Learning to Visualize Data: Creating Stacked Dot Plots in R

Authored by

Mohammed looti

November 6, 2025

The **stacked dot plot** stands as a highly effective graphical technique employed in statistical visualization to clearly illustrate the frequency distribution of a given dataset, whether it contains continuous or discrete variables. This visualization offers a significant advantage over methods like the histogram because it avoids grouping observations into arbitrary bins. Instead, the stacked dot plot retains and displays every single individual data point, thereby making the underlying distribution, frequency counts, and specific data values immediately transparent to the analyst.

This method of graphical representation is particularly well-suited for smaller to moderate-sized datasets, or when the data primarily consists of integer values. The inherent design clarity allows for rapid identification of key distributional features, such as distinct clusters, noticeable gaps in the data, and potential **outliers**. Fundamentally, in a stacked dot plot, each mark, or dot, signifies a single observation. The vertical arrangement, or stacking, of these dots directly corresponds to the frequency--indicating precisely how many times a specific value appears within the dataset. This high-resolution approach to displaying frequency provides superior interpretability compared to many other common frequency plots.

For data scientists and analysts operating within the **R** statistical computing environment, there are two primary, well-established methodologies for constructing these insightful plots. Each method caters to slightly different requirements concerning efficiency, customization, and integration with larger data workflows. Understanding both techniques is essential for comprehensive data visualization mastery in R.

Method 1: The Native Approach. This involves leveraging the core functions included in **base R**, specifically utilizing the versatile `stripchart()` function combined with the appropriate argument settings.
Method 2: The Modern Framework. This employs the immensely popular and robust **ggplot2** package, which offers advanced, layered control over graphical elements through its specialized function, `geom_dotplot()`.

This extensive tutorial serves as a practical guide, offering detailed, step-by-step instructions and reproducible code examples for successfully implementing both the base R and the ggplot2 methods. By mastering both approaches, users can generate precise, aesthetically pleasing stacked dot plots that are perfectly tailored to meet specific data analysis and reporting requirements.

Choosing the Right Tool for Dot Plots

The decision to utilize either the foundational tools provided by **base R** or the specialized capabilities of the **ggplot2** package often hinges on the project's precise needs. Key considerations include the required depth of visual customization, the need for adherence to specific aesthetic or publication standards, and the tolerance for external package dependencies.

Both environments are fully capable of generating statistically accurate and visually effective stacked dot plots, but they achieve their results through fundamentally divergent programming and visualization philosophies.

The **base R** methodology, centered around the `stripchart()` function, offers the significant advantage of speed and zero external dependencies. This makes it the ideal choice for quick, exploratory data analysis, rapid prototyping, or in environments where the installation or loading of additional packages is heavily restricted. However, the subsequent process of customizing the plot--such as fine-tuning elements like fonts, applying specific color palettes, or integrating the graphic seamlessly within a complex multi-panel figure--can sometimes prove cumbersome. Customization in base R often demands a deeper, more specialized familiarity with its legacy plotting functions and parameter settings, which can present a steeper learning curve for advanced aesthetic control.

In contrast, **ggplot2**, which is a core component of the tidyverse ecosystem, is built upon the foundational principles of the **Grammar of Graphics**. This paradigm dictates that plots are constructed systematically by layering distinct components: the data source, aesthetic mappings (how variables relate to visual attributes), geometric objects (the plot type), and scales. Although it necessitates loading an external package, the resultant plots are renowned for being highly customizable, exhibiting exceptional aesthetic consistency, and being remarkably straightforward to read, interpret, and modify for those familiar with its layered structure. For producing complex statistical graphics intended for formal publication, detailed reporting, or sharing within a standardized organizational aesthetic, **ggplot2** is overwhelmingly the preferred and most versatile choice among R practitioners.

Example 1: Generating a Stacked Dot Plot with Base R

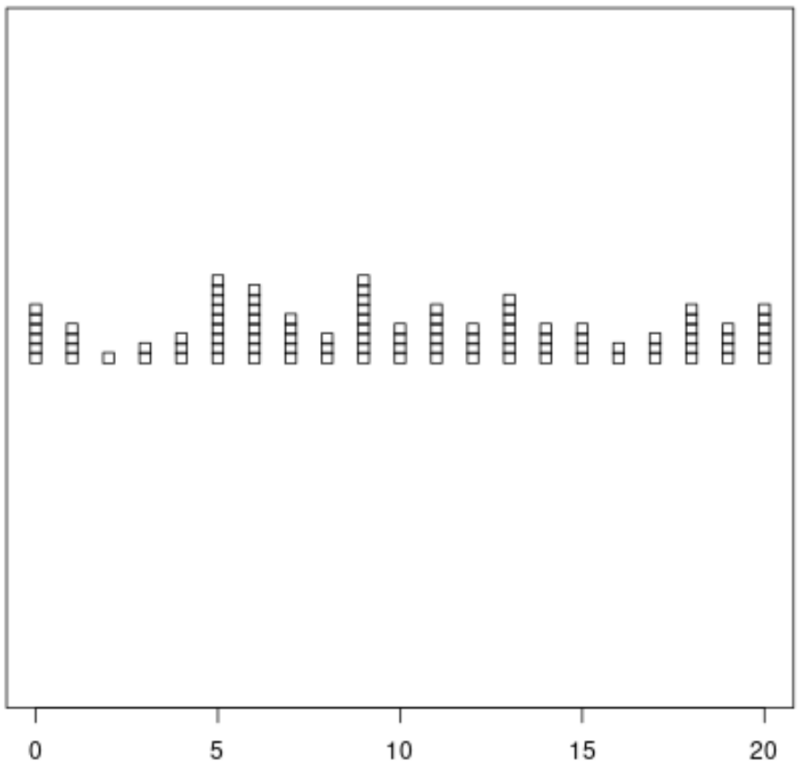
The `stripchart()` function, a component of **base R**, is traditionally designed for creating one-dimensional scatter plots. However, by strategically modifying the key `method` argument, this function can be repurposed to generate a highly functional **stacked dot plot**. This initial example focuses on the most basic implementation, showcasing the fundamental command needed to achieve the stacking effect, before moving on to a more visually refined version.

The necessary initial step involves generating a reproducible sample dataset that we can analyze. We begin by invoking `set.seed(0)` to ensure that the randomly generated data remains identical upon every execution of the script, guaranteeing reproducibility. Following this, we use the `sample()` function to generate 100 random integers that fall between 0 and 20. The essential plotting command is `stripchart()`, where the critical instruction is delivered via `method = "stack"`. This argument explicitly directs the function to stack the individual dots vertically whenever the underlying data values overlap, thereby creating the visual representation of

frequency counts.

```
# Set a seed for reproducible results
set.seed(0)
data <- sample(0:20, 100, replace = TRUE)

# Create the basic stacked dot plot using the 'stack' method
stripchart(data, method = "stack")
```



The resulting graphical output, generated quickly and efficiently by the basic code, successfully visualizes the frequency distribution of our generated sample data. While the plot is functionally correct and conveys the necessary statistical information, the default aesthetics are inherently minimal. It lacks crucial elements such as clear titles, informative axis labels, and customized dot appearance, which are necessary for professional or publication-ready graphics. The following detailed section outlines the essential steps and arguments required to significantly enhance the visual quality and interpretability of this plot.

Customizing Appearance in Base R: Enhancing Readability

To elevate the visual quality and significantly improve the readability of a dot plot produced by `stripchart()`, it is necessary to utilize several of **base R**'s key graphical parameters. Effective

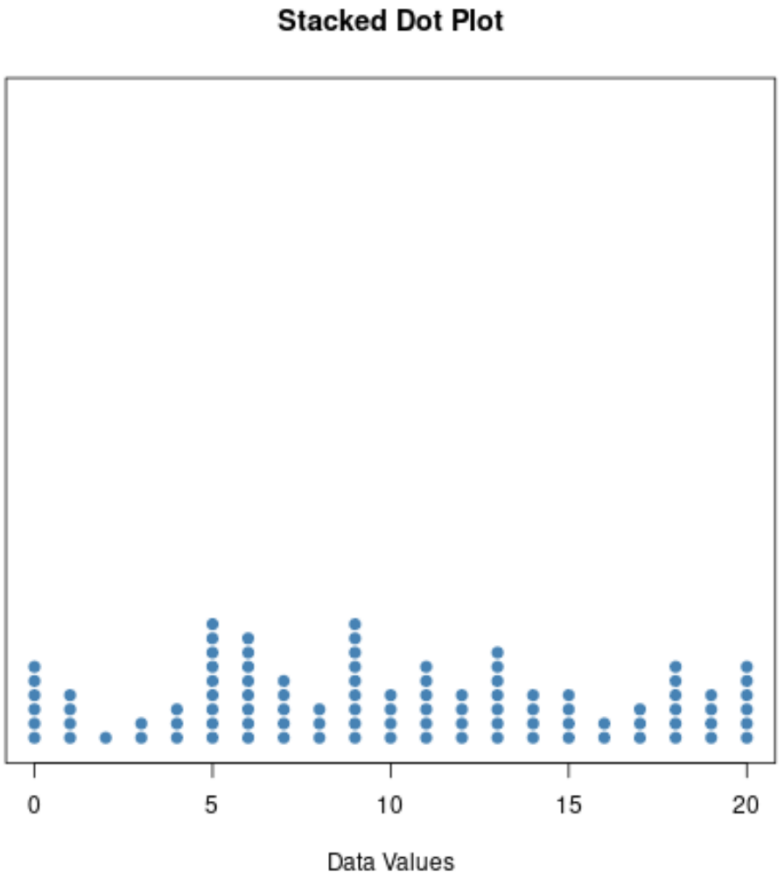
customization is often non-negotiable for creating publication-ready graphics that clearly and compellingly communicate the data's narrative. The parameters detailed below are foundational for achieving a polished and accurate visualization using the **base R** plotting system.

For fine-tuning the visual aspects of the stacked dots and the overall plot presentation, four specific parameters are particularly crucial. The `offset` argument controls the vertical spacing applied between the individual dots as they stack up. A moderate value, such as `.5`, is typically recommended as it achieves a good balance, preventing dot overlap while still maintaining a cohesive visual connection between the stacked points. The plotting character is determined by `pch`; setting this to `19` results in solid, filled circles, which are generally preferred for superior clarity over hollow or complex shapes. Furthermore, the `col` parameter defines the color of the dots; selecting a distinct and professional color, such as `"steelblue"`, dramatically enhances the visual appeal and focus. Finally, the `main` and `xlab` arguments are used to define the primary title of the plot and the label for the horizontal x-axis, respectively, providing essential contextual information for the viewer.

Applying these carefully chosen adjustments transforms the raw, default visualization into a sophisticated and easily digestible graphical representation. This demonstrates the powerful capacity of the `stripchart()` function when its comprehensive set of arguments is fully utilized within the robust environment of **base R**, allowing analysts to move beyond simple data representation toward clear visual storytelling.

```
# Generate reproducible sample data
set.seed(0)
data <- sample(0:20, 100, replace = TRUE)

# Create customized stacked dot plot with enhanced aesthetics
stripchart(data, method = "stack", offset = .5, at = 0, pch = 19,
col = "steelblue", main = "Stacked Dot Plot", xlab = "Data Values")
```



Example 2: Implementing Stacked Dot Plots with ggplot2

For users who prefer the highly structured, consistent, and aesthetically advanced framework offered by the **ggplot2** package, the process of creating a stacked dot plot is managed primarily through the dedicated `geom_dotplot()` geometry. This function is designed to intelligently handle the intricate processes of binning, scaling, and stacking the data required for this specific visualization type, all while remaining compliant with the package's governing principles of the Grammar of Graphics.

Prior to initiating the plotting process, two key preliminary steps must be addressed. First, the **ggplot2** library must be explicitly loaded into the R session using the `library()` function. Second, unlike **base R** which can often work directly with vectors, **ggplot2** requires the input data to be structured as an R **data frame**--this is the mandatory and preferred format for the package. The construction of the plot then follows the standard ggplot2 syntax: mapping the variable (in this case, the single variable `x`) to the aesthetic layer (`aes()`), and subsequently adding the `geom_dotplot()` layer, which is responsible for rendering the visually stacked dots.

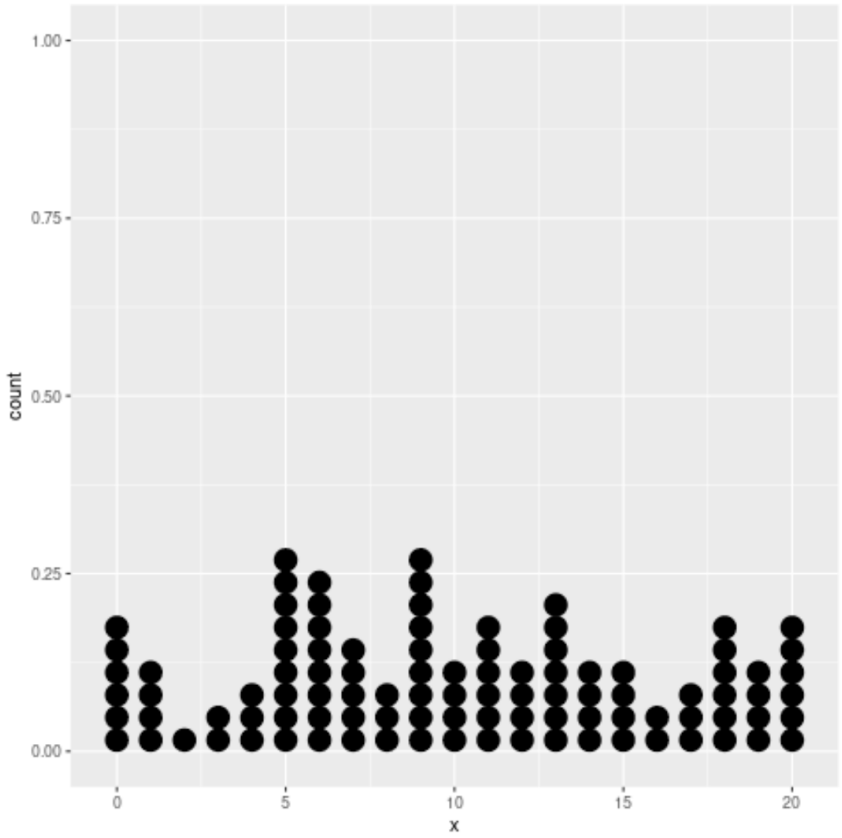
The code provided below illustrates the fundamental sequence of commands necessary to generate a basic **ggplot2** dot plot. A notable characteristic of this package is its reliance on smart

default settings; **ggplot2** often manages the default scaling, bin widths, and stacking ratios automatically and intelligently, frequently resulting in a visually appealing and statistically sound plot even in its most foundational form, minimizing the initial need for heavy customization.

```
# Load the ggplot2 package
library(ggplot2)

# Create data frame for ggplot2
set.seed(0)
data <- data.frame(x = sample(0:20, 100, replace = TRUE))

# Create the basic stacked dot plot using geom_dotplot
ggplot(data, aes(x = x)) +
geom_dotplot()
```



The resulting output from the basic `geom_dotplot()` function provides a clear and accurate visualization of the frequency distribution. One immediate and crucial distinction from the **base R** equivalent is **ggplot2**'s reliance on a continuous y-axis scale by default. This axis typically represents density or count information internally, rather than a specific, measured variable.

Consequently, a standard refinement technique in **ggplot2** involves adding extra commands specifically designed to suppress or remove these default y-axis labels and tick marks, as the stacking of the dots itself is intended to convey the frequency information visually, rendering the explicit vertical axis scale redundant for interpretation.

Refining Aesthetics: Advanced Customization in ggplot2

The true utility and flexibility of the **ggplot2** framework become evident when leveraging its extensive customization options to produce publication-quality graphics. For stacked dot plots specifically, detailed control over the visual presentation is achieved by manipulating parameters within the `geom_dotplot()` function, combined with specific scale and label functions that manage the overall plot appearance. These advanced refinements are essential steps for maximizing the graphic's clarity, ensuring visual impact, and standardizing the output for professional use.

Precision in dot appearance and stacking geometry is managed through specific arguments. The `dotsize` parameter governs the relative size of the plotted dots; using a smaller value, such as `.75`, is often highly effective in preventing overcrowding and overlap, particularly when dealing with columns that exhibit high frequencies. The `stackratio` parameter controls the precise vertical spacing between dots within a single stack; setting this ratio slightly greater than 1 (e.g., `1.2`) introduces subtle vertical gaps, which significantly improves the visual distinction between adjacent points. The `fill` argument handles the interior color of the dots. Crucially, **ggplot2** allows for the systematic removal of the often-redundant y-axis using `scale_y_continuous(NULL, breaks = NULL)`. This technique is standard practice in dot plots because the vertical stack inherently represents the frequency count, making an explicit, labeled y-axis unnecessary and potentially distracting.

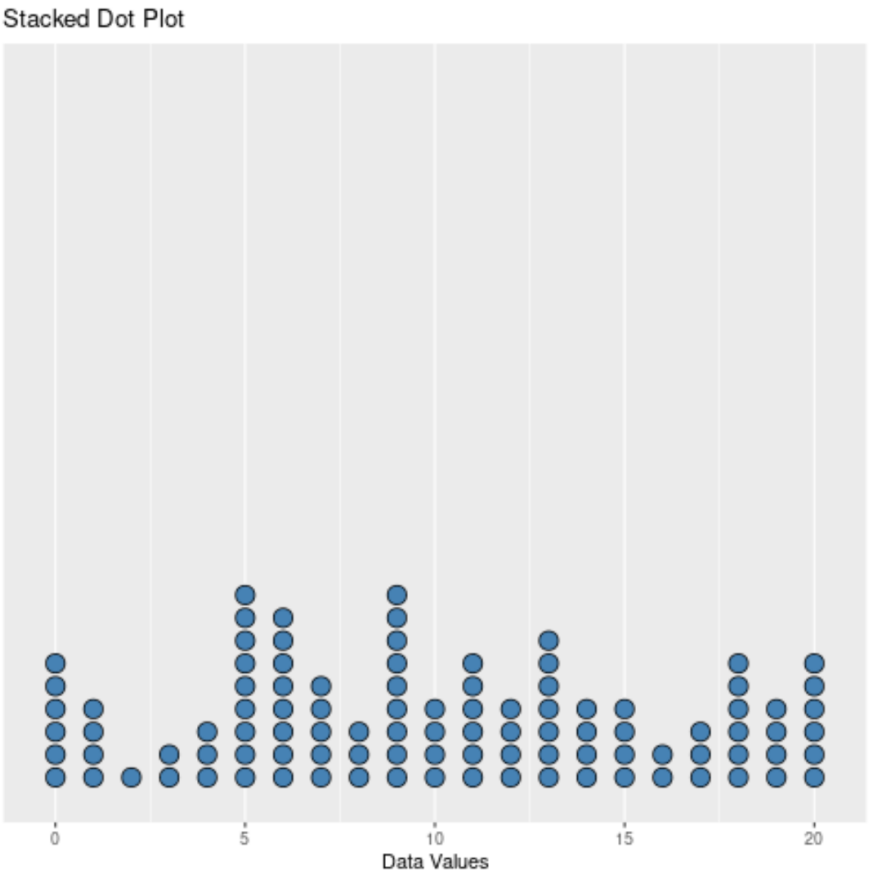
By integrating these advanced layers and parameters, we effectively transform the basic default plot into a highly refined, professional-grade graphic. This process demonstrates the sophisticated level of aesthetic control that the **ggplot2** framework provides, ensuring that the visualization is not only statistically accurate but also optimally presented for clear communication within the **R** environment.

```
# Load the ggplot2 package
library(ggplot2)

# Create data frame
set.seed(0)
data <- data.frame(x = sample(0:20, 100, replace = TRUE))

# Create customized stacked dot plot using detailed parameters
ggplot(data, aes(x = x)) +
```

```
geom_dotplot(dotsize = .75, stackratio = 1.2, fill = "steelblue") +
scale_y_continuous(NULL, breaks = NULL) +
labs(title = "Stacked Dot Plot", x = "Data Values", y = "")
```



Summary of Stacked Dot Plot Creation in R

The capacity to rapidly and effectively generate insightful data visualizations, such as the **stacked dot plot**, represents a foundational skill set for all exploratory data analysis conducted in the **R** environment. This specific visualization is invaluable for conveying frequency distributions in a granular manner, allowing the viewer to discern precise counts and data patterns without the loss of information associated with data aggregation.

The choice between the two primary methods--the efficient simplicity of **base R**'s `stripchart()` versus the structured, layered control provided by **ggplot2**'s `geom_dotplot()`--should be driven entirely by the operational context. **Base R** excels in scenarios demanding immediate results, minimal package dependencies, and lightweight scripting. It is the perfect solution for internal checks or initial data exploration where speed is paramount.

Conversely, **ggplot2** is the indispensable tool when the requirement shifts toward crafting highly

customized, complex, or aesthetically standardized visual reports destined for external audiences or formal publications. Its adherence to the Grammar of Graphics ensures consistency and facilitates the creation of complex, multi-layered statistical graphics with relative ease. Mastery of both these distinct visualization techniques guarantees exceptional versatility and adaptability in tackling virtually any statistical programming or data presentation challenge within R.

We encourage readers to explore more advanced R tutorials and comprehensive guides to further develop their expertise in statistical visualization methods and sophisticated data analysis techniques.