# How to Create Multiple Boxplots in a Single Plot in R

Authored by

**stats writer**

March 2, 2026

## Understanding the Boxplot and its Statistical Foundations

In the expansive field of **exploratory data analysis**, the **boxplot**, frequently referred to as a box-and-whisker plot, serves as a fundamental tool for visualizing the distribution of a **dataset**. This graphical representation is particularly adept at highlighting the dispersion, central tendency, and potential **outliers** within a sample without making assumptions about the underlying statistical distribution. By condensing complex data into a simplified visual format, researchers can quickly discern whether a variable is symmetric, skewed, or contains unusual values that warrant further investigation.

The primary utility of a **boxplot** lies in its ability to facilitate comparisons across different categories or groups. When a single plot contains multiple boxplots, it allows the analyst to observe shifts in the **median** or variations in the spread of data across distinct levels of a categorical variable. This comparative capacity is essential in fields ranging from environmental science to economics, where understanding the variance between groups is often as important as understanding the average value itself. Within the **R language** ecosystem, creating these visualizations is both efficient and highly customizable.

R provides several methodologies for generating these plots, catering to different user needs and aesthetic requirements. The core **R language** provides a robust **base graphics system** that is known for its speed and simplicity. For those seeking more sophisticated and publication-quality graphics, the **ggplot2** package offers a comprehensive framework based on the **Grammar of Graphics**. This tutorial will delve into both approaches, ensuring a comprehensive understanding of how to manage multiple boxplots effectively within a single charting area.

Beyond simple visualization, the boxplot provides a window into the **interquartile range** (IQR), which represents the middle 50% of the data. The "whiskers" of the plot typically extend to 1.5 times the IQR, providing a standardized method for identifying extreme values. As we explore the implementation in R, it is important to keep these statistical underpinnings in mind, as they inform how the functions interpret the **vectors** and **data frames** passed to them for processing.

## The Five-Number Summary and Data Interpretation

The visual structure of a boxplot is directly derived from what statisticians call the **five-number summary**. This summary provides a concise overview of the distribution's shape and scale by focusing on five critical values. By understanding these components, an analyst can interpret the boxplot's box, median line, and whiskers with high precision. In **R language**, these values are computed automatically when invoking the plotting functions, though they can also be extracted manually using the `summary()` or `fivenum()` functions.

The **five-number summary** consists of the following elements:

**The minimum value:** The lowest data point in the set, excluding any designated **outliers**.
**The first quartile (Q1):** The value at the 25th percentile, representing the boundary below which 25% of the data falls.
**The median value (Q2):** The central point of the dataset, dividing it into two equal halves.
**The third quartile (Q3):** The value at the 75th percentile, representing the boundary below which 75% of the data falls.
**The maximum value:** The highest data point in the set, excluding any designated **outliers**.

Each of these components plays a vital role in the final visual output. The "box" itself is defined by the first and third **quartiles**, with the **median** marked by a horizontal line inside the box. The length of this box indicates the variability of the middle half of the data. If the median is not centered within the box, it suggests that the distribution is skewed. For instance, a median closer to the bottom of the box indicates a positive or right-skewed distribution, whereas a median closer to the top suggests a negative or left-skewed distribution.

Furthermore, the "whiskers" extend from the box to the minimum and maximum values, providing a sense of the total range of the data. In many implementations, including those in **R language**, any data point that falls significantly outside the range of the whiskers is plotted individually as a dot or circle. These **outliers** are critical for data cleaning and for identifying anomalies in experimental results or financial datasets. Understanding how to interpret these symbols is the first step toward mastering the use of multiple boxplots in comparative analysis.

## Preparing the Environment and Exploring the Dataset

Before constructing our visualizations, we must first establish a suitable environment and select a **dataset** that illustrates the utility of grouped boxplots. For this tutorial, we will utilize the **airquality** dataset, which is a standard **built-in dataset** in R. This dataset contains daily air quality measurements in New York from May to September 1973, including variables such as ozone levels, solar radiation, wind speed, and temperature. Because it includes both continuous numerical data and categorical time variables (months), it is an ideal candidate for demonstrating how to group boxplots.

To begin the analysis, we first examine the structure and initial entries of the **data frame**. This step is crucial in any data science workflow to ensure the **syntax** used in the plotting functions correctly references the column names. We can use the `head()` function to view the first six rows, providing a snapshot of the data we will be working with. By inspecting the data, we can also identify any missing values (NAs) that might affect the resulting **boxplot**.

**#view first 6 rows of "airquality" dataset**
**head(airquality)**

```
# Ozone Solar.R Wind Temp Month Day
#1 41 190 7.4 67 5 1
#2 36 118 8.0 72 5 2
#3 12 149 12.6 74 5 3
#4 18 313 11.5 62 5 4
#5 NA NA 14.3 56 5 5
#6 28 NA 14.9 66 5 6
```

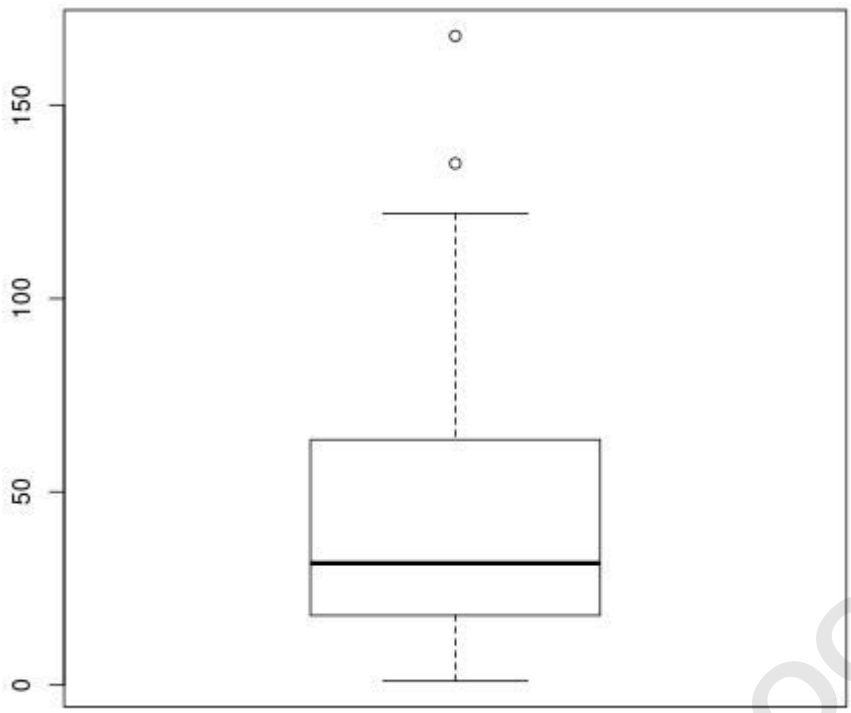
As observed in the output above, the dataset contains several numeric variables. To create a single **boxplot** for a specific variable, such as "Ozone," we would typically pass that specific **vector** to the plotting function. However, our goal is to expand this into a multi-boxplot chart. Notice that the "Month" column is represented by integers (5 through 9). When plotting in R, we often treat these integers as categorical factors to ensure the software groups the data correctly rather than treating the month as a continuous scale.

## Implementing a Single Boxplot Using Base R

The most straightforward method to visualize a distribution in the **R language** is by using the `boxplot()` function from the **base graphics** package. This function is highly versatile, accepting a variety of input types including numeric vectors, **lists**, and formulas. To generate a basic boxplot for the "Ozone" variable, we use the dollar sign **operator** to select the specific column from the **airquality data frame**.

```
#create boxplot for the variable "Ozone"
boxplot(airquality$Ozone)
```

The command above generates a vertical boxplot that provides a visual summary of the ozone concentrations across the entire measurement period. By default, R will handle missing values by omitting them, though this behavior can be adjusted using specific function **parameters**. This initial plot is useful for a quick check of the overall data distribution, but it lacks the context provided by comparative grouping and detailed labeling.






While this basic visualization is functional, it often serves as a precursor to more complex charts. In a real-world analysis, seeing the global distribution of ozone is rarely enough; researchers typically want to know how that distribution changes over time or under different conditions. This leads us to the necessity of creating multiple boxplots within a single frame, a task where R's formula **syntax** excels by allowing us to define a dependent variable and a grouping independent variable.

## Creating Grouped Boxplots with Formula Notation

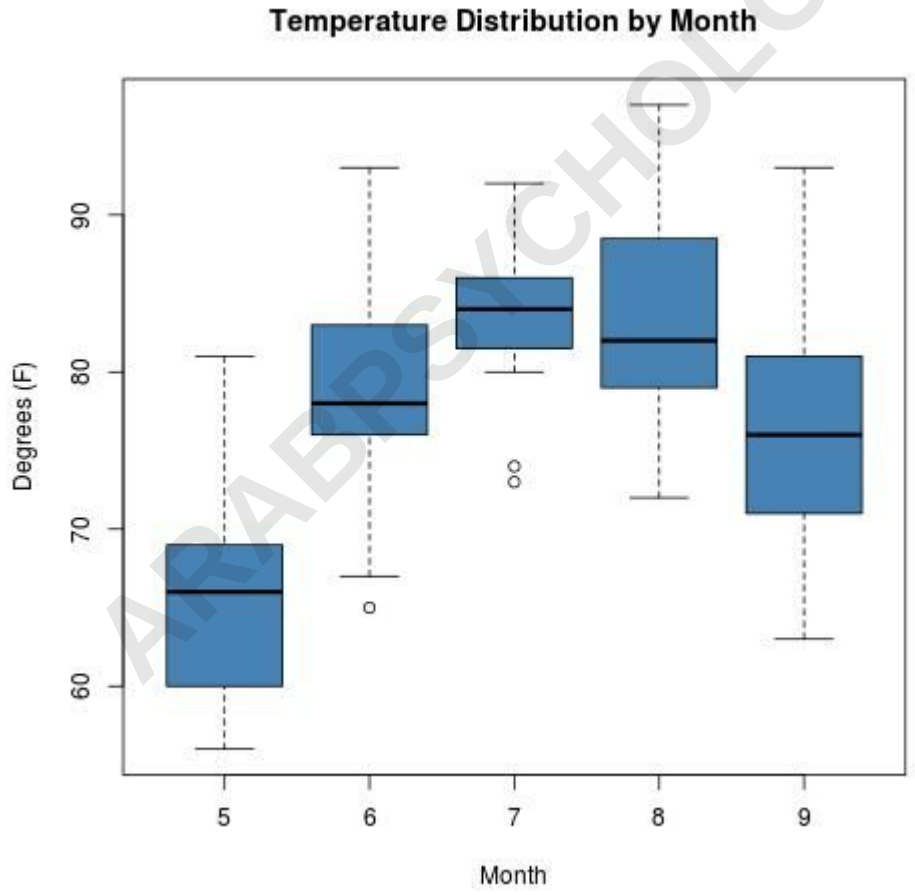
To produce a more informative chart that displays multiple boxplots side-by-side, we utilize the formula **syntax** in R. The formula takes the form `y ~ x`, where `y` represents the numerical data we wish to plot (the dependent variable) and `x` represents the categorical variable used for grouping (the independent variable). In our case, if we want to visualize the temperature distribution for each month, we would use `Temp ~ Month`. This tells R to split the "Temp" data into subsets based on the unique values found in the "Month" column.

In addition to the formula, we can pass several arguments to the `boxplot()` function to enhance the clarity and professionalism of the chart. The `main` argument adds a title, while `xlab` and `ylab` provide labels for the axes. We can also apply color to the boxes using the `col` **parameter** and define the **border** color. These enhancements make the **data visualization** significantly easier for an audience to interpret.

**#create boxplot that displays temperature distribution for each month in the dataset**

```
boxplot(Temp~Month,
data=airquality,
main="Temperature Distribution by Month",
xlab="Month",
ylab="Degrees (F)",
col="steelblue",
border="black"
)
```

The resulting chart displays five distinct boxplots, one for each month from May to September. This format allows for an immediate visual comparison of temperature trends. For example, one can easily see that the median temperature increases through the summer months and begins to decline as autumn approaches. The spread of temperatures (the height of the boxes) also varies by month, indicating periods of higher or lower thermal stability.



## Fine-Tuning Graphics with Base R Parameters

Beyond the basic aesthetic arguments, **R language** offers deep control over the layout of your charts through the `par()` function. The `par()` function allows you to set global graphical **parameters**, such as margins (`mar`), the number of plots per page (`mfrow`), and text magnification (`cex`). When dealing with multiple boxplots, adjusting the margins is often necessary to ensure that axis labels do not overlap or get cut off by the edge of the plotting device.

Another important aspect of fine-tuning involves the use of the `at` argument within the `boxplot()` function. This allows the user to specify the exact horizontal positions of each box, which can be useful if you want to group certain boxes closer together or leave gaps between specific categories. Furthermore, the `width` argument can be used to set the relative widths of the boxes, perhaps making them proportional to the sample size of each group. This level of detail is what makes **base R** a powerful tool for custom **data visualization**.

For users who need to present their findings in a professional context, these adjustments are not merely cosmetic; they improve the **communication** of data. A well-spaced plot with clear labels and intuitive colors reduces the cognitive load on the viewer, allowing the underlying statistical patterns to stand out. While **base R** requires a bit more manual configuration to achieve perfection compared to modern packages, it remains a favorite for many due to its lack of external **dependencies** and its predictable behavior.

## Transitioning to the Grammar of Graphics with ggplot2

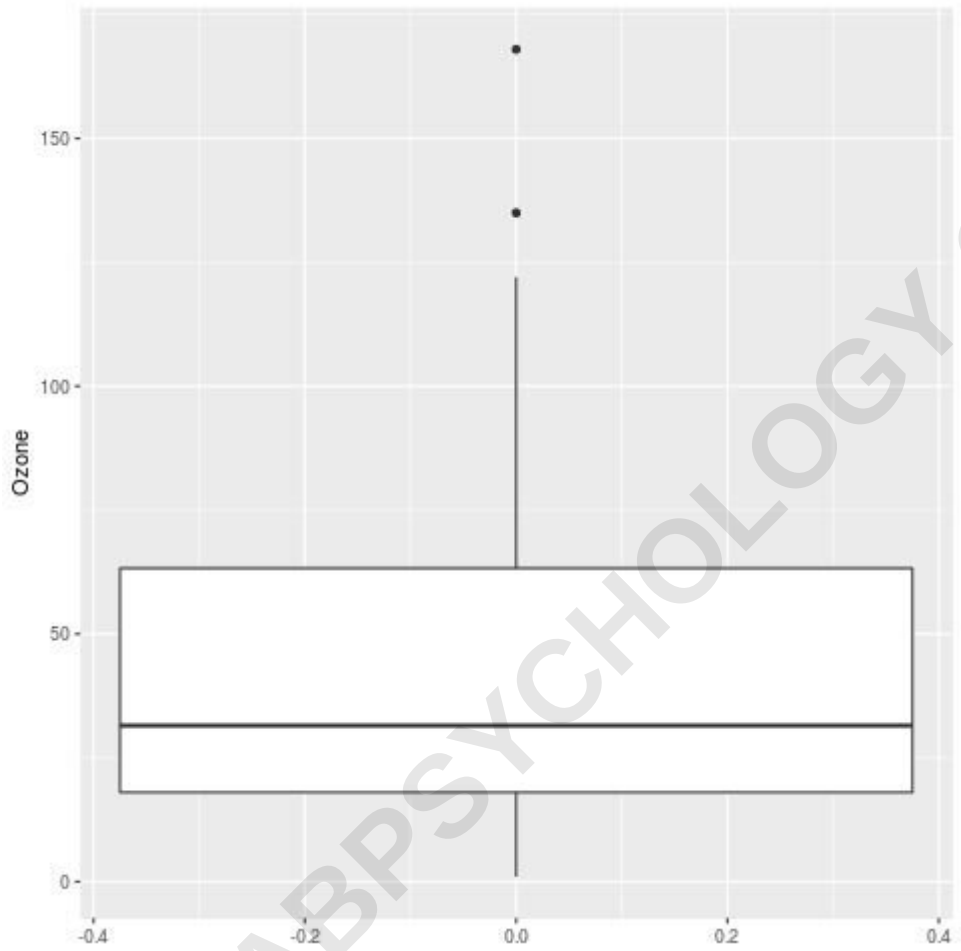
While base R is excellent for quick explorations, many **data scientists** prefer the **ggplot2** package for more complex or aesthetically refined visualizations. Developed by **Hadley Wickham**, **ggplot2** implements the **Grammar of Graphics**, a systematic way of describing and building graphs. Instead of calling a single function with many arguments, you build a plot layer by layer, starting with the data and then adding geometric objects (geoms), scales, and labels.

To use **ggplot2**, you must first ensure the package is installed and loaded into your R session. The fundamental function is `ggplot()`, where you define the **data frame** and the aesthetic mappings (`aes`). Mappings describe how variables in the data are mapped to visual properties like the x-axis, y-axis, color, and size. For a **boxplot**, we use the `geom_boxplot()` layer to tell R how to render the data points.

**#create boxplot for the variable "Ozone"**
**library(ggplot2)**

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

The code above creates a single boxplot for the "Ozone" variable. Note that in **ggplot2**, if you only provide a `y` mapping, it will generate a single plot for that variable across a default x-axis. The resulting visual is often cleaner than base R by default, utilizing a subtle grey grid and modern typography. This provides a professional baseline upon which further customizations can be layered.






## Advanced Multi-Category Plots in ggplot2

To create multiple boxplots in a single chart using **ggplot2**, we modify the aesthetic mapping to include both an `x` and a `y` variable. However, a common pitfall occurs when the grouping variable (like "Month") is stored as a numeric type. If **ggplot2** perceives the x-axis as continuous, it may try to create a single wide boxplot or place boxes on a continuous scale. To fix this, we wrap the grouping variable in the `as.character()` or `factor()` function, ensuring R treats it as a discrete category.

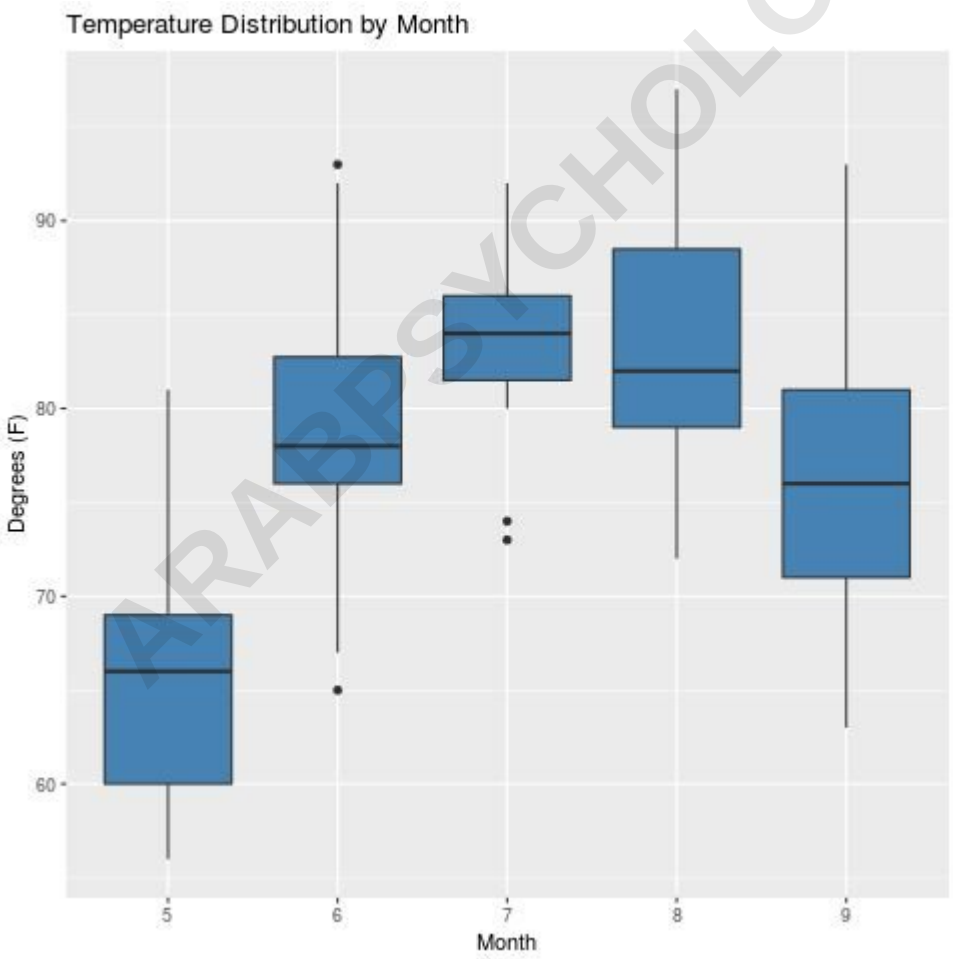
The following **syntax** demonstrates how to group temperature data by month. We also use the `fill` argument inside `geom_boxplot()` to color the interior of the boxes and the `labs()` function

to define the title and axis labels. This modular approach makes it very easy to swap variables or change the look of the plot without rewriting the entire command.

**#create boxplot that displays temperature distribution for each month in the dataset**
**library(ggplot2)**

```
ggplot(data = airquality, aes(x=as.character(Month), y=Temp)) +
geom_boxplot(fill="steelblue") +
labs(title="Temperature Distribution by Month", x="Month", y="Degrees (F)")
```

The resulting chart is highly readable and visually appealing. Using `as.character(Month)` ensures that each month from 5 to 9 gets its own distinct position on the x-axis. This method of **data visualization** is particularly powerful because it allows for easy "faceting"--creating a grid of plots--should you want to compare these distributions across even more variables, such as different years or locations.



## Comparative Analysis and Conclusion

Choosing between **base R** and **ggplot2** for generating multiple boxplots often depends on the specific goals of the **data analysis**. Base R is unsurpassed for rapid data checking and simple visualizations where speed is prioritized over aesthetics. Its formula **syntax** is intuitive and requires very little boilerplate code. However, as the complexity of the visualization increases, the code can become cluttered with many individual **parameters**.

On the other hand, **ggplot2** provides a more structured and scalable way to build plots. While it has a slightly steeper learning curve due to the **Grammar of Graphics** concept, it offers much more flexibility for creating publication-quality charts. Features like automatic legend generation, easy faceting, and a wide array of available themes make it the industry standard for professional **R programming**. Regardless of the tool chosen, the boxplot remains an indispensable asset in the statistician's toolkit for summarizing data distributions efficiently.

In summary, creating a single plot with multiple boxplots in R is an essential skill for any analyst. By utilizing the **airquality dataset**, we have demonstrated that both base R and **ggplot2** can effectively communicate the variance and central tendency across different groups. Mastery of these techniques enables clearer insights and more compelling storytelling through data, ultimately leading to more informed decision-making based on statistical evidence.

## Additional Resources and Further Reading

The following tutorials offer additional information about boxplots and related data visualization techniques in R:

To further expand your knowledge, consider exploring the official documentation for the **R language** and the extensive community support available on platforms like Stack Overflow. Understanding the mathematical properties of the **interquartile range** and **standard deviation** will also deepen your appreciation for what these visual tools represent. Continuous practice with diverse **datasets** is the best way to become proficient in choosing the right visualization for your specific analytical needs.