

# How to Plot a Chi-Square Distribution in R: A Simple Guide

Authored by  
**stats writer**

March 3, 2026

## RECOMMENDED CITATION

stats writer (2026). *How to Plot a Chi-Square Distribution in R: A Simple Guide*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=133700>

## The Theoretical Foundation of the Chi-Square Distribution in Statistical Analysis

The **Chi-Square distribution** represents one of the most vital concepts in the field of **inferential statistics**. It is widely utilized to determine the relationship between categorical variables and to assess how well an observed set of data fits a specific theoretical model. In the **R programming language**, visualizing this distribution is a fundamental skill for researchers who need to perform **goodness-of-fit** tests or evaluate the independence of variables in a contingency table. By generating a clear graphical representation, analysts can better understand the behavior of their data relative to the expected **probability density function**.

The process of plotting this distribution begins with an understanding of its unique mathematical properties. Unlike the normal distribution, the **Chi-Square distribution** is typically skewed to the right, and its shape is entirely determined by a single parameter known as **degrees of freedom**. As the **degrees of freedom** increase, the distribution gradually begins to resemble a bell curve, illustrating the **central limit theorem** in action. Within the R environment, several built-in functions facilitate the transition from theoretical equations to high-quality visual outputs, allowing for both exploration and presentation of results.

To create a density plot for a **Chi-Square distribution** in **R**, we can use the following functions:

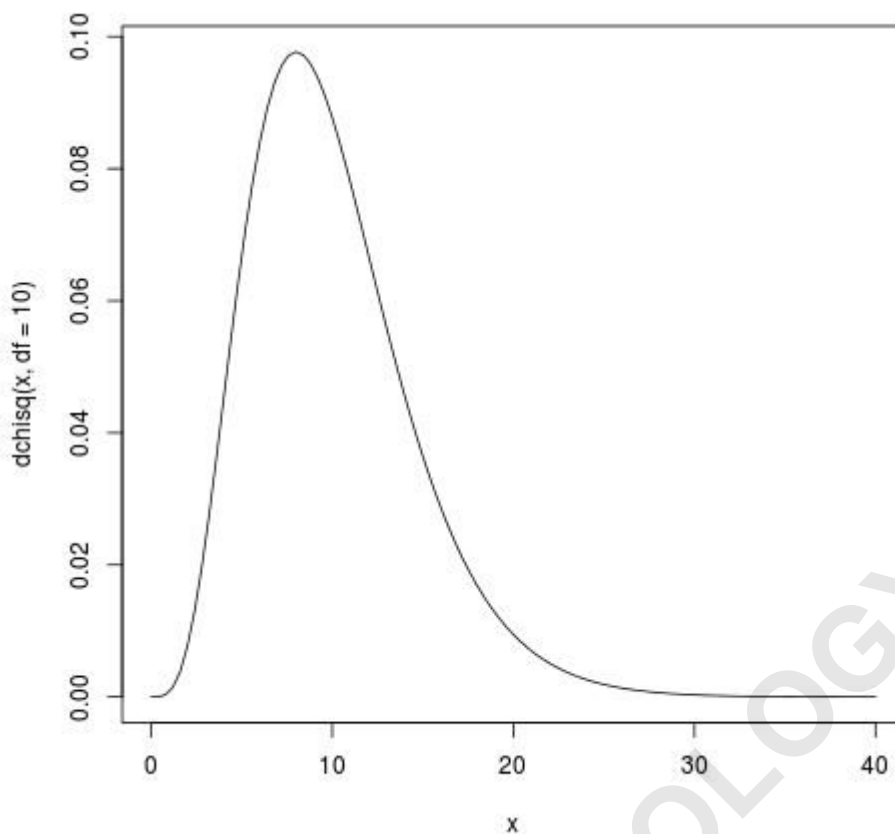
**dchisq()** to create the **probability density function curve()** to plot the **probability density function**

The initial step in this workflow involves defining the range of the x-axis and the specific parameters of the distribution. By using the `dchisq()` function, R calculates the density of the **Chi-Square distribution** at specific points, which serves as the y-coordinate for our plot. The `curve()` function then takes this mathematical definition and draws it across the specified interval. This approach is highly efficient because it does not require the manual generation of large datasets; rather, it evaluates the function dynamically, ensuring a smooth and accurate curve every time.

All we need to do to create the plot is specify the *degrees of freedom* for **dchisq()** along with the *to* and *from* points for **curve()**. This simplicity is what makes **R** a preferred choice for **data science** and academic research. Whether you are conducting a **Pearson's chi-squared test** or simply exploring the shape of different distributions, these core functions provide the necessary building blocks for robust statistical visualization.

For example, the following code illustrates how to create a density plot for a **Chi-Square distribution** with 10 **degrees of freedom** where the x-axis of the plot ranges from 0 to 40:

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



## Modifying the Density Plot for Professional Presentation

While a basic plot is sufficient for personal exploration, professional reporting often requires a higher level of customization. **R** offers an extensive array of graphical parameters that allow users to modify the appearance of their plots to improve clarity and aesthetic appeal. For instance, adding a descriptive title and modifying axis labels are essential steps for ensuring that the visualization is self-explanatory. This is particularly important when sharing results with stakeholders or publishing in academic journals, where precision and clarity are paramount.

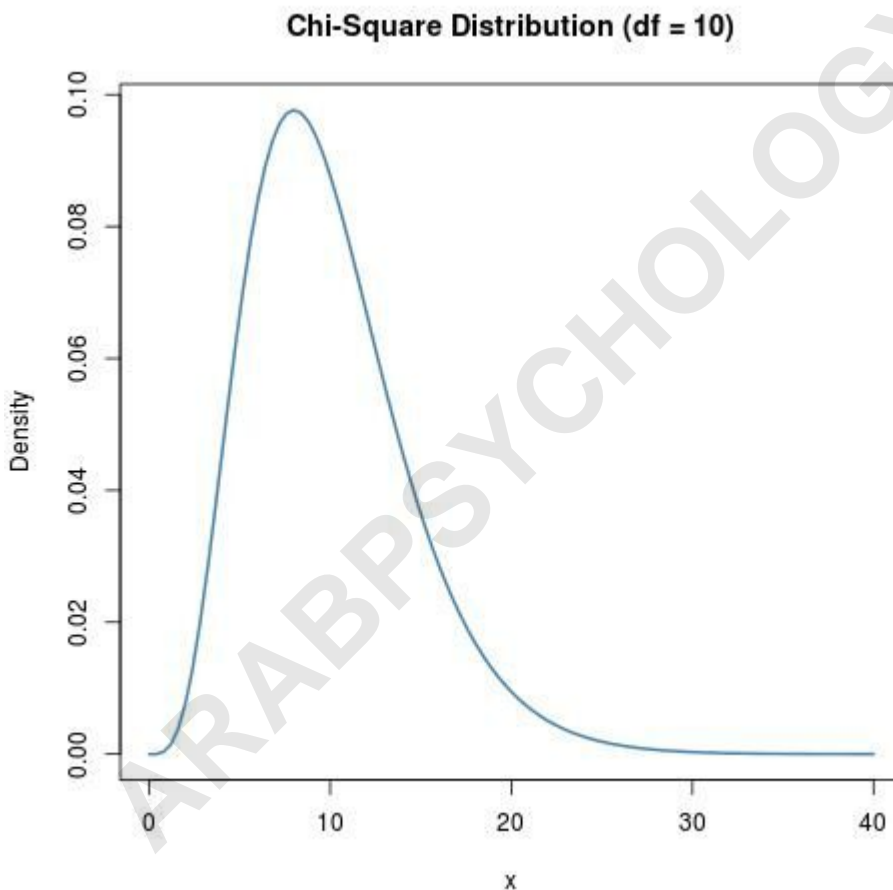
In addition to basic labeling, adjusting the physical properties of the curve can significantly impact the readability of the graph. By increasing the line width, the distribution curve becomes more prominent against the background grid. Changing the color of the line from the default black to a more distinct hue can also help in distinguishing the **Chi-Square distribution** when multiple curves are plotted on the same axes. These small adjustments transform a standard output into a publication-quality graphic that effectively communicates statistical insights.

The flexibility of the `curve()` function means that these modifications can be implemented directly within the function call using arguments like `main`, `ylab`, `lwd`, and `col`. This integrated approach allows for rapid iteration and refinement of the visual output. By experimenting with

different parameters, users can create a visualization style that aligns with their specific project requirements or organizational branding guidelines.

We can also modify the density plot by adding a title, changing the y-axis label, increasing the line width, and modifying the line color:

```
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
```



## Filling in the Density Plot to Highlight Specific Areas

In **statistical hypothesis testing**, it is often useful to highlight specific regions under the curve, such as the rejection region or the area corresponding to a specific **p-value**. Visualizing the area under the curve provides an intuitive sense of the cumulative probability associated with certain values of the test statistic. To achieve this in **R**, we utilize the ``polygon()`` function, which allows us

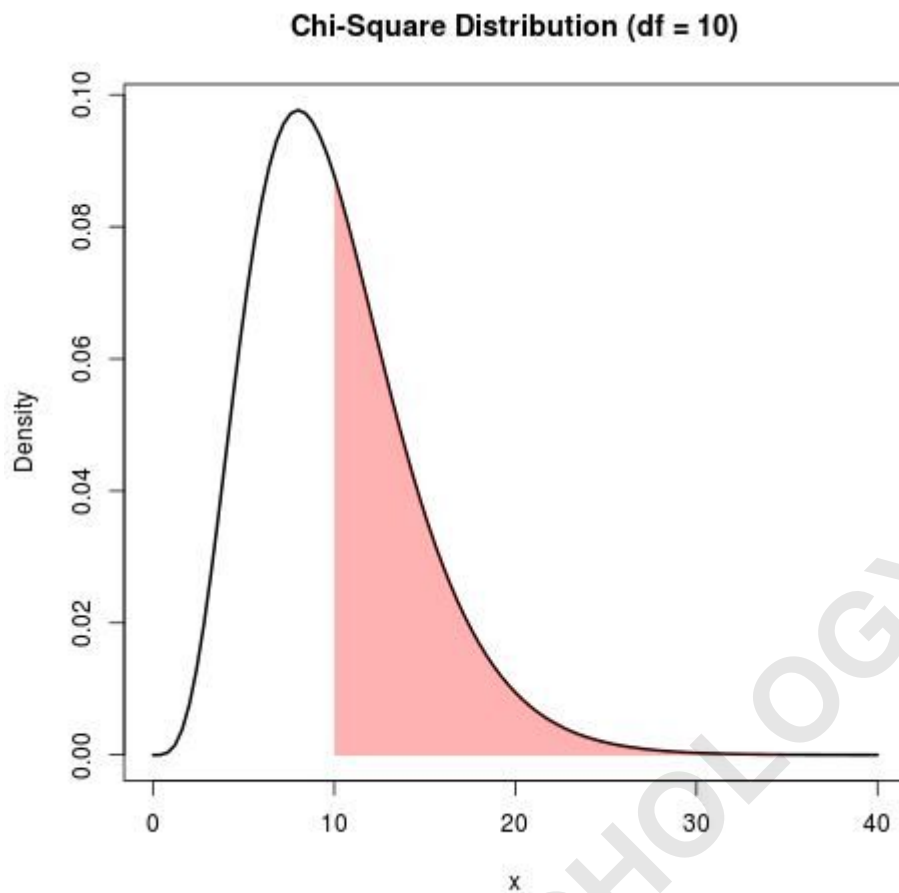
to fill in a specific area defined by a set of x and y coordinates.

The process of using ``polygon()`` involves creating a sequence of values along the x-axis and calculating their corresponding density values. These points define the upper boundary of the shaded area. By closing the shape along the x-axis (where y is zero), we create a filled region that represents the probability mass for that interval. This technique is invaluable for teaching statistical concepts, as it makes abstract ideas like "cumulative probability" and "area under the curve" visible and tangible.

In addition to creating the density plot, we can fill in part of the plot using the **polygon()** function based on a starting and ending value. This allows for a more detailed examination of the distribution's tail behavior or its central tendency. By adjusting the transparency of the fill color, typically through the ``adjustcolor()`` function, we can ensure that the underlying grid lines or other data points remain visible, maintaining the overall context of the graphic.

The following code illustrates how to fill in the portion of the density plot for the x values ranging from 10 to 40. This specific range covers the upper tail of the distribution, which is often the area of interest in right-tailed tests:

```
#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)
```



## Visualizing the Lower Tail of the Distribution

While the upper tail is frequently the focus of statistical inquiry, there are scenarios where visualizing the lower tail of the **Chi-Square distribution** is equally important. This might occur during diagnostic checks or when investigating specific lower-bound constraints in a model. By modifying the range of values passed to the `seq()` function, we can easily shift the focus of our visualization to the left side of the curve, highlighting the probability mass near zero.

The logic for shading the lower tail remains consistent with previous methods: define a **vector** of x-coordinates, calculate the corresponding density, and use the `polygon()` function to render the shape. This consistency in **R** syntax allows users to generalize their skills across different types of distributions and visualization tasks. Whether you are dealing with a Chi-Square, a **Normal**, or a **Student's t-distribution**, the workflow for shading areas remains largely the same.

The following code illustrates how to fill in the portion of the density plot for the x values ranging from 0 to 10. By focusing on this initial segment, we can see how much of the distribution's weight is concentrated at the lower end of the scale for a distribution with 10 **degrees of freedom**:

```
#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
```

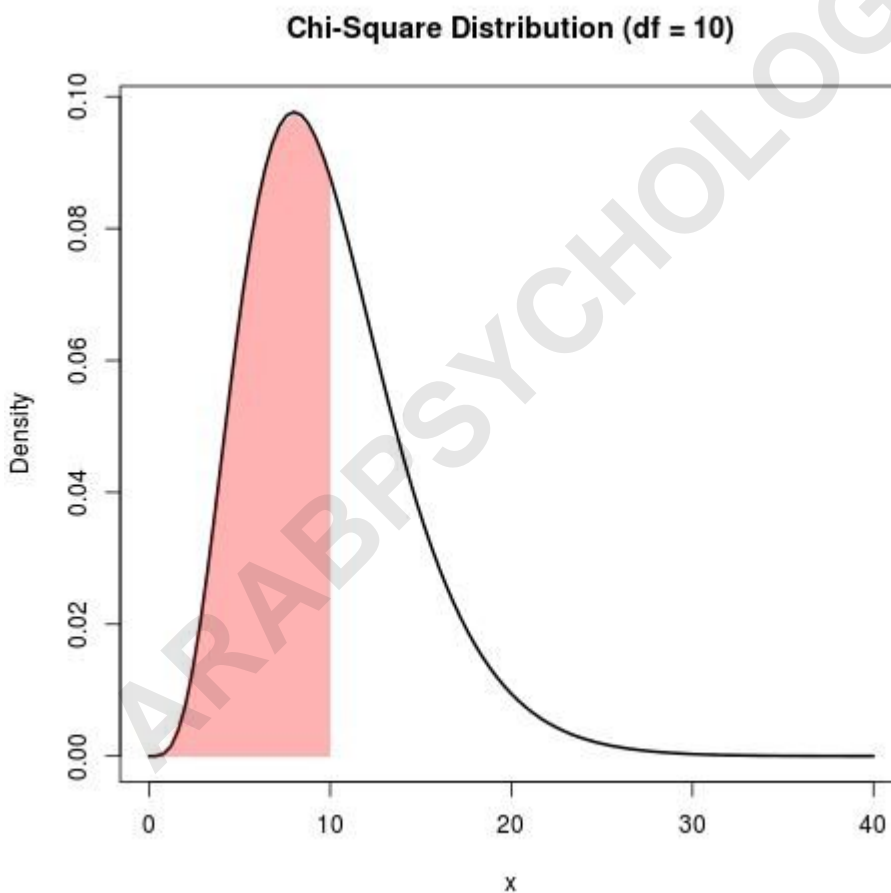
```
x_vector <- seq(0, 10)
```

```
#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 for Statistical Significance

One of the most powerful applications of distribution plotting is the visualization of **critical regions**. In a two-tailed test or when identifying outliers, we often want to shade the areas that lie outside

the central 95% of the distribution. These "tails" represent the extreme values that would lead us to reject the **null hypothesis** at a significance level of 0.05. Visualizing these regions helps in understanding the threshold at which a result becomes statistically significant.

To identify the exact cut-off points for these regions, we use the `qchisq()` function, which is the **quantile function** for the **Chi-Square distribution**. By providing the desired probabilities (e.g., 0.025 and 0.975 for a middle 95% range), R returns the corresponding values on the x-axis. These calculated values then serve as the boundaries for our shaded polygons. This automated approach ensures that the visualization is mathematically precise and directly linked to the theoretical thresholds of the test.

The following code illustrates how to fill in the portion of the density plot for the x values lying *outside* of the middle 95% of the distribution. This technique is particularly useful for demonstrating the concept of a **confidence interval** or for identifying the critical values in a **hypothesis test**:

```
#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
```

```
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
```

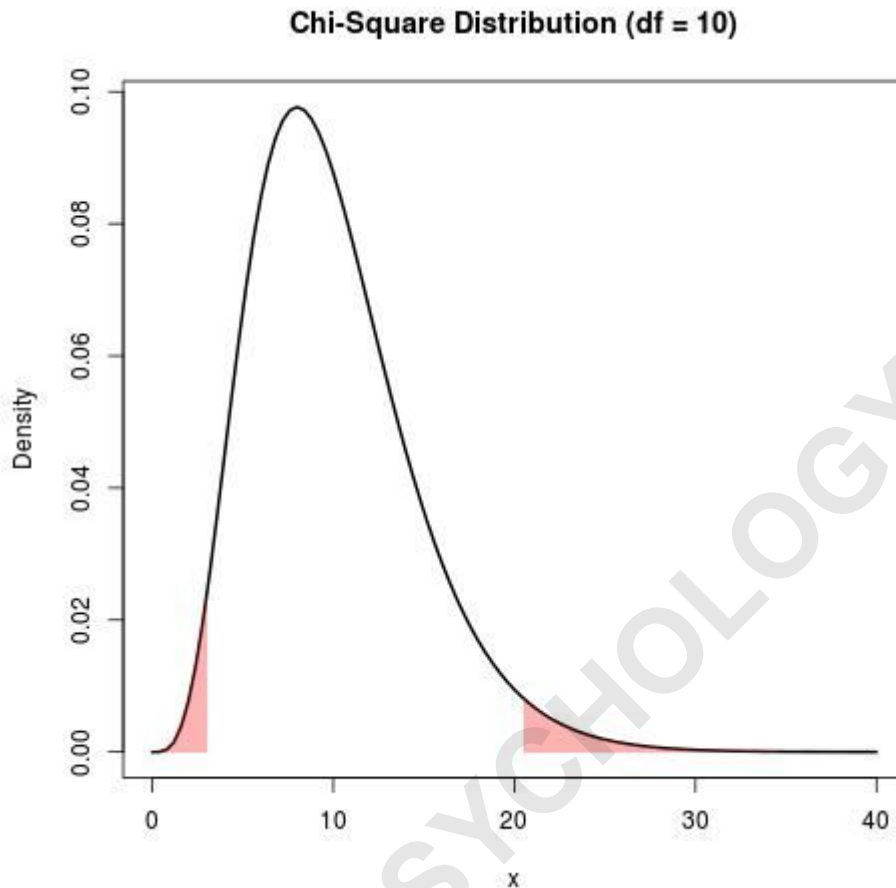
```
x_upper95 <- seq(upper95, 40)
```

```
#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)
```



## Visualizing the Central Confidence Interval

Conversely, there are times when the objective is to highlight the central portion of the distribution, representing the values where we are most likely to find the test statistic under the **null hypothesis**. Shading the middle 95% of the **Chi-Square distribution** provides a visual representation of the range in which we would fail to reject the null hypothesis. This "acceptance region" (though terminologically debated in some statistical circles) is a core component of how students and practitioners conceptualize statistical results.

Implementing this in **R** involves calculating the lower and upper quantiles and then creating a single sequence of values that spans the distance between them. This sequence is then used to generate a single large polygon that covers the main body of the distribution. By using a soft color or a specific alpha level, the user can create a professional-looking graphic that emphasizes the most probable outcomes of the statistical process.

Lastly, the following code illustrates how to fill in the portion of the density plot for the x values lying inside of the middle 95% of the distribution. This visualization is excellent for summarizing the results of an analysis and providing a clear, visual context for the calculated test statistics:

```
#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
```

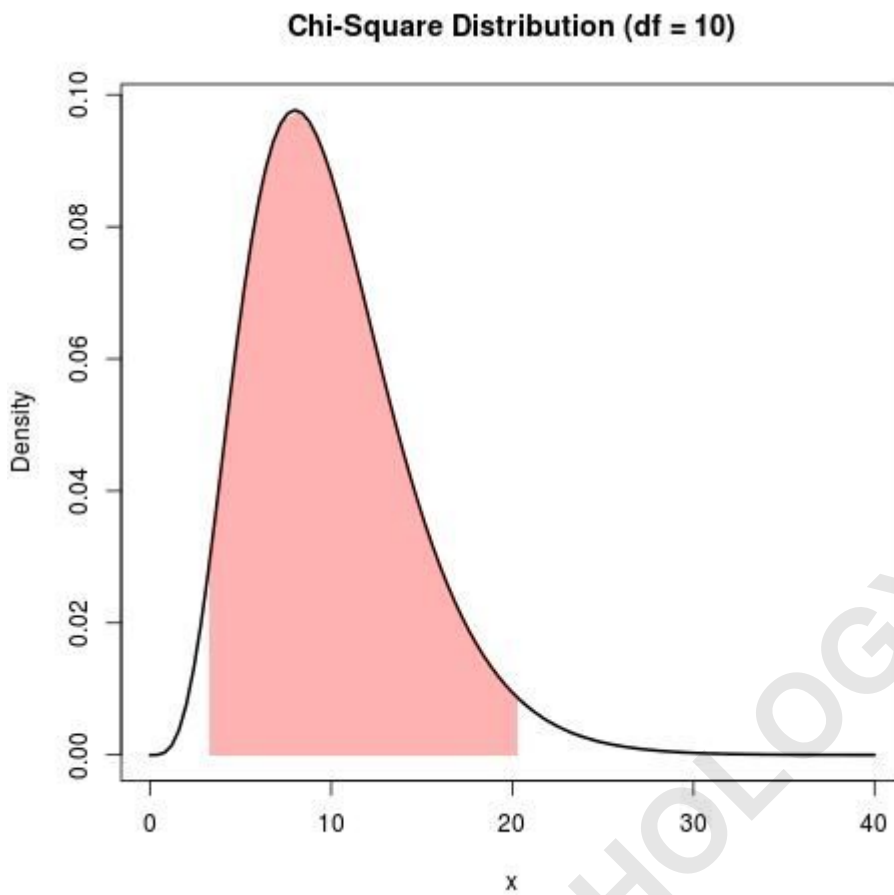
```
x_vector <- seq(lower95, upper95)
```

```
#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)
```



## Conclusion and Best Practices for Distribution Plotting

Plotting the **Chi-Square distribution** in **R** is a straightforward yet powerful process that significantly enhances the interpretability of statistical data. By moving beyond basic histograms and into theoretical density curves, analysts can compare their empirical findings against well-established mathematical models. The combination of ``curve()``, ``dchisq()``, and ``polygon()`` offers a flexible toolkit for creating informative and visually appealing graphics that cater to a wide range of analytical needs.

When creating these plots, it is important to remember that the primary goal is communication. Ensure that your **degrees of freedom** are clearly stated, as they define the entire shape of the distribution. Additionally, always consider the scale of your x-axis; setting the "to" and "from" points correctly ensures that the most relevant parts of the distribution are visible without excessive empty space. These best practices, combined with the technical skills outlined above, will allow you to produce high-quality statistical visualizations that stand up to rigorous academic and professional scrutiny.

As you continue to explore the capabilities of **R**, you will find that these visualization techniques are

applicable to nearly every statistical distribution. Mastering the art of shading and customizing density plots will not only improve your data storytelling but also deepen your own understanding of the underlying **probability theory**. Whether you are a student, a researcher, or a data professional, the ability to visualize the **Chi-Square distribution** is an essential arrow in your analytical quiver.

ARABPSYCHOLOGY.COM