `"1-2"` estimates mean1 - mean2. `null_value` refers to the difference in this direction. Ignored for one-sample tests.

`var_equal` Should the two population variances be treated as equal? If `TRUE`, a pooled-variance (Student) test is performed; if `FALSE` (the default), Welch's unequal-variance test is performed. Ignored for one-sample tests.

`data` A data frame, used only together with a formula in `data1` (e.g. `ht_mean_stats(weight ~ sex, data = df)`). With vectors, leave it `NULL` and subset the data yourself.

`first` What goes first in the difference (the difference is mean(first) - mean(other)). With a formula, `first` is the group value exactly as it appears in the data: `first = "male"`, `first = 1`, or a factor level's name as a string. With two vectors, `first` is a number, 1 or 2: which vector, counted in the order you typed them, goes first. See Details for the defaults when `first` is not supplied.

Details

`ht_mean()` is a short alias for `ht_mean_stats()`; both names are exported.

There are two ways to give the samples. With **vectors**, `ht_mean_stats(x)` is a one-sample test and `ht_mean_stats(x, y)` a two-sample test. With a **data frame**, `ht_mean_stats(y ~ group, data = df)` splits the column `y` by the two values of the column `group` – the tilde reads as "by", so this is "test y, by group". Exactly one of these patterns may be used; mixing them (a formula and a second vector, or `data` without a formula) is an error. The grouping variable must have exactly two groups; rows with a missing outcome or group are dropped and counted.

The **direction** of the difference – which mean is subtracted from which – follows one rule: the difference is mean(first) - mean(other), where `first` is chosen by the `first` argument. When `first` is not supplied, the defaults are: with two vectors, the second vector minus the first (treatment - control); with a formula and numeric groups, the larger code minus the smaller (1 - 0, 2 - 1); with a factor, the factor level order; with character groups, alphabetical order. The printed output always states the direction with the actual group labels (e.g. "Difference in Means: male - female"), so it never has to be guessed. (`diff_direction` is the older way to set the direction for vector calls and is still honoured; do not supply both it and `first`.)

Value

An object of class `"statelier_ht"` (a list) with every quantity shown in the printed output (sample statistics, test statistic, p-value, rejection decision, ...). Assigning the result to a variable suppresses printing.

See Also

`ci_mean_stats()` for the corresponding confidence interval, `ht_prop_stats()` for proportions.

Examples

```
set.seed(1)
x <- rnorm(30, mean = 100, sd = 15)
y <- rnorm(40, mean = 105, sd = 15)
```

```
# One-sample test of H0: mu = 100
ht_mean_stats(x, null_value = 100)

# Two-sample test of H0: mean2 - mean1 = 0, Welch
ht_mean_stats(x, y)

# One-sided test with a non-zero null difference
ht_mean_stats(x, y, null_value = 5, alternative = "less")

# Supply alpha to print the significance level and the conclusion
ht_mean_stats(x, null_value = 100, alpha = 0.05)

# The same two-sample test from a data frame:
# "test weight, by sex" (default direction: female - male, alphabetical)
df <- data.frame(
  weight = c(70, 82, 65, 90, 77, 68, 74, 88, 61, 79),
  sex    = c("male", "male", "female", "male", "female",
             "female", "male", "male", "female", "female")
)
ht_mean_stats(weight ~ sex, data = df)
ht_mean_stats(weight ~ sex, data = df, first = "male")
```

`ht_prop_stats` *Hypothesis test for a proportion or a difference in proportions*

Description

Tests a hypothesis about a single population proportion or about the difference between two population proportions, and prints the result in the package's consistent three-section layout: sample information, intermediary calculations, and the final answer – the same layout used by every function in the package. (The reject / fail-to-reject conclusion is shown only when `alpha` is supplied.)

Usage

```
ht_prop_stats(
  data1,
  data2 = NULL,
  p0 = 0.5,
  alpha = NULL,
  alternative = c("two.sided", "less", "greater"),
  diff_direction = c("2-1", "1-2"),
  data = NULL,
  first = NULL
)

ht_prop(
  data1,
  data2 = NULL,
```

```
  p0 = 0.5,
  alpha = NULL,
  alternative = c("two.sided", "less", "greater"),
  diff_direction = c("2-1", "1-2"),
  data = NULL,
  first = NULL
)
```

Arguments

data1	A numeric vector of 0s and 1s with the sample data (logical vectors are coerced), or a formula of the form `outcome ~ group` together with `data`, where the outcome column holds 0s and 1s (or a logical column). Missing values are dropped and counted.
data2	An optional second numeric vector of 0s and 1s. If `NULL` (the default), a one-sample z test is performed; otherwise a two-sample z test of equal proportions is performed. Must not be supplied together with a formula.
p0	The hypothesized population proportion under the null hypothesis for a one-sample test, a single number strictly between 0 and 1. Defaults to 0.5. Ignored (with a warning if non-default) for two-sample tests, where the null hypothesis is always that the difference in proportions is 0.
alpha	Significance level, a single number strictly between 0 and 1. If `NULL` (the default), no significance level is assumed: the significance level and the reject / fail-to-reject conclusion are omitted from the printed output, so the student must compare the p-value to a significance level of their own choosing.
alternative	The alternative hypothesis: `"two.sided"` (the default), `"less"`, or `"greater"`.
diff_direction	Direction of the estimated difference in the two-sample case: `"2-1"` (the default) estimates prop2 - prop1, `"1-2"` estimates prop1 - prop2. Ignored for one-sample tests. Not used with a formula; use `first` there instead.
data	A data frame, used only together with a formula in `data1` (e.g. `ht_prop_stats(passed ~ section, data = df)`). With vectors, leave it `NULL` and subset the data yourself.
first	What goes first in the difference (the difference is prop(first) - prop(other)). With a formula, `first` is the group value exactly as it appears in the data: `first = "morning"`, `first = 1`, or a factor level's name as a string. With two vectors, `first` is a number, 1 or 2: which vector, counted in the order you typed them, goes first. See Details for the defaults when `first` is not supplied.

Details

`ht_prop()` is a short alias for `ht_prop_stats()`; both names are exported.

There are two ways to give the samples. With **vectors**, `ht_prop_stats(x)` is a one-sample test and `ht_prop_stats(x, y)` a two-sample test. With a **data frame**, `ht_prop_stats(y ~ group, data = df)` splits the 0/1 column `y` by the two values of the column `group` – the tilde reads as "by", so this is "test y, by group". Exactly one of these patterns may be used; mixing them (a formula and a second vector, or `data` without a formula) is an error. The grouping variable must have exactly two groups; rows with a missing outcome or group are dropped and counted.

The **direction** of the difference – which proportion is subtracted from which – follows one rule: the difference is prop(first) - prop(other), where `first` is chosen by the `first` argument. When `first` is not supplied, the defaults are: with two vectors, the second vector minus the first (treatment - control); with a formula and numeric groups, the larger code minus the smaller (1 - 0, 2 - 1); with a factor, the factor level order; with character groups, alphabetical order. The printed output always states the direction with the actual group labels, so it never has to be guessed. (`diff_direction` is the older way to set the direction for vector calls and is still honoured; do not supply both it and `first`.)

Both tests use the large-sample normal approximation. The one-sample test standardizes with the standard error under the null hypothesis. The two-sample test standardizes with the pooled-proportion standard error. A warning is issued when the normal approximation looks unreliable (fewer than 5 expected successes or failures).

Value

An object of class `"statelier_ht"` (a list) with every quantity shown in the printed output. Assigning the result to a variable suppresses printing.

See Also

`ci_prop_stats()` for the corresponding confidence interval, `ht_mean_stats()` for means.

Examples

```
set.seed(2)
x <- rbinom(200, 1, 0.35)
y <- rbinom(250, 1, 0.45)

# One-sample test of H0: p = 0.3
ht_prop_stats(x, p0 = 0.3)

# Two-sample test of H0: prop2 - prop1 = 0
ht_prop_stats(x, y)

# Supply alpha to print the significance level and the conclusion
ht_prop_stats(x, p0 = 0.3, alpha = 0.05)

# The same two-sample test from a data frame:
# "test passed, by section" (default direction is alphabetical)
df <- data.frame(
  passed  = c(1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
              1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1),
  section = rep(c("morning", "evening"), each = 20)
)
ht_prop_stats(passed ~ section, data = df)
ht_prop_stats(passed ~ section, data = df, first = "morning")
```

print.statelier_ci *Print a confidence interval*

Description

Prints a `"statelier_ci"` object (as returned by `ci_mean_stats()` or `ci_prop_stats()`) in the package's three-section layout: sample information, intermediary calculations, and the final answer.

Usage

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

Arguments

`x`	A `"statelier_ci"` object.
`...`	Ignored.

Value

`x`, invisibly.

print.statelier_ht *Print a hypothesis test*

Description

Prints a `"statelier_ht"` object (as returned by `ht_mean_stats()` or `ht_prop_stats()`) in the package's three-section layout: sample information, intermediary calculations, and the final answer. The significance level and the reject / fail-to-reject conclusion are shown only when `alpha` was supplied.

Usage

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

Arguments

`x`	A `"statelier_ht"` object.
`...`	Ignored.

Value

`x`, invisibly.

Index

ci_mean *(*ci_mean_stats*)*, 2
ci_mean_stats, 2
ci_mean_stats(), *6*, *8*, *12*
ci_prop *(*ci_prop_stats*)*, 4
ci_prop_stats, 4
ci_prop_stats(), *3*, *11*, *12*

ht_mean *(*ht_mean_stats*)*, 6
ht_mean_stats, 6
ht_mean_stats(), *3*, *11*, *12*
ht_prop *(*ht_prop_stats*)*, 9
ht_prop_stats, 9
ht_prop_stats(), *6*, *8*, *12*

print.statelier_ci, 12
print.statelier_ht, 12