Learning Multiple Linear Regression with R: A Step-by-Step Guide

Authored by

Mohammed looti

November 9, 2025

This comprehensive guide provides a detailed, step-by-step walkthrough of how to perform Multiple Linear Regression (MLR) using the powerful statistical programming language R. MLR is a foundational statistical technique essential for modeling the relationship between a single response variable and two or more predictor variables.

A robust MLR analysis requires careful execution of several key stages. We will cover every essential step required for a reliable outcome, starting from the initial data exploration through to advanced model validation and the final interpretation of results.

Thoroughly examining and visualizing the data to check for linear relationships and potential issues.

Fitting the MLR model efficiently using R's core functions, specifically the **lm()** command.

Verifying the crucial statistical assumptions underlying the model, such as the normality and homoscedasticity of residuals.

Interpreting the statistical output, including regression coefficients, significance levels (p-values), and the overall F-test.

Assessing the model's goodness of fit using key metrics like R-squared and the Residual Standard Error (RSE).

Using the finalized, validated model to generate precise quantitative predictions for new data observations.

To demonstrate these concepts practically, we will now begin the process by setting up our environment and preparing a classic dataset for statistical analysis!

Setting Up the Analysis Environment and Data Preparation

For this practical example, we will utilize the widely known, built-in R dataset called mtcars. This dataset is an excellent resource for demonstrating regression concepts as it contains comprehensive information on various attributes for 32 different automobiles from the 1970s. Working with a readily available dataset ensures reproducibility and ease of learning.

Before fitting any regression model, it is considered best practice in statistical workflow to inspect the structure of the raw data. This preliminary inspection helps confirm the variable types and provides an initial overview of the observations. The following code snippet shows the initial six observations of the dataset, giving us a quick overview of the variables available for modeling:

#view first six lines of *mtcars*

head(mtcars)

```
# mpg cyl disp hp drat wt qsec vs am gear carb
#Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
#Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
#Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
#Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
#Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
#Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1
```

The primary objective of this specific MLR model is to predict a car's fuel efficiency, measured in miles per gallon (**mpg**). Therefore, **mpg** will serve as our response (dependent) variable. Based on automotive knowledge, we select three quantitative attributes--engine displacement (**disp**), gross horsepower (**hp**), and rear axle ratio (**drat**)--to act as our predictor (independent) variables. These variables are hypothesized to significantly influence fuel consumption, with displacement and horsepower expected to have negative relationships with efficiency, and the axle ratio potentially having a positive or modest effect.

To maintain a clean working environment and focus solely on the variables relevant to our model, we create a new, simplified data frame named **data**. This step, which involves selecting only the columns necessary for the regression equation, simplifies subsequent analysis and minimizes the risk of inadvertently including irrelevant variables in our model fitting process.

#create new data frame that contains only the variables we would like to use
data <- mtcars

```
#view first six rows of new data frame
head(data)
```

```
# mpg disp hp drat
#Mazda RX4 21.0 160 110 3.90
#Mazda RX4 Wag 21.0 160 110 3.90
#Datsun 710 22.8 108 93 3.85
#Hornet 4 Drive 21.4 258 110 3.08
#Hornet Sportabout 18.7 360 175 3.15
#Valiant 18.1 225 105 2.76
```

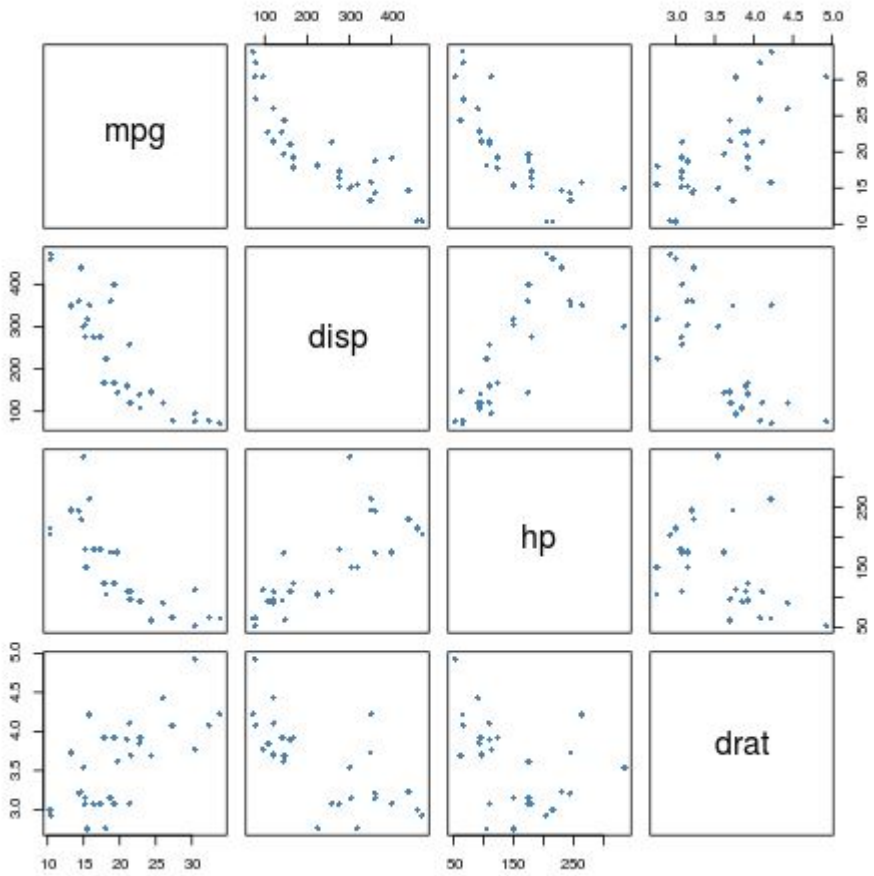
Initial Data Examination and Visualization for Linearity

Before constructing any linear regression model, it is statistically prudent to examine the

relationships between our chosen variables. This initial visual assessment serves two crucial purposes: first, it helps us gain a deeper understanding of the dataset's characteristics; and second, it allows us to verify a fundamental requirement of linear regression: that the predictor variables exhibit a roughly **linear association** with the response variable. If these relationships appear highly non-linear, a linear model may be inappropriate, necessitating data transformation or the use of alternative modeling techniques.

To efficiently visualize all possible pairwise relationships among our four variables (mpg, disp, hp, and drat), we use the built-in R function **pairs()**. This function generates a pairs plot (also known as a scatterplot matrix), where each cell displays the scatterplot between two specific variables. This matrix is an indispensable tool for identifying potential linear trends, checking for outliers, and assessing the risk of multicollinearity (strong correlation between predictors).

pairs(data, pch = 18, col = "steelblue")



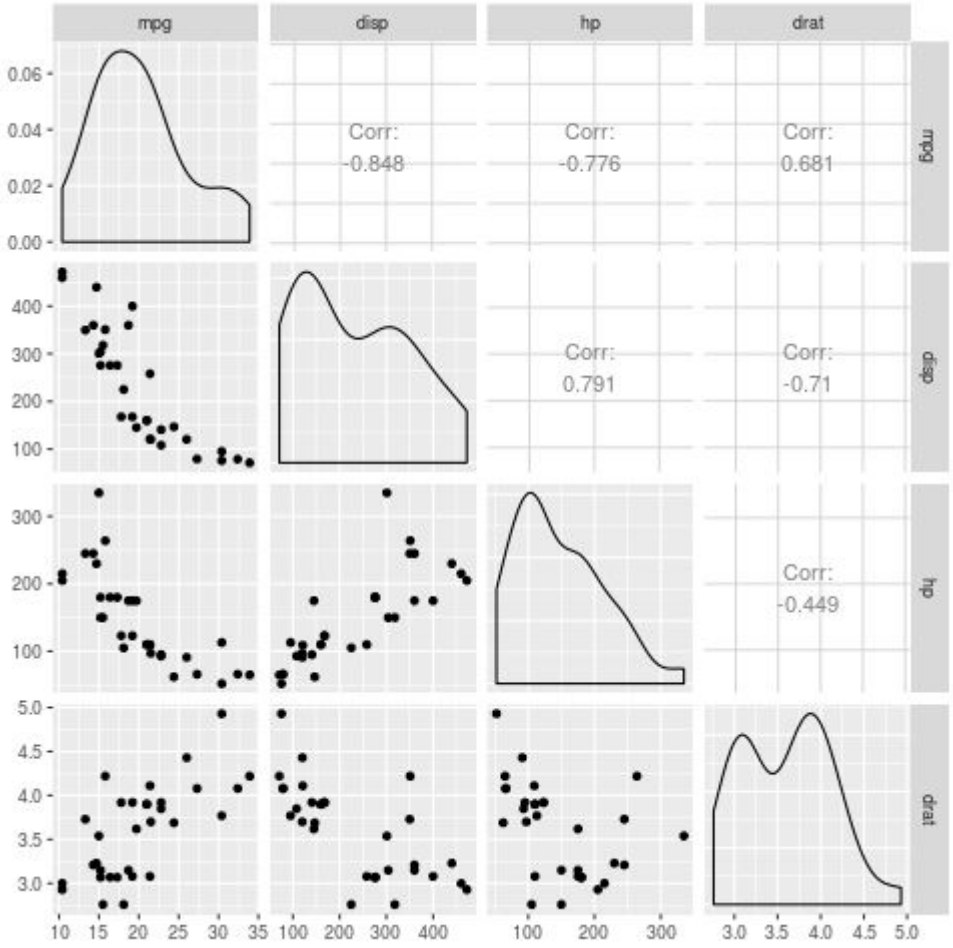
The resulting pairs plot offers immediate insights into the data structure. Focusing on the relationships involving **mpg** (found in the first row or column), we can observe distinct patterns: both **disp** and **hp** show a strong, inverse relationship with **mpg**, meaning higher power and

displacement correlate with lower fuel efficiency. Conversely, the relationship between **mpg** and **drat** shows a modest positive linear correlation. Since all selected predictors demonstrate a noticeable linear correlation with the response variable **mpg**, we have sufficient justification to proceed with constructing and fitting the model.

While the basic **pairs()** function is useful, we can enhance our understanding by using the **ggpairs()** function from the **GGally** library. This package generates a more comprehensive plot that includes not only scatterplots but also histograms of individual variables along the diagonal and the actual quantitative Pearson correlation coefficients for each pair. The inclusion of these coefficients provides precise numerical confirmation of the strength and direction of the linear relationships observed visually.

```
#install and load the GGally library
install.packages("GGally")
library(GGally)

#generate the pairs plot
ggpairs(data)
```



Constructing and Fitting the Multiple Linear Regression Model

In R, the standard function for fitting linear models, which encompasses both simple and multiple linear regression, is **lm()** (short for linear model). This function uses a formula interface to specify the precise mathematical relationship between the response variable and the set of predictor variables. Understanding this formula is key to successful modeling in R. The general syntax for defining a multiple linear regression model is intuitive and follows a clear pattern:

lm(response_variable ~ predictor_variable1 + predictor_variable2 + ..., data = data)

In this formula, the tilde (**~**) acts as the separator, placing the response variable (the target we wish to predict) on the left side and all the independent predictors on the right. The plus sign (**+**) is used to include multiple independent predictors simultaneously in the model, assuming an additive effect. Finally, the **data = data** argument specifies the data frame containing all the necessary variables.

Applying this syntax to our specific analysis, where we are modeling **mpg** based on **disp**, **hp**, and

drat, we fit the model using the following concise code. The output of this operation is stored in an object named **model**. This object is not merely a set of coefficients; it is a comprehensive structure containing all the necessary statistical information about the fitted regression line, including coefficients, residuals, degrees of freedom, and model diagnostics.

model <- lm(mpg ~ disp + hp + drat, data = data)

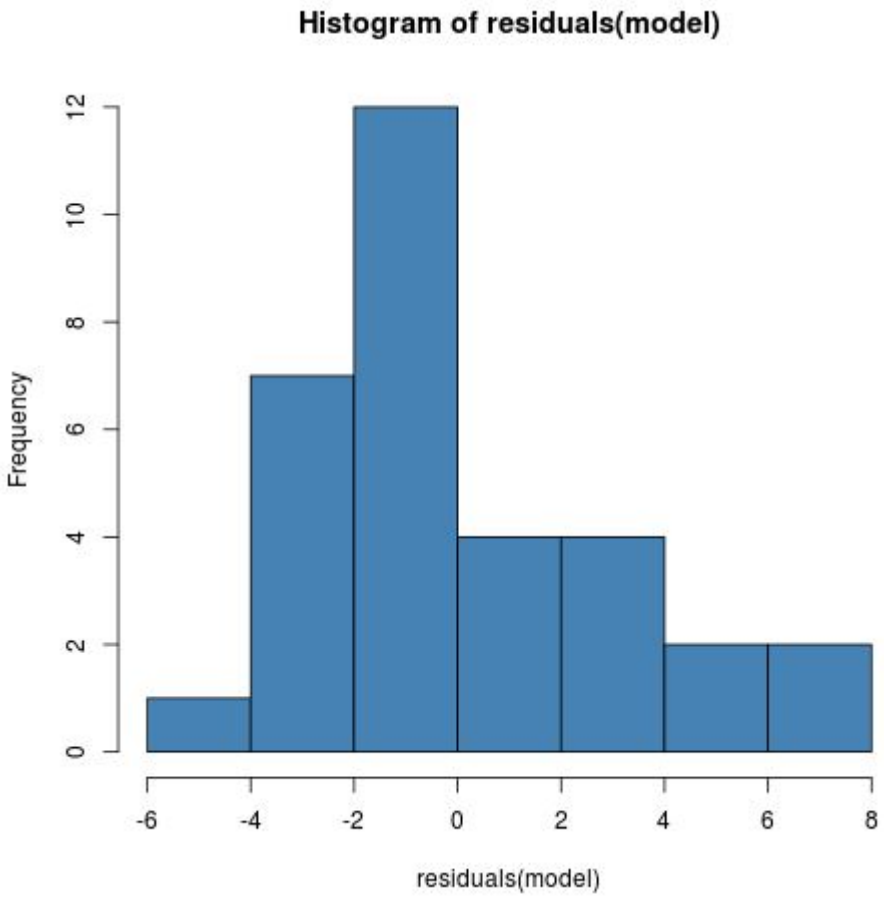
Verifying Model Assumptions: Diagnostics and Reliability

Before drawing meaningful conclusions from the model's coefficients or assessing its predictive power, we must first verify that the underlying statistical assumptions of the linear regression model are reasonably satisfied. If these assumptions are severely violated, the standard errors may be unreliable, p-values could be biased, and the resulting inferences may be incorrect or misleading. The two most critical assumptions to verify visually are the normality of residuals and the consistency of residual variance.

1. The distribution of model residuals should be approximately normal.

Residuals represent the vertical distance between the observed data points and the fitted regression line--they are the errors in prediction. For valid statistical inference, particularly hypothesis testing, it is assumed that these errors follow a normal distribution, centered around zero. We visually check this assumption by generating a histogram of the residuals from our fitted model:

hist(residuals(model), col = "steelblue")



Upon reviewing the histogram, we observe that the distribution may exhibit slight skewness or minor deviations from a perfect bell curve. However, it is generally centralized around zero and appears close enough to normal that it should not introduce major concerns for inference, especially given the relatively small sample size of the **mtcars** dataset (n=32). In larger samples, more formal tests (like the Shapiro-Wilk test) might be used, but visual checks are often sufficient for practical modeling.

2. The variance of the residuals should be consistent for all observations.

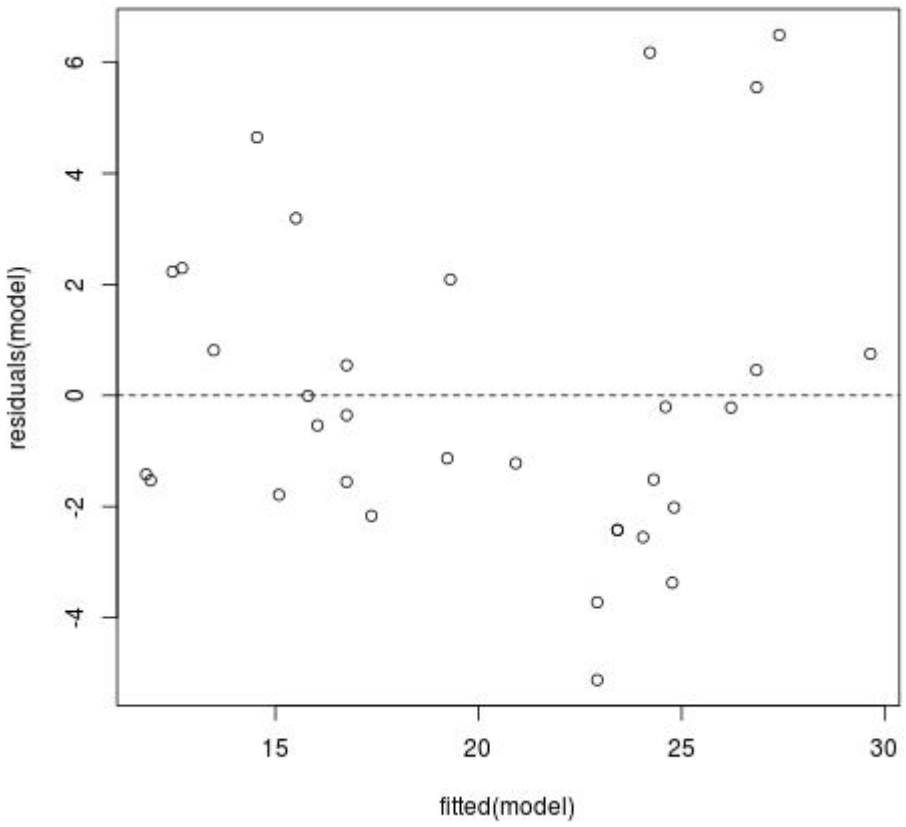
This preferred condition is known as homoscedasticity, meaning the spread of the residuals remains constant across all levels of the predicted response variable. Conversely, a violation of this assumption, where the variance changes systematically (often fanning out or tightening as the predicted value changes), is known as heteroskedasticity. Heteroskedasticity can inflate the calculated standard errors, leading to incorrect confidence intervals and p-values.

To check for homoscedasticity, we generate a fitted value versus residual plot. The ideal outcome is a random, patternless scatter of points clustered evenly around the horizontal line at zero:

#create fitted value vs residual plot

```
plot(fitted(model), residuals(model))

#add horizontal line at 0
abline(h = 0, lty = 2)
```



Upon inspection of the plot, we desire the residuals to be equally scattered across the range of fitted values. In this case, while not perfectly uniform, the points are distributed reasonably randomly. There is a slight visual indication that the scatter might become marginally larger for higher fitted values (which correspond to lower MPG values), suggesting a minor degree of heteroskedasticity. However, this pattern is not severe enough to necessitate complex remedies such as weighted least squares or model transformation, allowing us to proceed confidently to the interpretation phase.

Interpreting the Regression Output and Coefficients

With the model assumptions verified, the next critical step is to analyze the statistical output generated by the model. The **summary()** function in R provides a comprehensive report of the model's performance, the estimated coefficients for all variables, and the goodness-of-fit metrics.

summary(model)

```
#Call:
#lm(formula = mpg ~ disp + hp + drat, data = data)
#
#Residuals:
# Min 1Q Median 3Q Max
#-5.1225 -1.8454 -0.4456 1.1342 6.4958
#
#Coefficients:
# Estimate Std. Error t value Pr(>|t|)
#(Intercept) 19.344293 6.370882 3.036 0.00513 **
#disp -0.019232 0.009371 -2.052 0.04960 *
#hp -0.031229 0.013345 -2.340 0.02663 *
#drat 2.714975 1.487366 1.825 0.07863 .
#---
#Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
#Residual standard error: 3.008 on 28 degrees of freedom
#Multiple R-squared: 0.775, Adjusted R-squared: 0.7509
#F-statistic: 32.15 on 3 and 28 DF, p-value: 3.28e-09
```

First, we assess the model's overall statistical utility by looking at the bottom line. The overall F-statistic of the model is reported as **32.15**, associated with an extremely small p-value of **3.28e-09** (which is substantially less than the common significance threshold of 0.05). Since this p-value is highly significant, we reject the null hypothesis that all regression coefficients are zero. This confirms that the regression model, taken as a whole, is statistically useful and explains a significant portion of the variation in **mpg**.

Next, we examine the individual coefficients to determine the unique contribution of each predictor. The interpretation of these coefficients must always follow the "ceteris paribus" rule, meaning we interpret the effect of one variable while holding the values of all other variables in the model constant:

disp (Engine Displacement): The coefficient is **-0.019232**. This variable is statistically significant (p-value = 0.04960) at the 0.05 level. Interpretation: Holding horsepower and rear axle ratio constant, a one-unit increase in engine displacement is associated with an average decrease of 0.019232 units in **mpg**.

hp (Gross Horsepower): The coefficient is **-0.031229**. This variable is also statistically significant

(p-value = 0.02663). Interpretation: Holding displacement and rear axle ratio constant, a one-unit increase in horsepower is associated with an average decrease of 0.031229 units in **mpg**.

drat (Rear Axle Ratio): The coefficient is **2.714975**. This variable is statistically significant at the 0.10 level (p-value = 0.07863). Interpretation: Holding displacement and horsepower constant, a one-unit increase in the rear axle ratio is associated with an average increase of 2.714975 units in **mpg**.

Assessing Model Performance (Goodness of Fit)

To quantify how well the fitted regression line approximates the actual data points, we assess the model's goodness of fit using two primary metrics derived from the summary output. These metrics provide essential context regarding the model's predictive power and overall accuracy in capturing the variability of the response variable.

1. Multiple R-Squared and Adjusted R-Squared

The Multiple R-Squared value (or Coefficient of Determination) is a crucial measure that quantifies the strength of the linear relationship between the combined set of predictor variables and the response variable. More specifically, R-squared represents the proportion of the total variance in the response variable that is explained by the independent variables included in the model.

In our example, the Multiple R-squared is reported as **0.775**. This means that 77.5% of the total variability observed in **mpg** can be accounted for by the combined effects of engine displacement, horsepower, and rear axle ratio. This high value suggests a strong fit, indicating that the chosen predictors are highly relevant for modeling fuel efficiency. The Adjusted R-squared (**0.7509**) is also provided; this metric is generally preferred in multiple regression as it slightly penalizes the R-squared for the inclusion of multiple predictors, offering a more conservative and honest estimate of fit strength that accounts for model complexity.

2. Residual Standard Error (RSE)

The Residual Standard Error (RSE), often referred to as the standard error of the regression, is an estimate of the standard deviation of the error term (the residuals). Conceptually, it measures the average magnitude of the residuals--that is, the average distance that the observed data points fall from the fitted regression hyperplane. Critically, it is expressed in the same units as the response variable.

In this model, the RSE is **3.008 units**. This implies that, on average, the model's prediction for **mpg** will deviate from the car's actual **mpg** by approximately 3.008 miles per gallon. A smaller RSE value indicates a more precise and accurate model fit, suggesting the predictions are closer to the real-world observations.

Using the Validated Model to Make Predictions

The final step in our analysis is to utilize the statistically validated model to make quantitative predictions for new data points. Based on the coefficient estimates derived from the summary output, we can construct the explicit fitted multiple linear regression equation, which mathematically summarizes the relationships we found:

mpghat = 19.344 - 0.01923*disp - 0.03123*hp + 2.715*drat

We can now apply this equation to estimate the fuel efficiency (**mpg**) for a hypothetical new car, provided we know its predictor attributes. For instance, let's predict the **mpg** for a vehicle with the following characteristics:

Engine Displacement (*disp*) = **220** cubic inches

Gross Horsepower (*hp*) = **150** hp

Rear Axle Ratio (*drat*) = **3**

The R code below automates this calculation by first extracting the precise coefficient estimates from the model object and then substituting the new predictor values into the regression equation. This method leverages the precision of the stored model object rather than relying on manual coefficient rounding.

```
#define the coefficients from the model output
intercept <- coef(summary(model))
disp <- coef(summary(model))
hp <- coef(summary(model))
drat <- coef(summary(model))

#use the model coefficients to predict the value for mpg
intercept + disp*220 + hp*150 + drat*3

# 18.57373
```

Based on the calculated output, the model predicts that a car possessing an engine displacement of 220, 150 horsepower, and a rear axle ratio of 3 would have an estimated fuel efficiency (**mpg**) of approximately **18.57** miles per gallon. This confirms the practical utility and predictive power of the fitted multiple linear regression model in a real-world context.

You can find the complete R code used in this tutorial here.

Additional Resources for Regression Modeling

To further expand your knowledge of regression analysis in R, the following tutorials explore how to fit and interpret other specialized types of regression models:

How to Perform Exponential Regression in R