Learning to Plot Chi-Square Distributions in R: A Step-by-Step Guide

Authored by

Mohammed looti

November 9, 2025

The ability to visualize complex statistical distributions is fundamental to effective data analysis and communication. The Chi-square distribution, a cornerstone of many hypothesis tests, particularly those involving variance and categorical data, is often essential to plot. Fortunately, the **R programming language** offers powerful, built-in functions that make generating a high-quality density plot straightforward, requiring only a few lines of code. This guide details the process of plotting the Chi-square distribution and demonstrates how to customize and shade critical regions for advanced statistical visualization.

Essential Functions for Chi-Square Plotting in R

To successfully render a density plot of the Chi-square distribution in R's base graphics system, we primarily rely on two core functions. These functions work in tandem: one calculates the necessary data points, and the other handles the efficient plotting of those points as a continuous curve. Understanding the purpose of each function is key to mastering the plotting process and adapting it for other distributions, such as the Normal or t-distribution.

The two primary functions we utilize are:

dchisq(): This function calculates the height of the probability density function (PDF) for the Chi-square distribution at specified x-values. It is crucial for determining the shape and scale of the curve based on the provided degrees of freedom.
curve(): Designed specifically for plotting mathematical functions over a defined range, **curve()** takes the output of a density function like **dchisq()** and translates it into a visual representation on the plot area. This function streamlines the process by avoiding the manual creation of large vectors of x and y coordinates.

The only mandatory parameter required to define the distribution itself is the **degrees of freedom** (df), which dictates the specific shape of the Chi-square curve. The **curve()** function then requires the plotting range, typically defined by the *to* and *from* parameters, to specify the segment of the distribution we wish to display on the x-axis.

Generating the Basic Chi-Square Density Plot

The shape of the Chi-square distribution is highly dependent on its degrees of freedom (df). As the degrees of freedom increase, the distribution shifts from being heavily skewed right toward a more symmetric, bell-like shape, eventually resembling the Normal distribution. For visualization purposes, it is standard practice to specify a range that captures the majority of the distribution's mass, often extending slightly into the right tail.

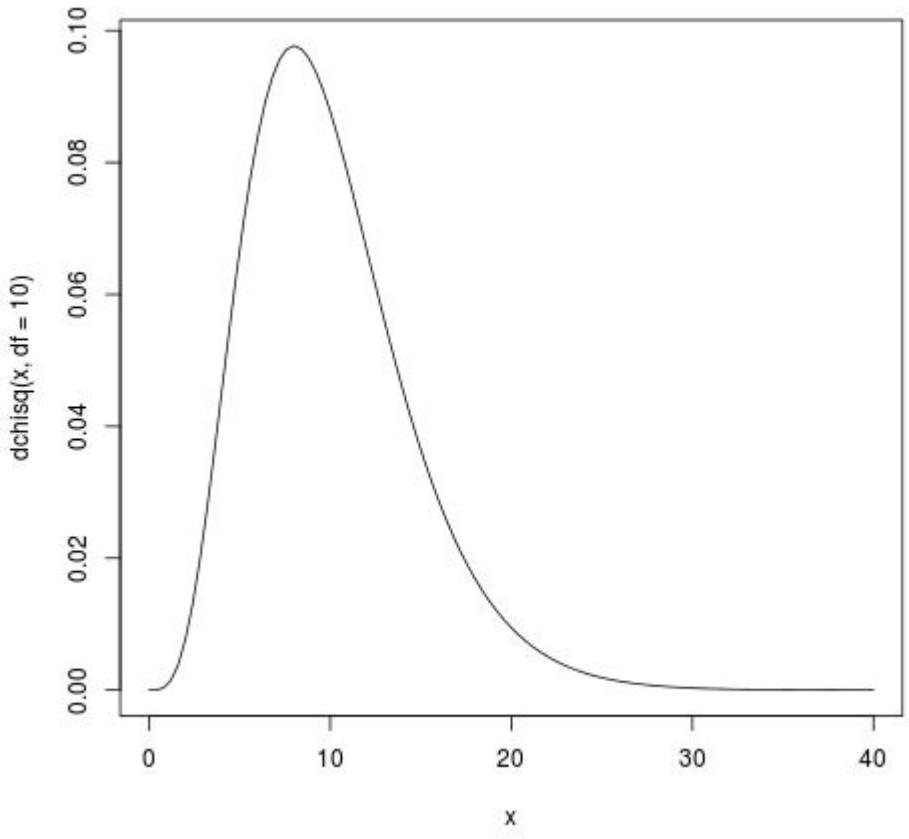
For instance, if we are analyzing a scenario involving 10 degrees of freedom, we must pass `df = 10` to the **dchisq()** function. To ensure the plot adequately displays the curve without excessive

empty space, we might choose an x-axis range from 0 (the starting point for all Chi-square distributions) up to 40. This range allows us to see how the density tapers off as values increase.

The following concise code block demonstrates the essential syntax required to generate a basic density plot for a Chi-square distribution with 10 degrees of freedom, ranging from x=0 to x=40:

curve(dchisq(x, df = 10), from = 0, to = 40)

Executing this command produces the initial visual output. While functional, the plot generated by the default settings typically lacks descriptive labels and aesthetic appeal, making the subsequent steps of modification essential for creating presentation-ready graphics.



Enhancing Visualization: Customizing Plot Aesthetics

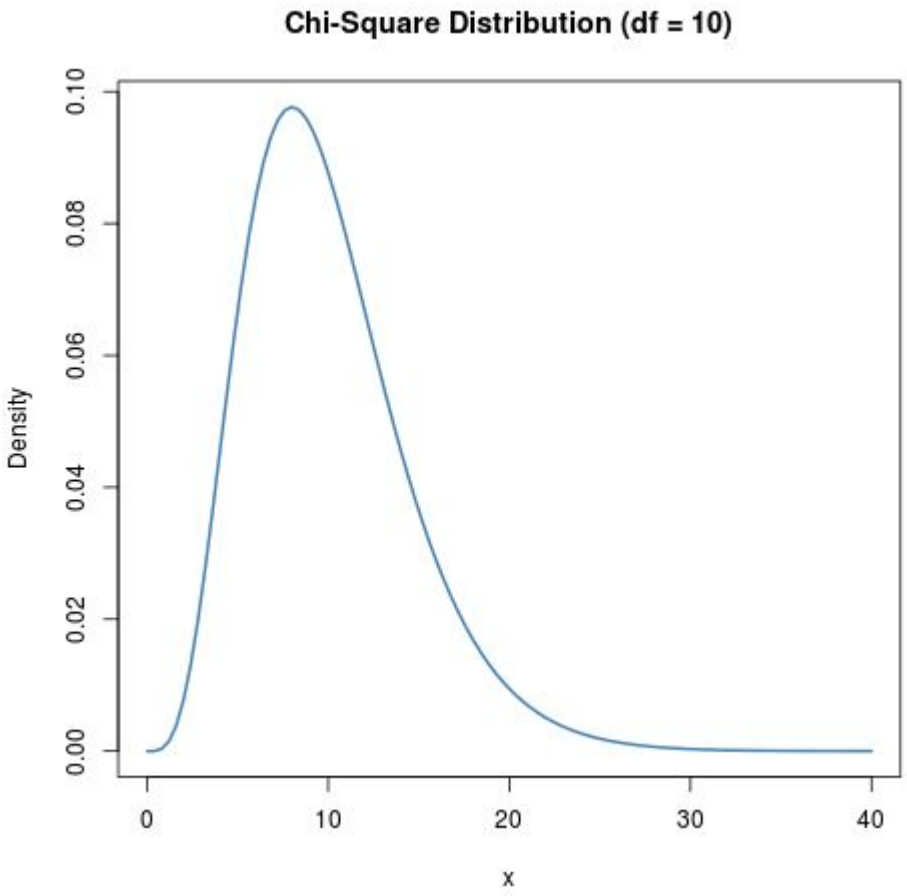
A basic density curve often needs further refinement to be useful in reports or publications. R's base plotting functions offer extensive graphical parameters that allow users to control nearly every element of the visualization. By integrating these parameters directly into the **curve()** function call, we can simultaneously define the plot content and enhance its appearance, improving clarity and professional quality.

Common modifications include adding a descriptive title, labeling the y-axis appropriately (since the plot represents density, not frequency), and adjusting the line properties to make the curve stand out. The `main` parameter sets the plot title, `ylab` controls the y-axis label, `lwd` (line width) increases the thickness of the line, and `col` allows the user to specify a precise line color, often using named colors or hexadecimal codes.

Here is the revised code illustrating how to apply these aesthetic enhancements to the Chi-square plot, making it clearer and more visually appealing:

curve(dchisq(x, df = 10), from = 0, to = 40,
main = 'Chi-Square Distribution (df = 10)', #add title
ylab = 'Density', #change y-axis label
lwd = 2, #increase line width to 2
col = 'steelblue') #change line color to steelblue

This modified approach ensures that the resulting visualization is immediately interpretable. By making deliberate choices regarding color and line weight, we guide the viewer's attention and provide essential context about the distribution being displayed.



Advanced Shading Techniques Using the polygon() Function

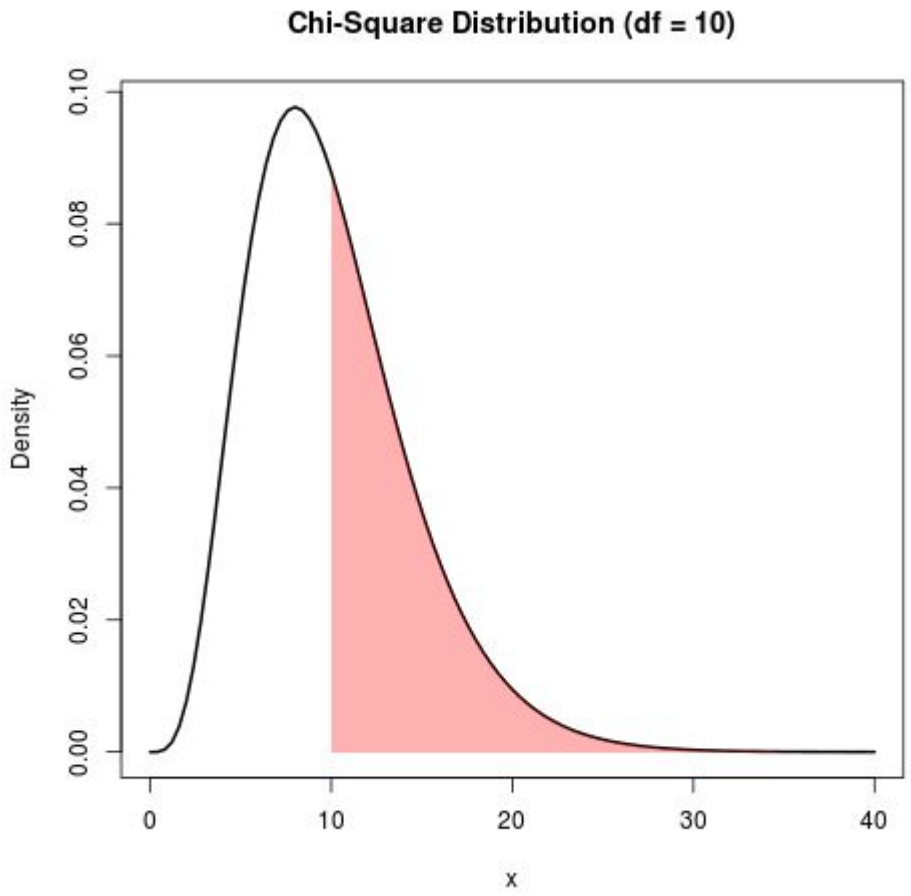
While the curve itself shows the theoretical probability density, statistical analyses often require highlighting specific regions, such as p-values or critical regions, which correspond to the area under the curve. To achieve this, we move beyond simple plotting and utilize the **polygon()** function. This function allows us to fill a closed shape defined by a set of coordinates, making it ideal for shading specific intervals on the density plot.

Shading a specific area requires a four-step process. First, the basic density curve must be drawn. Second, a sequence of x-values must be created for the region to be shaded. Third, the corresponding y-values (densities) for these x-values are calculated using **dchisq()**. Finally, the **polygon()** function uses these vectors to define the shaded region, typically closing the shape by appending a vector of reversed x-values and a vector of zeros (representing the baseline) to ensure the shape is properly closed off at the bottom.

For example, to visualize the upper tail of the distribution, perhaps representing a p-value, we can shade the area where x ranges from 10 to 40. This requires defining the x-vector using `seq(10, 40)` and calculating the corresponding densities:

```
#create density curve
curve(dchisq(x, df = 10), from = 0, to = 40,
main = 'Chi-Square Distribution (df = 10)',
ylab = 'Density',
lwd = 2)

#create vector of x values
x_vector <- seq(10, 40)

#create vector of chi-square density values
p_vector <- dchisq(x_vector, df = 10)

#fill in portion of the density plot from 0 to 40
polygon(c(x_vector, rev(x_vector)), c(p_vector, rep(0, length(p_vector))),
col = adjustcolor('red', alpha=0.3), border = NA)
```

The resulting plot clearly isolates the upper tail, allowing for an immediate visual assessment of the probability mass in that region. The `adjustcolor()` function is particularly useful here, as it allows us to add transparency (alpha=0.3), ensuring the curve line remains visible beneath the shading.



Visualizing Specific Tail Probabilities

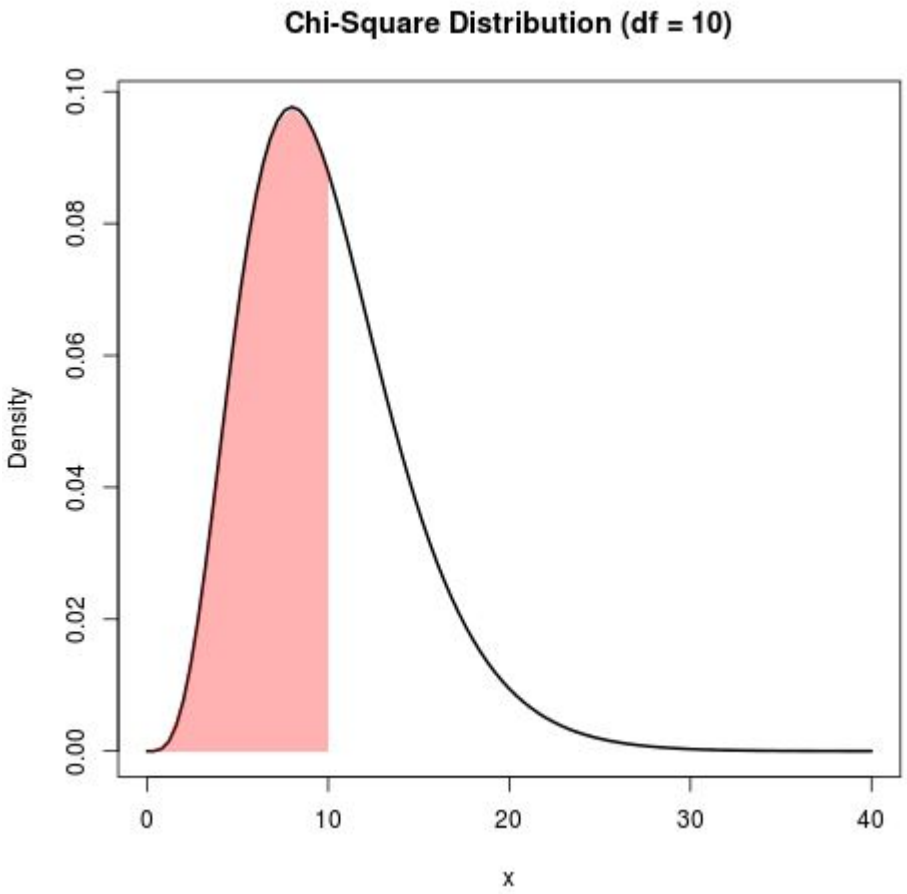
In addition to shading the upper tail, visualizing the lower tail is equally important, especially when dealing with two-tailed hypothesis tests or understanding the distribution's behavior near zero. The process for shading the lower tail is identical to the upper tail, but the sequence of x-values is adjusted to start at 0 and end at a specific point, such as x=10 in this demonstration. This highlights the region representing the smallest Chi-square values.

By defining the x-vector using `seq(0, 10)`, we instruct R to calculate the densities from the start of the distribution up to the specified boundary. This provides a clear contrast between the dense region near the peak of the curve and the tail area.

The following code block demonstrates how to shade the lower tail, specifically for x values ranging from 0 to 10, maintaining the same aesthetic parameters for consistency:

```
#create density curve
curve(dchisq(x, df = 10), from = 0, to = 40,
main = 'Chi-Square Distribution (df = 10)',
```

```
ylab = 'Density',
lwd = 2)

#create vector of x values
x_vector <- seq(0, 10)

#create vector of chi-square density values
p_vector <- dchisq(x_vector, df = 10)

#fill in portion of the density plot from 0 to 10
polygon(c(x_vector, rev(x_vector)), c(p_vector, rep(0, length(p_vector))),
col = adjustcolor('red', alpha=0.3), border = NA)
```



Highlighting Critical Regions: The Outer 5%

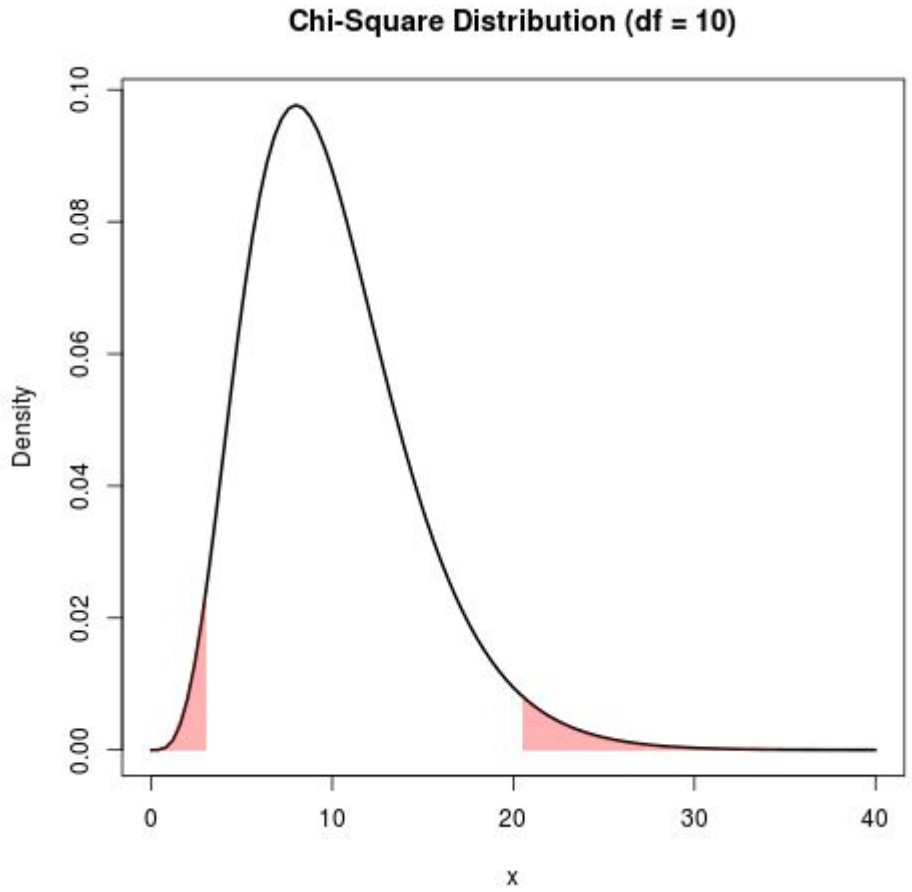
A frequent requirement in statistical analysis is visualizing the critical region--the area of the distribution that contains extreme values. For standard significance testing (e.g., alpha = 0.05), this often means shading the outer 5% of the distribution. Because the Chi-square distribution is

generally asymmetric, this critical region must be calculated using the quantile function, which determines the x-value corresponding to a given cumulative probability.

In R, the quantile function for the Chi-square distribution is **qchisq()**. To find the critical values defining the middle 95% (and thus isolating the outer 5%), we calculate the 2.5th percentile (`.025`) and the 97.5th percentile (`.975`). The regions outside these two points represent the extreme 5% of the distribution. The plotting process requires two separate **polygon()** calls--one for the lower tail (from 0 to `lower95`) and one for the upper tail (from `upper95` to the plot maximum).

This code illustrates how to dynamically find the critical values and shade the area representing the x values lying *outside* of the middle 95% of the distribution:

```
#create density curve
curve(dchisq(x, df = 10), from = 0, to = 40,
main = 'Chi-Square Distribution (df = 10)',
ylab = 'Density',
lwd = 2)

#find upper and lower values for middle 95% of distribution
lower95 <- qchisq(.025, 10)
upper95 <- qchisq(.975, 10)

#create vector of x values for lower tail
x_lower95 <- seq(0, lower95)

#create vector of chi-square density values
p_lower95 <- dchisq(x_lower95, df = 10)

#fill in portion of the density plot from 0 to lower 95% value
polygon(c(x_lower95, rev(x_lower95)), c(p_lower95, rep(0, length(p_lower95))),
col = adjustcolor('red', alpha=0.3), border = NA)

#create vector of x values for upper tail
x_upper95 <- seq(upper95, 40)

#create vector of chi-square density values
p_upper95 <- dchisq(x_upper95, df = 10)

#fill in portion of the density plot for upper 95% value to end of plot
polygon(c(x_upper95, rev(x_upper95)), c(p_upper95, rep(0, length(p_upper95))),
col = adjustcolor('red', alpha=0.3), border = NA)
```



Conversely, we may need to highlight the central 95% of the distribution, which represents the non-critical region or the area of least surprise. This is achieved by using the same `lower95` and `upper95` values found with **qchisq()**, but defining a single x-vector that spans between these two points.

```
#create density curve
curve(dchisq(x, df = 10), from = 0, to = 40,
main = 'Chi-Square Distribution (df = 10)',
ylab = 'Density',
lwd = 2)

#find upper and lower values for middle 95% of distribution
lower95 <- qchisq(.025, 10)
upper95 <- qchisq(.975, 10)

#create vector of x values spanning the middle 95%
x_vector <- seq(lower95, upper95)

#create vector of chi-square density values
p_vector <- dchisq(x_vector, df = 10)
```

```
#fill in density plot
polygon(c(x_vector, rev(x_vector)), c(p_vector, rep(0, length(p_vector))),
col = adjustcolor('red', alpha=0.3), border = NA)
```