Learning to Add Text Annotations to R Plots Using the text() Function

Authored by

Mohammed looti

October 30, 2025

In the realm of data visualization with R, effectively annotating your plots is crucial for conveying insights clearly and precisely. While R offers numerous plotting capabilities through its Base R graphics system, adding custom text labels directly onto a chart can significantly enhance its interpretability. This tutorial will guide you through using the versatile **text()** function to achieve this, transforming your static visualizations into more communicative data stories.

The **text()** function is a fundamental tool for adding textual annotations at specific coordinates within an existing plot. It provides a straightforward way to highlight specific data points, add explanations, or simply place descriptive labels where they are most impactful. Understanding its core syntax is the first step towards mastering plot annotation in R.

The basic syntax for the **text()** function is as follows:

text(x, y, "my text")

Here, the primary arguments are:

x, y: These numerical values specify the exact coordinates on the plot where the text string should be placed. The 'x' value determines the horizontal position, and 'y' determines the vertical position.
"my text": This is the actual character string that you wish to display on the plot. It can be any combination of letters, numbers, and symbols enclosed in quotation marks.

Let's explore several practical examples to illustrate how to leverage this function effectively in various scenarios, from adding a single label to customizing multiple text elements and labeling individual data points. Each example will build on the understanding of the **text()** function's capabilities, demonstrating its versatility for enhancing your plots.

Example 1: Adding a Single Text Element to a Plot

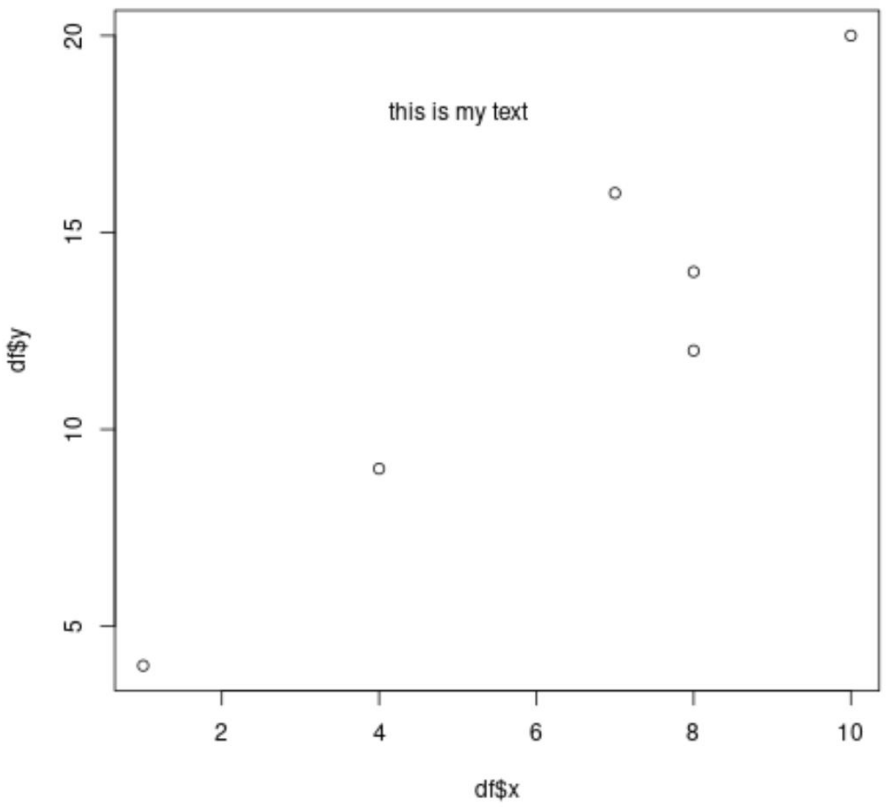
Often, you might need to draw attention to a specific region or add a singular explanatory note on your plot. The **text()** function makes this task simple and intuitive. This first example demonstrates how to place one text element at a precise location within a scatterplot using its x and y coordinates.

The following code block first creates a data frame and then generates a basic scatterplot. Subsequently, it utilizes the **text()** function to insert a custom message at the specified coordinates (5, 18), providing a clear annotation for the visual data.

```
#create data frame with values to plot
df <- data.frame(x=c(1, 4, 7, 8, 8, 10),
y=c(4, 9, 16, 14, 12, 20))
```

```
#create scatterplot
plot(df$x, df$y)

#add text element at (5, 18)
text(x=5, y=18, "this is my text")
```



As depicted in the resulting plot, the designated text element has been successfully added at the precise x, y coordinates of (5, 18). This demonstrates the function's precision in placing annotations exactly where intended, offering a foundational method for direct textual input onto your visualizations.

Example 2: Adding Multiple Text Elements to a Plot

While a single text label can be useful, many plots benefit from multiple annotations to explain different aspects of the data or to provide context for various points. Fortunately, adding several text elements to a plot using the **text()** function is straightforward: you simply call the function multiple times, each with its own set of coordinates and text string.

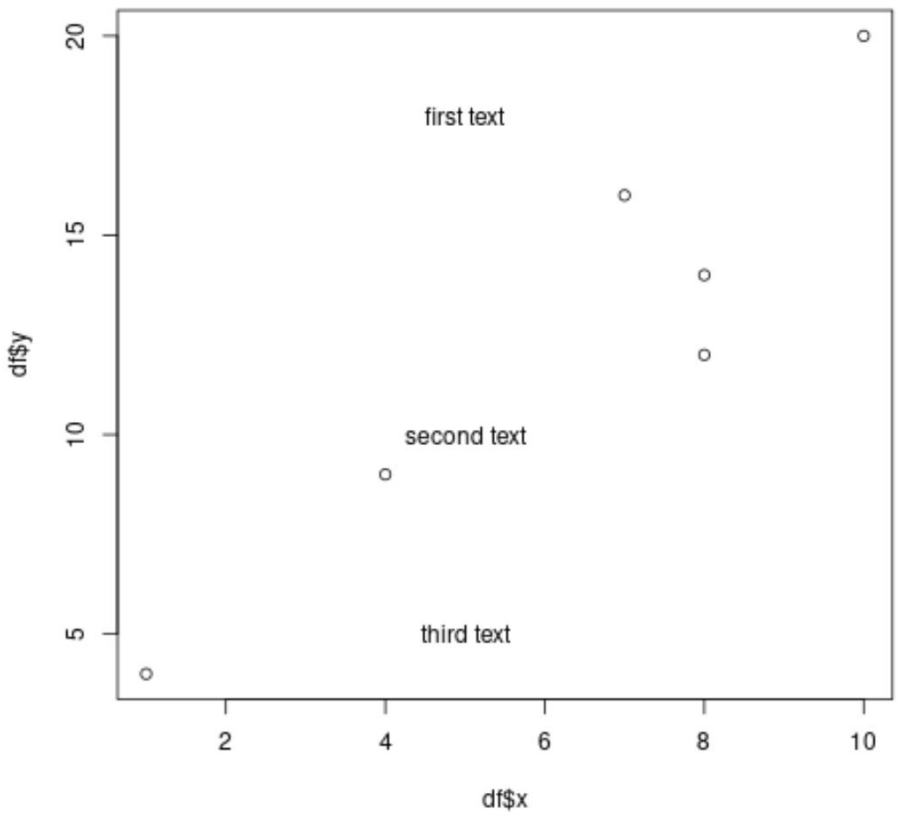
This example builds upon the previous one by illustrating how to overlay several distinct text labels on the same scatterplot. Each call to **text()** will place a new annotation without interfering with

previous ones, allowing for comprehensive labeling.

#create data frame with values to plot
df <- data.frame(x=c(1, 4, 7, 8, 8, 10),
y=c(4, 9, 16, 14, 12, 20))

#create scatterplot
plot(dfx, dfy)

#add text elements
text(x=5, y=18, "first text")
text(x=5, y=10, "second text")
text(x=5, y=5, "third text")



Upon reviewing the output plot, it's evident that three distinct text elements have been successfully integrated. Each label is positioned precisely according to the unique x, y coordinates provided in the respective **text()** function calls. This capability is invaluable for creating richly annotated and informative visualizations.

Example 3: Customizing Text Elements in a Plot

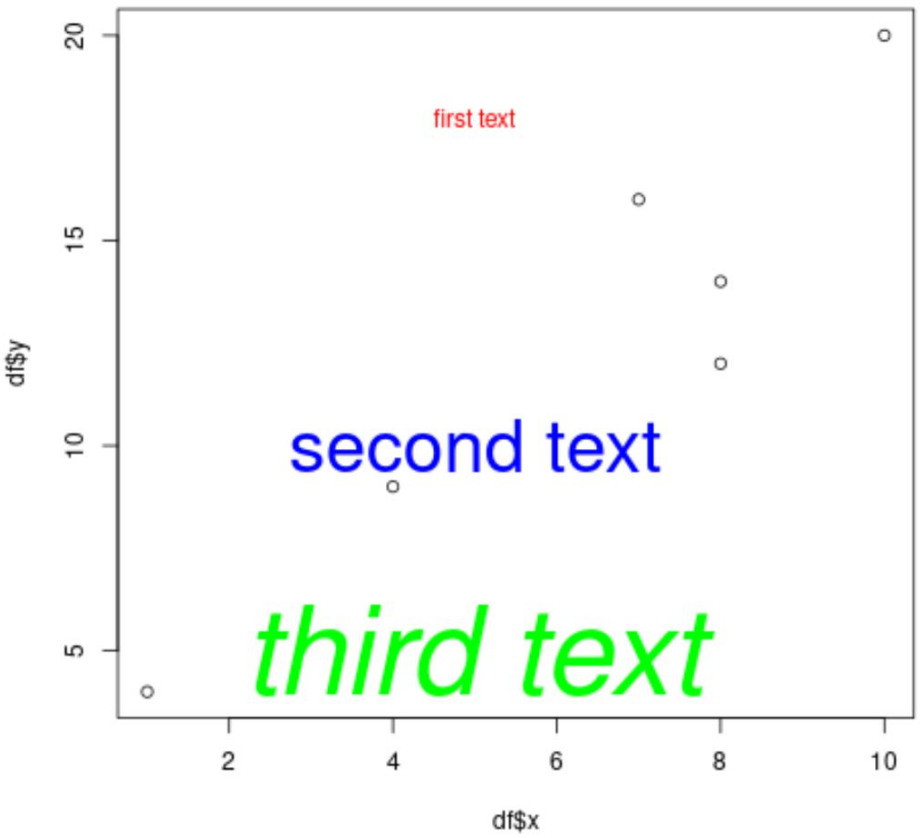
Beyond simply placing text, the **text()** function offers extensive options for customizing the appearance of your annotations. Tailoring the color, size, and font style of your text can significantly improve readability and visual hierarchy, ensuring that important messages stand out. This example demonstrates how to modify these aesthetic properties using additional arguments.

You can adjust the color using the `col` argument, change the size with `cex` (character expansion factor), and modify the font style using the `font` argument. Experimenting with these arguments allows for fine-grained control over your plot annotations, making them align with your overall data visualization aesthetic.

#create data frame with values to plot
df <- data.frame(x=c(1, 4, 7, 8, 8, 10),
y=c(4, 9, 16, 14, 12, 20))

#create scatterplot
plot(dfx, dfy)

#add text elements with custom appearance
text(x=5, y=18, "first text", col='red')
text(x=5, y=10, "second text", col='blue', cex=3)
text(x=5, y=5, "third text", col='green', cex=5, font=3)



As observed in the visual output, each of the three text elements now boasts a distinct, customized appearance. The `col` argument sets the color, `cex` scales the text size (where `cex=1` is the default), and `font` determines the typeface style. These options provide powerful control over the visual presentation of your textual annotations.

Specifically, the `font` argument accepts four distinct numerical values to control the text style:

1: Plain text (the default style).
2: **Bold** text.
3: *Italic* text.
4: ***Bold-italic*** text.

Since `font=3` was specified for the third text element in our example, its appearance is rendered in italic style, clearly demonstrating the effect of this argument.

Example 4: Adding Text Labels to Each Point in a Plot

A common requirement in data visualization is to label individual data points on a scatterplot, especially when dealing with categorical data or identifiers for each observation. The **text()** function facilitates this by allowing you to pass a vector of labels to its `labels` argument, associating each

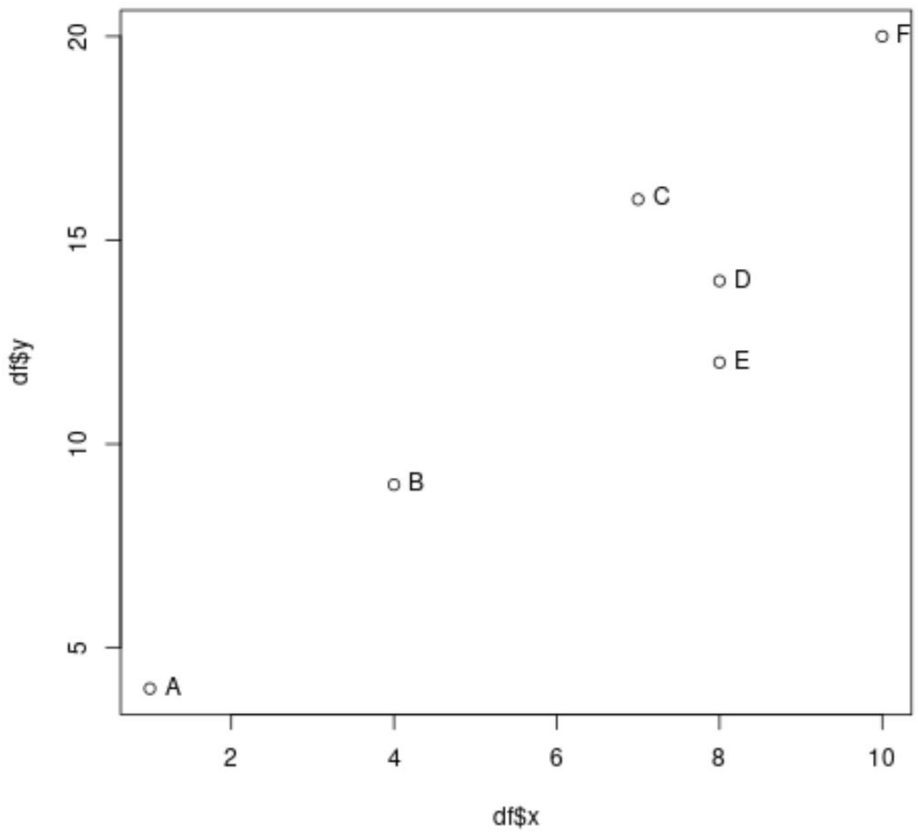
label with a corresponding data point.

This example demonstrates how to create a data frame that includes a column for team names, which will serve as our labels. We then generate a scatterplot and use the **text()** function with the `labels` argument to place these team names next to their respective data points. Additionally, the `pos` argument is used to control the labels' precise positioning relative to each point.

```
#create data frame with values to plot
df <- data.frame(teams=c('A', 'B', 'C', 'D', 'E', 'F'),
x=c(1, 4, 7, 8, 8, 10),
y=c(4, 9, 16, 14, 12, 20))

#create scatterplot
plot(df$x, df$y)

#add text label to each point in plot
text(df$x, df$y, labels=df$teams, pos=4)
```



The resulting plot clearly shows that each data point now features a corresponding text label, sourced from the 'teams' column of our data frame. This significantly enhances the interpretability

of the plot, allowing viewers to quickly identify individual observations.

The `pos` argument plays a critical role in controlling the placement of these text labels relative to their associated data points. It takes one of four integer values:

1: Positions the text label below the data point.
2: Positions the text label to the left of the data point.
3: Positions the text label above the data point.
4: Positions the text label to the right of the data point.

In our example, specifying `pos=4` ensures that each text label is neatly placed to the right of its respective data point, preventing overlap and maintaining clarity. This flexibility in positioning is vital for creating uncluttered and highly readable plots.

Conclusion and Additional Resources

The **text()** function in R is an indispensable tool for enhancing the clarity and communicative power of your data visualizations. By mastering its use, from simple single annotations to customized labels and point-specific text, you can create plots that are not only aesthetically pleasing but also profoundly informative. Effective annotation bridges the gap between raw data and meaningful insights, ensuring your audience grasps the key messages embedded in your graphics.

Continue exploring R's extensive graphical capabilities to further refine your data visualization skills. The following tutorials offer further insights into other common and powerful functions in R:

How to Use paste & paste0 Functions in R