Learning to Customize Point Colors in ggplot2 Scatter Plots

Authored by

Mohammed looti

October 28, 2025

The ggplot2 package in R stands as the cornerstone for generating professional and statistically rigorous graphics. When producing data visualizations, especially scatter plots, the ability to precisely control the visual characteristics of individual data points is paramount for enhancing clarity and aesthetic impact. This comprehensive guide delves into the mechanisms within ggplot2 that allow users to meticulously customize both the border and the interior fill colors of points, enabling sophisticated dual-color styling for maximum visual effect.

To modify the appearance of points, you primarily interact with two distinct aesthetic arguments within the `geom_point()` function: **`color`** and **`fill`**. The **`color`** argument is universally applied to define the outline or border color of the point geometry. Conversely, the **`fill`** argument is dedicated exclusively to altering the interior color of the point. This critical separation enables the creation of complex, visually layered points--a powerful feature for distinguishing data characteristics or simply improving the plot's design.

It is essential to recognize that utilizing both **`color`** and **`fill`** simultaneously is dependent on a specific condition: the type of point `shape` selected. If the selected shape does not possess a distinct internal area, the **`fill`** argument will be ignored, and the **`color`** argument will define the appearance of the entire point. Understanding this dependency is the key to successfully implementing dual-color styling in your visualizations.

Consider the foundational example below, which demonstrates the basic application of defining both a border and fill color for points in a ggplot2 visualization, using a shape that supports dual aesthetics:

```
#create scatter plot with points that have black border and pink fill
ggplot(df, aes(x=x, y=y)) +
geom_point(color='black', fill='pink', shape=21)
```

The Critical Role of Point Shapes for Dual Aesthetics

The ability to apply distinct **`color`** (border) and **`fill`** (interior) attributes via `geom_point()` is entirely contingent upon the value assigned to the `shape` aesthetic. Only a specific subset of point shapes within the `ggplot2` library is engineered to handle two separate color inputs.

For successful dual coloring, the `shape` argument must be set to an integer value ranging from 21 to 25, inclusive. These five shapes--which include filled circles, squares, diamonds, triangles up, and triangles down--are uniquely structured with an explicit internal area that is separate from the bounding outline. This separation allows the **`fill`** argument to color the interior while the **`color`** argument handles the perimeter.

In contrast, shapes numbered 0 through 20 are generally defined by a single aesthetic property.

For these standard shapes (e.g., solid circles, crosses, or open squares), only the `color` argument is utilized. If you attempt to use `fill` on a shape like 1 (empty circle), the `fill` setting will be silently ignored, and the entire point will be rendered according to the `color` value. Therefore, selecting the correct `shape` is a fundamental requirement for achieving the desired border and interior color control.

Example 1: Specifying a Uniform Fill and Border Color for All Points

This first practical illustration details the process of applying a consistent, static border and interior color to all data points within a ggplot2 scatter plot. This method is typically used when the points do not represent separate categories, or when the overall visual cohesion of the plot is prioritized over conveying categorical information through color variation.

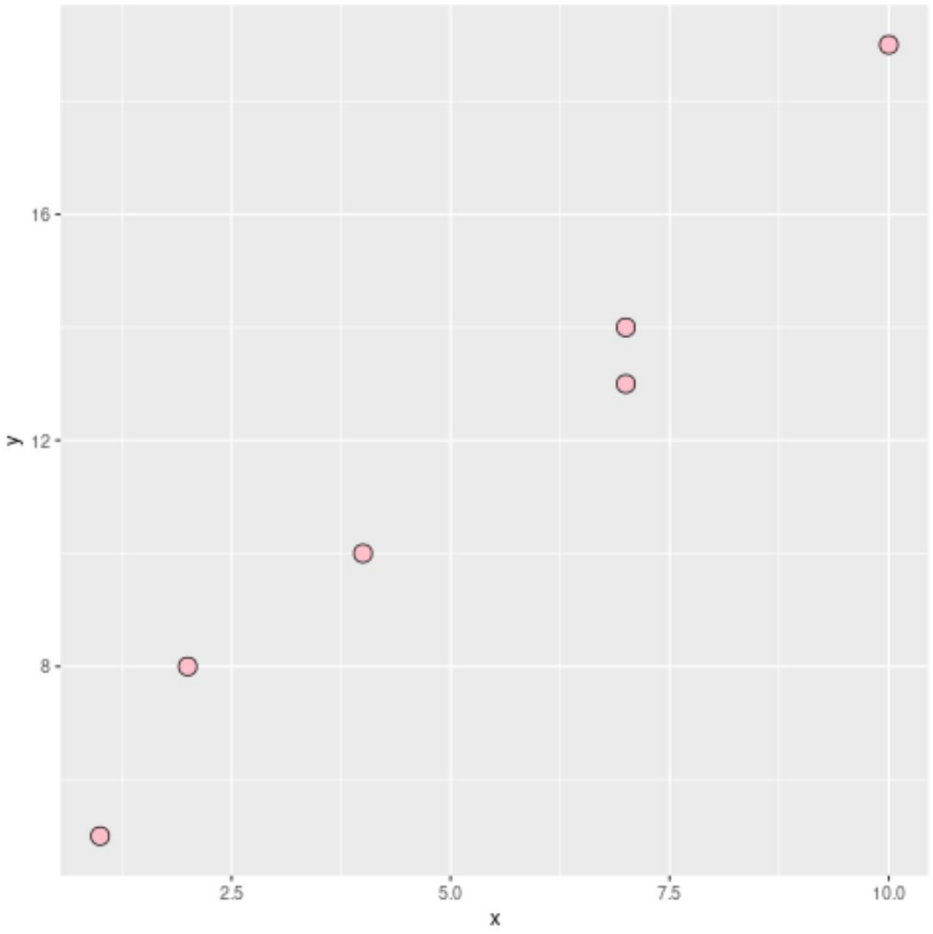
To begin, we must load the necessary `ggplot2` library and prepare a simple data.frame to house our coordinates. Within the plotting command, the crucial step involves placing the `color` and `fill` arguments outside of the `aes()` function, treating them as static parameter settings rather than mappings based on data variables.

library(ggplot2)

#create data frame
df <- data.frame(x=c(1, 2, 4, 7, 7, 10),
y=c(5, 8, 10, 14, 13, 19))

#create scatter plot
ggplot(df, aes(x=x, y=y)) +
geom_point(color='black', fill='pink', shape=21, size=4)

In the code block above, `ggplot(df, aes(x=x, y=y))` initializes the coordinate system. The subsequent `geom_point()` layer applies the styling: we set `color='black'` for a clear border and `fill='pink'` for the interior. We utilize `shape=21` (a filled circle) to ensure the dual coloring is effective, and `size=4` enhances visibility. This configuration results in a plot where every data point shares the exact same visual identity, presenting a clean and unified look, as visible in the plot image below.



While static styling is useful, the true power of data visualization often lies in mapping data variables to visual aesthetics. The next example demonstrates how to dynamically control the fill color based on categorical data, introducing a layer of informativeness to the plot.

Example 2: Specifying Multiple Fill and Border Colors for Points Based on a Grouping Variable

When data contains distinct categories or clusters, mapping these differences to visual properties greatly enhances interpretability. This example details the process of assigning varying fill colors to points based on a specific grouping variable in the dataset, while maintaining a consistent border color across all points. This technique is invaluable for effectively segmenting and highlighting different categories within your scatter plot.

We begin by constructing an expanded data.frame that includes a categorical column, `group`. The essence of this conditional styling lies in the use of `aes()` to map the `group` variable to the `fill` aesthetic. Subsequently, we employ `scale_fill_manual()` to gain explicit control over the color palette applied to these groups, ensuring optimal color choices for distinction.

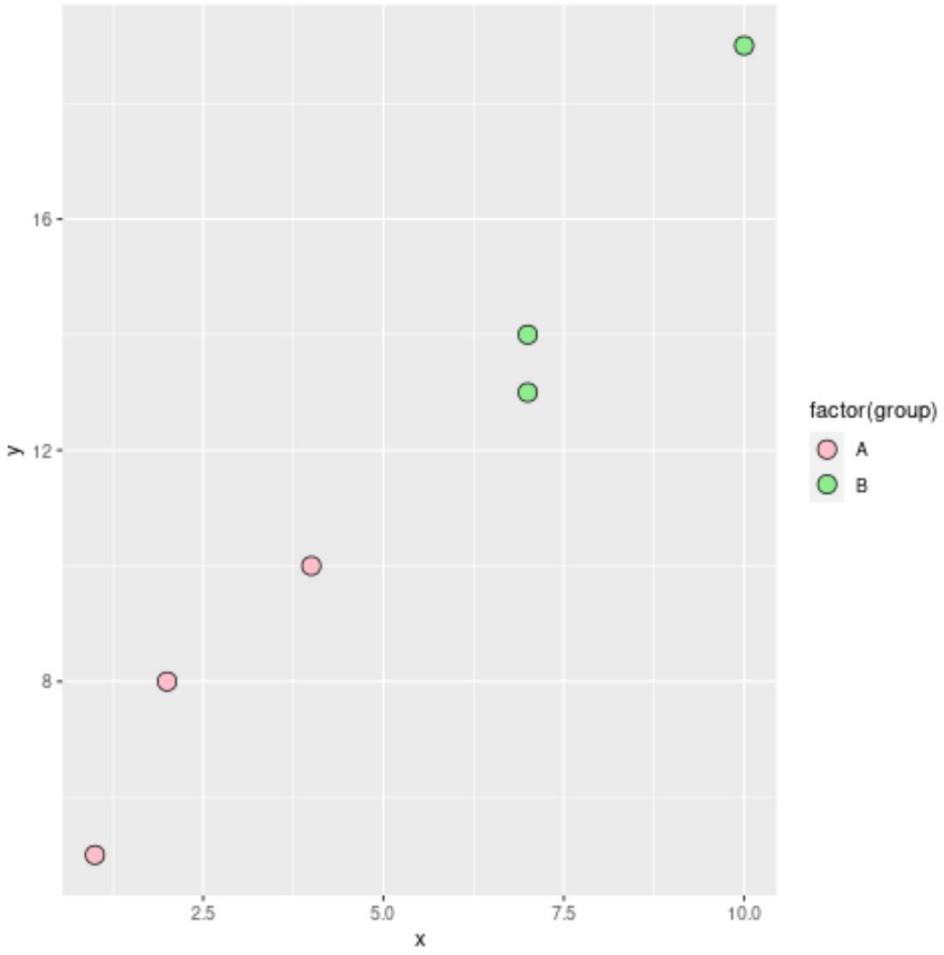
library(ggplot2)

#create data frame
df <- data.frame(x=c(1, 2, 4, 7, 7, 10),
y=c(5, 8, 10, 14, 13, 19),
group=c('A', 'A', 'A', 'B', 'B', 'B'))

#create scatter plot with multiple fill and border colors
ggplot(df, aes(x=x, y=y)) +
geom_point(color='black', shape=21, size=4, aes(fill=factor(group))) +
scale_fill_manual(values=c('pink', 'lightgreen'))

Within the `geom_point()` function, we set the static properties (`color='black'`, `shape=21`, `size=4`) outside of `aes()`. Crucially, we use `aes(fill=factor(group))` to dynamically map the `group` variable to the fill aesthetic. Wrapping `group` in `factor()` ensures that `ggplot2` treats the variable as discrete categories, which is necessary for discrete color assignment.

The final component, `scale_fill_manual(values=c('pink', 'lightgreen'))`, is essential for defining the specific colors associated with each factor level. Here, we manually assign 'pink' to the first level (A) and 'lightgreen' to the second level (B). This command overrides the default color palette and provides precise control, resulting in the informative visualization shown below, where the visual grouping is immediately apparent based on the interior point color.



Advanced Customization Considerations

Beyond the fundamental use of **`color`** and **`fill`**, ggplot2 offers a rich suite of aesthetic mappings and functions that facilitate even more nuanced point styling. One particularly valuable aesthetic is **`alpha`**, which controls the transparency of points. Using **`alpha`** within `geom_point()` is critical in dense scatter plots where numerous points overlap, helping to mitigate overplotting and reveal underlying density distributions.

Furthermore, while `scale_fill_manual()` provides absolute control over color assignment, ggplot2 offers automated scaling functions that leverage tested color palettes. Functions such as `scale_fill_brewer()` allow access to the popular ColorBrewer palettes, which are optimized for visual effectiveness and colorblind accessibility. For continuous data or for discrete data where perceptual ordering is important, functions like `scale_fill_viridis_d()` provide perceptually uniform color scales.

Combining diverse aesthetic mappings--such as mapping a categorical variable to **`fill`**, a continuous variable to **`size`**, and another variable to **`alpha`**--allows for the simultaneous

representation of three or more data dimensions within a single plot. Mastering these combinations is key to unlocking the full potential of ggplot2, enabling the creation of visualizations that are both beautiful and analytically powerful.

Conclusion and Further Learning

Achieving mastery over point aesthetics in ggplot2--specifically understanding the distinct roles and dependencies of the `color` and `fill` arguments--is a transformative skill for any data analyst. By diligently selecting appropriate point shapes (21-25) and applying conditional styling based on grouping variables, you can elevate standard scatter plots into sophisticated tools for data communication.

The inherent flexibility and layered structure of ggplot2 provide numerous avenues for tailoring visualizations to precise analytical needs and audience demands. Always prioritize the clarity and interpretability of your graphics; aesthetic choices should amplify the data's narrative, ensuring the message is received without confusion.

We encourage continuous learning and experimentation with the comprehensive suite of tools available in the package. Exploring supplementary tutorials and documentation covering advanced layering techniques, statistical transformations, and custom scaling options will further refine your expertise in creating compelling and effective graphical representations of complex data relationships.

Additional Resources

Tutorial on mapping different variables to different aesthetics in ggplot2.

Guide to customizing axis labels and titles in ggplot2 plots.

Exploring various themes and custom styling options for ggplot2 visualizations.

Understanding statistical transformations and smoothers in ggplot2.

How to create faceted plots to display subgroups in ggplot2.