Make Pie Charts in ggplot2 (With Examples)

Authored by

Mohammed looti

November 7, 2025

The Pie chart remains a fundamental component of effective data visualization. Its primary purpose is to visually represent numerical proportions, where the complete circle symbolizes **100%** of the whole, and each distinct segment, or slice, illustrates the proportional contribution of a given category. These diagrams are exceptionally useful for communicating the precise breakdown of categorical data in a highly accessible format.

This comprehensive tutorial offers a step-by-step methodology for generating, rigorously customizing, and professionally refining circular visualizations in R. We leverage the capabilities of the robust and highly adaptable ggplot2 package. By mastering the techniques outlined here, you will learn how to transform a standard stacked bar chart into a polished, publication-ready pie chart, bypassing the common pitfalls associated with coordinate transformations.

Setting Up the Environment and Preparing Proportional Data

Successful data visualization begins with proper preparation. Before initiating the plotting process, it is essential to ensure that the necessary libraries are loaded and that the input data is correctly structured for proportional mapping. For all examples in this guide, we rely exclusively on the `ggplot2` package, which is a foundational element within the broader tidyverse ecosystem in R.

The structure of a pie chart inherently requires two distinct types of variables: a categorical identifier (e.g., 'A', 'B', 'C') and a numerical measure representing the size or proportion of that category. The code block below defines a simple data frame that meets these requirements, containing four categories and their respective numerical amounts. This structure is foundational for mapping aesthetics in `ggplot2`.

Ensure that `ggplot2` is installed and loaded before proceeding, as all subsequent functions depend on its availability. This initial step of data loading and library integration guarantees a smooth execution of the visualization script.

The Core Technique: Transformation via coord_polar()

A common misconception is that `ggplot2` includes a dedicated `geom_pie()` function. In reality, the library achieves circular plots through a sophisticated coordinate transformation process. The standard approach involves constructing a stacked bar chart and then converting its Cartesian coordinates into polar coordinates using the powerful `coord_polar()` function. Understanding this transformation is critical to mastering pie chart generation within the `ggplot2` framework.

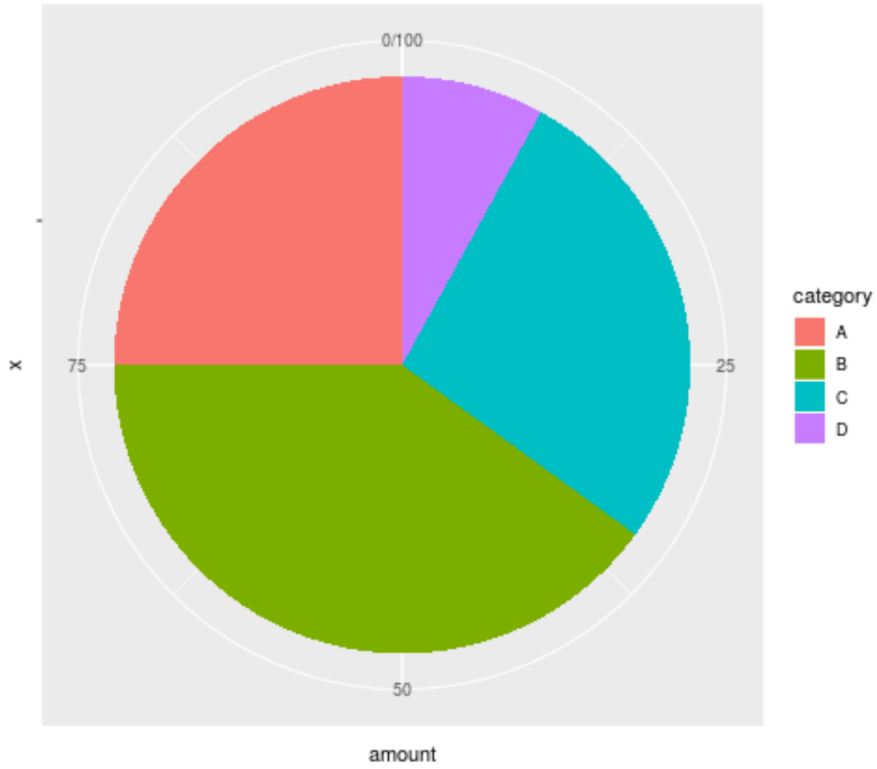
The initial visualization step utilizes `geom_bar(stat="identity")`. Setting the `stat` argument to `"identity"` ensures that the height of each bar is mapped directly to the numerical `amount` variable in our data frame. We also specify `x=""` in the aesthetic mapping to create a single stacked bar that fills the entire x-axis space, preparing it for the circular transformation.

The critical transformation occurs with `coord_polar("y")`. This function warps the rectangular bar plot: the y-axis (height/amount) becomes the radius of the circle, and the x-axis (category) is mapped to the angle. This conversion instantly produces the familiar circular structure of the pie chart.

library(ggplot2)

#create data frame
data <- data.frame("category" = c('A', 'B', 'C', 'D'),
"amount" = c(25, 40, 27, 8))

#create pie chart
ggplot(data, aes(x="", y=amount, fill=category)) +
geom_bar(stat="identity", width=1) +
coord_polar("y", start=0)



As evident in the output above, the chart is structurally sound but suffers from significant aesthetic clutter. Default elements such as the background grid lines, visible axes, and extraneous labels severely detract from the intended proportional message. The subsequent steps will focus entirely on eliminating this visual noise to produce a clean graphic.

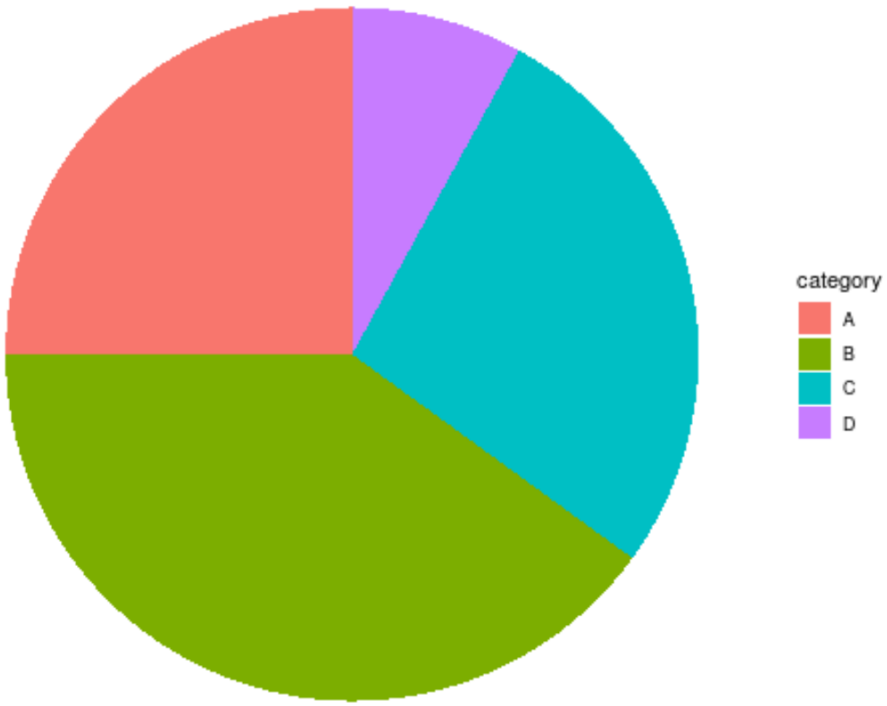
Enhancing Visual Clarity Using theme_void()

One of the primary challenges in transforming a Cartesian plot into a circular diagram is managing the residual grid lines and axis text inherited from the underlying bar plot. These non-data ink elements obscure the proportional relationships and can make quick interpretation of the chart extremely difficult for the audience. A clean presentation is paramount for effective communication.

The most straightforward and efficient solution for addressing these inherited aesthetic issues within the ggplot2 library is the application of the built-in `theme_void()` function. This specialized theme is meticulously engineered to strip away all superfluous graphical components, including the panel background, major and minor grid lines, axis lines, and axis labels.

By appending `theme_void()` to our plotting command, we immediately achieve a cleaner presentation, allowing the focus to shift entirely onto the colored proportional slices, thus maximizing the chart's readability and professional appeal.

```
ggplot(data, aes(x="", y=amount, fill=category)) +
geom_bar(stat="identity", width=1) +
coord_polar("y", start=0) +
theme_void()
```



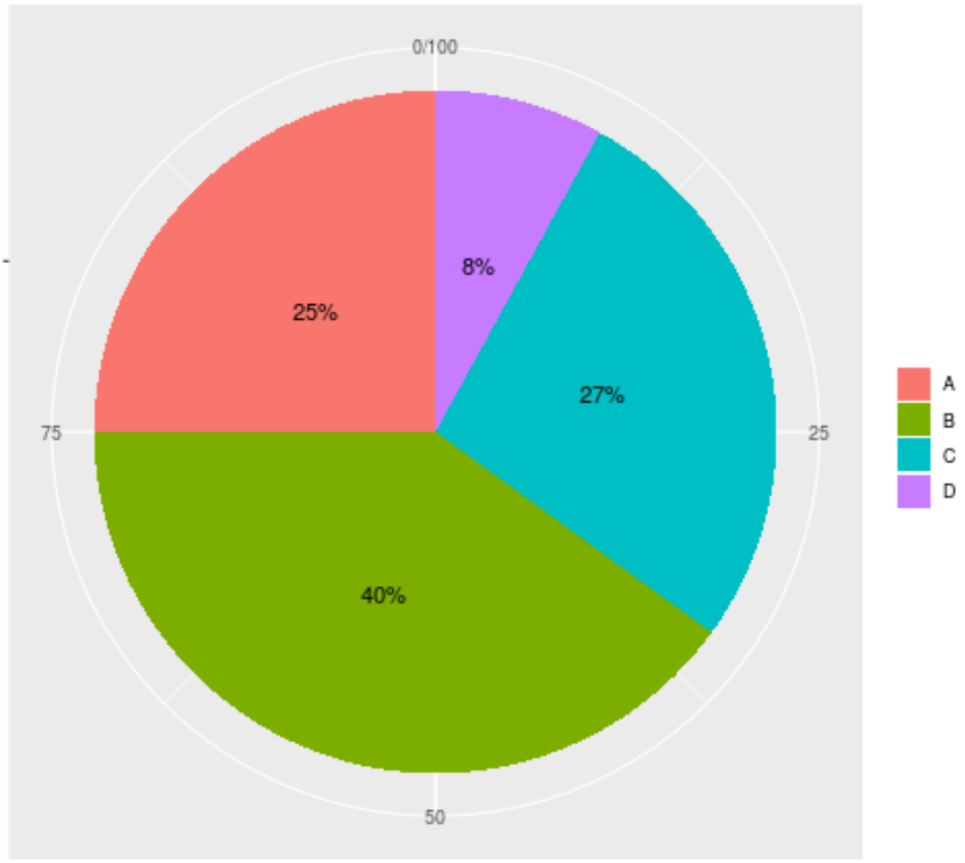
Incorporating Data Labels and Final Layout Refinement

While `theme_void()` successfully cleans the canvas, the chart loses essential quantitative context without axis markings. To restore clarity and ensure the chart is fully informative, we must strategically place data labels directly onto the slices. This process requires using the `geom_text()` layer combined with precise positioning arguments to prevent label overlap or protrusion outside the pie segments.

To achieve central placement of the labels within each slice, we rely on the `position_stack(vjust=0.5)` argument. This critical setting calculates the midpoint of the stacked bar segment before the polar transformation, ensuring the label is vertically centered within its corresponding slice. We also use the `labs()` function to remove any residual or unnecessary axis titles and legend titles, promoting a minimalist, self-contained graphic.

For enhanced audience readability, we integrate the `paste0()` function within the `geom_text()` aesthetic mapping. This allows us to append a percentage sign (or any desired unit) directly to the numerical value, clearly communicating that the numbers represent proportions of the whole.

```
ggplot(data, aes(x="", y=amount, fill=category)) +
geom_bar(stat="identity", width=1) +
coord_polar("y", start=0) +
geom_text(aes(label = paste0(amount, "%")), position = position_stack(vjust=0.5)) +
labs(x = NULL, y = NULL, fill = NULL)
```



Advanced Color Customization: Manual Hex Codes and Theme Refinement

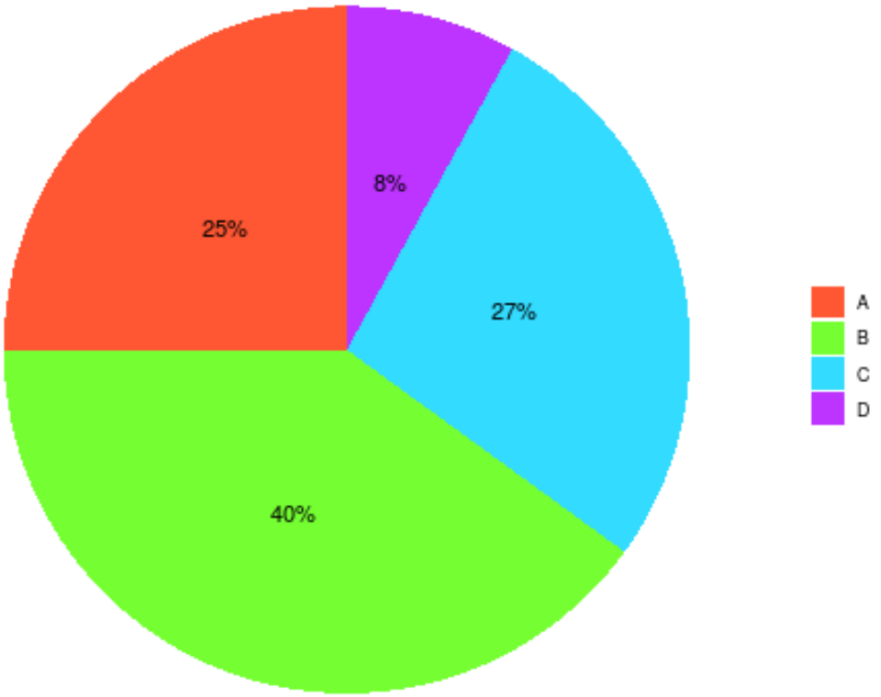
While default color palettes are sufficient for initial drafts, professional presentations often demand specific color schemes that align with corporate branding or specific visual themes. ggplot2 offers exceptional flexibility in color control through the `scale_fill_manual()` function. This tool allows the user to specify exact hex color codes for every category, ensuring precise aesthetic harmony.

In this advanced example, we demonstrate how to combine multiple refinement steps for a highly tailored output. We first utilize `theme_classic()`, which provides a clean baseline, and then explicitly remove specific axis elements (lines, text, ticks) using the comprehensive `theme()` function. This layering approach ensures that no unnecessary visual elements remain while providing a solid structural foundation.

The `scale_fill_manual(values=c(...))` argument is crucial here: the provided vector of hex color codes is mapped sequentially to the categories defined in the data frame, completely overriding the library's default color choices. This level of customization is vital for publication-quality graphics.

ggplot(data, aes(x="", y=amount, fill=category)) +

```
geom_bar(stat="identity", width=1) +
coord_polar("y", start=0) +
geom_text(aes(label = paste0(amount, "%")), position = position_stack(vjust=0.5)) +
labs(x = NULL, y = NULL, fill = NULL) +
theme_classic() +
theme(axis.line = element_blank(),
axis.text = element_blank(),
axis.ticks = element_blank()) +
scale_fill_manual(values=c("#FF5733", "#75FF33", "#33DBFF", "#BD33FF"))
```



If you need assistance in selecting appropriate colors, consider utilizing an external Hex Color Picker to find visually compelling and complementary hex color codes for your specific data narrative.

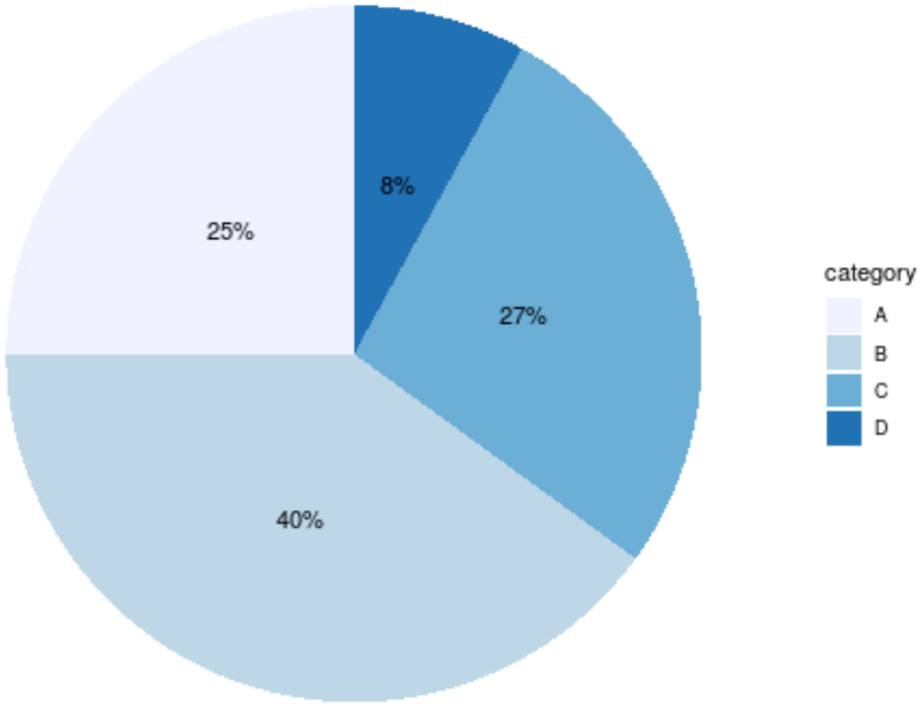
Utilizing Built-in Color Palettes via scale_fill_brewer()

When manual selection of individual hex color codes is impractical, or if compliance with accessibility standards (such as colorblind-friendliness) is required, `ggplot2` offers seamless integration with the robust ColorBrewer system through the `scale_fill_brewer()` function. This feature provides immediate access to a vast array of high-quality, predefined color schemes optimized for various data types (sequential, diverging, and qualitative).

To employ this function, you only need to specify the desired palette name (e.g., "Blues" for sequential shades, "Set1" for distinct categories) within the `palette` argument. This method represents a best practice in data visualization, as it guarantees professional color gradients and contrast without requiring intensive manual effort.

By switching from `scale_fill_manual()` to `scale_fill_brewer()`, we maintain all the structural and labeling refinements while instantly adopting a perceptually uniform and effective color scheme, showcasing the power and flexibility of theme layering in R.

```
ggplot(data, aes(x="", y=amount, fill=category)) +
geom_bar(stat="identity", width=1) +
coord_polar("y", start=0) +
geom_text(aes(label = paste0(amount, "%")), position = position_stack(vjust=0.5)) +
labs(x = NULL, y = NULL) +
theme_classic() +
theme(axis.line = element_blank(),
axis.text = element_blank(),
axis.ticks = element_blank()) +
scale_fill_brewer(palette="Blues")
```



Conclusion and Next Steps for ggplot2 Mastery

Generating a polished pie chart in ggplot2 is less about finding a specific geom function and more about skillfully executing coordinate transformation and applying detailed theme customizations. By mastering the sequence of building a stacked bar, transforming it with `coord_polar()`, and refining the aesthetics using advanced theme functions, you can create highly customized and informative statistical graphics.

The flexibility and power offered by the ggplot2 library make it the industry standard for virtually any type of statistical graphic required for robust analysis in R. We encourage you to continue exploring the depth of this package.

To further advance your skills in R data visualization, consider reviewing these related tutorials focused on advanced ggplot techniques:

How to Create a Grouped Boxplot in R Using ggplot2
How to Create a Heatmap in R Using ggplot2
How to Create a Gantt Chart in R Using ggplot2