Learning to Create Side-by-Side Plots: A ggplot2 and Patchwork Tutorial

Authored by

Mohammed looti

November 7, 2025

In advanced data visualization, the ability to display multiple graphics simultaneously is frequently essential, allowing for direct comparison and the clear illustration of complex relationships between variables. When operating within the **R statistical environment**, the industry-standard ggplot2 package provides the powerful foundation for generating sophisticated, highly customized graphics. However, arranging these individual plots into a coherent, multi-panel layout traditionally posed a significant structural challenge. This need is elegantly resolved by the specialized patchwork package, which offers an intuitive, syntax-driven approach to combining visualizations seamlessly.

The `patchwork` package dramatically simplifies what was previously a complex scripting task, often requiring cumbersome functions like `grid.arrange` or `cowplot::plot_grid`. By leveraging straightforward arithmetic operators--specifically the **plus sign** (`+`) for combining plots horizontally and the **forward slash** (`/`) for vertical stacking--users can construct intricate layouts with exceptional ease. This tutorial serves as a comprehensive guide, walking through practical examples that harness the combined power of ggplot2 and patchwork. We will demonstrate how to produce polished, side-by-side plots suitable for academic reports or professional presentations, exploring arrangements from simple two-plot comparisons to complex multi-row structures, and concluding with methods for adding unifying annotations to the overall visualization.

The Necessity of Multi-Panel Visualization Layouts

Effective data storytelling relies heavily on showcasing comparative views of data distributions or relationships. Simply generating individual plots often fails to provide the necessary visual context required for drawing meaningful conclusions. This is precisely where **multi-panel visualization** becomes indispensable. By placing plots side-by-side or stacked vertically, researchers can rapidly contrast different subsets of data, compare various statistical models, or examine distinct visualization techniques applied to the same core data structure. The primary objective is not merely aesthetic arrangement, but rather the maximization of visual efficiency and analytical clarity for the audience.

The ggplot2 package, founded on the principles of the grammar of graphics, excels at creating highly customized and detailed plots. Every visualization generated by `ggplot2` is stored in the R environment as a self-contained object. Before the advent of dedicated layout packages, combining these objects demanded cumbersome manipulation of the R graphical environment using base tools or external packages that required manual calculation of plot dimensions and boundaries. The emergence of layout managers like patchwork revolutionized this process, enabling seamless integration using high-level commands that utilize operator overloading, making the syntax highly intuitive for R users.

The core philosophy driving patchwork is treating plot objects as variables that can be combined

using simple mathematical syntax. This approach inherently reduces the complexity associated with common layout challenges such as alignment, spacing consistency, and dynamic resizing, ensuring that the resulting composite visualization maintains both visual harmony and professional quality. Before we delve into specific examples of plot arrangement, it is necessary to ensure the foundational packages required for this workflow are correctly installed and loaded into the active R session environment.

Foundation: Setting Up the R Environment with Required Packages

To successfully replicate the examples provided in this guide, two primary packages must be accessible in your R environment: ggplot2 for the actual plot creation, and patchwork for the sophisticated plot arrangement. If these packages are not already installed, the following commands must be executed using the `install.packages()` function. This utility retrieves the necessary package files from the **Comprehensive R Archive Network (CRAN)** and places them in the appropriate library directory on your system.

Once installation is complete, the packages must be loaded into the active R session using the `library()` command. Loading a package makes all its internal functions, datasets, and operators--including the specialized `+` and `/` operators defined by `patchwork`--available for immediate use. It is critical to load both packages before proceeding with the plot creation and combination steps. The following code block details the necessary setup steps for configuring your workspace:

```
#install ggplot2 and patchwork packages
install.packages('ggplot2')
install.packages('patchwork')

#load the packages
library(ggplot2)
library(patchwork)
```

With the necessary environment configured, we can now proceed to explore how to apply these packages to construct various multi-plot layouts, beginning with the most fundamental arrangement: placing two plots side-by-side. All ensuing examples will utilize the R built-in iris dataset, a classic resource for demonstrating statistical and graphical techniques, ensuring consistency across all visualizations presented here.

Example 1: Juxtaposing Two Distinct Visualizations Horizontally

A frequent requirement in exploratory data analysis (EDA) is the need to view two different graphical representations of the same underlying data structure simultaneously. For this initial

example, we will generate two distinct visualizations of the iris dataset: a box plot illustrating the distribution of Sepal Length across different species, and a density plot showing the overall Sepal Length distribution. The side-by-side display of these two chart types facilitates the immediate comparison of **central tendency** (from the box plot) and **distribution shape** (from the density plot).

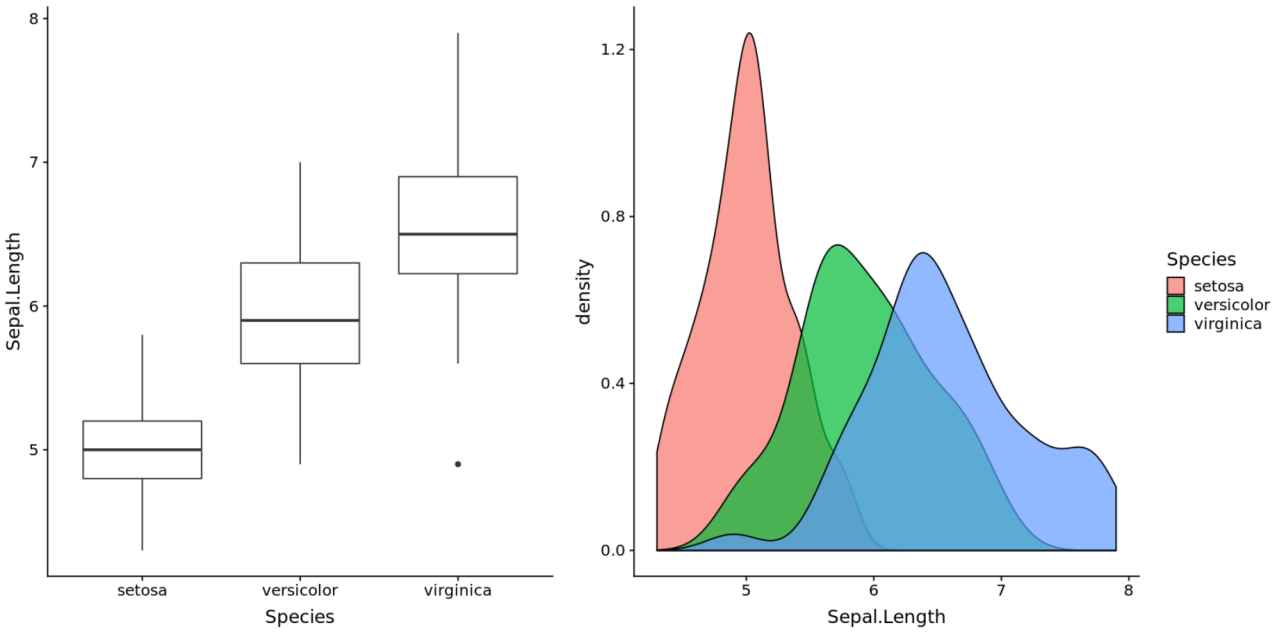
The code below first defines two plot objects, `plot1` and `plot2`, using standard ggplot2 syntax. `plot1` employs `geom_boxplot()` to visualize quartiles and potential outliers, while `plot2` utilizes `geom_density()`, mapping the fill color to the species variable to clearly delineate the distributions. Both are complete, self-contained visualization elements. The critical step is the final line, where the simple **addition operator** (`+`) is used to combine them. This operator, specially overloaded by the patchwork package, instructs R to automatically arrange the plots horizontally, ensuring perfect alignment and equal sizing within the designated plotting device.

```
#create box plot
plot1 <- ggplot(iris, aes(x = Species, y = Sepal.Length)) +
geom_boxplot()

#create density plot
plot2 <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
geom_density(alpha = 0.8)

#display plots side by side
plot1 + plot2
```

Executing this combined command produces a single, cohesive graphic where the two constituent plots are seamlessly displayed along a horizontal axis. This immediate visual juxtaposition is immensely valuable, for instance, in confirming that the species identified as having a larger median Sepal Length in the box plot also exhibits a density curve shifted further to the right in the density plot. The resulting image below illustrates this successful two-panel arrangement, showcasing the simplicity and power of the `patchwork` syntax for creating professional layouts.



Example 2: Arranging Multiple Plots in a Single Horizontal Row

The utility of the `patchwork` package is easily scalable, accommodating three or more plots in a single row or column without increased complexity. In this second example, we will scale up the visualization by adding a third plot--a scatterplot--to our existing box plot and density plot. This arrangement is highly practical when a unified visual field is needed to analyze three distinct facets of the same dataset, such as univariate distributions, bivariate relationships, and categorical summaries.

We retain the definitions for `plot1` (the box plot) and `plot2` (the density plot) from the previous section. We then introduce `plot3`, which uses `geom_point()` to create a scatterplot showing the relationship between Sepal Length and Sepal Width, a standard bivariate comparison in the iris dataset. The arrangement syntax remains extremely simple: by chaining the plots together using the additive operator (+) sequentially, `patchwork` automatically handles the necessary scaling and equal division of the output area. This chaining mechanism provides exceptional flexibility for quickly prototyping complex visual dashboards and reports.

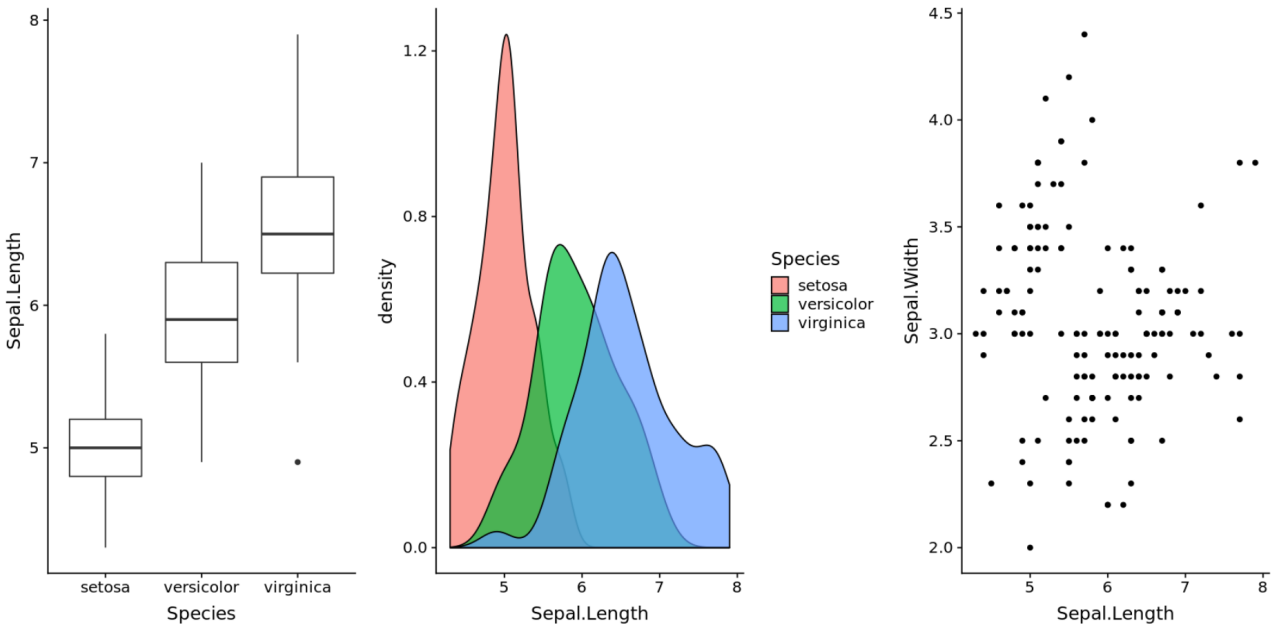
#create box plot
plot1 <- ggplot(iris, aes(x = Species, y = Sepal.Length)) +
geom_boxplot()

#create density plot
plot2 <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
geom_density(alpha = 0.7)

```
#create scatterplot
plot3 <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
geom_point()

#display three plots side by side
plot1 + plot2 + plot3
```

The resulting visualization efficiently packs three distinct analytical perspectives into a single row. This method offers a superior approach compared to displaying three separate images, as it forces the reader to acknowledge the simultaneous presence of the three perspectives on the data. patchwork automatically ensures that the plots are equally spaced and aligned along the horizontal plane, maintaining a clean and professional appearance, as evidenced by the output image below. This demonstrates the package's ability to handle increasing complexity without requiring manual configuration of viewport coordinates or layout dimensions.



Example 3: Constructing Stacked (Vertical) Plot Layouts with the Slash Operator

While horizontal arrangement is excellent for direct side-by-side comparison, vertical stacking is often the preferred layout when the component plots share a common x-axis, or when the overall aspect ratio of the combined graphic needs to be taller rather than wider. The patchwork package introduces the **forward slash operator** (`/`) specifically to manage this vertical arrangement, treating the division sign as a command to stack the plots one on top of the other. This simple

change in operator provides complete, algebraic control over the orientation of the final composite figure.

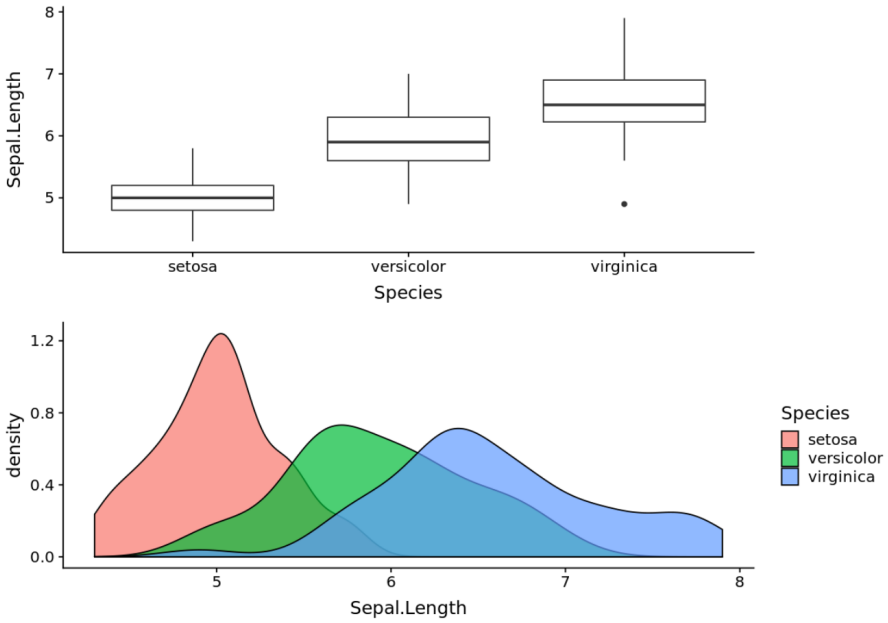
By using the same `plot1` (boxplot) and `plot2` (density plot) definitions from the previous examples, we can effortlessly switch from a horizontal display to a vertical stack. The vertical arrangement is especially effective for visual reporting where physical space constraints limit horizontal expansion, or where a clear, sequential flow from one stage of analysis to the next is desired. The key syntax manipulation involves replacing the `+` operator with the `/` operator. This small but fundamental change alters the layout structure handled by the package's underlying grid system.

```
#create box plot
plot1 <- ggplot(iris, aes(x = Species, y = Sepal.Length)) +
geom_boxplot()

#create density plot
plot2 <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
geom_density(alpha = 0.7)

#display plots stacked on top of each other
plot1 / plot2
```

When the `/` operator is used, the plots are placed in a single column, with `plot1` appearing above `plot2`. This simple, intuitive syntax confirms `patchwork` as an extremely powerful tool for quickly experimenting with different layout structures. Furthermore, these operators can be mixed to create sophisticated grid designs: for instance, `(plot1 + plot2) / plot3` would create a top row containing `plot1` and `plot2` side-by-side, stacked above a second row containing only `plot3`. The resulting stacked figure below demonstrates the clean output of the basic vertical arrangement, confirming its utility for creating tall, focused visualizations.



Example 4: Enhancing Presentation with Overarching Titles and Annotations

While individual plots often possess their own titles and axis labels, a complete composite figure requires an overarching narrative element that describes the entire visualization panel. The `patchwork` package provides the specialized function `plot_annotation()` for adding **global titles**, subtitles, and captions that span the full width or height of the combined graphic. These annotations are crucial for providing essential context, summarizing key findings, and citing sources for the entire collection of sub-plots, effectively transforming a set of individual charts into a single, comprehensive figure.

In this example, we first assign individual titles to `plot1` and `plot2` using the `ggtitle()` function within the ggplot2 definition. Next, we combine these plots horizontally and save the resulting composite object as `patchwork`. Finally, we apply `plot_annotation()` to this composite object. The arguments within `plot_annotation()`--specifically `title`, `subtitle`, and `caption`--allow the user to define high-level metadata that frames the entire visualization. This workflow ensures that the collective data story is immediately accessible to the audience, regardless of the granular details within the individual plots.

#create box plot
plot1 <- ggplot(iris, aes(x = Species, y = Sepal.Length)) +
geom_boxplot() +
ggtitle('Boxplot')

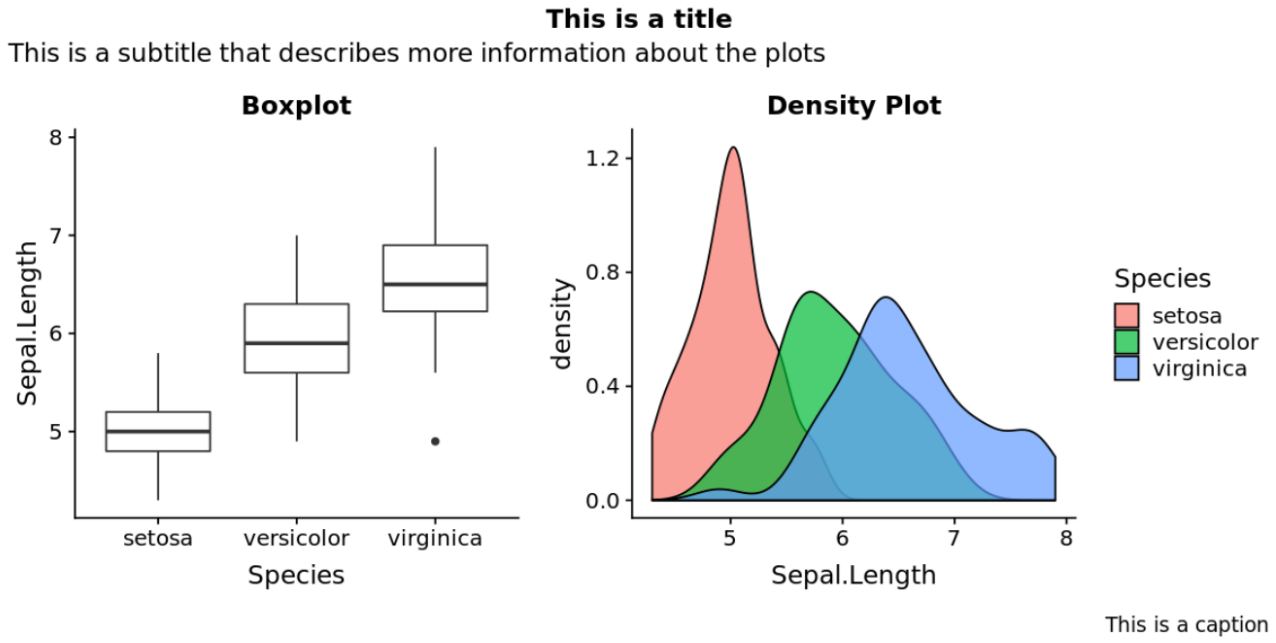
#create density plot

```
plot2 <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
geom_density(alpha = 0.7) +
ggtitle('Density Plot')

#display plots side by side with title, subtitle, and captions
patchwork <- plot1 + plot2

patchwork + plot_annotation(
title = 'This is a title',
subtitle = 'This is a subtitle that describes more information about the plots',
caption = 'This is a caption'
)
```

The resulting annotated figure clearly demonstrates how effectively global titles can bind multiple plots together conceptually. The main title provides the primary topic, the subtitle elaborates on the context or methodology, and the caption offers space for necessary source citation or technical notes. This level of professional annotation ensures that the composite visualization is self-contained and fully descriptive, meeting the highest standards for data reporting. The resulting image below showcases the final appearance of the annotated side-by-side layout.



Advanced Layout Control and Conclusion

Beyond the basic horizontal and vertical arrangements using `+` and `/`, the patchwork package provides sophisticated control over complex grid structures. For instance, users can manually

specify the number of columns (`ncol`) or rows (`nrow`) using the `plot_layout()` function, allowing for granular command over plot dimensions and relative placement within the figure. Furthermore, the use of parentheses `()` is crucial for grouping plots, enabling specific subsets of visualizations to be treated as a single unit before being combined with others. This technique allows for the creation of complex, nested layouts often necessary for detailed academic poster presentations or dashboard design.

The ability to use simple algebraic operators to define complex layouts is the defining strength of `patchwork`. This algebraic structure saves countless hours of manual graphic manipulation and ensures consistency across all visualizations generated within an analysis workflow. By integrating these design elements directly into the R code, the package strongly promotes **reproducibility** and scalability in large-scale data visualization projects.

In conclusion, combining the graphical prowess of ggplot2 with the layout simplicity of `patchwork` provides R users with a world-class toolset for creating clean, multi-panel visualizations. Whether arranging two simple comparison plots or assembling an intricate, annotated dashboard, this combination ensures that data presentation is both technically sound and visually compelling. Mastering these techniques is fundamental for anyone seeking to communicate complex data insights effectively in the modern statistical environment.

You can find more R tutorials here.