Learn How to Plot Predicted Values from Regression Models in R

Authored by

Mohammed looti

November 5, 2025

When working with regression models in data analysis, particularly within the **R** statistical environment, it is fundamental to visualize the model's performance. A crucial diagnostic technique involves plotting the predicted values against the actual observed values. This visualization allows analysts to quickly assess the fidelity of the model and identify potential biases or areas where the predictions deviate significantly from reality.

The goal of this comprehensive tutorial is to demonstrate how to effectively generate this type of diagnostic plot using two primary visualization systems available in **R**: the native **Base R** graphics system and the highly versatile **ggplot2** package. We will walk through the necessary steps for data preparation, model fitting, and detailed visualization, ensuring a clear understanding of the resulting graphical output.

Why Visualize Predicted vs. Actual Values?

Visual inspection is often the most intuitive and robust method for evaluating the success of a statistical model. While numerical metrics, such as R-squared or Root Mean Squared Error (RMSE), provide quantitative measures of fit, plotting the actual values against the predicted outcomes offers a visual confirmation of linearity and homoscedasticity. Ideally, if a model perfectly captures the relationship within the data, all data points should fall directly onto a diagonal line where the predicted value equals the actual value.

Deviations from this perfect diagonal line indicate prediction errors, also known as residuals. By visualizing these deviations, analysts can gain insight into whether the model systematically overestimates or underestimates the response variable across different ranges. This is particularly important in fields like finance, engineering, and social sciences, where precise predictive accuracy is paramount for decision-making.

Furthermore, this visualization serves as a powerful tool for communicating model reliability to non-technical stakeholders. A clear scatter plot showing points clustered tightly around the identity line provides undeniable evidence of a strong fit, whereas scattered points signal weak predictive power. We will explore two distinct methodologies for generating this vital plot, starting with the traditional approach using Base **R**.

Data Preparation and Fitting the Regression Model

Before visualization can occur, we must establish a working dataset and fit a statistical model. For this tutorial, we will utilize a small, custom dataset designed to illustrate the process of fitting a multiple linear regression model. This model predicts a response variable (y) based on two predictor variables (x1 and x2).

The following **R** code block outlines the steps necessary to create the sample data frame and then

fit the **multiple linear regression** model using the standard `lm()` function. This foundational model will generate the necessary **predicted values** that we intend to visualize.

```
#create data
df <- data.frame(x1=c(3, 4, 4, 5, 5, 6, 7, 8, 11, 12),
x2=c(6, 6, 7, 7, 8, 9, 11, 13, 14, 14),
y=c(22, 24, 24, 25, 25, 27, 29, 31, 32, 36))

#fit multiple linear regression model
model <- lm(y ~ x1 + x2, data=df)
```

The resulting `model` object contains all the information derived from the regression analysis, including the coefficients, residuals, and, crucially, the capability to generate predicted outcomes for the original observations using the `predict()` function. Once the model is fitted, we are ready to proceed with graphical diagnostics.

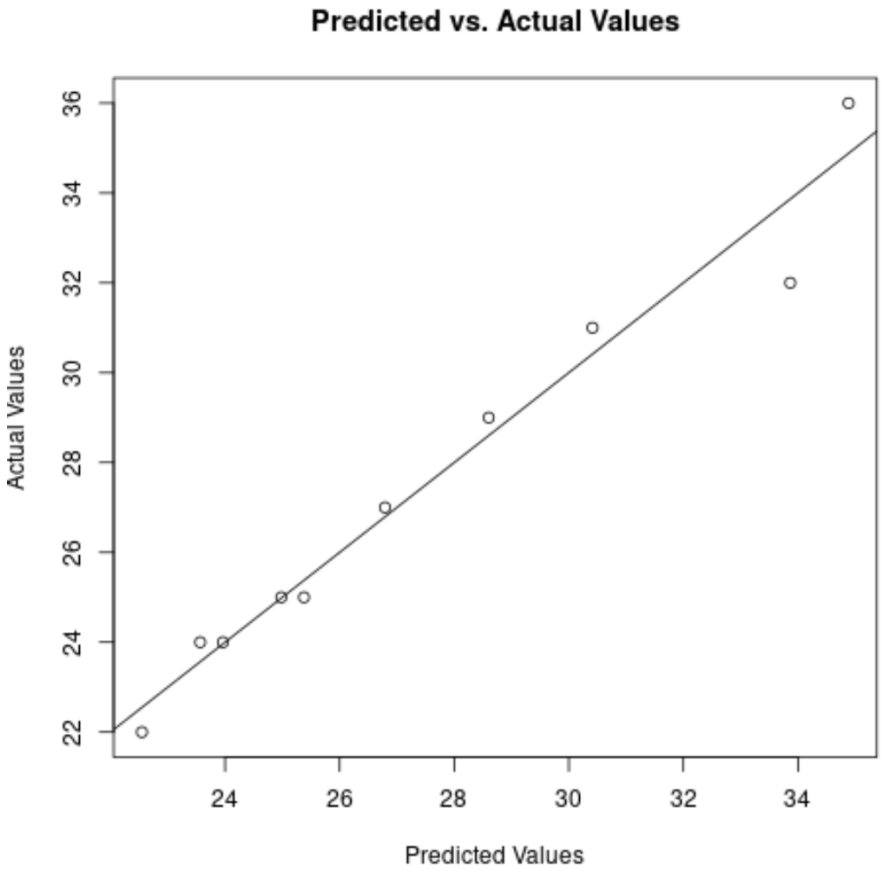
Example 1: Visualizing Predicted vs. Actual Values using Base R Graphics

The **Base R** plotting system offers a straightforward and efficient way to create scatter plots directly from the statistical output. We leverage the `plot()` function, providing the predicted values as the x-axis input and the actual values (y) as the y-axis input. This visual setup immediately establishes the diagnostic framework.

The following code snippet demonstrates how to generate the plot. Notice the inclusion of the `abline()` function, which is essential for drawing the identity line (y = x). We set the intercept (a) to 0 and the slope (b) to 1. This line represents the ideal scenario where the predicted outcome perfectly matches the observed outcome, acting as a visual benchmark for model performance.

```
#plot predicted vs. actual values
plot(x=predict(model), y=df$y,
xlab='Predicted Values',
ylab='Actual Values',
main='Predicted vs. Actual Values')

#add diagonal line for estimated regression line
abline(a=0, b=1)
```



The resulting output is a scatter plot where the horizontal axis represents the model's output (the **predicted values**) and the vertical axis represents the ground truth (the actual data points). The diagonal line serves as the crucial reference point. This visualization, generated quickly using **Base R**, provides immediate feedback on how well the regression model aligns with the underlying data structure.

Interpreting the Base R Plot and Model Fit

The interpretation of the predicted vs. actual plot is relatively straightforward but profoundly insightful. Every data point plotted represents one observation from the dataset. The distance of that point from the diagonal identity line quantifies the residual error for that specific observation. Small vertical distances indicate high accuracy, while large distances suggest poor prediction for that data point.

In the plot above, since the data points lie fairly close to the estimated regression line, we can deduce that the **regression model** exhibits a strong fit to the data. This suggests that the relationship between the predictors (x1, x2) and the response (y) is well-approximated by the linear function defined by the model. If, conversely, the points formed a pattern (e.g., a curve or a cone shape) or were widely dispersed, it would indicate model misspecification or heteroscedasticity.

To gain a numerical perspective on these errors, we can explicitly calculate and display a data frame showing the actual and predicted values side-by-side. This table allows for a point-by-point comparison, quantifying the magnitude of the residuals for each observation.

#create data frame of actual and predicted values
values <- data.frame(actual=df$y, predicted=predict(model))

#view data frame
values

actual predicted
1 22 22.54878
2 24 23.56707
3 24 23.96341
4 25 24.98171
5 25 25.37805
6 27 26.79268
7 29 28.60366
8 31 30.41463
9 32 33.86585
10 36 34.88415

Example 2: Enhancing Visualization with ggplot2

While **Base R** graphics are functional, many R users prefer the flexibility and aesthetic control offered by the ggplot2 package, which is based on the grammar of graphics philosophy. Using ggplot2 allows for easier customization of colors, themes, labels, and the addition of complex geometric layers.

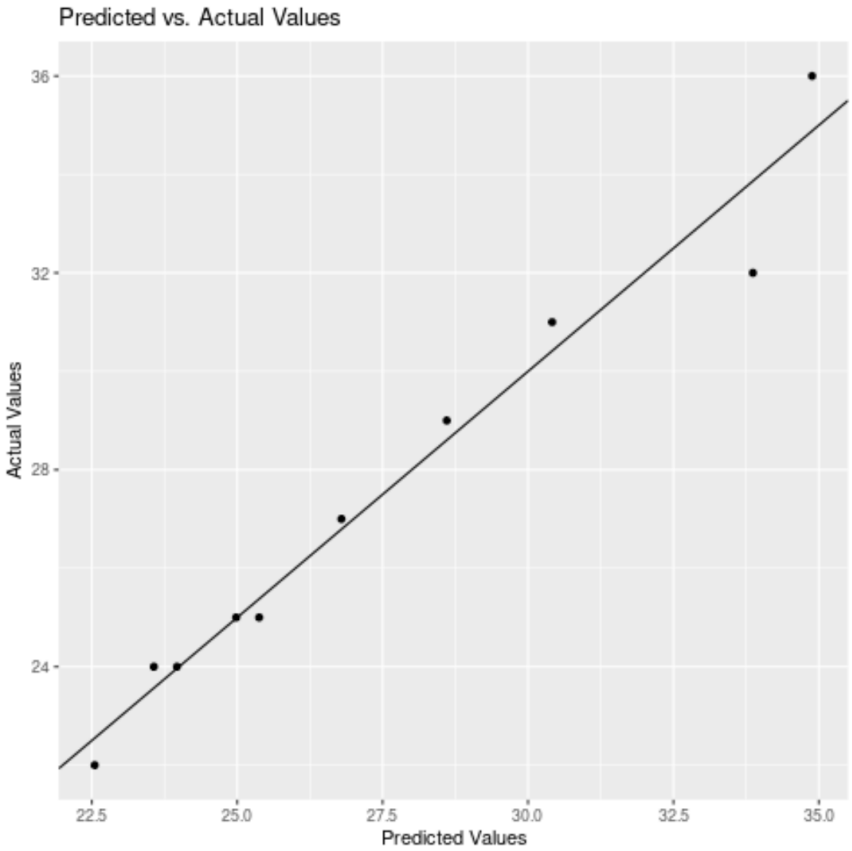
To recreate the predicted vs. actual plot using ggplot2, we first need to load the library. We then map the model's predicted output to the aesthetic property `x` and the actual response variable `y` to the aesthetic property `y` within the main `ggplot()` call. We use `geom_point()` to render the scatter plot points and `geom_abline()` to draw the identity line, specifying the intercept and slope just as we did with Base R's `abline()` function.

library(ggplot2)

#create data
df <- data.frame(x1=c(3, 4, 4, 5, 5, 6, 7, 8, 11, 12),
x2=c(6, 6, 7, 7, 8, 9, 11, 13, 14, 14),
y=c(22, 24, 24, 25, 25, 27, 29, 31, 32, 36))

```
#fit multiple linear regression model
model <- lm(y ~ x1 + x2, data=df)

#plot predicted vs. actual values
ggplot(df, aes(x=predict(model), y=y)) +
geom_point() +
geom_abline(intercept=0, slope=1) +
labs(x='Predicted Values', y='Actual Values', title='Predicted vs. Actual Values')
```



As demonstrated by the output, the visualization generated by ggplot2 conveys the same diagnostic information as the Base R plot--the x-axis displays the predicted values from the model, and the y-axis displays the actual values from the dataset. The primary benefit here is the ability to easily integrate this plot into a broader visualization workflow that often requires consistent aesthetic themes and layering of complex elements, such as confidence bands or coloring points by different categories.

Advanced Visualization Concepts and Diagnostics

While the predicted vs. actual plot is an excellent first step, a complete model diagnostic workflow

often requires further visualizations, especially if the initial plot reveals significant deviations or patterns. Analysts should consider generating plots that focus specifically on the residuals.

One critical supplementary plot is the Residuals vs. Fitted Values plot. This plot helps verify the assumption of homoscedasticity (constant variance of errors). If the residuals are randomly scattered around the horizontal line at zero, the assumption is met. If they form a funnel shape, it suggests heteroscedasticity, which may require model transformation or the use of weighted least squares.

Furthermore, for a truly robust assessment of the predictive capability of the regression model, one can use visualization techniques to incorporate uncertainty. This involves plotting the predicted outcomes along with their associated confidence intervals or prediction intervals. In R, functions like `predict(model, interval = "confidence")` or `predict(model, interval = "prediction")` can generate these bounds, which can then be layered onto the ggplot2 visualization using ribbons (`geom_ribbon()`), providing a comprehensive view of the model's reliability across the data range.

Summary of Best Practices in R

To summarize, visualizing the predicted versus actual values is a non-negotiable step in validating any statistical regression model. Whether utilizing the simplicity of **Base R** graphics or the advanced customization capabilities of ggplot2, the core principle remains consistent: evaluate the proximity of data points to the identity line (slope=1, intercept=0).

Key takeaways for generating these plots in R include:

Model Requirement: Ensure the multiple linear regression model is properly fitted using `lm()`.

Prediction Extraction: Always use the `predict()` function on the fitted model to obtain the **predicted values**.

Benchmark Line: The identity line (`abline(0, 1)` in Base R or `geom_abline(intercept=0, slope=1)` in ggplot2) is essential for diagnostic assessment.

Interpretation: Closeness to the diagonal line indicates a strong fit; systematic deviations signal potential model flaws requiring further investigation.

By integrating these visualization techniques into the standard workflow, data scientists can ensure greater transparency and confidence in their predictive modeling efforts.