

How to Easily Reorder Legend Items in ggplot2

Authored by
stats writer

December 3, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Reorder Legend Items in ggplot2*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=103904>

When creating complex data visualizations using the powerful **ggplot2** package in **R**, controlling the appearance of graphical elements is essential for effective communication. One common challenge users face is determining how to customize the order of items displayed in the plot **legend**. The default behavior of **ggplot2** often orders legend items alphabetically or based on the internal structure of the underlying data, which may not align with the narrative or logical sequence required for the visual analysis.

To successfully re-order items in the **ggplot2** legend, the underlying methodology relies on manipulating the sequence of categories. While it is often recommended to change the order of **factor** levels within the dataset before plotting--a fundamental practice in **R** data preparation--there is also a direct plotting approach that uses specific scaling functions to override the default display order. This guide explores both the preparatory method involving factors and the direct method using scaling functions, ensuring you have complete control over your visual output.

Understanding Default Legend Behavior in ggplot2

By default, **ggplot2** determines the order of legend entries based on the aesthetic mapping used (such as `fill`, `color`, or `shape`) and the type of variable assigned to that aesthetic. If the variable is a character string or an unordered factor, the items will usually be sorted alphabetically. If it is an ordered factor, the factor level hierarchy dictates the sequence. This automatic sorting is convenient for basic plots but becomes problematic when a specific logical or temporal flow must be maintained, such as showing "Low," "Medium," "High" or displaying teams in a championship rank order.

Effective data visualization requires that the legend complements the visual structure of the plot. If the visual elements (e.g., boxplots or bars) are ordered from left to right, the legend should ideally follow the same sequence to minimize cognitive load for the viewer. Therefore, mastering the techniques to manually impose a desired order is critical for producing publication-quality graphics.

Method 1: Using `scale_fill_discrete()` with the `breaks` Argument

The most direct method for overriding the default legend order in **ggplot2** involves utilizing the scale functions, specifically `scale_fill_discrete()` (or `scale_color_discrete()` if using color) and specifying the desired order using the `breaks` argument. The `breaks` argument accepts a character vector that explicitly lists the category names in the exact sequence they should appear in the legend.

This technique is particularly powerful because it allows the visualization structure (e.g., the X-axis order) to remain unchanged while independently controlling the legend structure. However, it is important to ensure that the values provided to the `breaks` argument exactly match the category names present in the underlying **data frame**, otherwise the items might be excluded from the

legend entirely.

You can use the following syntax to change the order of the items in a legend:

```
scale_fill_discrete(breaks=c('item4', 'item2', 'item1', 'item3', ...))
```

This function works by instructing **ggplot2** to look only for the values listed within `breaks` and arrange them in that specified sequence within the legend key. The following examples demonstrate how to use this syntax in practice, focusing on a scenario involving multiple boxplots.

Example 1: Setting up the Initial Visualization

To illustrate the process of reordering the legend, we will first create a baseline plot. This plot uses a simple **data frame** containing points scored by three different teams (A, B, and C). Notice that in the initial output, **ggplot2** orders the legend items alphabetically: A, B, C.

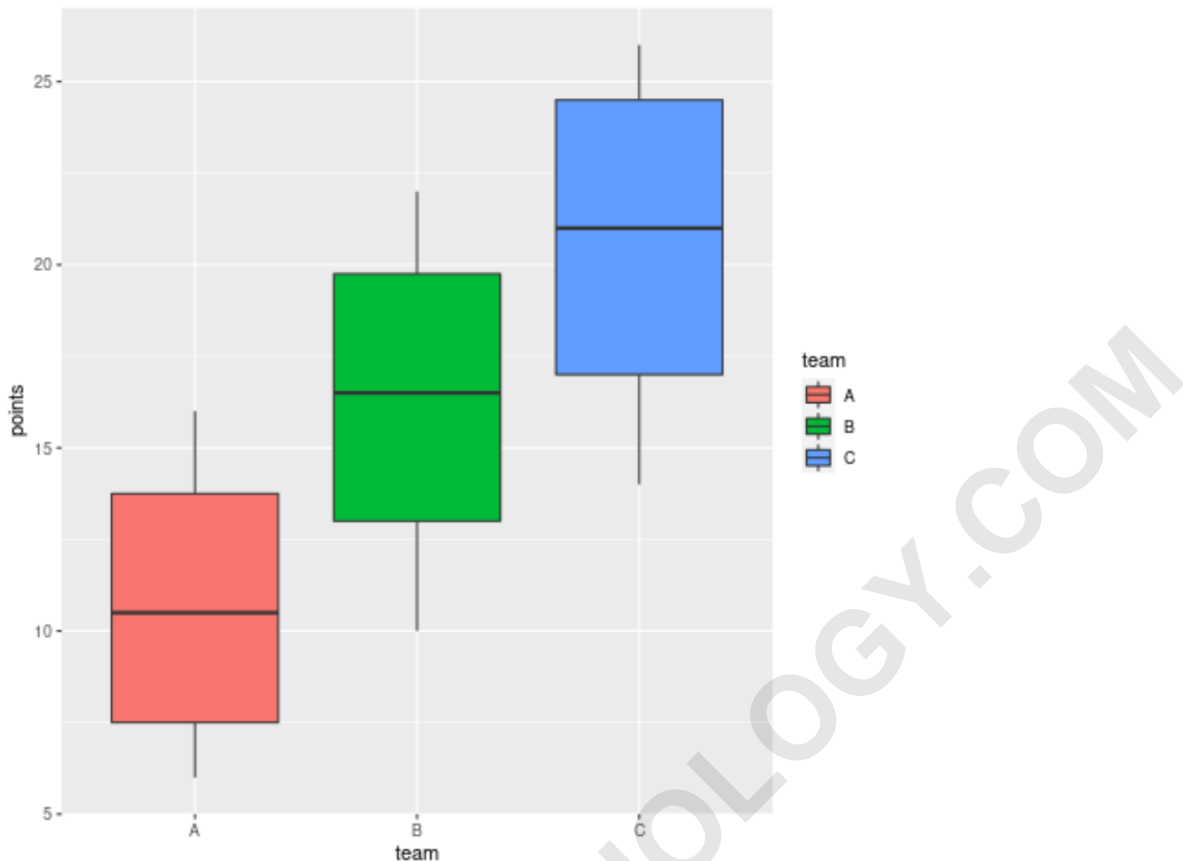
```
library(ggplot2)
```

```
#create data frame
```

```
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'C', 'C', 'C', 'C'),  
points=c(6, 8, 13, 16, 10, 14, 19, 22, 14, 18, 24, 26))
```

```
#create multiple boxplots to visualize points scored by team
```

```
ggplot(data=df, aes(x=team, y=points, fill=team)) +  
geom_boxplot()
```



This initial plot uses the `team` variable for both the x-axis mapping and the `fill` aesthetic, which is responsible for generating the legend. Because 'A', 'B', and 'C' are simple character strings, the legend defaults to an alphabetical arrangement. Our goal in the next step is to rearrange this legend without altering the order on the x-axis.

Example 2: Applying `scale_fill_discrete()` to Reorder the Legend

To change the order of the items in the legend, we can append the `scale_fill_discrete()` function to the existing plot code. We pass a vector to the `breaks` argument, specifying a non-standard order, such as B, C, A. This action only modifies the legend key, leaving the rest of the plot untouched, demonstrating the precision of this technique.

`library(ggplot2)`

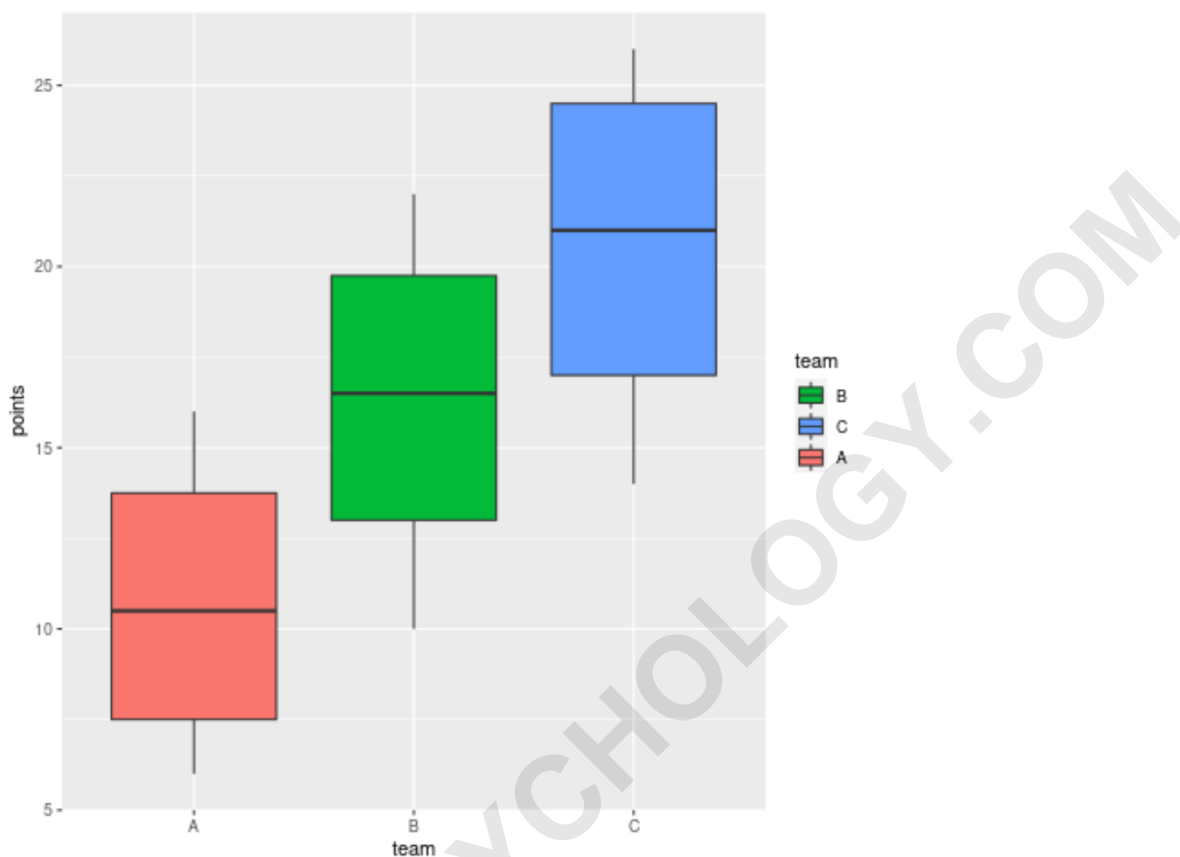
```
#create data frame
```

```
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'C', 'C', 'C', 'C'),
  points=c(6, 8, 13, 16, 10, 14, 19, 22, 14, 18, 24, 26))
```

```
#create multiple boxplots to visualize points scored by team
```

```
ggplot(data=df, aes(x=team, y=points, fill=team)) +
```

```
geom_boxplot() +
scale_fill_discrete(breaks=c('B', 'C', 'A'))
```



Notice that the order of the items in the legend has successfully changed from the alphabetical default (A, B, C) to the specified sequence (B, C, A). This demonstrates the efficacy of the `breaks` argument in providing granular control over the legend display. This method is highly flexible and should be the preferred approach when only the legend order needs adjustment.

Method 2: Manipulating Factor Levels for Consistent Ordering

Although using `scale_fill_discrete(breaks=...)` is efficient for legend reordering, best practice in **R** often dictates that the underlying data structure should reflect the desired ordering. If you wish the order on the X-axis, the legend, and any subsequent statistical analysis to all follow a specific sequence, you should manipulate the factor levels of the categorical variable.

In **R**, categorical variables are often stored as **factors**. The order of levels within a factor determines how those categories are treated in plotting and modeling functions. To change the order globally, you would typically use the base **R** function `levels()` or, more commonly and robustly, functions from the `forcats` package (part of the **tidyverse**), such as `fct_relevel()`. For

example, to set the levels to B, C, A across the entire dataset, the preparatory code would look like this:

```
library(dplyr)
```

```
library(forcats)
```

```
# Convert 'team' to factor and relevel it
```

```
df <- df %>%
```

```
mutate(team = fct_relevel(team, 'B', 'C', 'A'))
```

```
# Now, plotting without specifying breaks will automatically use the new factor order
```

```
ggplot(data=df, aes(x=team, y=points, fill=team)) +
```

```
geom_boxplot()
```

By modifying the factor levels, the resulting **ggplot2** plot would automatically display B, C, then A on the X-axis and B, C, then A in the legend, ensuring complete consistency between all visual elements derived from that variable. This is generally preferred for data consistency, whereas the `breaks` argument offers a quick fix exclusively for the legend.

Example 3: Customizing Legend Labels with the labels Argument

In addition to controlling the order, it is often necessary to change the descriptive text associated with the legend items without changing the underlying data values. For instance, the original data might use short codes (like 'A', 'B', 'C'), but the legend needs to display more informative text (like 'A Team', 'B Team'). This customization is achieved by adding the `labels` argument within the same `scale_fill_discrete()` function.

Crucially, when using both `breaks` and `labels`, the vectors must be of the same length and must correspond positionally. If `breaks` lists `c('B', 'C', 'A')`, then the `labels` vector must have three elements, where the first label corresponds to 'B', the second to 'C', and the third to 'A'.

```
library(ggplot2)
```

```
#create data frame
```

```
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B', 'C', 'C', 'C', 'C'),
```

```
points=c(6, 8, 13, 16, 10, 14, 19, 22, 14, 18, 24, 26))
```

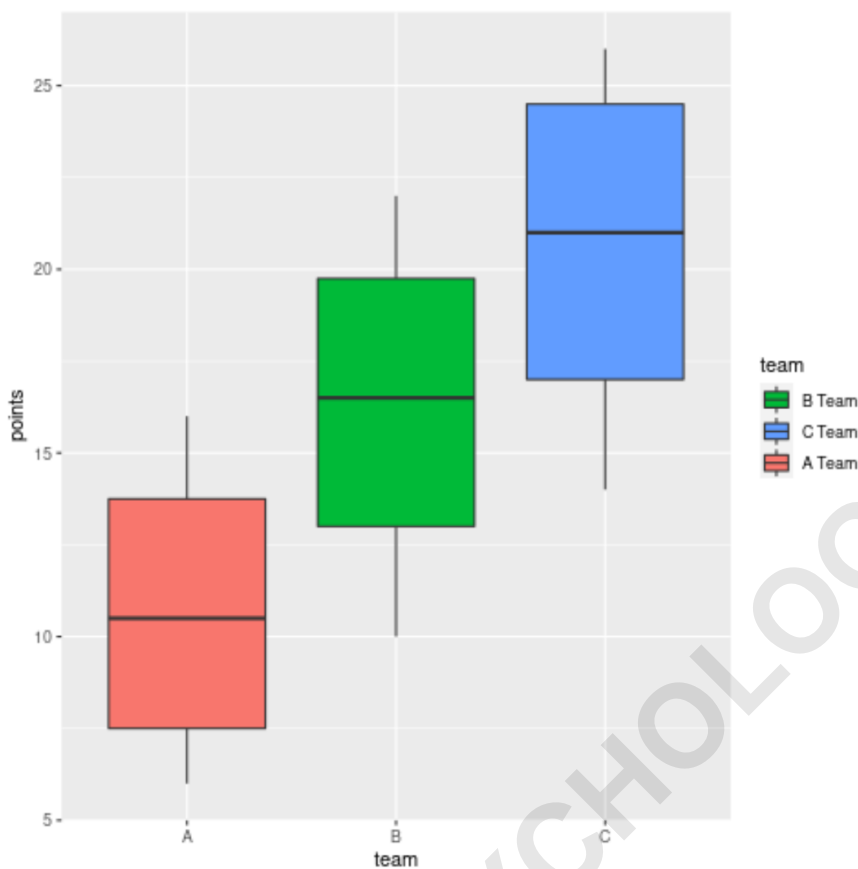
```
#create multiple boxplots to visualize points scored by team
```

```
ggplot(data=df, aes(x=team, y=points, fill=team)) +
```

```
geom_boxplot() +
```

```
scale_fill_discrete(breaks=c('B', 'C', 'A'),
```

```
labels=c('B Team', 'C Team', 'A Team'))
```



Notice that the legend labels have been updated to reflect the more descriptive titles ('B Team', 'C Team', 'A Team') while maintaining the custom order established by the `breaks` argument. This combined approach offers the ultimate flexibility in legend presentation, allowing for both reordering and relabeling within a single, coherent function call.

Advanced Considerations and Best Practices

When working with complex visualizations, remember that **ggplot2** provides various scale functions depending on the aesthetic being mapped and the variable type. If you are mapping colors to a discrete variable, you use `scale_color_discrete()`. If you are using continuous data and mapping size, you would use a continuous scale function, which manages legend entries differently. For manually ordering discrete aesthetics like `fill`, `color`, or `shape`, the use of the `breaks` argument is consistently the most reliable way to enforce a custom legend sequence.

Furthermore, always prioritize consistency. If you use the `breaks` argument to reorder the legend, consider whether the visual elements on the plot itself (e.g., bars or points) should also follow that order. If they should, it is highly recommended to preprocess the data by explicitly setting the

factor levels before plotting, ensuring that the visual mapping is intrinsically linked to the desired sequence.

Summary of Legend Reordering Techniques

To summarize the methods for controlling legend order in **ggplot2**, use the following guidelines:

For Legend-Only Reordering: Use `scale_fill_discrete(breaks = c(...))` or `scale_color_discrete(breaks = c(...))`. This is the fastest method when the X-axis order should remain the default or an independent order.

For Global Consistency: Preprocess the variable using factor manipulation (e.g., `fct_relevel` or base `levels()`). This ensures the order is consistent across all derived visualizations and statistical summaries.

For Custom Labels: Use the `labels` argument alongside `breaks` within the scale function to change the text displayed in the legend key while maintaining the desired sequence.

Mastery of these techniques allows R users to transform standard, automatically generated plots into custom, highly informative data visualizations that effectively guide the audience through the intended analytical narrative. Controlling the legend order is a fundamental step toward professional data reporting.

Further ggplot2 Resources

The following tutorials explain how to perform other common operations in **ggplot2**, helping you advance your data visualization skills:

Tutorial on modifying colors in ggplot2.

Guide to adjusting axis labels and titles.

Documentation on controlling facet wraps.