Create an Empty Plot in R (3 Examples)

Authored by

Mohammed looti

October 30, 2025

Creating an empty plot is a fundamental yet often overlooked task in advanced R programming and data visualization. Unlike simply generating a plot from existing data, initiating an empty canvas provides complete control over the graphical environment, allowing for complex, layered visualizations or specialized annotations. This control is essential when building custom graphics that deviate from standard plotting functions, or when integrating multiple data sources onto a single coordinate system.

The need for an empty plot arises primarily in scenarios where the user intends to add graphical elements sequentially using low-level functions like **points()**, **lines()**, or **text()**. The initial empty plot serves to establish the boundaries, coordinate system, and overall visual parameters (like background color or aspect ratio) before any data is rendered. Understanding the three primary methods for generating these blank canvases is key to mastering the R Base Graphics system.

There are three common and distinct approaches available in R for creating a blank graphical output, each offering a different level of initial configuration:

Method 1: Create Completely Empty Plot (Utilizing **plot.new()** to generate a minimal, unconfigured device.)
Method 2: Create Empty Plot with Axes (Employing **plot(NULL, ...)** to define coordinates but suppress visual markings.)
Method 3: Create Empty Plot with Axes & Labels (Using **plot(NULL, ...)** to establish a full coordinate system with user-defined labels.)

We will examine the syntax and practical implications of each method, followed by concrete examples demonstrating their application within the RStudio environment.

Understanding R's Base Graphics Philosophy

Before diving into the specific commands, it is crucial to appreciate how the R Base Graphics system operates. This system is procedural, meaning that plots are built layer by layer. The first command executed--whether **plot.new()** or a standard **plot()** call--initializes the plotting device and sets the overall plotting region (the 'canvas') and the data region (where the data will actually reside). Subsequent commands then add elements to this established environment.

When we call a function like **plot(x, y)**, R automatically determines the appropriate coordinate limits (**xlim** and **ylim**) based on the range of the provided data, and it automatically draws the axes and labels. However, when creating an empty plot, we bypass this automatic scaling. We must manually define the coordinate range, even if no data points are initially present. This manual definition is crucial for accurately positioning subsequent graphical elements, ensuring they fall within the desired viewable area. If **xlim** and **ylim** are not explicitly set when using **plot(NULL, ...)**, the default values might be too restrictive or unsuitable for the intended visualization.

The choice between the three methods largely depends on whether the final visualization requires visible axes, tick marks, or descriptive labels. If the goal is a minimalist design or a highly specialized visual where standard axes would be distracting, **Method 1** or **Method 2** is appropriate. If the plot needs context for data interpretation, **Method 3** provides the necessary framework.

Method 1: The Completely Blank Canvas (plot.new())

The most straightforward way to generate a truly empty plotting device in R is by invoking the **plot.new()** function. This command initializes a new graphical frame but crucially does not establish a coordinate system in the traditional sense, nor does it draw any borders, axes, or labels. It simply opens the device and prepares it for drawing, offering the minimum possible configuration.

Since **plot.new()** does not define user coordinates (like x and y limits), any subsequent use of low-level drawing functions (such as **points()** or **lines()**) will require an explicit call to **plot.window()** immediately afterward to define the coordinate range. If **plot.window()** is omitted, these low-level functions will fail to draw correctly because R lacks the necessary context to map data values to physical screen coordinates. This method is typically reserved for highly customized graphical layouts where the user needs complete control over every pixel and boundary, or when integrating plots into complex multi-panel figures.

The code structure for this approach is minimal, providing the most foundational command in the R Base Graphics system for initialization:

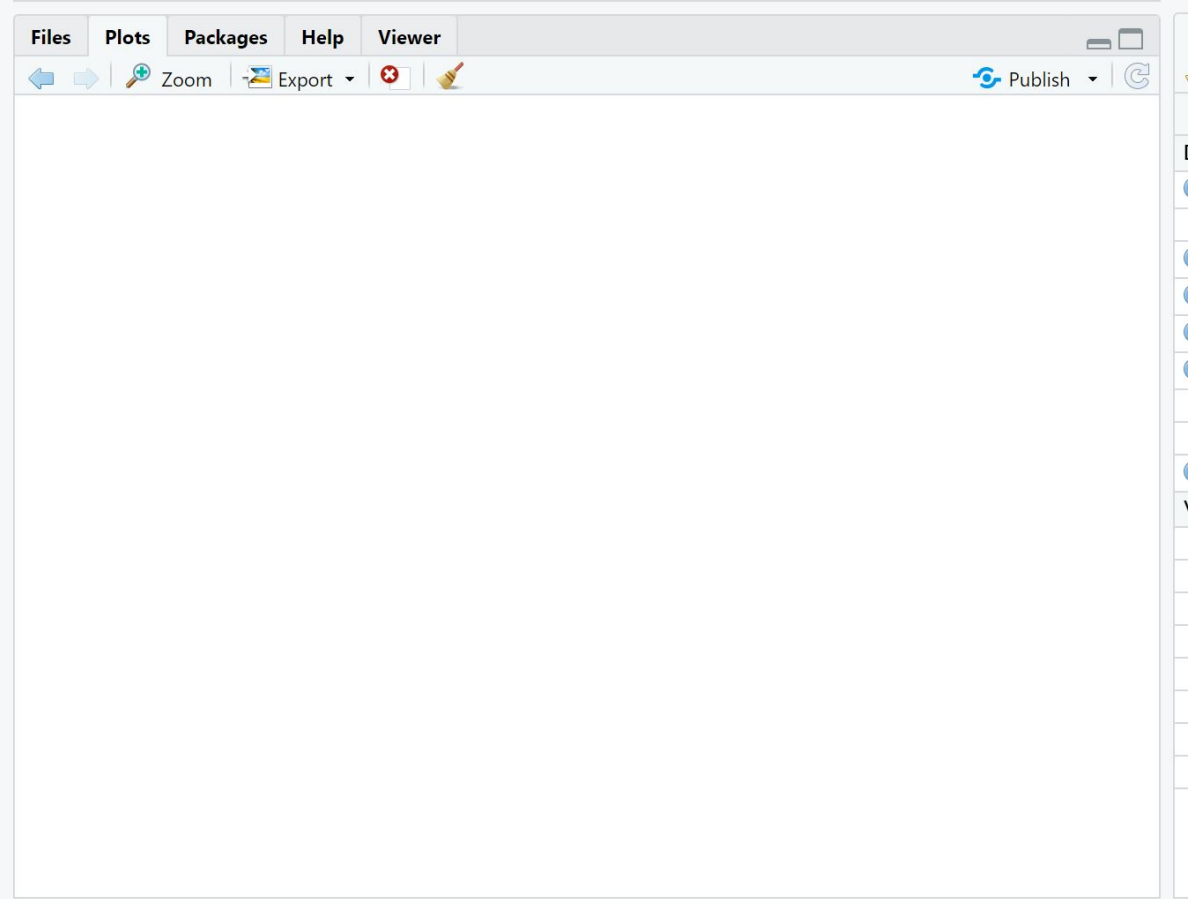
plot.new()

Example 1: Create Completely Empty Plot

We can use the following code to create a completely empty plot in R. Note that the output is simply a blank grey or white area, reflecting that no coordinate system or visual markers have been established. This approach requires subsequent explicit definition of plot boundaries if data is to be added.

plot.new()

Here's what the result looks like in the plotting window in RStudio, confirming the total absence of graphical elements:



Method 2: Defining Boundaries with Invisible Axes (Using plot(NULL, ...))

A more common and generally recommended approach for creating an empty plot that still defines a usable coordinate system is utilizing the generic **plot()** function but passing **NULL** as the first argument. By plotting **NULL**, we instruct R not to draw any data points. We then use graphical parameters to suppress the automatic drawing of axes and labels, while simultaneously defining the range using **xlim** and **ylim**.

This method automatically sets up the plot window and establishes the mapping between data values (e.g., 0 to 10) and physical screen locations, making it immediately ready for layering data using functions like **points()** or **lines()** without needing the extra step of calling **plot.window()** required by Method 1. The key to making the axes invisible relies on specific arguments related to axis drawing.

The critical parameters used here are **xaxt="n"** and **yaxt="n"**. The argument **xaxt** controls the appearance of the x-axis, and setting it to **"n"** (for "none") suppresses the drawing of the tick marks and numerical labels. Similarly, **yaxt="n"** hides the y-axis markings. We also explicitly set **xlab=""** and **ylab=""** to ensure that the axis titles themselves are not displayed, resulting in a

clean, framed space that is geometrically defined but visually minimalist.

The syntax demonstrates how we define the boundaries (0 to 10 for both axes) while ensuring the visual components are suppressed, creating a ready-to-use coordinate space for advanced plotting:

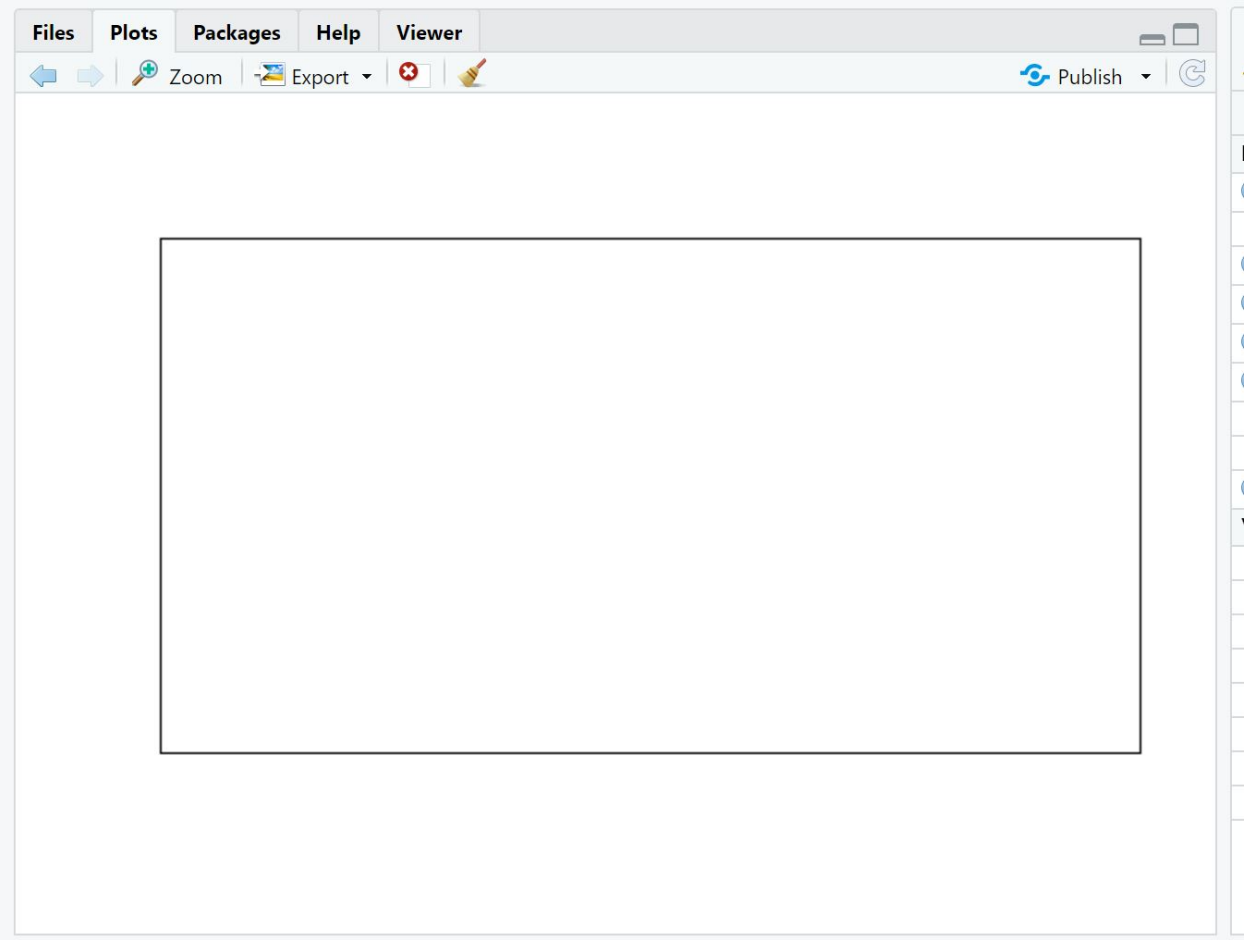
```
plot(NULL, xlab="", ylab="", xaxt="n", yaxt="n",
xlim=c(0, 10), ylim=c(0, 10))
```

Example 2: Create Empty Plot with Axes

We can use the following code to create an empty plot that has a predefined coordinate structure but visually lacks tick marks or labels. This is ideal when the visual space needs to be delimited but axes are intended to be drawn manually later, or are simply unnecessary for the context of the data visualization.

```
plot(NULL, xlab="", ylab="", xaxt="n", yaxt="n",
xlim=c(0, 10), ylim=c(0, 10))
```

Here's what the result looks like in the plotting window in RStudio. Notice the bounding box is present, defining the plot area from 0 to 10 on both axes, but the numerical labels and ticks are absent. This configuration is often preferred when creating custom grid systems or highly specialized graphical overlays.



It is important to understand the role of the axis arguments: the **xaxt** and **yaxt** arguments specifically suppress the automatic drawing of tick marks and numerical labels on the x-axis and y-axis, respectively. This gives the user the flexibility to add custom axes later using the **axis()** function, or to leave the plot frame entirely clean for aesthetic reasons.

Method 3: Preparing for Annotations (Axes and Labels)

The third method is perhaps the most useful for general-purpose data visualization where data will be added dynamically, but the plot still requires standard context. Similar to Method 2, it uses **plot(NULL, ...)** to establish the coordinates. However, in this case, we allow the automatic axes and tick marks to draw, and we explicitly define meaningful labels and a title using the appropriate parameters.

This method is the quickest way to set up a standard graphical environment that is ready for data insertion. The R Base Graphics system automatically calculates where to place the tick marks based on the specified **xlim** and **ylim** ranges (in this example, 0 to 10). By providing strings for **ylab**, **xlab**, and **main**, we immediately contextualize the resulting figure.

This approach establishes a complete framework, including the title (**main**), x-axis label (**xlab**), and y-axis label (**ylab**). It ensures that any data layered onto the plot immediately benefits from the necessary descriptive information, enhancing readability and interpretability. This approach minimizes post-setup commands required to make the plot usable for interpretation.

The code demonstrates the definition of boundaries and the inclusion of descriptive text elements, resulting in a fully configured plotting region:

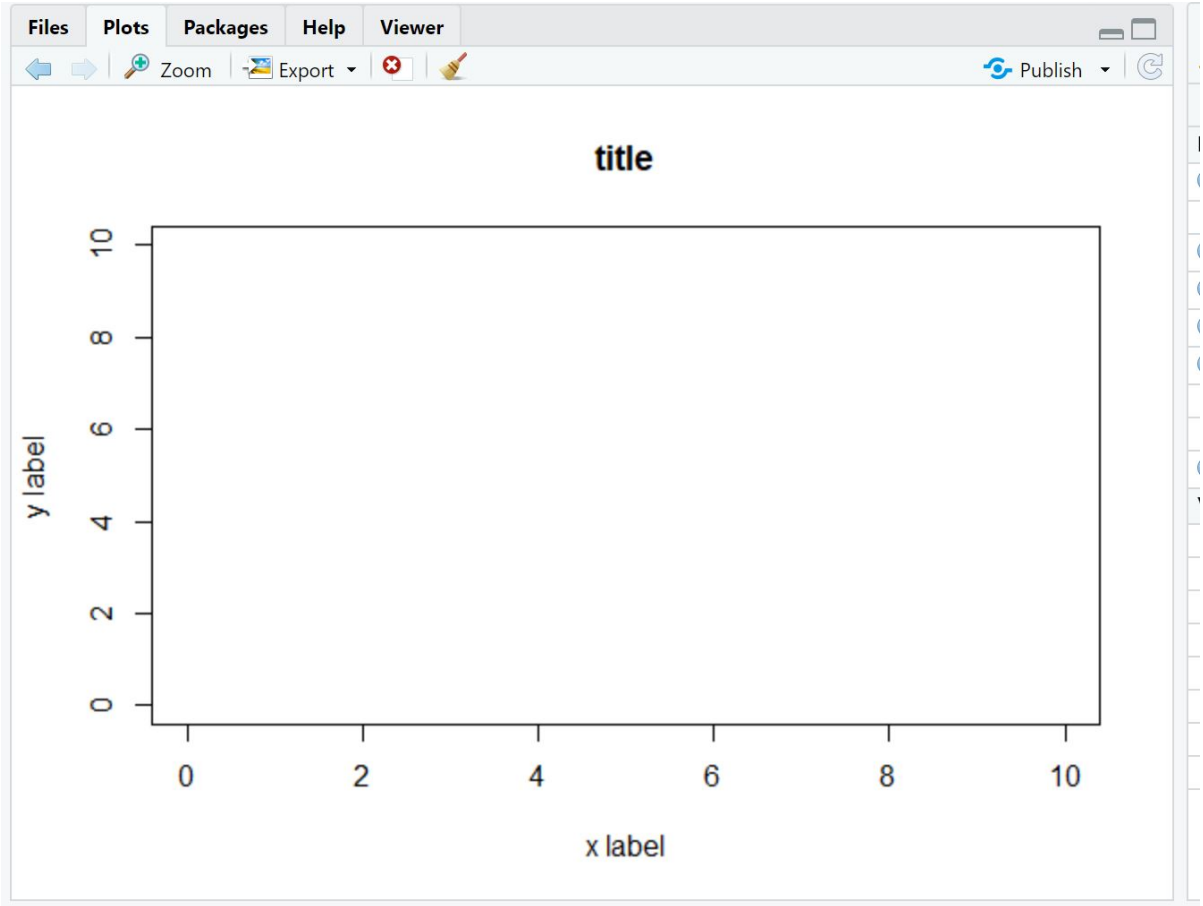
```
plot(NULL, ylab="y label", xlab="x label", main="title",
xlim=c(0, 10), ylim=c(0, 10))
```

Example 3: Create Empty Plot with Axes & Labels

We can use the following code to create an empty plot that includes a full set of axes, numerical tick marks, and descriptive labels. This serves as a fully prepared visualization template, ready for data points or lines to be added using subsequent low-level commands.

```
plot(NULL, ylab="y label", xlab="x label", main="title",
xlim=c(0, 10), ylim=c(0, 10))
```

Here's what the result looks like in the plotting window in RStudio. The plot now includes all standard graphical elements, establishing a clear coordinate context for future additions:



This method is highly recommended for users who plan to generate complex figures using functions like **segments()** or **rect()**, where the visual context provided by standard axes and labels is essential for accurately interpreting the location of the drawn objects within the defined 0-10 range. It represents the best balance between customization and built-in functionality.

Choosing the Right Empty Plot Method

Selecting the appropriate method for creating an empty plot depends entirely on the intended complexity and purpose of the final visualization. Understanding the distinction between establishing a plotting device and establishing a coordinate system is paramount in the R Base Graphics environment.

If the goal is to draw custom elements without any reliance on R's automatic axis generation--perhaps creating a non-Cartesian plot or a highly stylized graphic--then **Method 1 (plot.new())** is the foundation. It demands that the user manually define the window coordinates using **plot.window(xlim, ylim)** immediately after the initial call. This approach provides maximum flexibility but requires precise manual control.

For most standard layered graphics, where users intend to add statistical data, trend lines, or

custom annotations, **Method 2** or **Method 3** utilizing **plot(NULL, ...)** offers superior ease of use. These methods handle the coordinate system setup implicitly, making the plot immediately ready for data addition. Method 2 is ideal for clean, minimalist designs where the data itself is the focus and context is provided elsewhere, while Method 3 is the robust choice for standard, labeled charts and figures.

Mastering these three techniques ensures that R users can precisely control the graphical output, moving beyond simple automated plots to create sophisticated, publication-quality figures tailored exactly to their data visualization needs. The ability to start with a blank slate is the first step toward advanced customization in R.

Additional Resources for Advanced R Plotting

To further enhance your skills in creating and manipulating graphical output in R, consider exploring the documentation related to low-level graphical parameters and specialized plotting functions. These resources will provide deeper insight into customizing the appearance of axes, adding grid lines, and managing multiple plots simultaneously.

The following tutorials explain how to perform other common tasks in R, building upon the foundational knowledge of initiating a plotting device:

How to use the **par()** function to modify global graphic parameters (e.g., margins, colors).
Detailed guide on using the **axis()** function for custom axis placement and labeling.
Tutorial on advanced layering techniques using functions like **abline()** and **segments()**.
Understanding the difference between high-level plotting functions (like **plot()**) and low-level functions (like **points()**).