How to Perform Multiple Linear Regression in R: A Step-by-Step Guide

Authored by

stats writer

March 4, 2026

Understanding the Foundations of Multiple Linear Regression

Multiple linear regression is a sophisticated **statistical method** used to model the relationship between a single **continuous response variable** and two or more **explanatory variables**. By fitting a **linear equation** to observed data, researchers and analysts can discern how various **independent variables** simultaneously influence a **dependent variable**. This technique is a fundamental pillar of **econometrics**, **data science**, and **predictive modeling**, allowing for the isolation of specific effects while holding other factors constant. In the context of the **R programming language**, performing this analysis is highly efficient due to its robust built-in libraries and specialized functions designed for **statistical computing**.

The primary goal of **multiple linear regression** is to find the "best-fit" line--or more accurately, a **hyperplane** in higher dimensions--that minimizes the sum of the squares of the vertical deviations between each data point and the fitted surface. This process is commonly known as **Ordinary Least Squares** (OLS). Before diving into the technical implementation, it is essential to understand that the quality of the **predictive model** depends heavily on the quality of the input data and the adherence to specific **statistical assumptions**. A well-constructed model not only provides **predictions** but also offers deep insights into the **correlation** and **causality** underlying the dataset.

In this comprehensive guide, we will explore the practical steps required to execute a **multiple linear regression** analysis within the **R** environment. We will cover everything from initial **data exploration** and **model fitting** to the rigorous **diagnostic testing** required to ensure the model's validity. By utilizing the **mtcars** dataset--a classic **benchmark** in the **R** community--we will demonstrate how to transform raw information into actionable **statistical insights**. This walkthrough is designed to provide both the code necessary for execution and the theoretical context required for professional **data analysis**.

To begin the process, a researcher must ensure their **R environment** is properly configured. While **R** includes many powerful features in its **base package**, additional libraries such as **GGally** or **ggplot2** can significantly enhance the **visualization** capabilities. The ability to effectively communicate **statistical results** through clear, concise **data visualizations** is just as important as the mathematical accuracy of the model itself. As we progress, we will utilize several core functions, including **lm()** for modeling and **summary()** for evaluating the **statistical significance** of our findings.

Setting Up the Data Environment in R

The first practical step in any **data analysis** project involves loading the dataset and preparing the **data structures**. For this tutorial, we leverage the **mtcars** dataset, which is built directly into **R**.

This dataset originated from the 1974 **Motor Trend** US magazine and comprises fuel consumption and ten aspects of automobile design and performance for 32 automobiles. Using a well-known **dataset** allows us to focus on the **regression methodology** rather than the complexities of **data cleaning** or **external API** integration.

Before fitting the **linear model**, it is vital to inspect the **dataframe** to understand the scale and **data types** of the variables involved. We will focus on **mpg** (miles per gallon) as our **response variable**, which represents the efficiency of the vehicles. To predict this value, we will select three **predictor variables**: **disp** (displacement), **hp** (gross horsepower), and **drat** (rear axle ratio). By subsetting the data, we create a more manageable **data object** that contains only the information relevant to our specific **hypothesis**.

#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

Once the broader dataset is understood, we isolate the specific columns needed for our **multiple linear regression**. This practice of **feature selection** is critical in **machine learning** and **statistics** to prevent the model from becoming overly complex, a phenomenon known as **overfitting**. By focusing on a specific subset of **variables**, we can more clearly observe the **linear relationships** and ensure that our **degrees of freedom** remain appropriate for the sample size.

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

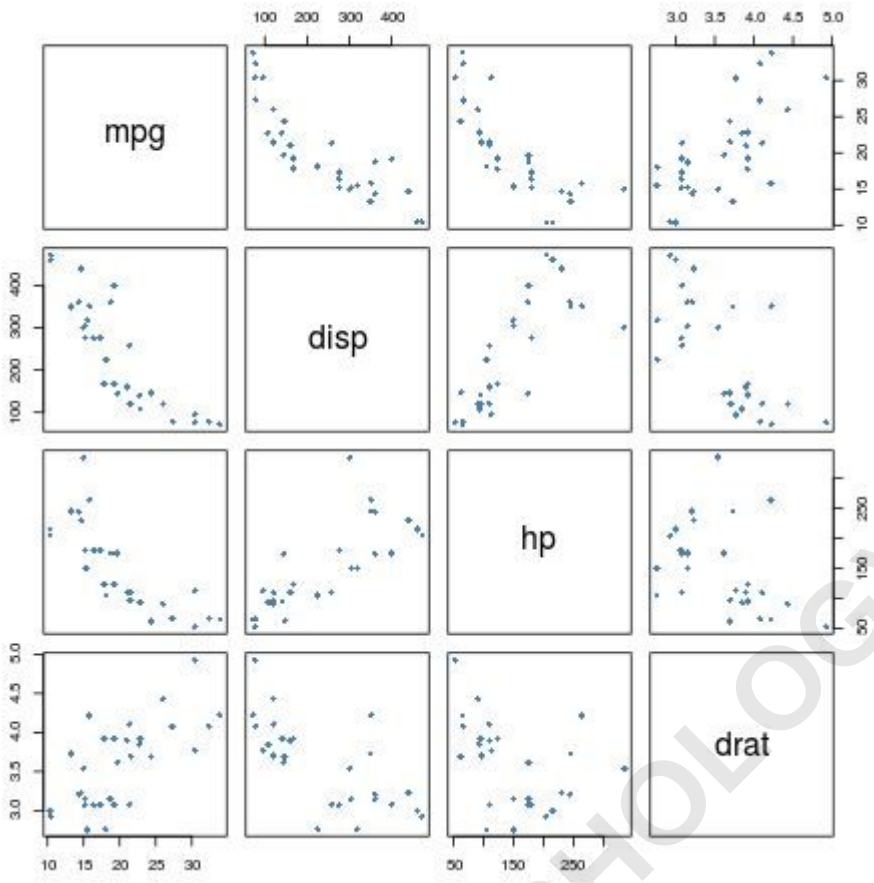
The **data manipulation** shown above uses **base R** syntax to select columns by name. This creates a clean **matrix-like** structure where each row represents an observation and each column represents a **statistical variable**. With the data properly organized, we can proceed to the **Exploratory Data Analysis** (EDA) phase, where we visually verify the viability of a **linear approach** before executing the formal **regression analysis**.

Exploratory Data Analysis and Visualization

Before performing a **multiple linear regression**, it is crucial to perform **Exploratory Data Analysis** to check for **linearity**. Linear regression assumes that the relationship between the **independent variables** and the **dependent variable** is, as the name suggests, linear. If the underlying relationship is **curvilinear** or **exponential**, a standard **linear model** will produce inaccurate **coefficients** and poor **predictive power**. A **scatterplot matrix** is an excellent tool for identifying these patterns across multiple variables simultaneously.

In **R**, the **pairs()** function provides a quick way to generate a **matrix of scatterplots**. This visualization allows us to see the **bivariate relationship** between every possible pair of variables in our **dataframe**. We are specifically looking for trends where the data points appear to follow a straight line, whether ascending or descending. Additionally, this step helps in identifying **outliers**--data points that deviate significantly from the rest of the observations and could potentially **skew** our **regression results**.

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



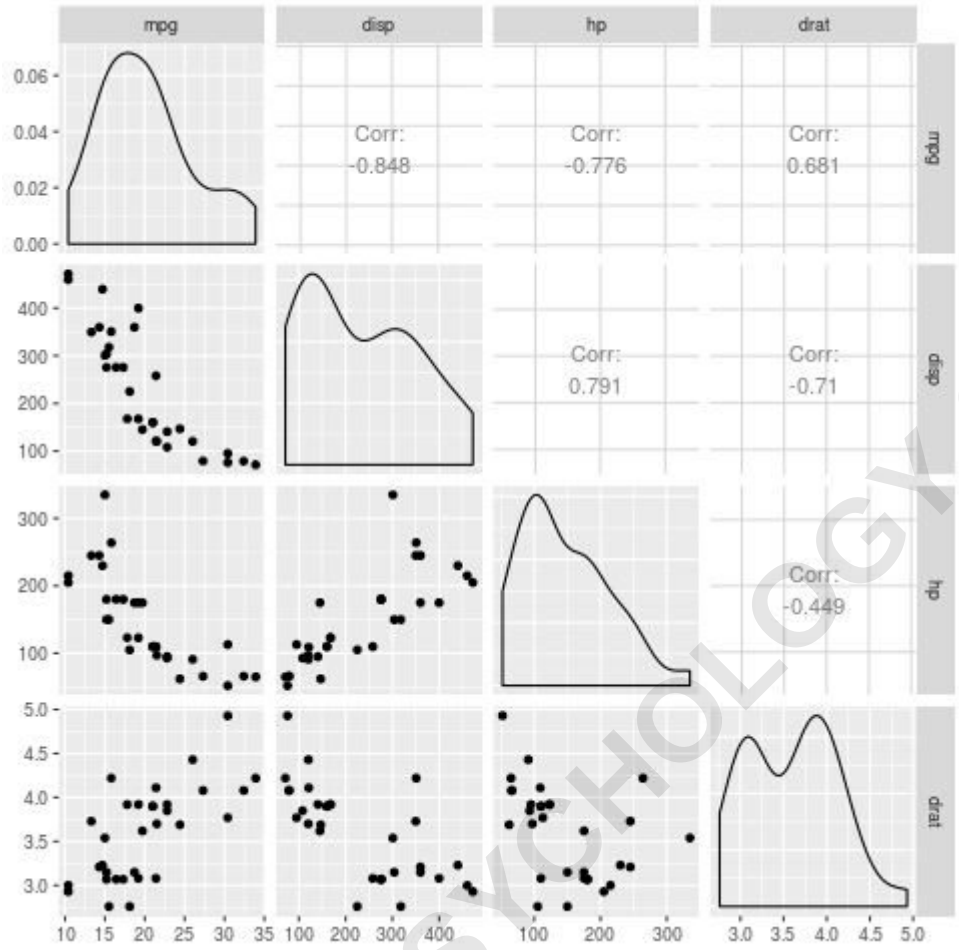
Upon reviewing the **pairs plot**, several key observations can be made regarding the **correlation** between the variables. We can see that **mpg** shares a strong **negative correlation** with **disp** and **hp**, meaning that as engine displacement and horsepower increase, fuel efficiency tends to decrease. Conversely, **drat** shows a **positive correlation** with **mpg**. These visual cues strongly suggest that a **linear model** is an appropriate choice for this specific **dataset**.

To gain even deeper insights, we can use the **GGally** library to calculate the exact **Pearson correlation coefficient** for each pair. This **coefficient** ranges from -1 to 1, where 1 indicates a perfect **positive correlation**, -1 indicates a perfect **negative correlation**, and 0 indicates no linear relationship. Quantifying these relationships helps in identifying **multicollinearity**, which occurs when **predictor variables** are too highly correlated with each other, potentially complicating the **regression analysis**.

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

#generate the pairs plot

ggpairs(data)



By interpreting the **correlation coefficients** provided by **ggpairs()**, we confirm the initial visual assessment. The high **correlation** values between the predictors and the **response variable** justify the inclusion of **disp**, **hp**, and **drat** in our **regression equation**. With this **exploratory** phase complete, we have a high degree of confidence in moving forward with **fitting the model** using the **lm()** function.

Implementing the Multiple Linear Regression Model

In **R**, the **lm()** function is the primary tool used for fitting **linear models**. The name "lm" stands for **linear model**, and it utilizes a **formula interface** that is standard across many **R** functions. The basic syntax follows the pattern **response ~ predictors**, where the **tilde** (~) operator separates the **dependent variable** from the **independent variables**. When performing **multiple linear regression**, we simply add the predictors together using the plus (+) sign.

The **lm()** function calculates the **regression coefficients** using the **least squares** method. This

mathematical approach seeks to minimize the **residual sum of squares**, which is the sum of the squared differences between the observed values and the values predicted by the **linear equation**. The resulting **model object** contains a wealth of information, including the **intercept**, the slope for each **predictor**, and the **residuals** produced during the fitting process.

 # Basic syntax structure:
 # lm(response_variable ~ predictor_variable1 + predictor_variable2 + ..., data = data)

# Applying the syntax to our specific car dataset:
model <- lm(mpg ~ disp + hp + drat, data = data)

Once the **model** has been assigned to a variable, in this case named **model**, it becomes a complex **list object** in **R**. This object stores not just the **coefficients**, but also the **fitted values**, **rank**, and **call** parameters. It is important to note that **R** handles **categorical variables** (if they were present) by automatically creating **dummy variables**, though in our current example, all predictors are **numeric**.

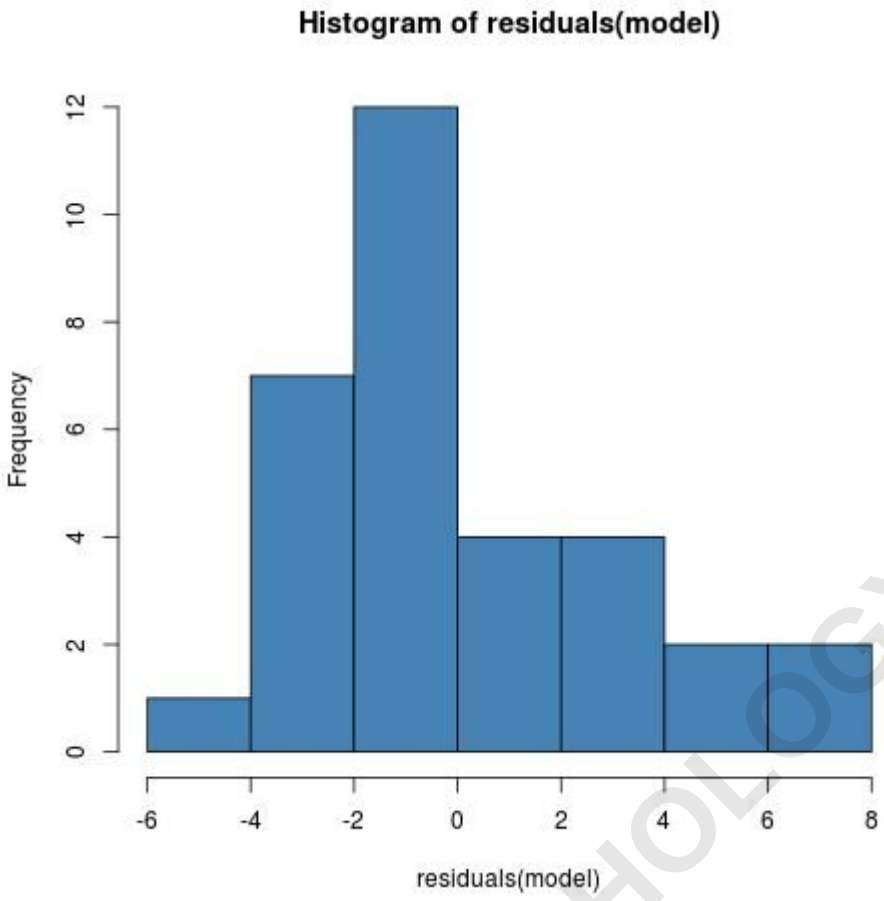
Executing the **lm()** function is only the beginning of the **statistical analysis**. While the function provides the mathematical solution for the **linear equation**, it does not automatically tell us if the model is "good" or if the results are **statistically significant**. To determine the utility of the model, we must proceed to check the **underlying assumptions** and then interpret the **summary output**, which provides the **p-values** and **standard errors** necessary for a rigorous evaluation.

Validating Statistical Assumptions

Before relying on the results of a **multiple linear regression**, a responsible **data scientist** must verify that the **Gauss-Markov assumptions** are satisfied. One of the most critical assumptions is that the **residuals** (the differences between observed and predicted values) follow a **normal distribution**. If the **residuals** are not **normally distributed**, the **standard errors** and **p-values** generated by the model may be biased, leading to incorrect **statistical inferences**.

To test for **normality**, we can generate a **histogram** of the **residuals**. In a perfectly specified model, the **histogram** should resemble a **bell curve** centered around zero. While minor deviations are common in real-world **data sets**, significant **skewness** or **kurtosis** would suggest that a **linear model** might not be the best fit for the data, or that a **variable transformation** (such as a **log transformation**) might be necessary.

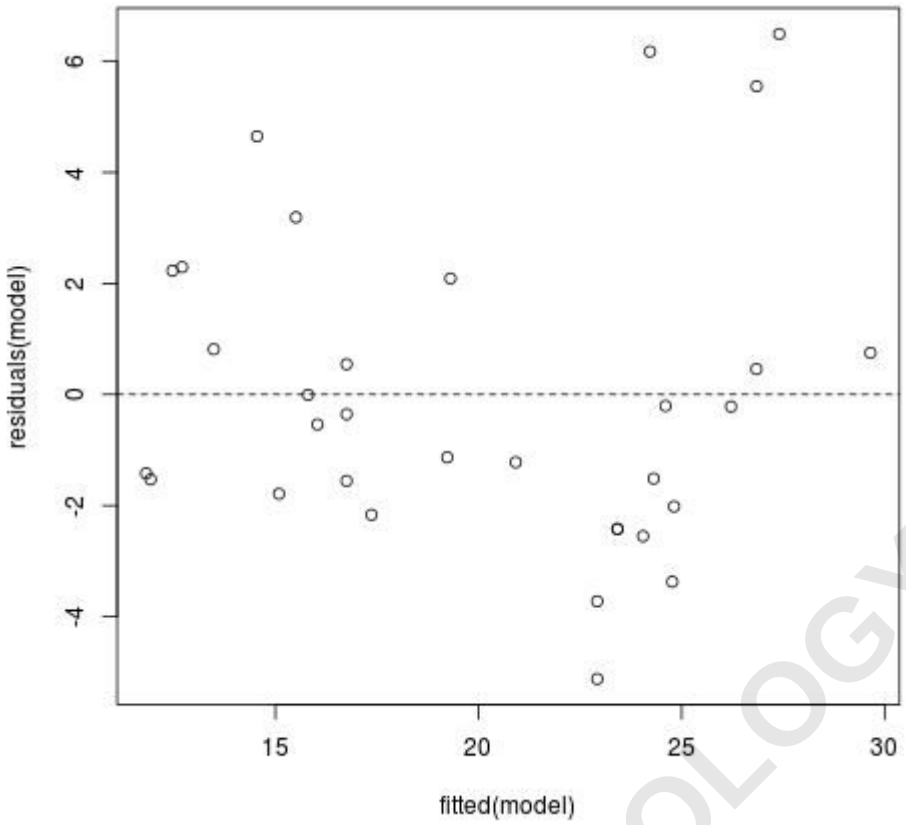
hist(residuals(model), col = "steelblue")



Another essential assumption is **homoskedasticity**, which implies that the **variance** of the **residuals** is constant across all levels of the **independent variables**. The opposite of this is **heteroskedasticity**, where the spread of the **residuals** changes systematically as the **fitted values** increase or decrease. This can be visually diagnosed by creating a **fitted values vs. residuals plot**. In a healthy model, the points should be randomly scattered around a horizontal line at zero, with no discernible patterns like "funneling" or "bowing."

#create fitted value vs residual plot
plot(fitted(model), residuals(model))

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



By examining these **diagnostic plots**, we can confirm the validity of our **multiple linear regression** model. If the **fitted vs. residuals plot** shows a random distribution, we have met the requirement for **homoskedasticity**. If these assumptions hold, we can proceed to interpret the **regression coefficients** with confidence, knowing that our **hypothesis tests** (like the **t-test** for individual coefficients) are **statistically valid**.

Interpreting the Model Summary and Coefficients

The **summary()** function in **R** provides the most comprehensive look at the performance of our **multiple linear regression**. This output is divided into several sections: the **Call**, **Residuals**, **Coefficients**, and **Global Statistics**. The **Coefficients** table is often the most scrutinized section, as it provides the **estimated slope** for each **predictor variable**, along with its **standard error**, **t-value**, and **p-value**.

A **p-value** less than a predefined **alpha level** (typically 0.05) indicates that a **predictor** is **statistically significant**. In our model, we examine how **disp**, **hp**, and **drat** influence **mpg**. The **Estimate** column tells us the magnitude of the effect; for example, a negative coefficient for **hp** suggests that as **horsepower** increases, **fuel economy** decreases, provided all other **variables** in the model remain constant.

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
```

The **F-statistic** and its associated **p-value** at the bottom of the output test the **null hypothesis** that all of the **regression coefficients** are equal to zero. A significant **F-test** (like the one seen above with a p-value of 3.28e-09) suggests that at least one of the **independent variables** in our model has a non-zero effect on the **response variable**. This confirms that the **multiple linear regression** model, as a whole, provides a better fit than a model with no **predictors**.

When interpreting these **coefficients**, it is important to remember the concept of **ceteris paribus**--"all other things being equal." The **coefficient** for **disp** (-0.019) means that for every one-unit increase in displacement, we expect **mpg** to drop by 0.019 units, assuming **horsepower** and **axle ratio** do not change. This level of detail allows for a granular understanding of how various **mechanical attributes** contribute to the overall **efficiency** of the vehicle.

Assessing Goodness of Fit and Error Metrics

To determine the overall quality and **predictive accuracy** of the **multiple linear regression** model, we look at the **R-squared** and **Residual Standard Error**. The **Multiple R-squared** value represents the proportion of the **variance** in the **dependent variable** that is explained by the **independent variables**. In our example, an **R-squared** of approximately 0.775 indicates that

77.5% of the variation in **mpg** can be accounted for by **displacement**, **horsepower**, and **axle ratio**.

However, **R-squared** will always increase as more **variables** are added to the model, even if those variables are not meaningful. This is why **statisticians** often prefer the **Adjusted R-squared**, which penalizes the score for adding **parameters** that do not improve the model significantly. The **Adjusted R-squared** provides a more honest assessment of the model's **explanatory power**, especially when comparing models with a different number of **predictors**.

The **Residual Standard Error** (RSE) is another vital metric, representing the **standard deviation** of the **residuals**. It provides an estimate of the average amount that the **observed values** deviate from the **regression line**. In this car analysis, an RSE of 3.008 means that our **predictions** of **mpg** are, on average, off by about three units. Lower values of RSE indicate a better fit, as the **data points** cluster more tightly around the **fitted model**.

Evaluating these **goodness-of-fit** metrics is a balancing act. While a high **R-squared** is desirable, one must be wary of **overfitting**, where the model becomes so tailored to the **training data** that it fails to generalize to new, unseen **observations**. By considering the **R-squared**, **Adjusted R-squared**, and **Residual Standard Error** together, we can form a holistic view of the model's **reliability** and **validity** for **forecasting** purposes.

Using the Regression Equation for Predictions

The ultimate utility of a **multiple linear regression** model lies in its ability to make **predictions** for new data points. Based on the **coefficients** derived from our **summary output**, we can construct a **mathematical equation** for our model. The **intercept** serves as the starting point, and each **predictor variable** is multiplied by its respective **coefficient** to calculate the **predicted response**. In this case, our equation is: **mpg = 19.344 - (0.019 * disp) - (0.031 * hp) + (2.715 * drat)**.

Suppose we want to predict the fuel efficiency of a hypothetical car with a **displacement** of 220, **horsepower** of 150, and a **rear axle ratio** of 3. By plugging these **independent variables** into our **regression equation**, we can calculate the **point estimate** for its **mpg**. While we can do this manually, **R** provides automated ways to extract **coefficients** and perform these calculations, ensuring **numerical precision** and reducing the risk of human error.

```
#define the coefficients from the model output
intercept <- coef(summary(model))
disp_coeff <- coef(summary(model))
hp_coeff <- coef(summary(model))
drat_coeff <- coef(summary(model))
```

```
#use the model coefficients to predict the value for mpg
intercept + (disp_coeff * 220) + (hp_coeff * 150) + (drat_coeff * 3)

# 18.57373
```

The result of **18.57373** is our model's **best estimate** for the vehicle's **miles per gallon**. Beyond simple **point estimates**, **R** can also generate **confidence intervals** and **prediction intervals**, which provide a range of values within which we expect the true **response variable** to fall. This is particularly useful in **risk management** and **strategic planning**, where understanding the **uncertainty** of a prediction is just as important as the prediction itself.

In conclusion, **multiple linear regression** in **R** is a powerful **analytical workflow** that moves from **data preparation** and **visualization** to **model implementation** and **validation**. By following these structured steps, you can uncover the complex **interdependencies** between variables and build robust **predictive models**. Whether you are analyzing **automotive data** or **financial markets**, the **lm()** function and its associated **diagnostic tools** provide a reliable framework for rigorous **statistical inquiry**.

The following tutorials explain how to fit other types of regression models in R:

How to Perform Exponential Regression in R