Plotting Log-Normal Distributions in R: A Step-by-Step Guide

Authored by

Mohammed looti

November 8, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Plotting Log-Normal Distributions in R: A Step-by-Step Guide*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=13395

Fundamentals of the Log-Normal Distribution and R Tools

The Log Normal Distribution is a cornerstone statistical model indispensable across numerous quantitative disciplines. It is frequently employed when modeling random variables that are inherently positive, such as financial asset prices, epidemiological incubation periods, or environmental pollutant concentrations, and typically exhibit a pronounced positive skewness. By definition, a variable follows a log-normal distribution if the logarithm of that variable adheres to a normal (Gaussian) distribution. To fully appreciate the characteristics and parameter effects of this model, visualizing its shape through the Probability Density Function (PDF) plot is absolutely essential.

To execute accurate statistical visualizations of the log-normal density, we utilize the powerful capabilities of the R programming environment. The base installation of R provides a comprehensive suite of statistical functions specifically designed for modeling and plotting probability distributions. These tools allow practitioners to generate complex density curves efficiently, providing immediate visual feedback on how distribution parameters influence the data's potential shape.

Generating a distribution plot in R fundamentally relies on the seamless integration of two key functions. One function calculates the precise density values for the distribution, while the other handles the crucial task of rendering these values graphically over a specified continuous range. Mastering the interaction between these two elements is central to effective statistical visualization in this environment.

Key R Functions for Log-Normal Density Calculation

For plotting the Log Normal Distribution, two specialized functions work synergistically to produce the desired PDF curve. These functions abstract the mathematical complexity, allowing users to focus on parameter selection and visualization design.

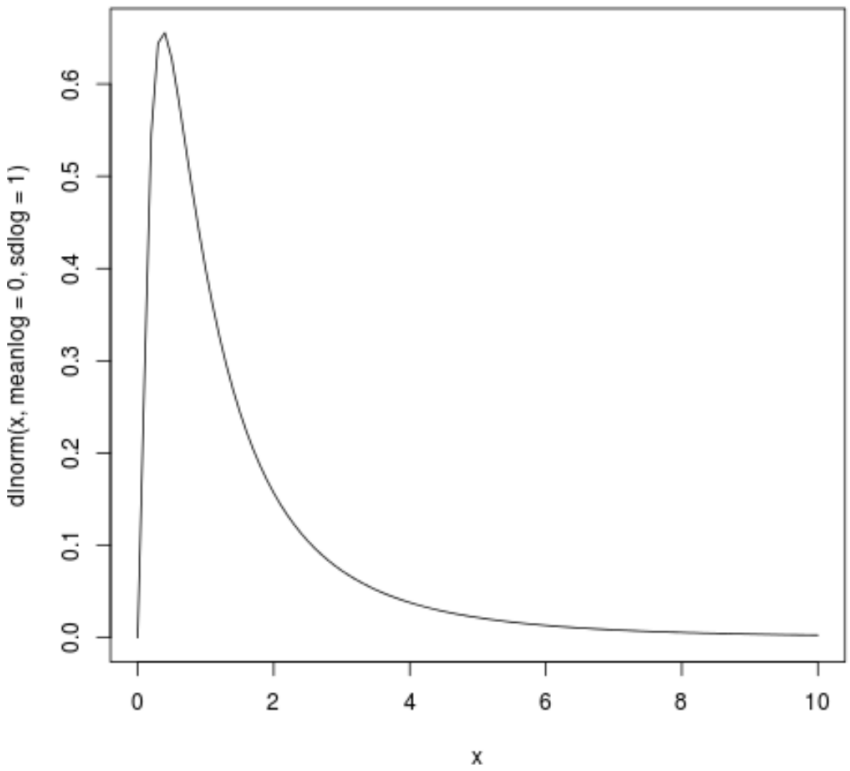
`dlnorm(x, meanlog = 0, sdlog = 1)`: This is the primary function used to calculate the probability density for a log-normal distribution at specific value `x`. The parameters are crucial: `meanlog` and `sdlog` do not represent the mean and standard deviation of the variable itself, but rather the mean and standard deviation of the variable's **logarithm**, which is assumed to be normally distributed.
`curve(function, from = NULL, to = NULL)`: This highly versatile graphical function is dedicated to plotting the results of any R function that accepts a single input variable (typically named `x`). We pass the `dlnorm()` call directly into the `curve()` function. The `from` and `to` arguments define the boundaries for the x-axis, determining the range over which the density curve will be calculated and displayed.

Generating the Initial Log-Normal Density Plot

The first step in any visualization project is establishing the parameters that define the distribution and the range required to capture its essential characteristics. We begin by plotting the standard log-normal distribution, which is defined by the default parameters: `meanlog = 0` and `sdlog = 1`. This configuration provides a baseline understanding of the distribution's characteristic positive skew. We will set the x-axis range from 0 to 10, ensuring that the visible plot encompasses the vast majority of the probability mass for this standard configuration.

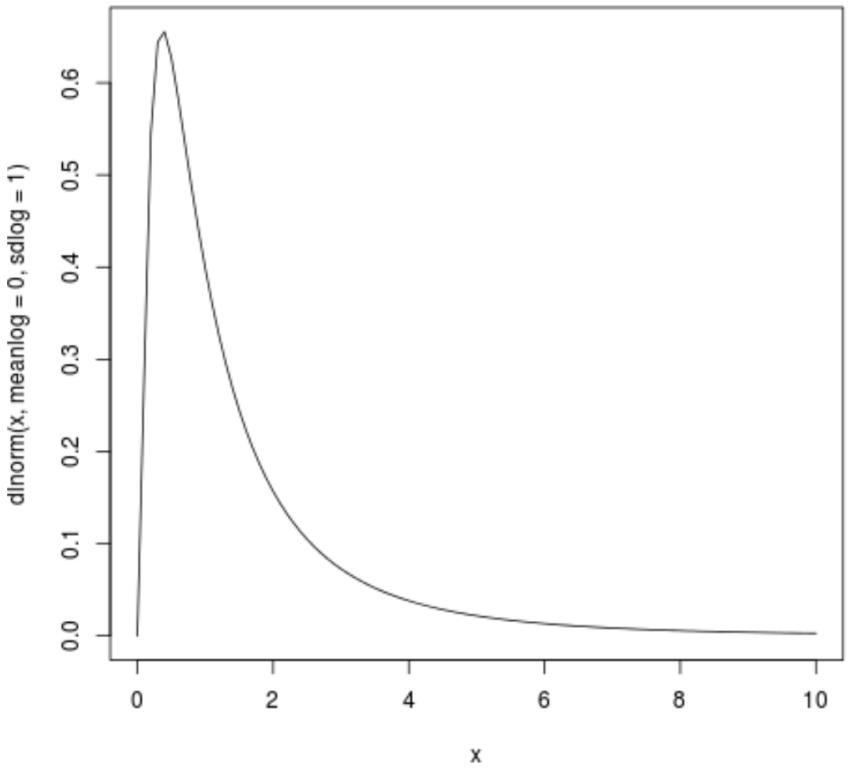
The code below demonstrates the concise syntax required to generate this basic plot. Notice how the `dlnorm()` function is nested within `curve()`. This structure tells R to calculate the density for every point between 0 and 10 according to the log-normal parameters specified, and then immediately render the resulting line graph.

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



A significant convenience feature of R's statistical functions is the use of intelligent default values. When plotting the standard log-normal distribution, the arguments `meanlog` and `sdlog` can be safely omitted from the `dlnorm()` call. R automatically substitutes the default values of 0 and 1, leading to a more streamlined and compact syntax without any change to the visual output. This simplification is highly beneficial when quick, standard visualizations are required.

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



Customizing R Plots: Enhancing Visual Aesthetics

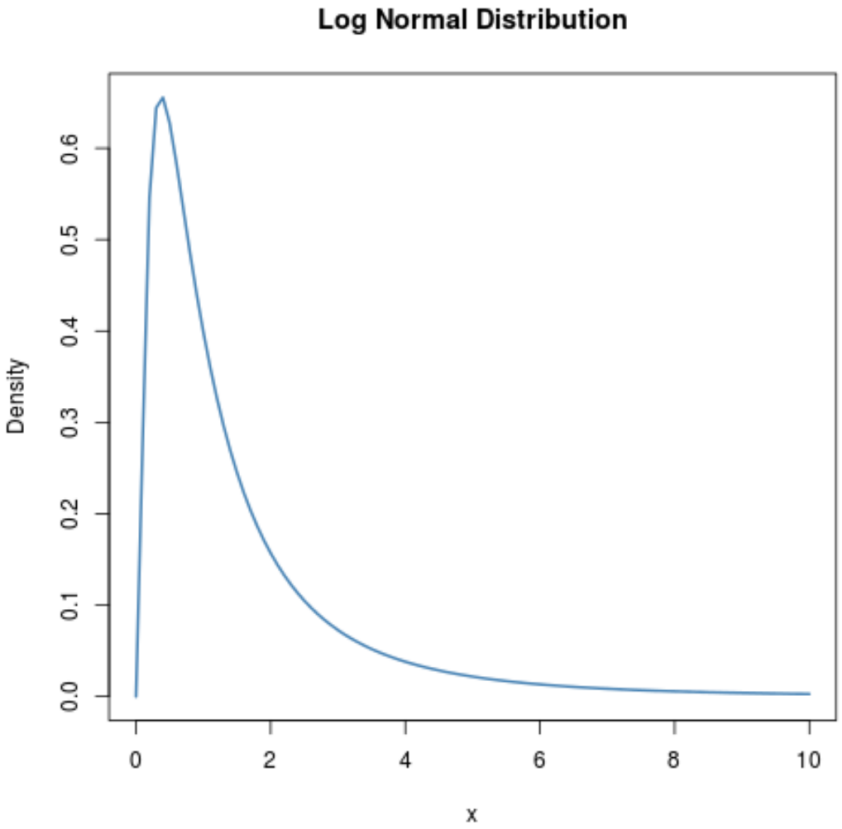
While the basic output accurately reflects the Probability Density Function, default R plots often lack the necessary visual refinement for professional presentation. To transform a simple graph into a compelling, publication-ready figure, it is crucial to leverage R's extensive graphical parameters to improve clarity, impact, and overall aesthetics. These parameters are integrated directly into the `curve()` function call, utilizing standard base R plotting arguments.

Enhancing the plot involves several key modifications: adding a descriptive title to contextualize the data, providing meaningful labels for the axes, and adjusting the visual attributes of the curve itself. Properly implemented, these elements significantly boost the plot's interpretability. The primary arguments used for these adjustments include `main` (for the plot title), `ylab` (for the y-axis label), `lwd` (controlling line width), and `col` (defining the line color).

The following example demonstrates how to apply these aesthetic enhancements in practice. We assign a clear title, explicitly label the y-axis as 'Density' for statistical precision, increase the line width (`lwd = 2`) to make the curve more visible, and change the color to 'steelblue' for a cleaner, more appealing visual presentation. These straightforward additions elevate the graph from a

rudimentary output to an informative visualization.

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



Visual Comparison of Multiple Distributions

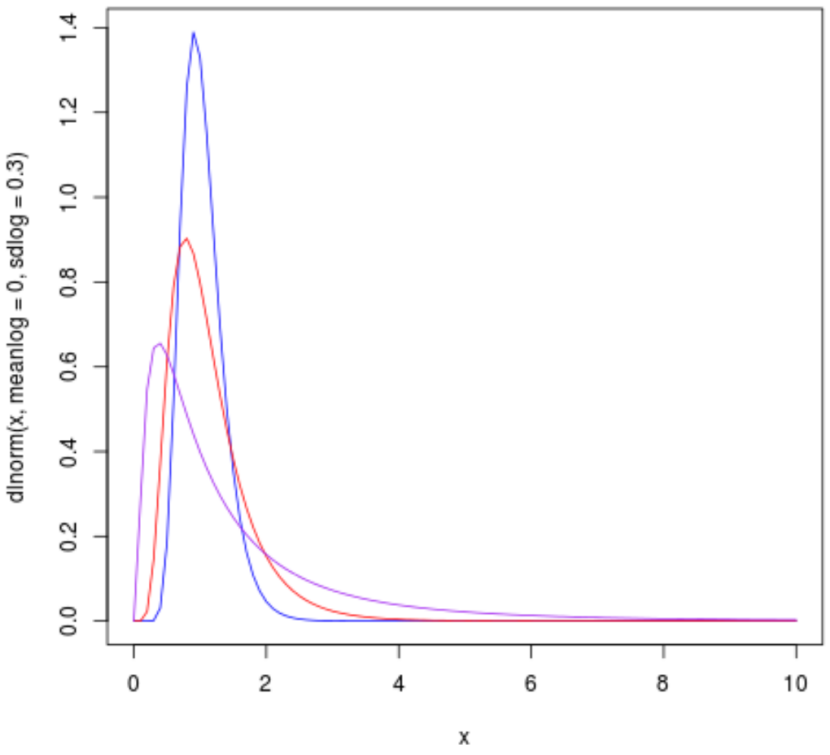
A frequent requirement in statistical analysis is the ability to visually compare how variations in parameters affect the distribution's shape. When exploring the Log Normal Distribution, analyzing the impact of changing `sdlog`--the standard deviation of the logarithm--is particularly illuminating, as this parameter directly controls the spread and kurtosis (peakedness) of the curve.

To plot several distinct density curves on a single graphical device in R, the process requires careful execution. The first distribution is plotted using the standard `curve()` command, which initializes the plot area. However, every subsequent call to `curve()` must include the critical argument **`add=TRUE`**. This flag prevents R from overwriting the previous plot and instead instructs

the system to superimpose the new curve onto the existing axes, thereby enabling direct, side-by-side visual comparison.

The example below illustrates a comparison of three log-normal distributions, all sharing the same `meanlog=0` but exhibiting significant differences in their spread due to varying `sdlog` values: 0.3, 0.5, and 1.0. Observe the resulting visual pattern: a lower value for the logarithmic standard deviation yields a curve that is much taller and narrower, reflecting a dataset with lower inherent variability and less spread.

```
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)
```



Ensuring Clarity: Adding an Interpretive Legend

Although overlaying multiple density curves is effective for direct visual comparison, the plot remains incomplete and potentially misleading without a clear key to identify which parameter set corresponds to which colored line. The **`legend()`** function is absolutely necessary in R to provide this essential context, ensuring that any reader can accurately interpret the relationship between the parameter values and the resulting curve shapes.

The structure of the `legend()` function is flexible, allowing for meticulous control over its location, content, and appearance:

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

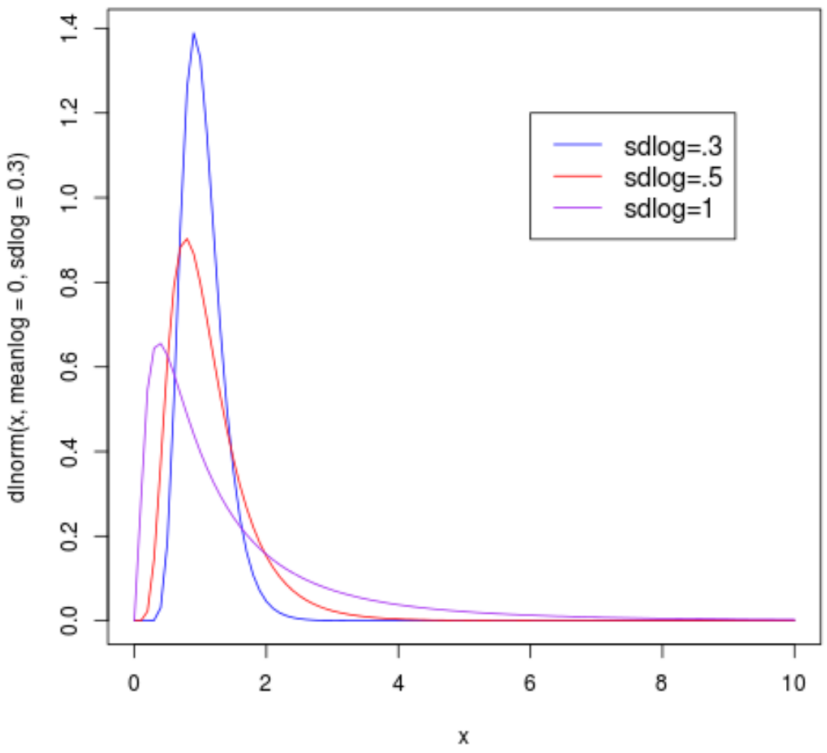
Several arguments are required to fully define the legend:

`x, y`: These coordinates specify the exact location of the legend's top-left corner on the graph. Alternatively, location keywords such as "topright" or "bottomleft" can be used for automatic placement.
`legend`: This is a character vector containing the text labels (descriptions) that will be displayed next to the corresponding line samples in the legend box.
`col`: This vector lists the colors of the lines being plotted. This list must be ordered precisely to match the colors and sequence of the labels defined in the `legend` argument.
`bg`: Designates the background color of the legend box itself. Setting this to "white" is standard practice for maximizing readability against the varying background tones of the plot area.
`lty`: Specifies the line type (e.g., 1 for solid, 2 for dashed) that will be displayed in the legend key.
`cex`: Controls the character expansion factor, which determines the overall size of the text within the legend box.

To finalize our comparative visualization, we integrate the legend into the existing plot. We position the legend box using the coordinates (6, 1.2) to ensure it is placed in an open area that does not obstruct the primary density curves. We link the descriptive labels ("sdlog=.3", "sdlog=.5", "sdlog=1") to their respective line colors (blue, red, purple) using the `legend` and `col` arguments, providing immediate clarity to the statistical visualization.

```
#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)
```


dlnorm(x, meanlog = 0, sdlog = 0.3)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
2
4
6
8
10
x
sdlog=.3
sdlog=.5
sdlog=1