

# How to Change the Legend Title in ggplot2 (With Examples)?

Authored by  
**stats writer**

December 20, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Change the Legend Title in ggplot2 (With Examples)?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=108057>

## The Role of Legends in Effective Data Visualization

Effective data visualization is paramount to communicating complex statistical findings clearly and accurately. Within the R ecosystem, the ggplot2 package, built upon the Grammar of Graphics principles, stands as the industry standard for creating elegant and informative plots. A critical component of any multi-variate plot created using this framework is the legend, which serves as the essential key linking visual aesthetics (like color, shape, or size) back to the underlying data variables. When generating a plot, ggplot2 automatically generates a legend, often using the column name from the source data as the default title.

While functional, default legend titles--which might include underscores, abbreviations, or inconsistent capitalization derived directly from dataset columns--are frequently not optimized for public consumption or formal reporting. To elevate the clarity and professionalism of a chart, analysts must customize these default elements. Renaming the legend title allows for the use of descriptive, plain language labels that immediately clarify what the colors or symbols represent, thereby reducing cognitive load for the viewer and improving overall comprehension of the visualized insights. This customization process is a fundamental step in polishing statistical graphics for publication or presentation.

Fortunately, the ggplot2 package offers highly flexible and straightforward methods for renaming legend titles. These methods are designed to integrate seamlessly into the existing plotting pipeline, usually requiring only a single additional function call to override the default settings. We will explore two primary, equally powerful techniques--one focused on simple labeling and the other offering granular control over scales and visual mappings--that ensure your legends are both informative and aesthetically polished.

### Overview of Legend Title Modification in ggplot2

When working within the ggplot2 framework, the method you choose for customizing a legend title depends primarily on the level of control required over the plot's aesthetics and scales. Both methods are highly effective but serve slightly different purposes in the overall plotting workflow. The two most common and effective ways to modify the legend title involve either the general labeling function or the specific scale function tied to the aesthetic you are modifying.

The first approach involves utilizing the labs() function. This is generally the easiest method, functioning as a shortcut to change labels for various plot elements--including the main title, axis labels, and, critically, legend titles associated with specific aesthetics like color, fill, or size. When using labs(), you simply specify the aesthetic being mapped (e.g., `fill`, `color`) and assign it a new character string, which immediately updates the corresponding legend title. This method is preferred for quick fixes and general renaming when the underlying colors or values do not need explicit manual specification.

The second approach employs the specific scale function, such as `scale_fill_manual()` or `scale_color_manual()`. These functions offer comprehensive control over the entire mapping process--allowing you to simultaneously define the legend title, the specific colors (or shapes/sizes) used, and the order of the levels. While slightly more complex as it requires defining the visual outputs (e.g., color names or hex codes), this method is indispensable when you need precise aesthetic control, ensuring that specific data categories are mapped to specific visual characteristics.

### Method 1: Use `labs()`

Below is the conceptual structure showing how the `labs()` function integrates into a standard `ggplot2` pipeline. Notice how the argument within `labs()` matches the aesthetic argument used in the main `aes()` call (in this case, `fill`).

```
ggplot(data, aes(x=x_var, y=y_var, fill=fill_var)) +  
geom_boxplot() +  
labs(fill='Legend Title')
```

### Method 2: Use `scale_fill_manual()`

Conversely, the structure for using the `scale_fill_manual()` function requires passing the new legend title as the first positional argument, followed by explicitly defining the visual values corresponding to the data levels. This provides maximal control over both the label and the appearance.

```
ggplot(data, aes(x=x_var, y=y_var, fill=fill_var)) +  
geom_boxplot() +  
scale_fill_manual('Legend Title', values=c('color1', 'color2'))
```

This extensive guide provides detailed, step-by-step examples demonstrating how to implement and utilize these two crucial methods in a practical data context.

## Method 1: Utilizing the `labs()` Function for Renaming

The `labs()` function is arguably the most straightforward utility for renaming plot elements in `ggplot2`. It operates by accepting aesthetic arguments (such as `x`, `y`, `color`, `fill`, `size`, etc.) and assigning them new string values. When `labs()` is used to redefine `fill` or `color`, it directly changes the title associated with the corresponding legend without altering any other aspect of the plot's visual scale or data mapping. This makes it ideal for situations where the existing color scheme is satisfactory, but the legend label needs refinement for clarity.

When preparing a plot, `ggplot2` links the legend to the variable mapped to the aesthetic. If you map the variable `program` to `fill`, the default legend title will be "program". By appending the `labs()` function and specifying `fill = 'New Title'`, the default label is immediately overridden. This function is extremely versatile and can be used to set multiple labels simultaneously, such as the plot title, subtitle, and both axis labels, all within a single function call, streamlining the code required for labeling.

## Detailed Example: Implementing `labs()` for Boxplot Legends

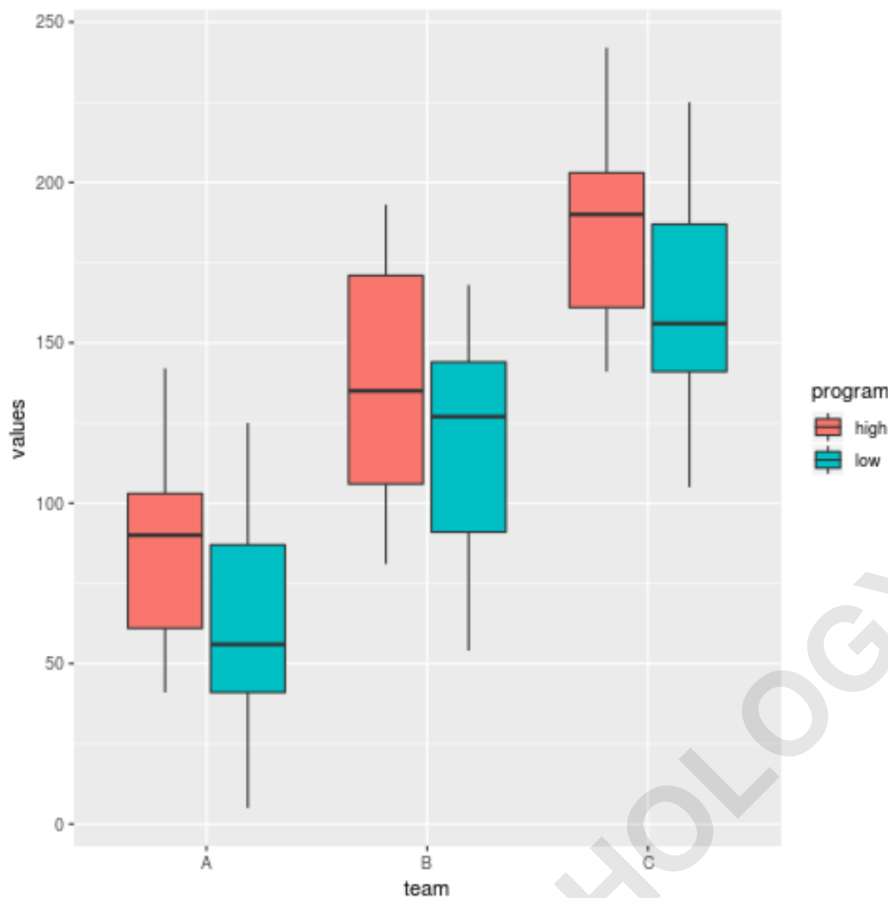
To illustrate the power and simplicity of the `labs()` function, we will first generate a sample dataset and a standard grouped `boxplot`. This base plot uses a categorical variable, `program`, to define the fill aesthetic, leading to the default, often undesirable, legend title. This initial demonstration establishes the baseline visualization before customization begins.

The following R code initializes the necessary library, creates a structured dataset containing three categorical variables (`team`, `program`) and one continuous variable (`values`), and then generates the initial grouped `boxplot`. Notice how the legend title automatically defaults to the variable name used for the `fill` aesthetic: `program`.

### **library(ggplot2)**

```
#create dataset
data <- data.frame(team=rep(c('A', 'B', 'C'), each=50),
  program=rep(c('low', 'high'), each=25),
  values=seq(1:150)+sample(1:100, 150, replace=TRUE))

#create boxplot
ggplot(data, aes(x=team, y=values, fill=program)) +
  geom_boxplot()
```



As demonstrated by the output, `ggplot2` defaults to using the variable name `program` as the legend title. This is functional but lacks professional polish. We can easily enhance the clarity by using the `labs()` function to assign a more descriptive title, such as "Program Type." This small adjustment significantly improves the plot's readability and prepares it for formal presentation.

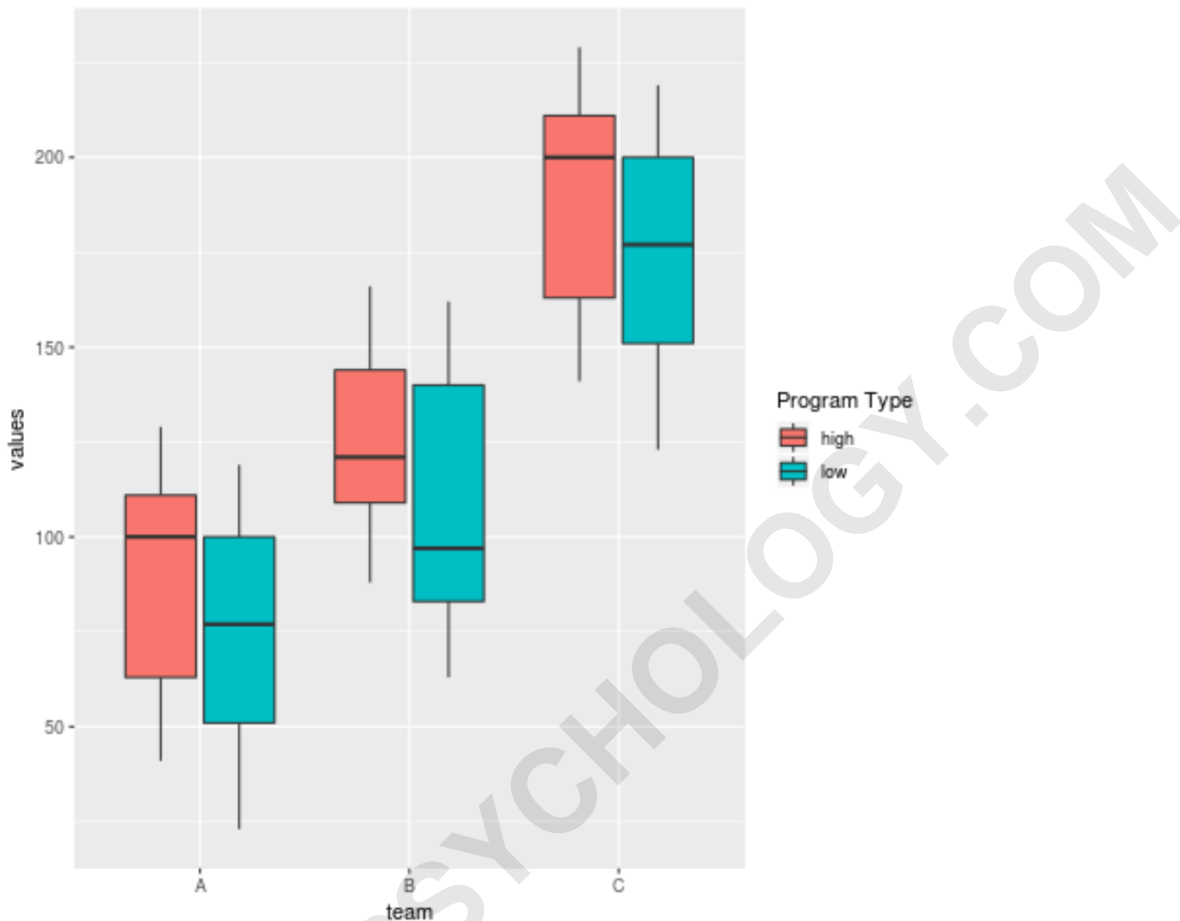
The following code block incorporates the `labs(fill='Program Type')` layer into the plot definition. Notice that the rest of the code remains unchanged, highlighting the non-destructive, additive nature of the `labs()` function. The key is ensuring the argument within `labs()` (`fill`) corresponds exactly to the aesthetic being mapped in the `aes()` call.

### library(ggplot2)

```
#create dataset
data <- data.frame(team=rep(c('A', 'B', 'C'), each=50),
  program=rep(c('low', 'high'), each=25),
  values=seq(1:150)+sample(1:100, 150, replace=TRUE))
```

```
#create boxplot
ggplot(data, aes(x=team, y=values, fill=program)) +
```

```
geom_boxplot() +  
labs(fill='Program Type')
```



## Integrating Line Breaks into Legend Titles

For plots with long or compound legend titles, fitting the text neatly into the available space can sometimes be challenging, particularly when exporting to formats with constrained dimensions. To improve the visual layout and prevent text from running off the plot boundaries, it is often desirable to introduce a line break within the legend title itself. `ggplot2` handles line breaks using the standard R syntax for new lines: `\n`.

By embedding the `\n` character string directly within the title argument of the `labs()` function, you instruct the rendering engine to start a new line at that exact position. This technique provides fine-grained control over the vertical spacing and visual density of the legend title, making it a valuable tool for detailed graphic design and ensuring complex titles remain readable.

In the example below, we aim to display "Program Type" across two lines, placing the line break

between "Program" and "Type." This transformation ensures the title is more compact, which can be beneficial when optimizing plot aesthetics alongside other elements like the legend position or overall chart dimensions.

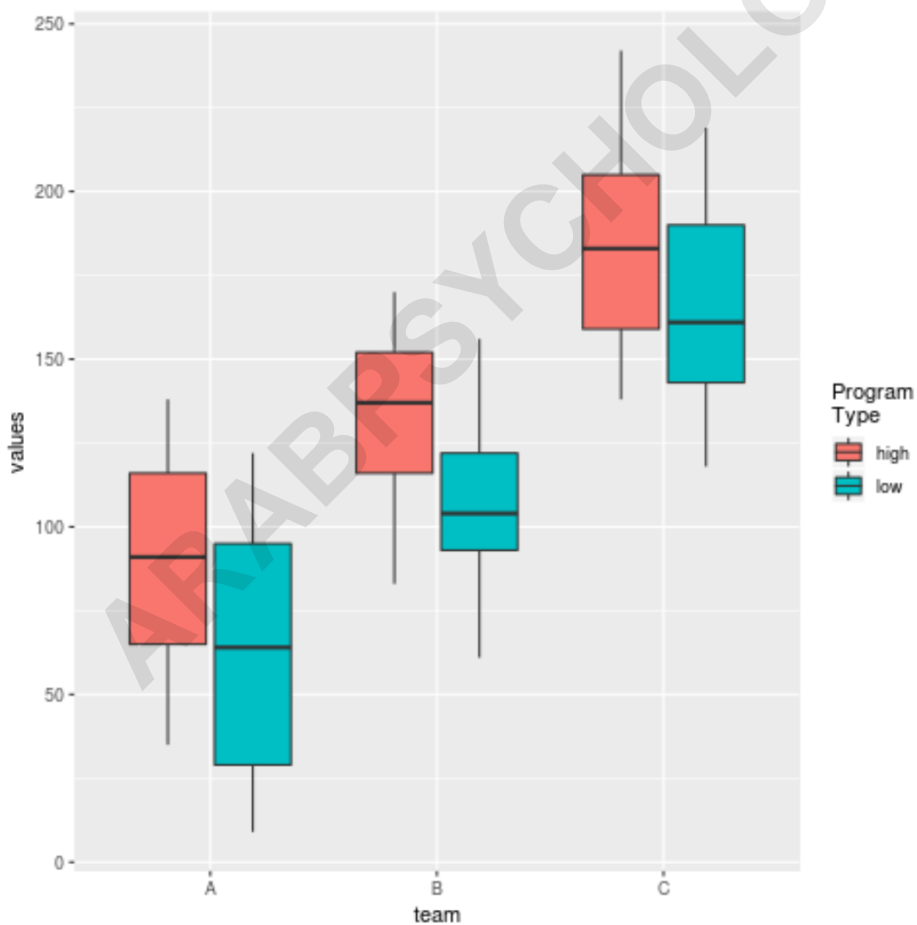
### library(ggplot2)

```
#create dataset
```

```
data <- data.frame(team=rep(c('A', 'B', 'C'), each=50),  
program=rep(c('low', 'high'), each=25),  
values=seq(1:150)+sample(1:100, 150, replace=TRUE))
```

```
#create boxplot
```

```
ggplot(data, aes(x=team, y=values, fill=program)) +  
geom_boxplot() +  
labs(fill='ProgramnType')
```



## Method 2: Leveraging `scale_fill_manual()` for Comprehensive Control

While `labs()` is excellent for simple renaming, the family of `scale_*_manual()` functions--such as `scale_fill_manual()`, `scale_color_manual()`, or `scale_shape_manual()`--provides a greater degree of control necessary for advanced customization. These functions are designed to manage the mapping between data values (the levels of your categorical variable) and the corresponding aesthetic output (e.g., specific colors).

The primary purpose of `scale_fill_manual()` is to allow the user to define precisely which color (or fill) corresponds to each level of the grouping variable. However, it also offers a built-in mechanism to rename the legend title. The new title is passed as the first argument, often labeled `name` in the function documentation, effectively overriding the default title derived from the aesthetic mapping. This dual functionality makes it the preferred choice when both visual appearance and labeling require explicit definition.

Using the manual scale functions is necessary when the default color palette chosen by `ggplot2` is not suitable, or when specific brand colors, accessibility requirements, or established visual standards must be applied to the chart. By defining the `values` argument within `scale_fill_manual()`, you ensure consistency and professional-grade aesthetics, while simultaneously ensuring the legend title is clear and descriptive.

### Practical Application of `scale_fill_manual()`

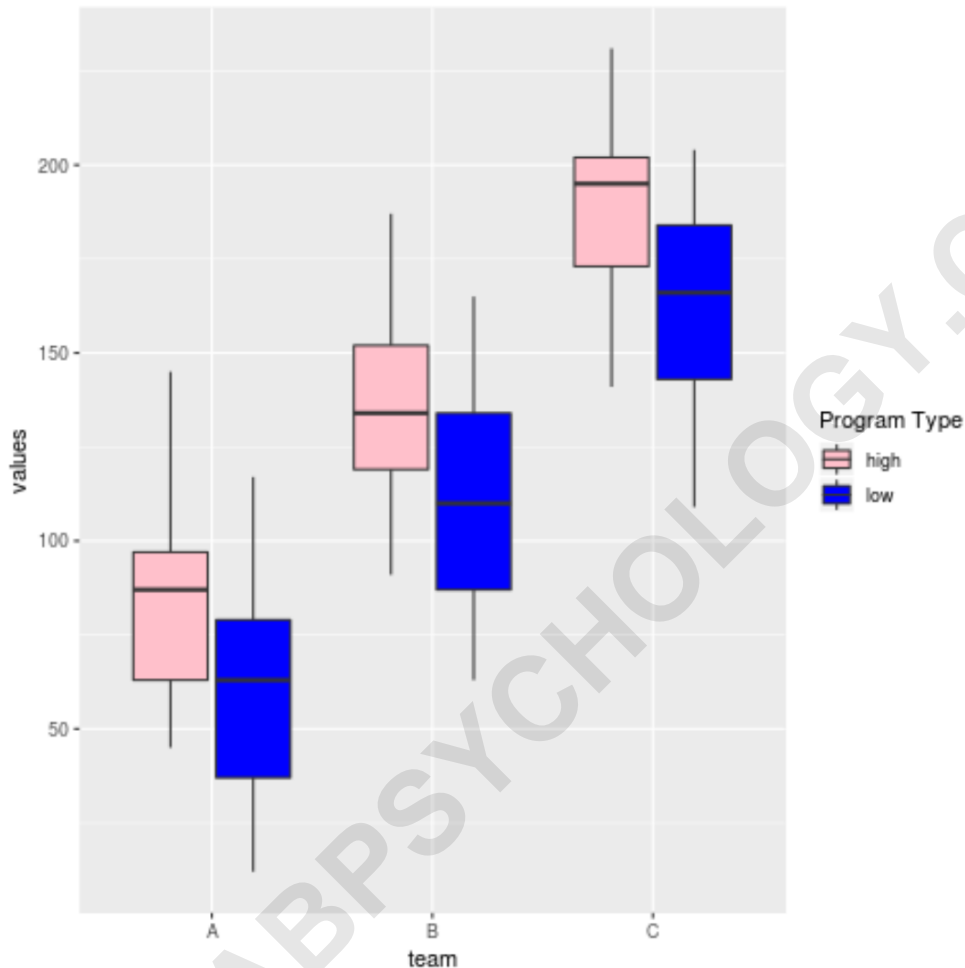
Continuing with our grouped `boxplot` example, we now apply `scale_fill_manual()`. This approach requires us to provide two essential pieces of information: the desired legend title (the `name` argument) and a vector of colors corresponding to the levels of the `program` variable (the `values` argument). Since our `program` variable has two levels ('low' and 'high'), we must supply exactly two color values.

In the code below, we achieve two goals in a single step: we rename the legend title to the more descriptive "Program Type," and we define the specific colors (pink and blue) that should be used for the boxplot fills. This demonstrates the superior control offered by manual scale functions compared to the simpler labeling functionality of `labs()`.

#### **library(ggplot2)**

```
#create dataset
data <- data.frame(team=rep(c('A', 'B', 'C'), each=50),
  program=rep(c('low', 'high'), each=25),
  values=seq(1:150)+sample(1:100, 150, replace=TRUE))
```

```
#create boxplot
ggplot(data, aes(x=team, y=values, fill=program)) +
  geom_boxplot() +
  scale_fill_manual('Program Type', values=c('pink','blue'))
```



It is important to note that the values provided in the `values` argument of `scale_fill_manual()` are highly flexible. They can be standard R color names (like 'pink' or 'blue'), or they can be specific hex color codes (e.g., '#FFC0CB' for pink and '#0000FF' for blue). Using hex codes provides exact control over the color shade, which is crucial for meeting corporate branding or scientific journal requirements.

## Choosing Between `labs()` and `scale_*_manual()` Functions

The decision regarding which function to use--`labs()` or a specific `scale_*_manual()` function--should be based on the scope of customization required. If your only goal is to change the title from the data variable name to a cleaner descriptive string, and you are satisfied with `ggplot2`'s

default colors (or if you are using a built-in color palette like `scale_color_brewer()`), then `labs()` is the simplest and most efficient tool. It focuses solely on labeling without interfering with the data-to-aesthetic mapping.

Conversely, if you need to enforce a specific color scheme, set custom shapes, or ensure precise ordering of legend items that deviates from the alphabetical default, then you must utilize the `scale_*_manual()` functions. These functions grant the granular control needed to manually map visual properties to data levels. While they require slightly more code (defining the values vector), they are essential for creating professional visualizations that adhere to strict design specifications.

## Further Customization: Position and Appearance

Beyond simply changing the title, the overall appearance and placement of the legend can also be modified using additional `ggplot2` functions. For instance, the general position, background, and text size of the legend are controlled using the powerful `theme()` function. Arguments such as `legend.position` (e.g., 'right', 'left', 'bottom', 'top', or coordinates like `c(0.8, 0.2)`) and `legend.title` (to adjust font size, color, or face) allow for detailed aesthetic refinement.

Furthermore, for advanced control over how the scales are mapped to the legend keys themselves--such as hiding specific legends or merging multiple legends into one--the `guides()` function is utilized. Although modifying the title is usually the first step in legend customization, mastering `theme()` and `guides()` is necessary to achieve fully polished, publication-ready graphics that integrate perfectly into the surrounding document layout.

Note that the values for the colors can either be names or hex color codes.

## Related ggplot2 Resources

[A Complete Guide to the Best ggplot2 Themes](#)

[The Complete Guide to ggplot2 Titles](#)

[How to Create Side-by-Side Plots in ggplot2](#)