Learning Logistic Regression: A Practical Guide to Plotting Curves in R

Authored by

Mohammed looti

November 5, 2025

In the expansive realm of statistical modeling, the logistic regression model stands as an indispensable tool for analyzing and predicting binary outcomes. Unlike its linear counterpart, which is constrained to modeling continuous dependent variables, logistic regression calculates the probability of a specific event occurring, inherently constraining the output to fall within the valid range of 0 and 1.

The ability to effectively visualize these models is paramount to understanding their predictive power. When working with these statistical frameworks in R, analysts must frequently plot the characteristic S-shaped curve. This visualization provides crucial, immediate insight into the non-linear relationship: how incremental changes in the predictor variable influence the likelihood of the designated outcome. Without this visual aid, interpreting the complex interplay between predictors and probability becomes significantly more challenging.

This comprehensive technical guide is designed to walk expert users through the precise methods required to accurately plot a fitted logistic regression curve. We will demonstrate two primary methodologies within R: first, using the traditional and highly robust base R graphics system, and second, employing the highly flexible, modern, and aesthetically superior ggplot2 package. Mastering both techniques ensures versatility regardless of the specific visualization task or environment.

The Mathematical Foundation of the Sigmoid Curve

The unique shape of the logistic regression curve stems directly from the underlying mathematical structure designed to handle binary predictions. The model utilizes the specialized logit function, which acts as a link function, transforming the linear combination of predictor variables into a value that represents the log-odds of the outcome. Subsequently, this log-odds value is mapped back to the probability scale (0 to 1) via the inverse logit function, known as the sigmoid function.

This resulting sigmoid curve is vital for interpretation because it mathematically enforces the necessary boundaries for probability. The curve visually represents the changing odds of the outcome (Y=1) as a continuous function of the predictor (X). The S-shape ensures that, regardless of how extreme the values of the predictor variable become, the predicted probability will never fall below 0 or exceed 1, approaching these limits asymptotically.

Effective visualization helps the analyst pinpoint critical segments of the curve. For instance, the steep central portion indicates where a small change in the predictor variable results in the greatest shift in probability. Conversely, the flattened tails (the plateau regions near 0 and 1) reveal areas where further changes in the predictor yield only minimal, statistically negligible change in the predicted probability. Understanding these inflection points is key to model validation and practical application.

Data Preparation and Model Specification in R

To provide a clear, reproducible demonstration of these plotting techniques, we will leverage the renowned mtcars dataset, a classical benchmark frequently utilized in R documentation. Our objective is to model the likelihood of a car possessing a V-shaped engine (represented by the variable `vs`, where 1 denotes a V-shaped engine and 0 indicates a straight engine) based exclusively on its gross horsepower (`hp`).

The initial step involves fitting the statistical model itself. In R, this is achieved using the Generalized Linear Model (`glm`) function. It is crucial to specify the `family=binomial` argument within the `glm` call; this directive instructs R to use the logit link function appropriate for handling binary response variables, establishing the foundational parameters before any visualization can occur.

Crucially, before plotting the curve, we must generate a sequence of predictor values that span the entire range of the observed data for the variable `hp`. If we simply connect the dots from the raw data, the resulting plot would appear jagged and inaccurate. By generating a new, high-resolution data frame specifically for prediction--typically containing 500 or more evenly spaced points--we ensure that the plotted logistic curve is smooth, continuous, and accurately reflects the model's prediction across all possible input values. This new data structure is fundamental for both base R and advanced plotting methods.

Visualizing the Curve Using Base R Graphics

The built-in base R plotting system offers a powerful, albeit more manual, method for generating the logistic curve. This approach necessitates two distinct phases: first, manually generating the predicted probabilities for the smooth sequence of predictor values, and second, sequentially layering the raw data and the calculated curve onto the plot canvas.

The following R script demonstrates the necessary steps. We fit the model and then use the `predict` function. It is essential to specify `type="response"` within the `predict` call; this ensures that the output is returned on the probability scale (0 to 1), which is necessary for visualization, rather than the default log-odds scale, which is not intuitive for direct interpretation.

The raw data points are then plotted, followed by the smooth curve. This two-step process is characteristic of base R's layering technique, offering granular control over every element.

 # Fit the logistic regression model using the binomial family
model <- glm(vs ~ hp, data=mtcars, family=binomial)

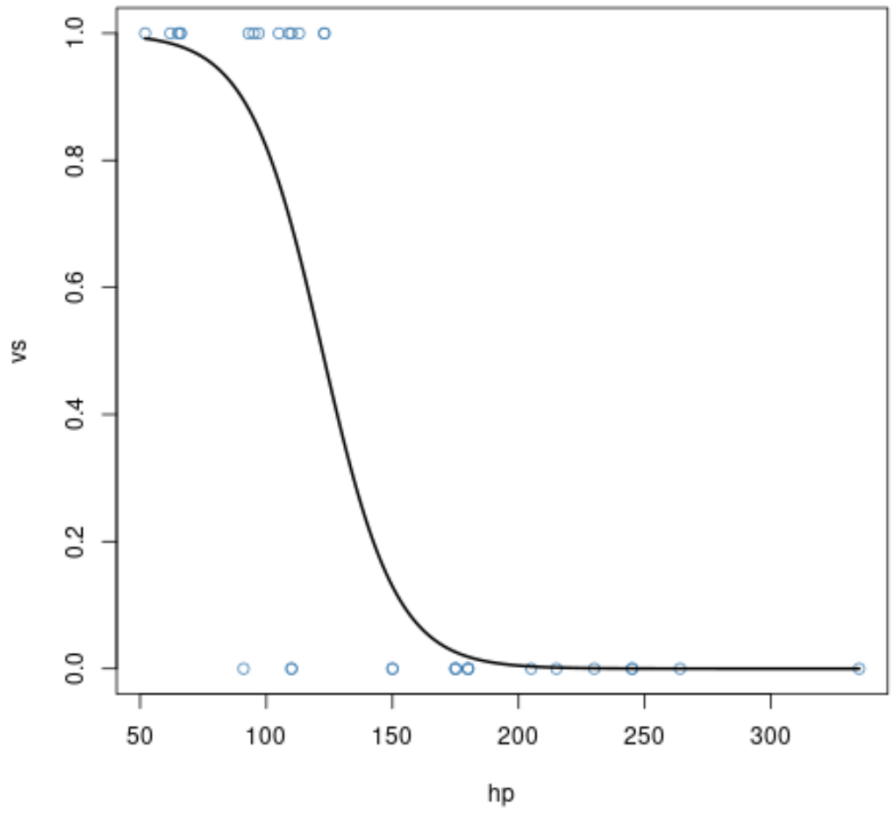
Define a new data frame containing 500 evenly spaced values of the predictor variable (hp)

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

# Use the fitted model to predict probability values for vs across the new data range
newdata$vs = predict(model, newdata, type="response")

# Plot the raw binary data points
plot(vs ~ hp, data=mtcars, col="steelblue", pch=16, main="Logistic Regression Curve (Base R)",
xlab="Horsepower (hp)", ylab="Predicted Probability (vs=1)")

# Overlay the smooth logistic regression curve using the predicted probabilities
lines(vs ~ hp, newdata, lwd=3, col="red")
```



The execution of the `plot()` function first renders the raw discrete data points, which are confined to the y-axis values of 0 and 1, representing the presence or absence of a V-shaped engine. Subsequently, the `lines()` function draws the smooth, continuous logistic curve, based on the calculated predicted probabilities stored in the `newdata` data frame. This curve serves as the graphical representation of the model's estimate--the estimated probability of `vs` being 1 for any corresponding value of `hp`.

This visualization confirms the modeled relationship: the x-axis displays the range of **horsepower**

(hp), while the y-axis shows the predicted likelihood of the response variable, **vs**, being 1. The visible negative slope confirms that higher values of the predictor variable **hp** are strongly associated with lower probabilities of the response variable **vs** being equal to 1, indicating that powerful cars are less likely to have a V-shaped engine in this dataset. This provides an immediate, intuitive understanding of the model's overall performance.

Advanced Visualization with ggplot2

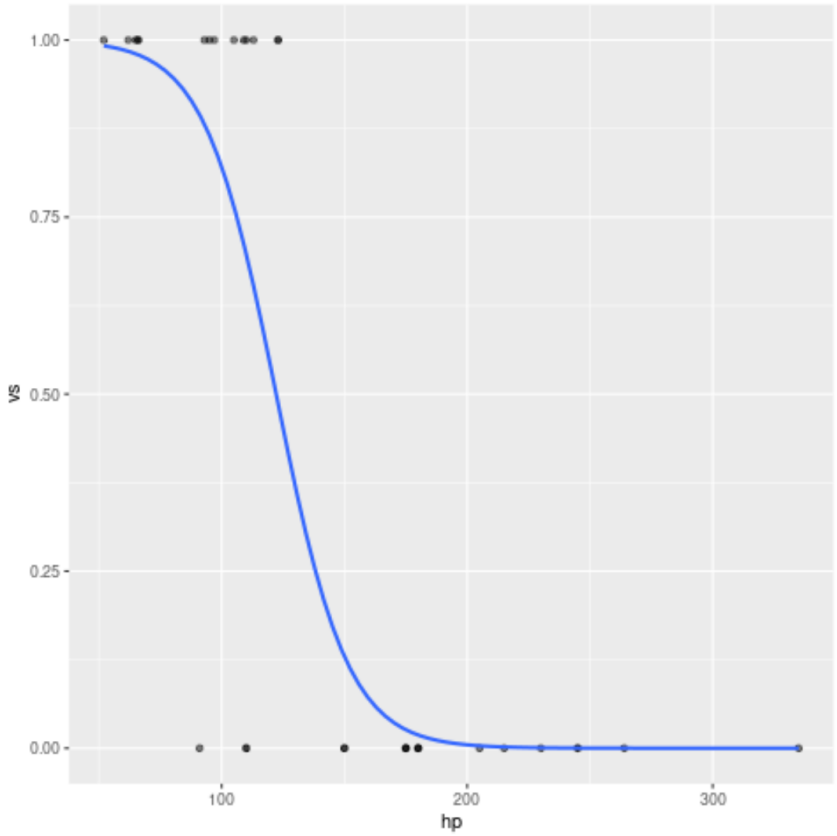
While base R remains functional, the ggplot2 library, a core component of the Tidyverse ecosystem, is widely regarded as the modern standard for sophisticated data visualization in R. ggplot2 simplifies the plotting of statistical models substantially by implementing the "Grammar of Graphics," which allows users to build plots by layering components.

A major advantage of using ggplot2 for this task is the automation provided by the `stat_smooth()` layer. Unlike base R, we do not need to manually create the `newdata` frame, nor do we need to calculate the predictions explicitly. The `stat_smooth()` function automatically handles the heavy lifting: it fits the specified model type (in this case, GLM with the binomial family) and generates the necessary smooth prediction curve based on the model parameters.

This approach is significantly more streamlined and reduces the potential for manual data manipulation errors. We initiate the plot using `geom_point()` to display the raw data, and then we utilize `stat_smooth()`, passing the method arguments needed to ensure it fits the correct logistic regression model.

library(ggplot2)

```
# Plot logistic regression curve using ggplot2
ggplot(mtcars, aes(x=hp, y=vs)) +
geom_point(alpha=.5, color="darkblue") +
stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial), color="red", size=1.5)
+
labs(title="Logistic Regression Model Visualization (ggplot2)",
x="Gross Horsepower (hp)",
y="Probability of V-Shaped Engine (vs=1)") +
theme_minimal()
```



It is important to emphasize that the curve generated through the `ggplot2` framework is mathematically identical to the one derived from the base R method, confirming the validity and fidelity of both visualization techniques. The fundamental difference lies entirely in the syntax used for implementation, the efficiency of execution, and the superior default aesthetics provided by the ggplot2 ecosystem, which often leads to publication-quality graphics with minimal effort.

Customizing the ggplot2 Curve Appearance

One of the most compelling reasons to adopt ggplot2 is the unparalleled ease with which visual aesthetics can be manipulated and refined. Customizing the appearance of the logistic curve allows analysts to enhance clarity, emphasize key model features, or ensure strict adherence to specific organizational or publishing standards.

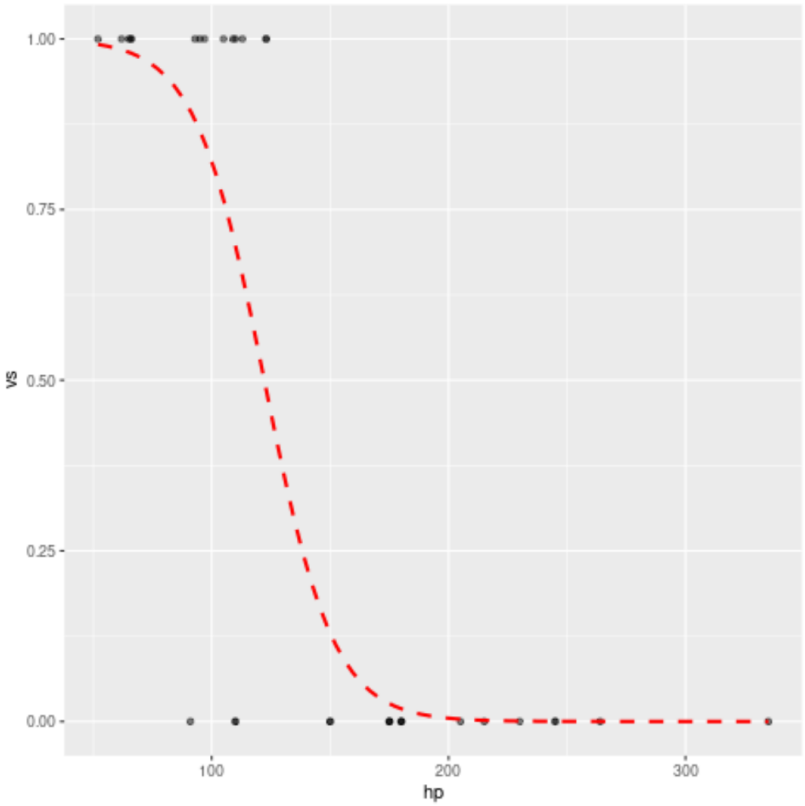
Aesthetic modifications are applied directly within the `stat_smooth()` function or via additional layers. For instance, if the goal is to draw immediate attention to the fitted model, one might opt to change the line color to a high-contrast hue, such as red, and modify the line style to a dashed or dotted pattern. This is achieved by setting the aesthetic parameters `col` (color) and `lty` (linetype) within the function arguments.

This precise, layered control over every visual element--from point opacity to line thickness and

color scheme--makes ggplot2 the preferred instrument for producing high-quality, professional statistical graphics that effectively communicate complex modeling results.

library(ggplot2)

```
# Plot logistic regression curve with aesthetic customizations
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, size=1) +
theme_classic() +
labs(title="Customized Logistic Curve Plot",
x="Horsepower",
y="Probability")
```



Conclusion and Next Steps in Binary Classification Visualization

Plotting the predicted curve of a logistic regression model is not merely an optional step; it is an essential procedure for validating model performance and effectively communicating the results of any binary classification analysis. Whether the analyst favors the detailed, step-by-step control

offered by base R or prefers the streamlined, layered efficiency of the ggplot2 package, the capacity to visualize the non-linear, S-shaped relationship is critical for deep data interpretation.

For those seeking to enhance the robustness and clarity of these plots, several advanced steps are highly recommended. A fundamental improvement involves integrating measures of uncertainty by adding confidence intervals around the predicted probabilities. This can be achieved by setting `se=TRUE` within `stat_smooth` in ggplot2, or by manually calculating standard errors and plotting them as ribbons or error bars in base R. These intervals provide a visual measure of the model's confidence across the range of the predictor variable.

Furthermore, when dealing with more complex models involving multiple predictor variables, the visualization technique must be adapted. To isolate and display the effect of a single predictor variable, such as `hp`, it is necessary to hold all other covariates constant--typically setting them to their mean, median, or a specific reference level. This technique ensures that the resulting curve accurately represents the isolated effect, preventing potential confounding influences from skewing the interpretation of the marginal relationship.

Additional Resources for Statistical Graphics in R

Detailed documentation and theoretical background for the GLM function in R, including considerations for link functions and family specifications.

A comprehensive guide to the underlying philosophy and practical application of the ggplot2 visualization package, covering advanced layers and aesthetic mappings.

In-depth tutorial resources focused on model diagnostics and assessment criteria specifically tailored for logistic regression models.