

How do I create a Histogram of Residuals in R?

Authored by
stats writer

December 13, 2025

RECOMMENDED CITATION

stats writer (2025). *How do I create a Histogram of Residuals in R?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=107362>

In the realm of statistical modeling, particularly when utilizing multiple linear regression (MLR), one of the fundamental assumptions is that the model's residuals are independent and normally distributed. This assumption is critical because it ensures that the statistical inferences derived from the model (such as p-values and confidence intervals) are reliable and accurate. Failure to meet the normality assumption can lead to misleading conclusions about the relationship between the predictors and the response variable.

Visual diagnostics provide an indispensable method for evaluating these assumptions. While several quantitative tests exist, a visual assessment often offers immediate insight into the distribution's shape. Specifically, creating a **histogram of the residuals** allows analysts to visually confirm whether the distribution of errors approximates the symmetrical, 'bell-shape' characteristic of the Normal distribution.

This comprehensive tutorial guides you through the process of generating, fitting, and visually assessing a regression model in R. We will focus specifically on using the popular visualization package, ggplot2, to construct a high-quality histogram of the residuals derived from a linear model. This technique is essential for validating the underlying assumptions of your linear model before drawing firm conclusions.

Step 1: Preparing the Statistical Environment and Generating Data

Before fitting any model, a crucial first step is to establish a working dataset within the R environment. For the purpose of demonstration, we will generate synthetic data using built-in R functions. This ensures that the example is fully reproducible, a best practice often achieved by setting a specific seed value using `set.seed()`. The consistency provided by this step is vital when sharing or validating statistical code.

We will create a dataset comprising 100 observations and three variables: two potential predictor variables, **x1** and **x2**, and one response variable, **y**. All variables are generated using the `rnorm()` function, drawing random samples from a specified Normal distribution. This design helps ensure that the generated data structure aligns with the underlying assumptions we wish to test later. For instance, **x1** is sampled from a Normal distribution with a mean of 2 and a standard deviation of 1, establishing its statistical properties.

The code block below demonstrates the generation of this artificial data, followed by a quick inspection using the `head()` function to verify the structure and initial values of the created `data.frame`. Note that generating data where both predictors and the response variable are normally distributed often results in a final model whose residuals are also normally distributed, making it an excellent case study for successful assumption verification.

Ensure the example is reproducible by setting the seed

set.seed(0)

```
# Create synthetic data for predictors and response
x1 <- rnorm(n=100, 2, 1)
x2 <- rnorm(100, 4, 3)
y <- rnorm(100, 2, 3)
data <- data.frame(x1, x2, y)

# View the first six rows of the constructed data frame
head(data)
```

```
x1 x2 y
1 3.262954 6.3455776 -1.1371530
2 1.673767 1.6696701 -0.6886338
3 3.329799 2.1520303 5.8081615
4 3.272429 4.1397409 3.7815228
5 2.414641 0.6088427 4.3269030
6 0.460050 5.7301563 6.6721111
```

Step 2: Fitting the Multiple Linear Regression Model

With the data prepared and structured in the appropriate `data.frame`, the next logical step is to fit the multiple linear regression model. In R, this is effortlessly achieved using the base function `lm()`, which stands for "linear model." This function is the workhorse for standard regression analysis and is designed to find the best linear relationship that minimizes the sum of squared errors between observed and predicted values.

The `lm()` function requires a specific formula argument (`y ~ x1 + x2`), which specifies that the response variable, **y**, is modeled as a linear combination of the predictor variables, **x1** and **x2**. Additionally, the `data` argument (`data=data`) explicitly links the formula terms to the variables within our previously created dataset. The result of running this function is an S3 object containing all the model parameters, which includes the estimated coefficients, fitted values, and, most critically for diagnostic purposes, the model's residuals.

It is crucial to store this resulting model object in a variable (here, named `model1`) so that we can easily access and extract the necessary components, such as the residuals, for diagnostic checks in the subsequent visualization steps. This modeling step forms the essential foundation for assessing the underlying assumptions of the linear relationship.

Fit the multiple linear regression model

```
model <- lm(y ~ x1 + x2, data=data)
```

Step 3: Extracting Residuals and Creating the Initial Histogram

Once the linear model is successfully fitted, the estimation errors, or residuals--representing the vertical distance between the actual data points and the regression line--can be extracted directly from the stored `model` object using the `model$residuals` command. These residuals are the raw data points whose distribution we must plot to verify the normality assumption.

For high-quality and customizable visualization in R, we utilize the powerful `ggplot2` package. If you haven't already, you must load this library using `library(ggplot2)`. The plotting sequence involves initializing the plot, linking the residuals to the x-axis aesthetic, and then applying the `geom_histogram()` geometry layer to automatically calculate and display the data frequency across various bins.

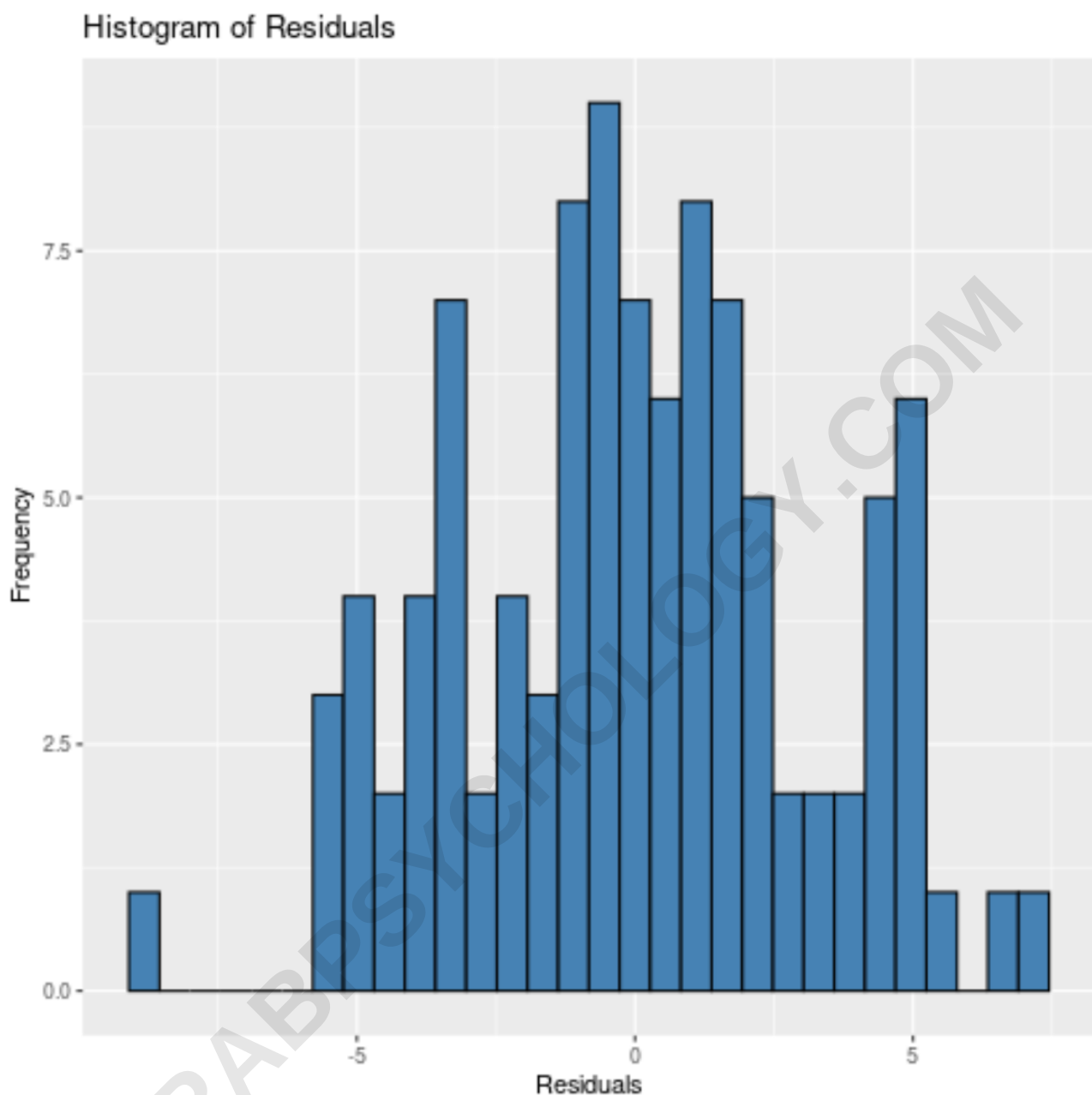
In the code below, we initiate the plot using `ggplot()`, define the aesthetic mapping (`aes(x = model$residuals)`), and then customize the appearance using `geom_histogram()`. We select a clear fill color ('steelblue') and a crisp border color ('black') to enhance readability. Finally, meaningful labels are applied using `labs()` to clearly communicate the purpose of the plot: assessing the distribution of the model's error terms.

Load the ggplot2 visualization package

library(ggplot2)

Create the initial histogram of residuals using default settings

```
ggplot(data = data, aes(x = model$residuals)) +  
geom_histogram(fill = 'steelblue', color = 'black') +  
labs(title = 'Histogram of Residuals', x = 'Residuals', y = 'Frequency')
```



Step 4: Customizing the Histogram with Specific Bin Settings

A critical factor in generating an effective histogram is the strategic selection of the number of bins. The bin count determines how the entire range of residual values is partitioned, and this choice profoundly affects the visual perception of the underlying distribution's shape. Using an inadequate number of bins can mask important features like skewness or multimodality, while an excessive number might lead to a noisy, overly detailed plot that is difficult to interpret as a smooth distribution.

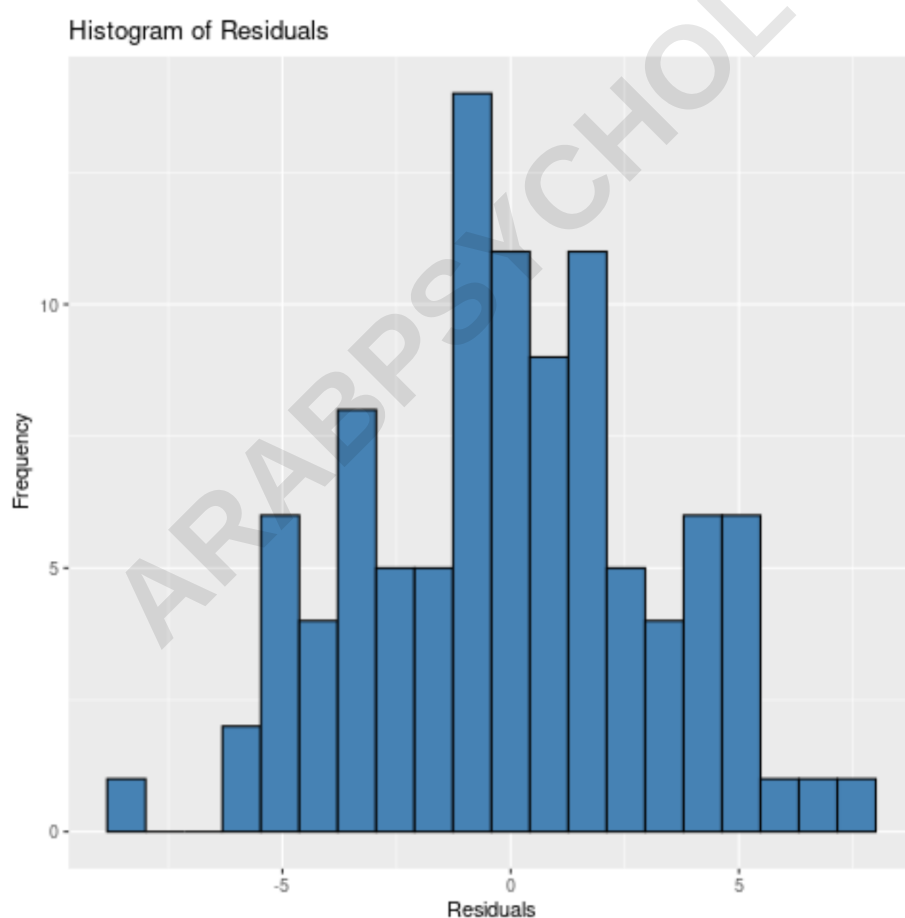
The `ggplot2` package employs a sensible default binning algorithm, but best analytical practice

often requires experimentation. We can override the default and explicitly control the number of bins by using the `bins` argument within the `geom_histogram()` layer. Generally, fewer bins result in wider bars and a more aggregated, smoother view, whereas more bins offer granular detail.

To illustrate the impact of bin selection, we will regenerate the histogram using two distinct settings: a finer resolution with **20 bins**, and a coarser resolution with **10 bins**. Comparing these plots provides valuable insight into the stability of the visual normality assessment, ensuring that the appearance of normality is not merely an artifact of the default bin selection.

First, let's visualize the distribution using a moderately fine resolution of **20 bins**. This choice tends to reveal more subtle patterns in the data density:

```
# Create histogram of residuals with 20 bins for finer detail
ggplot(data = data, aes(x = model$residuals)) +
  geom_histogram(bins = 20, fill = 'steelblue', color = 'black') +
  labs(title = 'Histogram of Residuals (20 Bins)', x = 'Residuals', y = 'Frequency')
```



Next, we reduce the number of bins significantly to **10**. This creates wider bars, providing a high-

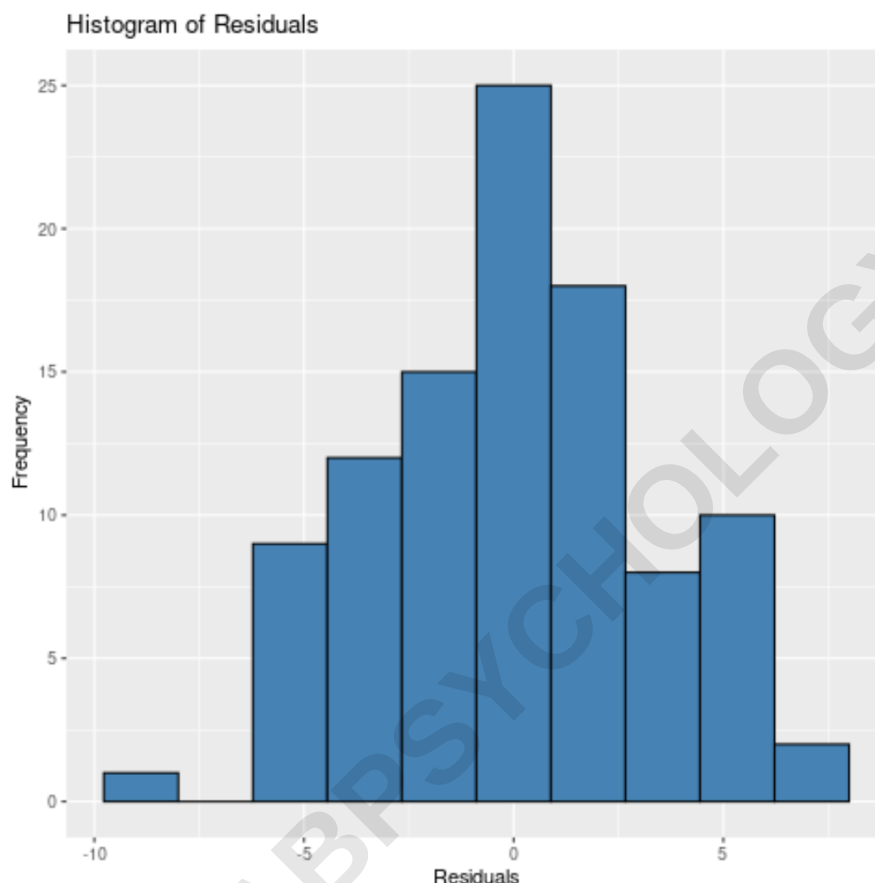
level summary of the residual distribution and emphasizing the central tendency:

Create histogram of residuals with 10 bins for a broader overview

ggplot(data = data, aes(x = model\$residuals)) +

geom_histogram(bins = 10, fill = 'steelblue', color = 'black') +

labs(title = 'Histogram of Residuals (10 Bins)', x = 'Residuals', y = 'Frequency')



Step 5: Interpreting the Visual Normality Assessment

Upon reviewing the generated histograms, regardless of whether 10, 20, or the default number of bins were used, a highly consistent pattern emerges. The distribution of the residuals appears reasonably symmetrical, centered near zero, and exhibits the characteristic single peak and tapering tails associated with a bell-shaped curve. This visual evidence strongly suggests that the model's error terms are approximately normally distributed.

It is crucial for analysts to understand that in empirical statistics, perfect adherence to the Normal distribution is an idealization rarely achieved. The objective of this visual assessment is to determine if the distribution is "close enough." Key diagnostic features to scrutinize include

checking for severe **skewness** (asymmetry), multiple distinct peaks (bimodality), or excessive kurtosis (tails that are too heavy or too light). In this synthetic example, the residuals demonstrate sufficient adherence to the characteristics required for sound linear modeling assumptions.

Had the histogram revealed significant deviations--such as a heavily right- or left-skewed distribution, or a flat, uniform distribution--it would indicate a violation of the normality assumption. Such a finding would necessitate subsequent analytical steps, possibly including transforming the response variable, re-evaluating model specification, or transitioning to non-parametric or robust regression techniques that are less sensitive to distributional assumptions.

Step 6: Considering Formal Statistical Tests for Normality

While visual tools like the histogram are powerful and intuitive, many statistical workflows require supplementing graphical assessment with formal statistical tests of normality. These tests provide a quantitative measure of the deviation of the sample distribution from a theoretical Gaussian distribution, yielding a p-value that aids in the hypothesis testing process.

Standard tests frequently employed for this purpose include the Shapiro-Wilk test, the Kolmogorov-Smirnov test (often adjusted, like the Lilliefors test), and the Jarque-Bera test. For instance, the Shapiro-Wilk test is widely favored due to its robust power for moderate sample sizes (typically $N < 5000$). If the p-value resulting from these tests is greater than the chosen significance level (e.g., $\alpha = 0.05$), we conclude that there is insufficient evidence to reject the null hypothesis, supporting the conclusion that the data are consistent with a normal distribution.

A crucial caveat regarding these formal tests, however, lies in their sensitivity to large sample sizes. When dealing with extensive datasets, these tests often possess enough power to detect even minor, substantively irrelevant deviations from perfect normality. This can lead to a statistical rejection of the normality assumption, even when the visual evidence (histogram, Q-Q plot) suggests the distribution is acceptable for the purposes of the regression model. Consequently, the consensus among statisticians is often to prioritize the **visual diagnostic assessment** over the rigid outcome of a formal test, especially in cases where the sample size is large. The visual assessment provides context that a single p-value cannot capture.