Learning Bootstrapping Techniques in R: A Step-by-Step Guide with Examples

Authored by

Mohammed looti

November 6, 2025

The technique of bootstrapping is one of the most powerful and flexible non-parametric methods available in modern statistics. It offers a robust approach for estimating the sampling distribution of almost any statistic, particularly when traditional analytical methods are difficult or impossible to apply. Fundamentally, bootstrapping allows researchers to estimate the standard error of a statistic and construct reliable confidence interval estimates without relying on strong distributional assumptions about the underlying population.

Understanding the Power of Bootstrapping

In classical inference, calculating the standard error and confidence intervals often requires knowing the population distribution (e.g., assuming normality) or deriving complex analytical formulas based on large sample theory. However, real-world data frequently violates these assumptions or involves complex statistics for which no simple formula exists. Bootstrapping elegantly bypasses these challenges by treating the observed sample as a proxy for the entire population.

This technique is particularly valuable in computational statistics, allowing researchers to assess the variability and stability of model parameters and summary statistics. By repeatedly resampling from the available data, bootstrapping simulates the process of drawing independent samples from the theoretical population, providing an empirical estimate of the sampling distribution. This computational intensity is easily handled by modern statistical software like R, making complex variance estimation accessible even for intricate models, such as generalized linear models or machine learning algorithms.

The Fundamental Process of Bootstrapping

The core principle behind bootstrapping is the concept of resampling with replacement. We use the original dataset, which represents our best estimate of the population distribution, and generate numerous new datasets, known as bootstrap samples. Each bootstrap sample is the same size as the original data, but because we sample with replacement, some original observations may appear multiple times, while others may not appear at all.

This iterative process generates an empirical approximation of the sampling distribution of the statistic of interest. By analyzing the spread and characteristics of the statistic calculated from these thousands of simulated samples, we can quantify its uncertainty. The resulting empirical distribution provides the foundation needed to calculate the standard error and derive various types of confidence intervals, such as percentile or bias-corrected and accelerated (BCa) intervals.

The basic computational procedure for performing a bootstrap analysis is streamlined into the following steps:

Take *k* repeated samples (often 1,000 to 10,000) with replacement from the given dataset. These are the bootstrap replicates.
For each bootstrap sample, calculate the specific statistic (e.g., mean, median, regression coefficient, R-squared value) that you are interested in estimating.
The result is *k* different estimates for that statistic. The standard deviation of these *k* estimates serves as the estimated standard error of the statistic, and the distribution of these estimates is used to create a reliable confidence interval.

Essential R Functions for Bootstrapping

In the R environment, bootstrapping is most efficiently performed using the dedicated boot library. This package provides two primary functions necessary for conducting a complete bootstrap analysis: the `boot()` function for generating bootstrap samples and computing the statistic, and the `boot.ci()` function for calculating confidence intervals based on the generated output.

The `boot()` function is central to the process. It requires the user to define a custom function that specifies the statistic to be calculated. This ensures maximum flexibility, as virtually any measurable property of the data--whether it is a simple mean or a complex output from a linear regression model--can be bootstrapped.

1. Generate bootstrap samples using the `boot()` function.

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

where the key parameters are:

data: The original dataset, which can be provided as a vector, matrix, or data frame.
statistic: A user-defined R function that specifies precisely which statistic(s) should be calculated for each bootstrap replicate. This function must accept three arguments: the data, a formula (if applicable), and an index vector (used internally by `boot()` to select the resampled observations).
R: The crucial parameter defining the number of bootstrap replicates (resamples) to perform. A higher value of R generally leads to greater stability and accuracy in the estimates, though 1,000 to 5,000 is often sufficient for basic estimates.

2. Generate a bootstrapped confidence interval using the `boot.ci()` function.

boot.ci(bootobject, conf, type)

This function takes the results of the resampling process and calculates the specified confidence interval:

bootobject: The object returned by the `boot()` function, containing all the bootstrap statistics.

conf: The confidence level desired for the interval. This parameter defaults to 0.95 (95% confidence).
type: The method used to calculate the confidence interval. Options include "norm" (normal approximation), "basic" (basic percentile), "stud" (studentized), "perc" (percentile), and "bca" (bias-corrected and accelerated). Using the default value, "all," computes all available types. The BCa interval is generally considered the most accurate method when the sampling distribution is skewed.

The following examples demonstrate how to implement these functions in practice, first for a single statistic like R-squared, and then for multiple simultaneous statistics, such as regression coefficients.

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

In this first example, we utilize the bootstrapping methodology to estimate the R-squared value of a simple linear regression model and accurately quantify the uncertainty surrounding that estimate. We will model the relationship between miles per gallon (`mpg`) and engine displacement (`disp`) using the built-in R dataset `mtcars`. Because R-squared is a complex statistic derived from the fit of the model, bootstrapping provides a robust way to determine its sampling variability.

To begin, we must define a custom function (`rsq_function`) that specifically tells the `boot()` function how to calculate R-squared for each resampled dataset. The `indices` argument is crucial here; it allows the `boot()` function to select the appropriate rows for each bootstrap iteration. We then execute 2,000 bootstrap replicates (R=2000) to ensure a stable estimate of the sampling distribution.

```
set.seed(0)
library(boot)

#define function to calculate R-squared
rsq_function <- function(formula, data, indices) {
d <- data #allows boot to select sample
fit <- lm(formula, data=d) #fit regression model
return(summary(fit)$r.square) #return R-squared of model
}
#perform bootstrapping with 2000 replications
reps <- boot(data=mtcars, statistic=rsq_function, R=2000, formula=mpg~disp)

#view results of boostrapping
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

The output of the `boot()` function provides three critical pieces of information derived from the 2,000 replicates. The **original** value (t1*) is the R-squared calculated using the complete, unresampled dataset. The **bias** indicates the systematic difference between the mean of the bootstrap estimates and the original estimate, which is generally small in well-behaved bootstraps. Most importantly, the **std. error** column provides the estimated standard error based on the standard deviation of the 2,000 R-squared estimates.

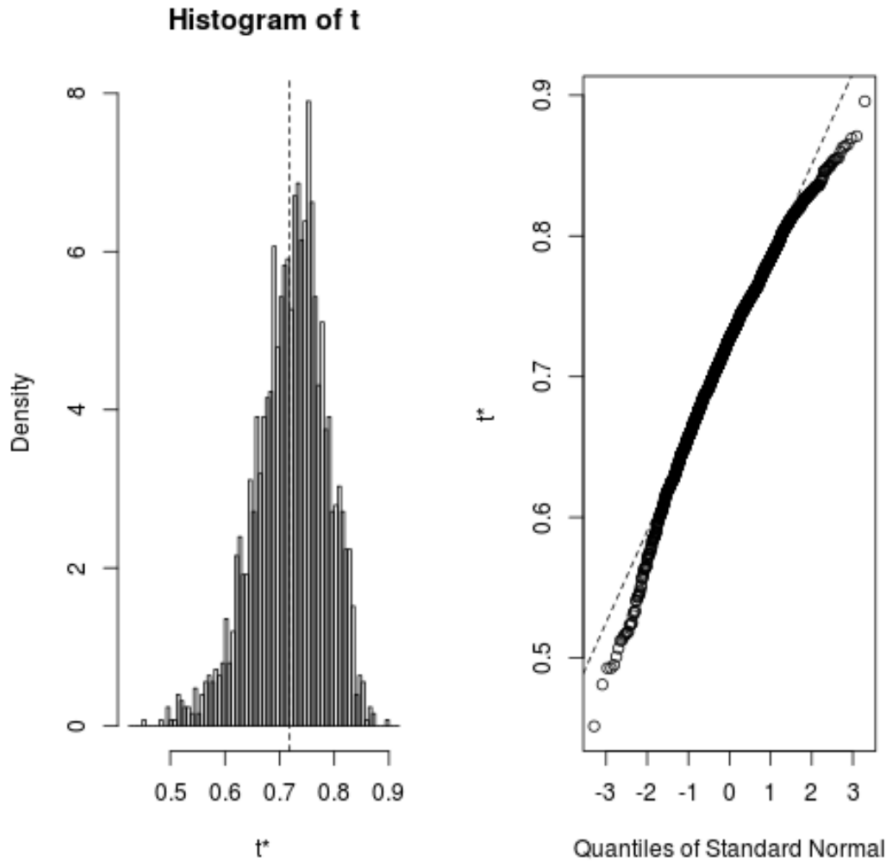
From these results, we can draw the following conclusions regarding the variability of the R-squared estimate:

The estimated R-squared for this simple regression model is approximately **0.7183433**, indicating that about 71.8% of the variability in MPG is explained by engine displacement.
The estimated **standard error** for this statistic is **0.06513426**. This value quantifies the precision of our R-squared estimate, suggesting how much this value might vary if we were to draw new samples from the population.

Analyzing the Distribution and Confidence Intervals

Visualization is a key component of understanding any sampling distribution. We can quickly view the empirical distribution of the 2,000 bootstrapped R-squared values using the standard plot function applied to the `reps` object. This visualization helps confirm if the distribution is approximately normal or if it exhibits skewness, which would justify using more sophisticated confidence interval methods like BCa.

plot(reps)



Beyond the standard error, the primary goal of bootstrapping is often to construct a reliable confidence interval. Since the distribution of R-squared is often non-normal (as it is bounded between 0 and 1), the bias-corrected and accelerated (BCa) method is typically preferred as it accounts for both the skewness and bias present in the bootstrap distribution. We use the `boot.ci()` function, specifying `type="bca"`, to calculate the 95% confidence interval for the true R-squared of the model.

#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

The output indicates that the 95% bootstrapped confidence interval for the true R-squared value

lies between 0.5350 and 0.8188. This interval provides a highly dependable range, reflecting the uncertainty inherent in the estimate, based solely on the observed data and the resampling process.

Example 2: Bootstrapping Multiple Regression Coefficients

Bootstrapping is not limited to single summary statistics; it is equally effective for estimating the standard errors and confidence intervals for multiple parameters simultaneously, such as the coefficients in a multiple linear regression model. When dealing with regression coefficients, bootstrapping is essential, especially when assumptions about the error terms might be questionable or when the sample size is small, making traditional standard errors less reliable.

For this example, we again use the `mtcars` dataset to model `mpg` as a function of `disp`. The procedure remains similar to Example 1, but the custom statistic function (`coef_function`) is modified to return the vector of coefficients (the intercept and the slope for `disp`) rather than the R-squared value. The `boot()` function automatically handles the simultaneous bootstrapping of these multiple statistics.

```
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 coefficient estimates of model
}

#perform bootstrapping with 2000 replications
reps <- boot(data=mtcars, statistic=coef_function, R=2000, formula=mpg~disp)

#view results of boostrapping
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

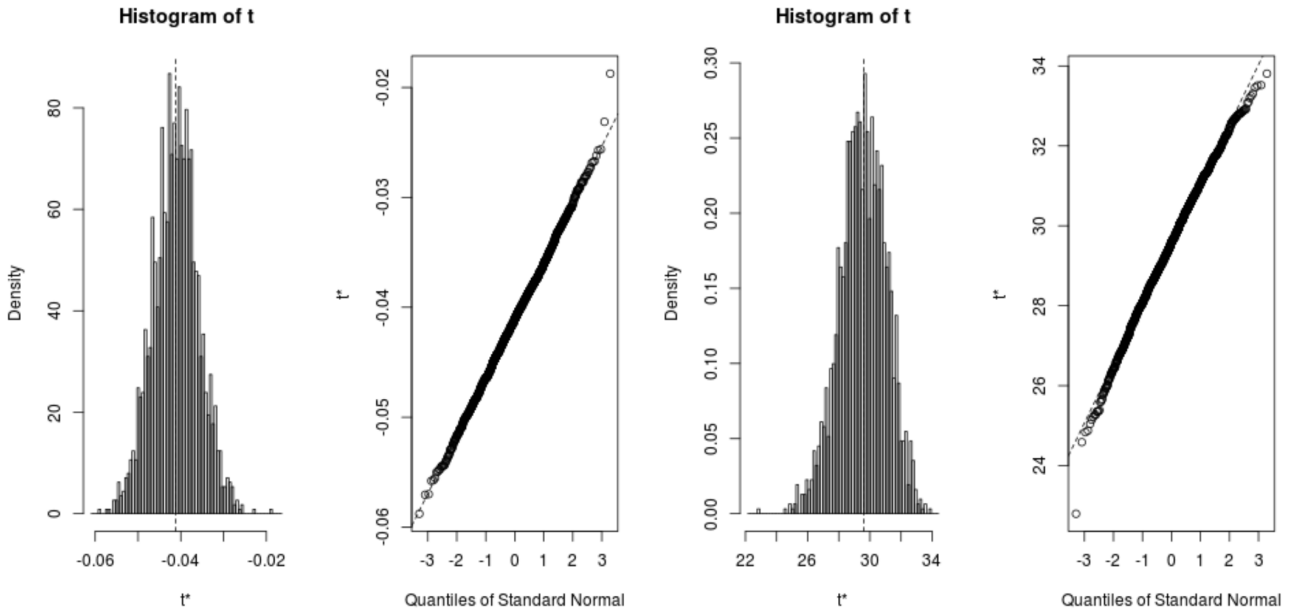
The output now shows two sets of bootstrap statistics: `t1*` corresponds to the first coefficient (the intercept) and `t2*` corresponds to the second coefficient (the slope for `disp`). We can interpret these rows to understand the stability of each parameter estimate.

Based on the Bootstrap Statistics table:

For the intercept (t1*), the original estimated coefficient is **29.59985476**, and the associated standard error is **1.49354577**.
For the predictor variable *disp* (t2*), the original estimated coefficient is **-0.04121512**, and the standard error of this estimate is **0.00527082**.

The low standard errors relative to the coefficient magnitudes suggest that both parameters are estimated with reasonable precision, given the sample size. We can visualize the distributions for these two statistics separately by specifying the index in the plot function. Index 1 refers to the intercept (t1*), and Index 2 refers to the slope (t2*).

plot(reps, index=1) #intercept of model
plot(reps, index=2) #disp predictor variable



Finally, we calculate the 95% confidence intervals for each coefficient individually, again using the BCa method to ensure robustness against potential skewness in the coefficient distributions. We

must specify the index parameter within `boot.ci()` to select the appropriate statistic.

#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

The calculated 95% bootstrapped confidence intervals for the model coefficients are as follows, providing the range of plausible true population values for each parameter:

C.I. for intercept: (26.78, 32.66).
C.I. for *disp* (slope): (-.0520, -.0312). Since this interval does not include zero, we have strong empirical evidence that engine displacement is a statistically significant predictor of MPG, consistent with the original regression model.

Conclusion and Further Exploration

The bootstrapping method, facilitated by the R `boot` package, provides an indispensable tool for estimating the uncertainty associated with complex statistics, particularly in situations where classical parametric methods fall short. By understanding the resampling process and the application of the `boot()` and `boot.ci()` functions, researchers can generate highly reliable standard errors and robust confidence intervals for virtually any statistic derived from their data. This computational approach reinforces the integrity of statistical inference across diverse fields.

To deepen your understanding of the underlying models used in these examples, consider exploring these resources:

How to Perform Simple Linear Regression in R
How to Perform Multiple Linear Regression in R
Introduction to Confidence Intervals