How to Plot a Log Normal Distribution in R: A Step-by-Step Guide

Authored by

stats writer

December 28, 2025

Generating visualizations of statistical distributions is fundamental for data analysis and modeling. The Log Normal Distribution (LND) is particularly common in fields such as finance, engineering, and environmental science, often describing phenomena where values are skewed and strictly positive. To effectively visualize the LND within the R statistical environment, we primarily rely on density functions provided by R's statistical package, specifically dlnorm(), combined with graphical tools like curve().

While the `qlnorm()` function is used to calculate quantiles--finding the value at a specific probability--the objective of plotting the distribution's shape requires working with the Probability Density Function (PDF). The PDF illustrates the likelihood of a continuous random variable falling within a particular range, providing a smooth curve that defines the distribution's characteristics. Understanding the interplay between the underlying parameters, the logarithmic mean (meanlog) and standard deviation (sdlog), is essential for accurately plotting the LND.

This guide will walk through the process of using R's powerful built-in functions to generate, customize, and compare visualizations of the Log Normal Distribution. We will focus on utilizing dlnorm() for density calculations and curve() for direct plotting, ensuring the resulting graphs are informative, aesthetically pleasing, and meet professional presentation standards.

Essential R Functions for Density Plotting

To accurately visualize the shape of the Log Normal Distribution, we must generate a sequence of density values across a defined range of the variable. The R language provides specialized functions within the standard statistical package that streamline this process, allowing users to define the distribution parameters easily and efficiently generate the plot.

The primary tool for calculating the density of the log-normal distribution is the dlnorm() function. This function calculates the probability density at specific points (x) given the distribution's parameters. Crucially, the parameters used here--meanlog and sdlog--refer to the mean and standard deviation of the variable's logarithm, not the variable itself. By default, R sets these parameters to 0 and 1, respectively, corresponding to a standard log-normal distribution.

Once the density function is defined using dlnorm(), the curve() function takes over the plotting responsibility. This function is designed specifically to generate a plot by evaluating an R expression, typically a function of 'x', over a specified range. It automatically handles the iteration and graphical rendering, making it far simpler than generating x and y vectors manually and using a generic plot function. The combination of these two functions is the standard, most robust method for visualizing the LND PDF.

The core functions utilized for plotting the Probability Density Function (PDF) for a log normal

distribution in R are summarized below:

dlnorm(x, meanlog = 0, sdlog = 1): This calculates the density values for the distribution based on the provided x-values and logarithmic parameters.
curve(function, from = NULL, to = NULL): This takes the density function expression and plots it smoothly across a user-defined interval, specified by the from and to arguments.

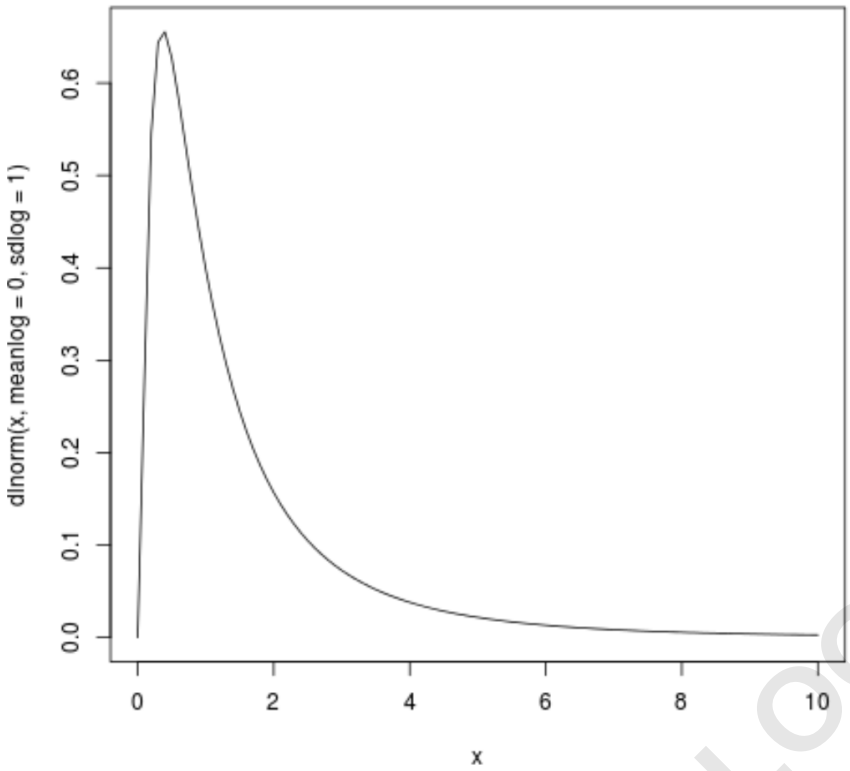
Generating the Basic Lognormal PDF Curve

To create a baseline visualization, we will illustrate how to plot a probability density function for a log normal distribution using standard settings. For this initial example, we will assume a logarithmic mean of 0 and a logarithmic standard deviation of 1. These parameters define the canonical form of the LND. We must also specify the range for the x-axis, which is particularly important for log-normal distributions since they are defined only for positive values.

In this scenario, we set the plotting range for the x-axis from 0 to 10. This range is selected to adequately capture the body and the long tail characteristic of the standard log-normal curve, which typically peaks early and decays slowly. The expression passed to the curve() function is simply dlnorm(x, meanlog=0, sdlog=1), where the variable 'x' is automatically substituted with values across the specified range by the plotting environment.

The execution of the command below yields the fundamental log-normal shape. Notice how the curve() function simplifies the entire process. Instead of manually defining hundreds of points and handling the graphical setup, R executes the density calculation and renders the plot with a single line of code, demonstrating the efficiency of R's statistical graphics capabilities.

curve(dlnorm(x, meanlog=0, sdlog=1), from=0, to=10)



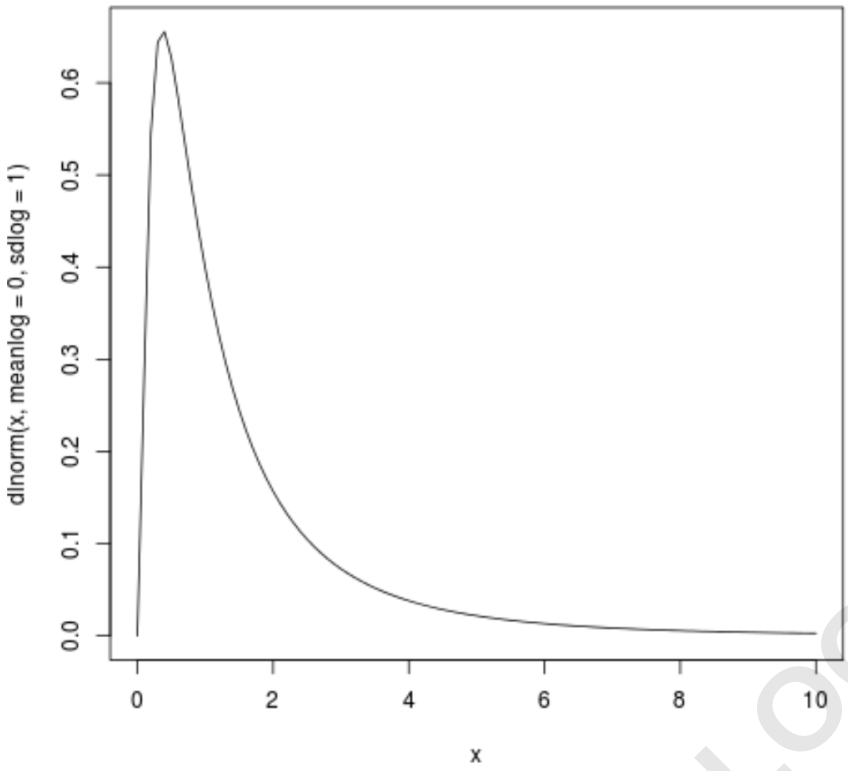
Leveraging Default Parameters for Quick Visualization

A key feature of R's statistical functions is the use of intelligent default parameters. For the standard dlnorm() function, the logarithmic meanlog is set to 0 and the logarithmic sdlog is set to 1. This means that if we intend to plot the standard log-normal distribution, we do not need to explicitly specify these parameters within the function call.

Omitting the default arguments simplifies the code substantially without altering the visual output. This is highly beneficial for quick exploratory data analysis (EDA) and when generating plots based on the most common distribution settings. The dlnorm() function recognizes that only the variable 'x' is necessary, relying on its internal definitions for the parameters. The resulting plot is mathematically identical to the previous example where the parameters were explicitly set.

The following code snippet demonstrates this concise approach. By calling dlnorm(x), we instruct R to use the default logarithmic mean of zero and standard deviation of one. This practice encourages cleaner code and minimizes potential input errors when dealing with standard distributions, while still plotting the full range from 0 to 10.

curve(dlnorm(x), from=0, to=10)



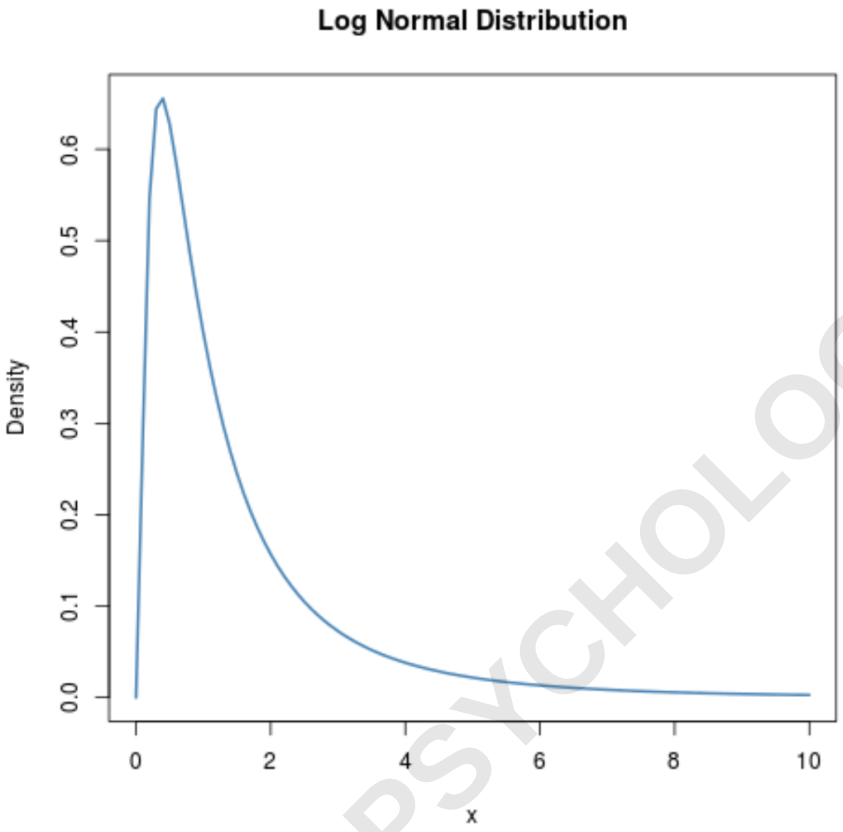
Enhancing Aesthetics: Customizing Plot Appearance

While the default plot generated by the curve() function provides the necessary mathematical information, customizing its aesthetics is essential for professional reporting and enhanced readability. R's base plotting system offers extensive graphical parameters that can be passed directly within the curve() command, allowing users to define titles, axis labels, line attributes, and colors with ease.

Common customizations include defining a descriptive title using the main argument, which clearly informs the viewer about the content of the visualization. Additionally, clarity is improved by redefining the y-axis label using ylab, often changed from the default function call to a more meaningful term like 'Density'. Furthermore, attributes like line width (lwd) and line color (col) can be modified to improve visual impact and distinguish the curve from the background elements. Increasing the line width, for instance, makes the curve more prominent and easier to track.

In the example provided, we apply several graphical parameters to transform the basic plot into a publication-ready figure. We set a clear title ('Log Normal Distribution'), update the vertical axis label to 'Density', increase the line width to 2, and change the line color to a distinct 'steelblue'. These modifications significantly improve the visual quality and interpretability of the Log Normal Distribution plot.

```
curve(dlnorm(x), from=0, to=10,
main = 'Log Normal Distribution', #add title
ylab = 'Density', #change y-axis label
lwd = 2, #increase line width to 2
col = 'steelblue') #change line color to steelblue
```



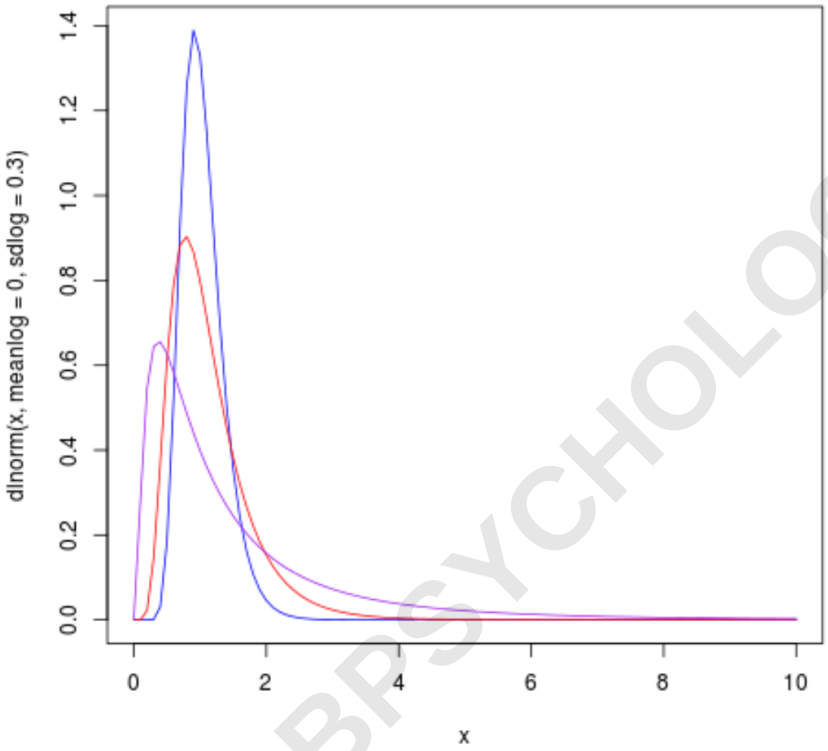
Comparative Analysis: Plotting Multiple Distributions

A powerful application of plotting distribution functions is the ability to compare multiple curves simultaneously. This is crucial for understanding how changes in distribution parameters, particularly the logarithmic standard deviation (sdlog), affect the spread and peak of the Log Normal Distribution. When comparing distributions, we typically hold the meanlog constant (here, meanlog = 0) and vary the sdlog parameter.

To overlay subsequent curves onto an existing plot, we must utilize the add = TRUE argument within the curve() function call. The first call to curve() establishes the graphical environment and axes; subsequent calls with add = TRUE simply draw the new curve without resetting the plot area. It is vital to assign a unique line color (col) to each distribution to ensure clear differentiation between them.

The following example demonstrates how three distinct log-normal distributions are plotted together. We examine the effect of varying sdlog values: 0.3 (least spread), 0.5, and 1.0 (most spread). Notice that as sdlog increases, the peak density value decreases, and the tail extends further along the x-axis, illustrating the increased variability inherent in the distribution.

curve(dlnorm(x, meanlog=0, sdlog=.3), from=0, to=10, col='blue')
curve(dlnorm(x, meanlog=0, sdlog=.5), from=0, to=10, col='red', add=TRUE)
curve(dlnorm(x, meanlog=0, sdlog=1), from=0, to=10, col='purple', add=TRUE)



Interpreting and Labeling: Implementing the Legend Function

When multiple distributions are overlaid on a single plot, adding a legend becomes absolutely mandatory for interpretation. The legend() function in R is used to map the visual attributes (like color or line type) of the curves to their corresponding statistical definitions (like the sdlog value). A well-placed and clearly defined legend prevents ambiguity and ensures the reader can correctly distinguish between the different curves being compared.

The legend() function requires several crucial parameters, the most important of which define its location, the descriptive text, and the corresponding visual characteristics. The location is usually defined by the x and y coordinates, or by using keywords like "topright" or "bottomleft." The legend argument accepts a vector of strings corresponding to the labels. Furthermore, the col argument

must contain a vector matching the colors used in the plot, ensuring the labels align correctly with the lines.

The general syntax for the legend() function highlights the necessary configuration options:

legend(x, y=NULL, legend, fill, col, bg, lty, cex)

where the key components manage the visual display:

x, y: These define the precise coordinates used to anchor the legend position within the plotting area.
legend: This vector contains the textual labels that will appear in the legend, describing each plotted element.
fill: Used for filled boxes (e.g., bar plots); not typically used for lines.
col: The list of line colors corresponding to the items in the legend text.
bg: Specifies the background color for the legend box, often set to white for contrast.
lty: Defines the line type (e.g., solid, dashed) used in the legend keys.
cex: Controls the scaling factor for the text size within the legend, useful for adjustment.

Finalizing the Comparative Plot with Legend

To finalize our comparative visualization, we integrate the definition of the three curves along with the appropriate legend. It is crucial that the plotting commands (curve()) are executed first, establishing the graphical viewport, before the legend() command is called, as the latter depends on the existence of an active plot.

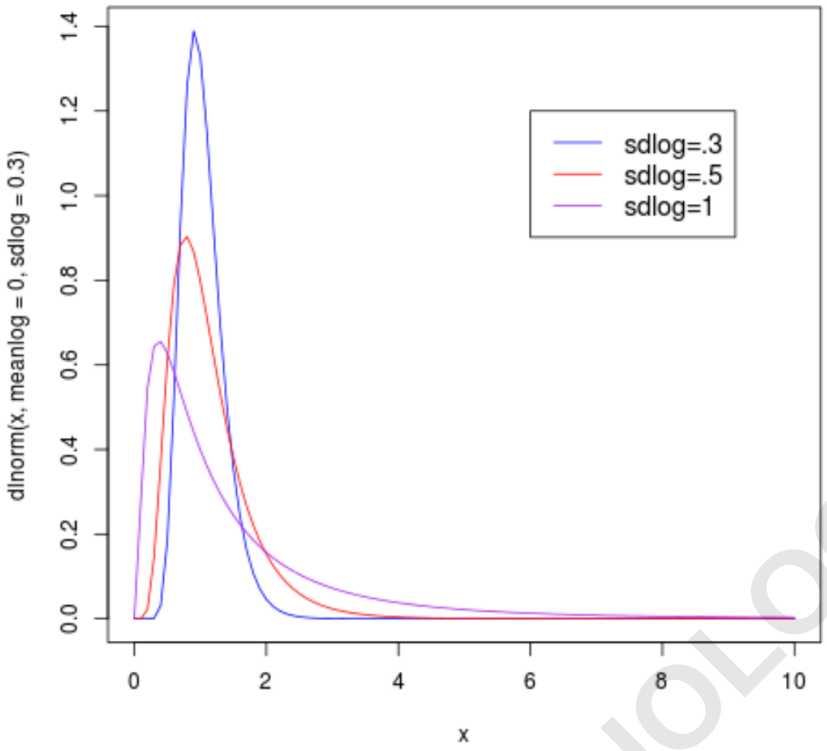
In our complete example, the legend is positioned using the coordinates x=6 and y=1.2, chosen carefully to avoid overlapping the dense section of the curves near the y-axis origin. The textual labels clearly indicate the sdlog value for each line. We use lty=1 for solid lines and cex=1.2 to slightly increase the text size, ensuring high visibility. This comprehensive approach provides a visually rich and statistically informative plot of the varying log-normal distributions.

The complete code block below demonstrates the necessary steps to generate the density plots and subsequently overlay the descriptive legend.

```
#create density plots
curve(dlnorm(x, meanlog=0, sdlog=.3), from=0, to=10, col='blue')
curve(dlnorm(x, meanlog=0, sdlog=.5), from=0, to=10, col='red', add=TRUE)
curve(dlnorm(x, meanlog=0, sdlog=1), from=0, to=10, col='purple', add=TRUE)

#add legend
legend(6, 1.2, legend=c("sdlog=.3", "sdlog=.5", "sdlog=1"),
```

col=c("blue", "red", "purple"), lty=1, cex=1.2)



Summary of Best Practices for R Visualization

Successfully plotting the Log Normal Distribution in R relies on a systematic approach utilizing specialized statistical and graphical functions. Always start by defining the distribution using dlnorm(), ensuring that the logarithmic mean and standard deviation parameters are correctly specified or allowed to default to 0 and 1, respectively. Utilize the curve() function for generating the smooth PDF curve over an appropriate positive range, such as 0 to 10.

For enhanced clarity, always invest time in customizing the plot aesthetics. Clear titles and axis labels (using main and ylab) are critical for contextualizing the data. When comparing multiple distributions, ensure that the add=TRUE argument is included in all subsequent curve() calls, and distinct colors are used for differentiation. The final, non-negotiable step for comparative plots is adding a descriptive legend using the legend() function, positioned strategically to avoid obscuring the data.

By following these best practices, R users can transition from simple, default graphical outputs to sophisticated visualizations that effectively communicate the nuances and characteristics of the Log Normal Distribution across various parameter settings, supporting rigorous statistical interpretation.