How to Easily Plot Predicted Regression Values in R

Authored by

stats writer

December 6, 2025

When developing statistical models, particularly those involving regression models, it is critical to assess how well the model's estimates align with the observed data. A fundamental step in model diagnostics involves plotting the **predicted values** against the **actual values** (the dependent variable). This visualization provides immediate, intuitive insight into the model's performance and identifies potential systematic errors.

The goal of this comprehensive guide is to demonstrate, step-by-step, how to generate these crucial diagnostic plots using the R programming language. We will explore two primary approaches: utilizing the built-in capabilities of Base R graphics and leveraging the highly flexible ggplot2 package, catering to different visualization preferences and requirements.

The Importance of Visualizing Model Fit

A statistical regression model is essentially a function designed to minimize the discrepancy between the values it predicts and the true observed outcomes. If a model fits the data perfectly, all data points on a predicted-vs.-actual plot would fall directly onto the 45-degree diagonal line, often referred to as the line of perfect fit. In reality, models are imperfect, but the closer the observed points cluster around this diagonal line, the better the model explains the variance in the data.

Visualizing this relationship helps analysts quickly identify if the model is systematically underestimating or overestimating outcomes across different ranges of the independent variables. For instance, if the points form a curved pattern instead of a straight line, it may indicate that the underlying relationship is non-linear, suggesting that a simple linear model is inappropriate, or that transformations are necessary. This visualization step is far more powerful and informative than merely reviewing numerical metrics like R-squared alone.

Understanding the visual distribution of prediction errors is vital for refining the modeling process. In contexts such as predictive modeling or forecasting, being able to quickly spot outliers or areas of poor fit is necessary for achieving robust and reliable results. We will use a standard multiple linear regression framework for our examples, showcasing the adaptability of these plotting techniques.

Prerequisites and Data Preparation in R

Before plotting, we must ensure the necessary data structure is in place and the regression model has been properly estimated. All examples in this tutorial utilize the R statistical environment. For the second example, you will need the ggplot2 package installed and loaded.

For demonstration purposes, we will create a small sample data frame designed to simulate a scenario where two predictors (x1 and x2) influence an outcome variable (y). This structure mirrors many real-world datasets where analysts use multiple independent variables in a multiple linear

regression approach.

The following code snippet initializes our data and fits the model, steps that are foundational to both the Base R and ggplot2 methods we will explore:

```
# Create the example data frame
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 the multiple linear regression model using the lm() function
model <- lm(y ~ x1 + x2, data=df)

# The 'model' object now contains all necessary coefficients and diagnostic information.
```

Method 1: Generating the Plot Using Base R Graphics

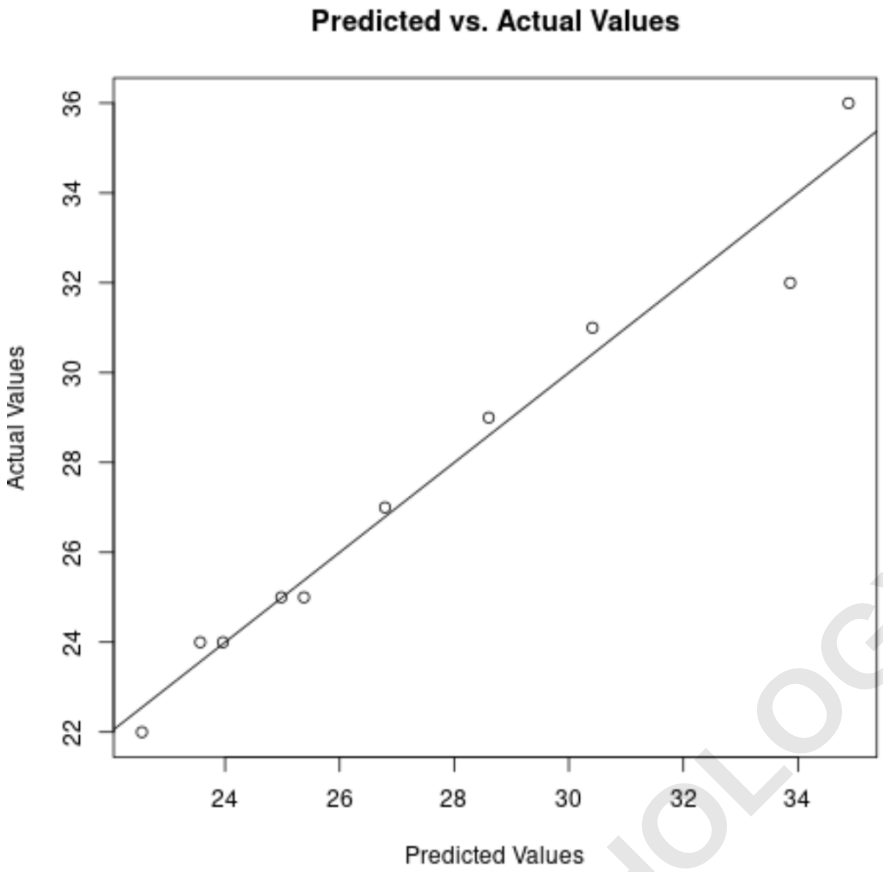
Base R provides powerful, built-in functionality for statistical visualization without requiring external packages. To generate a plot of predicted vs. actual values, we utilize the `predict()` function on our fitted model object. This function extracts the estimated values of the dependent variable based on the model and the input data, providing the crucial metric for the x-axis.

The standard `plot()` command in R is used here, mapping the predicted values to the x-axis and the actual observed values (`df$y`) to the y-axis. Crucially, we then add a diagonal line using the `abline()` function. Setting the intercept (a) to 0 and the slope (b) to 1 creates the identity line (y = x), which represents the ideal scenario of perfect prediction.

This approach is streamlined and highly efficient for quick diagnostic checks. We customize the appearance using arguments like `xlab`, `ylab`, and `main` to ensure clarity and professional presentation of the resulting graphical output. Here is the complete implementation for the Base R approach:

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

# Add the diagonal line (y=x) representing the ideal fit
abline(a=0, b=1)
```



Interpreting the Base R Predicted vs. Actual Plot

Upon examining the resulting scatter plot, the horizontal axis represents the **Predicted Values** generated by our multiple linear regression model, while the vertical axis shows the corresponding **Actual Values** observed in the dataset. The diagonal line running through the center of the plot serves as the benchmark: it signifies the estimated regression line of perfect fit, where Predicted = Actual.

The proximity of the data points to this diagonal line is the primary visual indicator of the model's predictive accuracy. In our example, the data points cluster tightly around the diagonal line. This close alignment strongly suggests that the estimated regression model is performing well and captures the underlying relationship between the predictors (x1 and x2) and the outcome (y) quite effectively. Minimal scatter indicates low prediction error (residuals).

If the points were widely dispersed, or if they showed a clear pattern (e.g., points below the line at low values and above the line at high values), this would signal potential issues such as heteroscedasticity or a mispecified functional form. Because the points in this visualization are clustered tightly, we gain confidence that our current linear model provides a robust explanation of the variance within our sample data.

Creating a Comparison Data Frame for Inspection

While the visual plot offers a powerful summary, it is often useful to inspect the specific numerical differences between the actual and predicted values for individual observations. This is especially helpful for identifying outliers or understanding where the model achieves its best and worst performance.

We can easily compile these values into a new data frame using the `data.frame()` function in R. This data structure facilitates a point-by-point comparison, allowing the analyst to calculate the exact residual (error) for each data point and linking it back to the original input variables if necessary. This step combines the visual assessment with detailed numerical confirmation.

The following code creates and displays this comprehensive comparison table, demonstrating the precise estimates derived from the fitted model:

 # Create a data frame summarizing actual and predicted values
values <- data.frame(actual=df$y, predicted=predict(model))

View the resulting comparison table
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

For example, looking at the first observation, the actual value was 22, while the model predicted 22.54878. The small difference (residual) confirms the visual finding that the model estimates are very close to the true values.

Method 2: Leveraging the Power of ggplot2 for Visualization

For users who prefer highly customizable and publication-quality graphics, the ggplot2 package (part of the R Tidyverse) is the gold standard in statistical visualization. ggplot2 operates on the

grammar of graphics, building plots layer by layer, offering precise control over aesthetics, geometries, and scaling.

To use ggplot2, we first load the library. We then initialize the plot using the `ggplot()` function, mapping the predicted values (derived from `predict(model)`) to the x-aesthetic and the actual values (`y`) to the y-aesthetic. The data frame `df` serves as the primary data source. Unlike the Base R approach, the predicted values must be calculated and often temporarily added within the aesthetic mapping or directly to the data frame before plotting.

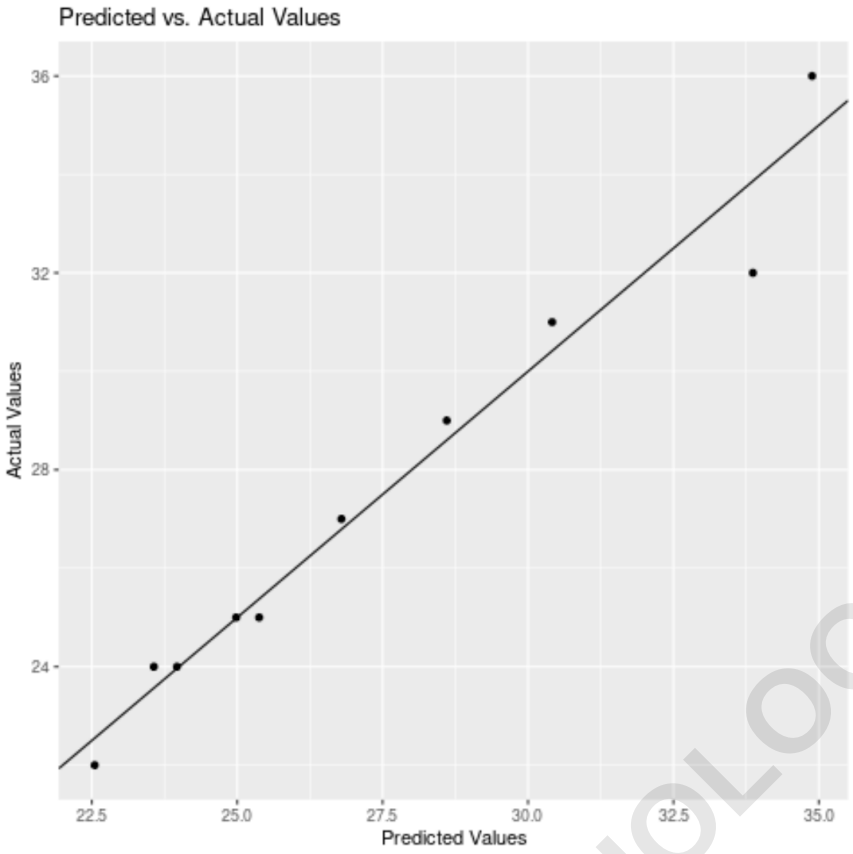
We add two main layers: `geom_point()`, which generates the scatter plot of the data points, and `geom_abline()`, which creates the reference line of perfect fit. Note that in ggplot2, `geom_abline()` requires explicit specification of the `intercept` (0) and `slope` (1) to draw the 45-degree line. Finally, the `labs()` function is used for comprehensive axis and title customization.

library(ggplot2)

# The data frame and model fitting steps remain the same as Example 1.
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))

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

Plot predicted vs. actual values using ggplot2
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 (ggplot2)')



Analyzing the ggplot2 Output

The resulting plot generated by ggplot2 conveys the exact same statistical information as the Base R plot, but typically benefits from cleaner default aesthetics and easier modification for advanced use cases (e.g., adding color based on a third variable, or adding smoothed regression lines).

As observed previously, the scatter of points confirms the strong performance of the multiple linear regression model. The core interpretation remains focused on the distance between the points and the diagonal line: a short distance implies high accuracy, while significant deviations suggest substantial error in the prediction for those specific observations.

By using the `labs()` function, we ensured that the axes are clearly labeled, associating the predicted values with the horizontal dimension and the actual outcomes with the vertical dimension. Understanding this mapping is crucial for correctly diagnosing where the model over- or under-predicts. Points falling above the diagonal line indicate instances where the actual value is higher than the predicted value (underestimation), while points below the line show the actual value is lower than the predicted value (overestimation).

Advanced Considerations: Beyond Simple Prediction Plots

While the Predicted vs. Actual plot is invaluable for assessing overall fit, a complete diagnostic analysis often requires examining other visualization tools. Specifically, plotting the **residuals** (the difference between Actual and Predicted values) against the predicted values is a standard method for checking key assumptions of linear regression.

A residual plot should ideally show no discernible pattern--the points should be randomly scattered around the zero horizontal line. Patterns in residual plots (such as a cone shape or a curve) signal potential violations of assumptions like homoscedasticity (constant variance of errors) or linearity, respectively. R's `plot(model)` function provides these standard diagnostic plots automatically.

Furthermore, analysts can use the predicted values in conjunction with cross-validation techniques to evaluate model generalization. By generating predicted values on a hold-out test set, we can create the Predicted vs. Actual plot for unseen data, offering a more realistic assessment of the model's expected performance in deployment. This comprehensive approach ensures that the model is not only performing well on training data but is also robust enough for real-world application.

Conclusion

Visualizing the relationship between predicted and actual values is a foundational step in validating any regression model. Whether using the straightforward plotting capabilities of Base R or the detailed customization offered by ggplot2, the resulting scatter plot immediately highlights the model's strengths and weaknesses.

The key takeaway is that a high-performing model will produce a plot where data points closely hug the diagonal line of perfect prediction (y = x). Analysts in R are well-equipped with powerful functions like `lm()` and `predict()` to generate these essential diagnostics quickly and efficiently, moving beyond mere numerical summaries to achieve a deeper understanding of their model fit.