

How to Easily Plot a Logistic Regression Curve in R

Authored by
stats writer

December 6, 2025

RECOMMENDED CITATION

stats writer (2025). *How to Easily Plot a Logistic Regression Curve in R*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=106410>

Data visualization is an essential component of statistical analysis, allowing analysts to convey complex mathematical relationships in an intuitive graphical format. When working with binary outcomes, the logistic regression model is the standard tool for predicting the probability of an event occurrence. While fitting the model provides coefficients, plotting the resulting sigmoid curve is crucial for visually understanding how a predictor variable influences the predicted probability. This comprehensive guide, tailored for users of the R statistical environment, details the methodologies for generating clean, informative logistic regression plots using both native Base R functions and the powerful Tidyverse package, ggplot2.

The ability to plot the fitted curve offers significant advantages over simply reading output tables. It immediately illustrates the non-linear relationship inherent in logistic models--a relationship characterized by the S-shaped curve that approaches 0 and 1 asymptotically. This visualization helps confirm the direction and strength of the relationship and allows for easy identification of the point of inflection, where the probability changes most rapidly. Understanding how to manually construct this plot, rather than relying solely on automated functions, provides deeper insight into the calculation of predicted probabilities.

Understanding the Theoretical Basis of the Sigmoid Curve

A logistic regression model utilizes the logit function to link the linear combination of predictor variables to the probability of the outcome. The core mathematical foundation is derived from the generalized linear model (GLM) framework, specifically using the binomial family and the logit link function. The predicted outcome is not a binary 0 or 1, but rather a probability value, $P(Y=1|X)$, constrained between 0 and 1. The resulting curve, often termed the S-curve, reflects this transformation and is fundamental to interpreting the model.

To accurately plot the curve, one must first generate a sequence of predictor values (the X-axis range) and then use the fitted model to calculate the corresponding predicted probabilities (the Y-axis values). This sequence must be sufficiently dense to ensure the resulting line is smooth and accurately represents the continuous nature of the theoretical probability function. In practical R implementation, this means creating a new data frame covering the full range of the independent variable, and then using the `predict` function with the argument `type="response"` to obtain the probability scale values.

Implementing the Logistic Model: Data Preparation in R

For demonstration purposes, we will use the well-known built-in `mtcars` dataset in R. We aim to model the probability of a car having a V/S engine (`vs`, where 1 indicates a V-engine) based on its horsepower (`hp`). This requires fitting a Generalized Linear Model using the `glm()` function and specifying `family=binomial`. This crucial step establishes the necessary statistical relationship

before we proceed to visualization. The `mtcars` dataset offers a practical scenario for illustrating how engine specifications relate probabilistically to car design features.

The initial preparation involves defining the model and creating the necessary scaffolding for plotting. This scaffolding, a new data frame, must span the entire observed range of the predictor variable, `hp`. If this step is overlooked, the resulting curve might only cover a sparse set of points, leading to a jagged or incomplete visualization. We typically use functions like `seq()` to generate hundreds of evenly spaced points between the minimum and maximum values of the predictor, ensuring the resulting sigmoid curve is perfectly smooth, reflecting the theoretical relationship derived from the model coefficients.

Step-by-Step Guide: Plotting the Curve using Base R Graphics

The Base R graphics system offers a robust, built-in method for plotting statistical models without requiring external packages. While often less aesthetically customizable than `ggplot2`, it is highly efficient and readily available in every R session. The process involves two primary stages: first, fitting the model and generating the predicted values across the range of the predictor; and second, overlaying the fitted line onto a scatter plot of the original data points. This manual approach provides explicit control over the steps involved in prediction and plotting.

The following code block executes the modeling and preparation steps. We fit the logistic regression model, define a new data frame named `newdata` containing 500 points across the range of `hp`, and then populate `newdata` with the calculated predicted probabilities using the `predict()` function. The use of `type="response"` ensures the predictions are returned as probabilities (0 to 1), which is essential for plotting the curve on the correct vertical axis.

#fit logistic regression model

```
model <- glm(vs ~ hp, data=mtcars, family=binomial)
```

```
#define new data frame that contains predictor variable
```

```
newdata <- data.frame(hp=seq(min(mtcars$hp), max(mtcars$hp),len=500))
```

```
#use fitted model to predict values of vs
```

```
newdata$vs = predict(model, newdata, type="response")
```

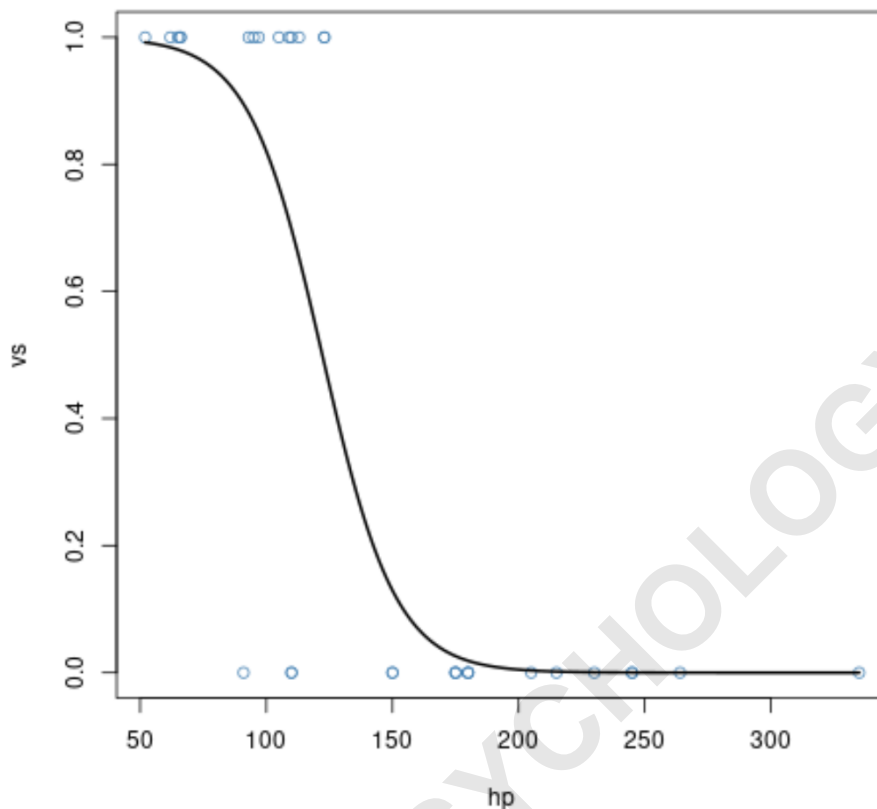
```
#plot logistic regression curve
```

```
plot(vs ~ hp, data=mtcars, col="steelblue")
```

```
lines(vs ~ hp, newdata, lwd=2)
```

The visualization is completed by first calling `plot()`, which creates a scatter plot of the original binary data (`vs` vs. `hp`). Since `vs` is a binary variable, the points will cluster along the y-axis values

of 0 and 1. We then use the `lines()` function, passing the prepared `newdata`, to overlay the smooth, continuous predicted curve. This combination clearly shows the relationship between the predictor variable and the modeled probabilities, providing a complete visual representation of the Base R output. The `lwd=2` argument is used to increase the line thickness for improved visibility.



Interpreting the Base R Output and Model Diagnostics

Upon examining the generated Base R plot, we can derive crucial interpretations regarding the model's performance and the directional relationship between the variables. The x-axis represents the continuum of the predictor variable, **hp** (horsepower), ranging from the lowest to the highest observed values in the `mtcars` dataset. The y-axis, conversely, displays the predicted probability of the response variable, **vs**, taking the value of 1 (V-engine). The scattered points at $Y=0$ and $Y=1$ represent the actual observed outcomes.

In this specific scenario, the curve exhibits a clear inverse relationship. We observe that higher values of the predictor variable **hp** are significantly associated with lower probabilities of the response variable **vs** being equal to 1. For vehicles with low horsepower, the predicted probability of having a V-engine approaches 1, whereas for vehicles with very high horsepower, the probability drops sharply toward 0. The inflection point, where the slope of the sigmoid curve is steepest, indicates the horsepower value at which small changes have the maximum impact on the

likelihood of the engine type.

While Base R is effective for simple plotting, the scatter plot of the raw data points often obscures the density of observations, especially when many points overlap at $Y=0$ and $Y=1$. For more detailed diagnostic plots in Base R, techniques like jittering (slightly displacing the points horizontally or vertically) or binning the data might be applied to better visualize the raw data distribution relative to the fitted curve, aiding in the assessment of model fit across different segments of the predictor variable.

Leveraging the Tidyverse: Plotting with ggplot2

For researchers and analysts prioritizing highly customizable and aesthetically refined visualizations, the `ggplot2` package, part of the Tidyverse ecosystem, is the preferred choice in R. `ggplot2` is built on the grammar of graphics, allowing users to layer components (data, aesthetics, geometries, and statistics) systematically. This approach often simplifies the plotting of statistical models, as `ggplot2` includes powerful statistical geometries that can automatically calculate and draw fitted curves, eliminating the need for manual data prediction.

Unlike Base R, which required manual computation of 500 data points, `ggplot2` utilizes the `stat_smooth()` function to handle the model fitting and curve generation internally. By setting `method="glm"` and passing the required arguments via `method.args = list(family=binomial)`, `ggplot2` automatically fits the logistic regression and renders the smooth curve, overlaying it on the raw data points (`geom_point`). This significantly reduces the code complexity for generating a basic plot while adhering to the principles of consistent, high-quality visualization design.

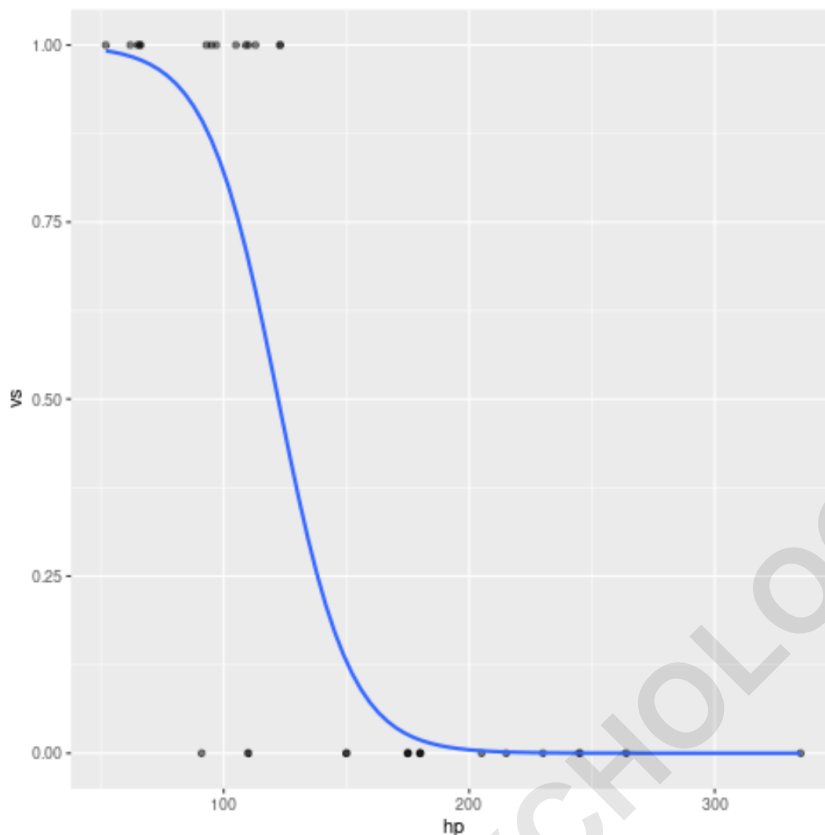
The following code demonstrates the streamlined process using `ggplot2`. Note the absence of the manual `seq()` and `predict()` steps. We start by loading the library, defining the aesthetic mapping (`aes`), adding points with some transparency (`alpha=.5`), and finally adding the statistical layer `stat_smooth`, instructing it to fit a binomial GLM. We suppress the standard error band (`se=FALSE`) for clarity in this initial example.

library(ggplot2)

```
#plot logistic regression curve
ggplot(mtcars, aes(x=hp, y=vs)) +
  geom_point(alpha=.5) +
  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial))
```

As evidenced by the resulting image, this method produces the exact same fitted sigmoid curve as produced in the previous Base R example, confirming the mathematical consistency across

visualization methods while offering the superior flexibility and layering capabilities of the `ggplot2` environment for future customizations.



Customizing and Enhancing the ggplot2 Visualization

The core strength of `ggplot2` lies in its comprehensive capacity for aesthetic modification. After generating the initial plot, analysts can easily modify visual elements such as color, line type (LTY), and overall thematic appearance without altering the underlying statistical calculation. These customizations are vital for creating plots that adhere to journal submission guidelines or corporate branding standards. Parameters like `col` (color) and `lty` (line type) can be specified directly within the `stat_smooth()` layer, allowing for immediate visual feedback.

For instance, if the curve needs to be highly distinct against a complex background or if multiple curves are being plotted simultaneously (e.g., for different subgroups), changing the default blue solid line is often necessary. We can transform the curve into a bright red dashed line by simply adding `col="red"` and `lty=2` to the `stat_smooth` call. This demonstrates how easily one can control the visual properties of the fitted model layer in `ggplot2`.

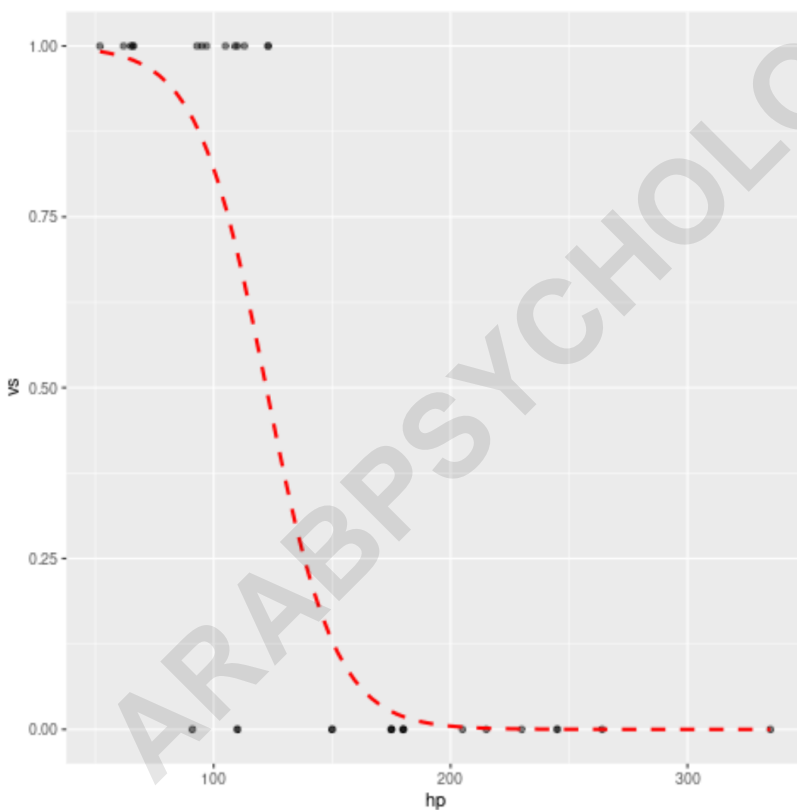
Below is the code required to apply these aesthetic changes, ensuring that the visual representation of the fitted logistic regression model is clear and visually compelling. The use of

`lty=2` specifically renders the line as a dashed pattern, which enhances differentiation.

library(ggplot2)

```
#plot logistic regression curve
ggplot(mtcars, aes(x=hp, y=vs)) +
  geom_point(alpha=.5) +
  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial),
  col="red", lty=2)
```

This resulting plot provides an immediate, customized visual summary of the model's predictions, highlighting the negative correlation between horsepower and the probability of a car having a V-engine.



Advanced Considerations for Logistic Regression Plots

Moving beyond basic visualizations, the complexity of real-world logistic models often demands the inclusion of additional visual elements to provide a complete picture of the fit and uncertainty. For models incorporating multiple predictor variables, analysts must be cautious; the standard 2D plot only shows the relationship between the outcome and one predictor. To maintain interpretability,

the relationship is typically plotted while holding all other predictors constant at a meaningful reference point, such as their mean or median values. This conditional plotting requires using the `predict()` function on a structured grid of new data, similar to the Base R setup.

A critical advanced consideration is the representation of model uncertainty, usually through the standard error (SE) band. The SE band, often set to represent a 95% confidence interval around the predicted probabilities, is crucial because it provides a visual assessment of how certain the model is about its predictions across the range of the predictor variable. It is common to observe widening confidence intervals at the extremes of the predictor range where data density is lower, visually reinforcing the concept of decreased predictive certainty due to extrapolation.

Furthermore, when the raw data points are sparse or highly clustered (as is common with binary outcomes), advanced visualization techniques can improve diagnostic clarity. These techniques might include adding a rug plot (marginal tick marks indicating data density along the axes) or employing binning methods where the plot is divided into segments, and the empirical probability of the outcome (i.e., the proportion of 1s) is calculated and plotted within each bin to contrast with the smooth fitted curve.

Summary and Best Practices

Effectively plotting a logistic regression curve is a fundamental skill in statistical reporting using R. Whether opting for the manual precision of Base R, which requires careful construction of the prediction data frame, or the layered flexibility of `ggplot2`'s `stat_smooth()` function, the objective remains the same: to accurately visualize the S-shaped relationship between the predictor and the predicted probability.

Key best practices include ensuring the prediction sequence is dense enough (e.g., 500 points) to generate a smooth curve, always plotting the curve on the response scale (probabilities between 0 and 1), and thoughtfully considering the inclusion of the standard error band to convey model uncertainty. By mastering these techniques, analysts can move beyond dry tables of coefficients to produce compelling and scientifically rigorous visual summaries of their binary outcome models. This methodology can be generalized to any binomial GLM, making it a versatile technique for visualizing predictive relationships across various domains.