Learning to Control Boxplot Outlier Display in R for Data Analysis

Authored by

Mohammed looti

November 7, 2025

In the realm of rigorous data visualization and statistical analysis, the precise control over graphical elements is paramount. A recurring requirement involves generating boxplots, where automatically calculated extreme values--known as outliers--may need to be deliberately suppressed. While these points hold significant analytical weight, their visual removal is often necessary to enhance clarity, especially when the goal is to focus the audience's attention squarely on the central distribution and spread of the data. Furthermore, professional publications frequently impose strict formatting guidelines that necessitate hiding these individual extreme data points. This in-depth tutorial provides precise, reproducible methods for suppressing the visual representation of these extreme values using R's two dominant statistical graphics frameworks: the foundational Base R plotting system and the highly sophisticated **ggplot2** package.

A serious R practitioner must be proficient in both methodologies. Base R offers exceptional speed and simplicity for rapid exploratory data analysis (EDA) and straightforward plotting tasks. Conversely, **ggplot2**, built on the principles of the grammar of graphics, provides unmatched granular control over aesthetics, layering, and plot composition, making it the industry standard for producing publication-quality graphics. We will systematically explore the specific arguments and logical steps required by each system to ensure clean, customized output that adheres to stringent data visualization best practices, ensuring that statistical integrity is maintained even as the visual presentation is refined.

Understanding How R Identifies Outliers

Before attempting any modification to a plot, it is essential to establish a clear understanding of the statistical definition R uses for identifying an outlier within a boxplot. Standard R plotting functions rely on the established Tukey method, which centers its definition around the concept of the Interquartile Range (IQR). The IQR represents the spread of the central 50% of the data, calculated as the difference between the third quartile (Q3, the 75th percentile) and the first quartile (Q1, the 25th percentile).

The boundaries, often referred to as 'fences' or 'whiskers,' are mathematically defined. Any observation that falls outside these fences is statistically flagged as an outlier. Specifically, the fences are set at 1.5 times the IQR added to Q3 or subtracted from Q1. This provides a robust, standardized criterion for identifying extreme values that are disproportionately far from the central mass of the data distribution.

The exact criteria for an observation to be designated as an outlier are:

Less than Q1 - (1.5 * IQR)
Greater than Q3 + (1.5 * IQR)

In default visualizations, these extreme points are plotted individually, usually as circles or

asterisks, extending beyond the boxplot's whiskers. It is crucial to emphasize that when we instruct R to "remove" these outliers visually, we are not deleting them from the actual underlying dataset used for calculation. Instead, we are manipulating the plotting function's visual output, focusing the viewer's attention solely on the central tendency, spread, and the non-extreme whiskers. This distinction is vital for maintaining data integrity while producing tailored graphical representations.

Removing Outliers in Boxplots Using Base R Graphics

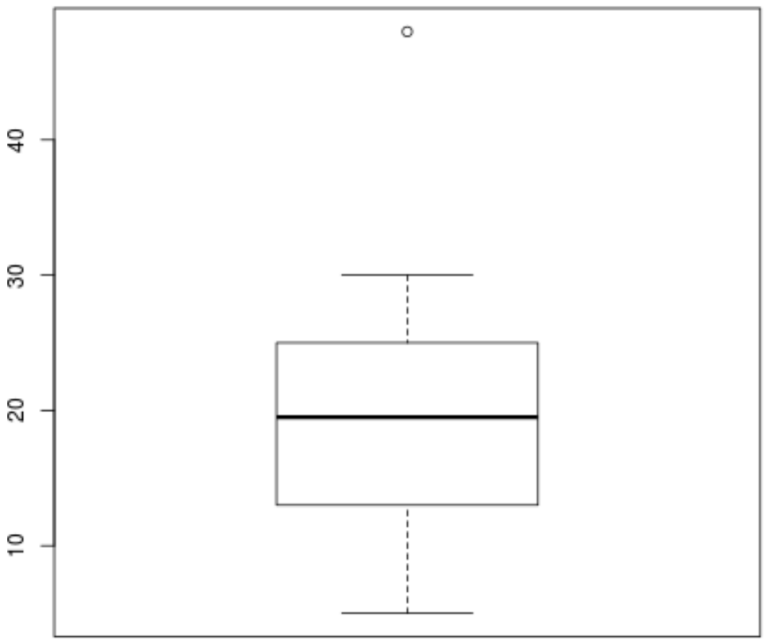
The Base R environment provides the fundamental `boxplot()` function, which is efficient and straightforward for generating quick statistical visualizations. To illustrate the exact process of suppressing outliers, we will begin by defining a simple numeric vector containing a known extreme value. This consistency allows us to compare the behavior of both Base R and ggplot2 accurately.

We define the following dataset, where the value 48 is intentionally included to serve as an obvious outlier based on the standard IQR calculation:

data <- c(5, 8, 8, 12, 14, 15, 16, 19, 20, 22, 24, 25, 25, 26, 30, 48)

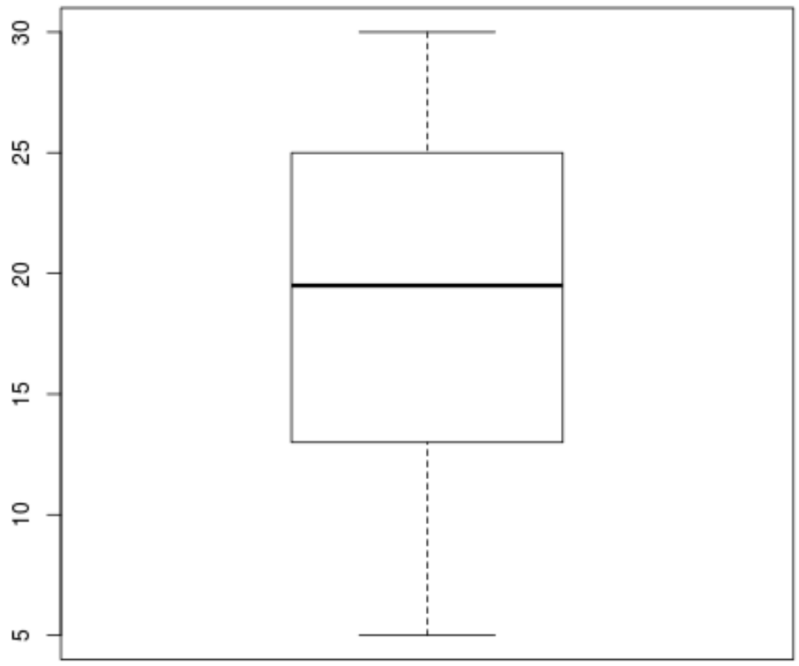
To produce the default boxplot in Base R, which includes the visual marker for every identified outlier point, we execute the primary function call:

boxplot(data)



The resulting plot clearly displays the value 48 as a distinct circle positioned high above the upper whisker. The most direct and simple mechanism provided by the `boxplot()` function to hide this outlier is the dedicated logical argument: `outline`. By setting `outline` to `FALSE`, we explicitly instruct the plotting function to suppress the plotting of any symbol representing a data point that extends beyond the statistically defined whisker limits. This is the most efficient method available in the Base R graphics system.

boxplot(data, outline=FALSE)



Upon viewing the updated plot, we confirm that the outlier marker has been successfully removed. A key feature of the Base R implementation is its automatic axis adjustment: the y-axis dynamically rescales itself to the maximum value of the upper whisker (the highest non-outlier observation). This auto-scaling behavior ensures optimal plot space utilization and is a significant convenience when utilizing Base R for rapid outlier suppression.

Advanced Visualization: Handling Outliers with ggplot2

When the visualization requirements extend beyond simple defaults, demanding sophisticated customization, layering, and precise aesthetic control, the **ggplot2** package--a cornerstone of the **tidyverse**--becomes the tool of choice. **ggplot2** adheres to the philosophy of the grammar of graphics, necessitating that data be structured within a data frame and plots be constructed sequentially through layers. Consequently, the method for suppressing outliers here is fundamentally different and more nuanced than the straightforward logical switch used in Base R.

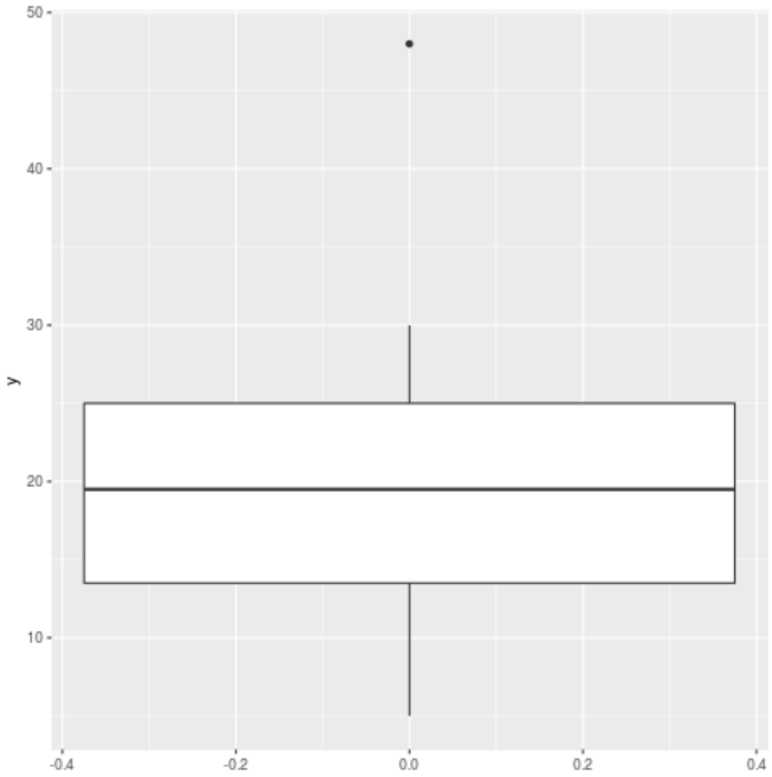
First, we must prepare our original numeric vector by converting it into a data frame structure suitable for **ggplot2**, ensuring the variable is correctly labeled for aesthetic mapping:

data <- data.frame(y=c(5, 8, 8, 12, 14, 15, 16, 19, 20, 22, 24, 25, 25, 26, 30, 48))

To generate the default boxplot using **ggplot2**, we load the necessary library and utilize the `geom_boxplot()` geometry, mapping our variable `y` to the vertical axis:

library(ggplot2)

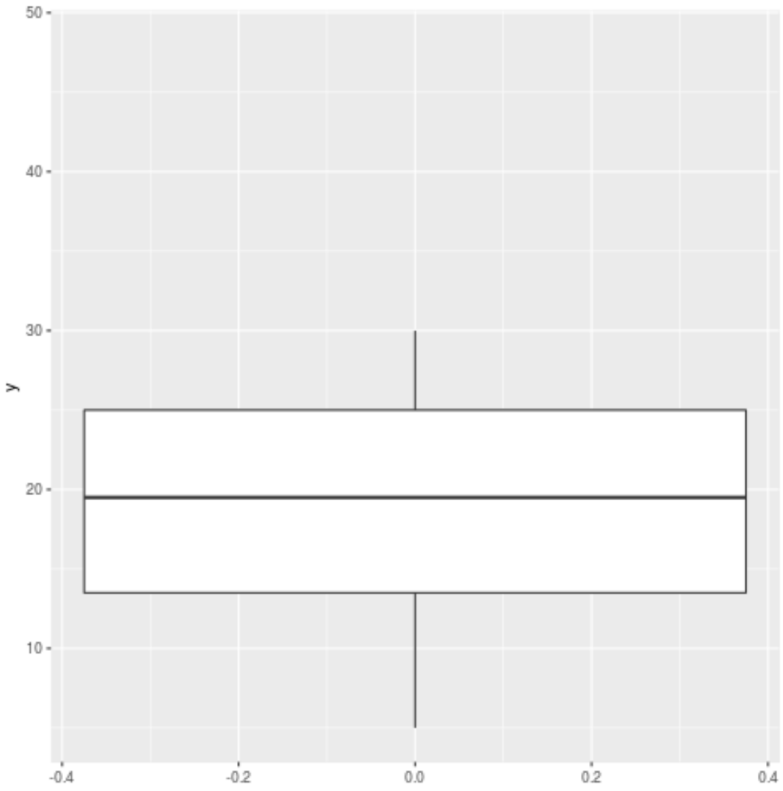
ggplot(data, aes(y=y)) +
geom_boxplot()



In **ggplot2**, the outliers are rendered as geometric shapes defined within the `geom_boxplot()` layer. To effectively hide them, we do not use a global plot setting; instead, we target the specific aesthetic property responsible for plotting the outlier shape. This is achieved by setting the `outlier.shape` argument to `NA` (Not Available). This action instructs the geometry layer not to draw the points, thereby making the outliers invisible without altering the statistical basis of the box, whiskers, or quartiles.

ggplot(data, aes(y=y)) +

geom_boxplot(outlier.shape = NA)



While the visual marker for the outlier is now suppressed, a critical distinction from Base R emerges: **ggplot2** does not automatically adjust the y-axis limits. As seen in the resulting plot, the axis still extends to 48, which was the maximum value of the original dataset, resulting in a large, visually distracting blank space above the upper whisker. This occurs because the axis scaling is calculated based on the full range of the data variable mapped to the axis, and simply hiding a shape (`outlier.shape = NA`) does not modify that underlying data range. To achieve a truly clean visualization, manual adjustment of the plot viewport is required.

Addressing the ggplot2 Axis Challenge with coord_cartesian

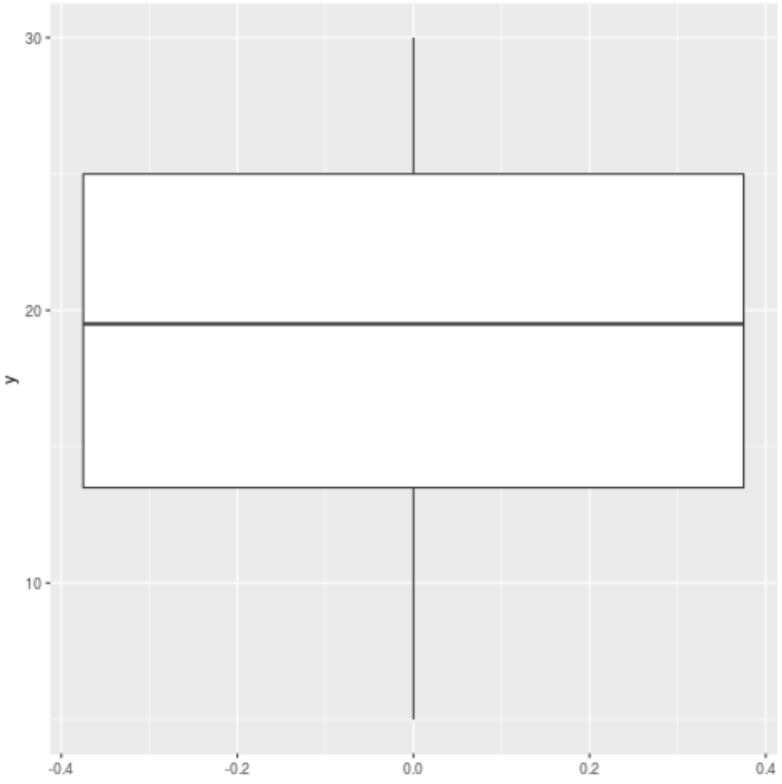
When hiding outliers using the `outlier.shape = NA` technique in **ggplot2**, the next mandatory step is to refine the plot's viewable area to eliminate the empty space created by the hidden points. We achieve this by adding the coordinate system layer, specifically `coord_cartesian()`.

Understanding the technical distinction between coordinate cropping and data filtering is vital in **ggplot2**. If a user were to attempt to set limits using `scale_y_continuous(limits=...)`, **ggplot2** would filter out all data points outside that range *before* the boxplot statistics (Q1, Median, Q3, Whiskers) are calculated. This would fundamentally alter the statistical output, potentially changing

the box's definition. Conversely, `coord_cartesian(ylim=...)` acts as a true "zoom" function, applied *after* all statistical calculations have been completed. This preserves the integrity of the box and whisker definitions while simply visually cropping the resulting plot space to the desired range.

To generate a final, professional plot focused only on the non-outlier range (approximately 5 to 30 in our example), we append the `coord_cartesian` layer to the visualization pipeline. We precisely define the new vertical limits using the `ylim` argument, ensuring the view extends only up to the upper whisker limit:

ggplot(data, aes(y=y)) +
geom_boxplot(outlier.shape = NA) +
coord_cartesian(ylim=c(5, 30))



The resulting visualization is now optimized. The y-axis ranges perfectly from 5 to 30, successfully eliminating the substantial blank space caused by the hidden outlier. This critical two-step sequence--setting `outlier.shape = NA` to hide the marker and then employing `coord_cartesian` to refine the view--constitutes the standard, best-practice methodology in **ggplot2** for achieving a visually clean boxplot while preserving underlying statistical calculations.

Summary of Visualization Techniques

We have thoroughly examined two powerful, yet methodologically distinct, approaches for controlling the visual output of extreme values in R boxplots. The decision between leveraging Base R and embracing **ggplot2** generally hinges on the complexity required by the final graphic and the user's need for detailed customization.

The methodology employing Base R is characterized by its remarkable efficiency and simplicity, requiring only the inclusion of `outline=FALSE` within the `boxplot()` call. A notable advantage here is the automatic adjustment of the axis limits, which streamlines the process for quick, high-speed exploratory graphics. However, this simplicity comes at the cost of reduced overall aesthetic and structural customization capabilities.

In contrast, the **ggplot2** methodology, although requiring a two-step process--hiding the shape (`outlier.shape = NA`) and then zooming the view (`coord_cartesian`)--offers unparalleled explicit control over every graphical element. This structured approach is essential because the grammar of graphics deliberately separates data mapping from visual rendering and coordinate systems. This separation ensures that the statistical calculations derived from the complete dataset remain uncompromised, while the visual output is precisely tailored. Mastering the correct application of `coord_cartesian` is absolutely paramount in **ggplot2** to prevent inadvertently filtering the dataset, which would corrupt the calculated quartiles and median.

Additional Resources for R Visualization

To continue developing expertise in high-quality data visualization using **ggplot2**, we recommend exploring resources that detail essential plot modifications, such as setting highly precise axis limits and arranging multiple complex plots for comparative analysis.

How to Set Axis Limits in ggplot2
How to Create Side-by-Side Plots in ggplot2