

# How to Use ggsave to Quickly Save ggplot2 Plots

Authored by  
**stats writer**

November 19, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Use ggsave to Quickly Save ggplot2 Plots*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=97100>

The process of creating compelling data visualizations using **R** is often streamlined by the powerful **ggplot2** package. While **ggplot2** excels at generating graphics within the R environment, the subsequent step--saving these plots for publication, reports, or sharing--requires an efficient tool. This is where the **ggsave()** function becomes indispensable.

**ggsave()** is an integral component of the **ggplot2 R package**, specifically designed to quickly and reliably export an existing **ggplot2** visualization to a file format. It manages the complexities of file type detection, scaling, and resolution automatically, allowing users to focus on the analysis rather than the tedious details of exporting graphics. Whether you need a high-resolution **PNG**, a versatile **JPEG**, or a scalable vector graphic like **PDF**, **ggsave()** handles it with minimal arguments.

Unlike methods that rely on external graphics devices or manual resizing processes, **ggsave()** automatically infers the desired output type and dimensions based on the filename extension and specified parameters. By taking the plot object name and the desired output file path as its primary arguments, this function automates the creation of the image file, providing a seamless transition from the R console to a finalized visual asset ready for integration into documents or presentations. Its efficiency makes it a favorite among data scientists and analysts utilizing the **R** environment for statistical plotting.

## Understanding the ggplot2 Ecosystem

To appreciate the utility of **ggsave()**, it is helpful to understand its context within the broader **ggplot2** visualization framework. The design philosophy of **ggplot2**, based on the Grammar of Graphics, separates the data, aesthetic mappings, geometric objects, and coordinate systems. This structured approach results in highly customizable and consistent visualizations.

When you execute a **ggplot2** command, **R** generates a plot object. This object holds all the necessary information--data, layers, themes, and scales--required to render the final image. When working interactively in the R console or RStudio, the plot is usually rendered immediately to the screen or the designated plots pane. However, to persist this visualization outside of the R session, it must be explicitly saved.

Prior to the widespread use of specialized functions like **ggsave()**, saving plots in R often involved opening a graphics device (e.g., `pdf()` or `png()`), printing the plot object to that device, and then explicitly closing the device using `dev.off()`. This multi-step process was prone to user error and required careful management of resolution and scaling. **ggsave()** streamlines this entire operation into a single, highly intuitive function call, greatly improving workflow efficiency for those producing numerous plots.

You can use the **ggsave()** function to quickly save plots created by **ggplot2**.

This function employs a robust set of default values, meaning that for most standard plots, specifying only the filename is sufficient. The function is designed to intelligently capture the last plot displayed, making rapid iteration and saving simple. The syntax below shows the complete set of arguments available:

## Core Syntax and Function Arguments

```
ggsave(  
filename,  
plot = last_plot(),  
device = NULL,  
path = NULL,  
scale = 1,  
width = NA,  
height = NA,  
units = c("in", "cm", "mm", "px"),)  
...  
)
```

Each argument within the **ggsave()** function serves a specific purpose, allowing fine-grained control over the output characteristics of the saved graphic. Understanding these arguments is key to leveraging the function's full power for both basic and advanced saving scenarios. The following list breaks down the purpose of each key parameter:

## Detailed Explanation of Key ggsave Parameters

**filename:** This is a mandatory parameter that specifies the desired file name, including the extension. The extension (e.g., `.pdf`, `.png`, `.tiff`) dictates the output format, which **ggsave()** automatically detects and handles. For example, specifying `"report_figure.pdf"` will save the plot as a **PDF**.

**plot:** This argument accepts the name of the specific plot object you wish to save. By default, it is set to `last_plot()`, which saves the most recently displayed `ggplot2` object. This is highly convenient for interactive sessions where you only need to run **ggsave()** immediately after generating the plot.

**device:** Used to manually specify the graphics device (e.g., `"png"`, `"jpeg"`, `"svg"`) if you do not wish for **ggsave()** to infer it from the `filename` extension. Although optional, explicitly setting the device can be useful in specialized scripting environments or when dealing with unusual file naming conventions.

**path:** This optional parameter allows the user to define the directory where the file should be

saved. If `NULL`, the file is saved to the current working directory. Specifying a path (e.g., `"~/Documents/Plots/"`) is crucial for organizing output in larger projects.

**scale:** The multiplicative scaling factor applied to the plot dimensions. A value of `1` (the default) uses the size of the current graphics device (or the specified `width/height`). Setting `scale = 2`, for instance, doubles the size, which is a common technique to effectively increase the resolution of the output image without changing the `width/height` parameters.

**width:** Specifies the width of the plot in the units defined by the `units` argument. If not specified (`NA`), the width of the current graphics device is used.

**height:** Specifies the height of the plot in the units defined by the `units` argument. Like `width`, if left as `NA`, the height of the current graphics device is used.

**units:** Defines the unit system for the `width` and `height` arguments. Standard options include `"in"` (inches), `"cm"` (centimeters), `"mm"` (millimeters), or `"px"` (pixels). Inches are often the default choice for print-quality graphics.

## Preparing the Visualization: The Sample Data and Plot

The following examples show how to use the `ggsave()` function in practice to save the scatter plot created below. We first need to set up our environment and generate the plot object.

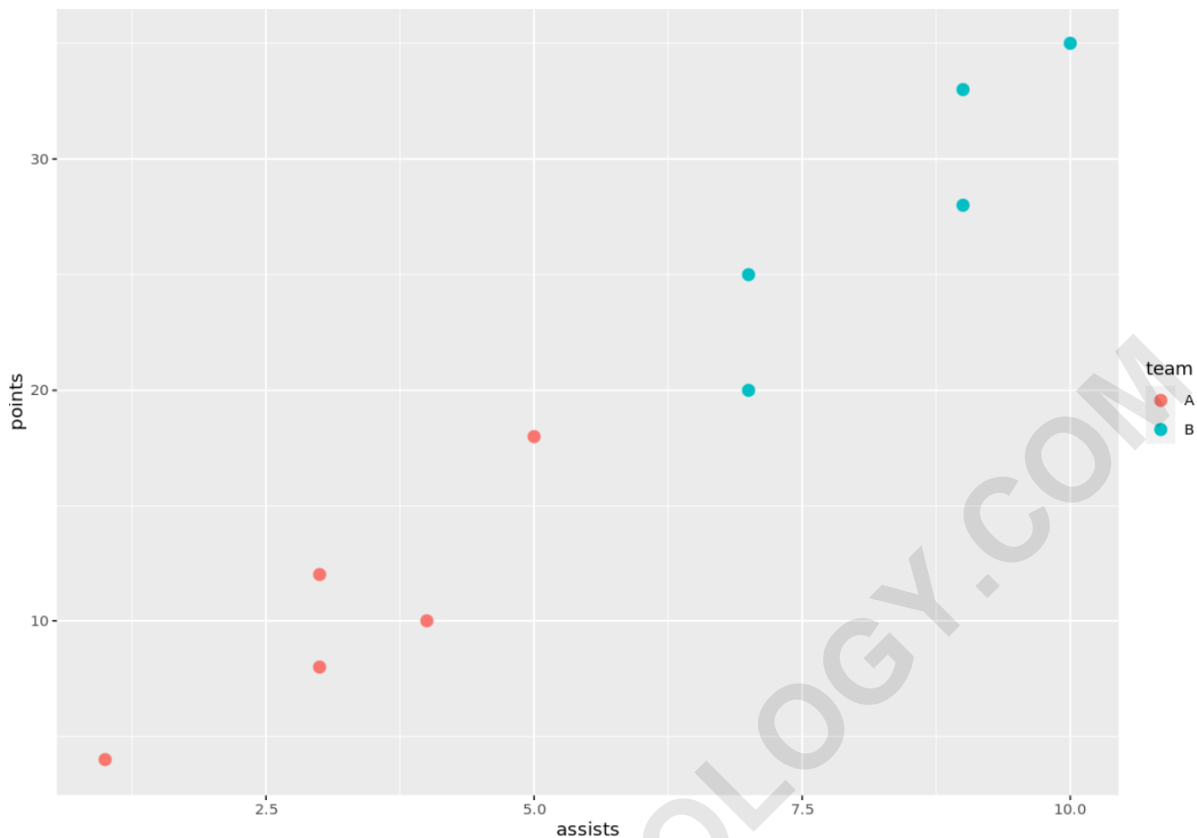
To demonstrate the practical application of the `ggsave()` function, we will first generate a simple scatter plot using the [R package ggplot2](#). This example showcases a basic data frame and the necessary code to create a visualization that will then be saved using various `ggsave()` configurations.

### library(ggplot2)

```
#create data frame
df <- data.frame(team=rep(c('A', 'B'), each=5),
  assists=c(1, 3, 3, 4, 5, 7, 7, 9, 9, 10),
  points=c(4, 8, 12, 10, 18, 25, 20, 28, 33, 35))
```

```
#create scatter plot
ggplot(df, aes(x=assists, y=points)) +
  geom_point(aes(color=team), size=3)
```

Executing the code above generates the scatter plot visualization. Note that because we did not explicitly assign the plot to an object, `ggplot2` automatically prints it to the current graphics device, making it the `last_plot()` available for `ggsave()` to capture. The resulting visualization is shown below:



### Example 1: Saving Plots Using Default Settings

The simplest and most common usage of **ggsave()** involves relying entirely on the default parameters. When only the `filename` is provided, the function automatically saves the `last_plot()` to the current working directory, using the dimensions of the active graphics device.

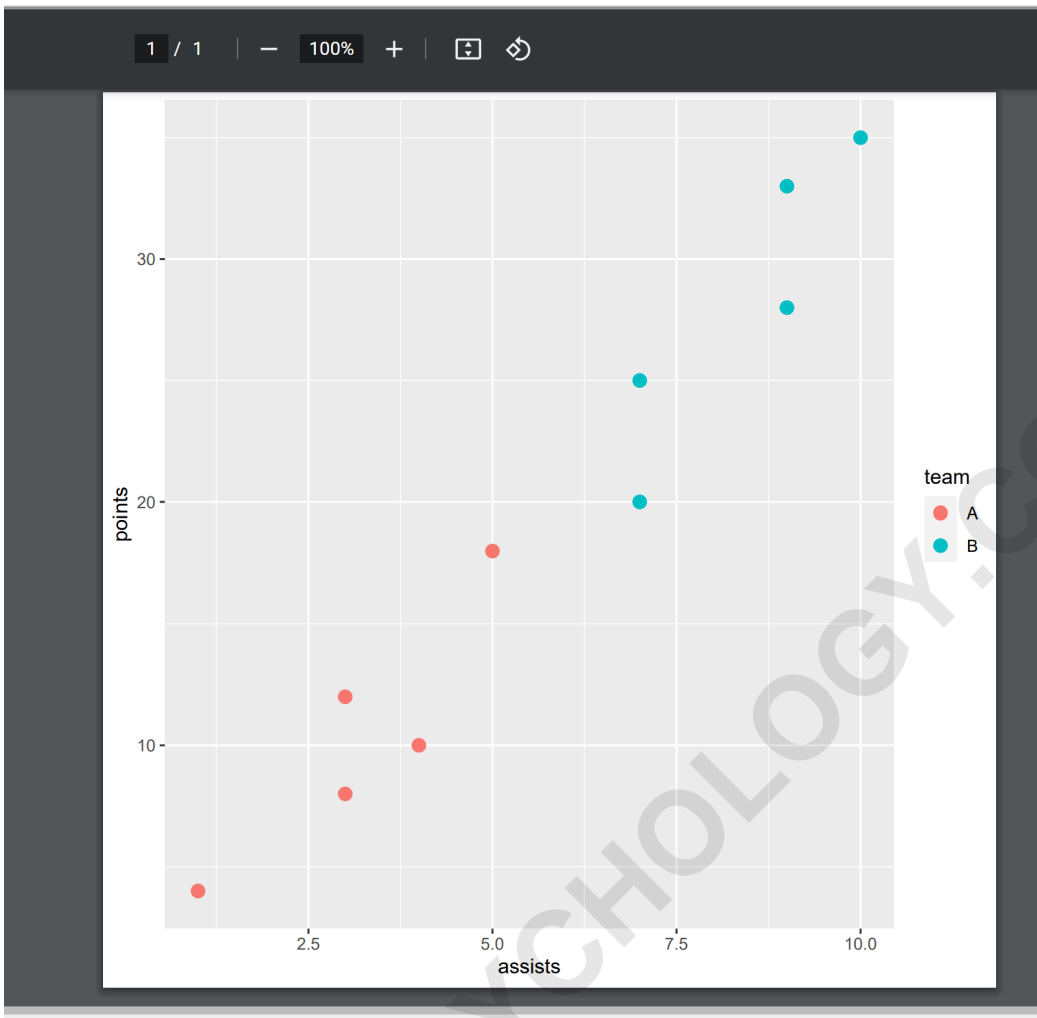
We can use the following syntax with **ggsave()** to save this scatter plot to a PDF file called **my\_plot.pdf** with all of the default settings:

```
library(ggplot2)
```

```
#save scatter plot as PDF file  
ggsave('my_plot.pdf')
```

Since we didn't specify a path or a size for our plot, the scatter plot will simply be saved as a PDF in the current working directory with the size of the current graphics device. This ensures the output maintains the exact aspect ratio and physical size of the visualization pane at the moment the function is executed.

If I navigate to my current working directory, I can view the resulting PDF file:



I can confirm that the plot has been saved as a [PDF](#) file, accurately reflecting the size and appearance of the plot rendered by the current graphics device. This method is quick and efficient for standard outputs.

## Example 2: Implementing Custom Dimensions and Units

For scenarios requiring precise control over the output size, the `width`, `height`, and `units` parameters must be utilized. This is vital when preparing figures for journals or documents with strict formatting guidelines.

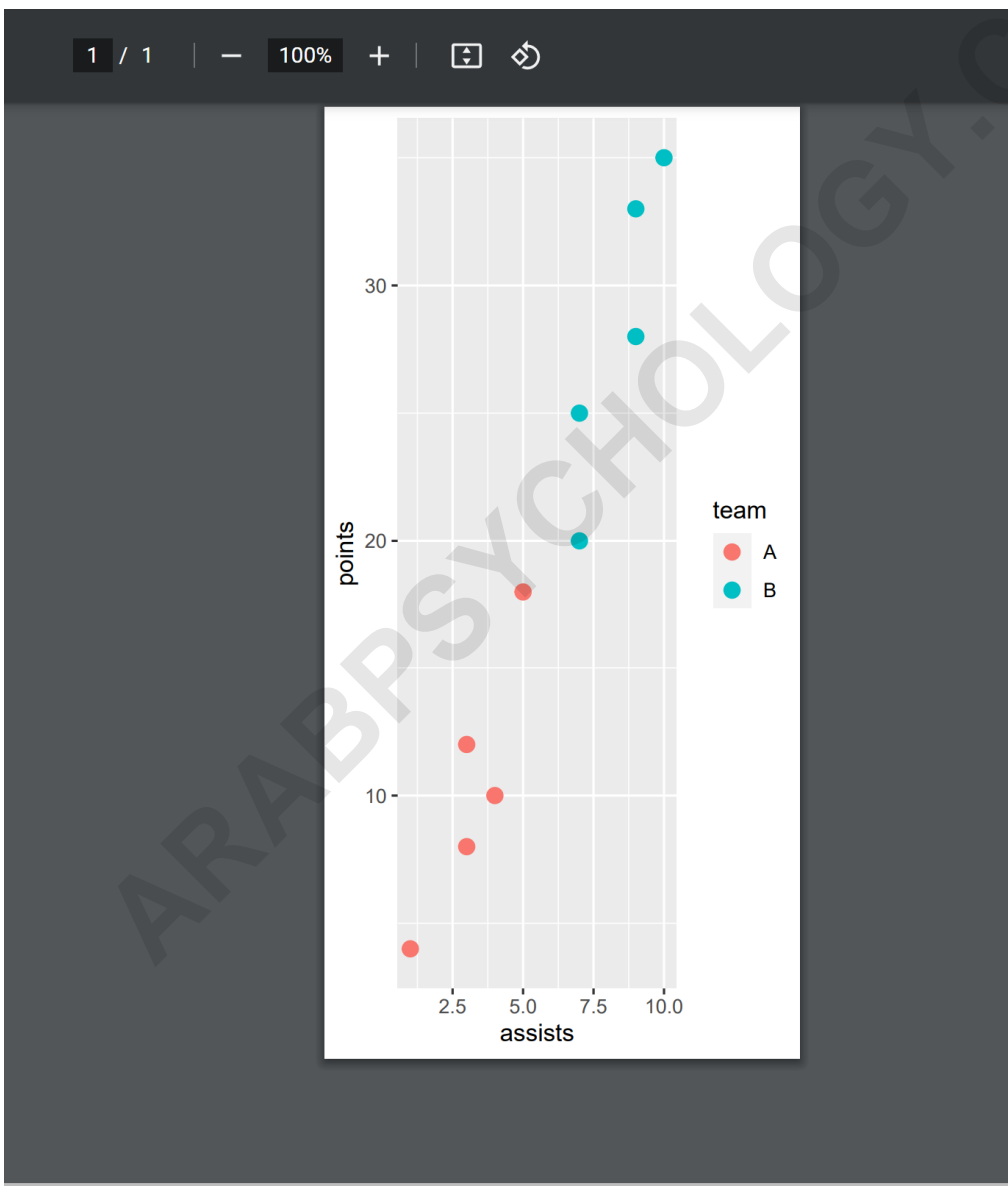
We can use the following syntax with `ggsave()` to save this scatter plot to a [PDF](#) file called `my_plot2.pdf`, but this time enforcing specific dimensions: 3 inches wide by 6 inches tall, resulting in a vertical layout.

```
library(ggplot2)
```

```
#save scatter plot as PDF file with specific dimensions  
ggsave('my_plot2.pdf', width=3, height=6, units='in')
```

By specifying these arguments, we override the default size and force the graphics device to render the plot exactly to the required size. This ensures consistency regardless of the interactive R session settings.

If I navigate to my current working directory, I can view the resulting PDF file, confirming the custom dimensions:



I can see that the plot has been saved as a PDF file with the dimensions that I specified (3x6 inches), demonstrating successful implementation of custom output sizing.

## Choosing the Right Output Device and File Format

A significant strength of **ggsave()** is its versatility in handling various output formats, or devices. While our examples focused on the PDF format, which is excellent for vector graphics and print quality, users frequently need raster formats for web or screen display.

**ggsave()** supports virtually all common graphic formats. To switch formats, simply change the file extension in the `filename` argument. For instance, replacing `'my_plot.pdf'` with `'my_plot.png'` will automatically instruct **ggsave()** to use the appropriate PNG device.

**Note:** In these examples we chose to save the plots from ggplot2 as **PDF** files, but you can also specify jpeg, png, or other file formats simply by changing the file extension. For high-resolution raster output suitable for printing, consider increasing the `scale` argument (e.g., `scale=2`) to effectively double the pixel density of the saved image.

## Conclusion and Summary

The **ggsave()** function provides a powerful, simplified interface for exporting high-quality visualizations generated by the ggplot2 R package. By abstracting the complexities of graphics device management, it allows users to quickly save plots with default settings or exercise meticulous control over output dimensions and resolution. This functionality is crucial for maintaining efficient workflows in data analysis projects.

We have demonstrated that whether you require quick default saving or precise dimensional control for publication, **ggsave()** delivers clean, accurate output. Always ensure you are aware of your current working directory when relying on default path settings, and utilize the `width`, `height`, and `units` parameters when integrating figures into formal documents.