Regression Model

Simple Linear Regression Model

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The simplest form is simple linear regression,which involves one dependent and independent variable.

Assumptions of the Linear Regression Model

1. **Linearity**:The relationship between the dependent and independent variable is linear.
2. **Independence**: The residuals (errors) are independent.
3. **Homoscedasticity**:The residuals have constant variance at every level of the independent variable.
4. **Normality**: The residuals of the model are normally distributed.
5. **No Multicollinearity**: In multiple regression, the independent variables are not highly correlated with each other.

Parameters of the Linear Regression Model

1. **Intercept (beta_0)**:The expected value of the dependent variable when all independent variables are zero.
2. **Slope (beta_1)**:The change in the dependent variable for a one-unit change in the independent variable.

Estimating the Intercept and Slope

The linear regression model can be written as:

y = beta_0 + beta_1X + epsilon

1. where (y) is the dependent variable, (x) is the independent variable (beta_0) is the intercept, (beta_1) is the slope, and (epsilon) is the error term.
2. The parameters (beta_0) and (beta_1) are estimated using the least squares method, which minimizes the sum of the squared residuals.

Fitting the Regression Model in R

Let's go through the steps to fit a linear regression model in R.

```r
# Load the Data
set.seed(123)
x <- rnorm(100, mean = 50, sd = 10)
y <- 2 * x + rnorm(100, mean = 0, sd = 5)
data <- data.frame(x, y)

# Fit the linear regression model
model <- lm(y ~ x, data = data)

# Summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5367 -3.4175 -0.4375  2.9032 16.4520
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.79778    2.76324   0.289    0.773
## x            1.97376    0.05344  36.935   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.854 on 98 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.9323
## F-statistic:  1364 on 1 and 98 DF,  p-value: < 2.2e-16
```

Checking Linearity in the Data

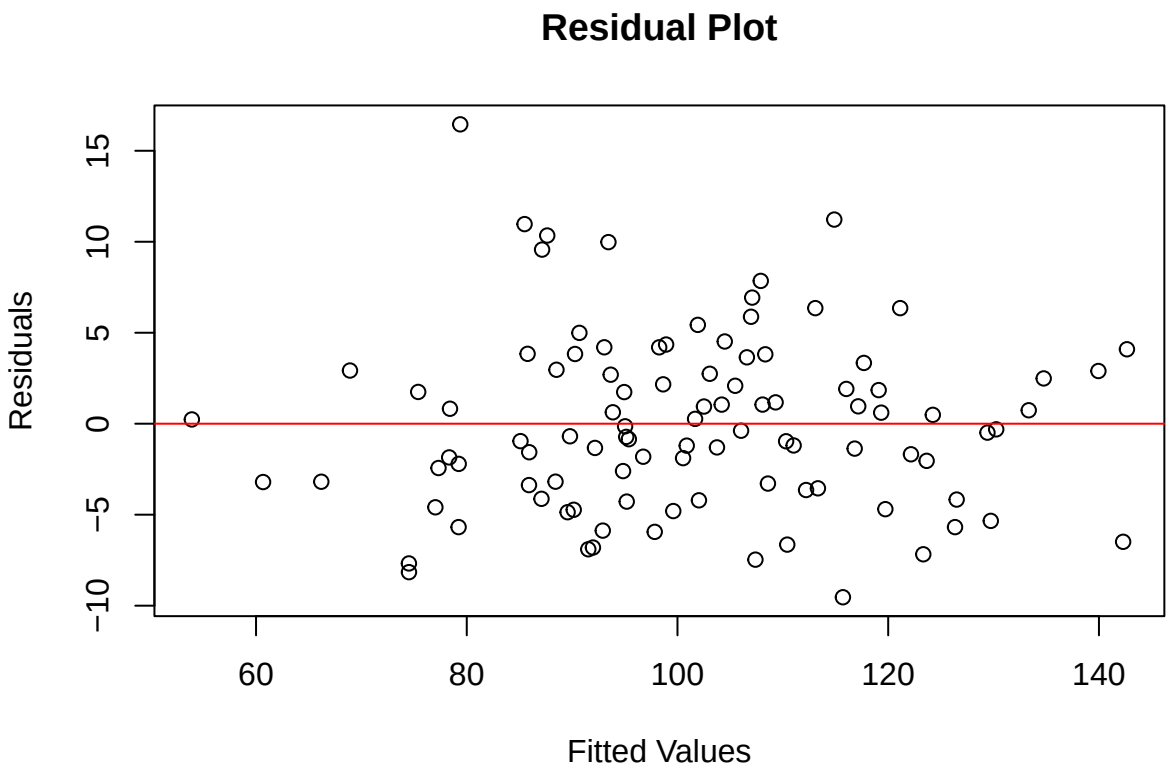
You can visualize the relationship between the dependent and independent variables using a scatter plot with a regression line.

```r
# Scatter plot with regression line
plot(data$x, data$y, main = "Scatter Plot with Regression Line",
     xlab = "Independent Variable (x)",
     ylab = "Dependent Variable (y)")
abline(model, col = "red")
```

Illustrating Residuals

Residuals are the differences between the observed and predicted values.You can plot the residuals to check the assumptions of the model.

```
# Residual plot
plot(model$fitted.values, model$residuals, main = "Residual Plot",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



Coefficient of Determination ((R^2))

The coefficient of determination, (R^2), measures the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It ranges from 0 to 1, with higher values indicating a better fit.

```
# Coefficient of determination (R^2)
r_squared <- summary(model)$r.squared
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.9329764
```

Prediction in Linear Models

In a linear model, prediction involves using the fitted model to estimate the value of the dependent variable (response variable) for given values of the independent variables (predictors).The linear model equation is typically of the form:

y = beta_0 + beta_1x_1 + beta_2x_2 +epsilon

1. where (y) is the dependent variable, (x_1, x_2) are the independent
2. variables,(beta_0, beta_1, beta_2) are the coefficients,and (epsilon) is the error term.

Prediction Interval of Mean Response Variable

A prediction interval provides a range within which we expect a new observation to fall, given specific values of the predictors. It accounts for both the uncertainty in the estimated mean response and the variability of the individual responses.

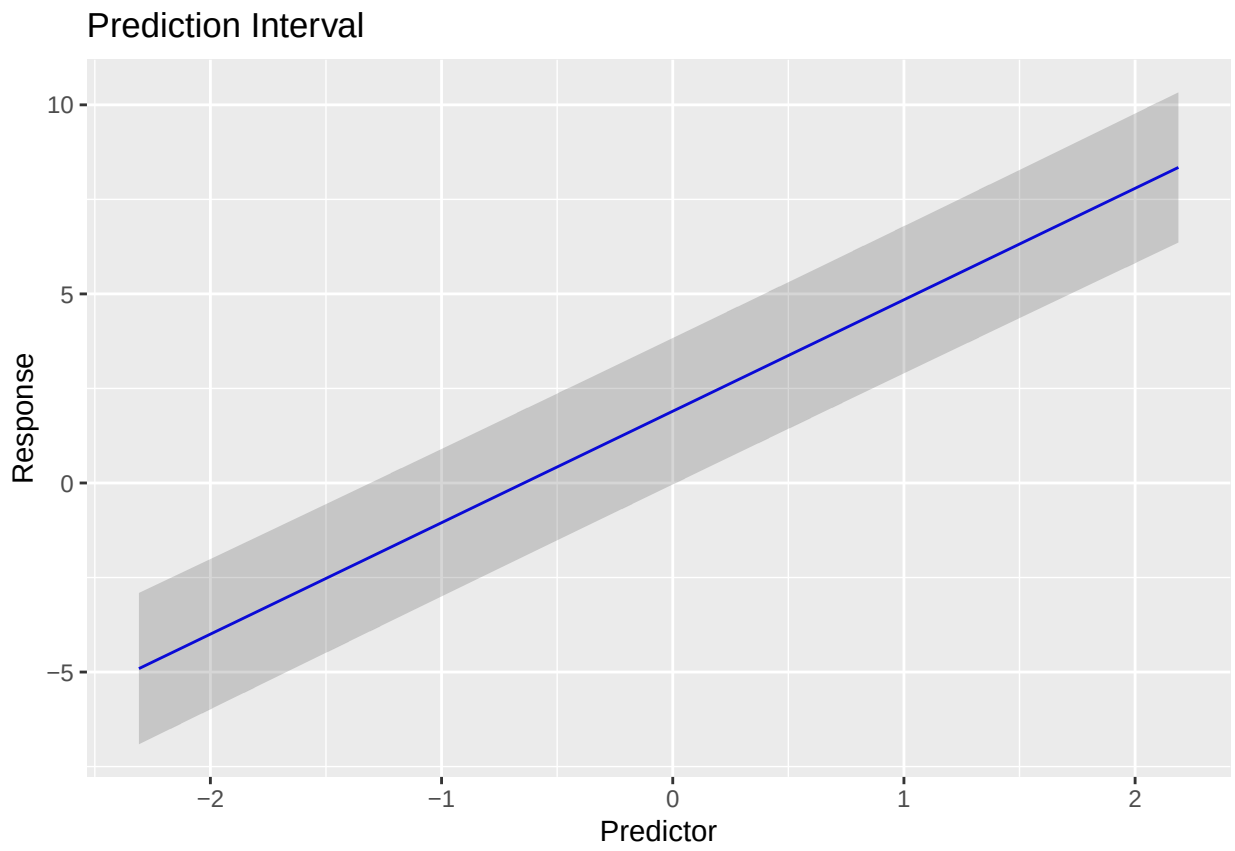
Plotting Prediction Interval

To visualize prediction intervals, we can plot the fitted line along with the intervals around it. Here's an example using R:

```r
# Load necessary library
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```r
# Sample data
set.seed(123)
x <- rnorm(100)
y <- 2 + 3 * x + rnorm(100)

# Fit linear model
model <- lm(y ~ x)

# Create new data for prediction
new_data <- data.frame(x = seq(min(x), max(x), length.out = 100))

# Predict with intervals
predictions <- predict(model, newdata = new_data, interval = "prediction")

# Combine predictions with new data
new_data <- cbind(new_data, predictions)

# Plot
ggplot(new_data, aes(x = x, y = fit)) +
  geom_line(color = "blue") +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.2) +
  labs(title = "Prediction Interval", x = "Predictor", y = "Response")
```



Interpolation and Extrapolation in SLM

Interpolation:Estimating the value of the response variable within the range of the observed data.

```
interpolated_value <- predict(model, newdata = data.frame(x = 0.5))
cat("Interpolated value at x = 0.5:", interpolated_value, "\n")
```

```
## Interpolated value at x = 0.5: 3.370961
```

Extrapolation:Estimating the value of the response variable outside the range of the observed data.

```
extrapolated_value <- predict(model, newdata = data.frame(x = 3))
cat("Extrapolated value at x = 3:", extrapolated_value, "\n")
```

```
## Extrapolated value at x = 3: 10.73978
```

Binary Variable Regression Model

A binary variable regression model, also known as logistic regression,is used when the dependent variable is binary (e.g.0 or 1,Yes or No).The logistic regression model estimates the probability that a given input point belongs to a certain class.

The logistic regression model is given by:

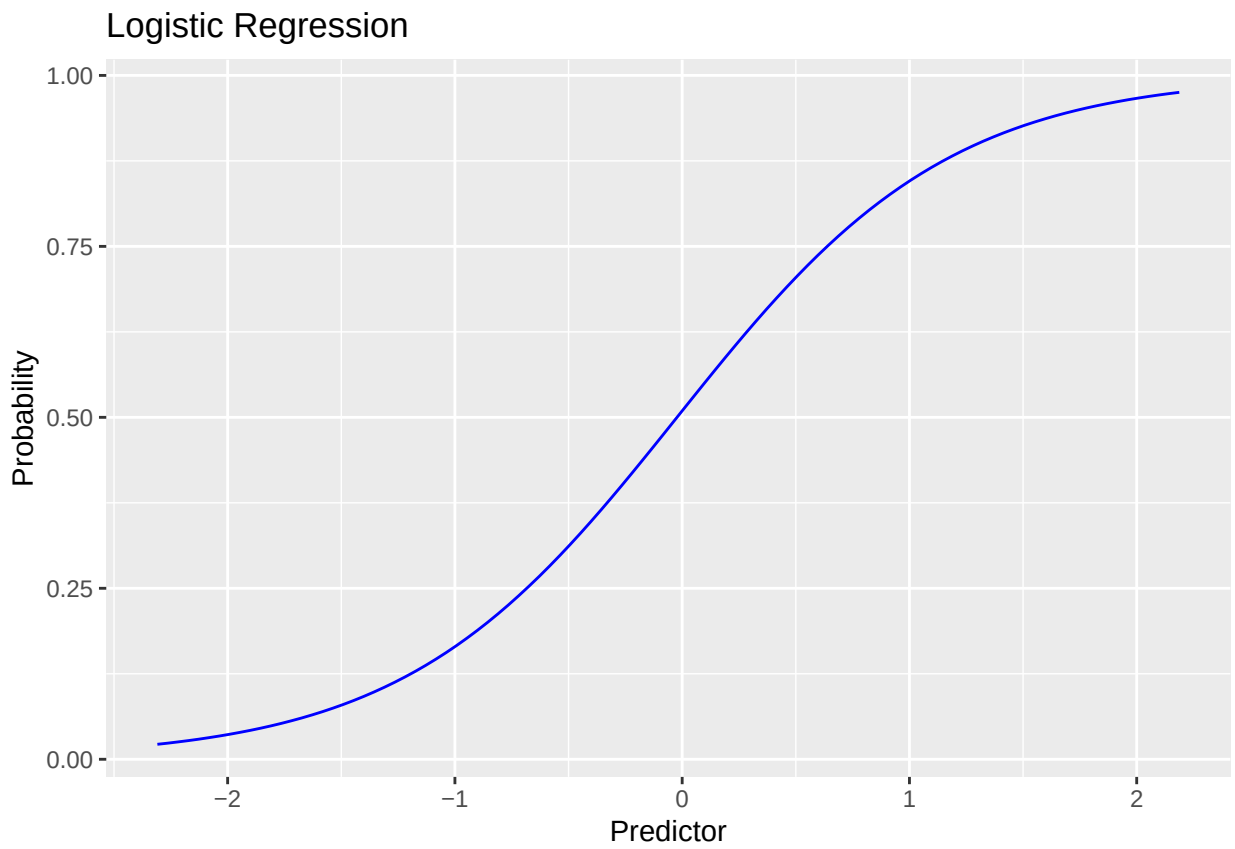
log (p/1-p) = beta_0 + beta_1x_1 + beta_2x_2 + ... + beta_px_p

where (p) is the probability of the event occurring.

```r
# Load necessary library
library(ggplot2)

# Sample data
set.seed(123)
x <- rnorm(100)
y <- ifelse(x + rnorm(100) > 0, 1, 0)

# Fit logistic regression model
model <- glm(y ~ x, family = binomial)

# Summary of the model
summary(model)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.03793    0.23846   0.159    0.874
## x            1.66210    0.36438   4.561 5.08e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.27  on 99  degrees of freedom
## Residual deviance: 104.98  on 98  degrees of freedom
## AIC: 108.98
##
## Number of Fisher Scoring iterations: 4
```

```r
# Predict probabilities
new_data <- data.frame(x = seq(min(x), max(x), length.out = 100))
new_data$prob <- predict(model, newdata = new_data, type = "response")

# Plot
ggplot(new_data, aes(x = x, y = prob)) +
  geom_line(color = "blue") +
  labs(title = "Logistic Regression", x = "Predictor",
       y = "Probability")
```



Multilevel Regression Model

Multi-level regression models, also known as hierarchical linear models,are used when data is nested (e.g.,students within schools, patients within hospitals). These models account for the hierarchical structure of the data.

```
# Load necessary library
library(lme4)
```

```
## Warning: package 'lme4' was built under R version 4.4.2
```

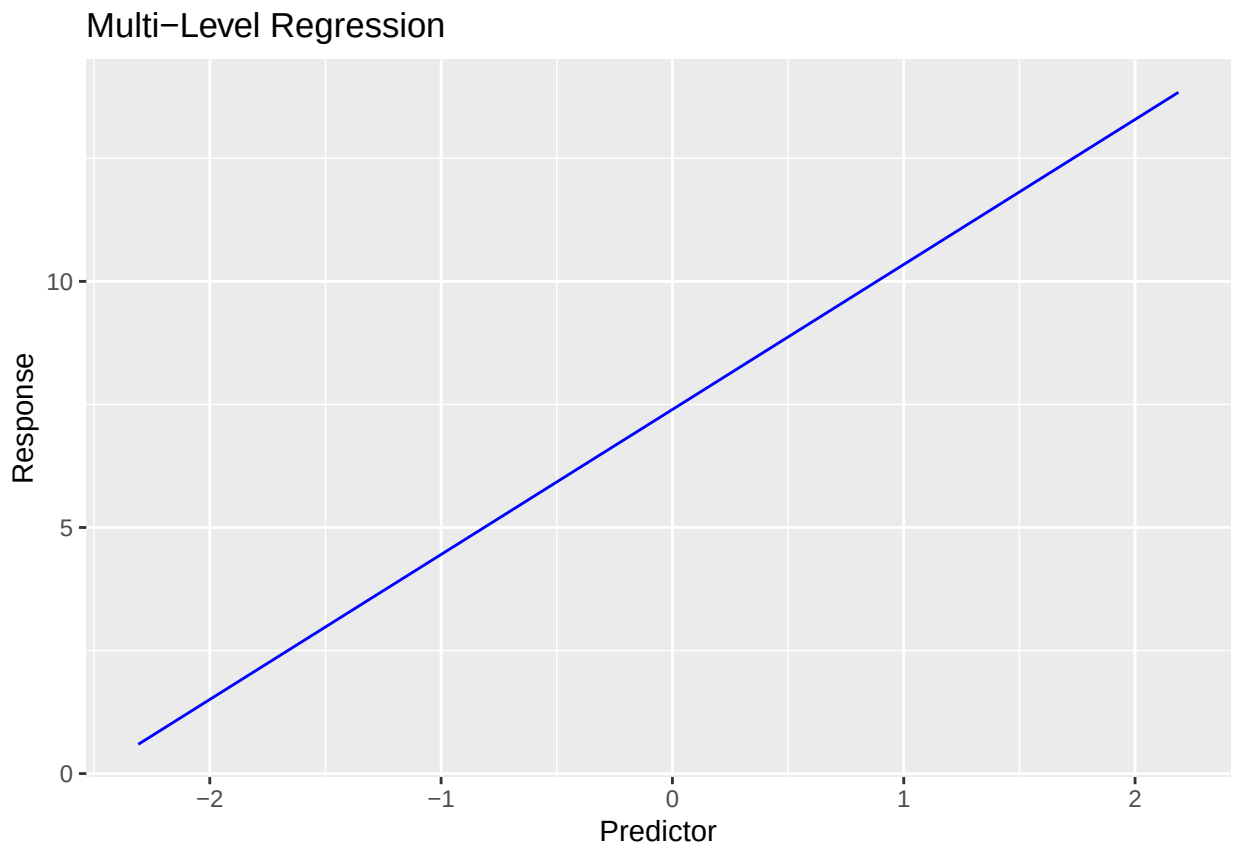
```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.4.2
```

```
# Sample data
set.seed(123)
school <- factor(rep(1:10, each = 10))
student <- rep(1:10, times = 10)
x <- rnorm(100)
y <- 2 + 3 * x + rnorm(100) + as.numeric(school)

# Fit multi-level model
model <- lmer(y ~ x + (1 | school))

# Summary of the model
summary(model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: y ~ x + (1 | school)
##
## REML criterion at convergence: 326.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.09987 -0.67902 -0.03339  0.48124  2.94489
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  school   (Intercept) 9.7737   3.1263
##  Residual             0.9748   0.9873
## Number of obs: 100, groups:  school, 10
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   7.3973     0.9936   7.445
## x             2.9459     0.1139  25.874
##
## Correlation of Fixed Effects:
##   (Intr)
## x -0.010
```

```r
# Predict
new_data <- data.frame(x = seq(min(x), max(x), length.out = 100),
                       school = factor(rep(1, 100)))
new_data$pred <- predict(model, newdata = new_data, re.form = NA)

# Plot
ggplot(new_data, aes(x = x, y = pred)) +
  geom_line(color = "blue") +
  labs(title = "Multi-Level Regression", x = "Predictor",
       y = "Response")
```



Changing the Reference Level of the Regression Model

In regression models with categorical variables,the reference level is the base line category against which other categories are compared.You can change the reference level using the relevel function in R.

```
# Sample data
data <- data.frame(
  y = rnorm(100),
  group = factor(rep(c("A", "B", "C"), length.out = 100))
)

# Change reference level
data$group <- relevel(data$group, ref = "B")

# Fit linear model
model <- lm(y ~ group, data = data)

# Summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = y ~ group, data = data)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.74454 -0.62061 -0.05018  0.73247  2.30507
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01199    0.16158  -0.074   0.9410
## groupA       0.46393    0.22682   2.045   0.0435 *
## groupC      -0.07661    0.22850  -0.335   0.7381
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9282 on 97 degrees of freedom
## Multiple R-squared:  0.06445,    Adjusted R-squared:  0.04516
## F-statistic: 3.341 on 2 and 97 DF,  p-value: 0.03951
```

Treating Categorical Variable as Numeric

Sometimes,categorical variables are treated as numeric,especially when they have an inherent order (ordinal variables).

```r
# Sample data
data <- data.frame(
  y = rnorm(100),
  group = factor(rep(1:3, length.out = 100), ordered = TRUE)
)

# Fit linear model treating group as numeric
model <- lm(y ~ as.numeric(group), data = data)

# Summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = y ~ as.numeric(group), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43117 -0.65552  0.01564  0.72436  2.45655
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.3340     0.2725  -1.225    0.223
## as.numeric(group)   0.1496     0.1267   1.181    0.240
##
## Residual standard error: 1.037 on 98 degrees of freedom
## Multiple R-squared:  0.01404,    Adjusted R-squared:  0.003981
## F-statistic: 1.396 on 1 and 98 DF,  p-value: 0.2403
```

Equivalence with the ANOVA Table in R

ANOVA (Analysis of Variance) is used to compare means across different groups.In R, you can use the anova function to get the ANOVA table for a linear model.

```r
# Sample data
data <- data.frame(
  y = rnorm(100),
  group = factor(rep(c("A", "B", "C"), length.out = 100))
)

# Fit linear model
model <- lm(y ~ group, data = data)

# ANOVA table
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value Pr(>F)
## group      2  0.504 0.25223  0.2538 0.7764
## Residuals 97 96.397 0.99379
```

Multiple Linear Regression Model

Multiple Linear Regression (MLRM) is a statistical technique that models the relationship between a dependent variable and two or more independent variables. The general form of the MLR equation is:

$$\mathbf{Y = 0 + 1X1 + 2X2 + ... + pXp +}$$

1. (Y) is the dependent variable.
2. (X_1, X_2,.....,X_p) are the independent variables.
3. (beta_0) is the intercept.
4. (beta_1,beta_2,.....,beta_p) are the coefficients.
5. (epsilon) is the error term.

Suppose we want to predict the price of house based on its size (in square feet) ,number of bedrooms, and age.The MLR model might look like this:

$$\mathbf{Price = 0 + 1(Size) + 2(Bedrooms) + 3(Age) +}$$

In MLR, you can add more predictors to improve the model.For example,adding the location of the house as a categorical variable:

$$\mathbf{Price = 0 + 1(Size) + 2(Bedrooms) + 3(Age) + 4(Location) +}$$

Plotting the Multiple Linear Regression Model

Plotting an MLR model can be challenging due to multiple dimensions.However, you can use added variable plots (partial regression plot) to visualize the relation between the dependent variable and each predictor while controlling for other predictors.

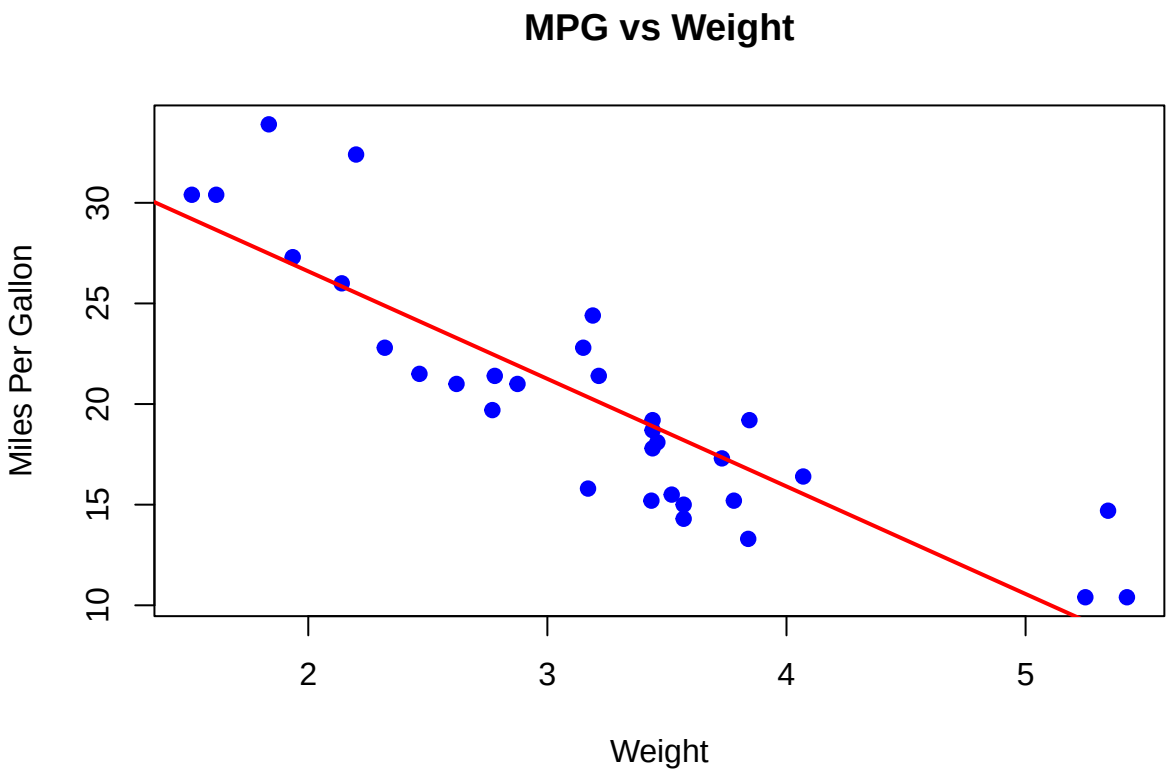
```r
# Load the mtcars dataset
data("mtcars")

# Display the first few rows of the dataset
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
```

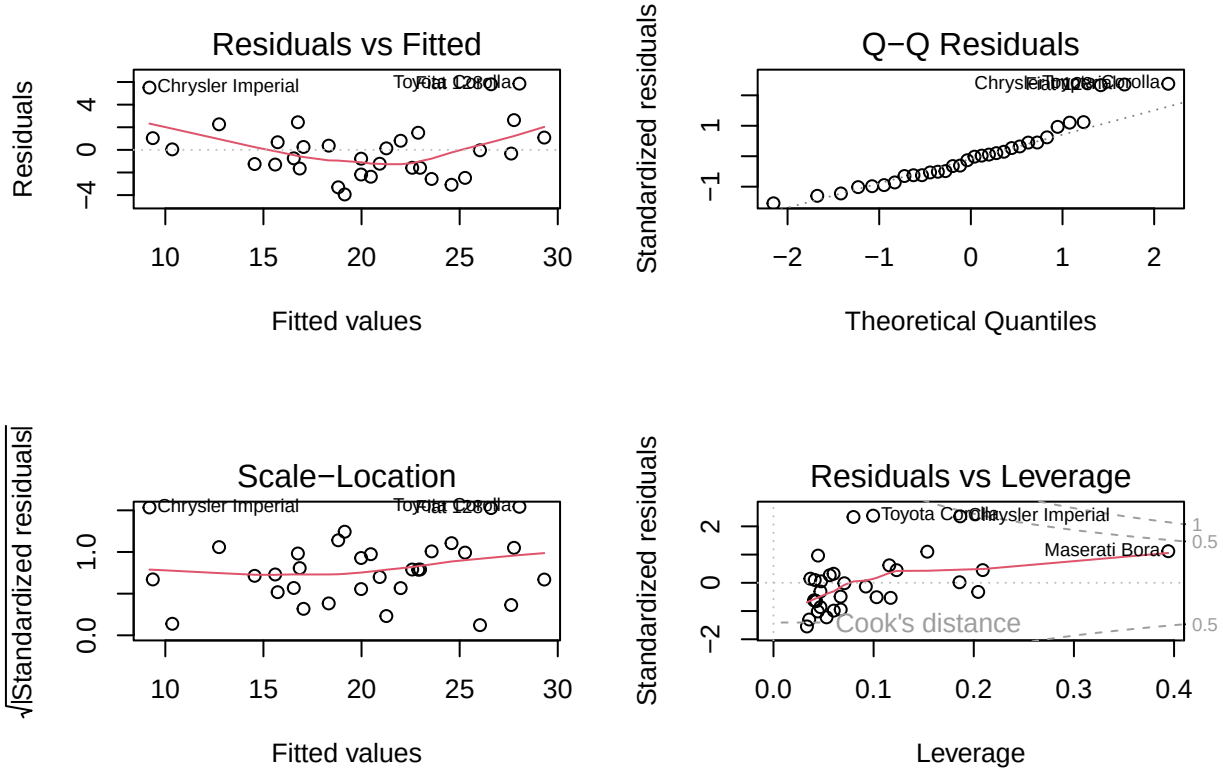
```r
# Create a multiple linear regression model
# Predicting 'mpg' (miles per gallon) based on 'wt' (weight) and 'hp' (horsepower)
model <- lm(mpg ~ wt + hp, data = mtcars)

# Summary of the regression model
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp, data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

```r
# Visualize the relationship (optional)
# Scatter plot of mpg vs wt with regression line
plot(mtcars$wt, mtcars$mpg,
     main = "MPG vs Weight",
     xlab = "Weight",
     ylab = "Miles Per Gallon",
     pch = 19, col = "blue")
abline(lm(mpg ~ wt, data = mtcars), col = "red", lwd = 2)
```




```r
# Check residuals
par(mfrow = c(2, 2)) # Set plot layout
plot(model)          # Diagnostic plots for the model
```



Finding Confidence Intervals for Coefficients

Confidence intervals for the coefficients in an MLR model can be found using the confint function in R:

```
confint(model)
```

```
##                   2.5 %      97.5 %
## (Intercept) 33.95738245 40.49715778
## wt          -5.17191604 -2.58374544
## hp          -0.05024078 -0.01330512
```

Transformation of Numeric Variables

Transformations can help meet the assumptions of linear regression.Common transformations include log,square root, and polynomial transformations.For example,applying a log transformation to the dependent variable:

```
# now we do the transformation of variable in mtcars data set and do multi- linear regression model.Cre

model <- lm(mpg ~ log(wt) + log(hp), data = mtcars)
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ log(wt) + log(hp), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9422 -1.4293 -0.4933  1.1784  4.9148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   58.231      4.439  13.119 1.01e-13 ***
## log(wt)      -11.400      1.729  -6.593 3.17e-07 ***
## log(hp)       -5.193      1.155  -4.495 0.000103 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.084 on 29 degrees of freedom
## Multiple R-squared:  0.8881, Adjusted R-squared:  0.8804
## F-statistic: 115.1 on 2 and 29 DF,  p-value: 1.611e-14
```

Polynomial Transformation

Polynomial regression involves adding polynomial terms to the model to capture non-linear relationships. For example, adding a quadratic term for size:

```r
# now we do the polynomial regression on the mtcars data set.
model_polynomial<-lm(mpg~wt+I(wt^2),data=mtcars)
summary(model_polynomial)
```

```
##
## Call:
## lm(formula = mpg ~ wt + I(wt^2), data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.483 -1.998 -0.773  1.462  6.238
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  49.9308     4.2113  11.856 1.21e-12 ***
## wt          -13.3803     2.5140  -5.322 1.04e-05 ***
## I(wt^2)       1.1711     0.3594   3.258  0.00286 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.651 on 29 degrees of freedom
## Multiple R-squared:  0.8191, Adjusted R-squared:  0.8066
## F-statistic: 65.64 on 2 and 29 DF,  p-value: 1.715e-11
```

MLR with Categorical and Continuous Variables

You can include both categorical and continuous variables in an MLR model.For example, with two categorical and two continuous variables:

```
# now we do the MLR with categorical and continuous variables in the mtcars data set.
model_mlr<-lm(mpg~factor(cyl)+factor(gear)+wt+hp,data=mtcars)
summary(model_mlr)
```

```
##
## Call:
## lm(formula = mpg ~ factor(cyl) + factor(gear) + wt + hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4215 -1.2428 -0.3401  0.8961  5.3657
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   34.46173    2.46379  13.987 2.52e-13 ***
## factor(cyl)6  -2.93920    1.47362  -1.995  0.05710 .
## factor(cyl)8  -1.33080    2.77885  -0.479  0.63617
## factor(gear)4  1.42125    1.51065   0.941  0.35580
## factor(gear)5  2.08577    2.14490   0.972  0.34015
## wt            -2.79186    0.85567  -3.263  0.00318 **
## hp            -0.03424    0.01770  -1.935  0.06444 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.477 on 25 degrees of freedom
## Multiple R-squared:  0.8638, Adjusted R-squared:  0.8311
## F-statistic: 26.42 on 6 and 25 DF,  p-value: 1.128e-09
```

Higher Order Interactions in R

Higher-order interactions involve terms that represent the interaction between two or more predictors. For example, an interaction between size and age:

```
# now we do the higher order interaction in the mtcars data set.
model_interaction<-lm(mpg~wt*hp,data=mtcars)
summary(model_interaction)
```

```
##
## Call:
## lm(formula = mpg ~ wt * hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0632 -1.6491 -0.7362  1.4211  4.5513
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
## wt          -8.21662    1.26971  -6.471 5.20e-07 ***
## hp          -0.12010    0.02470  -4.863 4.04e-05 ***
## wt:hp        0.02785    0.00742   3.753 0.000811 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.153 on 28 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8724
## F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```