

How to Perform Bootstrapping in R (With Examples)

Authored by
stats writer

December 18, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Perform Bootstrapping in R (With Examples)*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=107898>

Bootstrapping in R is an indispensable resampling technique widely used in modern statistical analysis. Its primary function is to estimate the sampling distribution of a statistic by repeatedly drawing subsamples from an existing data set with replacement. This method is particularly valuable when working with limited data or when the underlying population distribution is unknown or non-parametric, making traditional analytical methods that rely on distributional assumptions unreliable.

To effectively utilize bootstrapping within the R environment, one must typically employ the specialized functions available in the boot package. The process involves defining a custom function that calculates the desired statistic and then applying the primary bootstrap function to the data set to generate thousands of replicates. This robust methodology allows researchers to obtain crucial estimates like the standard error and precise confidence intervals for complex statistics, moving beyond simple means or standard deviations. Examples demonstrating the practical implementation of bootstrapping in R are detailed below, showcasing its application for both single and multiple statistical measures.

The concept of Bootstrapping, often attributed to Bradley Efron, provides a powerful non-parametric alternative to classical statistical inference. This methodology allows us to estimate the precision (e.g., standard error) of almost any statistic derived from a sample and subsequently produce reliable confidence intervals without relying on restrictive theoretical assumptions about the data's distribution.

The Core Mechanism of Bootstrapping

The foundational strength of bootstrapping lies in its simplicity and generality. Instead of assuming a theoretical population distribution, the bootstrap method treats the observed sample data as if it were the population itself. By repeatedly sampling from this observed data with replacement, we simulate the process of drawing independent samples from the underlying population. This iterative process generates an empirical distribution--the bootstrap distribution--which serves as a highly effective approximation of the true sampling distribution of the statistic of interest.

Executing the bootstrap procedure requires following a clear, systematic routine to ensure robust estimates. This process is highly automated in R through dedicated packages, but understanding the steps is crucial for proper interpretation. The process simulates drawing repeated samples, where each sample is known as a bootstrap replicate.

The basic process for performing bootstrapping is as follows:

Take **R** repeated samples with replacement from a given dataset, where R is typically a large number (e.g., 1000 or 5000). These are called bootstrap replicates.

For each resampled dataset, calculate the statistic you're interested in (e.g., the mean, median, R-squared value, or regression coefficient).

This process yields **R** different estimates for the given statistic, forming the empirical bootstrap distribution. This distribution is then used to calculate the standard error of the statistic and construct a reliable confidence interval.

Utilizing the R boot package for Statistical Inference

In the R environment, the heavy lifting of resampling and calculation is managed efficiently by the functions contained within the highly popular boot library. This package provides the primary tools necessary for generating bootstrap samples and calculating the resulting confidence intervals. Mastering two key functions, `boot()` and `boot.ci()`, is essential for implementing this technique correctly.

The workflow begins by defining a statistical function that calculates the parameter of interest. This function is then passed to `boot()` along with the data and the desired number of replicates (**R**). The output of `boot()`, a dedicated bootstrap object, contains all the necessary information about the empirical distribution, including the original estimate, bias, and standard error.

We can perform bootstrapping in R by utilizing the following functions from the boot library:

Generate bootstrap samples using the primary function:

boot(data, statistic, R, ...)

where the arguments define the core parameters of the resampling procedure:

data: This argument specifies the dataset to be bootstrapped. It can be provided as a vector, matrix, or data frame.

statistic: This is a user-defined function that calculates the statistic or statistics to be bootstrapped. Crucially, this function must accept three arguments: `data`, `formula` (if applicable), and `indices`.

R: This specifies the **Number of bootstrap replicates** (the number of times the resampling process is repeated). A higher R generally leads to more stable results.

Generate a bootstrapped confidence interval using the dedicated function:

boot.ci(bootobject, conf, type)

where the arguments govern the output of the precision analysis:

bootobject: This must be the output object generated by a prior call to the `boot()` function.

conf: This determines the desired confidence interval level to calculate. Default is 0.95 (representing 95%).

type: This specifies the methodology used to calculate the confidence interval. Options include "norm", "basic", "stud", "perc", "bca" and "all". Using "bca" (Bias-Corrected and Accelerated) is often recommended for its robustness against distribution skewness. Default is "all".

The following detailed examples illustrate how to implement these functions effectively in practice, focusing first on a single statistic, and then expanding to multiple statistics simultaneously.

Example 1: Bootstrapping a Single Statistic (R-squared)

One of the powerful applications of bootstrapping is determining the uncertainty associated with model fit metrics, such as the R-squared value in a linear regression. Traditional methods for calculating the standard error of R-squared are complex or rely on strong assumptions; bootstrapping bypasses these issues by directly simulating the variation across samples. This example demonstrates how to calculate the standard error and confidence interval for the R-squared of a simple linear regression model using the built-in `mtcars` dataset.

We begin by defining the statistical function, `rsq_function`, which specifically extracts the R-squared value from the linear model fit. It is crucial that this function accepts the `indices` argument, which allows the `boot()` function to select the appropriate resampled observations for each replication. We then execute 2,000 bootstrap replicates ($R=2000$) for the model predicting miles per gallon (`mpg`) based on displacement (`disp`).

```
set.seed(0)
```

```
library(boot)
```

```
#Define function to calculate R-squared. Note the 'indices' argument.
```

```
rsq_function <- function(formula, data, indices) {
```

```
  d <- data #Allows boot to select the resampled data subset
```

```
  fit <- lm(formula, data=d) #Fit the regression model using the resampled data
```

```
  return(summary(fit)$r.square) #Return the R-squared statistic
```

```
}
```

```
#Perform bootstrapping with 2000 replicates (R=2000)
```

```
reps <- boot(data=mtcars, statistic=rsq_function, R=2000, formula=mpg~disp)
```

```
#View the primary results of the bootstrapping simulation
```

```
reps
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
Call:
```

```
boot(data = mtcars, statistic = rsq_function, R = 2000, formula = mpg ~  
disp)
```

Bootstrap Statistics :

original bias std. error

```
t1* 0.7183433 0.002164339 0.06513426
```

From the results of the bootstrap simulation, we can extract critical information regarding the R-squared statistic:

The estimated R-squared for this regression model, obtained from the original data, is **0.7183433**.

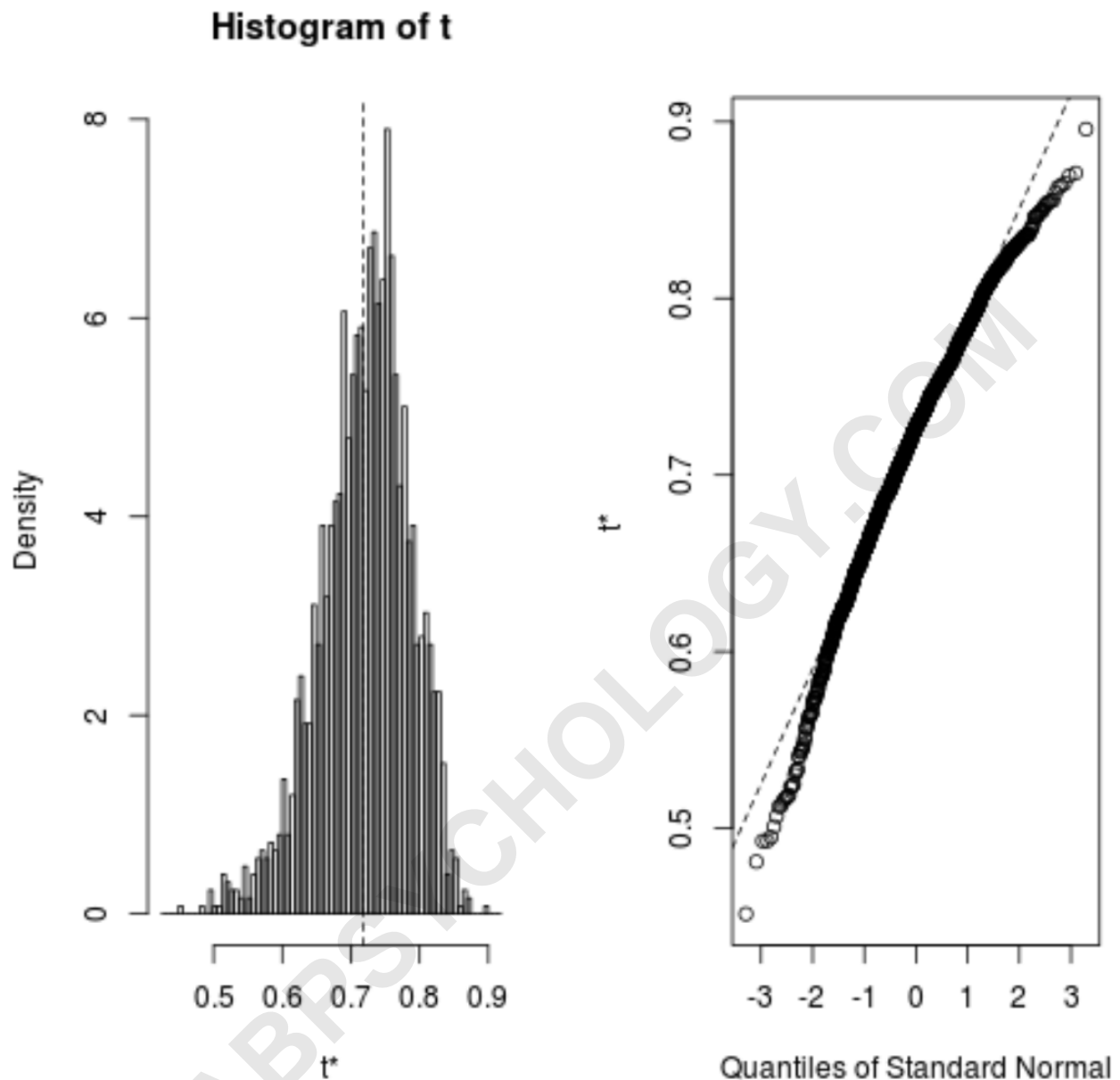
The standard error for this estimate, derived from the variability across the 2,000 bootstrap replicates, is **0.06513426**.

Visualizing the Bootstrap Distribution

Before calculating confidence intervals, it is advisable to visualize the empirical distribution of the bootstrapped estimates. This visualization, typically a histogram, provides insight into the shape, centrality, and spread of the calculated statistic, confirming whether the distribution is reasonably symmetrical.

We can quickly view the distribution of the bootstrapped R-squared samples using the generic `plot()` function applied to the bootstrap object:

```
plot(reps)
```



Calculating the Bootstrapped Confidence Interval

The ultimate goal of many bootstrapping procedures is to construct a reliable confidence interval that captures the true population parameter. Using the `boot.ci()` function, we can calculate various types of intervals. For robust estimation, particularly when the bootstrap distribution shows slight asymmetry, the Bias-Corrected and Accelerated (BCa) method is highly recommended, as it adjusts for both bias and skewness in the distribution.

We calculate the 95% BCa confidence interval for the estimated R-squared of the model using the following code:

#Calculate adjusted bootstrap percentile (BCa) interval

```
boot.ci(reps, type="bca")
```

CALL :

```
boot.ci(boot.out = reps, type = "bca")
```

Intervals :

Level BCa

95% (0.5350, 0.8188)

Calculations and Intervals on Original Scale

From the output we can see that the 95% bootstrapped confidence interval for the true R-squared value is (.5350, .8188). This interval provides a robust measure of uncertainty around the model's fit.

Example 2: Bootstrapping Multiple Regression Coefficients

Bootstrapping is highly flexible and is not limited to single statistics; it can simultaneously estimate the standard error and confidence intervals for multiple parameters, such as the coefficients in a linear regression model. This application is vital for understanding the variability associated with each predictor variable in the model.

In this second example, we adapt the statistical function to return all fitted regression coefficients (the intercept and the slope for `disp`). The output of the `boot()` function will now contain separate bootstrap statistics (t_1^* , t_2^* , etc.) for each coefficient estimated by the model.

```
set.seed(0)
```

```
library(boot)
```

```
#Define function to calculate fitted regression coefficients
```

```
coef_function <- function(formula, data, indices) {
```

```
  d <- data #Allows boot to select sample
```

```
  fit <- lm(formula, data=d) #Fit regression model
```

```
  return(coef(fit)) #Return all coefficient estimates
```

```
}
```

```
#Perform bootstrapping with 2000 replications
```

```
reps <- boot(data=mtcars, statistic=coef_function, R=2000, formula=mpg~disp)
```

```
#View results of bootstrapping
```

```
reps
```

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

```
boot(data = mtcars, statistic = coef_function, R = 2000, formula = mpg ~  
disp)
```

Bootstrap Statistics :

original bias std. error

```
t1* 29.59985476 -5.058601e-02 1.49354577
```

```
t2* -0.04121512 6.549384e-05 0.00527082
```

From the results we can see the statistics derived for both the intercept ($t1^*$) and the predictor variable ($t2^*$):

The estimated coefficient for the intercept of the model is **29.59985476** and the standard error of this estimate is **1.49354577**.

The estimated coefficient for the predictor variable `disp` in the model is **-0.04121512** and the standard error of this estimate is **0.00527082**.

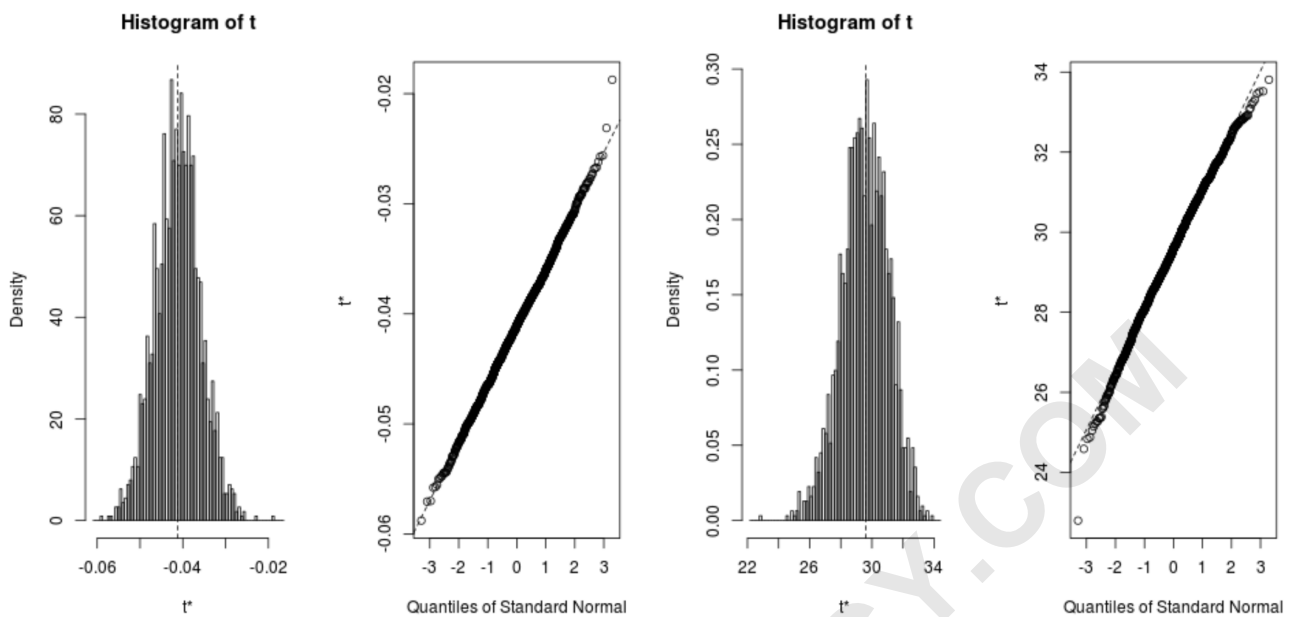
The standard error provides a measure of how much the coefficient estimate varies across the 2,000 resampled datasets, offering a clear picture of the coefficient's stability.

Visualizing and Calculating Confidence Intervals for Multiple Statistics

To individually inspect the distributions for these multiple coefficients, we must specify the index of the statistic when calling `plot()`. Index 1 corresponds to the intercept, and Index 2 corresponds to the `disp` variable.

```
plot(reps, index=1) #Distribution for the intercept of the model
```

```
plot(reps, index=2) #Distribution for the disp predictor variable
```



We can also use the following code to calculate the 95% BCa confidence intervals for each coefficient, ensuring we specify the appropriate index for each call to `boot.ci()`:

```
#Calculate adjusted bootstrap percentile (BCa) intervals
boot.ci(reps, type="bca", index=1) #Intercept of model
boot.ci(reps, type="bca", index=2) #disp predictor variable
```

CALL :

```
boot.ci(boot.out = reps, type = "bca", index = 1)
```

Intervals :

Level BCa

95% (26.78, 32.66)

Calculations and Intervals on Original Scale

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS

Based on 2000 bootstrap replicates

CALL :

```
boot.ci(boot.out = reps, type = "bca", index = 2)
```

Intervals :

Level BCa

95% (-0.0520, -0.0312)

Calculations and Intervals on Original Scale

From the output we can see that the 95% bootstrapped confidence intervals for the model coefficients are as follows:

C.I. for intercept: (26.78, 32.66)

C.I. for `disp`: (-0.0520, -0.0312)

These intervals provide strong evidence regarding the statistical significance and estimated range of effect for each variable, utilizing the robust methodology of the bootstrap to ensure accuracy.

Further Resources on Related Topics

For those interested in deepening their understanding of the underlying statistical models utilized in these examples, the following resources provide additional context:

[How to Perform Simple Linear Regression in R](#)

[How to Perform Multiple Linear Regression in R](#)

[Introduction to Confidence Intervals](#)