How to Easily Fit Probability Distributions in R Using fitdistr()

Authored by

stats writer

November 19, 2025

The `fitdistr()` function, a staple tool within the statistical computing environment of the R programming language, provides a highly efficient mechanism for fitting observed data to theoretical probability distributions. This process involves estimating the necessary parameters that define the distribution, thereby creating a mathematical model that best characterizes the underlying data structure. This capability is essential across various fields, including finance, engineering, and biostatistics, where understanding the data generation process is paramount.

When modeling real-world phenomena, data rarely follows perfect, theoretical assumptions. `fitdistr()` addresses this challenge by applying established statistical methodologies to find the optimal fit. It supports a wide array of classical distributions--such as the **Normal**, **Gamma**, **Poisson**, and **Weibull**--making it versatile for different types of continuous and discrete data analysis. The core objective is to identify the parameters (like mean and standard deviation for the Normal distribution) that maximize the likelihood of observing the actual data set.

Beyond simple parameter estimation, utilizing `fitdistr()` is the first step in a thorough statistical analysis pipeline. Although the function itself primarily focuses on parameter estimation via Maximum Likelihood Estimation, the resulting model parameters are crucial for subsequent analyses, including hypothesis testing and simulation studies. Understanding the output, including parameter estimates and associated standard errors, is vital for assessing the quality and reliability of the fitted model.

Theoretical Foundation: Maximum Likelihood Estimation (MLE) in R

The primary mechanism employed by the `fitdistr()` function is **Maximum Likelihood Estimation (MLE)**. MLE is a powerful statistical technique used for estimating the parameters of a statistical model. The core idea is to find the parameter values that make the observed data most probable. In simpler terms, the method searches for the distribution parameters that maximize the likelihood function, which quantifies how well the hypothesized distribution explains the data.

The reason MLE is so widely adopted is due to its desirable statistical properties, particularly for large samples. When the sample size is large, MLE estimates are typically consistent (they converge to the true parameter value), asymptotically unbiased, and asymptotically efficient (they achieve the lowest possible variance). The `fitdistr()` function handles the complex iterative optimization required to find these maximizing parameters automatically, providing a convenient interface for complex statistical procedures.

While `fitdistr()` is highly effective, users should be aware that MLE relies on the assumption that the chosen theoretical distribution is indeed the correct model for the data. If the underlying distribution is fundamentally different from the one being fitted (e.g., trying to fit a Normal distribution to highly skewed data), the resulting parameter estimates, while maximizing the likelihood of the chosen distribution, may not accurately reflect the true data generation process.

Therefore, preliminary data visualization and exploratory analysis are always recommended before applying `fitdistr()`.

Accessing the Function: The MASS Package

To use the powerful parameter estimation capabilities of `fitdistr()`, users must first ensure they have access to the **Modern Applied Statistics with S (MASS)** package. This package, developed by renowned statisticians W. N. Venables and B. D. Ripley, is an essential resource in the MASS package ecosystem, providing a comprehensive collection of functions and datasets for statistical modeling.

The process of utilizing the function involves two standard steps in R: installation and loading. Although **MASS** is often installed by default with R, it still needs to be explicitly loaded into the current session using the `library(MASS)` command before `fitdistr()` can be called. Failing to load the package will result in an error indicating that the function cannot be found, a common pitfall for new R users.

It is important to recognize that `fitdistr()` is not a standalone function but part of a broader statistical toolkit. By providing tools for robust parameter estimation across numerous distribution types, the **MASS** package significantly enhances R's functionality for applied statistical work. The reliance on this established package assures users that the underlying computational methods are rigorous and well-tested, delivering reliable parameter estimates crucial for academic research and professional data analysis.

Syntax and Arguments of `fitdistr()`

The `fitdistr()` function offers a streamlined syntax that makes parameter estimation relatively straightforward, requiring only the data vector and the name of the target distribution. Understanding the required arguments is crucial for successful execution, especially when dealing with distributions that require specific starting values or additional optional parameters.

The fundamental syntax structure of the function is provided below. It is important to note the reliance on the density function names available within R, which dictate the specific distribution being analyzed.

```
fitdistr(x, densefun, ...)
```

This structure outlines the minimal requirements for fitting a distribution. Let us break down the essential components that drive the function's calculations:

`x`: This is a **numeric vector** representing the sample data set for which the distribution parameters are to be estimated. The quality and size of this input vector directly influence the accuracy and

convergence of the MLE process.
`densefun`: This argument specifies the name of the probability distribution whose parameters are to be estimated. This must be supplied as a character string corresponding to the density function suffix (e.g., `"normal"` for `dnorm`, `"gamma"` for `dgamma`).
`...`: This placeholder represents optional arguments, which often include **start values**. For many complex distributions (such as Gamma or Weibull), providing reasonable starting values for the optimization routine is necessary to ensure convergence and prevent the algorithm from settling into local maxima instead of the global maximum likelihood.

The `densefun` argument accepts a broad range of well-known distributions, allowing for flexibility in modeling diverse data characteristics. The accepted values include, but are not limited to, the following key theoretical models:

Continuous Distributions: **beta**, **cauchy**, **chi-squared**, **exponential**, **gamma**, **lognormal**, **logistic**, **normal**, **t**, and **Weibull**.
Discrete Distributions: **geometric**, **negative binomial**, and **Poisson**.

Selecting the correct distribution name is paramount, as the function will try to fit the data to the specified model regardless of whether that model is appropriate. The selection process should always be guided by theoretical knowledge of the data and careful visual inspection, ensuring the chosen distribution aligns with the characteristics (e.g., support, skewness, modality) of the observed data.

Practical Demonstration: Setting up Sample Data

To illustrate the practical application of `fitdistr()`, we will begin by generating synthetic data that is known to follow a specific distribution. This approach allows us to compare the estimated parameters obtained by `fitdistr()` against the true, known parameters, serving as a powerful validation check.

In this demonstration, we use the `rnorm()` function in R to simulate a sample of 200 observations drawn from a **Normal distribution**. We define the true parameters as a mean ((mu)) of 10 and a standard deviation ((sigma)) of 3. Crucially, we use `set.seed(1)` to ensure that our results are reproducible, allowing others to verify the process precisely.

The following code block demonstrates the data generation process and an initial inspection of the resulting vector:

#make this example reproducible
set.seed(1)

#generate sample of 200 observations that follows normal dist with mean=10 and sd=3
data <- rnorm(200, mean=10, sd=3)

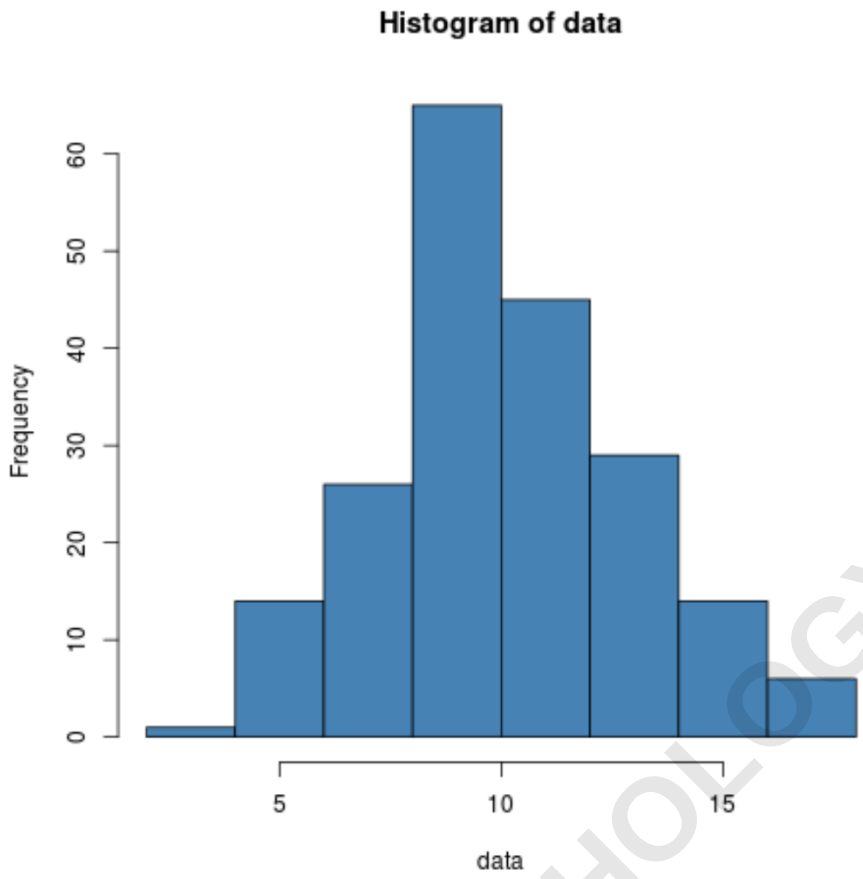
#view first 6 observations in sample
head(data)

8.120639 10.550930 7.493114 14.785842 10.988523 7.538595

Once the data is generated, visual inspection becomes the next crucial step. Before attempting to fit any distribution, it is standard practice to visualize the data's shape using a histogram. This graphical representation provides an immediate assessment of the data's central tendency, spread, and symmetry, confirming whether the assumption of normality is plausible.

We use the built-in `hist()` function in R to create a histogram of our simulated data. This visualization helps confirm that the generated sample data roughly approximates the expected bell shape characteristic of a Normal distribution, even though random sampling introduces minor deviations.

hist(data, col='steelblue')



The resulting histogram visually confirms that the data is concentrated around the center (near 10) and tails off symmetrically, strongly suggesting that the **Normal distribution** is an appropriate model candidate for this specific data set. This visual confirmation provides the necessary justification for proceeding with the `fitdistr()` estimation for the Normal parameters.

Fitting the Normal Distribution and Interpreting Output

With the sample data prepared and the distributional assumption verified, we can now execute the primary function of interest: using `fitdistr()` to estimate the mean and standard deviation of our data set. This calculation utilizes the iterative optimization inherent to the MLE process to find the parameter pair that maximizes the likelihood function.

We must first load the **MASS** library, and then we pass the data vector and the distribution name (`"normal"`) to the function. Since the Normal distribution is relatively simple and robust, we do not need to provide explicit starting values for the optimization.

library(MASS)

#estimate parameters of distribution

```
fitdistr(data, "normal")

mean sd
10.1066189 2.7803148
( 0.1965979) ( 0.1390157)
```

The output of `fitdistr()` is highly informative, providing more than just the estimated parameters. The top row displays the estimated parameters themselves. In this case, the estimated mean is approximately **10.1066**, and the estimated standard deviation is approximately **2.7803**. These estimates are very close to the true parameters used to generate the data (mean=10, sd=3), demonstrating the effectiveness of the Maximum Likelihood approach.

The values enclosed in parentheses immediately below the estimates represent the **standard errors** of those parameter estimates. The standard error measures the variability or uncertainty associated with the estimates. A smaller standard error indicates higher precision and reliability of the estimate. For the mean, the standard error is 0.1966, and for the standard deviation, it is 0.1390. These standard errors are critical for constructing confidence intervals and performing subsequent statistical inference regarding the fitted distribution.

Thus, the `fitdistr()` function estimates that our vector of values follows a Normal distribution with a mean of **10.1066189** and a standard deviation of **2.7803148**. This robust estimation provides the mathematical foundation for using this fitted distribution in further predictive modeling or simulation studies.

Extending Analysis and Considering Goodness-of-Fit

While the parameter estimates derived from `fitdistr()` are essential, they do not inherently prove that the chosen distribution provides a good description of the data. For instance, if we had attempted to fit a Gamma distribution to this normally distributed data, `fitdistr()` would still return parameters, but the model fit would be poor.

Therefore, after successful parameter estimation, the next crucial step in distributional fitting is evaluating the quality of the fit using Goodness-of-fit tests. These statistical tests assess the compatibility between the observed sample data and the hypothesized distribution model. Common goodness-of-fit methods include the **Kolmogorov-Smirnov test**, the **Anderson-Darling test**, and the **Chi-squared test**.

Although `fitdistr()` focuses on parameter optimization via Maximum Likelihood Estimation, the resulting estimates are used as input for these subsequent goodness-of-fit procedures (often found in other R packages like `fitdistrplus`). A high-quality model fit is confirmed when the test statistic indicates that the observed data does not significantly differ from the theoretical distribution

defined by the estimated parameters. If the goodness-of-fit test rejects the null hypothesis of a suitable fit, the analyst must return to the drawing board, perhaps selecting an alternative distribution or transforming the data.

In conclusion, `fitdistr()` is an indispensable function for applying statistical theory to empirical data. By accurately estimating distribution parameters using MLE and providing the necessary standard errors, it equips the data scientist with the tools required for robust statistical modeling. Always pair parameter estimation with rigorous goodness-of-fit evaluation to ensure the chosen model truly represents the underlying reality of the data.

How to Plot a Normal Distribution in R

How to Perform a Shapiro-Wilk Test for Normality in R