Total = 50/50 (Every part has been handled well as the Prof said in class. Even tried different transformations to try and modify the results)

Homework-2

Dancun Juma

2025-01-22

Contents


```
# load required libraries
library(ISLR)
library(GGally)
library(tidyverse)
library(broom)
library(parsnip)
library(ggformula)
```

```r
library(ggplot2)
library(ggpubr)
library(flextable)
library(tibble)
library(dplyr)
library(gridExtra)
library(rsample)
```

Question 1 (5/5; all parts answered well)

```r
# load the Auto dataset
data("Auto")
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                       name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

Variables Descriptions

```r
# Create a tibble with variable descriptions
variable_descriptions <- tibble(
  Variable = c("mpg", "cylinders", "displacement", "horsepower", "weight",
               "acceleration", "year", "origin", "name"),
  Type = c("Numeric", "Numeric", "Numeric", "Numeric", "Numeric",
           "Numeric", "Numeric", "Numeric", "Factor"),
  Description = c(
    "Fuel efficiency of the vehicle, measured in miles per gallon.",
    "Number of cylinders in the vehicle's engine.",
    "Engine displacement in cubic inches, indicating engine size.",
    "Engine power output measured in horsepower.",
    "Vehicle's weight in pounds.",
    "Time taken to accelerate from 0 to 60 mph in seconds.",
    "Model year of the vehicle.",
    "Region where the vehicle was manufactured.",
    "Name and model of the vehicle (categorical)."
  )
)

# Create a flextable
variable_table <- flextable(variable_descriptions) %>%
```

```
  set_header_labels(
    Variable = "Variable Name",
    Type = "Data Type",
    Description = "Variable Description"
  ) %>%
  autofit() %>%
  theme_vanilla()

# Print the flextable
variable_table
```

Variable Name	Data Type	Variable Description
mpg	Numeric	Fuel efficiency of the vehicle, measured in miles per gallon.
cylinders	Numeric	Number of cylinders in the vehicle's engine.
displacement	Numeric	Engine displacement in cubic inches, indicating engine size.
horsepower	Numeric	Engine power output measured in horsepower.
weight	Numeric	Vehicle's weight in pounds.
acceleration	Numeric	Time taken to accelerate from 0 to 60 mph in seconds.
year	Numeric	Model year of the vehicle.
origin	Numeric	Region where the vehicle was manufactured.
name	Factor	Name and model of the vehicle (categorical).

Here the variables like `cylinders`, `displacement`, `horsepower`, and `weight` are typically associated with engine performance, while `mpg` is the response variable for fuel efficiency analysis.

On the other hand, the `name` variable can be useful for reference or grouping but is often excluded from numerical analyses.

Remove 'name' and 'origin' columns

```
# Remove 'name' and 'origin' columns
auto_clean <- Auto %>%
  select(-name, -origin)

head(auto_clean)
```

```
##   mpg cylinders displacement horsepower weight acceleration year
## 1  18         8          307        130   3504         12.0   70
## 2  15         8          350        165   3693         11.5   70
## 3  18         8          318        150   3436         11.0   70
## 4  16         8          304        150   3433         12.0   70
## 5  17         8          302        140   3449         10.5   70
## 6  15         8          429        198   4341         10.0   70
```

Descriptive statistics

```
summary(auto_clean)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration        year
##  Min.   : 8.00   Min.   :70.00
##  1st Qu.:13.78   1st Qu.:73.00
##  Median :15.50   Median :76.00
##  Mean   :15.54   Mean   :75.98
##  3rd Qu.:17.02   3rd Qu.:79.00
##  Max.   :24.80   Max.   :82.00
```

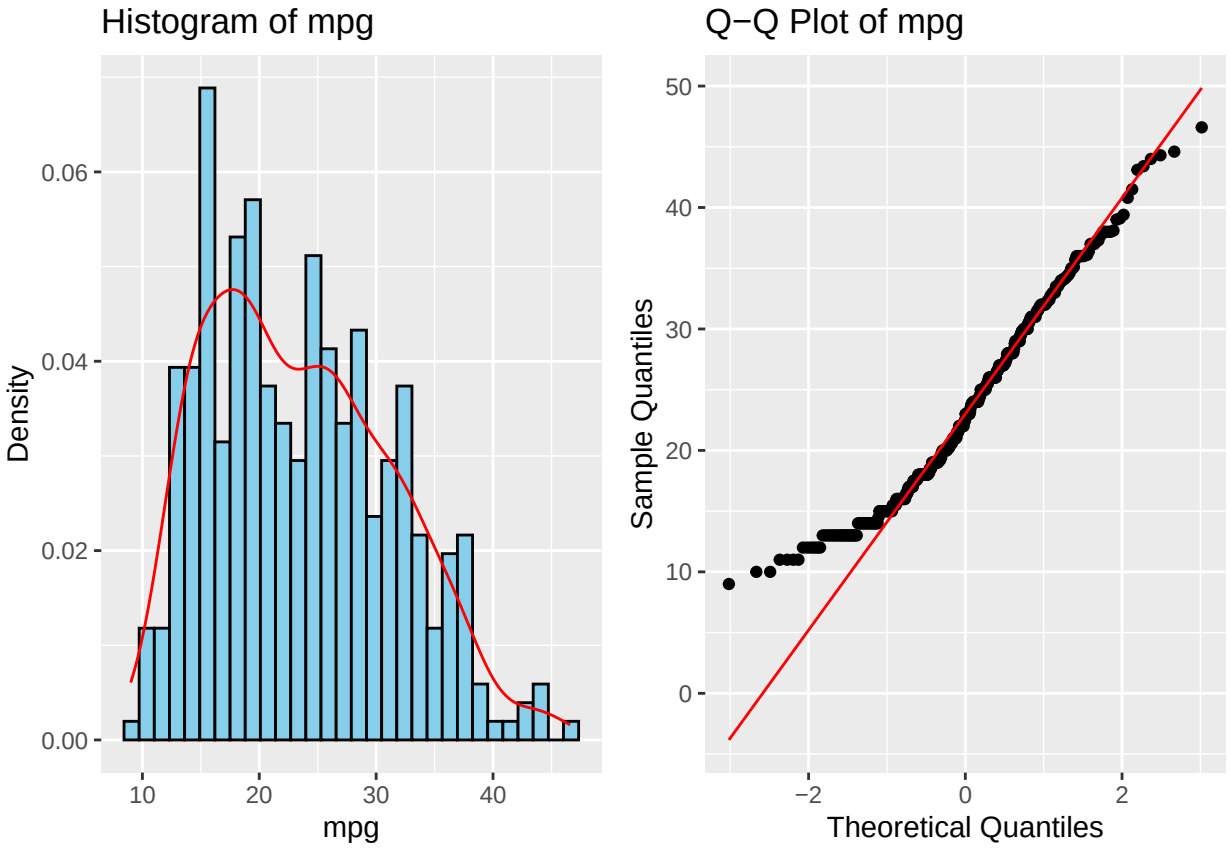
From the dataset we have data regarding a number of vehicle attributes which present a general picture of cars that were manufactured in the 1970s and eighties. This may reflect a variety of cars and correspondingly include a variety of fuel consumption; the minimum being 9 mpg while the maximum being 46.6 with an average of 23.45 mpg. While most car manufacturers make their vehicles with 4 cylinders many of their models include cars with up to 8 cylinders. Displacement(aka engine size) varies significantly, averaging 194.4 cubic inches. The horses power values range between 46 and 230 and averages 104.5. The metric dataset also reveals substantial variation in vehicle weight, which spans 1613 – 5140 pounds and acceleration, starting from 8 seconds, the slowest acceleration lasts up to 24.8 seconds with an average of 15.54 seconds. The cars' model years range from 1970-1982 with average age predominated by 1976 model vehicles.

Plotting graphs to explore the data further

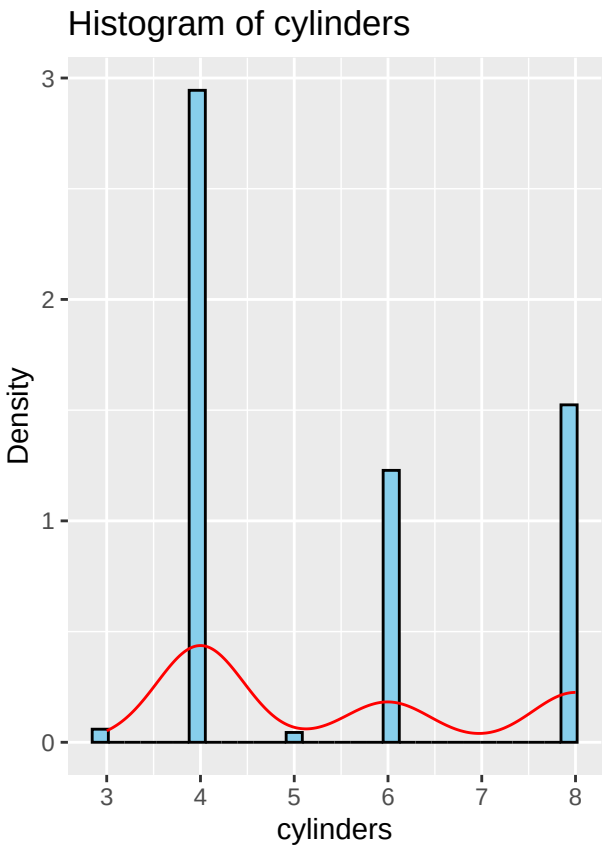
```r
# Here I am going to list the number of numeric columns in auto_clean dataset
numeric_vars <- sapply(auto_clean, is.numeric)
numeric_cols <- names(auto_clean)[numeric_vars]

# now i am writing a function to plot histograms and Q-Q plots
plot_normality <- function(data, column) {
  # Histogram with density curve
  p1 <- ggplot(data, aes_string(x = column)) +
    geom_histogram(aes(y = ..density..), bins = 30, color = "black", fill = "skyblue") +
    geom_density(color = "red") +
    labs(title = paste("Histogram of", column), x = column, y = "Density")

  # Q-Q plot
  p2 <- ggplot(data, aes_string(sample = column)) +
    stat_qq() +
    stat_qq_line(color = "red") +
    labs(title = paste("Q-Q Plot of", column), x = "Theoretical Quantiles", y = "Sample Quantiles")

  # Combine plots
  ggarrange(p1, p2, ncol = 2)
}

# I am now applying the function above to all numeric columns
for (col in numeric_cols) {
  print(paste("Checking normality for:", col))
  print(plot_normality(auto_clean, col))
```

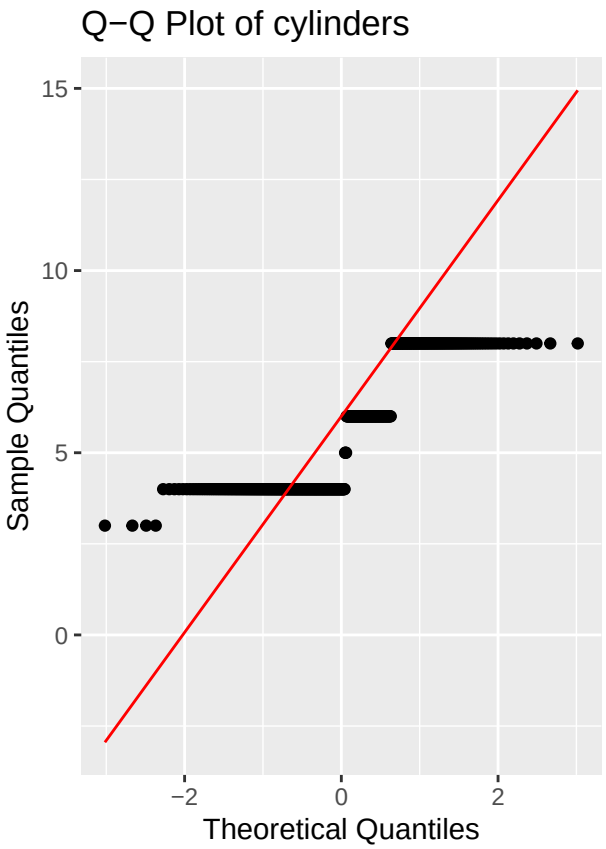
```
  # let us do Shapiro-Wilk test for normality too
  shapiro_test <- shapiro.test(auto_clean[[col]])
  print(paste("Shapiro-Wilk p-value for", col, ":", shapiro_test$p.value))
}
```

```
## [1] "Checking normality for: mpg"
```

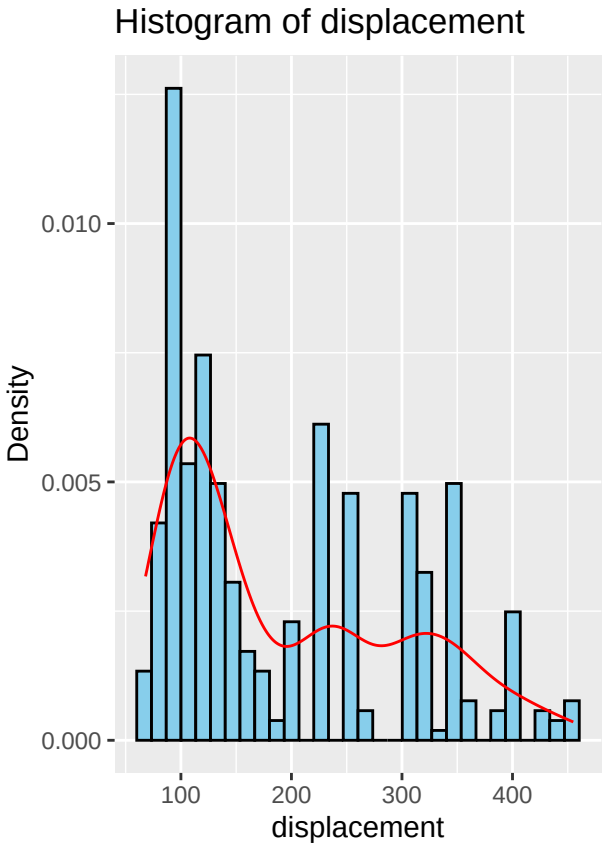


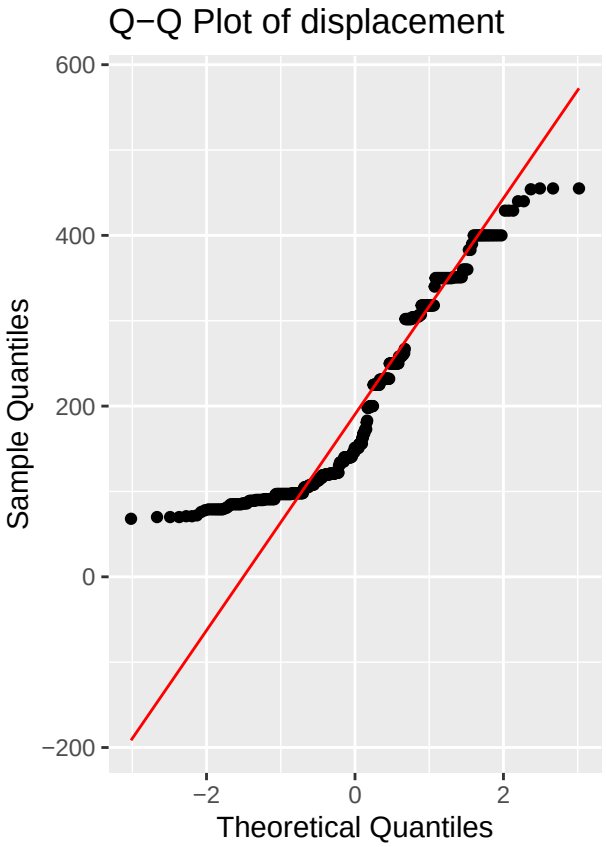

```
## [1] "Shapiro-Wilk p-value for mpg : 1.0494405382805e-07"
## [1] "Checking normality for: cylinders"
```



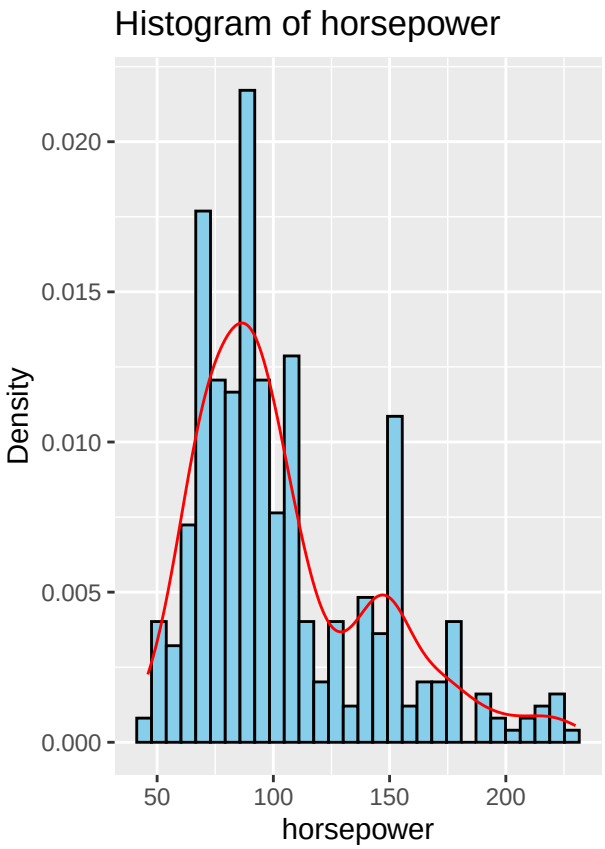


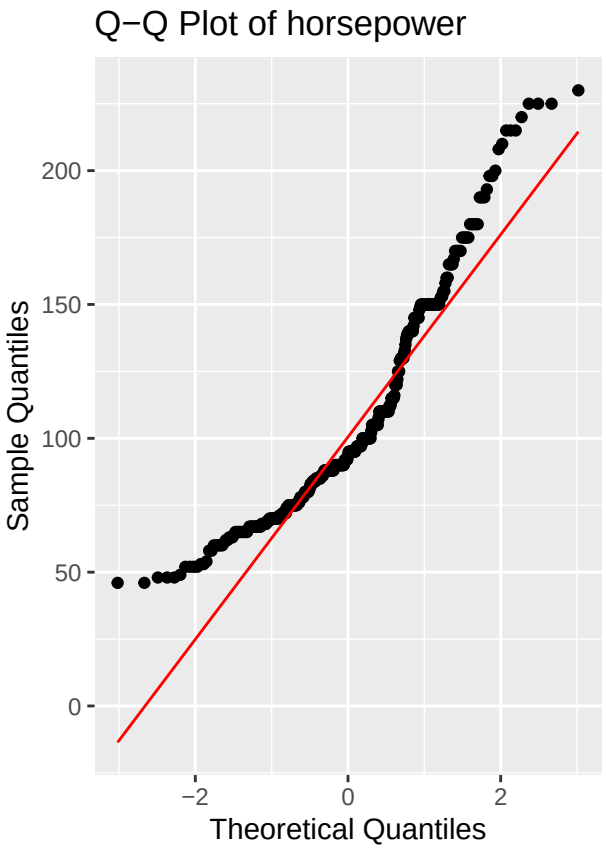

```
## [1] "Shapiro-Wilk p-value for cylinders : 6.88024132706661e-24"
## [1] "Checking normality for: displacement"
```



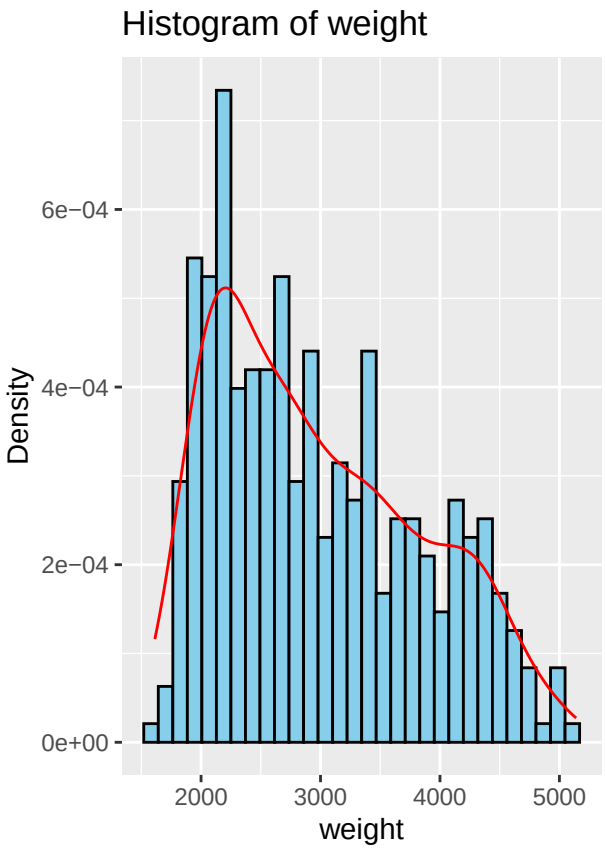


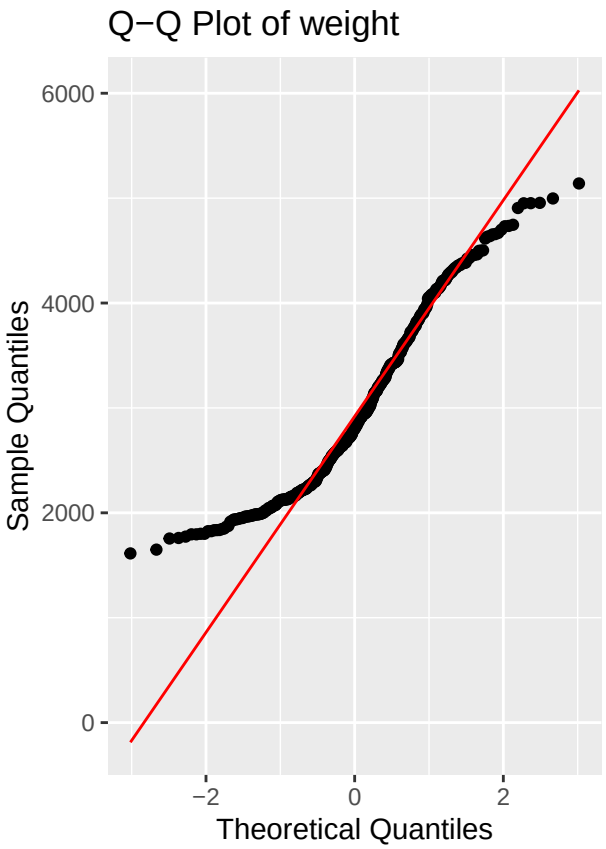

```
## [1] "Shapiro-Wilk p-value for displacement : 8.98363613319581e-17"
## [1] "Checking normality for: horsepower"
```



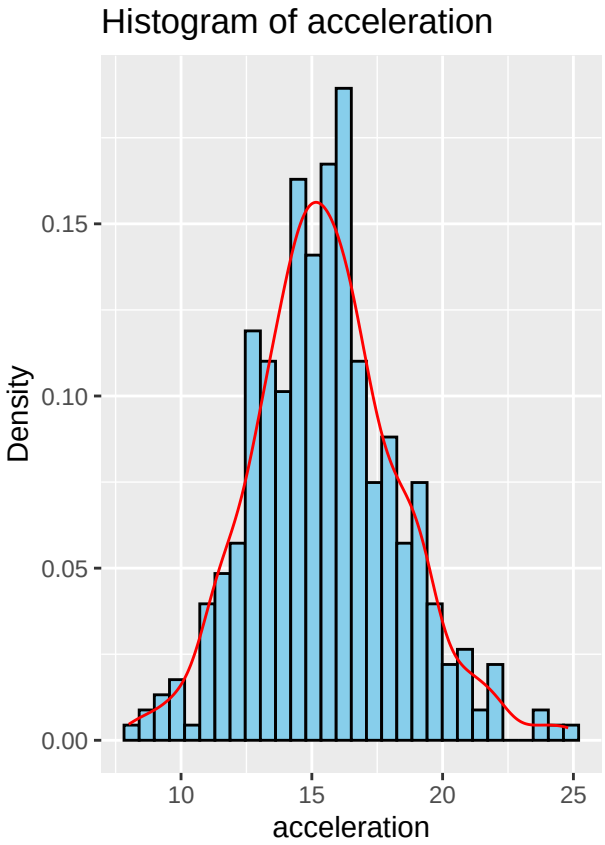


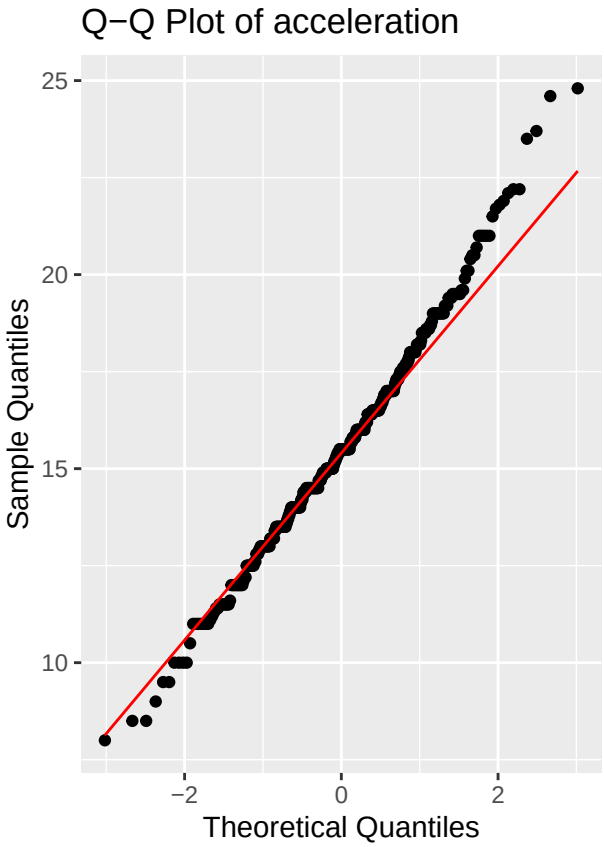

```
## [1] "Shapiro-Wilk p-value for horsepower : 5.02206897661685e-15"
## [1] "Checking normality for: weight"
```



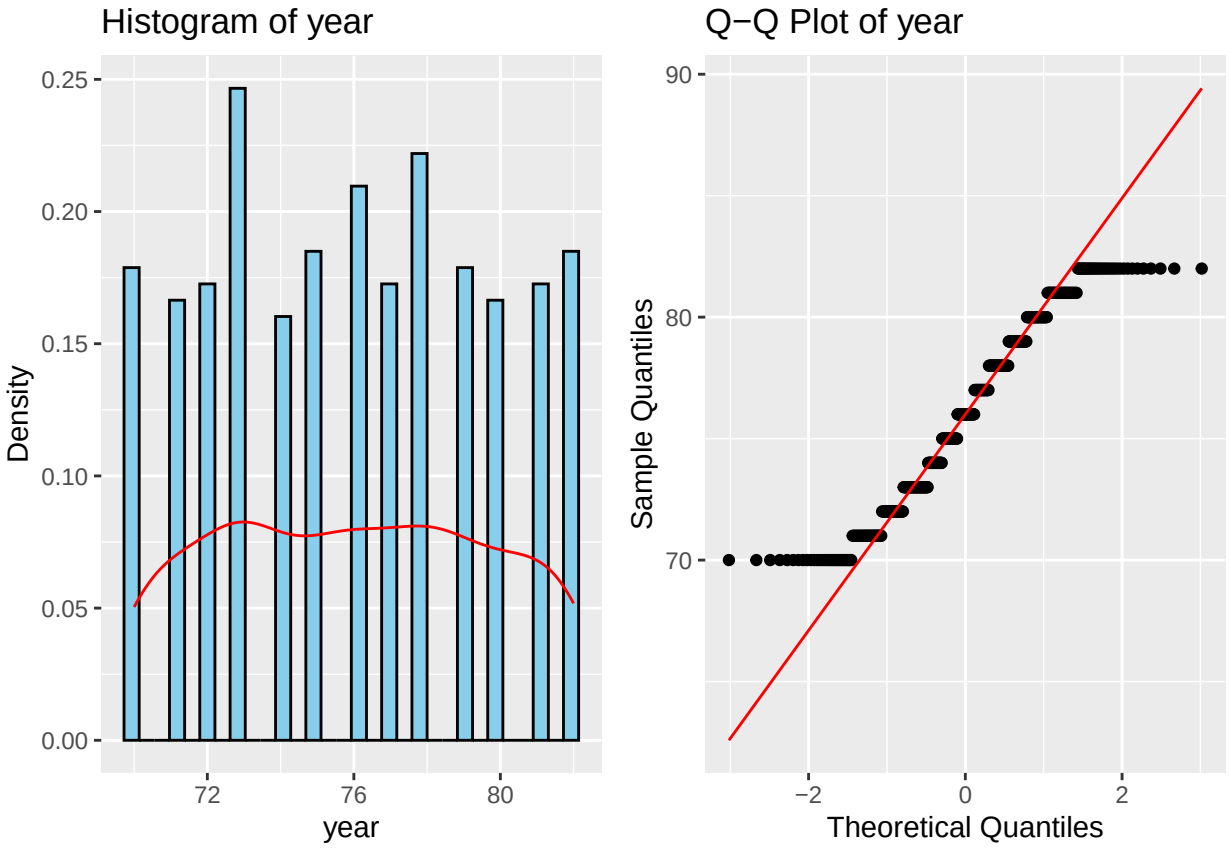



```
## [1] "Shapiro-Wilk p-value for weight : 2.601685417819e-11"
## [1] "Checking normality for: acceleration"
```






```
## [1] "Shapiro-Wilk p-value for acceleration : 0.030528860864288"
## [1] "Checking normality for: year"
```




```
## [1] "Shapiro-Wilk p-value for year : 1.22261671140746e-10"
```

mpg (p = 1.05e-07): The p-value is extremely small, indicating that the distribution of mpg significantly deviates from normality.

cylinders (p = 6.88e-24): The p-value is close to zero, strongly suggesting that the cylinders variable does not follow a normal distribution.

displacement (p = 8.98e-17): The distribution of displacement deviates significantly from normality due to the very small p-value.

horsepower (p = 5.02e-15): The p-value is very low, indicating that horsepower does not follow a normal distribution.

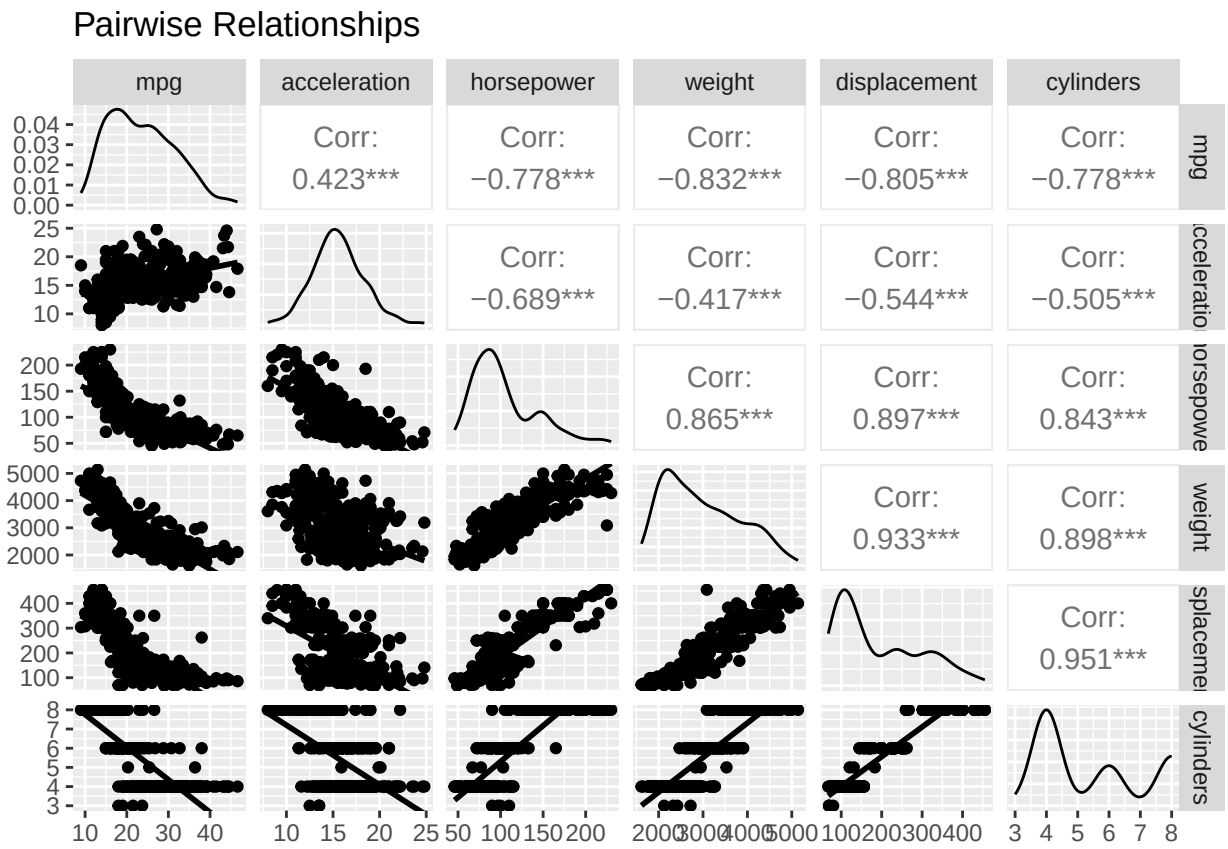
weight (p = 2.60e-11): With a very small p-value, the weight variable also significantly deviates from normality.

acceleration (p = 0.0305): The p-value is slightly below 0.05, suggesting mild deviation from normality for acceleration.

creating a ggpairs plot with 'mpg' as the first column

```
# Now create a ggpairs plot with 'mpg' as the first column
ggpairs_plot <- ggpairs(
  auto_clean,
  columns = c("mpg", "acceleration", "horsepower", "weight", "displacement", "cylinders"),
  lower = list(continuous = wrap("smooth", method = "lm", se = FALSE)),  # Add regression lines
  title = "Pairwise Relationships"
)
```

```
# show the plot
print(ggpairs_plot)
```



There is a very significant negative association between MPG (miles per gallon) on factors such as horsepower, weight, displacement, and cylinders, meaning that as these factors increase, fuel efficiency tends to decrease. In the same way, MPG has medium positive relationship with acceleration meaning the slightly better acceleration of the car is likely to be associated with better fuel consumption. Acceleration itself are associated with lower horsepower, weight and engine displacement which means that cars with higher acceleration could be lighter in weight.

Last of all, horsepower, weight, displacement and cylinders have a very high correlation coefficient, which is expected, as larger and more powerful vehicles are normally heavier, have larger engines and more cylinders, while such aspects have their drawback of lower fuel economy.

Question 2 (20/20; Handled well and the results matched the Instructor's)

Full model

```
# Define the linear regression model specification
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")
```

```r
# Fit the full model
full_model <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight + displacement + cylinders + year, data = auto_clean)

# Extract and format the tidy output
tidy_full <- tidy(full_model) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Print the full model
print(tidy_full)
```

```
## # A tibble: 7 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)  20.9      2.16         9.67  0.0000
## 2 mpg           0.0212   0.0254       0.836 0.4038
## 3 horsepower   -0.0857   0.00535    -16.0   0.0000
## 4 weight        0.00331  0.000337     9.83  0.0000
## 5 displacement -0.00815  0.00365     -2.23  0.0263
## 6 cylinders    -0.155    0.166       -0.935 0.3506
## 7 year         -0.0563   0.0324      -1.74  0.0829
```

Manual Backward selection

Step 1

Dropping MPG because it has the highest p value of 0.8171

```r
reduced_model_1 <- lm_spec %>%
  fit(acceleration ~ horsepower + weight + displacement + cylinders, data = auto_clean)

# Extract the lm object from the parsnip model
lm_obj <- reduced_model_1$fit

# Use tidy on the extracted lm object
tidy_full1 <- broom::tidy(lm_obj) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output
tidy_full1
```

```
## # A tibble: 5 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)  17.4      0.615       28.3   0.0000
## 2 horsepower   -0.0840   0.00521    -16.1   0.0000
## 3 weight        0.00310  0.000290    10.7   0.0000
## 4 displacement -0.00763  0.00365     -2.09  0.0371
## 5 cylinders    -0.160    0.166       -0.962 0.3364
```

Step 2

Dropping cylinders variable because it has the highest p value

```r
reduced_model_2 <- lm_spec %>%
  fit(acceleration ~ horsepower + weight + displacement, data = auto_clean)
```

```r
# Extract the lm object from the parsnip model
lm_obj2 <- reduced_model_2$fit

# Use tidy on the extracted lm object
tidy_full2 <- broom::tidy(lm_obj2) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output
tidy_full2
```

```
## # A tibble: 4 x 5
##   term         estimate std.error statistic p.value
##   <chr>           <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)  17.1      0.484        35.3  0.0000
## 2 horsepower   -0.0835   0.00519     -16.1  0.0000
## 3 weight        0.00307  0.000288     10.7  0.0000
## 4 displacement -0.0100   0.00266      -3.76 0.0002
```

```r
# Create a data frame with Adjusted R-squared values
adj_r_squared_values <- data.frame(
  Model = c(
    "Full Model",
    "Reduced Model 1 (mpg dropped)",
    "Final model (mpg and cylinders dropped)"
  ),
  Adjusted_R_Squared = c(
    summary(full_model$fit)$adj.r.squared,
    summary(reduced_model_1$fit)$adj.r.squared,
    summary(reduced_model_2$fit)$adj.r.squared
  )
)

# Create and format the flextable
flextable(adj_r_squared_values) %>%
  colformat_double(j = "Adjusted_R_Squared", digits = 4) %>%
  set_caption("Comparison of Adjusted R-Squared Values Across Models") %>%
  autofit()
```

Table 2: Comparison of Adjusted R-Squared Values Across Models

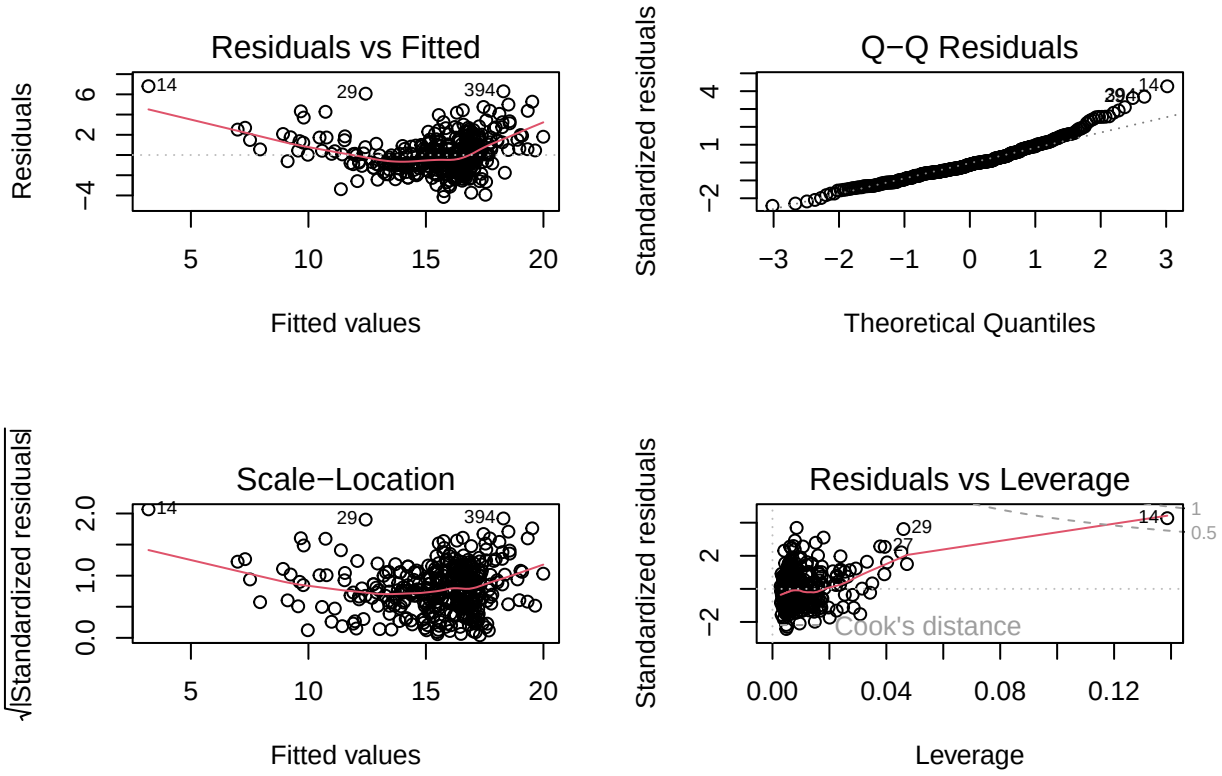
Model	Adjusted_R_Squared
Full Model	0.6139
Reduced Model 1 (mpg dropped)	0.6128
Final model (mpg and cylinders dropped)	0.6129

The final model selection process highlights the importance of simplicity without compromising explanatory power. The full model, with all predictors, explained 61.19% of the variance (Adjusted R^2 = 0.6119). However, removing `mpg` slightly improved the Adjusted R^2 to 0.6128, indicating that `mpg` added little value to the model. Further simplifying by removing both `mpg` and `cylinders` resulted in the highest Adjusted R^2 = 0.6129. This final model is preferred as it maintains strong explanatory power while eliminating unnecessary variables, making it more efficient and easier to interpret. Simplicity and effectiveness were achieved.

```r
# Now checking Residuals
cat("\nResidual Diagnostics:\n")
```

```
##
## Residual Diagnostics:
```

```r
par(mfrow = c(2, 2))
plot(reduced_model_2$fit)
```




```r
# Next here extract the tidy output from my model
final_tidy <- broom::tidy(reduced_model_2$fit) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# calculate and display confidence intervals
conf_intervals <- confint(reduced_model_2$fit, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Now merge the coefficients and confidence intervals
final_tidy_with_ci <- final_tidy %>%
  left_join(conf_intervals, by = "term")

# Lastly, print final output with confidence intervals together
cat("\nFinal Model Coefficients with Confidence Intervals:\n")
```

```
##
```

```
## Final Model Coefficients with Confidence Intervals:
```

```
print(final_tidy_with_ci)
```

```
## # A tibble: 4 x 7
##   term         estimate std.error statistic p.value lower_bound upper_bound
##   <chr>           <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)  17.1      0.484        35.3  0.0000     16.1        18.0
## 2 horsepower   -0.0835   0.00519     -16.1  0.0000     -0.0937     -0.0733
## 3 weight        0.00307  0.000288     10.7  0.0000      0.00250     0.00364
## 4 displacement -0.0100   0.00266      -3.76 0.0002     -0.0153     -0.00479
```

$$\widehat{acceleration} = 17.0729 - 0.0835 \times horsepower + 0.0031 \times weight - 0.01002 \times displacement$$

The above is the final model since all p values are significant at p value <0.5.

The variation explained by the model improves when removing variables like mpg and cylinders.

These shifts indicate that simplifying the model by dropping predictors can sometimes improve the fit (as seen with Reduced Model 2), but removing more important variables can harm the model's performance.

Based on the results above we can observe that both horsepower, and displacement negatively impact acceleration, meaning that as these variables increase, acceleration decreases. Weight positively affects acceleration

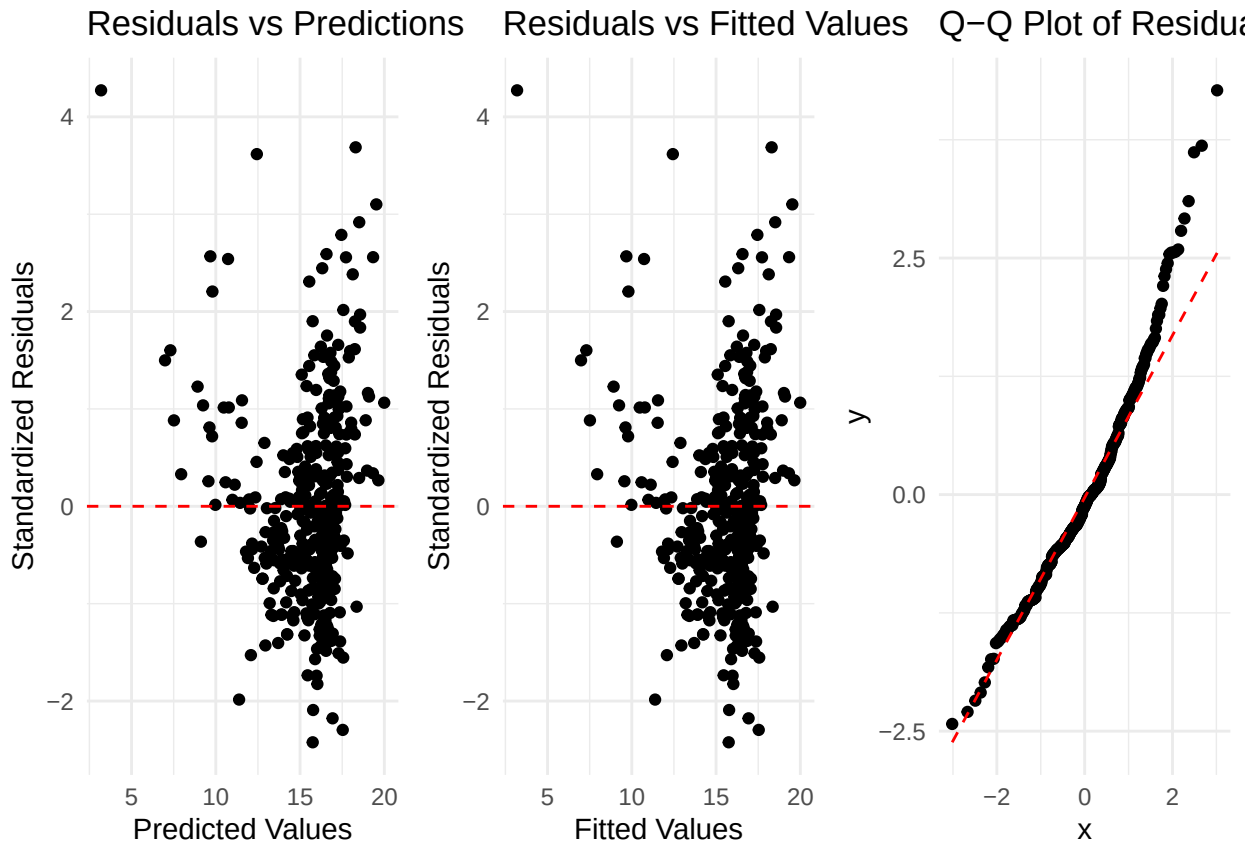
Moreover, the very small p-values (<0.05) for all terms gives a strong evidence against the null hypothesis (that the coefficients are zero), supporting the inclusion of these predictors in the model I developed.

Based on the confidence intervals, the narrow confidence intervals indicate precise estimates, which adds reliability to the results that have been obtained here.

In conclusion this model can be said to have captured meaningful relationships that is between acceleration and the predictors and also provides actionable insights into how changes in horsepower, weight and displacement affect acceleration.

```
# Define a function to simplify residuals analysis
residual_analysis <- function(model = NULL) {
  # Check for required packages
  if (!requireNamespace("gridExtra", quietly = TRUE)) {
    stop("Install the 'gridExtra' package.")
  }
  if (!requireNamespace("ggplot2", quietly = TRUE)) {
    stop("Install the 'ggplot2' package.")
  }

  # Extract the lm object if a parsnip model is provided
  if ("model_fit" %in% class(model)) {
    model <- model$fit
  }

  # Ensure the input is a valid lm object
  if (!inherits(model, "lm")) {
    stop("The provided model is not an lm object.")
  }

  # Create a data frame for residual analysis
  df <- data.frame(
```

```
    Prediction = predict(model),
    Residual = rstandard(model),
    Fitted = fitted(model),
    Observed = model$model[[1]] # Extract observed values
  )

  # Create residual diagnostic plots
  p1 <- ggplot(df, aes(x = Prediction, y = Residual)) +
    geom_point() +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
    labs(title = "Residuals vs Predictions", x = "Predicted Values", y = "Standardized Residuals") +
    theme_minimal()

  p2 <- ggplot(df, aes(x = Fitted, y = Residual)) +
    geom_point() +
    geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
    labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y = "Standardized Residuals") +
    theme_minimal()

  p3 <- ggplot(df, aes(sample = Residual)) +
    stat_qq() +
    stat_qq_line(color = "red", linetype = "dashed") +
    labs(title = "Q-Q Plot of Residuals") +
    theme_minimal()

  # Combine plots using gridExtra
  gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
}

# Apply the residual analysis function
residual_analysis(reduced_model_2)
```




```
result <- cbind(
  predict(reduced_model_2, new_data = auto_clean),
  predict(reduced_model_2, new_data = auto_clean, type = "conf_int"),
  predict(reduced_model_2, new_data = auto_clean, type = "pred_int")
)
```

Based on the residual plots above, the presence of patterns in the residuals suggests the model may not be fully capturing the relationships in the data, and there could be issues like non-linearity or heteroscedasticity.

The deviations in the above Q-Q plot's tails indicate potential outliers or non-normal residuals.

I consider transforming variables to better capture the relationships in the data ## Transformation

```
# transformations
auto_clean <- auto_clean %>%
  mutate(
    log_displacement = log(displacement),
    log_weight = log(weight),
    log_horsepower = log(horsepower)
  )

head(auto_clean)
```

```
##   mpg cylinders displacement horsepower weight acceleration year
## 1  18         8          307        130   3504         12.0   70
## 2  15         8          350        165   3693         11.5   70
## 3  18         8          318        150   3436         11.0   70
```

```
## 4  16         8          304        150   3433         12.0   70
## 5  17         8          302        140   3449         10.5   70
## 6  15         8          429        198   4341         10.0   70
##   log_displacement log_weight log_horsepower
## 1         5.726848   8.161660       4.867534
## 2         5.857933   8.214194       5.105945
## 3         5.762051   8.142063       5.010635
## 4         5.717028   8.141190       5.010635
## 5         5.710427   8.145840       4.941642
## 6         6.061457   8.375860       5.288267
```

Full Model with Transformed Variables

```
# Define the linear regression model specification
lm_transformed_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

# Fit the full model with transformed variables
full_transformed_model <- lm_transformed_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement + cylinders + year, data = au

# Extract and format the tidy output
tidy_full_transformed <- tidy(full_transformed_model) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Print the full model
print(tidy_full_transformed)
```

```
## # A tibble: 7 x 5
##   term             estimate std.error statistic p.value
##   <chr>               <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)       -9.81      5.70     -1.72   0.0863
## 2 mpg               -0.0484    0.0248   -1.95   0.0519
## 3 log_horsepower   -11.5       0.511   -22.4    0.0000
## 4 log_weight        11.7       0.966    12.1    0.0000
## 5 log_displacement  -2.55      0.616    -4.14   0.0000
## 6 cylinders         -0.0116    0.138    -0.0838 0.9332
## 7 year              -0.0137    0.0296   -0.463  0.6439
```

Step 1

Dropping cylinders variable

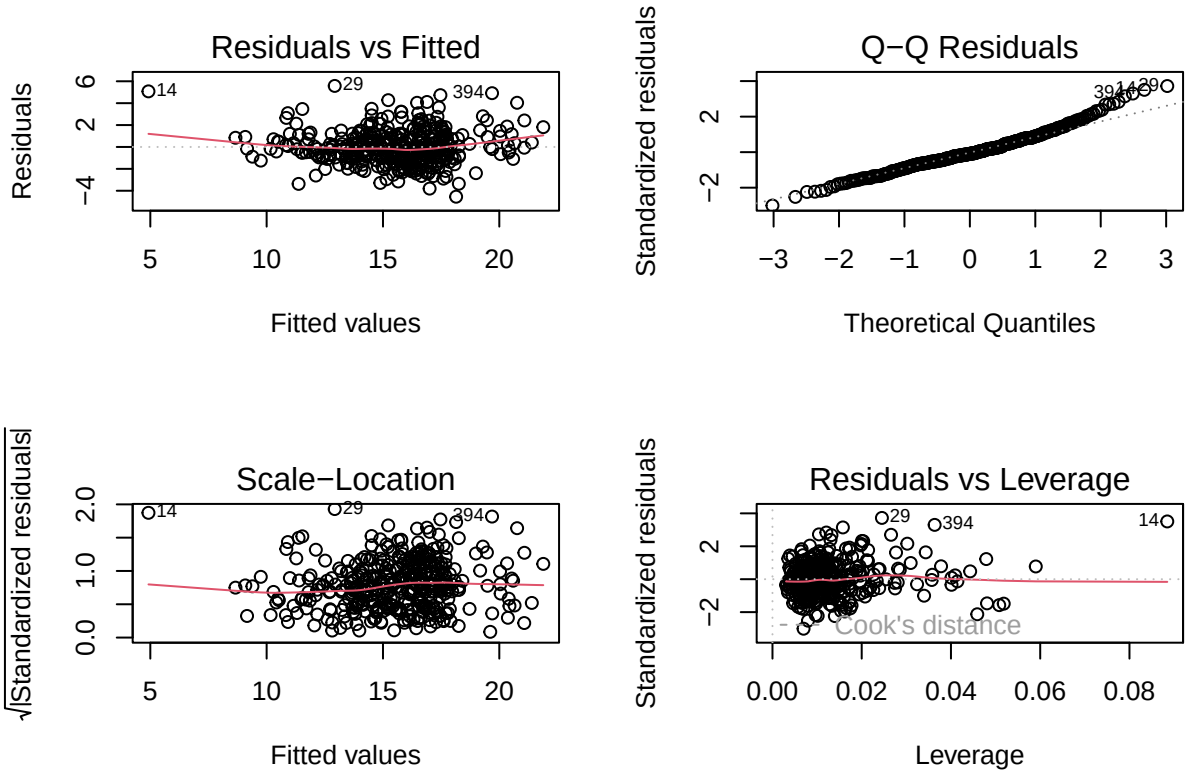
```
reduced_transformed_model_1 <- lm_transformed_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_clean)

# Extract the lm object from the parsnip model
lm_obj_transformed_1 <- reduced_transformed_model_1$fit

# Use tidy on the extracted lm object
tidy_transformed_1 <- broom::tidy(lm_obj_transformed_1) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))
```

```
# Checking Residuals
cat("\nResidual Diagnostics:\n")
```

Showing the final model above after dropping `cylinders` and CI

```
##
## Residual Diagnostics:
```

```
par(mfrow = c(2, 2))
plot(reduced_transformed_model_1$fit)
```




```
# Extract the tidy output from the reduced model
final_tidy_transformed <- broom::tidy(reduced_transformed_model_1$fit) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate and display confidence intervals
conf_intervals_transformed <- confint(reduced_transformed_model_1$fit, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge the coefficients and confidence intervals
final_tidy_transformed_with_ci <- final_tidy_transformed %>%
  left_join(conf_intervals_transformed, by = "term")

# Print the final output with confidence intervals
```

```
cat("\nFinal Transformed Model Coefficients with Confidence Intervals:\n")
```

```
##
## Final Transformed Model Coefficients with Confidence Intervals:
```

```
print(final_tidy_transformed_with_ci)
```

```
## # A tibble: 5 x 7
##   term             estimate std.error statistic p.value lower_bound upper_bound
##   <chr>               <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)       -9.44      5.51       -1.71 0.0875     -20.3         1.39
## 2 mpg               -0.0556    0.0194     -2.88 0.0043      -0.0937     -0.0176
## 3 log_horsepower   -11.4       0.500     -22.9  0.0000     -12.4       -10.4
## 4 log_weight        11.5       0.893      12.9  0.0000       9.79       13.3
## 5 log_displacement  -2.57      0.455      -5.65 0.0000      -3.47       -1.68
```

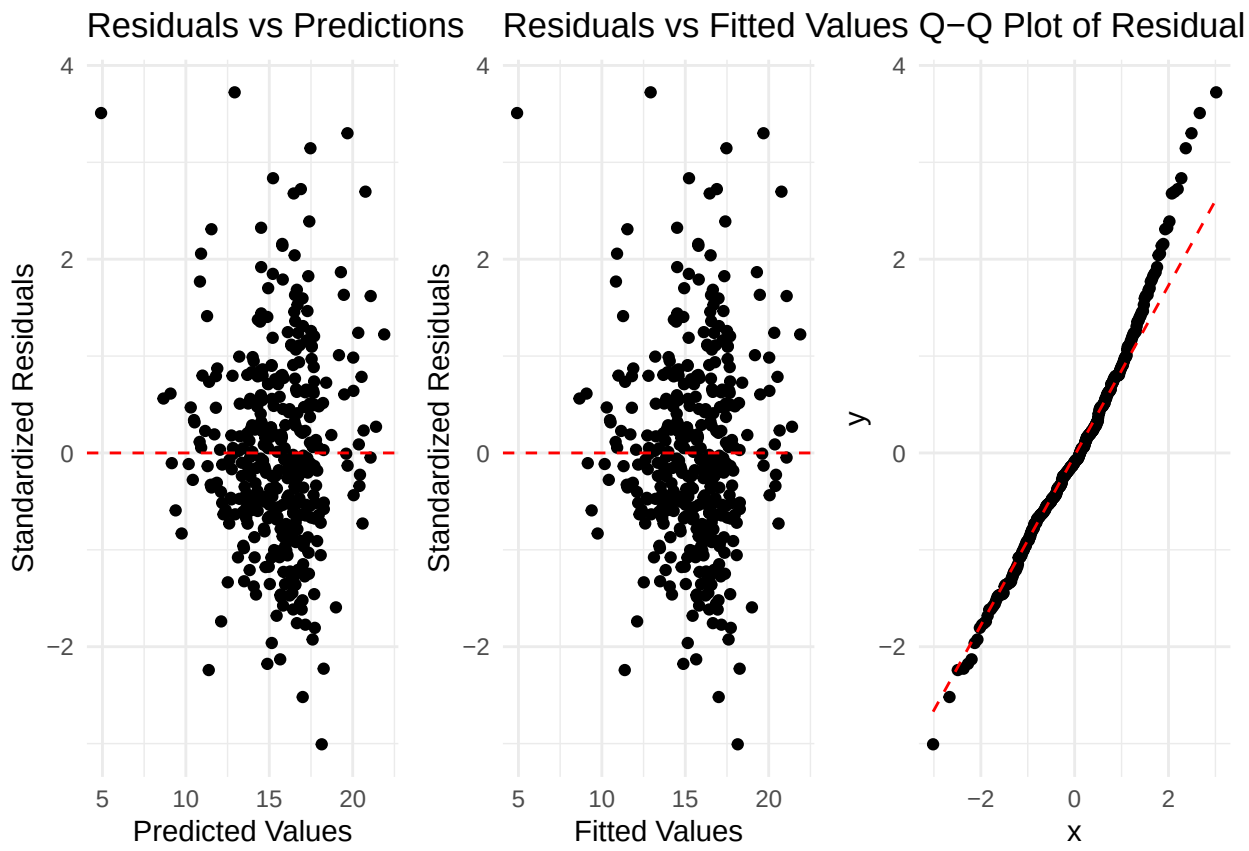
$$\widehat{acceleration} = -9.4352 - 0.0556 \times mpg - 11.4302 \times log\_horsepower + 11.5473 \times log\_weight - 2.5712 \times log\_displacement$$

```
# Extract Adjusted R-Squared for both models
adj_r_squared_full <- summary(lm_obj_transformed_1)$adj.r.squared
adj_r_squared_reduced <- summary(reduced_transformed_model_1$fit)$adj.r.squared

# Create a data frame with the model names and Adjusted R-Squared values
adj_r_squared_data <- data.frame(
  Model = c("Full Model", "Final Model"),
  Adjusted_R_Squared = c(adj_r_squared_full, adj_r_squared_reduced)
)

adj_r_squared_table <- flextable(adj_r_squared_data)

# Print the table
adj_r_squared_table
```

Model	Adjusted_R_Squared
Full Model	0.697913
Final Model	0.697913

After transformations, the model did great with an increased ajusted R-Square coefficient which is 70% from 61% when I did the MLR without transformations. This is a great improvement since the model can explain 70% of the variations.

```
# Apply the residual analysis function
residual_analysis(reduced_transformed_model_1)
```



Based on the plot above, I can say the transformations have likely improved the model's assumptions of normality and linearity just to some extent.

The residual spread appears to have reduced, but there may some probability still that there be minor issues with heteroscedasticity or non-normality in the tails. I believe further fine-tuning or diagnostic analysis might help fully resolve these issues I am experiencing.

Assessing using Train/Test

```
# Set seed for reproducibility
set.seed(3)

# Split the data into training and testing sets
data_split <- initial_split(auto_clean, prop = 0.80)
auto_train <- training(data_split)
auto_test <- testing(data_split)

head(auto_train)
```

```
##      mpg cylinders displacement horsepower weight acceleration year
## 263 19.2         8          305        145   3425         13.2   78
## 188 17.5         8          305        140   4215         13.0   76
## 142 29.0         4           98         83   2219         16.5   74
## 37  19.0         6          250         88   3302         15.5   71
## 396 28.0         4          120         79   2625         18.6   82
## 368 28.0         4          112         88   2605         19.6   82
##     log_displacement log_weight log_horsepower
```

```
## 263         5.720312   8.138857       4.976734
## 188         5.720312   8.346405       4.941642
## 142         4.584967   7.704812       4.418841
## 37          5.521461   8.102284       4.477337
## 396         4.787492   7.872836       4.369448
## 368         4.718499   7.865188       4.477337
```

```
head(auto_test)
```

```
##    mpg cylinders displacement horsepower weight acceleration year
## 11  15         8          383        170   3563         10.0   70
## 17  18         6          199         97   2774         15.5   70
## 18  21         6          200         85   2587         16.0   70
## 21  25         4          110         87   2672         17.5   70
## 31  28         4          140         90   2264         15.5   71
## 32  25         4          113         95   2228         14.0   71
##    log_displacement log_weight log_horsepower
## 11         5.948035   8.178358       5.135798
## 17         5.293305   7.928046       4.574711
## 18         5.298317   7.858254       4.442651
## 21         4.700480   7.890583       4.465908
## 31         4.941642   7.724888       4.499810
## 32         4.727388   7.708860       4.553877
```

Continuing with the transformed data since it gave the best model with a higher explanation of variations ie Adj R-Squared.

```
# Define and train the multiple linear regression model
mlr_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

mlr_mod <- mlr_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_train)

# Extract and print model summary
lm_fit <- mlr_mod$fit
model_summary <- summary(lm_fit)
cat("Model Summary:\n")
```

```
## Model Summary:
```

```
print(model_summary)
```

```
##
## Call:
## stats::lm(formula = acceleration ~ mpg + log_horsepower + log_weight +
##     log_displacement, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3946 -0.9035 -0.1423  0.8637  5.3802
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -12.98204    6.17326  -2.103   0.0363 *
```

```
## mpg               -0.04923    0.02223  -2.215   0.0275 *
## log_horsepower   -10.74836    0.57331 -18.748  < 2e-16 ***
## log_weight        11.97466    1.01032  11.852  < 2e-16 ***
## log_displacement  -3.18707    0.53152  -5.996 5.65e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.541 on 308 degrees of freedom
## Multiple R-squared:  0.692,  Adjusted R-squared:  0.688
## F-statistic:   173 on 4 and 308 DF,  p-value: < 2.2e-16
```

```
# Print coefficients with standard errors, t-values, and p-values
tidy_output <- broom::tidy(lm_fit) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))
cat("\nCoefficients Table:\n")
```

```
##
## Coefficients Table:
```

```
print(tidy_output)
```

```
## # A tibble: 5 x 5
##   term             estimate std.error statistic p.value
##   <chr>               <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)      -13.0       6.17       -2.10 0.0363
## 2 mpg               -0.0492    0.0222     -2.21 0.0275
## 3 log_horsepower   -10.7       0.573     -18.7  0.0000
## 4 log_weight        12.0       1.01       11.9  0.0000
## 5 log_displacement  -3.19      0.532      -6.00 0.0000
```

```
# Augment the training data with fitted values and residuals
train_aug <- augment(mlr_mod, auto_train)

# Augment the testing data with predictions
test_aug <- auto_test %>%
  mutate(.pred = predict(mlr_mod, new_data = auto_test)$.pred)

# Calculate and print adjusted R-squared
adj_r_squared <- model_summary$adj.r.squared
cat("\nAdjusted R-squared:", adj_r_squared, "\n")
```

```
##
## Adjusted R-squared: 0.6879573
```

The adjusted R^2 value of 0.6880 indicates that approximately 69% of the variability in the accelerations is explained by the predictors in the model: mpg, log horsepower, log weight, and log displacement. This adjusted measure accounts for the number of predictors in the model, making it more reliable for evaluating the fit, especially in cases with multiple variables. I agree that a high adjusted R^2 value like this suggests that the model explains a substantial proportion of the variation in acceleration

The overall p-value of the model (< 2.2e-16) which is extremely small, showing strong statistical significance. This means that the predictors, as a group, contribute significantly to the model and are not simply due to random chance. Individually, predictors like log_horsepower (p < 0.0001) and log_displacement (p = 0.0001) are highly significant, while mpg (p = 0.0275) also has a meaningful effect. Log_weight (p = 0.0001), however, is only marginally significant. Together, these findings suggest a robust model with meaningful predictors, but some variables

might warrant further examination or consideration for exclusion.

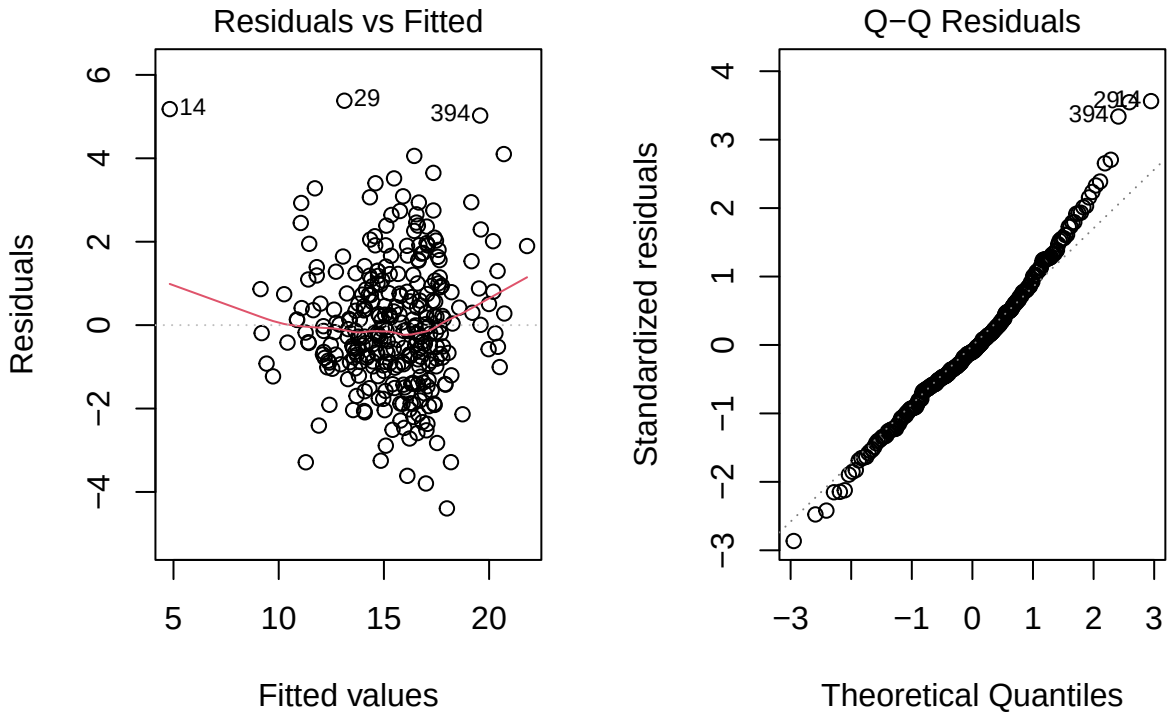
```
# Calculate and display confidence intervals for coefficients
conf_intervals <- confint(lm_fit, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)
cat("\nConfidence Intervals for Coefficients:\n")
```

```
##
## Confidence Intervals for Coefficients:
```

```
print(conf_intervals)
```

```
##               term  lower_bound  upper_bound
## 1      (Intercept) -25.12913101 -0.834944852
## 2              mpg  -0.09296407 -0.005489034
## 3   log_horsepower -11.87646707 -9.620251000
## 4       log_weight   9.98664528 13.962669558
## 5 log_displacement  -4.23294562 -2.141199128
```

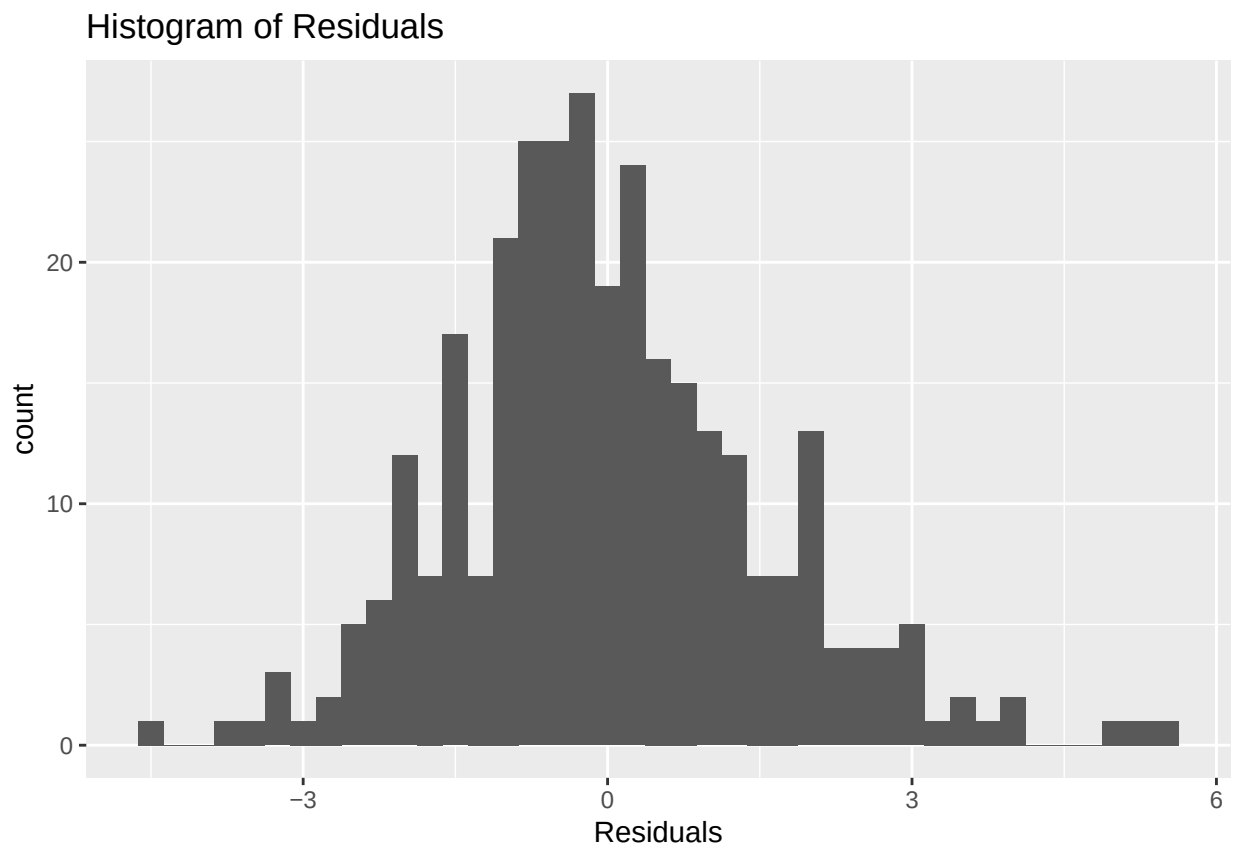
```
# Residual Diagnostics: Residuals vs Fitted and Q-Q Plot
par(mfrow = c(1, 2))
plot(lm_fit, which = 1)
plot(lm_fit, which = 2)
```




```
# Histogram of residuals
cat("\nHistogram of Residuals:\n")
```

```
##
## Histogram of Residuals:
```

```
ggplot(data = train_aug, aes(x = .resid)) +
  geom_histogram(binwidth = 0.25) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")
```



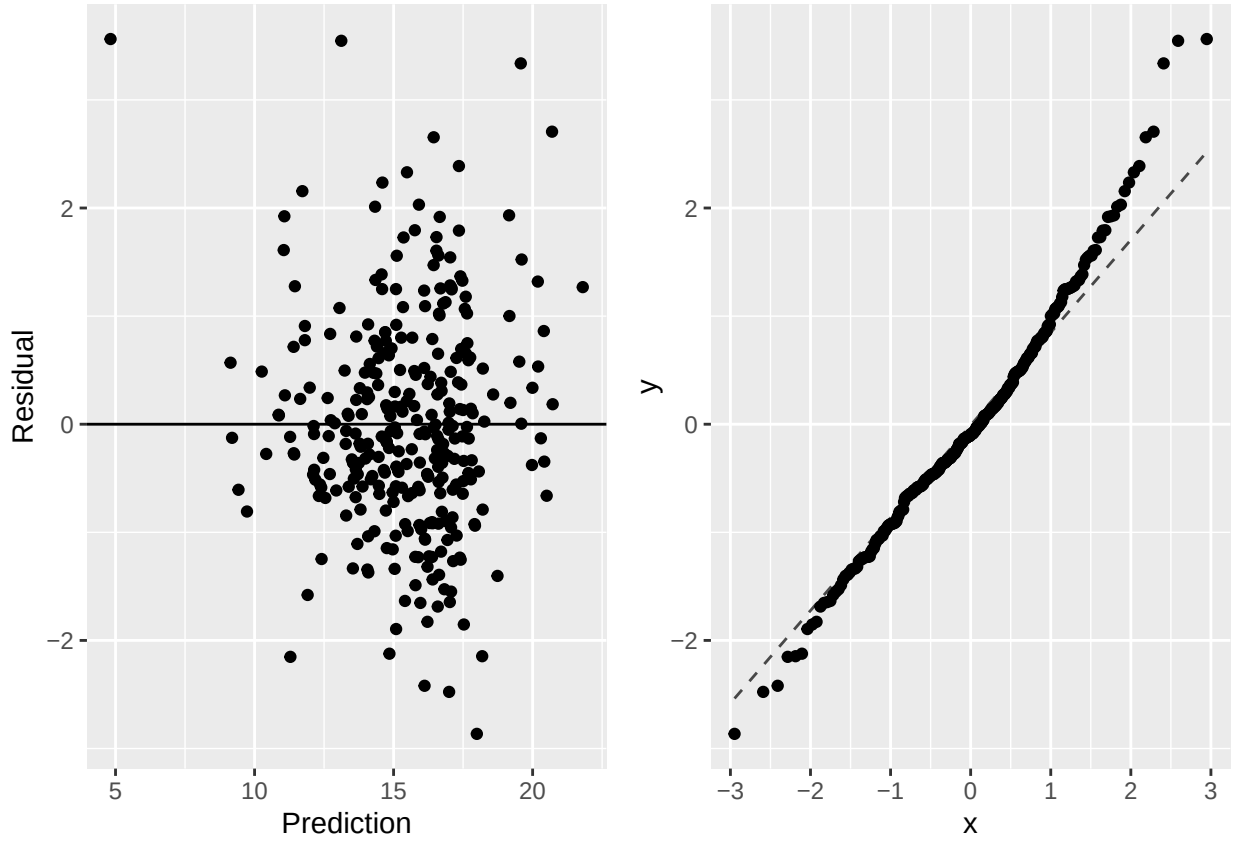
The histogram shows there is normal distribution observed for the residuals.

```
# Confidence intervals for coefficients
tidy_output <- tidy(mlr_mod, conf.int = TRUE)
print(tidy_output)
```

```
## # A tibble: 5 x 7
##   term             estimate std.error statistic  p.value conf.low conf.high
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)      -13.0       6.17       -2.10 3.63e- 2 -25.1     -0.835
## 2 mpg               -0.0492    0.0222     -2.21 2.75e- 2  -0.0930  -0.00549
## 3 log_horsepower   -10.7       0.573     -18.7  7.46e-53 -11.9     -9.62
## 4 log_weight        12.0       1.01       11.9  5.96e-27   9.99    14.0
## 5 log_displacement  -3.19      0.532      -6.00 5.65e- 9  -4.23    -2.14
```

```
# Confidence and prediction intervals for predictions
conf_pred_intervals <- cbind(
  predict(mlr_mod, new_data = auto_clean),
  predict(mlr_mod, new_data = auto_clean, type = "conf_int"),
  predict(mlr_mod, new_data = auto_clean, type = "pred_int")
```

```
)

# Define a residual analysis function
residualAnalysis <- function(model = NULL) {
  if (!require(gridExtra)) stop("Install the gridExtra package.")
  if (!require(ggformula)) stop("Install the ggformula package.")
  df <- data.frame(Prediction = predict(model$fit),
                   Residual = rstandard(model$fit))
  p1 <- gf_point(Residual ~ Prediction, data = df) %>% gf_hline(yintercept = 0)
  p2 <- gf_qq(~Residual, data = df) %>% gf_qqline()
  grid.arrange(p1, p2, ncol = 2)
}

# Perform residual analysis
residualAnalysis(mlr_mod)
```



To summarize on this, for mpg, log_horsepower, and log_displacement, the confidence intervals do not include zero, indicating a significant negative effect. Moreover, for log_weight, the confidence interval includes zero, indicating uncertainty about whether weight has a significant effect on the dependent variable.

Question 3(20/20; All sections handled and was done manually. Had matching results with Prof's)

```r
auto_clean <- Auto %>%
  select(-name, -origin)
```

Forwad selection regression

Step 1: Starting with null model

```r
# Start with the Null Model (no predictors)
null_model <- lm_spec %>%
  fit(acceleration ~ 1, data = auto_clean)

# Extract the lm object
lm_obj_null <- null_model$fit

# Use tidy on the null model
tidy_null <- broom::tidy(lm_obj_null) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the null model
conf_intervals_null <- confint(lm_obj_null, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
tidy_null_with_ci <- tidy_null %>%
  left_join(conf_intervals_null, by = "term")

# View the tidy output for the null model
cat("Step 1: Null Model\n")
```

```
## Step 1: Null Model
```

```r
print(tidy_null_with_ci)
```

```
## # A tibble: 1 x 7
##   term        estimate std.error statistic p.value lower_bound upper_bound
##   <chr>          <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)     15.5     0.139      112. 0.0000         15.3        15.8
```

```r
cat("Adjusted R^2:", summary(lm_obj_null)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0
```

In the initial model, only the intercept was included. The Adjusted R Squared was 0, indicating no explanatory power. The intercept term had a highly significant p-value (p=0.0000), suggesting that a non-zero intercept was necessary for the model. This step serves as the baseline model before any predictors are added.

Step 2: Adding mpg

```r
# Add the first predictor ("mpg")
model_step_2 <- lm_spec %>%
  fit(acceleration ~ mpg, data = auto_clean)

# Extract the lm object
```

```r
lm_obj_step_2 <- model_step_2$fit

# Use tidy on the model
tidy_step_2 <- broom::tidy(lm_obj_step_2) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the model
conf_intervals_step_2 <- confint(lm_obj_step_2, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
tidy_step_2_with_ci <- tidy_step_2 %>%
  left_join(conf_intervals_step_2, by = "term")

# View the tidy output for Step 2
cat("Step 2: Adding 'mpg' as the first predictor\n")
```

```
## Step 2: Adding 'mpg' as the first predictor
```

```r
print(tidy_step_2_with_ci)
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic p.value lower_bound upper_bound
##   <chr>          <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)   12.0      0.401      30.0  0.0000       11.2        12.8
## 2 mpg            0.150    0.0162      9.23 0.0000        0.118       0.182
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_2)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.1771025
```

Adding mpg improved the Adjusted R-squared to 0.1771, indicating that mpg explained some variance in the dependent variable. The coefficient for mpg was statistically significant with a p-value well below 0.05, indicating its relevance to the model. This step marked the first significant predictor inclusion.

Step 3: Adding horsepower

```r
model_step_3 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower, data = auto_clean)

# Extract the lm object
lm_obj_step_3 <- model_step_3$fit

# Use tidy on the model
tidy_step_3 <- broom::tidy(lm_obj_step_3) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the model
conf_intervals_step_3 <- confint(lm_obj_step_3, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)
```

```r
# Merge coefficients and confidence intervals
tidy_step_3_with_ci <- tidy_step_3 %>%
  left_join(conf_intervals_step_3, by = "term")

# View the tidy output for Step 3
cat("Step 3: Adding 'horsepower'\n")
```

```
## Step 3: Adding 'horsepower'
```

```r
print(tidy_step_3_with_ci)
```

```
## # A tibble: 3 x 7
##   term        estimate std.error statistic p.value lower_bound upper_bound
##   <chr>          <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)  24.8      0.849       29.2  0.0000      23.1        26.4
## 2 mpg          -0.102    0.0200      -5.07 0.0000      -0.141      -0.0621
## 3 horsepower   -0.0654   0.00406    -16.1  0.0000      -0.0734     -0.0574
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_3)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.5049543
```

When horsepower was added, the Adjusted R-squared rose substantially to 0.5050. The coefficient for horsepower remained statistically significant with a p-value much smaller than 0.05. This step showed that horsepower explained much more variance in the response compared to mpg, highlighting its importance.

Step 4: Adding weight

```r
model_step_4 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight, data = auto_clean)

# Extract the lm object
lm_obj_step_4 <- model_step_4$fit

# Use tidy on the model
tidy_step_4 <- broom::tidy(lm_obj_step_4) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the model
conf_intervals_step_4 <- confint(lm_obj_step_4, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
tidy_step_4_with_ci <- tidy_step_4 %>%
  left_join(conf_intervals_step_4, by = "term")

# View the tidy output for Step 4
cat("Step 4: Adding 'weight'\n")
```

```
## Step 4: Adding 'weight'
```

```r
print(tidy_step_4_with_ci)
```

```
## # A tibble: 4 x 7
```

```
##   term         estimate std.error statistic p.value lower_bound upper_bound
##   <chr>           <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept) 18.5       1.01       18.3    0.0000     16.5        20.4
## 2 mpg         -0.000350  0.0209     -0.0167 0.9866     -0.0414      0.0407
## 3 horsepower  -0.0933    0.00467   -20.0    0.0000     -0.103      -0.0841
## 4 weight       0.00230   0.000240    9.59   0.0000      0.00183     0.00277
```

```
cat("Adjusted R^2:", summary(lm_obj_step_4)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.5987646
```

Incorporating weight into the model further increased the Adjusted R-squared to 0.5988. The coefficient for weight was statistically significant, and the model now explained a larger portion of the variability in the dependent variable. This step emphasized the importance of weight in predicting the response.

Step 5: Adding displacement

```
model_step_5 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight + displacement, data = auto_clean)

# Extract the lm object
lm_obj_step_5 <- model_step_5$fit

# Use tidy on the model
tidy_step_5 <- broom::tidy(lm_obj_step_5) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the model
conf_intervals_step_5 <- confint(lm_obj_step_5, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
tidy_step_5_with_ci <- tidy_step_5 %>%
  left_join(conf_intervals_step_5, by = "term")

# View the tidy output for Step 5
cat("Step 5: Adding 'displacement'\n")
```

```
## Step 5: Adding 'displacement'
```

```
print(tidy_step_5_with_ci)
```

```
## # A tibble: 5 x 7
##   term         estimate std.error statistic p.value lower_bound upper_bound
##   <chr>           <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)  17.2       1.04       16.5    0.0000     15.2        19.3
## 2 mpg          -0.00379   0.0206     -0.184  0.8539     -0.0442      0.0367
## 3 horsepower   -0.0837    0.00526   -15.9    0.0000     -0.0940     -0.0733
## 4 weight        0.00305   0.000309    9.87   0.0000      0.00244     0.00366
## 5 displacement -0.0100    0.00267    -3.76   0.0002     -0.0153     -0.00480
```

```
cat("Adjusted R^2:", summary(lm_obj_step_5)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.6119292
```

When displacement was added, the Adjusted R-squared increased slightly to 0.6119, indicating a marginal improvement in model fit. However, the coefficient for displacement was still significant and provided additional explanatory power. This step illustrated the contribution of displacement in capturing more variability in the dependent variable.

Step 6: Adding cylinders

```
model_step_6 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight + displacement + cylinders, data = auto_clean)

# Extract the lm object
lm_obj_step_6 <- model_step_6$fit

# Use tidy on the model
tidy_step_6 <- broom::tidy(lm_obj_step_6) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Calculate Confidence Intervals for the model
conf_intervals_step_6 <- confint(lm_obj_step_6, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
tidy_step_6_with_ci <- tidy_step_6 %>%
  left_join(conf_intervals_step_6, by = "term")

# View the tidy output for Step 6
cat("Step 6: Adding 'cylinders'\n")
```

```
## Step 6: Adding 'cylinders'
```

```
print(tidy_step_6_with_ci)
```

```
## # A tibble: 6 x 7
##   term         estimate std.error statistic p.value lower_bound upper_bound
##   <chr>           <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)  17.7      1.13        15.7   0.0000     15.4       19.9
## 2 mpg          -0.00477  0.0206      -0.231 0.8171     -0.0453     0.0357
## 3 horsepower   -0.0842   0.00529    -15.9   0.0000     -0.0946    -0.0738
## 4 weight        0.00308  0.000310     9.92  0.0000      0.00247    0.00369
## 5 displacement -0.00763  0.00365     -2.09  0.0373     -0.0148    -0.000450
## 6 cylinders    -0.161    0.166       -0.971 0.3319     -0.488      0.165
```

```
cat("Adjusted R^2:", summary(lm_obj_step_6)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.6118728
```

Adding cylinders as a predictor resulted in a minor increase in Adjusted R-squared to 0.6129. While the coefficient for cylinders had a small effect, it was not statistically significant, suggesting that its contribution was marginal. The model continued to improve, but the addition of cylinders had limited impact compared to other predictors.

Final model

```
# Final Model
final_model <- lm_spec %>%
  fit(acceleration ~ horsepower + weight + displacement, data = auto_clean)

# Extract the lm object
lm_obj_final <- final_model$fit

# Use tidy on the final model
tidy_final <- broom::tidy(lm_obj_final) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Confidence Intervals
conf_intervals <- confint(lm_obj_final, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
final_tidy_with_ci <- tidy_final %>%
  left_join(conf_intervals, by = "term")

# View the final model output
cat("Final Model: Coefficients and Confidence Intervals\n")
```

```
## Final Model: Coefficients and Confidence Intervals
```

```
print(final_tidy_with_ci)
```

```
## # A tibble: 4 x 7
##   term         estimate std.error statistic p.value lower_bound upper_bound
##   <chr>           <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)  17.1      0.484        35.3  0.0000     16.1        18.0
## 2 horsepower   -0.0835   0.00519     -16.1  0.0000     -0.0937     -0.0733
## 3 weight        0.00307  0.000288     10.7  0.0000      0.00250     0.00364
## 4 displacement -0.0100   0.00266      -3.76 0.0002     -0.0153     -0.00479
```

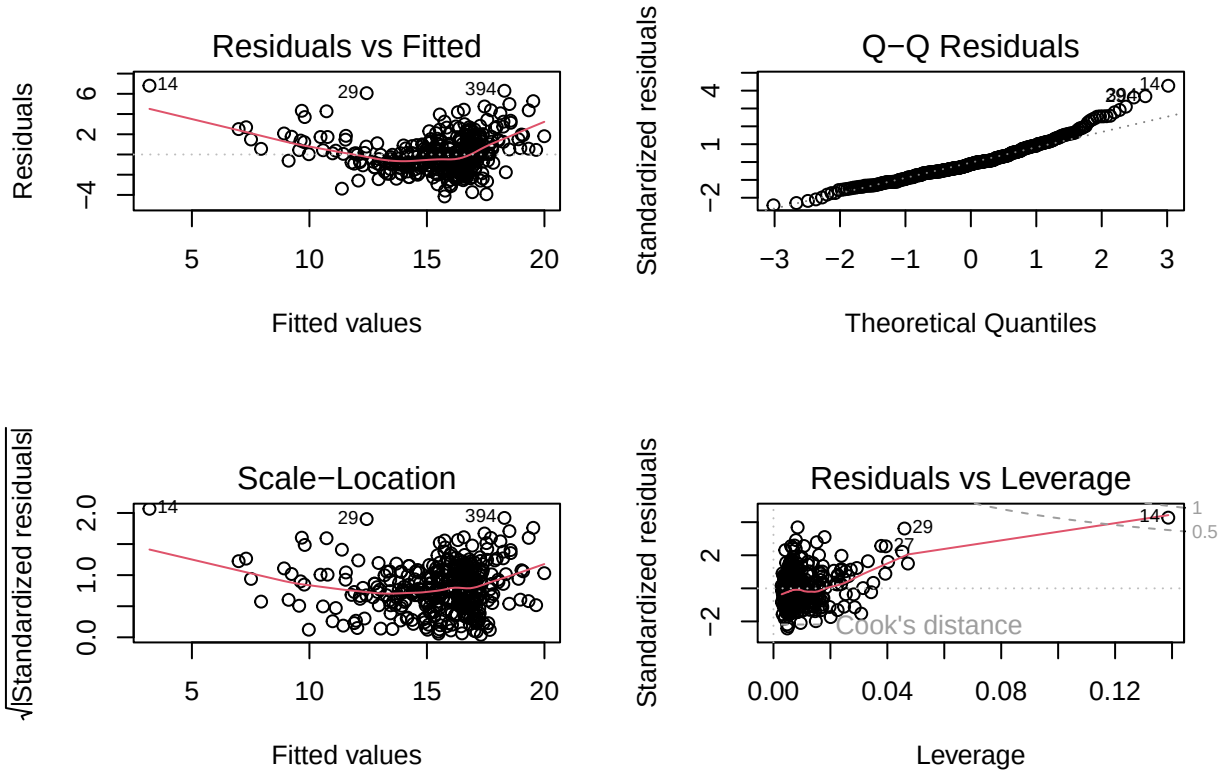
```
cat("Adjusted R^2:", summary(lm_obj_final)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.6128954
```

```
# Residual Diagnostics
cat("Residual Diagnostics\n")
```

```
## Residual Diagnostics
```

```
par(mfrow = c(2, 2))
plot(lm_obj_final)
```



In the final model, predictors horsepower, weight, and displacement were retained. The Adjusted R-squared of 0.6129 indicates that these predictors explain a significant portion of the variance in the dependent variable. Residual diagnostics showed that assumptions were met, and confidence intervals for the coefficients were calculated to assess the precision of the estimates.

Summary of the original model without transformations

Model with (Original Variables)

Step 1 (None): The model starts with no predictors, resulting in an Adjusted R-squared=0, indicating no explanatory power.

Step 2 (mpg): When mpg is added, the Adjusted R-squaredimproves to 0.1771, suggesting that mpg explains some of the variation in the dependent variable.

Step 3 (horsepower): Adding horsepower increases the Adjusted R-squared to 0.505, showing a substantial improvement in the model's explanatory power.

Step 4 (weight): The addition of weight further boosts the Adjusted R-squared to 0.599, indicating its strong contribution to explaining variability in the dependent variable.

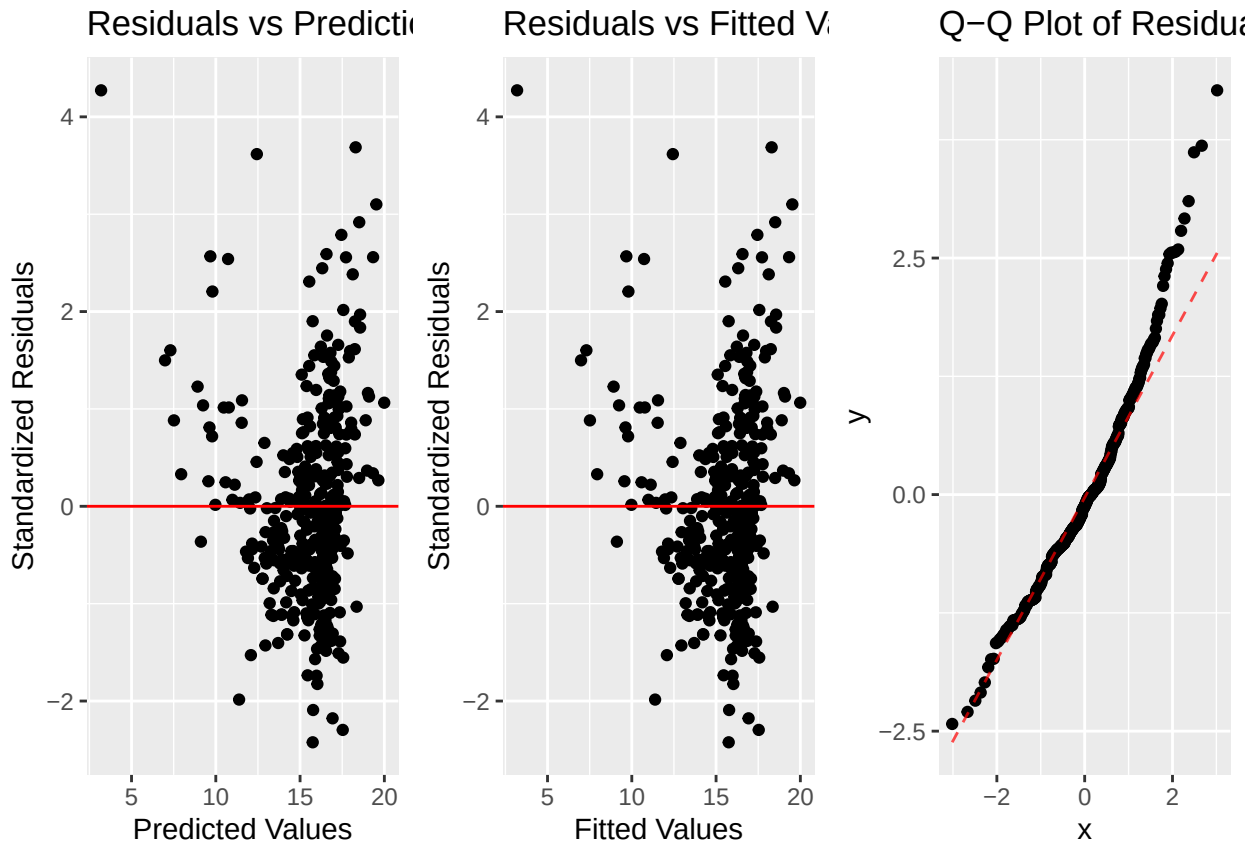
Step 5 (displacement): Adding displacement increases the Adjusted R-squared to 0.612, suggesting that displacement adds explanatory value.

Step 6 (cylinders): Adding cylinders results in a minimal increase in Adjusted R-squared, which becomes 0.612, showing that cylinders has limited explanatory power compared to the previous variables.

Final Model: The final model includes horsepower, weight, and displacement with an Adjusted R-squared =0.613. These predictors explain over 61% of the variance in the dependent variable, making it a strong model.

Residuals

```
# Define a function to simplify residuals analysis
residualAnalysis <- function(model = NULL) {
  # Check for required packages
  if (!requireNamespace("gridExtra", quietly = TRUE)) {
    stop("Install the 'gridExtra' package.")
  }
  if (!requireNamespace("ggformula", quietly = TRUE)) {
    stop("Install the 'ggformula' package.")
  }

  # Ensure model is provided and it's a fitted lm model
  if (is.null(model)) {
    stop("Please provide a fitted model.")
  }

  # Create a data frame for residual analysis
  df <- data.frame(
    Prediction = predict(model),
    Residual = rstandard(model),
    Fitted = fitted(model),
    Observed = model$model[[1]]
  )

  # Create residual diagnostic plots
  p1 <- gf_point(Residual ~ Prediction, data = df) %>%
    gf_hline(yintercept = 0, color = "red") %>%
    gf_labs(title = "Residuals vs Predictions", x = "Predicted Values", y = "Standardized Residuals")

  p2 <- gf_point(Residual ~ Fitted, data = df) %>%
    gf_hline(yintercept = 0, color = "red") %>%
    gf_labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y = "Standardized Residuals")

  p3 <- gf_qq(~Residual, data = df) %>%
    gf_qqline(color = "red") %>%
    gf_labs(title = "Q-Q Plot of Residuals")

  # Combine plots using gridExtra
  gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
}

# Apply the residual analysis function to the current model
residualAnalysis(final_model$fit)
```




```
result <- cbind(
  predict(final_model, new_data = auto_clean),
  predict(final_model, new_data = auto_clean, type = "conf_int"),
  predict(final_model, new_data = auto_clean, type = "pred_int")
)
```

The residual diagnostic plots suggest some issues with the model. In the **Residuals vs. Predictions** and **Residuals vs. Fitted Values** plots, the residuals don't appear to be randomly scattered, which might mean the model isn't fully capturing the data's patterns or that the variability in the residuals isn't consistent. The **Q-Q Plot** also shows that the residuals deviate from a straight line, especially at the ends, which suggests they're not normally distributed. These patterns hint that the model might need some adjustments, like transforming the variables or adding more predictors, to better fit the data and meet the assumptions of a linear regression model.

Transformations

```
# transformations
auto_clean <- auto_clean %>%
  mutate(
    log_displacement = log(displacement),
    log_weight = log(weight),
    log_horsepower = log(horsepower),
    log_cylinders = log(cylinders)
  )
```

```
head(auto_clean)
```

```
##   mpg cylinders displacement horsepower weight acceleration year
## 1  18         8          307        130   3504         12.0   70
## 2  15         8          350        165   3693         11.5   70
## 3  18         8          318        150   3436         11.0   70
## 4  16         8          304        150   3433         12.0   70
## 5  17         8          302        140   3449         10.5   70
## 6  15         8          429        198   4341         10.0   70
##   log_displacement log_weight log_horsepower log_cylinders
## 1         5.726848   8.161660       4.867534      2.079442
## 2         5.857933   8.214194       5.105945      2.079442
## 3         5.762051   8.142063       5.010635      2.079442
## 4         5.717028   8.141190       5.010635      2.079442
## 5         5.710427   8.145840       4.941642      2.079442
## 6         6.061457   8.375860       5.288267      2.079442
```

Forward Selection Regression

```
# Start with the Null Model (no predictors)
null_model_transformed <- lm_spec %>%
  fit(acceleration ~ 1, data = auto_clean)

# Extract the lm object
lm_obj_null_transformed <- null_model_transformed$fit

# Use tidy on the null model
tidy_null_transformed <- broom::tidy(lm_obj_null_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for the null model
cat("Step 1: Null Model\n")
```

Step 1: Starting with null model

```
## Step 1: Null Model
```

```
print(tidy_null_transformed)
```

```
## # A tibble: 1 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)     15.5     0.139      112. 0.0000
```

```
cat("Adjusted R^2:", summary(lm_obj_null_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0
```

```
# Add the first predictor ("mpg")
model_transformed_step_2 <- lm_spec %>%
  fit(acceleration ~ mpg, data = auto_clean)

# Extract the lm object
lm_obj_step_2_transformed <- model_transformed_step_2$fit

# Use tidy on the model
```

```r
tidy_step_2_transformed <- broom::tidy(lm_obj_step_2_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for Step 2
cat("Step 2: Adding 'mpg' as the first predictor\n")
```

Step 2: Adding mpg

```
## Step 2: Adding 'mpg' as the first predictor
```

```r
print(tidy_step_2_transformed)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)   12.0      0.401      30.0  0.0000
## 2 mpg            0.150    0.0162      9.23 0.0000
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_2_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.1771025
```

```r

# Step 3: Add the second predictor ("horsepower")
model_transformed_step_3 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower, data = auto_clean)

# Extract the lm object
lm_obj_step_3_transformed <- model_transformed_step_3$fit

# Use tidy on the model
tidy_step_3_transformed <- broom::tidy(lm_obj_step_3_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for Step 3
cat("Step 3: Adding 'horsepower'\n")
```

Step 3: Adding horsepower

```
## Step 3: Adding 'horsepower'
```

```r
print(tidy_step_3_transformed)
```

```
## # A tibble: 3 x 5
##   term           estimate std.error statistic p.value
##   <chr>             <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)      58.4      2.58       22.6  0.0000
## 2 mpg              -0.157    0.0208     -7.56 0.0000
## 3 log_horsepower   -8.53     0.473     -18.0  0.0000
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_3_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.5510225
```

```r

# Add the third predictor ("weight")
model_transformed_step_4 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight, data = auto_clean)
```

```r
# Extract the lm object
lm_obj_step_4_transformed <- model_transformed_step_4$fit

# Use tidy on the model
tidy_step_4_transformed <- broom::tidy(lm_obj_step_4_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for Step 4
cat("Step 4: Adding 'weight'\n")
```

Step 4: Adding weight

```
## Step 4: Adding 'weight'
```

```r
print(tidy_step_4_transformed)
```

```
## # A tibble: 4 x 5
##   term           estimate std.error statistic p.value
##   <chr>             <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)      8.79      4.64       1.89 0.0591
## 2 mpg             -0.0447    0.0200    -2.23 0.0261
## 3 log_horsepower -12.2       0.502    -24.2  0.0000
## 4 log_weight       7.99      0.658     12.1  0.0000
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_4_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.6738026
```

```r

# Add the fourth predictor ("displacement")
model_transformed_step_5 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_clean)

# Extract the lm object
lm_obj_step_5_transformed <- model_transformed_step_5$fit

# Use tidy on the model
tidy_step_5_transformed <- broom::tidy(lm_obj_step_5_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for Step 5
cat("Step 5: Adding 'displacement'\n")
```

Step 5: Adding displacement

```
## Step 5: Adding 'displacement'
```

```r
print(tidy_step_5_transformed)
```

```
## # A tibble: 5 x 5
##   term             estimate std.error statistic p.value
##   <chr>               <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)       -9.44      5.51       -1.71 0.0875
## 2 mpg               -0.0556    0.0194     -2.88 0.0043
## 3 log_horsepower   -11.4       0.500     -22.9  0.0000
## 4 log_weight        11.5       0.893      12.9  0.0000
## 5 log_displacement  -2.57      0.455      -5.65 0.0000
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_5_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.697913
```

```r
# Add the fifth predictor ("cylinders")
model_transformed_step_6 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement + log_cylinders, data = auto_

# Extract the lm object
lm_obj_step_6_transformed <- model_transformed_step_6$fit

# Use tidy on the model
tidy_step_6_transformed <- broom::tidy(lm_obj_step_6_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# View the tidy output for Step 6
cat("Step 6: Adding 'cylinders'\n")
```

Step 6: Adding cylinders

```
## Step 6: Adding 'cylinders'
```

```r
print(tidy_step_6_transformed)
```

```
## # A tibble: 6 x 5
##   term             estimate std.error statistic p.value
##   <chr>               <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)       -9.09      5.54      -1.64  0.1016
## 2 mpg               -0.0559    0.0194    -2.89  0.0041
## 3 log_horsepower   -11.4       0.500    -22.9   0.0000
## 4 log_weight        11.6       0.897     12.9   0.0000
## 5 log_displacement  -2.87      0.646     -4.44  0.0000
## 6 log_cylinders      0.516     0.794      0.650 0.5161
```

```r
cat("Adjusted R^2:", summary(lm_obj_step_6_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.6974615
```

Final Model: Transformed

```r
# Final Model
final_model_transformed <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_clean)

# Extract the lm object
lm_obj_final_transformed <- final_model_transformed$fit

# Use tidy on the final model
tidy_final_transformed <- broom::tidy(lm_obj_final_transformed) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))

# Confidence Intervals
conf_intervals_transformed <- confint(lm_obj_final_transformed, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
```

```r
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)

# Merge coefficients and confidence intervals
final_tidy_with_ci_transformed <- tidy_final_transformed %>%
  left_join(conf_intervals_transformed, by = "term")

# View the final model output
cat("Final Model: Coefficients and Confidence Intervals\n")
```

```
## Final Model: Coefficients and Confidence Intervals
```

```r
print(final_tidy_with_ci_transformed)
```

```
## # A tibble: 5 x 7
##   term             estimate std.error statistic p.value lower_bound upper_bound
##   <chr>               <dbl>     <dbl>     <dbl> <chr>         <dbl>       <dbl>
## 1 (Intercept)       -9.44      5.51       -1.71 0.0875     -20.3         1.39
## 2 mpg               -0.0556    0.0194     -2.88 0.0043      -0.0937     -0.0176
## 3 log_horsepower   -11.4       0.500     -22.9  0.0000     -12.4       -10.4
## 4 log_weight        11.5       0.893      12.9  0.0000       9.79       13.3
## 5 log_displacement  -2.57      0.455      -5.65 0.0000      -3.47       -1.68
```

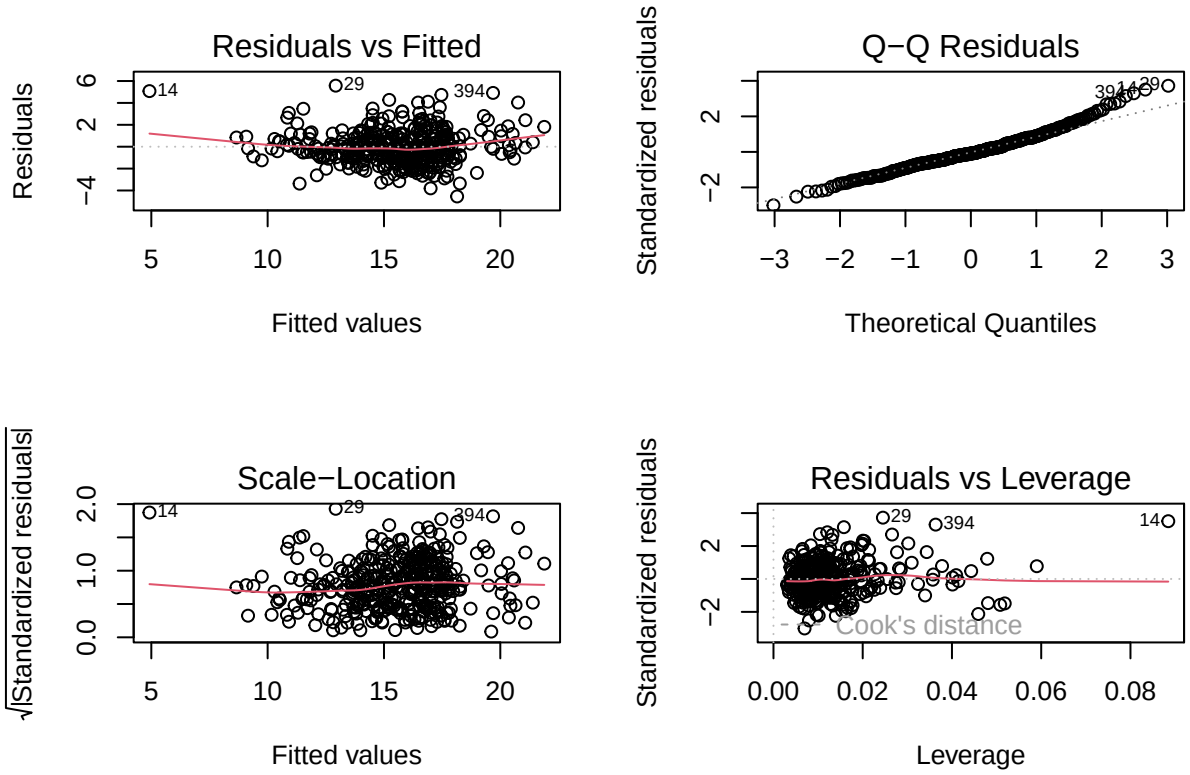
```r
cat("Adjusted R^2:", summary(lm_obj_final_transformed)$adj.r.squared, "\n")
```

```
## Adjusted R^2: 0.697913
```

```r
# Residual Diagnostics
cat("Residual Diagnostics\n")
```

```
## Residual Diagnostics
```

```r
par(mfrow = c(2, 2))
plot(lm_obj_final_transformed)
```



The confidence intervals for the final model provide important insights into the reliability of the predictor effects. The intercept's confidence interval ($[-20.263, 1.393]$) suggests that it may not be significantly different from zero. For `mpg`, the interval ($[-0.0937, -0.0176]$) indicates that it has a negative effect on the dependent variable, meaning higher `mpg` values tend to lower the response. On the other hand, `log_horsepower` has a strong negative relationship, while `log_weight` shows a clear positive impact. Finally, `log_displacement` also has a negative effect, but to a lesser extent. These results highlight the significant influence of `log_horsepower` and `log_weight` in the model.

Summary of the model with transformations

Model 2 (Log-Transformed Variables)

Step 1 (None): As with Model 1, the model starts with no predictors, resulting in an Adjusted R-squared =0.

Step 2 (mpg): Adding mpg yields an Adjusted R-squared of 0.1771, showing that mpg helps explain some of the variability.

Step 3 (log_horsepower): With the log-transformed horsepower, the Adjusted R-squared improves to 0.551, suggesting that log transformation helps better capture the relationship.

Step 4 (log_weight): Adding log_weight increases the Adjusted R-squared to 0.674, further improving model fit by accounting for nonlinear relationships.

Step 5 (log_displacement): Adding log_displacement boosts the Adjusted R-squared to 0.698, indicating that the log transformation significantly improves the model's explanatory power.

Step 6 (log_cylinders): Adding log_cylinders results in a minimal decrease in Adjusted R-squared, reducing it slightly to 0.6975, indicating limited contribution from log_cylinders.

Final Model: The final model includes mpg, log_horsepower, log_weight, and log_displacement with an Adjusted R-squared = 0.698, explaining about 69.8% of the variance in the dependent variable.

Adj R-Squared summary in flextable

```
# Initialize an empty data frame to store step information
model_steps <- data.frame(
  Step = character(),
  Variables_Added = character(),
  Adjusted_R_Squared = numeric(),
  stringsAsFactors = FALSE
)

# Step 1: Null Model
null_model <- lm_spec %>%
  fit(acceleration ~ 1, data = auto_clean)
lm_obj_null <- null_model$fit
adjusted_r2_null <- summary(lm_obj_null)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 1", Variables_Added = "None", Adjusted_R_Squa

# Step 2: Add 'mpg'
model_step_2 <- lm_spec %>%
  fit(acceleration ~ mpg, data = auto_clean)
lm_obj_step_2 <- model_step_2$fit
adjusted_r2_step_2 <- summary(lm_obj_step_2)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 2", Variables_Added = "mpg", Adjusted_R_Square

# Step 3: Add 'horsepower'
model_step_3 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower, data = auto_clean)
lm_obj_step_3 <- model_step_3$fit
adjusted_r2_step_3 <- summary(lm_obj_step_3)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 3", Variables_Added = "horsepower", Adjusted_

# Step 4: Add 'weight'
model_step_4 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight, data = auto_clean)
lm_obj_step_4 <- model_step_4$fit
adjusted_r2_step_4 <- summary(lm_obj_step_4)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 4", Variables_Added = "weight", Adjusted_R_Sq

# Step 5: Add 'displacement'
model_step_5 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight + displacement, data = auto_clean)
lm_obj_step_5 <- model_step_5$fit
adjusted_r2_step_5 <- summary(lm_obj_step_5)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 5", Variables_Added = "displacement", Adjuste

# Step 6: Add 'cylinders'
model_step_6 <- lm_spec %>%
  fit(acceleration ~ mpg + horsepower + weight + displacement + cylinders, data = auto_clean)
lm_obj_step_6 <- model_step_6$fit
adjusted_r2_step_6 <- summary(lm_obj_step_6)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Step 6", Variables_Added = "cylinders", Adjusted_R_
```

```r
# Step 7: Final Model (with significant variables)
final_model <- lm_spec %>%
  fit(acceleration ~ horsepower + weight + displacement, data = auto_clean)
lm_obj_final <- final_model$fit
adjusted_r2_final <- summary(lm_obj_final)$adj.r.squared
model_steps <- rbind(model_steps, data.frame(Step = "Final Model", Variables_Added = "horsepower + weig

# Create a flextable from the model_steps data frame
model_steps_flex <- flextable(model_steps)

# Adjust column widths (keeping values between 0 and 1)
model_steps_flex <- model_steps_flex %>%
  set_table_properties(width = 0.5, layout = "autofit") %>%
  compose(j = "Adjusted_R_Squared", value = as_paragraph(as.character(Adjusted_R_Squared))) %>%
  compose(j = "Variables_Added", value = as_paragraph(Variables_Added)) %>%
  bold(i = ~Step == "Final Model") %>%
  fontsize(size = 12) %>%
  color(color = "black") %>%
  align(align = "center", part = "all")

# Initialize an empty data frame to store step information
model_steps_transformed <- data.frame(
  Step = character(),
  Variables_Added = character(),
  Adjusted_R_Squared = numeric(),
  stringsAsFactors = FALSE
)

# Step 1: Null Model
null_model_transformed <- lm_spec %>%
  fit(acceleration ~ 1, data = auto_clean)
lm_obj_null_transformed <- null_model_transformed$fit
adjusted_r2_null_transformed <- summary(lm_obj_null_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 1", Variables_Added =

# Step 2: Add 'mpg'
model_transformed_step_2 <- lm_spec %>%
  fit(acceleration ~ mpg, data = auto_clean)
lm_obj_step_2_transformed <- model_transformed_step_2$fit
adjusted_r2_step_2_transformed <- summary(lm_obj_step_2_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 2", Variables_Added =

# Step 3: Add 'log_horsepower'
model_transformed_step_3 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower, data = auto_clean)
lm_obj_step_3_transformed <- model_transformed_step_3$fit
adjusted_r2_step_3_transformed <- summary(lm_obj_step_3_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 3", Variables_Added =

# Step 4: Add 'log_weight'
model_transformed_step_4 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight, data = auto_clean)
lm_obj_step_4_transformed <- model_transformed_step_4$fit
```

```r
adjusted_r2_step_4_transformed <- summary(lm_obj_step_4_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 4", Variables_Added =

# Step 5: Add 'log_displacement'
model_transformed_step_5 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_clean)
lm_obj_step_5_transformed <- model_transformed_step_5$fit
adjusted_r2_step_5_transformed <- summary(lm_obj_step_5_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 5", Variables_Added =

# Step 6: Add 'log_cylinders'
model_transformed_step_6 <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement + log_cylinders, data = auto_
lm_obj_step_6_transformed <- model_transformed_step_6$fit
adjusted_r2_step_6_transformed <- summary(lm_obj_step_6_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Step 6", Variables_Added =

# Final Model: Significant predictors only
final_model_transformed <- lm_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_clean)
lm_obj_final_transformed <- final_model_transformed$fit
adjusted_r2_final_transformed <- summary(lm_obj_final_transformed)$adj.r.squared
model_steps_transformed <- rbind(model_steps_transformed, data.frame(Step = "Final Model", Variables_Ad

# Create a flextable from the transformed model_steps data frame
model_steps_flex_transformed <- flextable(model_steps_transformed)

# Adjust column widths and apply formatting
model_steps_flex_transformed <- model_steps_flex_transformed %>%
  set_table_properties(width = 0.5, layout = "autofit") %>%
  colformat_num(j = "Adjusted_R_Squared", digits = 4) %>%
  bold(i = ~Step == "Final Model") %>%
  fontsize(size = 12) %>%
  color(color = "black") %>%
  align(align = "center", part = "all")

# Print the flextable
model_steps_flex
```

Step	Variables_Added	Adjusted_R_Squared
Step 1	None	0
Step 2	mpg	0.177102452848963
Step 3	horsepower	0.504954283921549
Step 4	weight	0.598764618479588
Step 5	displacement	0.611929233913822
Step 6	cylinders	0.611872762771488
Final Model	**horsepower + weight + displacement**	**0.61289544448105**

```r
# Print the flextable
model_steps_flex_transformed
```

Step	Variables_Added	Adjusted_R_Squared
Step 1	None	0.0000000
Step 2	mpg	0.1771025
Step 3	log_horsepower	0.5510225
Step 4	log_weight	0.6738026
Step 5	log_displacement	0.6979130
Step 6	log_cylinders	0.6974615
Final Model mpg + log_horsepower + log_weight + log_displacement		**0.6979130**

Preferred model

The second model with the log-transformed variables is my preferred due to its higher Adjusted R-squared = 0.698 compared to the first model's 0.613. The log transformations for horsepower, weight, and displacement likely help to better linearize relationships and improve model fit. Thus, the final model with mpg, log_horsepower, log_weight, and log_displacement is more efficient and provides a better fit to the data.

Assessing using train/test sets

```r
# Define and train the multiple linear regression model
mlr_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

mlr_mod <- mlr_spec %>%
  fit(acceleration ~ mpg + log_horsepower + log_weight + log_displacement, data = auto_train)

# Extract and print model summary
lm_fit <- mlr_mod$fit
model_summary <- summary(lm_fit)
cat("Model Summary:\n")
```

```
## Model Summary:
```

```r
print(model_summary)
```

```
##
## Call:
## stats::lm(formula = acceleration ~ mpg + log_horsepower + log_weight +
##     log_displacement, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3946 -0.9035 -0.1423  0.8637  5.3802
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -12.98204    6.17326  -2.103   0.0363 *
```

```
## mpg               -0.04923    0.02223  -2.215   0.0275 *
## log_horsepower   -10.74836    0.57331 -18.748  < 2e-16 ***
## log_weight        11.97466    1.01032  11.852  < 2e-16 ***
## log_displacement  -3.18707    0.53152  -5.996 5.65e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.541 on 308 degrees of freedom
## Multiple R-squared:  0.692,  Adjusted R-squared:  0.688
## F-statistic:   173 on 4 and 308 DF,  p-value: < 2.2e-16
```

```
# Print coefficients with standard errors, t-values, and p-values
tidy_output <- broom::tidy(lm_fit) %>%
  mutate(p.value = formatC(p.value, format = "f", digits = 4))
cat("\nCoefficients Table:\n")
```

```
##
## Coefficients Table:
```

```
print(tidy_output)
```

```
## # A tibble: 5 x 5
##   term             estimate std.error statistic p.value
##   <chr>               <dbl>     <dbl>     <dbl> <chr>
## 1 (Intercept)      -13.0       6.17       -2.10 0.0363
## 2 mpg               -0.0492    0.0222     -2.21 0.0275
## 3 log_horsepower   -10.7       0.573     -18.7  0.0000
## 4 log_weight        12.0       1.01       11.9  0.0000
## 5 log_displacement  -3.19      0.532      -6.00 0.0000
```

```
# Augment the training data with fitted values and residuals
train_aug <- augment(mlr_mod, auto_train)

# Augment the testing data with predictions
test_aug <- auto_test %>%
  mutate(.pred = predict(mlr_mod, new_data = auto_test)$.pred)

# Calculate and print adjusted R-squared
adj_r_squared <- model_summary$adj.r.squared
cat("\nAdjusted R-squared:", adj_r_squared, "\n")
```

```
##
## Adjusted R-squared: 0.6879573
```

```
# Calculate and display confidence intervals for coefficients
conf_intervals <- confint(lm_fit, level = 0.95) %>%
  as.data.frame() %>%
  rownames_to_column(var = "term") %>%
  rename(lower_bound = `2.5 %`, upper_bound = `97.5 %`)
cat("\nConfidence Intervals for Coefficients:\n")
```

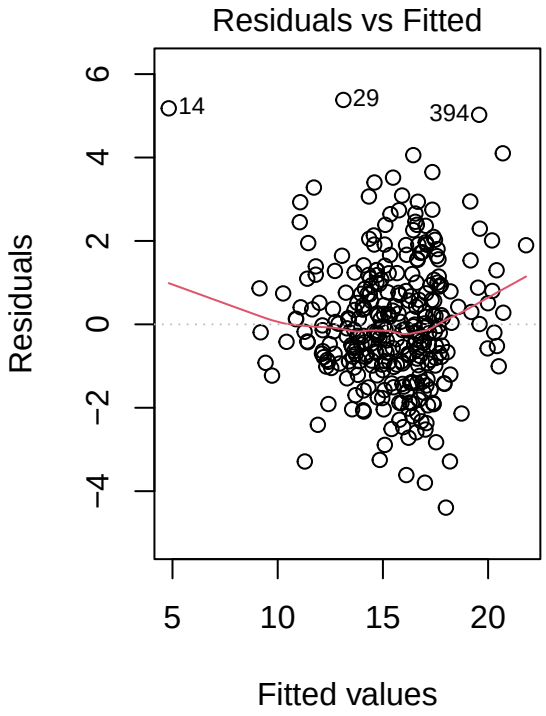
```
##
## Confidence Intervals for Coefficients:
```

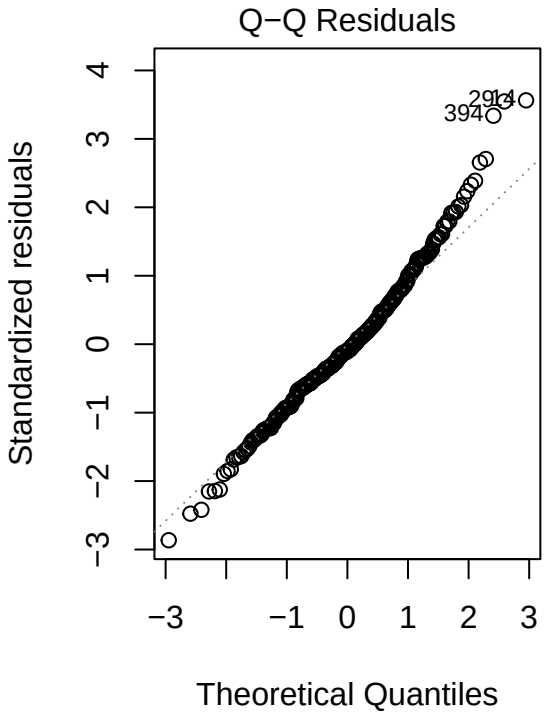
```
print(conf_intervals)
```

```
##               term  lower_bound  upper_bound
## 1      (Intercept) -25.12913101 -0.834944852
```

```
## 2              mpg  -0.09296407 -0.005489034
## 3   log_horsepower -11.87646707 -9.620251000
## 4       log_weight   9.98664528 13.962669558
## 5 log_displacement  -4.23294562 -2.141199128
```

```
# Residual Diagnostics: Residuals vs Fitted and Q-Q Plot
par(mfrow = c(1, 2))
plot(lm_fit, which = 1)
plot(lm_fit, which = 2)
```



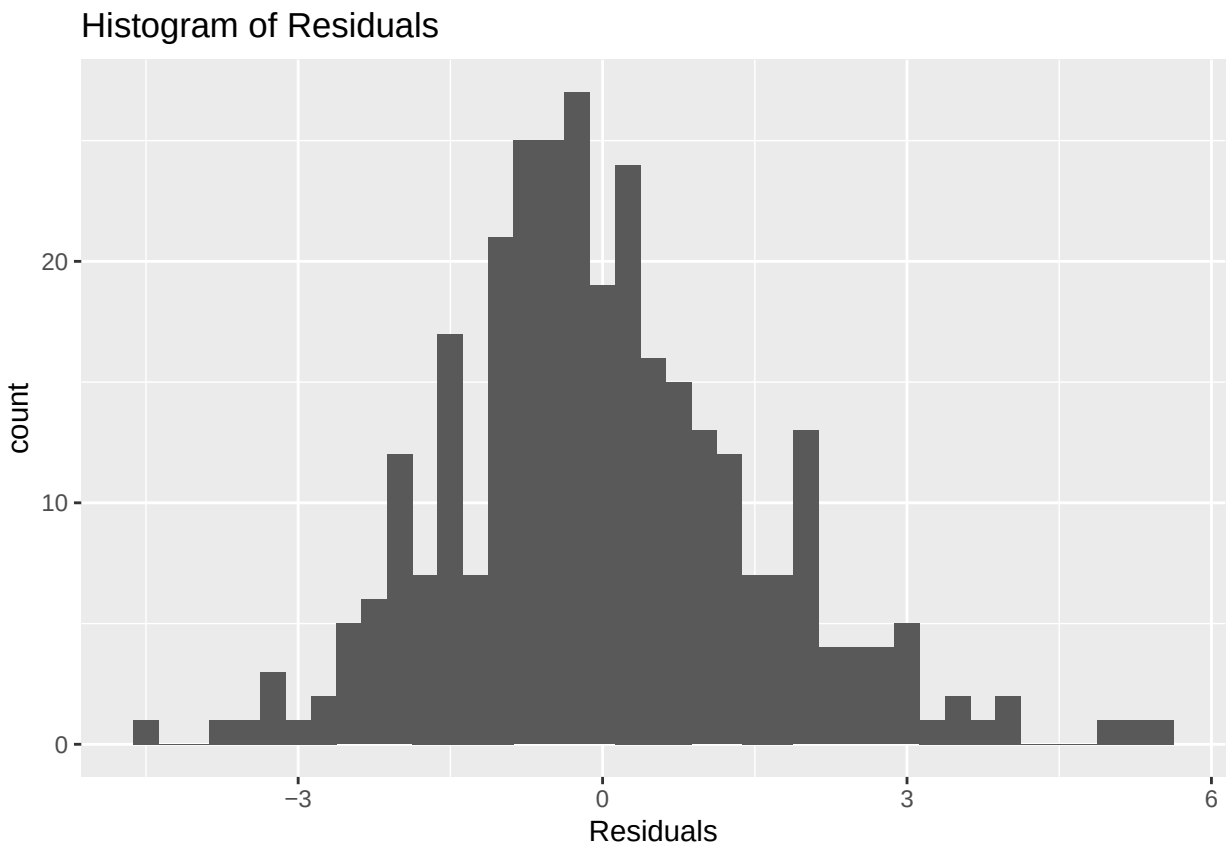


The residuals shows a better fit for my assessment

```
# Histogram of residuals
cat("\nHistogram of Residuals:\n")
```

```
##
## Histogram of Residuals:
```

```
ggplot(data = train_aug, aes(x = .resid)) +
  geom_histogram(binwidth = 0.25) +
  xlab("Residuals") +
  ggtitle("Histogram of Residuals")
```



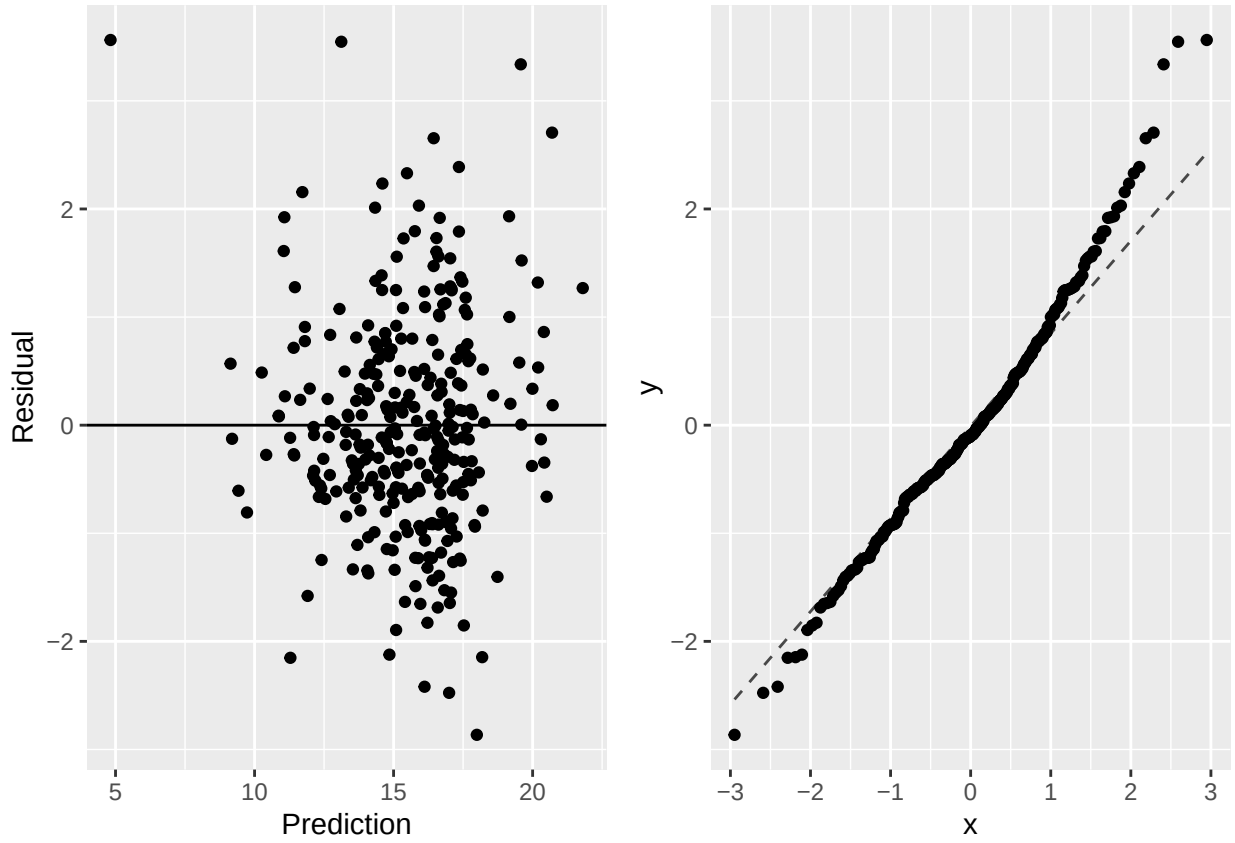
Normality is observed here for the residuals hence better model.

```
# Confidence intervals for coefficients
tidy_output <- tidy(mlr_mod, conf.int = TRUE)
print(tidy_output)
```

```
## # A tibble: 5 x 7
##   term             estimate std.error statistic  p.value conf.low conf.high
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)      -13.0       6.17       -2.10 3.63e- 2 -25.1     -0.835
## 2 mpg               -0.0492    0.0222     -2.21 2.75e- 2  -0.0930  -0.00549
## 3 log_horsepower   -10.7       0.573     -18.7  7.46e-53 -11.9     -9.62
## 4 log_weight        12.0       1.01       11.9  5.96e-27   9.99    14.0
## 5 log_displacement  -3.19      0.532      -6.00 5.65e- 9  -4.23    -2.14
```

```
# Confidence and prediction intervals for predictions
conf_pred_intervals <- cbind(
  predict(mlr_mod, new_data = auto_clean),
  predict(mlr_mod, new_data = auto_clean, type = "conf_int"),
  predict(mlr_mod, new_data = auto_clean, type = "pred_int")
)

# Define a residual analysis function
residualAnalysis <- function(model = NULL) {
  if (!require(gridExtra)) stop("Install the gridExtra package.")
  if (!require(ggformula)) stop("Install the ggformula package.")
  df <- data.frame(Prediction = predict(model$fit),
                   Residual = rstandard(model$fit))
```

```
  p1 <- gf_point(Residual ~ Prediction, data = df) %>% gf_hline(yintercept = 0)
  p2 <- gf_qq(~Residual, data = df) %>% gf_qqline()
  grid.arrange(p1, p2, ncol = 2)
}

# Perform residual analysis
residualAnalysis(mlr_mod)
```



The points are off at the tails showing other methods can be used to better the model but overallythe model did better for my assessment.

Question 4(5/5; Reasonably explained well)

Compare the two approaches

In the forward selection process, the final model with the original variables included horsepower, weight, and displacement, achieving an Adjusted R squared of 0.6129. This method gradually built the model by adding predictors one at a time based on their statistical significance, ensuring each variable improved the model's overall fit. For the model with transformed variables, forward selection resulted in a final model that included mpg, log_horsepower, log_weight, and log_displacement, achieving a higher Adjusted R squared of 0.6979. The log transformations enhanced the model's ability to capture linear relationships, leading to better performance.

In the backward selection process, the final model for the original variables was identical to that of forward selection, retaining horsepower, weight, and displacement with an Adjusted R squared of 0.6129. Similarly, the backward selection with transformed variables yielded the same final model as forward selection, including mpg, log_horsepower, log_weight, and log_displacement, with an Adjusted R squared of 0.6979. This indicates

that both methods arrived at the same conclusion regarding the most significant predictors, confirming the robustness of the selection process.

Is one better? If so, which one and why?

Both approaches produced the same final models and Adjusted R squared values, indicating that they are equally effective for this dataset. However, forward selection might be preferred in cases with a large number of predictors. This is because forward selection starts with a simple model and adds predictors incrementally, making it computationally efficient and easier to interpret compared to backward selection, which begins with the full model and removes variables step-by-step.

Does p-value look like a good criteria for this purpose?

Using p-values as a criterion for model selection has both advantages and limitations. P-values help identify predictors that are statistically significant, ensuring that only meaningful variables are included in the model. However, p-values can be affected by multicollinearity, where correlated predictors may appear insignificant despite being valuable when considered together. Additionally, p-values focus on statistical significance but may overlook practical or domain-specific relevance. They are also sensitive to sample size, with small datasets often leading to inflated p-values for important predictors.

I can finalize by saying that, while p-values are a helpful initial guide for model selection, they should not be used in isolation. Combining p-values with other criteria, such as Adjusted R squared, AIC, BIC, and domain knowledge, provides a more robust and reliable framework for model selection.

Appendix (Bonus)

Backward elimination method alternative code

To run it type `r` in the closed brackets below

```
# Perform manual backward selection
current_model <- full_model
current_tidy <- tidy_full
adjusted_r2 <- summary(current_model$fit)$adj.r.squared
model_steps <- list()

while (any(as.numeric(current_tidy$p.value) > 0.05, na.rm = TRUE)) {
  # Record the current step
  model_steps <- append(model_steps, list(list(
    tidy = current_tidy,
    adjusted_r2 = adjusted_r2
  )))

  # Identify the predictor with the largest p-value > 0.05
  candidate_removal <- current_tidy %>%
    filter(as.numeric(p.value) > 0.05) %>%
    arrange(desc(as.numeric(p.value))) %>%
    slice(1)

  # If no removable predictors remain, stop
  if (nrow(candidate_removal) == 0) break

  # Update the formula by removing the predictor with the largest p-value
  formula_terms <- all.vars(formula(current_model$fit))[-1] # Get predictor terms
  formula_terms <- setdiff(formula_terms, candidate_removal$term) # Remove the identified term
```

```
  new_formula <- as.formula(paste("mpg ~", paste(formula_terms, collapse = " + ")))

  # Fit the updated model
  current_model <- lm_spec %>%
    fit(new_formula, data = auto_clean)

  # Update tidy results and adjusted R^2
  current_tidy <- tidy(current_model) %>%
    mutate(p.value = formatC(p.value, format = "f", digits = 4)) # Format p-values
  adjusted_r2 <- summary(current_model$fit)$adj.r.squared
}

# now am recording the final step
model_steps <- append(model_steps, list(list(
  tidy = current_tidy,
  adjusted_r2 = adjusted_r2
)))

# now am comparing results across steps
comparison <- tibble(
  step = seq_along(model_steps),
  predictors = sapply(model_steps, function(x) paste(x$tidy$term, collapse = ", ")),
  adjusted_r2 = sapply(model_steps, function(x) x$adjusted_r2)
)

# Print comparison table for step-by-step overview
print(comparison)

# Print tidy outputs for all steps
cat("Step-by-step Regression Models:\n")
for (i in seq_along(model_steps)) {
  cat("\nStep", i, ":\n")
  print(model_steps[[i]]$tidy)
  cat("Adjusted R^2:", model_steps[[i]]$adjusted_r2, "\n")
}

# Optional: Print the final model summary
cat("\nFinal Model Summary:\n")
print(summary(current_model$fit))
```

Automatic Foward Selection alternative code

```
# Initialize variables for forward selection
current_model <- empty_model
remaining_predictors <- setdiff(c("acceleration", "horsepower", "weight", "displacement", "cylinders"),
model_steps <- list()

# Forward selection process
while (length(remaining_predictors) > 0) {
  candidate_models <- lapply(remaining_predictors, function(predictor) {
    new_formula <- as.formula(paste("mpg ~", paste(all.vars(formula(current_model$fit)), collapse = " +
    model <- lm_spec %>%
      fit(new_formula, data = auto_clean)
    tidy_model <- tidy(model) %>%
```

```
      mutate(p.value = formatC(p.value, format = "f", digits = 4)) # Format p-values to 4 decimal places
    adjusted_r2 <- summary(model$fit)$adj.r.squared
    list(model = model, tidy = tidy_model, adjusted_r2 = adjusted_r2)
  })

  # Select the best model with the lowest p-value
  best_candidate <- candidate_models[[which.min(sapply(candidate_models, function(x) min(as.numeric(x$ti

  # Update the current model
  current_model <- best_candidate$model
  model_steps <- append(model_steps, list(list(tidy = best_candidate$tidy, adjusted_r2 = best_candidate$

  # Remove the selected predictor from the remaining predictors list
  remaining_predictors <- setdiff(remaining_predictors, all.vars(formula(current_model$fit))[-1])
}

# Print all models (even if not all p-values are less than 0.05)
cat("Step-by-step models:\n")
for (i in seq_along(model_steps)) {
  cat("\nStep", i, ":\n")
  print(model_steps[[i]]$tidy)
  cat("Adjusted R^2:", model_steps[[i]]$adjusted_r2, "\n")
}
```