

# How to Easily Create Kernel Density Plots in R

Authored by  
**stats writer**

December 2, 2025

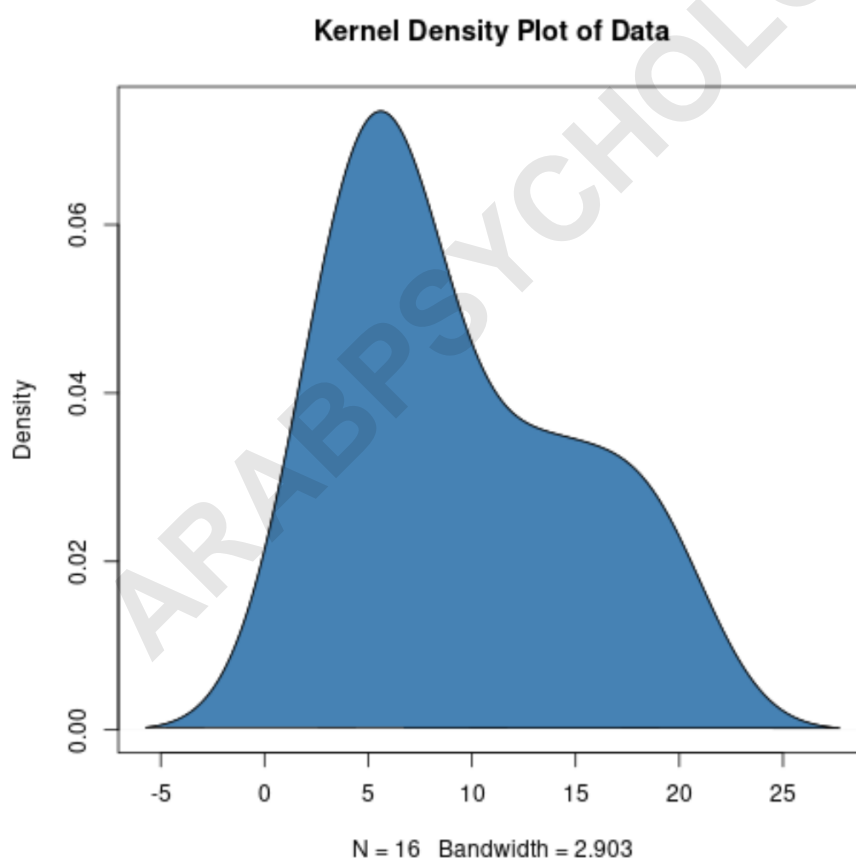
## RECOMMENDED CITATION

stats writer (2025). *How to Easily Create Kernel Density Plots in R*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=103720>

The ability to visualize data distributions is fundamental to statistical analysis and data science. Among the myriad of visualization tools available in R, the **kernel density plot** offers a superior method for understanding the underlying shape and characteristics of a dataset. These plots are essential for comparing how values are distributed across different groups or samples.

To successfully generate a kernel density plot in R, the process involves two primary steps: defining the density estimate using the robust `density()` function (part of the base R `stats` package), and then rendering the visualization using the `plot()` function. This guide provides comprehensive, step-by-step examples detailing how to achieve high-quality density visualizations, including customizing parameters such as the bandwidth and plotting multiple densities for direct comparison.

A **kernel density plot** is a non-parametric method used to estimate the probability density function of a random variable. It visualizes the distribution of values in a dataset using a single, smooth, continuous curve. Unlike discrete representations, the KDE curve provides an intuitive visual representation of where data points are concentrated and how they thin out.



While conceptually similar to a histogram, the kernel density plot is often preferred for its superior ability to display the true underlying shape of a statistical distribution. Crucially, the KDE curve is

not affected by the arbitrary choice of the number or width of bins, which can significantly skew the interpretation of a traditional histogram. This smoothness is achieved through the use of a kernel function applied across the data points.

## Understanding Kernel Density Estimation (KDE)

KDE works by placing a small, smooth function--the kernel--over each individual data point. These individual kernels are then summed up to create the final continuous density curve. The resulting plot shows the relative frequency or probability density across the range of values, allowing analysts to quickly identify modes (peaks) and potential skewness or multimodality within the dataset.

The two most critical parameters influencing the shape of the resulting KDE curve are the choice of the kernel function and the bandwidth. The default settings in R often utilize a Gaussian kernel and an automatically calculated bandwidth based on Silverman's or Scott's rule of thumb. However, advanced users should understand how to adjust these parameters to refine the estimation.

## The Role of Bandwidth and Kernel Selection

The bandwidth parameter is arguably the most influential factor in KDE. It controls the width of the smoothing kernel applied to each data point. A narrow bandwidth results in an undersmoothed plot that closely follows the individual data points (potentially showing spurious peaks), while a wide bandwidth results in an oversmoothed plot that may obscure important structural features of the underlying distribution. Experimentation is often necessary to find the optimal balance for visualization.

While the default Gaussian kernel is robust for most applications, R's `density()` function supports others, such as Epanechnikov, rectangular, triangular, and biweight kernels. Although the choice of the kernel usually has less impact than the bandwidth, different kernels can produce slightly different estimates, particularly in the tails of the distribution.

## Core Methods for Generating KDE Plots in R

We can use the following methods to create a kernel density plot in R, starting from the simplest visualization to more complex comparative plots:

### Method 1: Create One Basic Kernel Density Plot (Line Plot)

This method utilizes the `density()` function to calculate the KDE estimates for a single dataset, followed by the standard `plot()` command to render the result as a simple line graph.

### #define kernel density object using the density() function

**kd <- density(data)**

```
#create kernel density plot using the plot() function  
plot(kd)
```

**Method 2: Create an Enhanced Filled-In Kernel Density Plot**

To improve visual appeal and clarity, we can fill the area beneath the density curve. This is achieved by using the `polygon()` function immediately after plotting the density line, allowing for specific color customization.

**#define kernel density****kd <- density(data)**

```
#create kernel density plot (initial setup)  
plot(kd)
```

```
#fill in kernel density plot with specific color and black border  
polygon(kd, col='blue', border='black')
```

**Method 3: Create Multiple Kernel Density Plots for Comparison**

When comparing multiple datasets, it is necessary to plot the first density using `plot()` and subsequently add additional densities using the `lines()` function to overlay them on the same graph, distinguishing them by color and line thickness.

**#plot first kernel density plot (uses plot() to initialize graph)****kd1 <- density(data1)****plot(kd1, col='blue')**

```
#plot second kernel density plot (uses lines() to add to existing graph)
```

```
kd2 <- density(data2)
```

```
lines(kd2, col='red')
```

```
#plot third kernel density plot
```

```
kd3 <- density(data3)
```

```
lines(kd3, col='purple')
```

```
...
```

The following practical examples illustrate how to implement each method using sample data,

providing clear visualization of the concepts outlined above.

## Detailed Implementation: Single KDE Plot (Method 1)

This first example demonstrates the most straightforward way to generate a kernel density plot for a single, small dataset. We define the raw data vector, calculate the density object, and then use the `plot()` function to render the result, adding a descriptive title for clarity.

```
#create sample data vector
```

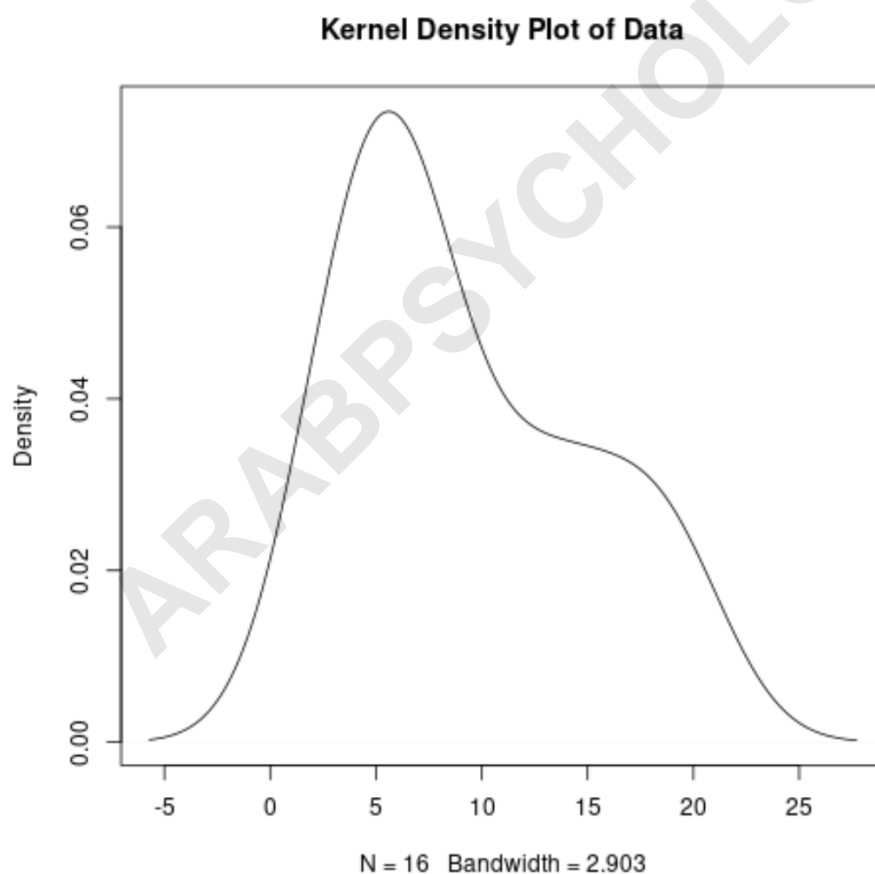
```
data <- c(3, 3, 4, 4, 5, 6, 7, 7, 7, 8, 12, 13, 14, 17, 19, 19)
```

```
#define kernel density object
```

```
kd <- density(data)
```

```
#create kernel density plot with custom title
```

```
plot(kd, main='Kernel Density Plot of Sample Data')
```



In this visualization, the **x-axis** represents the range of values within the dataset, while the **y-axis** indicates the estimated probability density. The height of the curve at any point reflects the relative

frequency of values occurring near that point. Observing the plot, we can see two distinct peaks, suggesting a potential bimodal distribution where values are highly concentrated around 7 and 19.

## Detailed Implementation: Aesthetic Enhancements (Method 2)

While the line plot provides the necessary information, filling the area under the curve significantly improves the visualization's aesthetic quality and ease of interpretation. Method 2 shows how to integrate the `polygon()` function to achieve this effect, using the calculated density object `kd` as the boundary for the shape.

```
#create sample data
```

```
data <- c(3, 3, 4, 4, 5, 6, 7, 7, 7, 8, 12, 13, 14, 17, 19, 19)
```

```
#define kernel density object
```

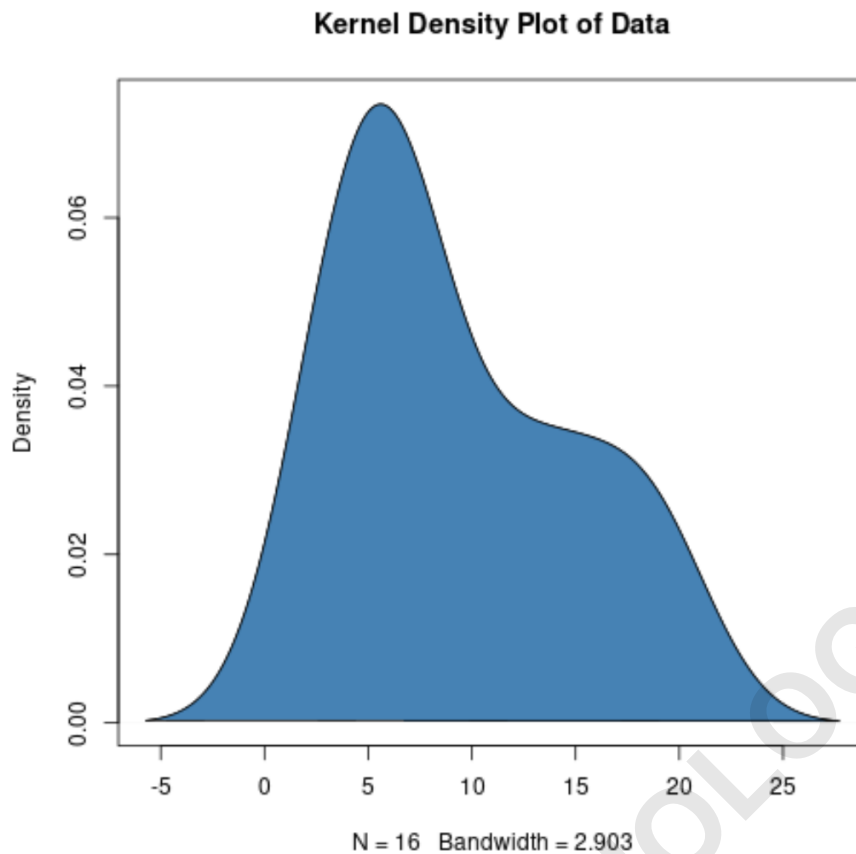
```
kd <- density(data)
```

```
#create kernel density plot (initializes axes without filling)
```

```
plot(kd)
```

```
#add color filling using polygon()
```

```
polygon(kd, col='steelblue', border='black')
```



Note that the `polygon()` function requires the density object (`kd`) as its primary input. The `col` argument specifies the fill color (here, 'steelblue'), and the `border` argument ensures the outline of the density curve remains visible and distinct. This technique is especially useful when presenting findings, as it emphasizes the continuous nature of the estimated probability function.

### Detailed Implementation: Comparative Analysis (Method 3)

One of the most powerful applications of kernel density plots is the direct, visual comparison of the distributions of two or more groups. This requires careful use of the base R plotting functions. We use `plot()` only for the first dataset (to set up the plotting area and axes) and then use `lines()` for all subsequent datasets, preventing the graph from being reset each time.

**#create two distinct datasets for comparison**

```
data1 <- c(3, 3, 4, 4, 5, 6, 7, 7, 7, 8, 12, 13, 14, 17, 19, 19)
```

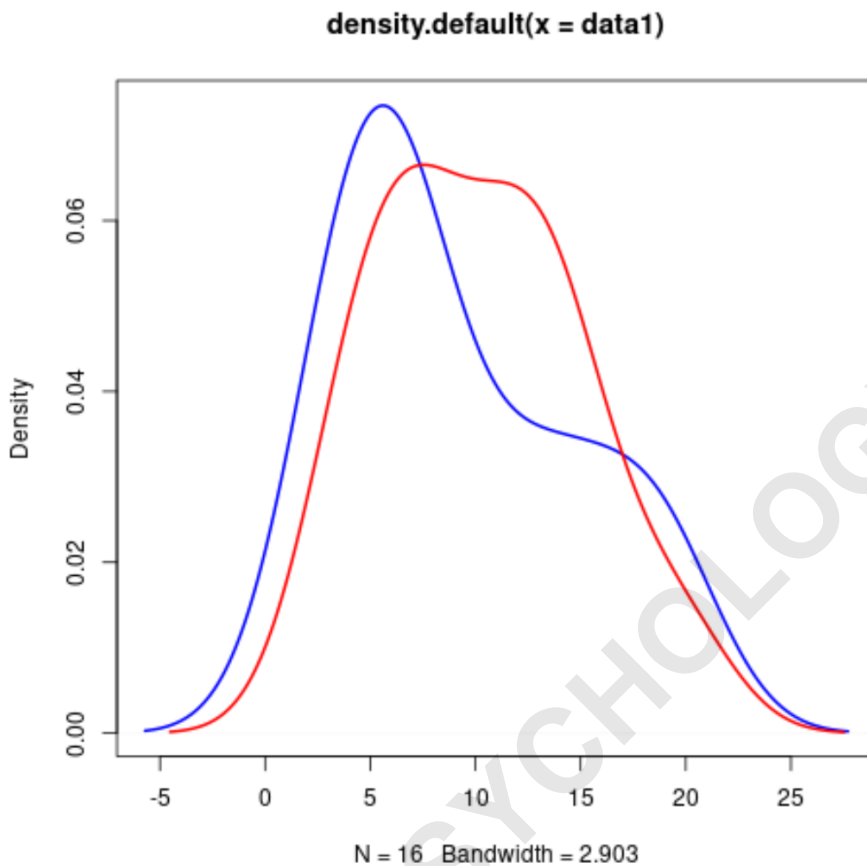
```
data2 <- c(12, 3, 14, 14, 4, 5, 6, 10, 14, 7, 7, 8, 10, 12, 17, 20)
```

**#Calculate and plot first density (data1)**

```
kd1 <- density(data1)
```

```
plot(kd1, col='blue', lwd=2, main="Comparative Kernel Density Plot")
```

```
#Calculate and plot second density (data2)
kd2 <- density(data2)
lines(kd2, col='red', lwd=2)
```



In this comparative visualization, the blue curve (Data 1) clearly shows a multimodal structure, while the red curve (Data 2) appears more centralized around a single peak, albeit with a broader spread. The use of the `lwd` argument (line width) enhances the visibility of the lines, making differentiation easier. This methodology is scalable; one can use the `lines()` function repeatedly to add as many kernel density plots as needed for complex, multi-group analyses on a single chart. It is generally recommended to also add a legend when comparing multiple groups to ensure clear identification.

### Advanced Customization of KDE Parameters

For more sophisticated analysis, the `density()` function provides several arguments that allow fine-tuning of the estimation process. The two most important parameters for customization are `kernel` and `bw` (for bandwidth).

**Kernel Selection:** To change the underlying smoothing function, use the `kernel` argument. For example, `density(data, kernel = "epanechnikov")` replaces the default Gaussian kernel with the Epanechnikov kernel, which is often considered optimal in a mean-square error sense.

**Bandwidth Adjustment:** To manually control the degree of smoothing, use the `bw` argument. If the default automatic bandwidth estimate seems too rough (undersmoothed), a larger numerical value for `bw` will result in a smoother curve: `density(data, bw = 2.5)`. Conversely, a smaller value will reveal more detail.

Understanding these parameters is key to generating plots that accurately reflect the underlying structure of the data without introducing visual artifacts due to over- or under-smoothing.

## Conclusion and Further Resources

Kernel density plots are an indispensable tool in the R visualization toolkit, offering a robust and aesthetically pleasing alternative to the traditional histogram for exploring data distributions. By mastering the `density()` and `plot()` functions, and utilizing enhancement techniques like coloring and overlaying multiple plots, users can create informative and publication-ready graphics for statistical reporting.

For those looking to deepen their R visualization skills, the following tutorials explain how to create other common plots and analyses:

Understanding the calculation of the bandwidth parameter in R.

Creating advanced visualizations using the `ggplot2` package, which offers more sophisticated control over KDE aesthetics.

Comparing continuous distributions using boxplots and violin plots.