

# Package ‘ci’

October 22, 2025

**Type** Package

**Title** Confidence Intervals for Education

**Version** 0.0.1

**Description** An educational package providing intuitive functions for calculating confidence intervals (CI) for various statistical parameters. Designed primarily for teaching and learning about statistical inference (particularly confidence intervals). Offers user-friendly wrappers around established methods for proportions, means, and bootstrap-based intervals. Integrates seamlessly with Tidyverse workflows, making it ideal for classroom demonstrations and student exercises.

**License** GPL (>= 3)

**URL** <https://github.com/GegznaV/ci>

**BugReports** <https://github.com/GegznaV/ci/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Depends** R (>= 4.1.0)

**Imports** DescTools, dplyr, tidyr, purrr, forcats, tibble, checkmate

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Vilmantas Gegzna [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-9500-5167>>)

**Maintainer** Vilmantas Gegzna <GegznaV@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-10-22 19:30:02 UTC

Contents

ci_binom . . . . .	2
ci_boot . . . . .	3
ci_mean_t . . . . .	6
ci_mean_t_stat . . . . .	8
ci_multinom . . . . .	10
<b>Index</b>	<b>12</b>

---

ci_binom	<i>Proportion CI: Binary Variable (2 groups)</i>
----------	--

---

Description

Calculates confidence intervals (CI) for proportions in binary variables. This enhanced version of [DescTools::BinomCI\(\)](#) returns a data frame.

Usage

```
ci_binom(x, n, method = "modified wilson", conf.level = 0.95, ...)
```

Arguments

x	Number of events of interest or favorable outcomes.
n	Total number of events.
method	Calculation method: "modified wilson", "wilson", "agresti-coull", and others. See <a href="#">DescTools::BinomCI()</a> documentation.
conf.level	Confidence level. Default: 0.95.
...	Additional parameters for <a href="#">DescTools::BinomCI()</a> . See the documentation for that function.

Details

Similar to [DescTools::BinomCI\(\)](#), but uses the modified Wilson method by default and returns a data frame instead of a vector, enabling plotting with **ggplot2**.

Value

- A data frame with columns:
- est (<dbl>) – proportion estimate;
  - lwr.ci, upr.ci (<dbl>) – lower and upper CI bounds;
  - x (<int>) – number of events of interest;
  - n (<int>) – total number of events.

## Examples

```
# Example 1: Survey responses
# 54 out of 80 people agree with a statement
# What is the true proportion of agreement in the population?
ci_binom(x = 54, n = 80)
# Interpretation: We're 95% confident the true proportion
# is between lwr.ci and upr.ci (roughly 0.57 to 0.78)

# Example 2: Medical treatment success
# 23 out of 30 patients recovered
ci_binom(x = 23, n = 30)

# Example 3: Coin flips
# Testing if a coin is fair: 58 heads in 100 flips
ci_binom(x = 58, n = 100)
# If 0.5 is in the CI, we can't rule out the coin being fair

# Example 4: Effect of sample size
# Same proportion (54/80 is approximately 0.675) but different sample sizes
ci_binom(x = 54, n = c(80, 100, 200, 500))
# Notice: Larger samples give narrower (more precise) CIs

# Example 5: Two separate groups
# Group A: 23 successes, Group B: 45 successes
successes <- c(23, 45)
ci_binom(successes, n = sum(successes))

# Example 6: Student exam pass rates
# 67 out of 85 students passed
ci_binom(x = 67, n = 85)

# Example 7: Different confidence levels
# 90% confidence (narrower interval, less confident)
ci_binom(x = 54, n = 80, conf.level = 0.90)

# 99% confidence (wider interval, more confident)
ci_binom(x = 54, n = 80, conf.level = 0.99)

# Example 8: Comparing different methods
ci_binom(x = 15, n = 25, method = "wilson")
ci_binom(x = 15, n = 25, method = "agresti-coull")
# Different methods can give slightly different results
```

---

ci\_boot

*Confidence Intervals via Bootstrap*


---

## Description

Calculates confidence intervals (CI) using bootstrap methods. This enhanced version of `DescTools::BootCI()` returns a data frame.

**Usage**

```
ci_boot(.data, x, y = NULL, conf.level = 0.95, ...)
```

**Arguments**

<code>.data</code>	Data frame.
<code>x, y</code>	Column names (unquoted).
<code>conf.level</code>	Confidence level. Default: 0.95.
<code>...</code>	Additional parameters for <code>DescTools::BootCI()</code> , including: <ol style="list-style-type: none"> <li>1. <code>FUN</code> – function for which CI is calculated;</li> <li>2. <code>bci.method</code> – interval method:           <ul style="list-style-type: none"> <li>• "perc" – percentile method,</li> <li>• "bca" – bias-corrected and accelerated (BCa) method (see note below),</li> <li>• others;</li> </ul> </li> <li>3. <code>R</code> – number of replications, typically 1,000 to 10,000.</li> </ol>

**Details**

Similar to `DescTools::BootCI()`, but:

- First argument is a data frame;
- Arguments `x` and `y` are unquoted column names;
- Responds to `dplyr::group_by()` for subgroup calculations;
- Returns a data frame for convenient plotting with **ggplot2**.

**Value**

A data frame with confidence intervals. Columns depend on arguments and grouping:

- (if grouped) grouping variable names;
- Column matching the statistic name (from `FUN`) containing the estimate;
- `lwr.ci`, `upr.ci` – lower and upper CI bounds.

**Note****Notes:**

1. Each group should have **at least 20** observations for bootstrap methods.
2. Use `set.seed()` for reproducible results.
3. If using `bci.method = "bca"` produces the warning "extreme order statistics used as endpoints", the BCa method is unsuitable; use "perc" instead ([https://rcompanion.org/handbook/E\\_04.html](https://rcompanion.org/handbook/E_04.html)).

## Examples

```
# Bootstrap is useful when:
# - Data is skewed (not normal)
# - You want CI for statistics other than the mean (e.g., median, SD)
# - You don't want to assume a specific distribution

data(iris, package = "datasets")
head(iris)

set.seed(123) # For reproducible results

# Example 1: CI for the median (resistant to outliers)
iris |>
  ci_boot(Petal.Length, FUN = median, R = 1000, bci.method = "perc")
# Compare to mean CI - median is often more robust

# Example 2: CI for the median by group
iris |>
  dplyr::group_by(Species) |>
  ci_boot(Petal.Length, FUN = median, R = 1000, bci.method = "perc")
# Useful when groups have different distributions

# Example 3: CI for standard deviation
# How variable is petal length?
set.seed(456)
iris |>
  ci_boot(Petal.Length, FUN = sd, R = 1000, bci.method = "perc")

# Example 4: CI for interquartile range (IQR)
# IQR = 75th percentile - 25th percentile
set.seed(789)
iris |>
  ci_boot(Petal.Length, FUN = IQR, R = 1000, bci.method = "perc")

# Example 5: CI for correlation coefficient (Pearson's r)
# How related are petal length and width?
set.seed(101)
iris |>
  dplyr::group_by(Species) |>
  ci_boot(
    Petal.Length, Petal.Width,
    FUN = cor, method = "pearson",
    R = 1000, bci.method = "perc"
  )
# Look for CIs that don't include 0 (suggests real correlation)

# Example 6: Comparing BCa and percentile methods
set.seed(111)
# BCa method (often more accurate but requires more assumptions)
iris |> ci_boot(Petal.Length, FUN = median, R = 1000, bci.method = "bca")

# Percentile method (simpler, more robust)
```

```

iris |> ci_boot(Petal.Length, FUN = median, R = 1000, bci.method = "perc")

# Example 7: Effect of number of bootstrap replications
set.seed(222)
# Fewer replications (faster but less stable)
iris |> ci_boot(Petal.Length, FUN = median, R = 500, bci.method = "perc")

# More replications (slower but more stable)
iris |> ci_boot(Petal.Length, FUN = median, R = 5000, bci.method = "perc")
# For teaching: 1000 is usually enough; for research: 5000-10000

# Example 8: Handling missing values
set.seed(333)
iris |>
  ci_boot(
    Petal.Length,
    FUN = median, na.rm = TRUE,
    R = 1000, bci.method = "bca"
  )

# Example 9: With mtcars dataset
set.seed(444)
data(mtcars, package = "datasets")
mtcars |>
  dplyr::group_by(cyl) |>
  ci_boot(mpg, FUN = median, R = 1000, bci.method = "perc")
# Compare median MPG for different cylinder counts

# Example 10: Spearman correlation (rank-based, robust to outliers)
set.seed(555)
iris |>
  dplyr::group_by(Species) |>
  ci_boot(
    Petal.Length, Petal.Width,
    FUN = cor, method = "spearman",
    R = 1000, bci.method = "perc"
  )

```

---

ci\_mean\_t

---

Mean CI from Data

---

## Description

ci\_mean\_t() calculates the mean's confidence interval (CI) using the *classic* formula with Student's t coefficient for data in data frame format. This enhanced version of [DescTools::MeanCI\(\)](#) responds to [dplyr::group\\_by\(\)](#), enabling subgroup calculations. Result is a data frame.

## Usage

```
ci_mean_t(.data, x, conf.level = 0.95, ...)
```

## Arguments

<code>.data</code>	Data frame.
<code>x</code>	Column name (unquoted).
<code>conf.level</code>	Confidence level. Default: 0.95.
<code>...</code>	Additional parameters for <a href="#">DescTools::MeanCI()</a> . See that function's documentation.

## Value

A data frame with columns:

- (if present) grouping variable names;
- `mean (<dbl>)` – mean estimate;
- `lwr.ci`, `upr.ci (<dbl>)` – lower and upper CI bounds.

## Examples

```
# Example with built-in dataset
data(npk, package = "datasets")
head(npk)

# Basic CI calculation for crop yield
ci_mean_t(npk, yield)
# Interpretation: We're 95% confident the true mean yield
# falls between lwr.ci and upr.ci

# Using pipe operator (tidyverse style)
npk |> ci_mean_t(yield)

# Compare yields with nitrogen (N) treatment vs. without
npk |>
  dplyr::group_by(N) |>
  ci_mean_t(yield)
# Look at the CIs: Do they overlap? Non-overlapping CIs suggest
# a potential difference between groups

# More complex grouping: Three factors at once
npk |>
  dplyr::group_by(N, P, K) |>
  ci_mean_t(yield)

# Example with iris dataset: Petal length by species
data(iris, package = "datasets")
iris |>
  dplyr::group_by(Species) |>
  ci_mean_t(Petal.Length)
# Notice how the three species have clearly different intervals

# Example with mtcars: MPG by number of cylinders
data(mtcars, package = "datasets")
```

```
mtcars |>
  dplyr::group_by(cyl) |>
  ci_mean_t(mpg)

# 90% confidence interval (less confident, narrower interval)
npk |> ci_mean_t(yield, conf.level = 0.90)

# 99% confidence interval (more confident, wider interval)
npk |> ci_mean_t(yield, conf.level = 0.99)
```

---

ci\_mean\_t\_stat

*Mean CI from Descriptive Statistics*


---

## Description

ci\_mean\_t\_stat() calculates the mean's confidence interval (CI) using the *classic* formula with Student's t coefficient when descriptive statistics (mean, standard deviation, sample size) are provided. Useful when these values are reported in scientific literature.

## Usage

```
ci_mean_t_stat(mean_, sd_, n, group = "", conf.level = 0.95)
```

## Arguments

mean_	Vector of group means.
sd_	Vector of group standard deviations.
n	Vector of group sizes.
group	Group name. Default: empty string ("").
conf.level	Confidence level. Default: 0.95.

## Value

A data frame with columns:

- group (<fct>) – group name;
- mean (<dbl>) – mean estimate;
- lwr.ci (<dbl>) – lower CI bound (lwr. = lower);
- upr.ci (<dbl>) – upper CI bound (upr. = upper);
- sd (<dbl>) – standard deviation;
- n (<int>) – sample/group size.

Calculations can be performed for multiple groups simultaneously.



**Note**

Each of mean\_, sd\_, n, group must have length (a) of one value, or (b) matching the longest vector in this argument group.

See examples for clarification.

**Examples**

```
# Basic example: Test scores
# Suppose a class of 25 students has a mean score of 75 with SD of 10
ci_mean_t_stat(mean_ = 75, sd_ = 10, n = 25)

# The result tells us we can be 95% confident that the true mean score
# lies between the lower and upper CI bounds

# Example from literature: A study reports mean = 362, SD = 35, n = 100
ci_mean_t_stat(mean_ = 362, sd_ = 35, n = 100)

# Without argument names (order matters: mean, sd, n):
ci_mean_t_stat(362, 35, 100)

# Comparing multiple groups (e.g., teaching methods):
# Method A: mean = 78, SD = 8, n = 30 students
# Method B: mean = 82, SD = 7, n = 28 students
# Method C: mean = 75, SD = 9, n = 32 students
mean_val <- c(78, 82, 75)
std_dev  <- c(8, 7, 9)
n        <- c(30, 28, 32)
group    <- c("Method A", "Method B", "Method C")

ci_mean_t_stat(mean_val, std_dev, n, group)

# Educational example: Effect of sample size on CI width
# Same mean and SD, but different sample sizes
ci_mean_t_stat(mean_ = 75, sd_ = 10, n = c(10, 25, 50, 100))
# Notice: Larger samples give narrower (more precise) confidence intervals

# Educational example: Changing confidence level (default is 95%)
ci_mean_t_stat(mean_ = 75, sd_ = 10, n = 25, conf.level = 0.99)
# 99% CI is wider than 95% CI (more confident = less precise)

# NOTE: Changing conf.level just to get narrower CI is a BAD PRACTICE!
# Please choose confidence level based on study design, not desired CI width.

# To display more decimal places, convert tibble to data frame:
result_ci <- ci_mean_t_stat(75, 10, 25)
as.data.frame(result_ci)

# Or use:
```

```
# View(result_ci)
```

---

ci\_multinom

*Proportion CI: Multinomial Variable (3 or more groups)*


---

## Description

Calculates simultaneous confidence intervals (CI) for proportions in multinomial variables ( $k \geq 3$ ). This enhanced version of `DescTools::MultinomCI()` returns a data frame.

## Usage

```
ci_multinom(
  x,
  method = "goodman",
  conf.level = 0.95,
  gr_colname = "group",
  ...
)
```

## Arguments

x	Vector of group sizes. Best if elements have meaningful names (see examples).
method	Calculation method: "goodman", "sisonglaz", "cplus1", and others. See <a href="#">DescTools::MultinomCI()</a> documentation.
conf.level	Confidence level. Default: 0.95.
gr_colname	Column name (quoted) for group names. Default: "group".
...	Additional parameters for <a href="#">DescTools::MultinomCI()</a> .

## Details

Similar to [DescTools::MultinomCI\(\)](#), but uses the Goodman method by default and returns a data frame, enabling convenient plotting with **ggplot2**.

## Value

A data frame with columns:

- group or user-specified name (<fct>) – group names;
- est (<dbl>) – proportion estimate;
- lwr.ci, upr.ci (<dbl>) – lower and upper CI bounds;
- x (<int>) – group size;
- n (<int>) – total number of events.

**Examples**

```
# Example 1: Student grade distribution
# A: 20 students, B: 35 students, C: 25 students, D/F: 15 students
grades <- c("A" = 20, "B" = 35, "C" = 25, "D/F" = 15)
ci_multinom(grades)
# Each row shows the CI for that grade's proportion

# Example 2: Transportation preferences
transport <- c("Car" = 45, "Bus" = 30, "Bike" = 15, "Walk" = 20)
ci_multinom(transport)

# Example 3: Blood type distribution
blood_types <- c("O" = 156, "A" = 134, "B" = 38, "AB" = 22)
ci_multinom(blood_types)

# Example 4: Political party preference
parties <- c("Party A" = 380, "Party B" = 420, "Party C" = 200)
ci_multinom(parties)

# Unnamed frequencies (groups will be numbered)
ci_multinom(c(20, 35, 54))

# Using pipe operator
c("Small" = 20, "Medium" = 35, "Large" = 54) |>
  ci_multinom()

# Different method for simultaneous intervals
c("Small" = 33, "Medium" = 35, "Large" = 30) |>
  ci_multinom(method = "sisonglaz")

# Custom column name for groups
c("Dog" = 65, "Cat" = 48, "Bird" = 22, "Other" = 15) |>
  ci_multinom(gr_colname = "pet_type")

# Example 5: Teaching method effectiveness
# Outcome categories: Poor, Fair, Good, Excellent
outcomes <- c("Poor" = 8, "Fair" = 22, "Good" = 45, "Excellent" = 35)
ci_multinom(outcomes)
# Look for non-overlapping CIs to identify categories that differ significantly
```

# Index

`ci_binom`, [2](#)  
`ci_boot`, [3](#)  
`ci_mean_t`, [6](#)  
`ci_mean_t_stat`, [8](#)  
`ci_multinom`, [10](#)  
  
`DescTools::BinomCI()`, [2](#)  
`DescTools::BootCI()`, [3](#), [4](#)  
`DescTools::MeanCI()`, [6](#), [7](#)  
`DescTools::MultinomCI()`, [10](#)  
`dplyr::group_by()`, [4](#), [6](#)  
  
`set.seed()`, [4](#)