Learning to Create Frequency Polygons in R for Data Visualization

Authored by

Mohammed looti

November 2, 2025

The frequency polygon stands as a cornerstone method in modern data visualization, essential for effective statistical analysis and data science workflows. This graphical tool is specifically designed to illustrate the distribution of **continuous variables** within a given dataset. Unlike a conventional histogram, which relies on vertical bars to represent frequencies, the frequency polygon connects points plotted at the midpoint of each class interval using line segments, offering a distinct visual interpretation.

Frequency polygons are highly advantageous when the primary objective is to compare **multiple distributions** simultaneously or to achieve a smoother, more generalized representation of underlying data trends. By connecting the central points of the frequency bins, this chart clearly outlines the shape of the distribution, facilitating rapid identification of key characteristics such as **skewness**, **modality**, and overall spread. This streamlined approach offers greater clarity, especially when overlays are necessary.

The fundamental purpose of constructing this visualization is to intuitively grasp how data values are clustered and to pinpoint where the highest concentration of observations resides. Its methodology is intrinsically linked to the histogram; in practice, the frequency polygon is often derived directly from the same **binning structure** used to create the corresponding bar chart visualization.

Prerequisites: Setting up the R Environment

Generating a frequency polygon effectively within the R environment necessitates the use of the renowned and highly adaptable data visualization package, ggplot2. This package is built upon the foundational principles of the Grammar of Graphics, offering a robust, structured, and visually sophisticated framework for plotting statistical data. It is imperative that users confirm **ggplot2** is installed and accessible within their R session before attempting any plotting commands outlined here.

The initial procedural step involves loading the library using the `library()` function. This action makes available its extensive suite of tools, most importantly the critical `geom_freqpoly()` geometry, which handles the core plotting logic for frequency polygons. The standardized syntax requires defining the input **dataset** (often named `df`) and subsequently mapping the variable of interest--the measured value--to the x-axis aesthetic using the `aes()` function.

The fundamental code template below illustrates the minimal structure required to initialize a plot using **ggplot2** in **R**. This concise format defines the data source and specifies the appropriate geometry layer:

library(ggplot2)

```
ggplot(df, aes(value)) +
geom_freqpoly()
```

This straightforward structure forms the basis for constructing and customizing all subsequent frequency polygon visualizations. The practical examples that follow will demonstrate how this core syntax is extended and refined using simulated data to produce insightful and publication-ready graphics.

Example 1: Generating a Basic Frequency Polygon

To launch our practical demonstration, we begin by generating a basic frequency polygon using a controlled, simulated dataset. We initialize a data frame, creatively named `df`, populated with 100 observations sampled from a standard normal distribution, specifically parameterized with a mean (μ) of 50 and a standard deviation (σ) of 10. Crucially, we employ `set.seed(0)` to ensure complete **reproducibility**, allowing any user to generate the exact same data and corresponding visualization.

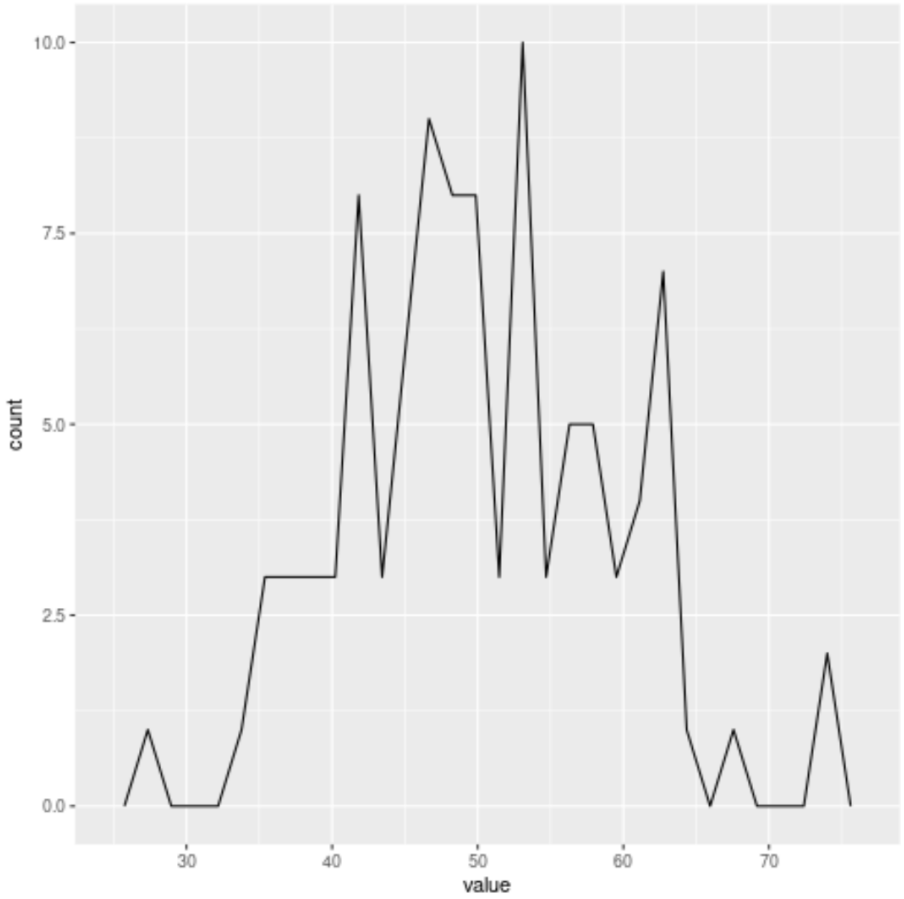
The heart of the plotting procedure involves binding the `value` column of our data frame to the x-axis aesthetic. Once this mapping is established, the `geom_freqpoly()` function takes over. This function performs the necessary statistical transformations: it automatically calculates the optimal number of **bins** (30 by default in **ggplot2**), determines the frequency count for observations within each bin, and finally plots the line segments connecting these midpoints. This results in the default, un-customized visualization.

The comprehensive code block below outlines both the data generation process and the command required to produce the initial plot:

```
library(ggplot2)

#make this example reproducible
set.seed(0)

#create data frame
df <- data.frame(index=1:100,
value=rnorm(100, mean=50, sd=10))

#create frequency polygon
ggplot(df, aes(value)) +
geom_freqpoly()
```

Upon reviewing the resulting visualization, the expected bell shape of the distribution is clearly

confirmed, highlighting the **central tendency** around the defined mean of 50. This fundamental plot establishes a critical baseline, demonstrating the intrinsic behavior of the plotting function before any advanced modifications are applied.



Controlling Data Resolution: Understanding and Customizing Bins

The visualization quality of both histograms and frequency polygons is critically dependent on the number of **bins** used to segment the continuous data range. By default, the **ggplot2** package employs **30 bins** when processing data for the frequency polygon. While this provides a balanced initial overview, this high granularity can often lead to a visualization that appears overly jagged, noisy, or complex, particularly when working with datasets containing smaller sample sizes.

The fundamental trade-off lies in the choice of bin size, which directly governs the visual smoothness and overall interpretability of the frequency polygon. Adopting wider bins, which corresponds to using fewer bins across the entire range, effectively smooths out minor, potentially artifactual fluctuations. This technique is excellent for clearly revealing the macro shape of the underlying distribution. Conversely, utilizing narrower bins (a greater number of bins) provides meticulous detail concerning local variations, but risks introducing visual noise or falsely

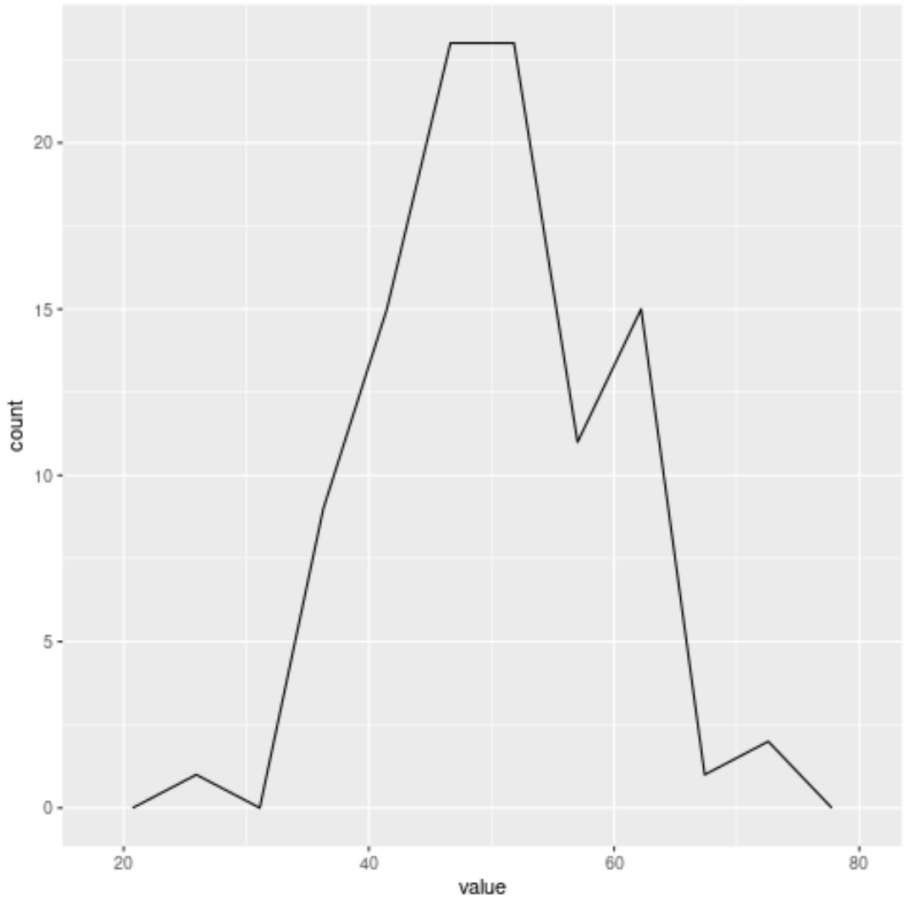
emphasizing minor modes that are merely random variations inherent to sampling.

For strategic data presentation, reducing the number of bins--for example, specifying only **10 bins**--forces the aggregation of data points into significantly larger intervals. This intentional smoothing effect is highly beneficial when the goal is to communicate overarching trends rather than focusing on granular data points. In **ggplot2**, this crucial customization is easily achieved by supplying the `bins` argument directly within the `geom_freqpoly()` function call, successfully overriding the package's default setting of 30.

The following revised code block applies this customization, utilizing the identical simulated data but explicitly setting the number of bins to **10** to achieve a demonstrably smoother line representation:

```
library(ggplot2)

#make this example reproducible
set.seed(0)

#create data frame
df <- data.frame(index=1:100,
value=rnorm(100, mean=50, sd=10))

#create frequency polygon
ggplot(df, aes(value)) +
geom_freqpoly(bins=10)
```

As demonstrated by the resulting plot below, the visualization now exhibits a significantly smoother curve when contrasted with the default output. This adjustment successfully makes the characteristic bell shape of the normal distribution more apparent, substantially improving the overall visual clarity and readability of the graphic.



Example 3: Creating a Filled Frequency Area Plot

While the line-based frequency polygon is excellent for structural comparisons, many data analysts prefer visualizations where the area beneath the curve is shaded. This technique, often referred to as an area plot, is highly effective for emphasizing **cumulative frequency** or visualizing the density of observations across defined ranges. However, it is important to note that the native `geom_freqpoly()` function in **ggplot2** is strictly limited to drawing lines and points and does not inherently support area filling.

To successfully achieve a filled frequency polygon visualization, we must substitute `geom_freqpoly()` with the more versatile `geom_area()` function. This geometry is designed to facilitate shading based on statistical transformation results. A critical configuration step involves explicitly mapping the y-aesthetic: we must use `aes(y=..count..)`. This specific command directs the geometry layer to utilize the calculated frequency count--derived during the internal binning process--as the vertical height for the area plot, ensuring accuracy.

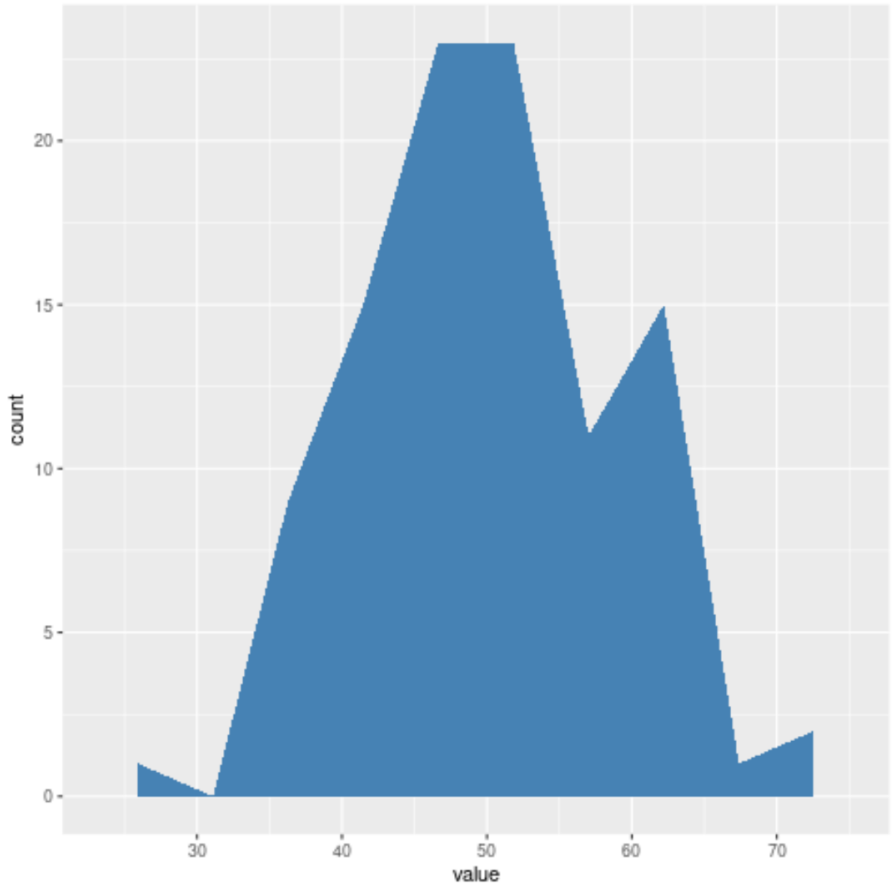
Furthermore, to ensure `geom_area()` processes the continuous data correctly into frequency counts, we must include the argument `stat='bin'` within the function call. This explicitly instructs

the layer to perform the necessary binning and counting steps, perfectly mirroring the implicit processes of `geom_freqpoly()`. Aesthetic customization is then applied using the `fill` argument, allowing us to select a distinct color, such as 'steelblue', to significantly enhance the visual impact and professionalism of the resulting distribution graphic.

The comprehensive code below illustrates how to effectively employ `geom_area()` to construct a frequency plot with a customized fill color, seamlessly integrating the previously established setting of **10 bins**:

```
library(ggplot2)

#make this example reproducible
set.seed(0)

#create data frame
df <- data.frame(index=1:100,
value=rnorm(100, mean=50, sd=10))

#create frequency polygon filled with custom color
ggplot(df, aes(value)) +
geom_area(aes(y=..count..), bins=10, stat='bin', fill='steelblue')
```

This resulting visualization delivers a clear, highly aesthetic representation of the data distribution, effectively using the shaded area to emphasize the magnitude and clustering of the data frequencies.



Interpreting the Frequency Polygon: Distributional Insights

Effective interpretation of a frequency polygon requires careful analysis of its primary visual features. The vertical height of the plotted points and the corresponding line segments directly corresponds to the **frequency count**--the number of observations contained within that specific class interval. More broadly, the overall contour or shape of the line is highly revealing, providing analysts with a rapid method to assess critical distributional patterns, including **symmetry**, skewness, and modality.

The morphology of the curve offers immediate diagnostic information. For instance, a polygon that rapidly ascends, achieves a single central peak, and then descends symmetrically suggests a highly idealized distribution, such as a normal distribution, frequently encountered in natural and controlled experimental data. Deviation from this symmetry is categorized by skewness: a tail extending toward higher values indicates **positive skewness**, while a tail extending toward lower values signals **negative skewness**. Additionally, the presence of multiple, clearly defined peaks (modes) strongly suggests that the dataset may be heterogeneous, potentially consisting of observations drawn from two or more distinct underlying populations, necessitating further statistical scrutiny.

The frequency polygon's true strength emerges when performing **comparative analysis** across multiple datasets. In contrast to superimposing multiple histograms, which often result in confusing overlap and data occlusion, frequency polygons allow for several distinct distributions to be plotted cleanly on a single set of axes using differentiating colors or line types. This elegant overlay technique facilitates an immediate and intuitive visual comparison of **central tendencies** and the overall variability between the groups under examination.

As a fundamental best practice in data communication, analysts must always align the chosen bin size with the intended audience and the specific purpose of the visualization. While the `geom_freqpoly()` function provides a default of 30 bins, strategic adjustment of this parameter is paramount for maximizing clarity. Utilizing too many bins can unnecessarily clutter and complicate the graphic, whereas employing too few bins risks oversimplifying or masking important underlying distributional features.

Further Learning and Resources

For users committed to advancing their proficiency in data visualization within the R ecosystem, especially those utilizing the robust ggplot2 package and its Grammar of Graphics framework, several authoritative resources are essential reading. These resources delve into advanced topics, including sophisticated aesthetic customization, techniques for managing complex multivariate data structures, and strategies for producing graphics suitable for academic publication.

Continued detailed exploration of **ggplot2** functionality will uncover powerful methods for layering various complex geometries, integrating automated statistical summaries directly onto plots, and mastering specialized coordinate transformations. Developing mastery over these advanced elements is crucial for effective statistical communication and for consistently generating highly sophisticated and informative visual reports.

We strongly recommend that readers prioritize consulting the official documentation for ggplot2. This documentation provides comprehensive details regarding all available parameters for `geom_freqpoly()`, enabling precise control over visual attributes such as **line thickness** (using the `size` argument), **line color** (`color`), and **plot transparency** (`alpha`). Utilizing these parameters ensures the creation of truly tailored and high-impact visualizations.