STAT452 Final Project

Luu Quoc Bao - 22125008, Le Minh Hoang - 22125029, Le Duc Nhuan - 22125070, Dang Minh Nhut - 22125071

2024-08-18

Task 1

Intial setup

Library

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.1
```

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.4.1
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.4.1
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.4.1
```

```
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.4.1
```

```
##
## Attaching package: 'faraway'
```

```
## The following objects are masked from 'package:car':
##
##     logit, vif
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.1
```

```
## corrplot 0.92 loaded
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.1
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Set seed

```
set.seed(200)
```

Reading and preprocessing data

Load data

```
df <- read.csv("auto_mpg.csv", header = TRUE, sep=";")
dim(df)
```

```
## [1] 398   9
```

Summary of data

```
summary(df)
```

```
##       mgp          cylinders      displacement    horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Length:398
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   Class :character
##  Median :23.00   Median :4.000   Median :148.5   Mode  :character
##  Mean   :23.51   Mean   :5.455   Mean   :193.4
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0
##      weight      acceleration     model_year        origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2224   1st Qu.:13.82   1st Qu.:73.00   1st Qu.:1.000
##  Median :2804   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2970   Mean   :15.57   Mean   :76.01   Mean   :1.573
##  3rd Qu.:3608   3rd Qu.:17.18   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##    car_name
##  Length:398
##  Class :character
##  Mode  :character
##
##
##
```

There are total 398 rows and 9 columns in the data. The columns are “mpg”, “cylinders”, “displacement”, “horsepower”, “weight”, “acceleration”, “model year”, “origin”, and “car name”. The description of each column is as follows:

- “mpg”: (continuous) fuel consumption in miles per gallon,
- “cylinders”: (multi-valued discrete) number of cylinders,
- “displacement”: (continuous) engine size,
- “horsepower”: (continuous) engine power,
- “weight”: (continuous) mass,
- “acceleration”: (continuous) vehicle acceleration,
- “model year”: (multi-valued discrete) model year (last 2 digits)
- “origin”: (multi-valued discrete) place of manufacture: 1 - North American, 2 - Europe, 3 - Asia
- “car name”: (multi-valued discrete) car name

Remove car name

The car name is not useful for our analysis, so we remove it.

```
df$car_name <- NULL
```

Remove rows with missing values

Observing the data, we see that there are some rows with ‘?’ values in the horsepower column. We remove these rows.

```
nrow(df[df$horsepower == '?',]) #6 rows with missing value in horsepower
```

```
## [1] 6
```

```
df <- df[df$horsepower != '?',]
```

Convert horsepower to numeric

```
is.numeric(df$horsepower)
```

```
## [1] FALSE
```

As we can see, the horsepower column is not numeric, so we convert it to numeric.

```
df$horsepower <- as.numeric(df$horsepower)
is.numeric(df$horsepower)
```

```
## [1] TRUE
```

Remove duplicate rows

We count the number of duplicate rows in the data.

```
nrow(df[duplicated(df),])
```

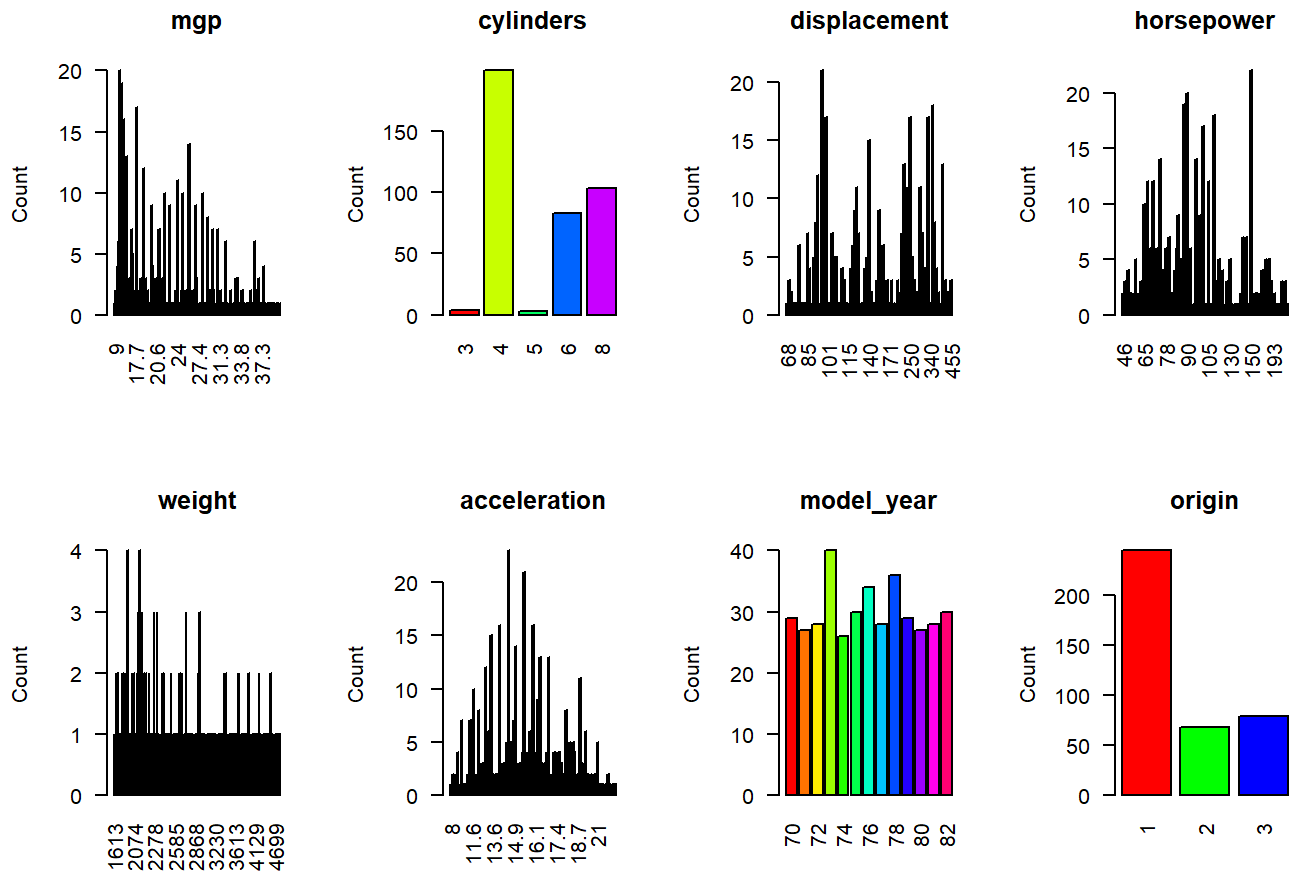
```
## [1] 0
```

There are no duplicate rows in the data. Therefore, we do not need to remove any rows.

Change variables to factor

We first plot barplots of each column to see if there are any columns that should be changed to factors.

```
# Determine the number of columns in df
num_cols <- ncol(df)

# Calculate the number of rows and columns for the plot layout
plot_rows <- 2
plot_cols <- 4

# Set up the plot layout
par(mfrow = c(plot_rows, plot_cols))

# Loop through each column and create a barplot
for (col_name in names(df)) {
  if (col_name == 'mpg') {
    next
  }
  # Get counts for the current column
  col_counts <- table(df[[col_name]])

  # Create a barplot for the current column
  barplot(col_counts,
          main = col_name,
          xlab = "",
          ylab = "Count",
          col = rainbow(length(col_counts)),
          las = 2)  # Rotate x-axis labels if needed
}
```



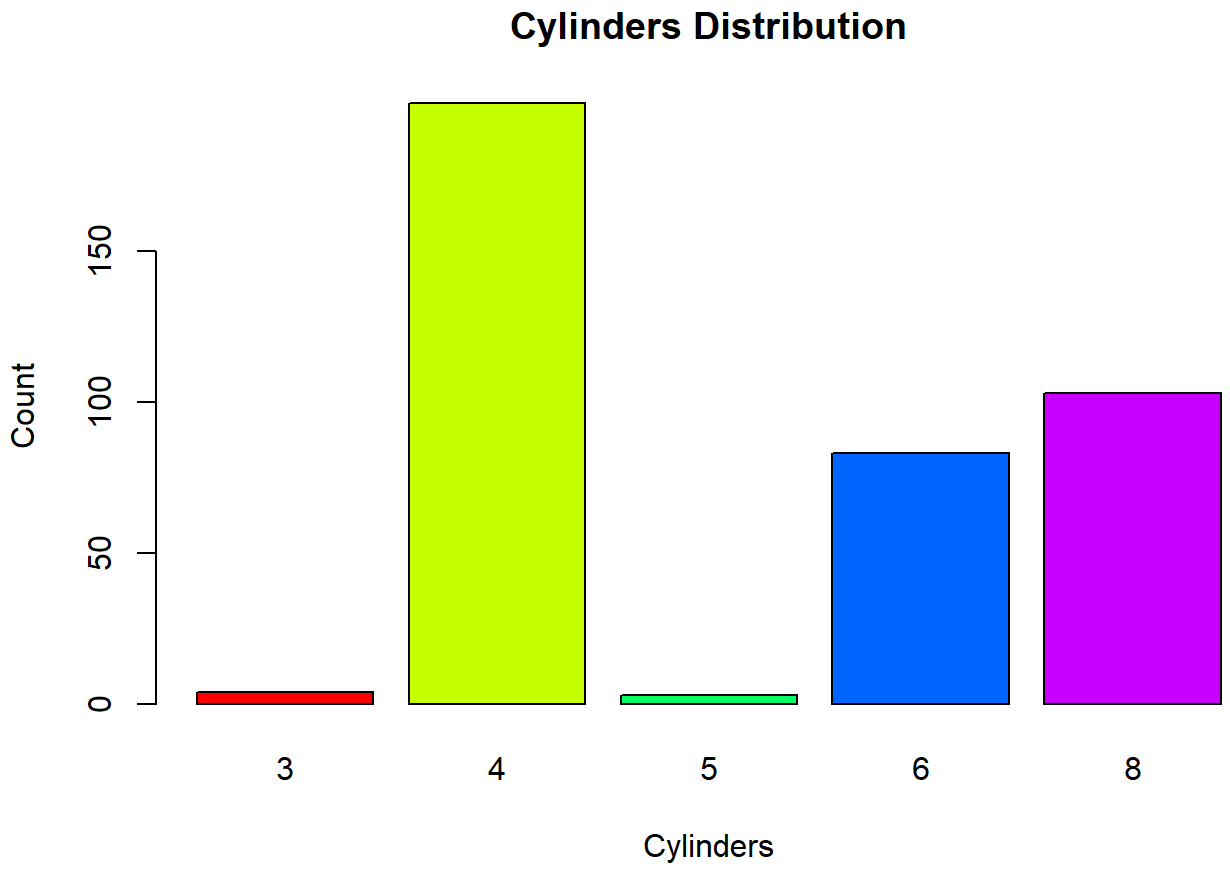

```
# Reset the plot layout
par(mfrow = c(1, 1))
```

The “cylinders” column

```
is.factor(df$cylinders)
```

```
## [1] FALSE
```

```
cylinders_counts <- table(df$cylinders)
barplot(cylinders_counts, main = "Cylinders Distribution", xlab = "Cylinders", ylab = "Count", c
ol = rainbow(length(cylinders_counts)))
```



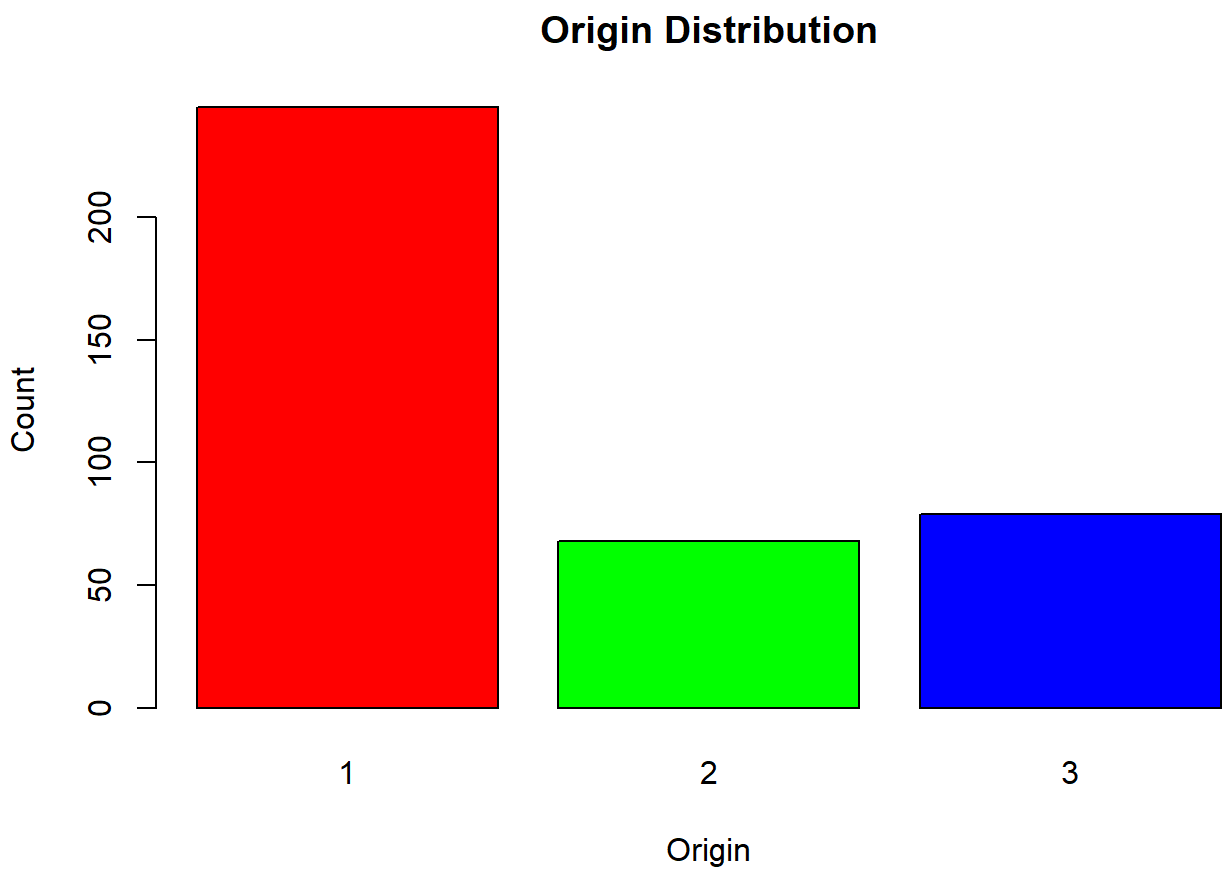
Even though the “cylinders” column has only 5 values, we will not change it to a factor because it is quantitatively meaningful.

The “origin” column

```
is.factor(df$origin)
```

```
## [1] FALSE
```

```
origin_counts <- table(df$origin)
barplot(origin_counts, main = "Origin Distribution", xlab = "Origin", ylab = "Count", col = rain
bow(length(origin_counts)))
```



As we can see, the “origin” column has only 3 values. Moreover, these 3 values are not quantitatively meaningful. Therefore, we will change the “origin” column to a factor.

```
df$origin <- as.factor(df$origin)
```

Data after preprocessing

```
dim(df)
```

```
## [1] 392   8
```

The data now has 392 rows and 8 columns.

Data splitting

We split the data into training and testing sets. The ratio of the training set to the testing set is 80:20.

```
# Define the split ratio
train_ratio <- 0.8

# Determine the number of rows in the training set
train_size <- floor(train_ratio * nrow(df))

# Randomly sample row indices for the training set
train_indices <- sample(seq_len(nrow(df)), size = train_size)

# Split the data into training and testing sets
train_set <- df[train_indices, ]
test_set <- df[-train_indices, ]

# Display the number of rows in each set
dim(train_set) # Should be approximately 80
```

```
## [1] 313   8
```

```
dim(test_set)  # Should be approximately 20
```

```
## [1] 79  8
```

Model fitting

Baseline model

We fit a linear regression model with all original variables to predict “mpg”.

```
baseline <- lm(mgp ~ ., data = train_set)
summary(baseline)
```

```
##
## Call:
## lm(formula = mgp ~ ., data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1028 -2.2080  0.0172  2.0307 13.3297
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.575e+01  5.535e+00  -2.846 0.004727 **
## cylinders    -4.303e-01  3.637e-01  -1.183 0.237645
## displacement  2.417e-02  8.454e-03   2.860 0.004535 **
## horsepower   -2.481e-02  1.550e-02  -1.601 0.110478
## weight       -6.747e-03  7.279e-04  -9.269  < 2e-16 ***
## acceleration  3.113e-02  1.131e-01   0.275 0.783335
## model_year    7.656e-01  6.054e-02  12.647  < 2e-16 ***
## origin2       2.337e+00  6.442e-01   3.628 0.000336 ***
## origin3       2.709e+00  6.234e-01   4.345  1.9e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.376 on 304 degrees of freedom
## Multiple R-squared:  0.8207, Adjusted R-squared:  0.816
## F-statistic: 173.9 on 8 and 304 DF,  p-value: < 2.2e-16
```

To check for multicollinearity, we calculate the Variance Inflation Factor (VIF) of the variables.

```
vif(baseline)
```

```
##    cylinders displacement   horsepower       weight acceleration   model_year
##    10.427154    21.558433     9.945168    10.523347     2.527312     1.348539
##      origin2      origin3
##     1.697476     1.795863
```

We see that the “displacement” variable has a high VIF value. Therefore, we will remove it from the model.

```
baseline<-update(baseline, . ~ . -displacement)
vif(baseline)
```

```
##    cylinders   horsepower       weight acceleration   model_year      origin2
##     6.104170     9.133879     9.027234     2.487504     1.332108     1.483110
##      origin3
##     1.619644
```

Now all variables have VIF values less than 10, indicating that there is no strong multicollinearity in the model.

We apply Stepwise Algorithm to select the best model.

```
step_baseline<-step(baseline, direction = "both")
```

```
## Start:  AIC=776.81
## mgp ~ cylinders + horsepower + weight + acceleration + model_year +
##     origin
##
##                Df Sum of Sq    RSS    AIC
## - acceleration  1      0.08 3557.9 774.81
## - horsepower    1      7.63 3565.4 775.48
## - cylinders     1      8.43 3566.2 775.55
## <none>                      3557.8 776.81
## - origin        2    170.89 3728.7 787.49
## - weight        1    891.22 4449.0 844.77
## - model_year    1   1754.45 5312.2 900.28
##
## Step:  AIC=774.81
## mgp ~ cylinders + horsepower + weight + model_year + origin
##
##                Df Sum of Sq    RSS    AIC
## - cylinders     1      8.78 3566.6 773.58
## - horsepower    1     12.10 3570.0 773.88
## <none>                      3557.9 774.81
## + acceleration  1      0.08 3557.8 776.81
## - origin        2    170.99 3728.8 785.50
## - weight        1   1138.25 4696.1 859.69
## - model_year    1   1783.77 5341.6 900.01
##
## Step:  AIC=773.58
## mgp ~ horsepower + weight + model_year + origin
##
##                Df Sum of Sq    RSS    AIC
## - horsepower    1      7.74 3574.4 772.26
## <none>                      3566.6 773.58
## + cylinders     1      8.78 3557.9 774.81
## + acceleration  1      0.43 3566.2 775.55
## - origin        2    162.21 3728.9 783.51
## - weight        1   1475.82 5042.5 879.97
## - model_year    1   1775.17 5341.8 898.02
##
## Step:  AIC=772.26
## mgp ~ weight + model_year + origin
##
##                Df Sum of Sq    RSS     AIC
## <none>                      3574.4  772.26
## + horsepower    1       7.7 3566.6  773.58
## + cylinders     1       4.4 3570.0  773.88
## + acceleration  1       2.2 3572.2  774.07
## - origin        2     156.9 3731.2  781.71
## - model_year    1    2099.6 5674.0  914.90
## - weight        1    4799.9 8374.3 1036.74
```

```
summary(step_baseline)
```

```
##
## Call:
## lm(formula = mgp ~ weight + model_year + origin, data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6999 -2.2907  0.0147  1.8108 13.4983
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.693e+01  4.664e+00  -3.630 0.000331 ***
## weight      -6.034e-03  2.967e-04 -20.337  < 2e-16 ***
## model_year   7.596e-01  5.647e-02  13.451  < 2e-16 ***
## origin2      1.560e+00  5.855e-01   2.665 0.008106 **
## origin3      1.997e+00  5.827e-01   3.427 0.000694 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.407 on 308 degrees of freedom
## Multiple R-squared:  0.815,  Adjusted R-squared:  0.8126
## F-statistic: 339.2 on 4 and 308 DF,  p-value: < 2.2e-16
```

To validate that the removed variables are not significant, we perform an ANOVA test.

```
anova(step_baseline, baseline)
```

```
## Analysis of Variance Table
##
## Model 1: mgp ~ weight + model_year + origin
## Model 2: mgp ~ cylinders + horsepower + weight + acceleration + model_year +
##     origin
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    308 3574.4
## 2    305 3557.8  3    16.607 0.4746 0.7002
```

The p-value of the ANOVA test is much higher than the significance level of 0.05, indicating that the removed variables are not significant.

We check for the multicolinearity of the variables in the model by applying the VIF test.

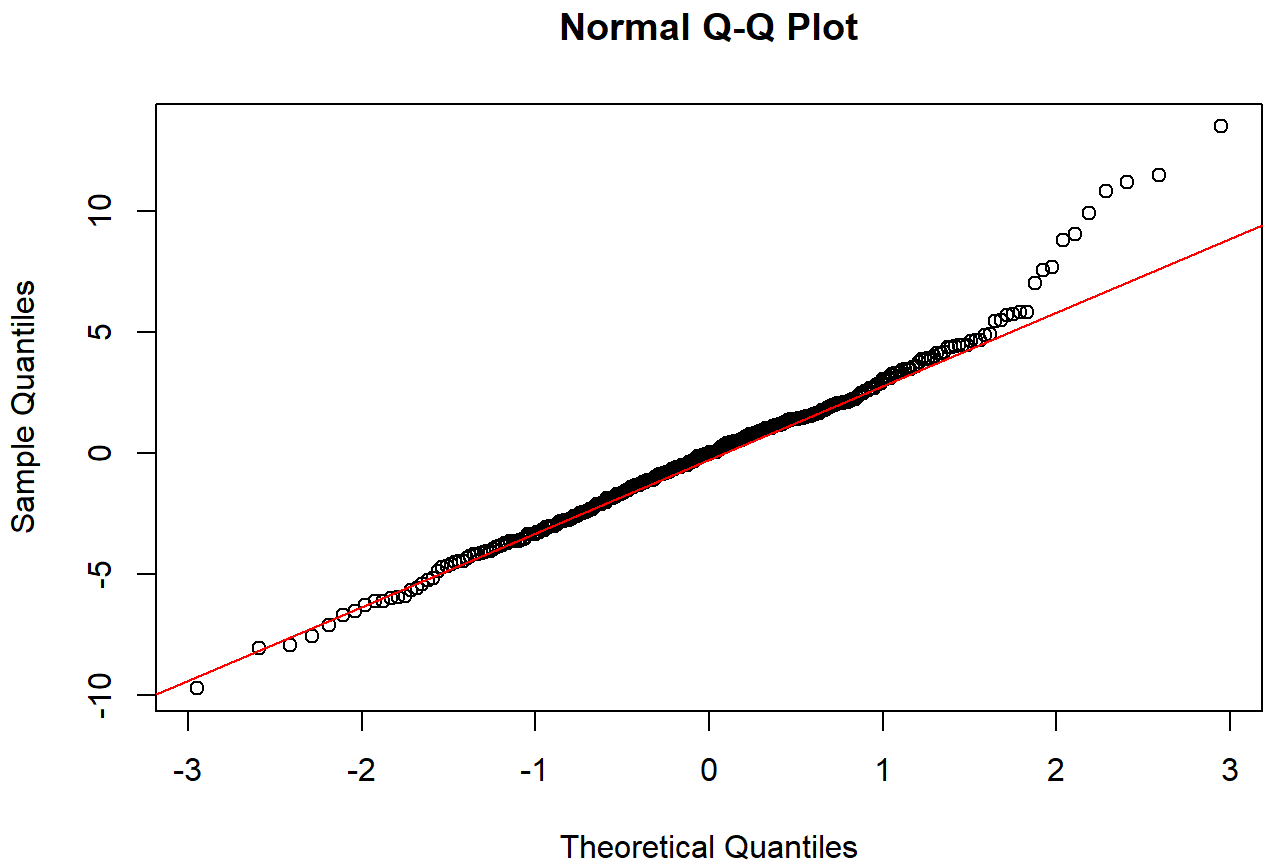
```
vif(step_baseline)
```

```
##     weight model_year    origin2    origin3
##   1.716598   1.152584   1.377340   1.540846
```

Compared to the previous model, the VIF values of the variables are much lower, indicating that there is no multicollinearity in the model.

We now plot the Q-Q plot of the residuals to check for normality.

```
qqnorm(step_baseline$residuals)
qqline(step_baseline$residuals, col = "red")
```



The residuals are close to the line in the middle, suggesting that the residuals in this range are approximately normally distributed. However, the points at the ends (tails) of the plot deviate from the red line, especially on the right side where the points are higher than the line. This is potentially due to the presence of outliers in the data, which can affect the normality of the residuals.

To test the normality of the residuals, we perform the Shapiro-Wilk test.

```
shapiro.test(step_baseline$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  step_baseline$residuals
## W = 0.97942, p-value = 0.0001818
```

The p-value of the Shapiro-Wilk test is less than 0.05, indicating that the residuals are not normally distributed. This is consistent with the Q-Q plot, where the residuals have heavier tails than a normal distribution.

We also perform the Breusch-Pagan test and Durbin-Watson test to check for homoscedasticity and autocorrelation, respectively.

```
lmtest::bptest(step_baseline)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  step_baseline
## BP = 20.345, df = 4, p-value = 0.0004269
```

```
lmtest::dwtest(step_baseline, alternative="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  step_baseline
## DW = 1.8988, p-value = 0.3679
## alternative hypothesis: true autocorrelation is not 0
```

The p-value of the Breusch-Pagan test is less than 0.05, indicating that the residuals are heteroscedastic. The Durbin-Watson test gives a DW value close to 2, indicating that there is no autocorrelation.

In conclusion, the model is not ideal as the residuals are not normally distributed and heteroscedastic. However, the model is still acceptable as the residuals are approximately normally distributed in the middle range and there is no autocorrelation in the residuals.

Testing on the test set, we get the following results.

```
predictions <- predict(step_baseline, newdata = test_set)
# Actual values from the test set
actual_values <- test_set$mgp
# Calculate Mean Squared Error (MSE)
mse <- mean((predictions - actual_values)^2)
# Calculate R-squared
rss <- sum((predictions - actual_values)^2)
tss <- sum((actual_values - mean(actual_values))^2)
r_squared <- 1 - (rss / tss)
# Print metrics cat("Mean Squared Error (MSE):", mse, "\n")
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.8267948
```

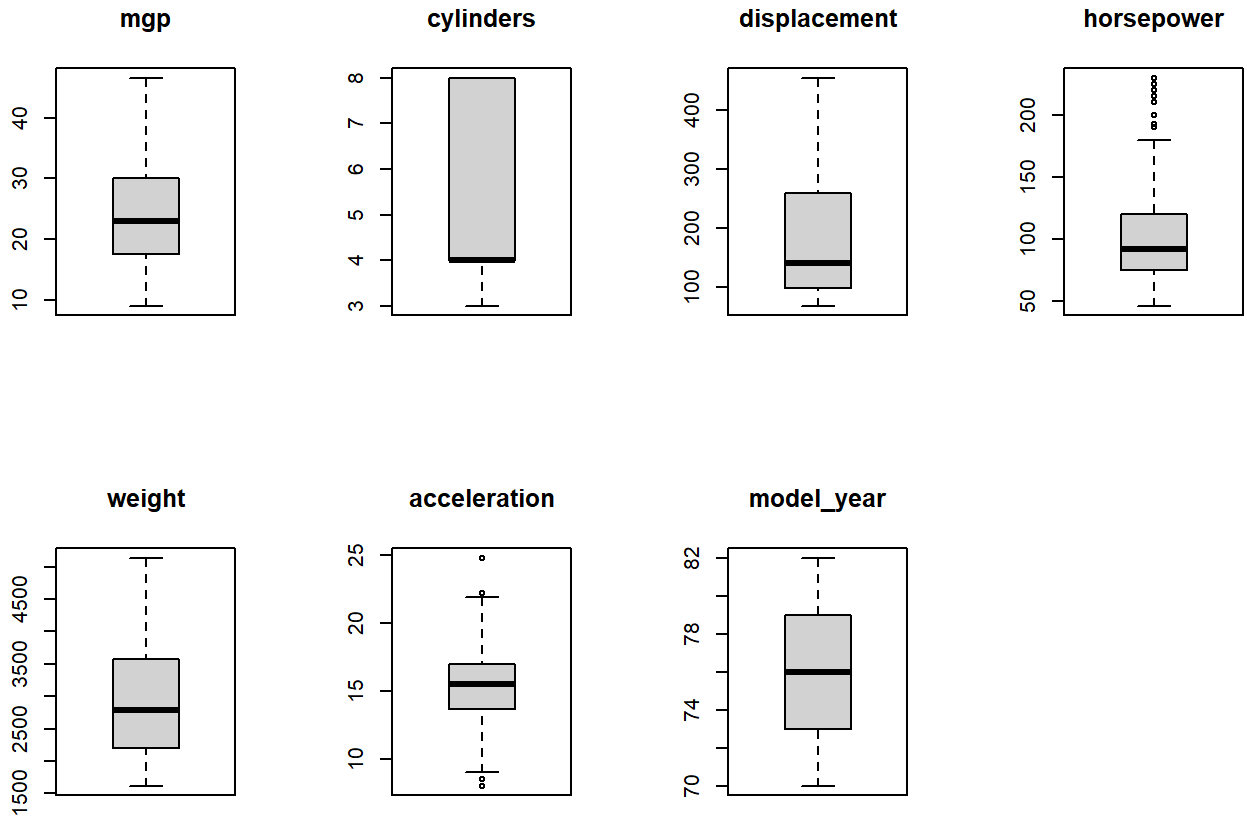
The R-squared is approximately 0.82, indicating that the model explains 82% of the variance in the data. This is a good result, suggesting that the model is effective in predicting “mpg”. However, we believe that the absence of “horsepower” in the model is contradictory to the real-world relationship between “mpg” and “horsepower”, as the more powerful the engine, the higher the fuel consumption. This suggests that the model can be improved by including “horsepower” in the model.

Improving the model

We investigate the relationship between “mpg” and other variables.

First, we plot boxplots of all variables.

```
par(mfrow = c(2, 4))

for (col in names(train_set)) {
  if (is.numeric(train_set[[col]]))
    boxplot(train_set[col], main = col)
}

par(mfrow = c(1, 1))
```



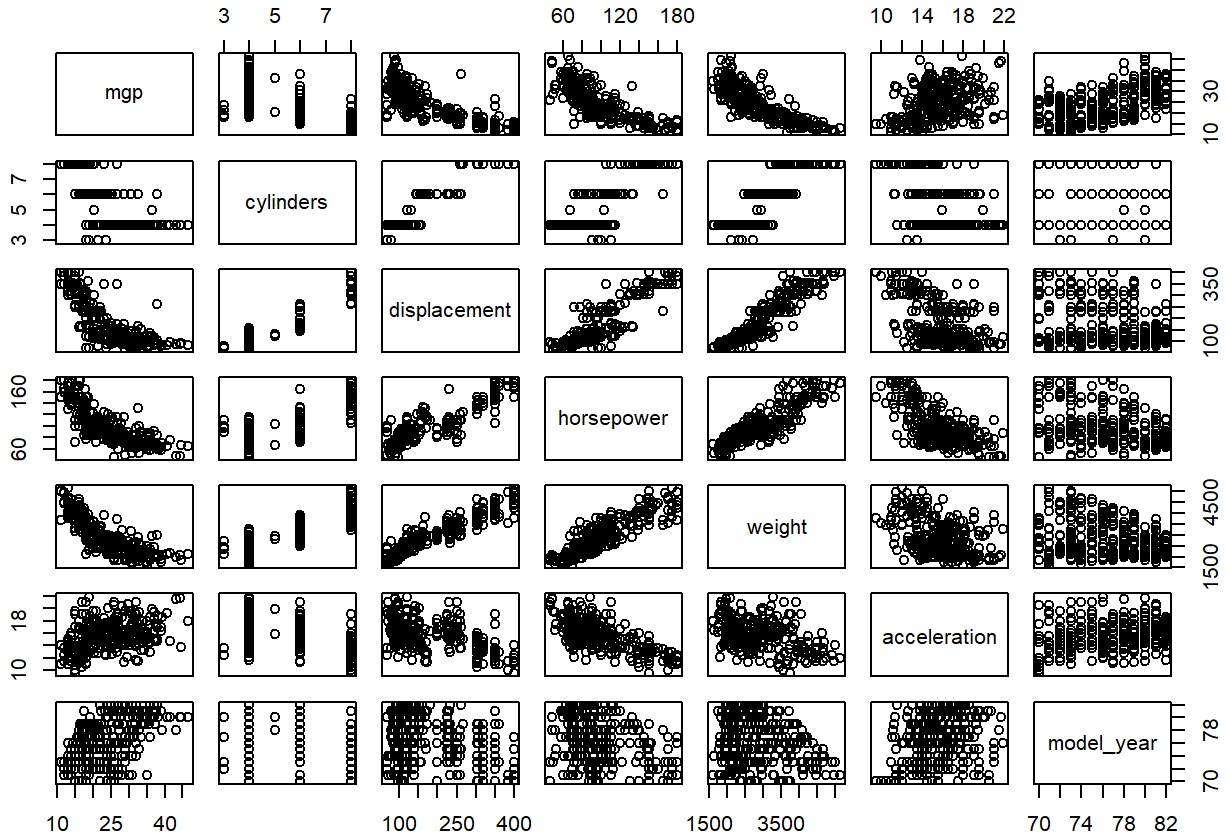
We can observe that “horsepower” has many outliers. Therefore, we will remove these outliers using the Interquartile Range (IQR) method.

```
# Function to remove outliers using IQR for a single column
identify_outliers_IQR <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  !(x >= lower_bound & x <= upper_bound)
}

# Identify outliers for each numeric column
outlier_matrix <- train_set %>%
  select(where(is.numeric)) %>%
  mutate(across(everything(), identify_outliers_IQR))

# Filter out rows with any outlier
train_set <- train_set[!apply(outlier_matrix, 1, any), ]

dim(train_set)
```

```
## [1] 296   8
```

To further investigate the relationship between “mpg” and other variables, we plot scatter plots of “mpg” and other variables.

```
pairs(subset(train_set, select = - c(origin)))
```

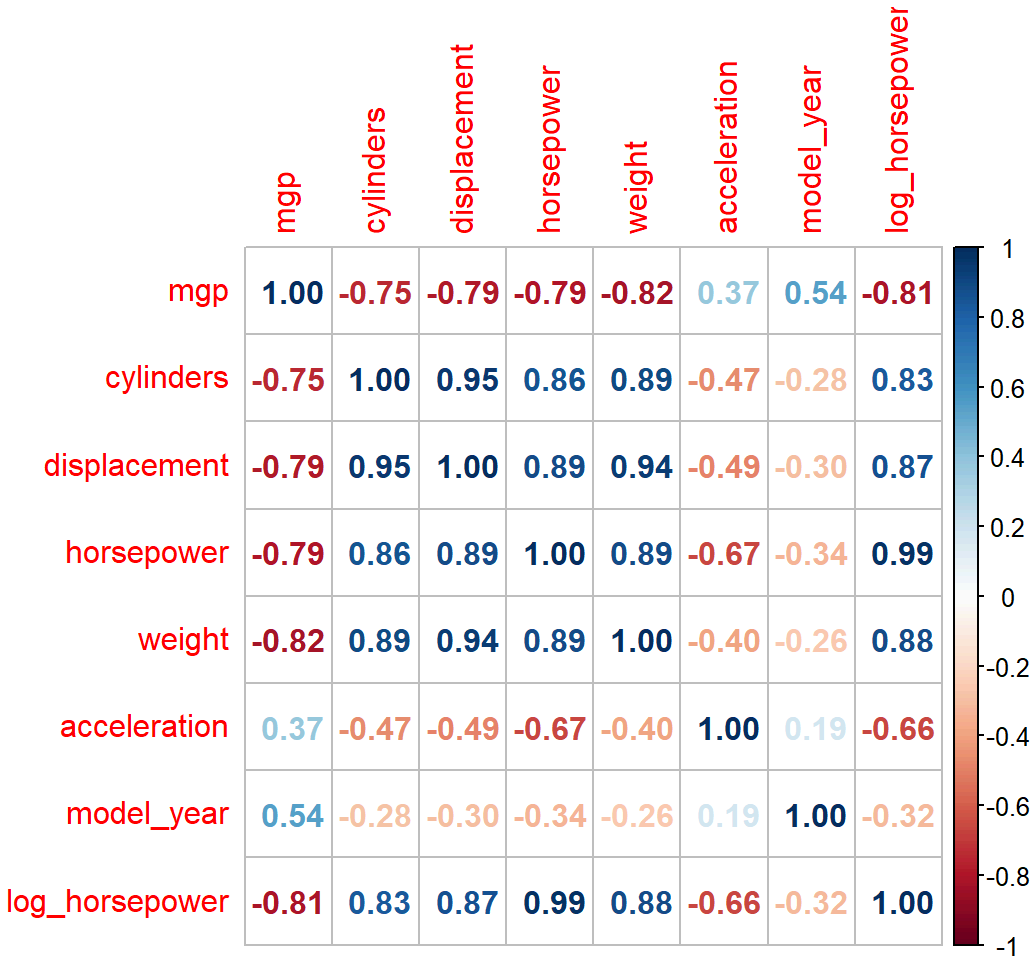


We can see a curvilinear relationship between “mpg” and “horsepower”. Therefore, we will include a log transformation of “horsepower” in the data. Not only aligning with the relationship between “mpg” and “horsepower”, the log transformation also helps to reduce the effect of outliers in the “horsepower” column.

```
train_set$log_horsepower <- log(train_set$horsepower)
```

Moreover, the scatter plots of “displacment”, “weight” with respect to “mpg” has similar pattern to that of “horsepower”. To validate our assumption, we use correlation matrix to see the relationship between the variables.

```
corrplot(cor(subset(train_set, select = -origin)), method = "number")
```



As “horsepower”, “displacement”, and “weight” are highly correlated, we will include the log transformation of “displacement” and “weight” in the data.

```
train_set$log_displacement <- log(train_set$displacement)
train_set$log_weight <- log(train_set$weight)
```

In reality, the work done by the engine is the product of the force applied to the car and the distance the car moves. The force applied to the car is the product of the mass of the car and the acceleration ($F = m \times a$, where F is the force, m is the mass, and a is the acceleration). Therefore, we will include the interaction between “weight” and “acceleration” in the data.

Moreover, the fuel consumption from the 1970s to the 1980s has a quadratic increase. Therefore, we will include the square of time from “model year” to 1970 in the data.

Our new model is as follows:

```
new_model <- lm(mgp ~ . + I(weight * acceleration) + I((model_year-70)^2), data = train_set)
summary(new_model)
```

```
##
## Call:
## lm(formula = mgp ~ . + I(weight * acceleration) + I((model_year -
##     70)^2), data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5206 -1.6561 -0.0187  1.3647 12.1445
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.543e+02  5.323e+01   4.778 2.85e-06 ***
## cylinders                  1.527e-01  3.698e-01   0.413   0.6800
## displacement               6.968e-03  1.824e-02   0.382   0.7027
## horsepower                 2.109e-02  6.251e-02   0.337   0.7361
## weight                     5.901e-03  3.792e-03   1.556   0.1208
## acceleration               7.764e-02  4.307e-01   0.180   0.8571
## model_year                -7.233e-02  2.001e-01  -0.361   0.7180
## origin2                    1.100e+00  6.198e-01   1.775   0.0770 .
## origin3                    1.041e+00  6.067e-01   1.715   0.0874 .
## log_horsepower            -1.036e+01  6.132e+00  -1.689   0.0922 .
## log_displacement          -2.196e+00  3.396e+00  -0.647   0.5184
## log_weight                -2.368e+01  8.747e+00  -2.707   0.0072 **
## I(weight * acceleration) -1.144e-04  1.521e-04  -0.752   0.4526
## I((model_year - 70)^2)     7.044e-02  1.532e-02   4.597 6.46e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.922 on 282 degrees of freedom
## Multiple R-squared:  0.8624, Adjusted R-squared:  0.8561
## F-statistic:   136 on 13 and 282 DF,  p-value: < 2.2e-16
```

To check for multicollinearity, we calculate the Variance Inflation Factor (VIF) of the variables. As the VIF values of many variables are high, we progressively remove the variable with the highest VIF value, and repeat calculating the VIF values until all VIF values are less than 10.

```
vif(new_model)
```

```
##               cylinders            displacement              horsepower
##               12.659915              107.814578              139.521941
##                  weight            acceleration              model_year
##              329.672968               37.189358               17.637871
##                 origin2                 origin3          log_horsepower
##                2.042741                2.233911              124.086112
##         log_displacement              log_weight I(weight * acceleration)
##              105.269530              197.930606              106.333785
##   I((model_year - 70)^2)
##               17.128472
```

```
new_model <- update(new_model, . ~ . -weight)
vif(new_model)
```

```
##               cylinders            displacement              horsepower
##               12.641865               77.515031               96.153799
##            acceleration              model_year                 origin2
##               21.401736               17.560187                2.011906
##                 origin3          log_horsepower        log_displacement
##                2.233903               86.835928               75.401528
##              log_weight I(weight * acceleration)   I((model_year - 70)^2)
##               93.011727               57.144250               17.120187
```

```
new_model <- update(new_model, . ~ . -horsepower)
vif(new_model)
```

```
##               cylinders            displacement            acceleration
##               12.296124               40.147740               19.272366
##              model_year                 origin2                 origin3
##               17.550021                1.985936                2.233727
##          log_horsepower        log_displacement              log_weight
##               12.701638               44.553691               85.506804
## I(weight * acceleration)   I((model_year - 70)^2)
##               50.546367               17.120006
```

```
new_model <- update(new_model, . ~ . -log_weight)
vif(new_model)
```

```
##               cylinders            displacement            acceleration
##               12.077993               39.928385                9.545467
##              model_year                 origin2                 origin3
##               17.535179                1.960817                2.232296
##          log_horsepower        log_displacement I(weight * acceleration)
##               10.164391               42.217688               10.623918
##   I((model_year - 70)^2)
##               17.078197
```

```
new_model <- update(new_model, . ~ . -log_displacement)
vif(new_model)
```

```
##               cylinders            displacement            acceleration
##               11.864052               21.237471                9.398750
##              model_year                 origin2                 origin3
##               17.533619                1.719686                1.791455
##          log_horsepower I(weight * acceleration)   I((model_year - 70)^2)
##                9.974336                9.764120               17.041498
```

```
new_model <- update(new_model, . ~ . -displacement)
vif(new_model)
```

```
##                cylinders             acceleration                model_year
##                 4.917148                 8.304136                16.556143
##                  origin2                  origin3           log_horsepower
##                 1.499229                 1.652089                 9.965083
## I(weight * acceleration)   I((model_year - 70)^2)
##                 7.992294                16.436721
```

```
new_model <- update(new_model, . ~ . -model_year)
vif(new_model)
```

```
##                cylinders             acceleration                  origin2
##                 4.897735                 8.098146                 1.494226
##                  origin3           log_horsepower I(weight * acceleration)
##                 1.642312                 9.692959                 7.526612
##   I((model_year - 70)^2)
##                 1.244208
```

We now apply Stepwise Algorithm to select the best model.

```
stepwise_new_model <- step(new_model, direction = "both")
```

```
## Start:  AIC=661.98
## mgp ~ cylinders + acceleration + origin + log_horsepower + I(weight *
##     acceleration) + I((model_year - 70)^2)
##
##                            Df Sum of Sq    RSS    AIC
## <none>                                  2624.7 661.98
## - acceleration              1     18.99 2643.7 662.11
## - cylinders                 1     20.15 2644.8 662.24
## - origin                    2    145.79 2770.5 673.98
## - I(weight * acceleration)  1    297.32 2922.0 691.74
## - log_horsepower            1    365.90 2990.6 698.61
## - I((model_year - 70)^2)    1   1872.07 4496.8 819.34
```

```
summary(stepwise_new_model)
```

```
##
## Call:
## lm(formula = mgp ~ cylinders + acceleration + origin + log_horsepower +
##     I(weight * acceleration) + I((model_year - 70)^2), data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6909 -1.7676 -0.0663  1.6709 12.2148
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                7.537e+01  9.504e+00   7.930 4.87e-14 ***
## cylinders                  3.533e-01  2.376e-01   1.487 0.138142
## acceleration               2.997e-01  2.076e-01   1.444 0.149965
## origin2                    1.656e+00  5.476e-01   3.025 0.002714 **
## origin3                    1.971e+00  5.374e-01   3.667 0.000292 ***
## log_horsepower            -1.122e+01  1.770e+00  -6.336 9.04e-10 ***
## I(weight * acceleration) -2.387e-04  4.180e-05  -5.712 2.79e-08 ***
## I((model_year - 70)^2)     6.114e-02  4.266e-03  14.332  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.019 on 288 degrees of freedom
## Multiple R-squared:  0.8501, Adjusted R-squared:  0.8464
## F-statistic: 233.3 on 7 and 288 DF,  p-value: < 2.2e-16
```

The Stepwise Algorithm selects the same model as the one we manually selected, suggesting that no other variables should be excluded from the model.

As the p-values of “cylinders” and “acceleration” are higher than the significance level of 0.05, we suspect that these variables are not significant. To validate this, we perform an ANOVA test.

```
anova(stepwise_new_model, update(stepwise_new_model , . ~ . -cylinders -acceleration))
```

```
## Analysis of Variance Table
##
## Model 1: mgp ~ cylinders + acceleration + origin + log_horsepower + I(weight *
##     acceleration) + I((model_year - 70)^2)
## Model 2: mgp ~ origin + log_horsepower + I(weight * acceleration) + I((model_year -
##     70)^2)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    288 2624.7
## 2    290 2654.2 -2     -29.5 1.6185    0.2
```

The p-value of the ANOVA test is much higher than 0.05, indicating that the removed variables are not significant.

```
final_model <- update(stepwise_new_model, . ~ . -cylinders -acceleration)
summary(final_model)
```

```
##
## Call:
## lm(formula = mgp ~ origin + log_horsepower + I(weight * acceleration) +
##     I((model_year - 70)^2), data = train_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9949 -1.7673 -0.0892  1.6506 12.3661
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                8.515e+01  3.361e+00  25.334  < 2e-16 ***
## origin2                    1.492e+00  5.291e-01   2.820 0.005136 **
## origin3                    2.017e+00  5.170e-01   3.901 0.000119 ***
## log_horsepower            -1.249e+01  7.420e-01 -16.835  < 2e-16 ***
## I(weight * acceleration) -1.782e-04  1.899e-05  -9.386  < 2e-16 ***
## I((model_year - 70)^2)     5.950e-02  4.172e-03  14.262  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.025 on 290 degrees of freedom
## Multiple R-squared:  0.8484, Adjusted R-squared:  0.8458
## F-statistic: 324.5 on 5 and 290 DF,  p-value: < 2.2e-16
```

To check for multicollinearity of our final model, we calculate the Variance Inflation Factor (VIF) of the variables.

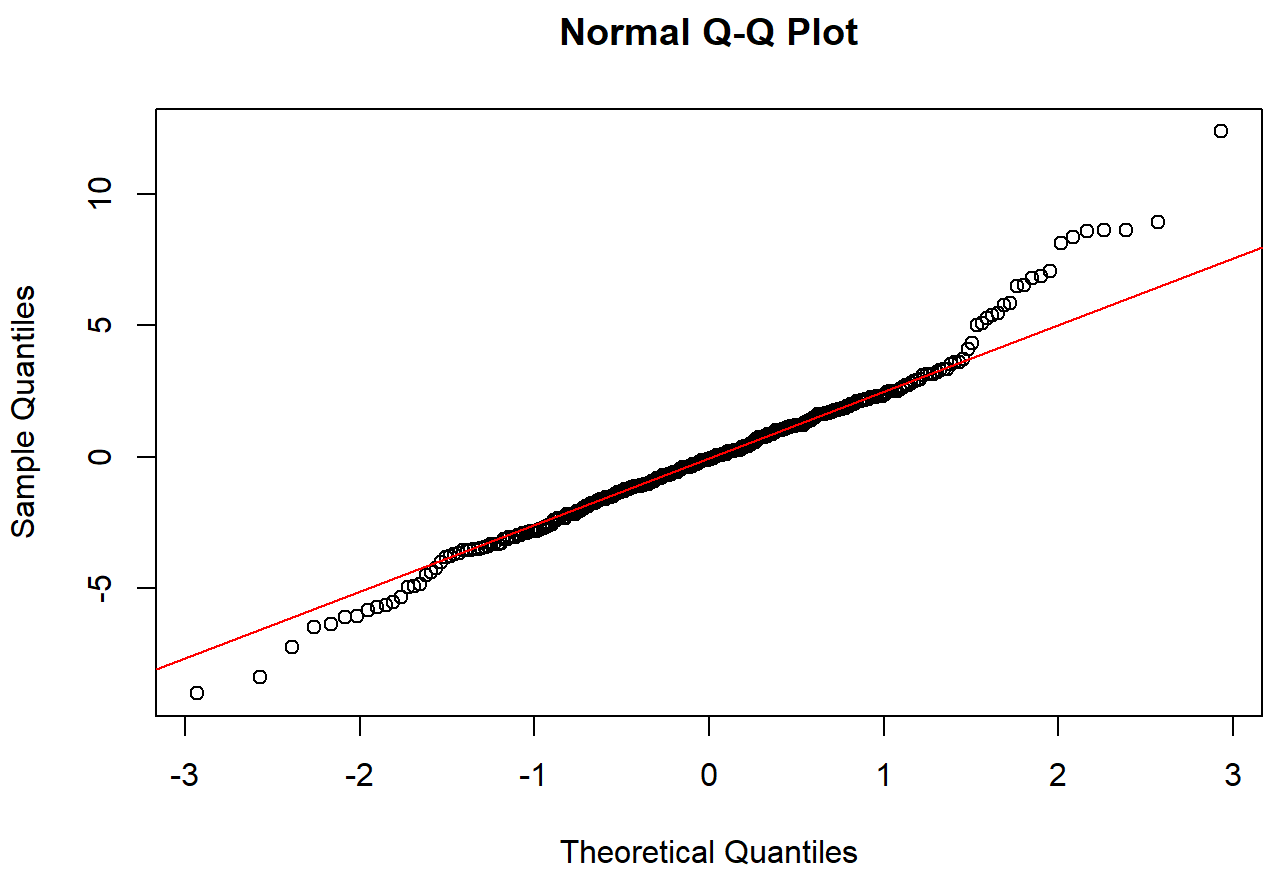
```
vif(final_model)
```

```
##                  origin2                  origin3           log_horsepower
##                 1.388687                 1.513624                 1.695485
## I(weight * acceleration)   I((model_year - 70)^2)
##                 1.547234                 1.184841
```

All VIF values are now much lower than 10, indicating that there is no multicollinearity in the model.

We now plot the Q-Q plot of the residuals to check for normality.

```
qqnorm(final_model$residuals)
qqline(final_model$residuals, col = "red")
```



The Q-Q plot shows that most points lie close to the line in the center, but there are deviations at both ends (tails). This suggests that while the residuals are roughly normally distributed in the middle range, there are issues in the tails. This suggests that the normality assumption may not be fully satisfied.

We perform the Shapiro-Wilk test to check for normality of the residuals.

```
shapiro.test(final_model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  final_model$residuals
## W = 0.97361, p-value = 2.877e-05
```

The p-value of the Shapiro-Wilk test is less than 0.05, indicating that the residuals are not normally distributed. This once again aligns with the Q-Q plot, where the residuals have heavier tails than a normal distribution.

We also perform the Breusch-Pagan test and Durbin-Watson test to check for homoscedasticity and autocorrelation, respectively.

```
lmtest::bptest(final_model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  final_model
## BP = 23.484, df = 5, p-value = 0.0002728
```

```
lmtest::dwtest(final_model, alternative="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  final_model
## DW = 1.9581, p-value = 0.7139
## alternative hypothesis: true autocorrelation is not 0
```

The p-value of the Breusch-Pagan test is less than 0.05, indicating that the residuals are heteroscedastic. However, the Durbin-Watson test gives a DW value close to 2, indicating that there is no autocorrelation.

In conclusion, the model is not ideal as the residuals are not normally distributed and heteroscedastic. However, the model is still acceptable as the residuals are approximately normally distributed in the middle range and there is no autocorrelation in the residuals. Moreover, the R-squared value of the model is approximately 0.85, indicating that the model explains 85% of the variance in the data. This is a better result than the baseline model, suggesting that the new model is more effective in predicting “mpg”.

To conduct testing on the test set, we first add the log transformation of “horsepower”, “displacement”, and “weight” to the test set.

```
# add log to test set
test_set$log_horsepower <- log(test_set$horsepower)
test_set$log_displacement <- log(test_set$displacement)
test_set$log_weight <- log(test_set$weight)
```

Now we can test the final model on the test set.

```
predictions <- predict(final_model, newdata = test_set)
# Actual values from the test set
actual_values <- test_set$mgp
# Calculate Mean Squared Error (MSE)
mse <- mean((predictions - actual_values)^2)
# Calculate R-squared
rss <- sum((predictions - actual_values)^2)
tss <- sum((actual_values - mean(actual_values))^2)
r_squared <- 1 - (rss / tss)
# Print metrics cat("Mean Squared Error (MSE):", mse, "\n")
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.8634455
```

The R-squared is approximately 0.86, which is higher than the R-squared of the baseline model. This suggests that the final model is more effective in predicting “mpg” than the baseline model.

Conclusion

In this task, we first preprocessed the data by removing the “car name” column, removing rows with missing values, converting the “horsepower” column to numeric, and changing the “origin” column to a factor. We then split the data into training and testing sets. We fitted a baseline linear regression model with all original variables to predict “mpg”.

To improve our baseline model, we **apply the knowledge of the in-reality relationship** between “mpg” and other variables to create a new model. We include the log transformation of “horsepower”, “displacement”, and “weight”, the interaction between “weight” and “acceleration”, and the square of time from “model year” to 1970 in the data. We then fit a new model with these variables. The new model has a higher R-squared value than the baseline model, suggesting that it is more effective in predicting “mpg”.

However, the new model is not ideal as the residuals are not normally distributed and heteroscedastic. Nevertheless, the new model is still acceptable as the residuals are approximately normally distributed in the middle range and there is no autocorrelation in the residuals. Finally, we test the new model on the test set and find that it has a higher R-squared value than the baseline model, indicating that the new model is more effective in predicting “mpg”.