

How to Easily Change the Legend Position in ggplot2

Authored by
stats writer

December 6, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Change the Legend Position in ggplot2*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=106404>

Effective data visualization relies heavily on clarity, and the placement of graphical elements, particularly the **ggplot2** legend, is critical for achieving optimal readability. The legend serves as the map for the data aesthetics, explaining which colors, shapes, or sizes correspond to specific variables in the plot. However, a poorly positioned legend can obscure important data points or waste valuable plotting space. Fortunately, the **R** package **ggplot2** provides robust and highly flexible tools for precisely controlling where this key element resides. This control is primarily managed through the powerful **theme()** function, which allows users to modify non-data elements of the plot.

Understanding the arguments within the **theme()** function dedicated to legend control is essential for crafting professional-quality visualizations. The primary argument we manipulate is **legend.position**, which accepts various inputs--from simple directional keywords like "top" or "bottom" to highly granular, numeric coordinate pairs. This flexibility ensures that regardless of the complexity or layout of the visualization, the legend can be seamlessly integrated without compromising the visual narrative. Mastering these positioning techniques moves a visualization from merely functional to highly effective and polished.

The default positioning of the legend is usually on the right side of the plot, which works well in many standard scenarios. However, complex layouts, small multiples (faceting), or specific publication requirements often necessitate moving the legend to another location, sometimes even embedding it within the plotting area itself. In the following sections, we will explore the precise syntax and practical applications for manipulating the **legend.position** argument, using canonical examples based on the built-in **iris** dataset to illustrate these concepts clearly.

The following syntax demonstrates the basic structure for specifying the position of a **ggplot2** legend:

```
theme(legend.position = "right")
```

The subsequent examples utilize this powerful syntax in practice, demonstrating its application using the well-known **iris** dataset built into **R**.

The Core Mechanism: Utilizing the **theme()** Function

The cornerstone of aesthetic customization in **ggplot2** is the **theme()** function. This function is designed to manage all non-data ink on the plot, including titles, axis labels, gridlines, background colors, and, most importantly for this discussion, the legend. By adding a **theme()** call as a layer to your plot object, you override the default settings established by the package, enabling granular control over the final output. This modular approach is characteristic of the grammar of graphics framework that **ggplot2** is built upon, ensuring that visual modifications are clearly separated from

data mapping and geometric representation.

To change the location of the legend, we focus specifically on setting the **`legend.position`** argument within the **`theme()`** function. The basic syntax for this operation is straightforward, allowing users to quickly specify a new general location for the legend box relative to the entire plot area. This simplicity is crucial for rapid prototyping and visualization refinement. The argument accepts a variety of inputs, making it versatile for both quick adjustments and highly precise fine-tuning.

When preparing your code, the **`theme()`** layer is typically the final element added to the ggplot object before rendering, ensuring that the visual styling is applied last. Below is the fundamental syntax structure used to define the legend's position using one of the predefined directional strings. This syntax forms the basis for all external legend placements and should be memorized as a core technique in **ggplot2** manipulation.

Positioning the Legend Outside the Plot (Keywords: "top", "bottom", "left", "right")

The most common requirement for legend placement is positioning it adjacent to the plotting panel. **ggplot2** simplifies this process by accepting explicit string values that correspond to the four cardinal directions relative to the plot area: "top", "bottom", "left", and "right". When one of these character strings is passed to **`legend.position`**, the plot automatically resizes the plotting panel to accommodate the legend box without overlap, ensuring a clean separation between the data visualization and its explanatory key.

Choosing the correct side depends heavily on the plot's aspect ratio and the density of the data. For instance, if the plot is tall and narrow, moving the legend to the "top" or "bottom" often conserves horizontal space. Conversely, if the plot is wide, retaining the default "right" position or moving it to the "left" may be more efficient. Furthermore, placing the legend at the "bottom" or "top" generally encourages a horizontal arrangement of legend items, which can be further controlled using the **`legend.direction`** argument (not explicitly detailed here, but relevant for advanced fine-tuning).

The key benefit of using these external positioning keywords is reliability and simplicity. The placement is always guaranteed to be outside the data panel, making the code highly readable and robust across different device sizes and rendering contexts. We will now examine specific implementations of the "top" and "bottom" positions, demonstrating how these simple adjustments drastically alter the overall layout and flow of the visualization.

Example: Placing the Legend at the Top or Bottom

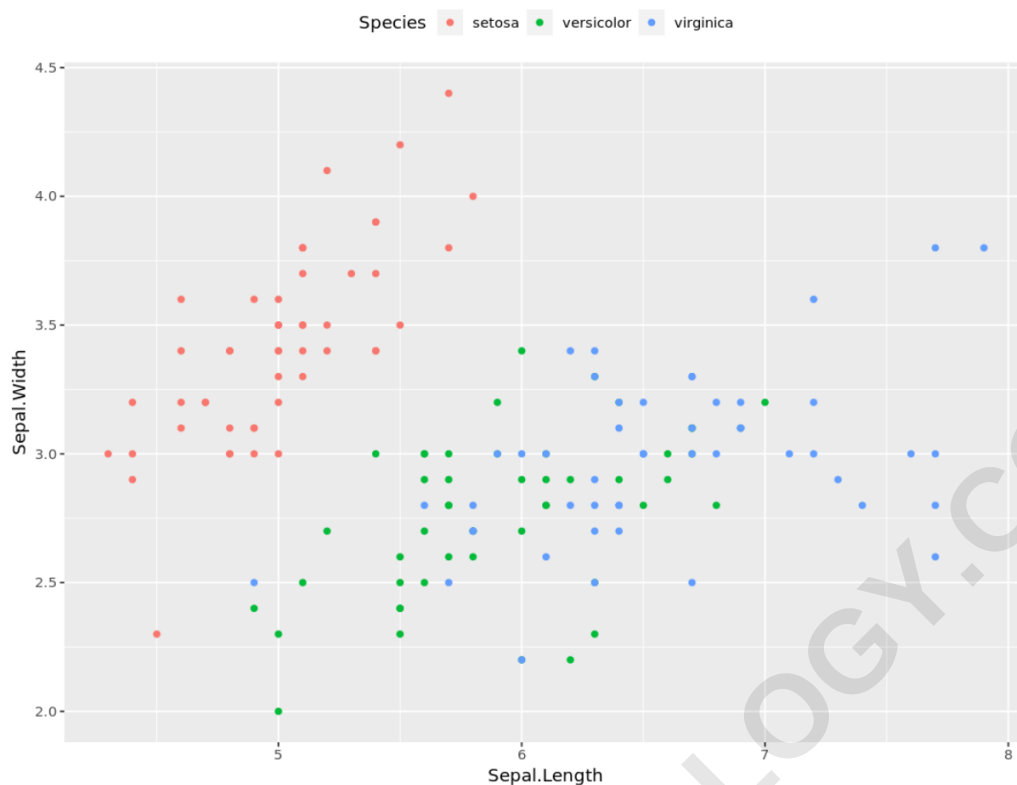
When the legend is placed at the top of the plot, it typically adopts a horizontal orientation, stacking the keys and labels side-by-side. This arrangement is particularly useful for wide visualizations or when the legend contains only a few discrete categories, such as the three species in the **iris** dataset. Placing the legend above the plot often helps guide the reader's eye, allowing them to identify the aesthetic mappings before they examine the spatial relationships of the data points themselves.

To place the legend above the plotting area, we simply assign the string "top" to the **legend.position** argument within the **theme()** function. The following code snippet demonstrates this operation, utilizing the **aes** (aesthetics) mapping to color the points based on the **Species** variable and employing **geom_point()** to generate a scatter plot of sepal dimensions.

Observe the concise nature of the command: the entire modification is contained within the final **theme()** layer, ensuring that the structure of the data mapping and geometric layer remains untouched, adhering to the best practices of the grammar of graphics.

```
library(ggplot2)
```

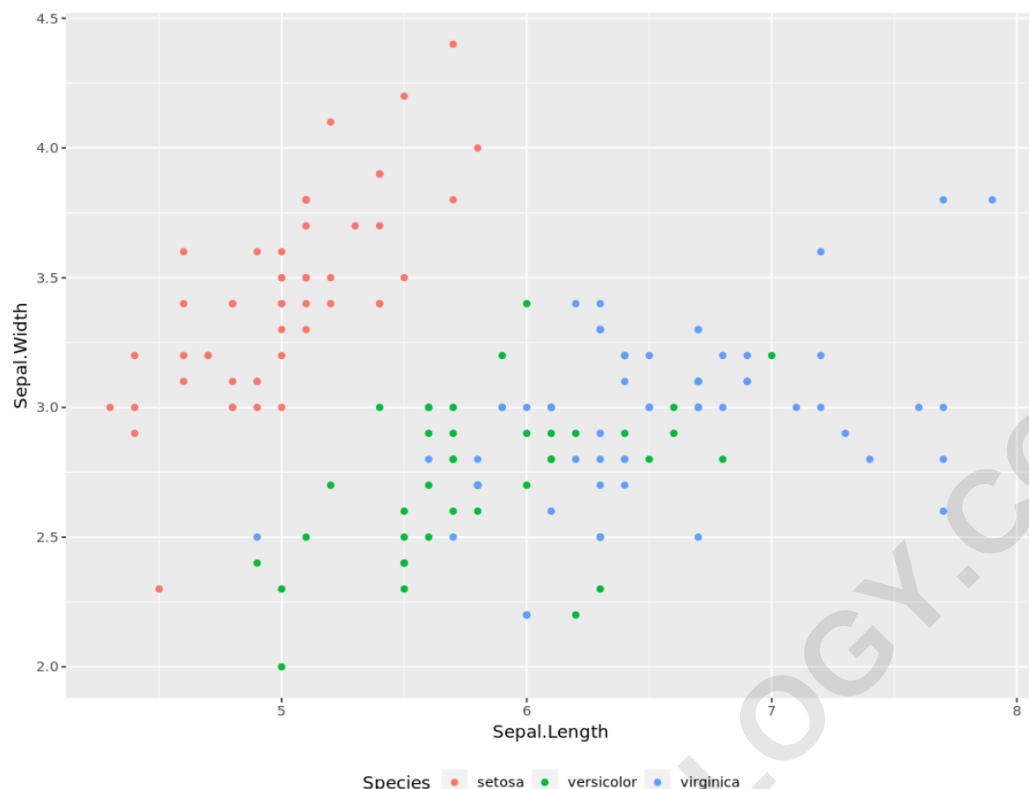
```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +  
  geom_point() +  
  theme(legend.position = "top")
```



Conversely, placing the legend at the bottom follows the exact same pattern, replacing "top" with "bottom". This position is often preferred when the top area is reserved for a primary title or subtitle, maintaining a visual hierarchy where explanatory text precedes the data, and the key follows the visual representation. This placement is particularly effective when the plot itself is visually busy, providing a clear visual break at the bottom margin for the legend information.

library(ggplot2)

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +  
geom_point() +  
theme(legend.position = "bottom")
```



Positioning the Legend Inside the Plot Using Coordinates

While external positioning (top, bottom, left, right) is sufficient for many scenarios, advanced customization often requires placing the legend directly within the data panel itself. This technique, known as "insetting," is highly valuable when space is extremely limited or when the legend naturally fits into a low-density region of the plot, preventing the visualization from expanding unnecessarily. To achieve insetting, the **`legend.position`** argument must be supplied with a numeric vector of length two: `c(x, y)`.

These `(x, y)` coordinates represent relative positions within the plotting area, normalized between 0 and 1. The origin `(0, 0)` corresponds to the bottom-left corner of the plot, and `(1, 1)` corresponds to the top-right corner. Therefore, specifying `c(0.5, 0.5)` would center the legend precisely in the middle of the plot. It is crucial to remember that these coordinates determine where the anchor point of the legend box is positioned, although the exact alignment relative to this anchor point can be further controlled using the `legend.justification` argument (which defaults to centering the box).

Using coordinate pairs offers unparalleled precision, allowing the analyst to place the legend exactly where it causes the least interference with the underlying data trends. For example, if the top-right corner of the scatter plot contains very few data points, it becomes an ideal candidate for

legend placement, ensuring that the visual evidence remains clear while still providing the essential key. Let's examine how to place the legend in the top-right and bottom-right corners using this coordinate system.

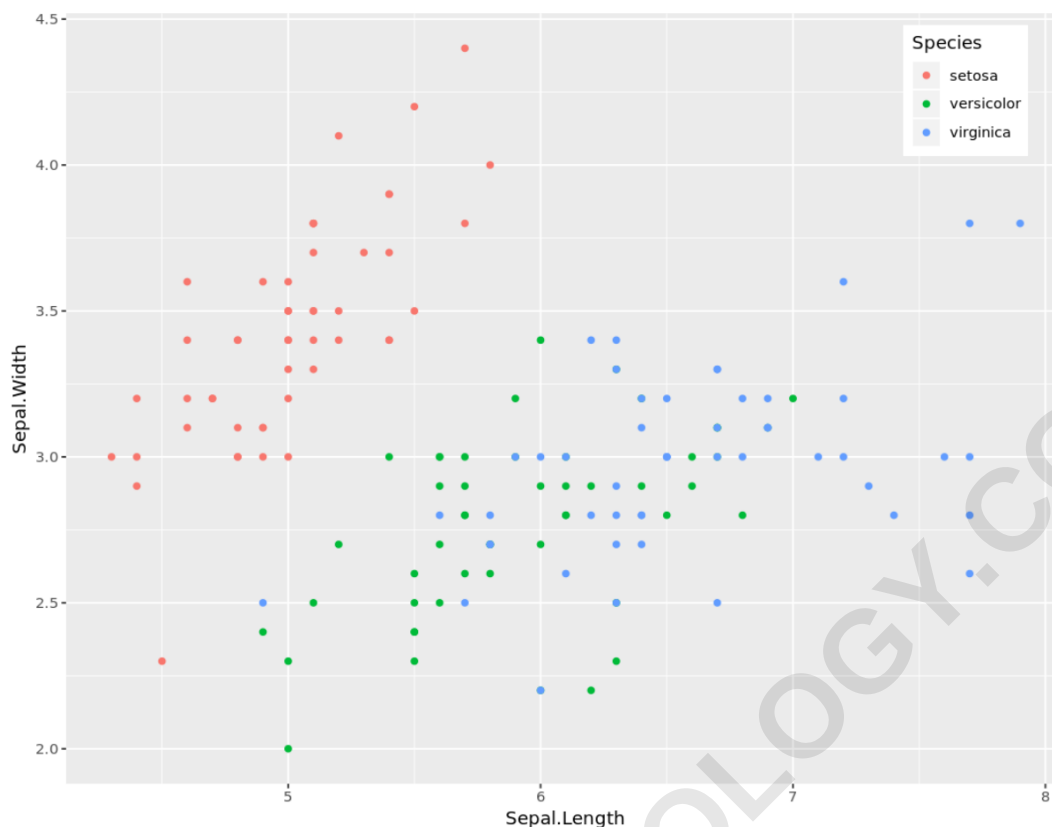
Implementing Internal Coordinate Placement

To demonstrate placing the legend in the top-right corner, we use coordinates close to (1, 1), such as `c(.9, .9)`. These values place the anchor point near the maximum x and y bounds, pushing the legend box into the upper-right region of the visual space. It is important to experiment with these values, as choosing (1, 1) might sometimes push part of the legend box slightly off-panel, depending on the legend's overall size. Using slightly smaller values like `(.95, .95)` or `(.9, .9)` often yields better results by providing a small buffer from the edge.

The following code block updates the **`legend.position`** argument to accept this coordinate vector instead of a directional string. Notice the shift in syntax, where the input is now a numeric vector rather than a quoted character string, signaling to **ggplot2** that internal positioning based on relative coordinates is required.

```
library(ggplot2)
```

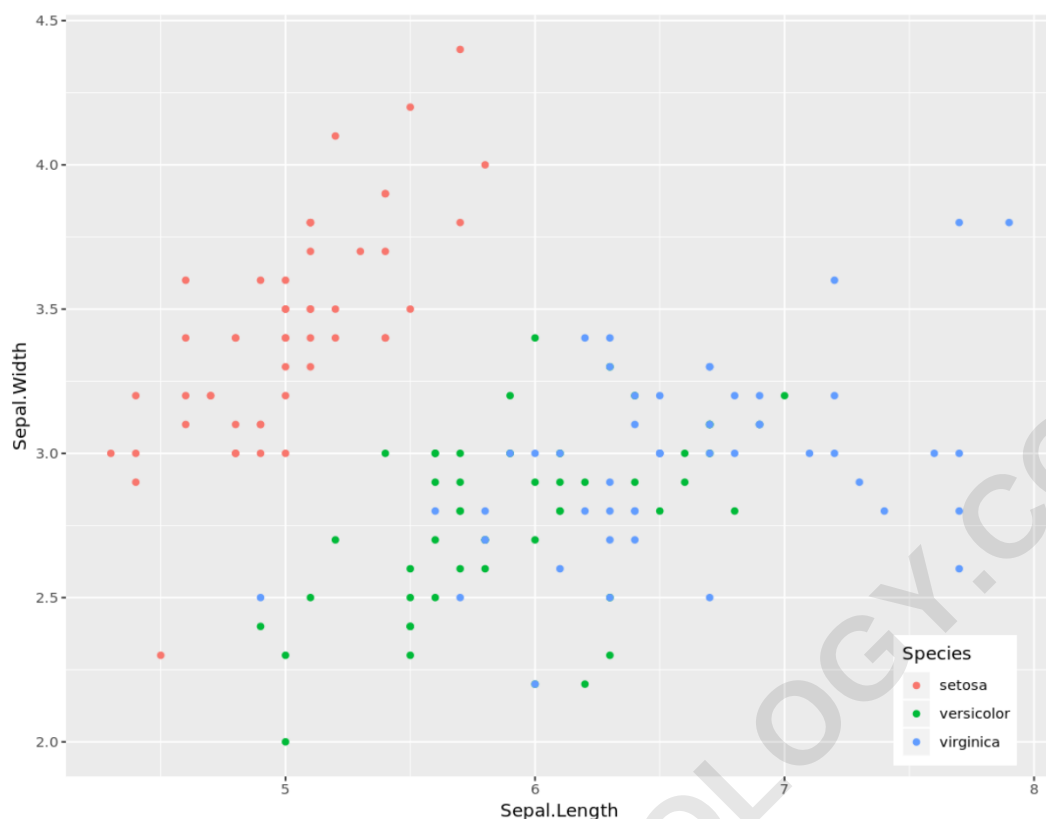
```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +  
geom_point() +  
theme(legend.position = c(.9, .9))
```



For the bottom-right corner, we keep the x-coordinate high (near 1) and set the y-coordinate low (near 0). Using `c(.9, .1)` anchors the legend near the intersection of the right edge and the bottom edge of the plot. This position is frequently used when the bottom-left area is dedicated to a footnote or source citation, ensuring the legend is easily accessible but does not overlap with axis labels or other important graphical elements. The successful placement of the legend within the plot boundaries confirms the power and precision afforded by the coordinate system approach, making it a favorite technique for bespoke visualizations.

library(ggplot2)

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +  
geom_point() +  
theme(legend.position = c(.9, .1))
```



Example: Remove Legend Completely

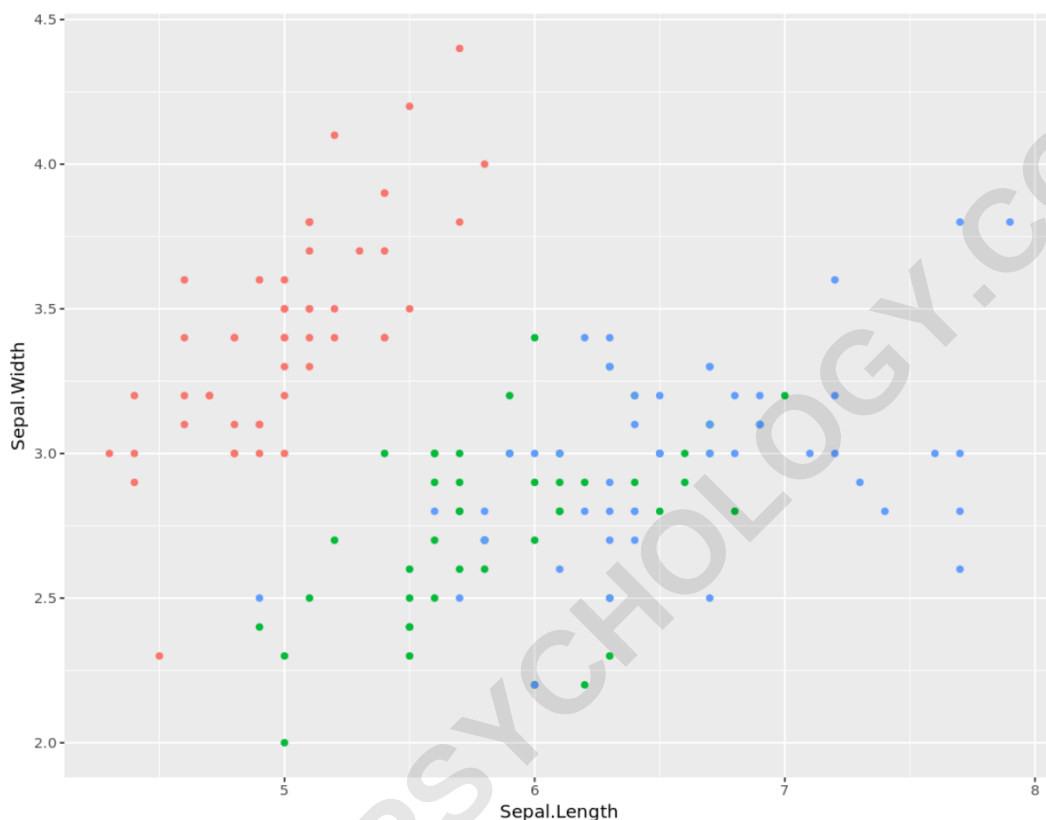
In specific contexts, the legend may become redundant or unnecessary. This usually occurs when the aesthetic mapping is self-explanatory, if the categories are labelled directly on the plot, or if the visualization is part of a larger, faceted display where a single, common legend is used elsewhere. In such cases, removing the legend entirely can greatly simplify the visual appearance and maximize the area dedicated to the data itself, enhancing data-ink ratio. **ggplot2** provides a specific value for the **legend.position** argument to achieve this deletion.

To suppress the legend, the string value "none" is passed to **legend.position**. This command instructs the **theme()** function to discard the generated legend graphical object completely. It is important to note that removing the legend does not remove the aesthetic mapping itself (e.g., the points are still colored by **Species**); it only removes the explanatory key. If the colors need to be standardized across multiple plots even without the key, this method is highly effective, allowing the visual attributes to persist without the accompanying informational box.

The final example below illustrates the clean and uncluttered plot that results from setting **legend.position="none"**. This technique is a fundamental tool for expert **ggplot2** users who prioritize visual economy and data density in their final outputs, ensuring that the visual focus remains entirely on the distribution and relationships within the data.

library(ggplot2)

```
ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +  
geom_point() +  
theme(legend.position = "none")
```



Summary of Legend Positioning Strategies

The ability to precisely control the placement and visibility of the legend is a key component of generating high-quality visualizations using **ggplot2**. Whether the goal is to save space by moving the legend internally, or to adjust the layout using external directional keywords, the **theme()** function provides the necessary toolkit. Users should always prioritize clarity and ensure the legend placement complements the data being presented, rather than obstructing it.

In summary, there are three primary approaches for modifying the legend position, each tailored to different design requirements:

Setting **legend.position** to a predefined string ("top", "bottom", "left", "right") for reliable external placement, which automatically adjusts plot margins.

Setting **legend.position** to a normalized numeric vector `c(x, y)` (0 to 1) for precise, internal placement within the data panel.

Setting **legend.position** to the string "none" to completely suppress the legend when it is unnecessary or redundant.

By mastering these arguments and understanding the implications of each placement strategy on the overall visual hierarchy, developers can ensure their **ggplot2** outputs are both aesthetically pleasing and scientifically rigorous. Further customization can be achieved by exploring related arguments like `legend.justification` and `legend.box`, which offer even finer control over layout and alignment, but the core positioning remains defined by the `legend.position` argument.