

# How to Easily Change Font Size in Base R Plots

Authored by  
**stats writer**

December 3, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Easily Change Font Size in Base R Plots*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=104277>

Generating high-quality visualizations is a crucial step in any data analysis workflow. When using R language for statistical graphics, the built-in system, commonly referred to as **Base R plots**, offers powerful yet straightforward tools for creating professional figures. However, achieving aesthetic perfection often requires fine-tuning elements like **font size** to ensure readability and visual impact, especially when preparing figures for publication or presentations. Small, illegible text can undermine even the most compelling data narrative.

In Base R, controlling typography is primarily managed through scaling arguments passed directly to the plot function or through global graphical parameters. This guide provides an exhaustive look at how to precisely control the size of every textual element--titles, subtitles, axes labels, and tick marks--within your Base R visualizations, ensuring your figures are both accurate and accessible.

The fundamental approach to dynamically scaling text size in Base R involves using the `cex` (Character Expansion) parameter family. By understanding and applying these specific `cex` arguments, you gain granular control over the visual hierarchy of your plot elements.

## Understanding the Character Expansion (cex) Parameters

The Character Expansion, or `cex`, family of arguments is the workhorse for font size modification in Base R graphics. These parameters take a numerical value, where a value of 1 represents the default size. Values greater than 1 increase the size (e.g., 2 doubles the size), and values less than 1 decrease the size (e.g., 0.5 halves the size). It is essential to remember that `cex` arguments are applied locally within the specific plotting function call.

Different textual components of a Base R plot are controlled by distinct `cex` arguments. Mastering these distinctions allows for sophisticated typography control, enabling you to highlight important information, such as the main title, while maintaining appropriate sizing for secondary elements, like axis tick labels. This level of detail ensures that the visual structure supports the analytical message.

The primary arguments used for controlling text scale in Base R plots are:

`cex.main`: Specifically controls the font size of the **main title** of the plot.

`cex.sub`: Controls the font size of the **subtitle** (if one is provided).

`cex.lab`: Controls the font size of the **X and Y axis labels** (the text describing what the axes represent).

`cex.axis`: Controls the font size of the **axis tick mark values** (the numerical or categorical text plotted along the axes).

`cex`: This general argument controls the size of plotted symbols and the default text size for annotations or legends if not overridden by the specific `cex.*` arguments above.

## Applying Font Size Control: The General Syntax

To demonstrate the implementation of these parameters, we will first look at the general syntax used within the primary `plot` function call. This structure shows how to simultaneously modify the font size of various components. We recommend using a high value, like 2, initially to clearly visualize the effect of the scaling parameter before fine-tuning it to a more suitable publication standard (often between 1.2 and 1.5 for titles).

You can use the following syntax to change the font size of various elements in base R plots:

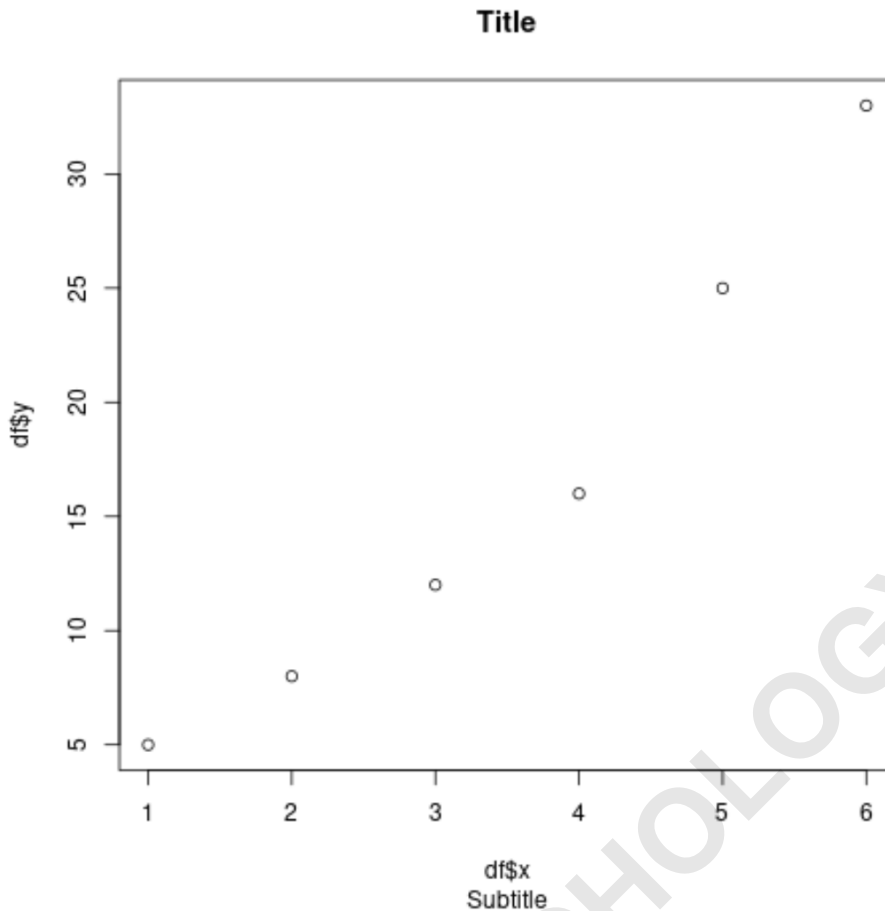
```
plot(df$x, df$y, main='Title', sub='Subtitle',  
cex.main=2, #change font size of title  
cex.sub=2, #change font size of subtitle  
cex.lab=2, #change font size of axis labels  
cex.axis=2) #change font size of axis text
```

Before diving into specific examples, we must establish a reproducible data source. We will utilize a simple data frame for creating a standard `scatterplot`. This baseline plot will serve as a reference point for all subsequent font scaling modifications, allowing easy comparison between the default settings and the scaled results.

The following examples show how to use this syntax with the following scatterplot in base R:

```
#create data frame  
df <- data.frame(x=c(1, 2, 3, 4, 5, 6),  
y=c(5, 8, 12, 16, 25, 33))  
  
#create scatterplot with all default font sizes  
plot(df$x, df$y, main='Title', sub='Subtitle')
```

The resulting plot, generated using the default settings, illustrates the standard text size Base R assigns to the title, subtitle, and axis components. This initial visualization is crucial for understanding the need for customization, especially if the plot is destined for a large poster or a high-resolution display where default text might appear too small.



### Example 1: Changing the Font Size of the Main Title (`cex.main`)

The main title is often the most critical text element, summarizing the figure's content. Therefore, it typically requires the largest font size to draw immediate attention. We use `cex.main` to specifically scale this component, ensuring it stands out from the rest of the plot elements.

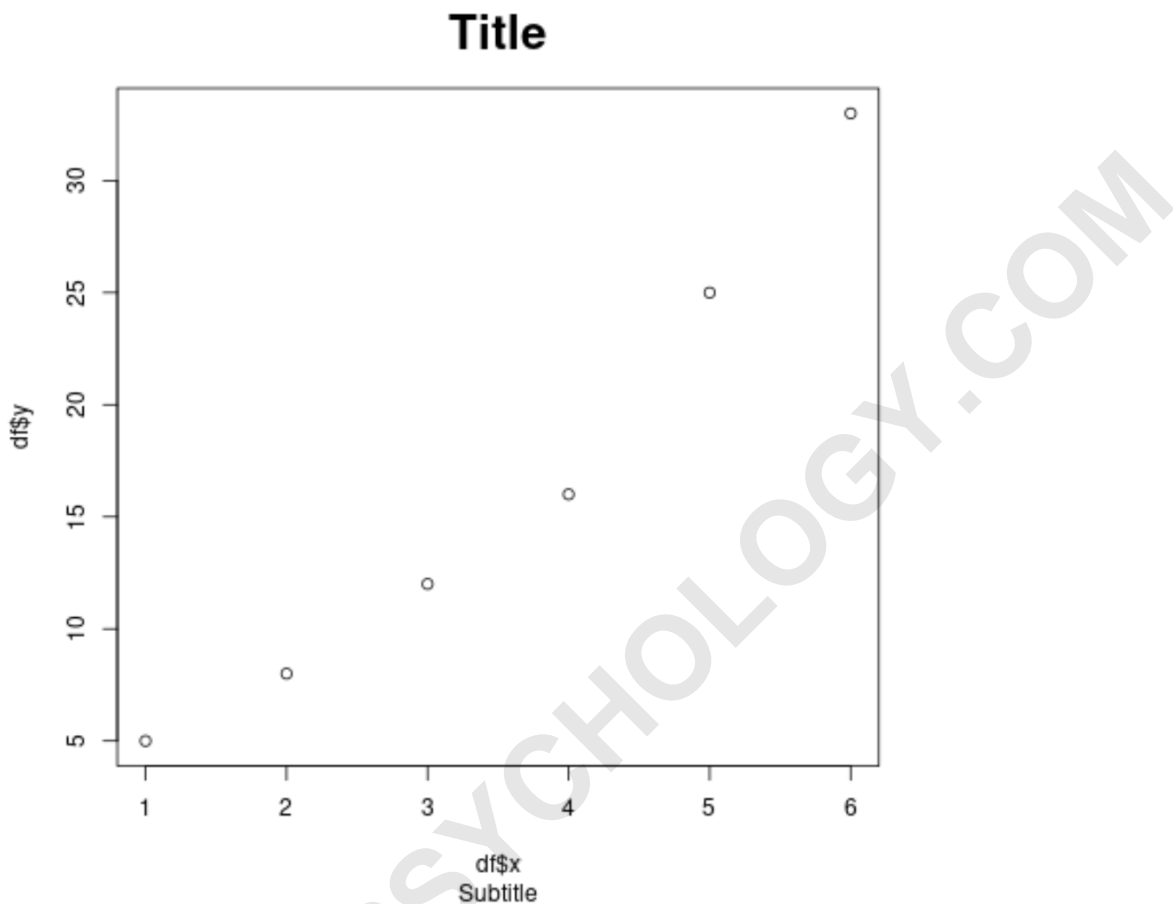
In this example, setting `cex.main=2` doubles the default size of the main title. Notice how all other textual elements--the subtitle, axis labels, and axis text--remain at their default size (which corresponds to `cex=1`). This isolation demonstrates the targeted nature of the `cex.*` arguments, providing precise control without affecting unrelated elements.

The following code shows how to change the font size of the title of the plot:

```
#create scatterplot with increased font size of title  
plot(df$x, df$y, main='Title', sub='Subtitle'  
cex.main=2)
```

The visual result clearly shows the amplified title, which is now significantly more prominent than

the other elements. When selecting a value for `cex.main`, consider the context: while a value of 2 is illustrative, for publication, values between 1.2 and 1.5 often strike a better balance between prominence and overall plot aesthetics.



### Example 2: Changing the Font Size of the Subtitle (`cex.sub`)

The subtitle usually provides auxiliary information, such as the data source, specific caveats, or the date of generation. Since it is secondary to the main title, it often requires a smaller or equivalent font size, typically equal to or slightly larger than the axis labels. The `cex.sub` parameter controls this element independently.

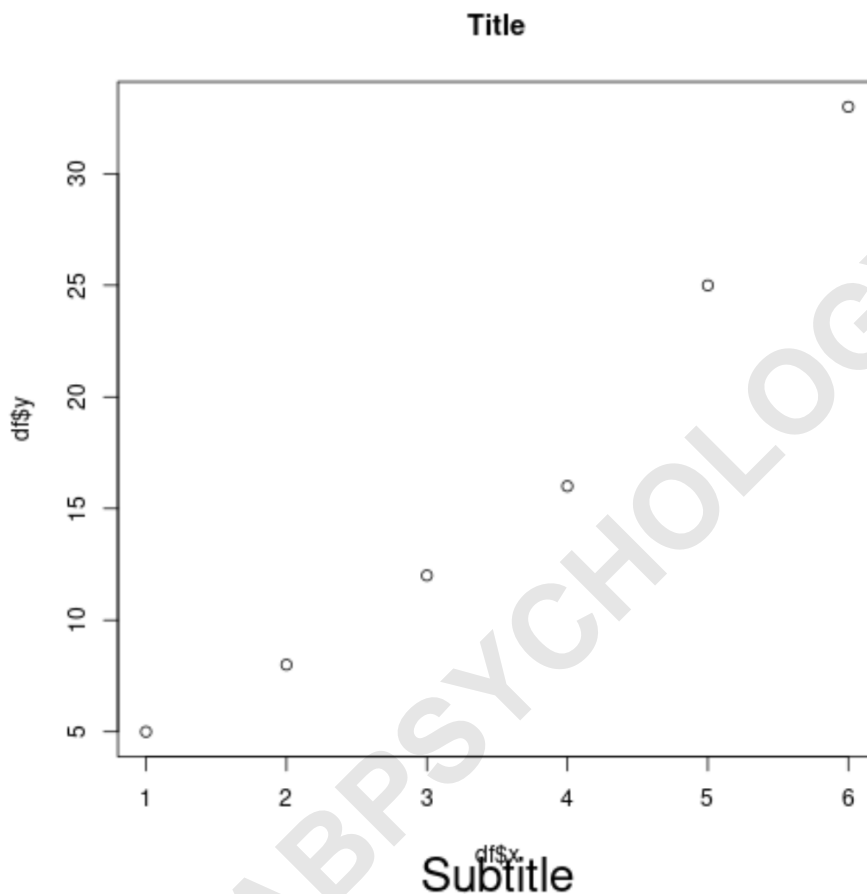
Here, we apply `cex.sub=2` to illustrate its impact. Although subtitles are less frequently used in highly minimalist visualizations, they are invaluable for internal reporting or documentation where supplementary text is required below the main heading. The default behavior places the subtitle centered beneath the main title area.

The following code shows how to change the font size of the subtitle of the plot:

```
#create scatterplot with increased font size of subtitle
```

```
plot(df$x, df$y, main='Title', sub='Subtitle'.  
cex.sub=2)
```

As depicted in the resulting image, only the subtitle text has been scaled up, maintaining the default size for all other components. Strategic use of `cex.sub` helps establish a clear hierarchical relationship between the main topic and the supporting context of the visualization.



### Example 3: Changing the Font Size of Axis Labels (`cex.lab`)

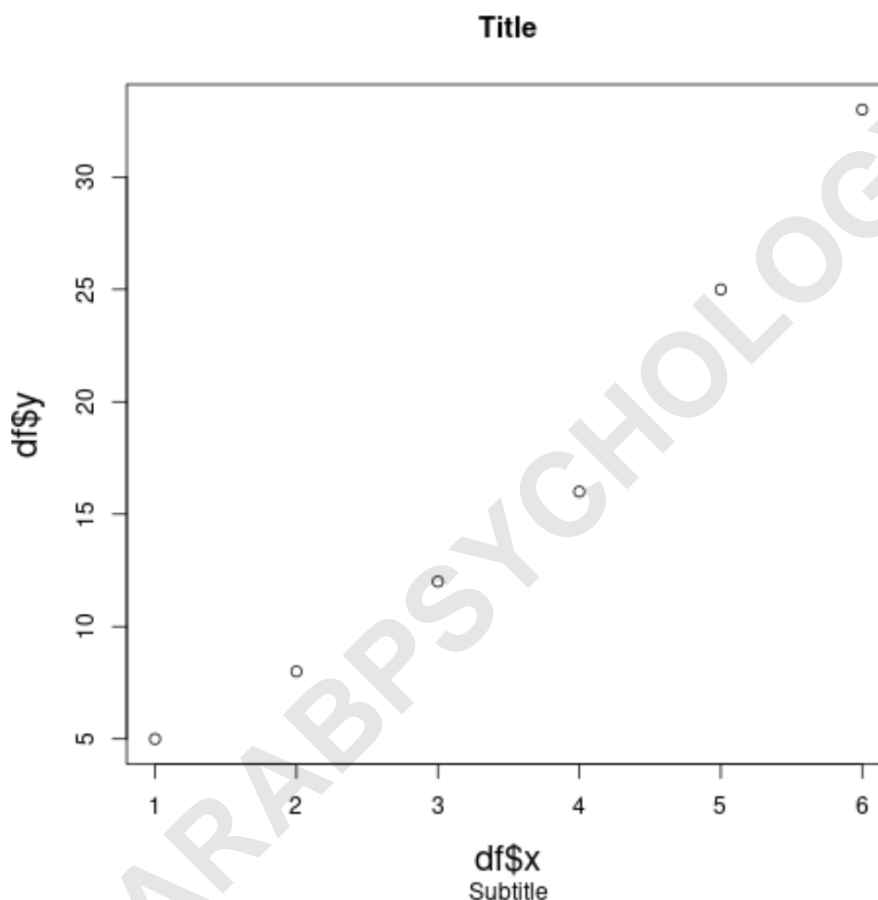
Axis labels--the text describing the X and Y variables (e.g., 'Time' or 'Measured Value')--are critical for interpretation. If these labels are too small, the audience cannot easily understand what the axes represent. The `cex.lab` parameter controls the size of these descriptions.

For most professional graphics, axis labels should be clearly readable, often requiring a slight increase in size compared to the default, typically achieved by setting `cex.lab` between 1.1 and 1.3. Increasing this size is crucial when dealing with plots that contain long variable names or when readability must be maintained across various display sizes.

The following code shows how to change the font size of the axis labels of the plot:

```
#create scatterplot with increased font size of axis labels  
plot(df$x, df$y, main='Title', sub='Subtitle',  
cex.lab=2)
```

In this demonstration, setting `cex.lab=2` results in significantly larger axis titles. It is worth noting the relationship between `cex.lab` and `cex.axis`: while `cex.lab` handles the descriptive titles, `cex.axis` (covered next) handles the numerical markers along the axes.



#### Example 4: Changing the Font Size of Axis Text/Tick Marks (`cex.axis`)

The axis text, or the tick mark labels, represents the numerical values corresponding to the plotted data. These must be clearly visible for viewers to quantify the positions of the data points. The `cex.axis` parameter manages the font size of these numerical markers.

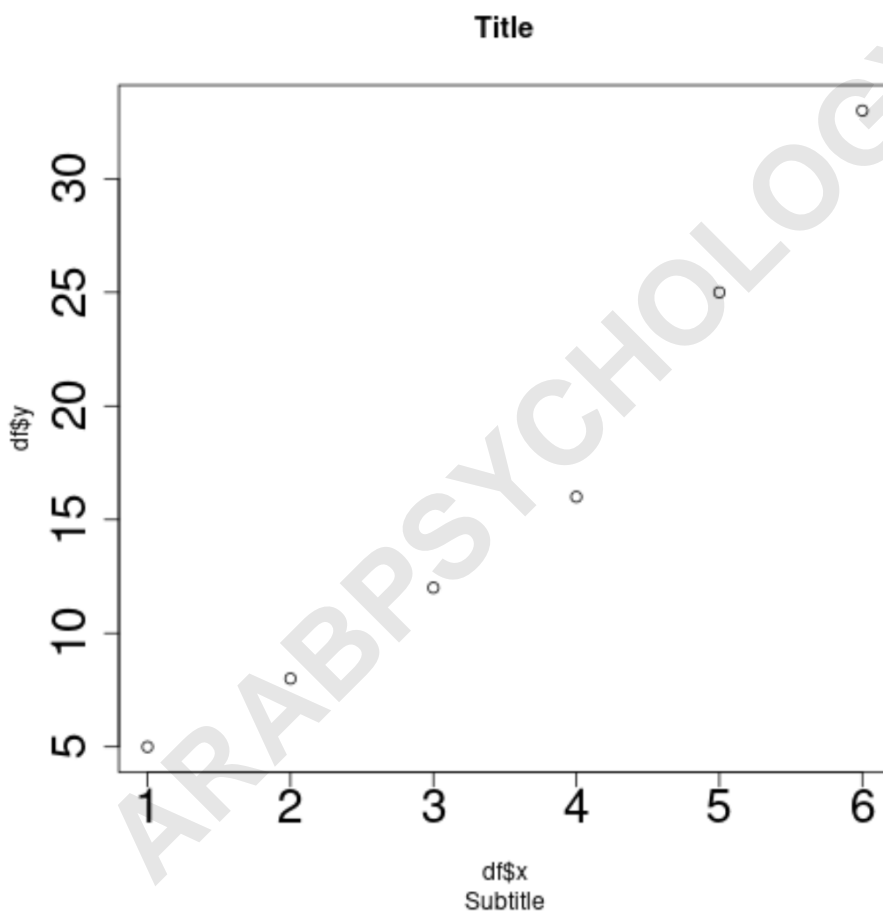
If you are generating complex plots with many tick marks or overlapping numerical labels, increasing `cex.axis` might exacerbate clutter. For standard plots, ensuring `cex.axis` is large

enough to be easily read from a distance is paramount.

The following code shows how to change the font size of the axis text of the plot:

```
#create scatterplot with increased font size of axis text  
plot(df$x, df$y, main='Title', sub='Subtitle',  
cex.axis=2)
```

Observing the output confirms that only the numerical values on the X and Y axes have been scaled by a factor of 2. This granular control allows the user to differentiate the importance of the numerical scale versus the descriptive variable labels.



## Implementing Global Font Changes with the `par()` Function

While passing `cex.*` arguments directly to the `plot` function is suitable for one-off modifications, it becomes tedious when generating multiple plots that require consistent styling. For persistent or global changes that affect all subsequent plots within the current R session (or within a script block), the `par()` function is the definitive solution.

The `par()` function is used to set or query graphical parameters. By setting `cex.main`, `cex.lab`, etc., within `par()`, you establish default settings that the Base R plotting functions will inherit until they are explicitly reset or overridden by local function calls. This ensures uniformity across figures, which is crucial for publications or cohesive reports.

It is best practice to save the existing graphical parameters before making global changes and restore them afterward. This prevents unintended side effects on later plots or scripts. For example, to globally increase the axis text size for several plots, you would execute the following sequence:

```
# Save current graphical parameters
```

```
old_par <- par(no.readonly = TRUE)
```

```
# Globally increase axis font size
```

```
par(cex.axis = 1.5)
```

```
# Plot 1 (inherits the new cex.axis)
```

```
plot(df$x, df$y)
```

```
# Plot 2 (also inherits the new cex.axis)
```

```
hist(df$y)
```

```
# Restore original graphical parameters
```

```
par(old_par)
```

If the general `cex` argument is modified within `par()`, it acts as a universal scaling factor for all text elements, including plot symbols, unless specific `cex.*` arguments are provided to override it. This can be a quick way to scale an entire visualization, but specific control is generally preferred for creating professional-grade graphics.

## Customizing Font Families, Styles, and Weights

While the `cex` arguments control the size of the text, Base R also provides mechanisms for controlling the actual font face, style (like bold or italic), and family. These controls are managed primarily through the `font` and `family` arguments, which can be applied globally via `par()` or locally within the plotting function.

The `font` argument takes an integer value to specify the text style:

```
font=1: Plain text (default).
```

`font=2`: Bold text.

`font=3`: Italic text.

`font=4`: Bold and italic text.

Similar to `cex`, there are specific counterparts for each plot component: `font.main`, `font.sub`, `font.lab`, and `font.axis`. For example, to make the main title bold, you would use `font.main=2` within your plotting function call. Combining these style controls with precise size adjustments offers maximum typographical flexibility.

The `family` argument allows you to specify the actual font typeface (e.g., "serif," "sans," or a specific system font like "Helvetica"). While Base R has limited built-in font families, users often rely on packages like `extrafont` or `showtext` to incorporate system-installed or external fonts, providing a much broader range of typographical choices essential for meeting specific journal or corporate branding requirements.

## Summary of Best Practices for Base R Font Management

Effective control over font size in Base R plots is essential for producing visualizations that are both informative and visually appealing. By leveraging the `cex` parameter family, analysts can precisely define the visual hierarchy of their plot elements, ensuring that titles are prominent, labels are readable, and tick marks provide clarity without clutter.

Key takeaways for managing typography in Base R graphics:

Use specific `cex.*` arguments (`cex.main`, `cex.lab`, `cex.axis`) for targeted control over individual text components.

Remember that `cex` values are multiplicative factors relative to the default size (1 is default).

Employ the `par()` function for setting global or persistent font size changes across multiple plots, but always remember to restore the original parameters afterward.

Combine `cex` parameters with `font.*` parameters to control both the size and the style (bold, italic) of the text elements, optimizing readability and visual structure.

For advanced font selection beyond the Base R defaults, consider integrating specialized packages to access system fonts.

By implementing these techniques, you move beyond default settings and gain the ability to customize your R language visualizations to meet any demanding graphical standard.

The following tutorials explain how to perform other common plotting operations in R: