Learning Plot Composition in R: Combining ggplot2 Objects with the patchwork Package

Authored by

Mohammed looti

November 13, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning Plot Composition in R: Combining ggplot2 Objects with the patchwork Package*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=24211

The Challenge of Plot Composition in R

When conducting thorough data visualization and statistical analysis, researchers frequently need to present several related graphical outputs simultaneously. Displaying multiple charts, such as different types of scatterplots, histograms, or box plots, in a single, cohesive figure is crucial for effective storytelling and comparison. Historically, achieving clean and professional multi-panel graphics in R programming language required complex solutions or external packages that often involved cumbersome layout matrices.

Fortunately, the process has been streamlined significantly with modern tools. One of the most intuitive and powerful methods for combining various ggplot2 objects into a singular, integrated graphic is by utilizing the **patchwork** package. This package, developed specifically for R, treats plots like modular components that can be assembled using simple mathematical operators.

The core benefit of the **patchwork** package is its straightforward syntax. It replaces complicated grid specifications with simple arithmetic symbols, allowing developers and analysts to focus more on the visualization content and less on the arrangement mechanics. This approach enables rapid iteration and creation of sophisticated layouts that would otherwise be difficult to manage.

Setting Up the Environment and Installation

To begin leveraging the power of plot composition, you must first ensure that the necessary packages are installed and loaded into your R session. The **patchwork** package works exclusively with graphics generated by **ggplot2**, which is the foundational visualization library in the tidyverse ecosystem. If you do not already have **ggplot2** installed, you should install it first.

The **patchwork** package itself is available on CRAN. If you are using R for the first time or if you have not previously installed this specific tool, you can install it using the standard `install.packages()` function directly in your R console. The command is concise and requires no additional parameters, ensuring a quick setup process:

install.packages('patchwork')

Once the installation is complete, you must load both the **ggplot2** library (to create the individual plots) and the patchwork package (to combine them) using the `library()` function. This step makes all the functions and operators provided by the packages available for use within your current R session, setting the stage for generating and arranging your visualizations effectively.

Preparing Data and Generating Core ggplot2 Objects

To demonstrate the functionality of **patchwork**, we will use a well-known, built-in R dataset: **mtcars**. This dataset originates from the 1974 Motor Trend US magazine and contains comprehensive measurements on 11 different attributes for 32 different automobiles. It provides a robust foundation for creating various scatterplots that we can then arrange.

Before diving into the visualization process, it is good practice to examine the structure of the data. We can quickly inspect the first few observations of the mtcars dataset using the **head()** function. This confirms the variables available for plotting, such as miles per gallon (mpg), cylinder count (cyl), displacement (disp), and horsepower (hp).

#view first 6 rows of mtcars dataset
head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

For our composition example, we will generate three distinct scatterplots. Each plot will use miles per gallon (mpg) as the primary dependent variable, paired with different independent measures. It is essential that we assign each generated plot object to a variable (e.g., `plot1`, `plot2`, `plot3`); this allows **patchwork** to reference and manipulate them easily during the combination phase. The three visualizations we aim to create are defined as follows:

Plot 1: Comparison of **mpg** vs. **wt** (weight)
Plot 2: Comparison of **mpg** vs. **disp** (displacement)
Plot 3: Comparison of **mpg** vs. **hp** (horsepower)

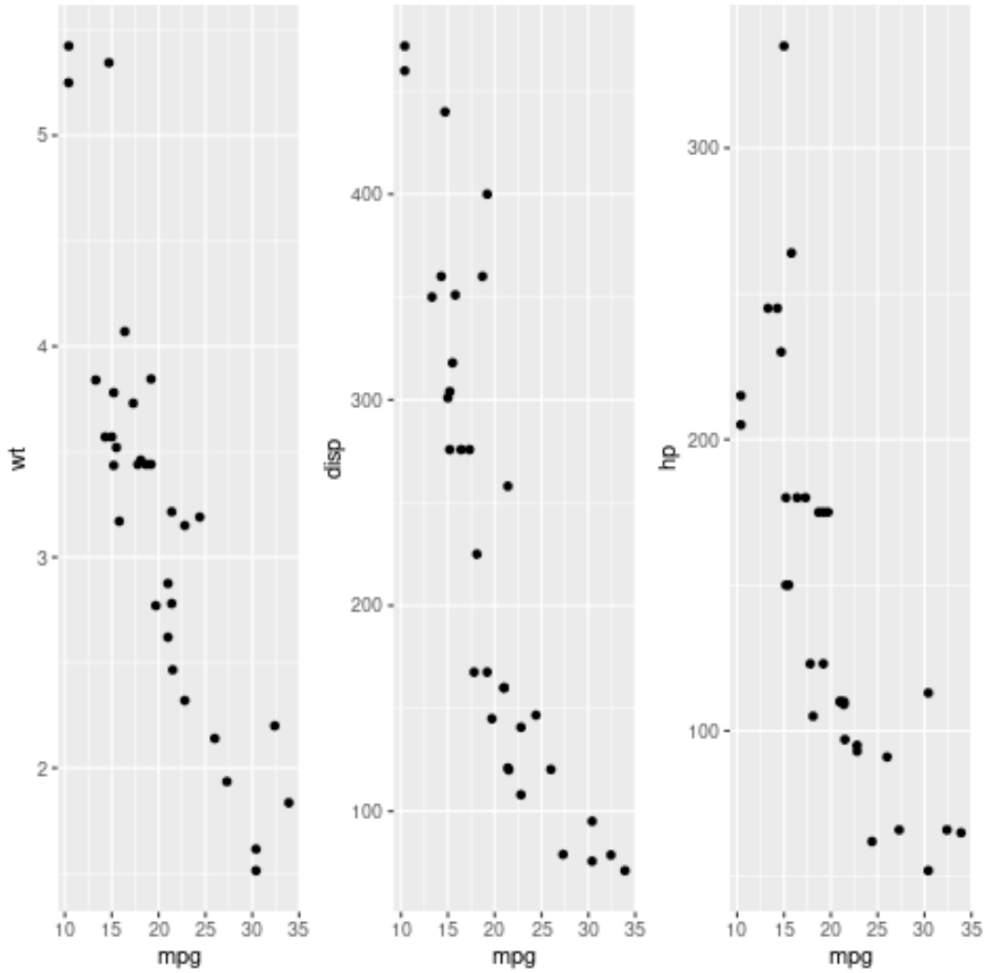
Arranging Plots Horizontally using + or |

Once the individual **ggplot2** objects are generated, the process of combining them into a single graphic is remarkably simple. **patchwork** utilizes two primary operators for horizontal arrangement: the plus sign (**+**) and the straight line (**|**). Both operators serve the same function, specifying that the subsequent plot should be placed adjacent to the preceding plot, maintaining a single row structure.

To display all three scatterplots side-by-side in one row, we simply list the plot variables separated by the **+** operator. This syntax is highly intuitive, making the code read almost like a natural language instruction to "add plot 2 next to plot 1, and plot 3 next to plot 2." The resulting composite plot automatically handles scaling and alignment across the panels.

```
library(ggplot2)
library(patchwork)

#generate three scatterplots
plot1 <- ggplot(mtcars, aes(mpg, wt)) +
geom_point()

plot2 <- ggplot(mtcars, aes(mpg, disp)) +
geom_point()

plot3 <- ggplot(mtcars, aes(mpg, hp)) +
geom_point()

#display all three scatterplots in same graphic using +
plot1 + plot2 + plot3
```

Executing the code above yields a graphic where all three visualizations are aligned horizontally, occupying equal width within the viewing area. This configuration is ideal for direct, row-wise comparisons of related data subsets.

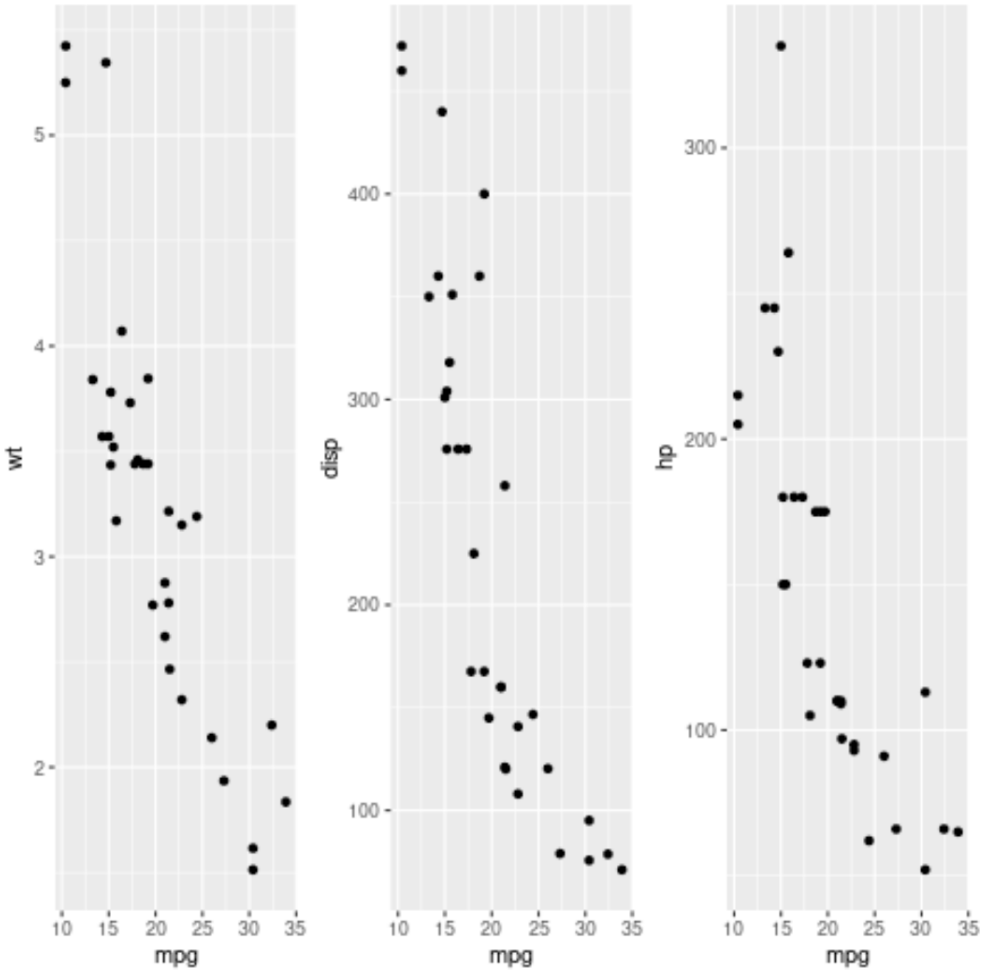


As an alternative to the plus sign, we can achieve an identical result by substituting the **|** (pipe) operator. While functionally equivalent for simple sequential arrangements, choosing one operator over the other often comes down to stylistic preference or clarity when dealing with more complex nested layouts, which we will discuss shortly.

```
library(ggplot2)
library(patchwork)

#generate three scatterplots
plot1 <- ggplot(mtcars, aes(mpg, wt)) +
geom_point()

plot2 <- ggplot(mtcars, aes(mpg, disp)) +
geom_point()

plot3 <- ggplot(mtcars, aes(mpg, hp)) +
geom_point()
```

```
#display all three scatterplots in same graphic using |
plot1 | plot2 | plot3
```

Using the pipe operator produces the exact same layout, confirming that both **+** and **|** are suitable for placing plots horizontally in a single row.



Creating Complex Grid Structures (Rows and Columns)

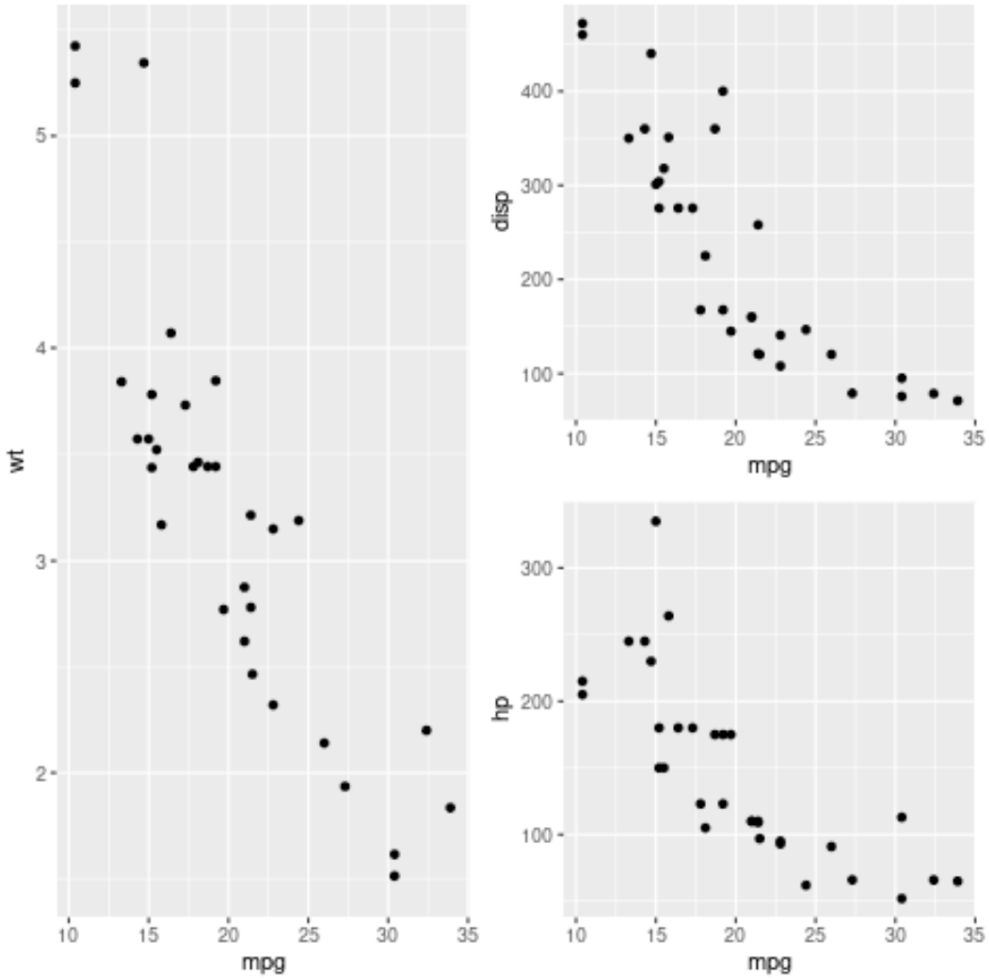
The true power of **patchwork** emerges when we need to combine horizontal and vertical arrangements to create multi-row grids. To indicate a vertical stack--that is, placing a plot below the preceding one, effectively starting a new row--we use the forward slash operator (*/*).

By combining the vertical operator (*/*) with the horizontal operators (**+** or **|**), we can define intricate layouts quickly. For instance, suppose we want `plot1` and `plot2` to remain side-by-side in the top row, but we want `plot3` to occupy a second row beneath `plot2`. The arrangement expression becomes intuitive: `plot1 | plot2 / plot3`.

The **patchwork** package interprets these operators based on standard order of operations, typically evaluating horizontal arrangements before vertical splits, unless parentheses are used for grouping. In the following example, the structure `plot1 | plot2` creates the first row, and the `/ plot3` command ensures that `plot3` starts a new column structure below the existing arrangement.

```
library(ggplot2)
library(patchwork)

#generate three scatterplots (definitions remain the same)
plot1 <- ggplot(mtcars, aes(mpg, wt)) +
geom_point()

plot2 <- ggplot(mtcars, aes(mpg, disp)) +
geom_point()

plot3 <- ggplot(mtcars, aes(mpg, hp)) +
geom_point()

#display plot1 and plot2 horizontally, with plot3 below plot2
plot1 | plot2 / plot3
```

This specific sequence produces a composite image where `plot1` and `plot2` share the top space, while `plot3` is relegated to the bottom left, spanning the width of the space it occupies beneath the combination of `plot1` and `plot2`.



By carefully selecting and combining the horizontal (**|** or **+**) and vertical (*/*) operators, you gain precise control over the layout. For instance, to stack all three plots vertically in a single column, the syntax would simply be `plot1 / plot2 / plot3`. The flexibility offered by these simple operators makes generating complex, publication-ready figures remarkably efficient.

Advanced Layout Control and Nesting

While simple operator chaining is effective, more sophisticated layouts often require explicit grouping to ensure the desired arrangement precedence. Just like in standard mathematics, parentheses **()** in **patchwork** are used to define groups of plots that should be treated as a single unit before being combined with other plots or groups. This is often referred to as nesting.

For example, if we want `plot2` and `plot3` to be stacked vertically, and then place that entire vertical stack next to `plot1` horizontally, we must enclose the vertical arrangement in parentheses: `plot1 | (plot2 / plot3)`. This ensures that the vertical division happens first, resulting in a two-column layout where the left column is `plot1` and the right column is the stacked unit of `plot2`

over `plot3`.

Furthermore, **patchwork** offers functions to fine-tune the aesthetics of the combined graphic. The `plot_layout()` function provides granular control over relative plot widths and heights, grid management, and spacing. Additionally, the `plot_annotation()` function allows you to add titles, subtitles, and captions that span the entire composite figure, which is essential for creating professional, self-contained graphics suitable for reports or publications. These advanced features ensure that the resulting patchwork is not just an arrangement of separate images, but a single, integrated visualization unit.

Conclusion and Further Resources

The **patchwork** package stands out as an indispensable tool for anyone working extensively with R and **ggplot2**. It radically simplifies the creation of multi-panel figures, replacing complex grid calculations with intuitive operators (**+**, **|**, ***/***) that mimic basic arithmetic. This ease of use, combined with powerful features for nesting and annotation, makes it the preferred method for generating cohesive and informative visualizations in R.

By mastering the use of these simple operators, you can efficiently transition from generating individual plots to constructing sophisticated, multi-layered dashboards and comparative figures. For those seeking to delve deeper into the full capabilities of plot arrangement, including shared axes, complex nesting weights, and thematic adjustments across panels, the official documentation serves as the most complete reference guide.

For comprehensive details, examples, and advanced usage scenarios, you can find the complete documentation for the **patchwork** package in R online.

The following tutorials explain how to perform other common tasks in ggplot2: