

How to Easily Adjust Plot Element Size with 'cex'

Authored by
stats writer

November 20, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Adjust Plot Element Size with 'cex'*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=98467>

When generating visualizations using the R statistical environment, the ability to control the aesthetics of the resulting graph is paramount for effective communication. Among the most fundamental aspects requiring customization is the size of various graphical parameters, including data symbols, axis labels, and titles. This essential control is managed primarily through the cex argument, a powerful scaling tool available across a multitude of plotting commands in R's base graphics system.

The **cex** argument operates as a scaling factor, accepting a numerical value that determines the proportional increase or decrease relative to the default size established by R. By default, most plot elements are rendered using a **cex** value of **1.0**. Understanding this scaling mechanism is critical: supplying a value greater than 1.0, such as **cex=2**, will effectively double the size of the targeted elements, enhancing their visibility, which is often useful for presentation slides or large displays. Conversely, inputting a fractional value less than 1.0, such as **cex=0.5**, will reduce the size of the plot elements by half, allowing for tighter packing of data points or less visual distraction in complex multi-panel figures. Mastery of this simple proportional scaling allows developers and analysts precise control over the visual impact of their generated plots.

Understanding Graphical Parameters in R

The **cex** argument is central to managing size within R's graphical system. It stands for "character expansion factor," and it is one of the most frequently used graphical parameters when creating custom visualizations. While the plot() function itself is where we initiate the graph, **cex** and its related arguments are passed to this function to override the default size settings.

By default, all plot elements--symbols, axis annotations, labels, and the main title--inherit a scaling factor of **1**. This default size provides a standard, readable output suitable for basic analysis. However, when preparing figures for publication or presentation, analysts often need to highlight specific elements or ensure readability in different output formats. This necessity is precisely why the scaling mechanism provided by the **cex** family of arguments becomes indispensable for professional data visualization.

It is crucial to differentiate between the general **cex** argument and its specialized counterparts. The generalized **cex** argument, when used alone in the **plot()** function, applies its scaling factor primarily to the data symbols (points) drawn on the graph. To achieve more refined control over the text elements, such as those associated with the axes and titles, R offers a specific set of five **cex** arguments that allow for granular control over the scaling of individual components within the plot area.

The Five Specialized cex Parameters

While the base **cex** handles point size, R provides four additional, specialized arguments dedicated solely to scaling text components. Utilizing these distinct parameters ensures that the visual hierarchy of the plot remains balanced, preventing, for instance, a disproportionately large title relative to the axis labels. These five parameters collectively grant the user total control over scaling elements proportional to their default size:

cex: This is the primary scaling factor and changes the size of plot symbols, such as points used in scatterplots.

cex.axis: Specifically targets the numerical or categorical annotations associated with the axis tick marks.

cex.lab: Controls the size of the explanatory text used for the x-axis and y-axis labels (e.g., "X Variable" or "Response Measurement").

cex.main: Dedicated to changing the size of the primary plot title, often located at the top center of the visualization.

cex.sub: Used to adjust the size of the optional plot subtitle, usually displayed below the main plot area.

The following practical demonstration illustrates how to effectively utilize these five arguments to create a highly customized and visually balanced figure. We will start by defining a simple data frame in R and then proceed through the steps of baseline plotting and advanced scaling adjustments.

Setting Up the Environment: Creating a Data Frame

To demonstrate the utility of the **cex** arguments, we first require a dataset within the R environment. In statistical computing, a data frame is the most common structure used to store tabular data, analogous to a spreadsheet, where rows represent observations and columns represent variables. We will create a small sample data frame named `df` containing ten pairs of (x, y) coordinates, suitable for plotting a simple scatterplot.

The construction of this **data frame** is essential for providing the raw material for our visualization exercise. Pay close attention to the syntax required to instantiate the variables `x` and `y` and combine them into the coherent structure of `df` using the `data.frame()` function. This ensures reproducibility of the examples that follow.

Below is the R code snippet used to generate and inspect our sample dataset:

```
#create data frame  
df <- data.frame(x=c(1, 2, 2, 4, 5, 3, 5, 8, 12, 10),
```

```
y=c(5, 9, 12, 14, 14, 13, 10, 6, 15, 18))
```

```
#view data frame
```

```
df
```

```
x y
```

```
1 1 5
```

```
2 2 9
```

```
3 2 12
```

```
4 4 14
```

```
5 5 14
```

```
6 3 13
```

```
7 5 10
```

```
8 8 6
```

```
9 12 15
```

```
10 10 18
```

With the data prepared, we are now ready to proceed to the initial visualization step, where we will examine the default appearance of the plot elements before any scaling adjustments are applied.

Baseline Visualization: The Default plot() Function

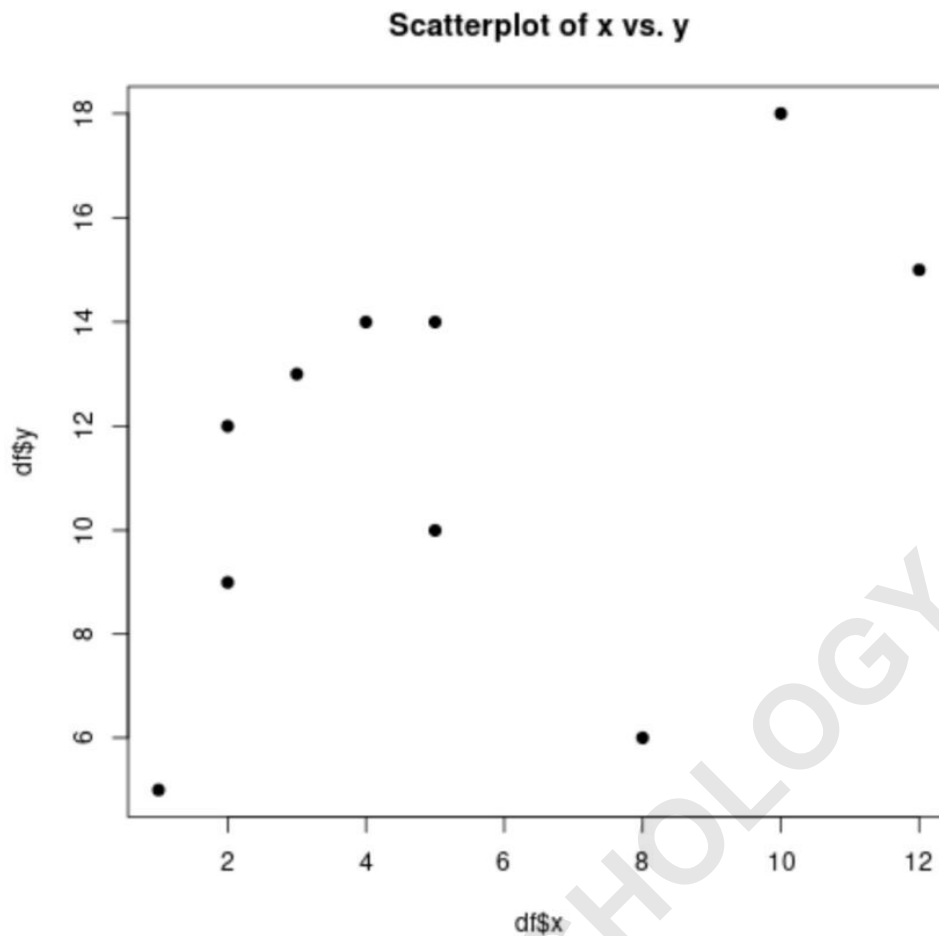
The first step in any visualization customization process is establishing a baseline plot using the default settings. We will use the generic `plot()` function to generate a simple scatterplot of the `x` variable against the `y` variable from our newly created `df` data frame. We include the `pch=19` argument to specify that filled-in circles should be used as the symbols for the data points, which are highly visible and easy to scale.

This initial plot demonstrates the standard appearance where all elements--the points, the axis annotations, and the title--are rendered with the inherited `cex` value of `1`. This visual provides the critical reference point against which we will measure the impact of our subsequent scaling manipulations. It allows us to clearly observe how increasing or decreasing the `cex` parameters affects the aesthetics of the final figure.

The code for generating the baseline scatterplot is shown below:

```
#create scatterplot of x vs. y
```

```
plot(df$x, df$y, pch=19, main='Scatterplot of x vs. y')
```



It is worth making a specific note regarding the **pch** argument: `pch=19` is a graphical parameter that dictates the plotting character used for symbols. In this case, 19 corresponds to a solid circle. Although **pch** controls the shape, **cex** controls the size of that shape.

Adjusting Symbols: Using the Standard **cex** Parameter

The primary use case for the generic **cex** argument is adjusting the size of the data points or symbols. When our scatterplot points are too small to be easily discernible, or conversely, too large and begin to overlap excessively, adjusting the general **cex** parameter is the necessary intervention. Recall that a value of **2** will double the size, while a value of **0.5** will halve it. This simple scaling is applied universally to all plotting symbols in the graph.

For example, if we wanted the data points to be dramatically larger to emphasize the data clusters in a presentation, we would incorporate `cex=2` directly into the `plot()` function call. This adjustment immediately enhances the visual weight of the data points, making the central distribution of the data more prominent to the viewer. However, notice that applying only the general **cex** argument does not affect the size of the title or the axis labels; these textual elements remain fixed at their default size of **cex=1**, thus requiring the specialized parameters for further

refinement.

The ability to selectively adjust the symbol size independently of the text is a fundamental requirement for creating clear data graphics. A large title and tiny points, or vice versa, often leads to confusion regarding the plot's focus. Therefore, while the general **cex** argument is powerful for data emphasis, it is only the first step in a complete plot customization process. The next section addresses how to apply similar scaling control to the crucial textual elements of the visualization.

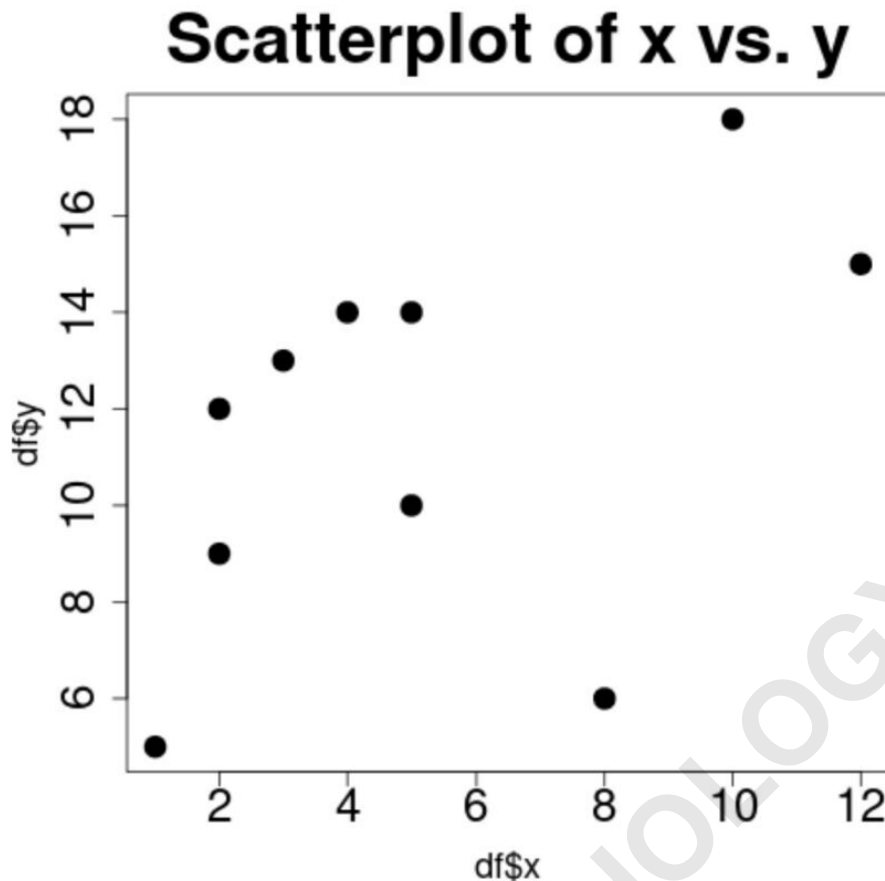
Fine-Tuning Text Elements: Applying Specialized cex Parameters

Effective data visualization requires that all textual components--titles, labels, and tick mark annotations--are easily readable and appropriately sized relative to one another and to the overall plot dimensions. This is where the specialized **cex** arguments provide precise control. Instead of relying on the general scaling factor, we use **cex.main**, **cex.lab**, and **cex.axis** to manage the typography independently.

Consider the necessity of a bold title. Setting **cex.main=3** triples the size of the main title, ensuring it commands attention and clearly states the purpose of the figure. Simultaneously, we might want the axis labels (e.g., "x" and "y") to be moderately larger than the default for better readability, achieved by setting **cex.lab=1.5**. Finally, the numerical annotations along the axes, controlled by **cex.axis**, might need to be doubled (**cex.axis=2**) to ensure they are legible when the plot is viewed at a distance or when the output figure is smaller than intended.

By employing these specific parameters, we create a visually cohesive plot where the scaling factors are chosen purposefully for each element based on its hierarchical importance. This strategy avoids the visual clutter that results from using a single, uniform scaling factor across all elements. The following code demonstrates the simultaneous application of these parameters:

```
#create scatterplot with custom symbol and text sizes  
plot(df$x, df$y, pch=19, main='Scatterplot of x vs. y',  
cex=2, cex.main=3, cex.lab=1.5, cex.axis=2)
```



This execution results in a plot where the visual impact is carefully managed according to the role of each component. Here is a breakdown of the specific scaling effects achieved:

cex=2: Increased the size of the data symbols (circles) by 200%, making them highly prominent.

cex.main=3: Increased the size of the plot title text by 300%, ensuring maximum visibility.

cex.lab=1.5: Increased the size of the x and y-axis labels by 150%, improving their readability without overpowering the title.

cex.axis=2: Increased the size of the tick mark annotations (the numbers along the axes) by 200%, ensuring clarity.

Conclusion and Best Practices for Custom Scaling

The **cex** family of graphical parameters provides fundamental control over the aesthetic appearance of R visualizations, allowing for precise scaling of both symbols and text. Understanding that the default scaling factor is 1, and that all **cex** inputs operate proportionally to this baseline, is the key to mastering plot customization. Leveraging the specialized parameters (**cex.axis**, **cex.lab**, **cex.main**, and **cex.sub**) ensures that every element of the plot is sized appropriately for its function and the intended display medium.

When applying these scaling factors, a critical best practice is to always evaluate the final output critically. Over-scaling elements can lead to overlap and visual distortion, defeating the purpose of enhanced clarity. It is recommended to use small, iterative changes in **cex** values until the optimal visual balance is achieved. Furthermore, remember that these parameters are not restricted to the basic `plot()` function; they can be utilized in many other base R plotting functions and when customizing elements within the `par()` function for global settings. Feel free to experiment liberally with the values for each of these **cex** arguments to create a plot that perfectly meets your specific analytical and presentation needs.

ARABPSYCHOLOGY.COM