Learning to Visualize Data in R: A Guide to Drawing Circles in Plots

Authored by

Mohammed looti

October 30, 2025

Data visualization is a cornerstone of effective data analysis, allowing researchers and analysts to convey complex information clearly and concisely. Within the powerful **R programming language**, creating compelling visualizations often involves adding various graphical elements to highlight specific insights or contextual areas. This comprehensive guide will explore two primary and highly effective methods for drawing geometric circles directly onto plots in R: leveraging the foundational **Base R plotting system** in conjunction with the specialized **plotrix** package, and utilizing the flexible **ggplot2** package extended by **ggforce**. By the conclusion of this tutorial, you will possess the proficiency required to accurately and aesthetically add circles to your **scatter plots**, significantly enhancing their explanatory power and visual appeal.

Method 1: Drawing Circles Using Base R and the plotrix Package

The **Base R** graphics system provides a fundamental, robust set of built-in functions essential for generating a wide variety of statistical plots. This core plotting capability is the standard starting point for many R users. However, while **Base R** excels at standard charts, specialized geometric shapes, such as precisely drawn perfect circles, are not inherently included in its core functions. To seamlessly overcome this limitation and expand the plotting toolkit, the R community has developed powerful extension packages that integrate flawlessly with the Base R environment.

One such invaluable resource is the **plotrix package**. This extension is specifically designed to provide a rich variety of specialized plotting functions that simplify the process of adding complex graphical elements to existing plots. For the purpose of accurately drawing circles, **plotrix** provides the highly versatile function, `draw.circle()`. This function allows for precise control over the circle's central position, its radial size, and its overall aesthetic appearance, making it the definitive choice for Base R users.

To successfully utilize the `draw.circle()` function, it is necessary to first install and subsequently load the **plotrix** package into your active R session. This preparatory step is standard practice whenever incorporating any third-party package into an R workflow. The following code snippet illustrates the necessary commands required to ensure your environment is correctly prepared for circle plotting:

```
library(plotrix)

#create scatter plot
plot(x, y)

#add circle at specific (x, y) coordinates with specific radius
draw.circle(x=3, y=8, radius=.5)
```

Once the package is active, you are ready to overlay circles onto your existing plots. The

`draw.circle()` function mandates the specification of at least three core arguments: `x` and `y`, which define the exact center coordinates of the circle; and `radius`, which determines its size. Furthermore, the function supports a robust set of optional arguments for extensive customization, including `border` for setting the outline color, `col` for defining the interior fill color, `lwd` for controlling the line width, and `lty` for specifying the line type (e.g., solid, dashed, dotted).

Example 1: Drawing a Single Circle with Base R

Let us proceed through a practical, step-by-step example demonstrating how to draw a single, isolated circle onto a **scatter plot** using the fundamental **Base R** graphics system, enhanced by the **plotrix** package. Our initial step involves creating a simple **data frame** to hold our synthetic data, followed by generating a basic scatter plot from this data. After the foundational plot has been successfully established, we will accurately overlay a circle at a pair of specified coordinates.

To commence the analysis, ensure that the **plotrix package** is both installed on your system and loaded into your current R session. If this preparation has not yet been completed, execute the following commands using the `install.packages()` and `library()` functions:

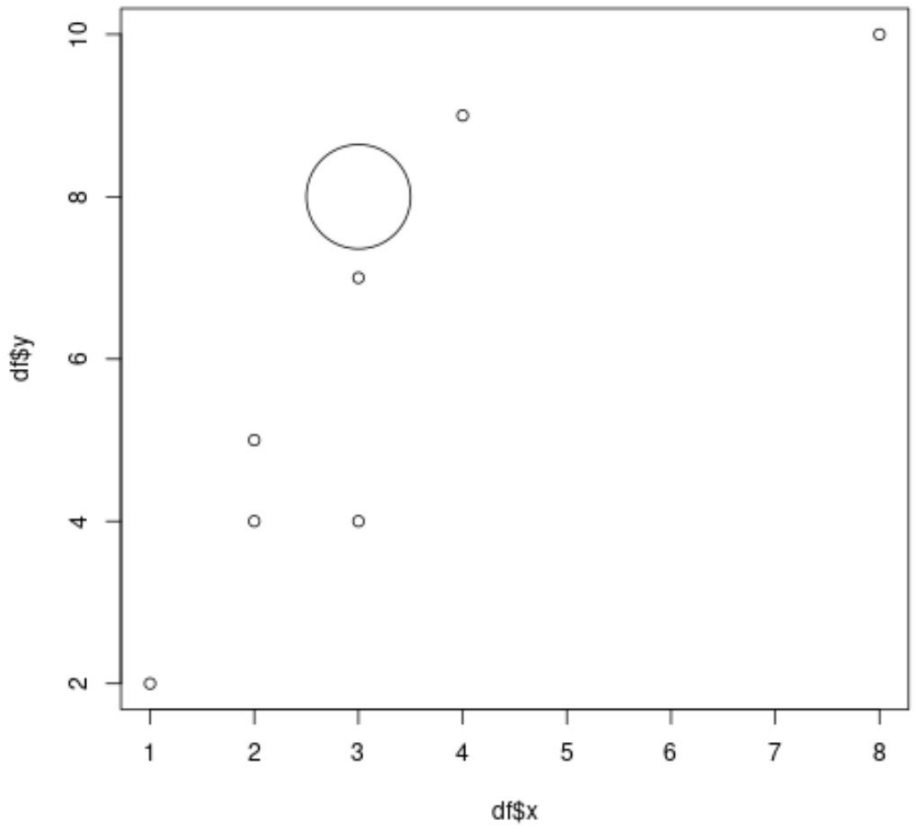
```
install.packages('plotrix')
library(plotrix)
```

Imagine a scenario in which you are working with a collection of data points and need to visually isolate or highlight a specific clustered region or an unusual outlier. The `draw.circle()` function proves exceptionally useful in this context. The subsequent code block first constructs a small synthetic dataset, then generates the scatter plot based on this data, and finally adds a prominent, unfilled circle centered precisely at the coordinates (3, 8) with a defined radius of 0.5 units.

```
#create data frame
df <- data.frame(x=c(1, 2, 2, 3, 3, 4, 8),
y=c(2, 4, 5, 4, 7, 9, 10))

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

#add circle
draw.circle(x=3, y=8, radius=.5)
```



As clearly illustrated in the resulting plot above, a distinct, geometrically accurate circle is now visible, precisely centered at the specified (x, y) coordinates. This straightforward graphical addition can significantly enhance the interpretability of your data visualization, immediately drawing the viewer's attention to the designated area of interest. The capability to precisely control the position and size of these circles establishes `draw.circle()` as an exceptionally powerful and flexible tool for annotating and enriching Base R plots.

Example 2: Adding Multiple and Customized Circles with Base R

The utility of the `draw.circle()` function extends far beyond merely plotting a single circle; it facilitates the addition of multiple circles to the same visualization, each capable of having its own independent set of aesthetic customizations. This inherent flexibility is invaluable when the analytical objective requires highlighting several distinct regions, outlier groups, or categories within the displayed data simultaneously. You retain the ability to specify unique colors, line types, and line widths for every circle added, enabling a sophisticated level of visual distinction and clarity.

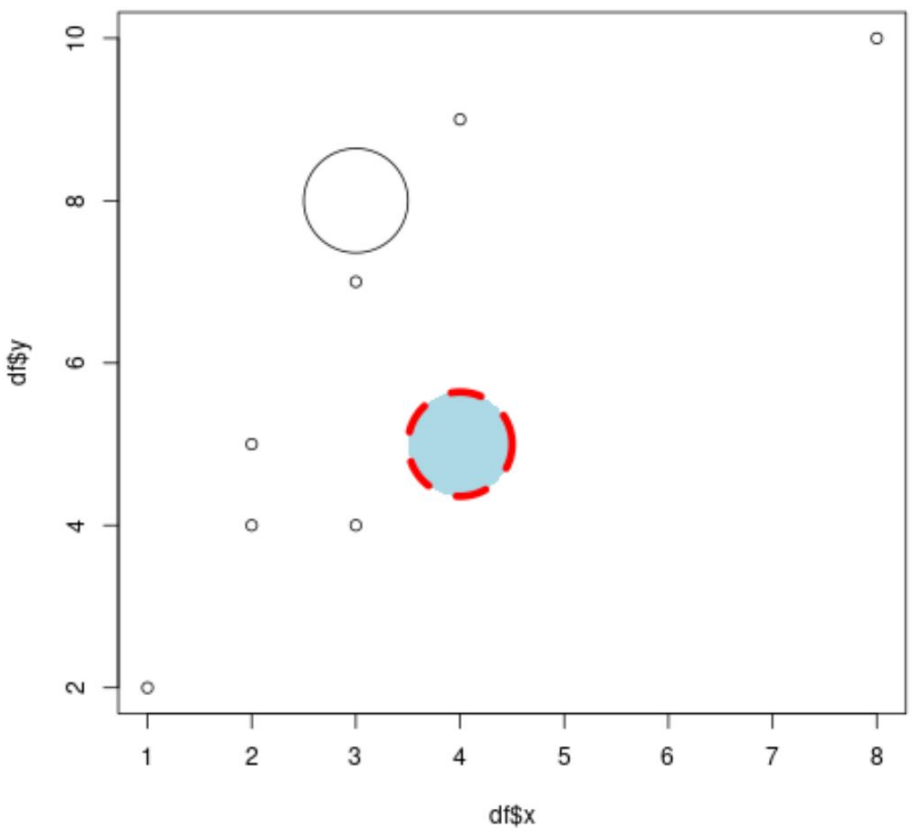
To effectively demonstrate this advanced capability, we will reuse the identical **data frame** and scatter plot setup utilized in the previous example. This time, however, we will overlay two distinct

circles. The second circle will be extensively customized using several optional parameters: it will feature a vivid red border, a contrasting light blue fill, a significantly thicker line weight, and a dashed line style. This demonstration effectively showcases the extensive range of customization options made available through the function's parameters, allowing for highly specific visualization requirements.

#create data frame
df <- data.frame(x=c(1, 2, 2, 3, 3, 4, 8),
y=c(2, 4, 5, 4, 7, 9, 10))

#create scatter plot
plot(dfx, dfy)

#add multiple circles to plot
draw.circle(x=3, y=8, radius=.5)
draw.circle(x=4, y=5, radius=.5, border='red', col='lightblue', lwd=5, lty='dashed')



Observe how the resulting plot now prominently features two distinct circles, positioned precisely at the (x, y) coordinates we specified. The first circle maintains its default, unadorned appearance, while the second circle immediately stands out due to its customized aesthetic properties. This

visually compelling result powerfully demonstrates the capability of the `draw.circle()` function to not only add geometric shapes but also to style them according to sophisticated visualization needs, making your plots dramatically more informative and visually appealing. It is important to remember that in **Base R**, each separate call to `draw.circle()` effectively adds a new, independent graphical layer directly on top of the existing plot.

Method 2: Drawing Circles Using ggplot2 and the ggforce Package

For data analysts and researchers who favor a more structured, declarative, and layered approach to data visualization within R, the renowned **ggplot2 package** is an indispensable and widely adopted tool. Constructed upon the theoretical foundation of the "grammar of graphics," **ggplot2** enables users to construct complex, high-quality plots layer by layer, providing unparalleled control, consistency, and flexibility over every visual element. While **ggplot2** includes a comprehensive collection of default geometric objects (geoms), the functionality to draw geometrically perfect circles requires the integration of an additional, specialized extension package: **ggforce**.

The **ggforce package** serves to significantly extend **ggplot2**'s native capabilities by introducing a range of new geoms, most notably `geom_circle()`. This function is purpose-built for the task of drawing circles and integrates seamlessly into the established **ggplot2** workflow. Using this powerful geom, you can define your circles using standard **aesthetic mappings** and precisely control their appearance within the structured, layered plotting system. Just as with the **plotrix** method, the prerequisite for using this approach is the installation and concurrent loading of both the **ggplot2** and **ggforce** packages before initiating your plotting code.

```
library(ggplot2)
library(ggforce)

#create scatter plot with circle at specific location with specific radius
ggplot(data = df, aes(x, y)) +
geom_point() +
geom_circle(aes(x0=3, y0=8, r=1), inherit.aes=FALSE) +
coord_fixed()
```

The `geom_circle()` function within **ggforce** requires specific **aesthetic mappings** to define the circle's geometry: `x0` and `y0` for the center coordinates, and `r` for the radius. These three parameters are absolutely critical for accurately defining the circle's position and size. Additionally, users can define visual properties such as `color` for the border, `fill` for the interior, `linetype`, and `lwd` (line width) either directly within the function call or by mapping them to variables in the underlying data. A fundamental and critical component when adding circles using **ggplot2** is the

function **coord_fixed()**, which must be included to ensure that the plot's aspect ratio is fixed. This crucial step prevents the circle from being geometrically distorted and appearing as an ellipse.

Example 3: Drawing a Circle with ggplot2

We will now illustrate the precise methodology for integrating `geom_circle()` into a high-quality **ggplot2** **scatter plot** visualization. This example will clearly demonstrate the process of defining a **data frame**, generating a basic scatter plot using the `geom_point()` layer, and subsequently layering a highly customized circle using `geom_circle()`. We will pay particular attention to highlighting the indispensable role of **coord_fixed()** in maintaining the true, undistorted circular shape.

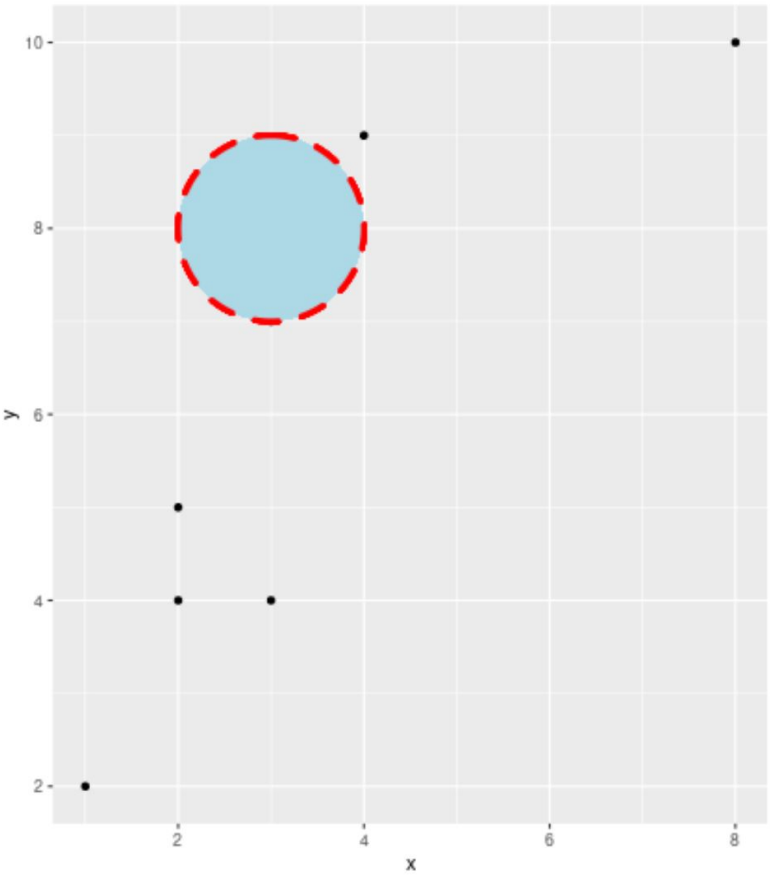
To begin, ensure the necessary packages--**ggplot2** and **ggforce**--are installed and successfully loaded into your R session:

```
install.packages('ggplot2')
install.packages('ggforce')
library(ggplot2)
library(ggforce)
```

The following code block constructs a dataset analogous to those used in our previous examples. It then meticulously builds a **ggplot2** object, adds the scatter data points, and finally layers the customized circle. This circle is precisely centered at the coordinates (3, 8) with a radius of 1 unit, defined with a dashed red border and a semi-transparent light blue fill. The argument `inherit.aes = FALSE` is a common and recommended practice when the aesthetics for the specific geom (in this case, the circle's position) are hardcoded or defined independently of the main plot's global aesthetic mappings.

```
#create data frame
df <- data.frame(x=c(1, 2, 2, 3, 3, 4, 8),
y=c(2, 4, 5, 4, 7, 9, 10))

#create scatter plot with circle
ggplot(data = df, aes(x, y)) +
geom_point() +
geom_circle(aes(x0=3, y0=8, r=1), linetype='dashed', color='red',
fill='lightblue', lwd=1.5, inherit.aes=FALSE) +
coord_fixed()
```



The final plot clearly displays the circle accurately positioned at the specified (x, y) coordinates, exhibiting its defined radius and custom visual styles. This outcome successfully demonstrates the powerful and inherently intuitive layering capabilities that define the **ggplot2** visualization framework. It is absolutely vital to include the **coord_fixed()** function within your **ggplot2** code whenever drawing circles or other geometric shapes requiring proportional scaling. Failure to include this function means the plot's default aspect ratio might inadvertently distort the circle, causing it to render improperly as an ellipse, especially if the scaling of the x and y axes are unequal. **coord_fixed()** guarantees that one unit length on the x-axis is visually equivalent to one unit length on the y-axis, thereby preserving the true, accurate geometry of your shapes.

Conclusion

Drawing accurate circles on plots in R is an exceptionally valuable technique for substantially enhancing data visualizations, providing a robust method for highlighting specific data points, emphasizing critical regions, or offering essential contextual annotations. This guide has thoroughly examined two distinct, yet equally effective, methodological approaches for accomplishing this task: utilizing the powerful `draw.circle()` function provided by the **plotrix package** within the foundational **Base R plotting system**, and employing the specialized

`geom_circle()` function from the **ggforce package**, which functions as a critical extension to the structured **ggplot2** environment.

The choice between these two methods ultimately depends on your preference for visualization style: you may opt for the rapid simplicity and directness offered by **Base R**, or you might prefer the aesthetic flexibility, structured layering, and consistency inherent in **ggplot2**. Regardless of your preference, both approaches provide the capability to create highly informative and visually engaging plots. Always remember the crucial preparatory step: installing and loading the necessary packages (**plotrix** for Base R, and **ggplot2** combined with **ggforce** for the latter) before executing your plotting code. We highly recommend experimenting extensively with the various customization parameters available for each function to ensure your circles' appearance perfectly aligns with and complements the narrative of your data story.

By achieving mastery of these essential techniques, you can effectively introduce an additional layer of sophistication and analytical depth to your R data visualizations, ensuring that your analyses are conveyed with greater clarity and your presentations achieve maximum impact. We strongly encourage you to practice the examples provided and integrate these circle annotation methods seamlessly into your personal analytical workflows.

Additional Resources

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

How to Create a Boxplot in R (With Examples)

How to Create a Histogram in R (With Examples)

How to Calculate the Correlation Between Two Variables in R