Car Price Forecast

[...]

2023/5/21

Contents

Introduction

Problem Description

Geely Auto, a Chinese automobile company, aims to penetrate the US market by establishing a local manufacturing unit and producing cars to compete with American and European counterparts. To gain insight into the pricing factors specific to the American market, Geely Auto has enlisted the services of an automobile consulting company. The consulting company's objective is twofold:

- Identify the significant variables that influence car prices.
- Assess the effectiveness of these variables in describing the price of a car.

To achieve these goals, the consulting firm has compiled a comprehensive dataset of various car types available in the American market, drawing from numerous market surveys.

Business Objective

My objective is to develop a pricing model for cars based on various independent variables. This model will enable the management to gain a clear understanding of how the prices of cars are influenced by these independent variables. With this knowledge, the management can make informed decisions regarding the design of the cars, the business strategy, and other relevant factors to achieve specific price levels. Additionally, the model will serve as a valuable tool for the management to comprehend the pricing dynamics in a new market.

Dataset Description

The dataset contains information about cars and their respective attributes. Here is a summary of the data:

1. Car_ID: A unique identifier for each observation (integer).
2. Symboling: The assigned insurance risk rating, ranging from -3 (probably pretty safe) to +3 (risky) (categorical).
3. CarCompany: The name of the car company (categorical).
4. Fueltype: The type of car fuel, either gas or diesel (categorical).
5. Aspiration: The type of aspiration used in a car (categorical).
6. Doornumber: The number of doors in a car (categorical).
7. Carbody: The body type of the car (categorical).
8. Drivewheel: The type of drive wheel (categorical).
9. Enginelocation: The location of the car engine (categorical).
10. Wheelbase: The wheelbase of the car (numeric).
11. Carlength: The length of the car (numeric).
12. Carwidth: The width of the car (numeric).
13. Carheight: The height of the car (numeric).
14. Curbweight: The weight of the car without occupants or baggage (numeric).
15. Enginetype: The type of engine (categorical).
16. Cylindernumber: The number of cylinders in the car (categorical).
17. Enginesize: The size of the car's engine (numeric).
18. Fuelsystem: The fuel system of the car (categorical).
19. Boreratio: The bore ratio of the car (numeric).
20. Stroke: The stroke or volume inside the engine (numeric).
21. Compressionratio: The compression ratio of the car (numeric).
22. Horsepower: The horsepower of the car (numeric).
23. Peakrpm: The peak revolutions per minute (rpm) of the car (numeric).
24. Citympg: The mileage in miles per gallon (mpg) in city driving conditions (numeric).

25. Highwaympg: The mileage in mpg on the highway (numeric).
26. Price (Dependent variable): The price of the car (numeric).

These attributes provide information about various aspects of the cars, such as their specifications, dimensions, performance, and pricing.

Step 1: Data Reading and Understanding

To begin, we will follow these steps:

1. Import the basic library to work with the data.
2. Read the dataset and load it into a pandas DataFrame.
3. Gain an understanding of the data's structure and format.

By performing these initial steps, we can proceed with further analysis and exploration of the dataset.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Reading the data
cars <- read.csv('CarPrice_Assignment.csv')

# Displaying the first few rows of the data
head(cars)
```

```
##   car_ID symboling                  CarName fueltype aspiration doornumber
## 1      1         3       alfa-romero giulia      gas        std        two
## 2      2         3      alfa-romero stelvio      gas        std        two
## 3      3         1 alfa-romero Quadrifoglio      gas        std        two
## 4      4         2              audi 100 ls      gas        std       four
## 5      5         2               audi 100ls      gas        std       four
## 6      6         2                 audi fox      gas        std        two
##       carbody drivewheel enginelocation wheelbase carlength carwidth carheight
## 1 convertible        rwd          front      88.6     168.8     64.1      48.8
## 2 convertible        rwd          front      88.6     168.8     64.1      48.8
## 3   hatchback        rwd          front      94.5     171.2     65.5      52.4
## 4       sedan        fwd          front      99.8     176.6     66.2      54.3
## 5       sedan        4wd          front      99.4     176.6     66.4      54.3
## 6       sedan        fwd          front      99.8     177.3     66.3      53.1
##   curbweight enginetype cylindernumber enginesize fuelsystem boreratio stroke
## 1       2548       dohc           four        130       mpfi      3.47   2.68
## 2       2548       dohc           four        130       mpfi      3.47   2.68
```

```
## 3       2823       ohcv           six        152       mpfi      2.68   3.47
## 4       2337        ohc          four        109       mpfi      3.19   3.40
## 5       2824        ohc          five        136       mpfi      3.19   3.40
## 6       2507        ohc          five        136       mpfi      3.19   3.40
##   compressionratio horsepower peakrpm citympg highwaympg price
## 1              9.0        111    5000      21         27 13495
## 2              9.0        111    5000      21         27 16500
## 3              9.0        154    5000      19         26 16500
## 4             10.0        102    5500      24         30 13950
## 5              8.0        115    5500      18         22 17450
## 6              8.5        110    5500      19         25 15250
```

```
print(dim(cars))
```

```
## [1] 205  26
```

```
summary(cars)
```

```
##      car_ID      symboling         CarName            fueltype        
##  Min.   :  1   Min.   :-2.0000   Length:205         Length:205        
##  1st Qu.: 52   1st Qu.: 0.0000   Class :character   Class :character  
##  Median :103   Median : 1.0000   Mode  :character   Mode  :character  
##  Mean   :103   Mean   : 0.8341                                        
##  3rd Qu.:154   3rd Qu.: 2.0000                                        
##  Max.   :205   Max.   : 3.0000                                        
##   aspiration         doornumber          carbody           drivewheel       
##  Length:205         Length:205         Length:205         Length:205        
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##  enginelocation       wheelbase        carlength        carwidth    
##  Length:205         Min.   : 86.60   Min.   :141.1   Min.   :60.30  
##  Class :character   1st Qu.: 94.50   1st Qu.:166.3   1st Qu.:64.10  
##  Mode  :character   Median : 97.00   Median :173.2   Median :65.50  
##                     Mean   : 98.76   Mean   :174.0   Mean   :65.91  
##                     3rd Qu.:102.40   3rd Qu.:183.1   3rd Qu.:66.90  
##                     Max.   :120.90   Max.   :208.1   Max.   :72.30  
##    carheight       curbweight    enginetype        cylindernumber    
##  Min.   :47.80   Min.   :1488   Length:205         Length:205        
##  1st Qu.:52.00   1st Qu.:2145   Class :character   Class :character  
##  Median :54.10   Median :2414   Mode  :character   Mode  :character  
##  Mean   :53.72   Mean   :2556                                        
##  3rd Qu.:55.50   3rd Qu.:2935                                        
##  Max.   :59.80   Max.   :4066                                        
##    enginesize     fuelsystem          boreratio        stroke     
##  Min.   : 61.0   Length:205         Min.   :2.54   Min.   :2.070  
##  1st Qu.: 97.0   Class :character   1st Qu.:3.15   1st Qu.:3.110  
##  Median :120.0   Mode  :character   Median :3.31   Median :3.290  
##  Mean   :126.9                      Mean   :3.33   Mean   :3.255  
##  3rd Qu.:141.0                      3rd Qu.:3.58   3rd Qu.:3.410  
##  Max.   :326.0                      Max.   :3.94   Max.   :4.170  
##  compressionratio   horsepower       peakrpm        citympg     
```

```
##  Min.   : 7.00    Min.   : 48.0   Min.   :4150   Min.   :13.00
##  1st Qu.: 8.60    1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00
##  Median : 9.00    Median : 95.0   Median :5200   Median :24.00
##  Mean   :10.14    Mean   :104.1   Mean   :5125   Mean   :25.22
##  3rd Qu.: 9.40    3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00
##  Max.   :23.00    Max.   :288.0   Max.   :6600   Max.   :49.00
##    highwaympg        price
##  Min.   :16.00   Min.   : 5118
##  1st Qu.:25.00   1st Qu.: 7788
##  Median :30.00   Median :10295
##  Mean   :30.75   Mean   :13277
##  3rd Qu.:34.00   3rd Qu.:16503
##  Max.   :54.00   Max.   :45400
```

```
str(cars)
```

```
## 'data.frame':    205 obs. of  26 variables:
##  $ car_ID          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ symboling       : int  3 3 1 2 2 2 1 1 1 0 ...
##  $ CarName         : chr  "alfa-romero giulia" "alfa-romero stelvio" "alfa-romero Quadrifoglio" "audi 1
##  $ fueltype        : chr  "gas" "gas" "gas" "gas" ...
##  $ aspiration      : chr  "std" "std" "std" "std" ...
##  $ doornumber      : chr  "two" "two" "two" "four" ...
##  $ carbody         : chr  "convertible" "convertible" "hatchback" "sedan" ...
##  $ drivewheel      : chr  "rwd" "rwd" "rwd" "fwd" ...
##  $ enginelocation  : chr  "front" "front" "front" "front" ...
##  $ wheelbase       : num  88.6 88.6 94.5 99.8 99.4 ...
##  $ carlength       : num  169 169 171 177 177 ...
##  $ carwidth        : num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...
##  $ carheight       : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...
##  $ curbweight      : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...
##  $ enginetype      : chr  "dohc" "dohc" "ohcv" "ohc" ...
##  $ cylindernumber  : chr  "four" "four" "six" "four" ...
##  $ enginesize      : int  130 130 152 109 136 136 136 136 131 131 ...
##  $ fuelsystem      : chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
##  $ boreratio       : num  3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.13 ...
##  $ stroke          : num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 3.4 ...
##  $ compressionratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...
##  $ horsepower      : int  111 111 154 102 115 110 110 110 140 160 ...
##  $ peakrpm         : int  5000 5000 5000 5500 5500 5500 5500 5500 5500 5500 ...
##  $ citympg         : int  21 21 19 24 18 19 19 19 17 16 ...
##  $ highwaympg      : int  27 27 26 30 22 25 25 25 20 22 ...
##  $ price           : num  13495 16500 16500 13950 17450 ...
```

This is the basic information of the data.

Step 2 : Data Cleaning and Preparation

```
# Splitting company name from CarName column
cars <- cars %>%
  mutate(CompanyName = str_split_fixed(CarName, " ", 2)[,1])
```

```r
cars <- subset(cars, select = -CarName)

print(head(cars))
```

```
##   car_ID symboling fueltype aspiration doornumber     carbody drivewheel
## 1      1         3      gas        std        two convertible        rwd
## 2      2         3      gas        std        two convertible        rwd
## 3      3         1      gas        std        two   hatchback        rwd
## 4      4         2      gas        std       four       sedan        fwd
## 5      5         2      gas        std       four       sedan        4wd
## 6      6         2      gas        std        two       sedan        fwd
##   enginelocation wheelbase carlength carwidth carheight curbweight enginetype
## 1          front      88.6     168.8     64.1      48.8       2548       dohc
## 2          front      88.6     168.8     64.1      48.8       2548       dohc
## 3          front      94.5     171.2     65.5      52.4       2823       ohcv
## 4          front      99.8     176.6     66.2      54.3       2337        ohc
## 5          front      99.4     176.6     66.4      54.3       2824        ohc
## 6          front      99.8     177.3     66.3      53.1       2507        ohc
##   cylindernumber enginesize fuelsystem boreratio stroke compressionratio
## 1           four        130       mpfi      3.47   2.68              9.0
## 2           four        130       mpfi      3.47   2.68              9.0
## 3            six        152       mpfi      2.68   3.47              9.0
## 4           four        109       mpfi      3.19   3.40             10.0
## 5           five        136       mpfi      3.19   3.40              8.0
## 6           five        136       mpfi      3.19   3.40              8.5
##   horsepower peakrpm citympg highwaympg price CompanyName
## 1        111    5000      21         27 13495 alfa-romero
## 2        111    5000      21         27 16500 alfa-romero
## 3        154    5000      19         26 16500 alfa-romero
## 4        102    5500      24         30 13950        audi
## 5        115    5500      18         22 17450        audi
## 6        110    5500      19         25 15250        audi
```

```r
print(unique(cars$CompanyName))
```

```
##  [1] "alfa-romero" "audi"        "bmw"         "chevrolet"   "dodge"
##  [6] "honda"       "isuzu"       "jaguar"      "maxda"       "mazda"
## [11] "buick"       "mercury"     "mitsubishi"  "Nissan"      "nissan"
## [16] "peugeot"     "plymouth"    "porsche"     "porcshce"    "renault"
## [21] "saab"        "subaru"      "toyota"      "toyouta"     "vokswagen"
## [26] "volkswagen"  "vw"          "volvo"
```

Correcting Invalid Values:

There are some spelling errors in the "CompanyName" column that need to be fixed. Here are the corrections:

- maxda should be corrected to mazda.
- Nissan should be corrected to nissan.
- porsche should be corrected to porsche.
- toyota should be corrected to toyota.
- vokswagen should be corrected to volkswagen or vw.

Please note that "volkswagen" and "vw" both refer to the same company.

```r
# Converting CompanyName to lowercase
cars <- mutate(cars, CompanyName = tolower(CompanyName))

# Fixing misspelled names
replace_name <- function(a, b) {
  cars$CompanyName[cars$CompanyName == a] <- b
}

replace_name('maxda', 'mazda')
replace_name('porcshce', 'porsche')
replace_name('toyouta', 'toyota')
replace_name('vokswagen', 'volkswagen')
replace_name('vw', 'volkswagen')



# Checking for duplicates
print(cars[duplicated(cars), ])
```

```
##  [1] car_ID           symboling        fueltype         aspiration
##  [5] doornumber       carbody          drivewheel       enginelocation
##  [9] wheelbase        carlength        carwidth         carheight
## [13] curbweight       enginetype       cylindernumber   enginesize
## [17] fuelsystem       boreratio        stroke           compressionratio
## [21] horsepower       peakrpm          citympg          highwaympg
## [25] price            CompanyName
## <0 > ( 0- row.names)
```

Visualization

Visualising dependent variable

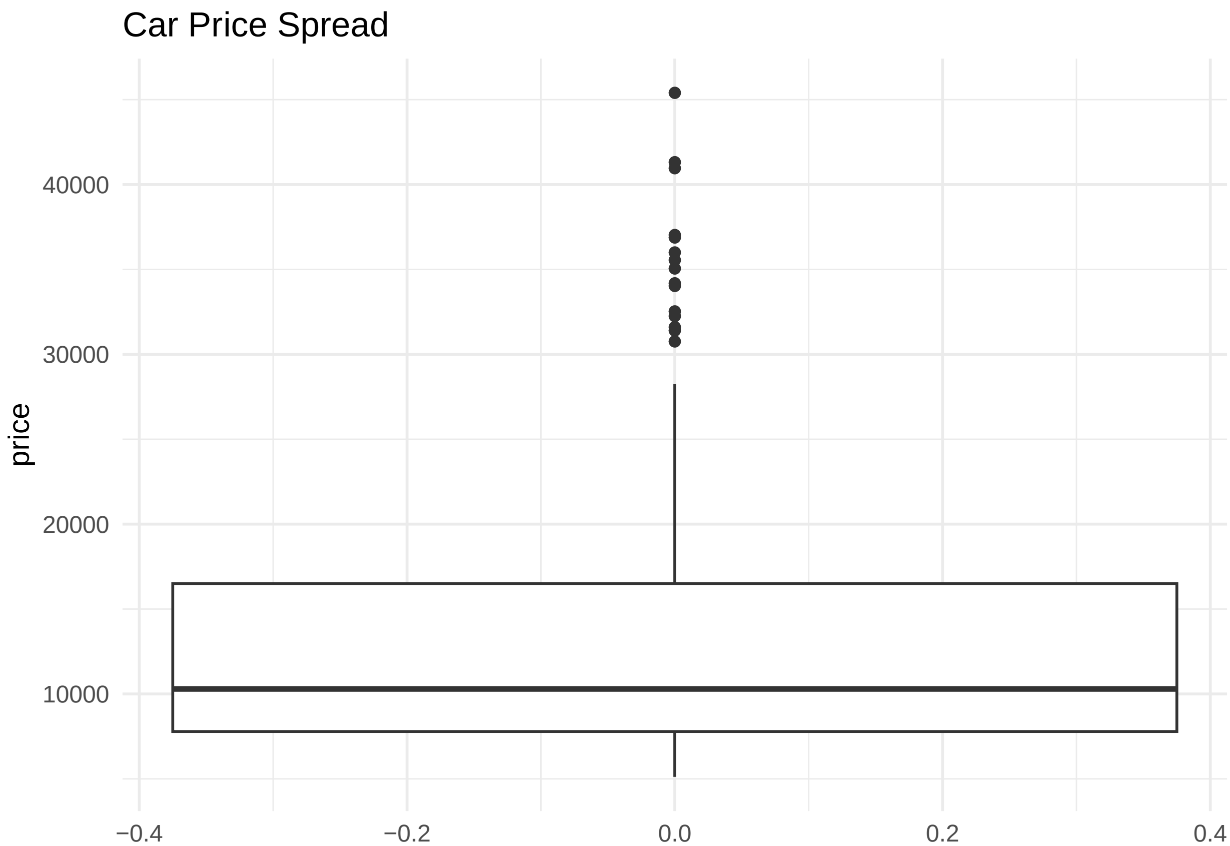
```r
library(ggplot2)

# Car Price Distribution Plot
p1 <- ggplot(cars, aes(x = price)) +
  geom_histogram(aes(y = ..density..), colour = "black", fill = "white") +
  geom_density(alpha = .2, fill = "#FF6666") +
  ggtitle("Car Price Distribution Plot") +
  theme_minimal()
print(p1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


```r
print(quantile(cars$price, c(0.25, 0.50, 0.75, 0.85, 0.90, 1)))
```

```
##   25%   50%   75%   85%   90%  100%
##  7788 10295 16503 18500 22563 45400
```

```r
# Car Price Spread
p2 <- ggplot(cars, aes(y = price)) +
  geom_boxplot() +
  ggtitle("Car Price Spread") +
  theme_minimal()
print(p2)
```

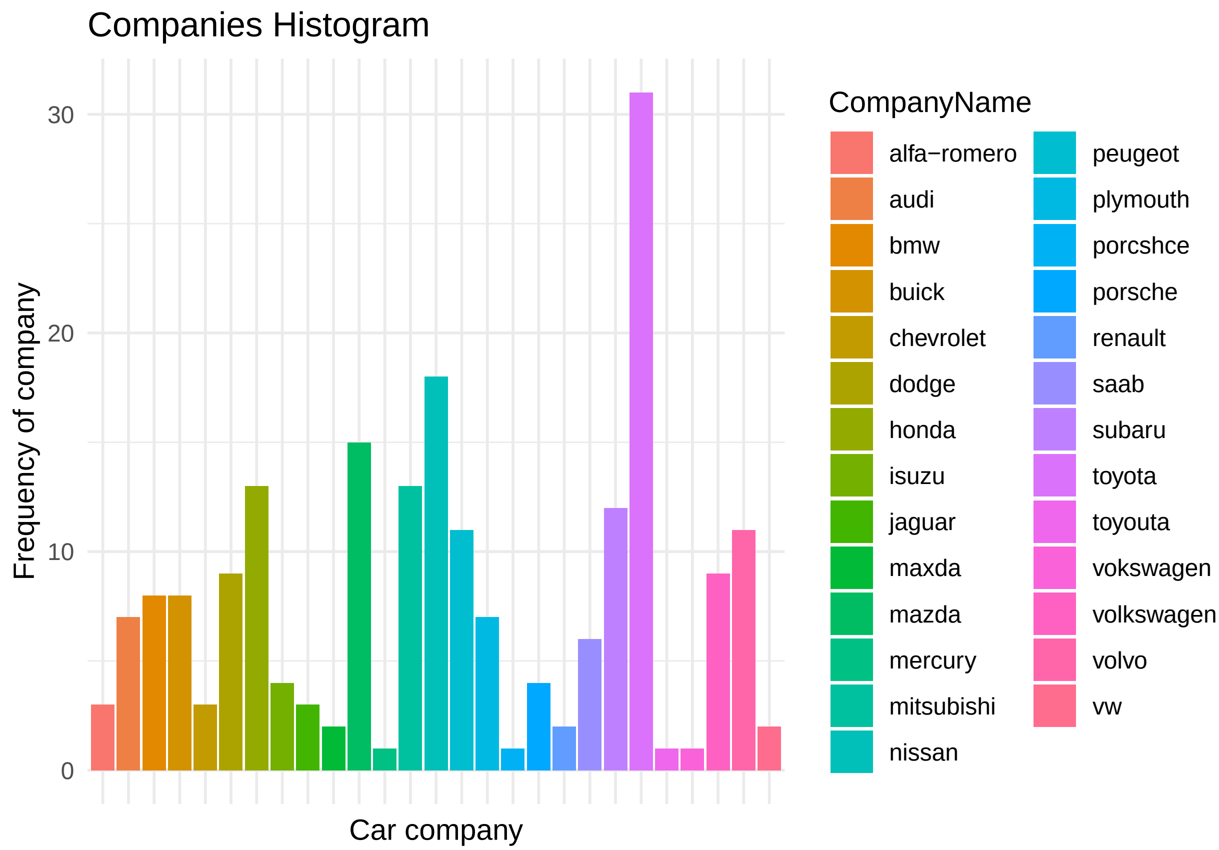


The distribution of car prices in the dataset appears to be right-skewed, indicating that a majority of the prices are lower (below $15,000). There is a noticeable disparity between the mean and median values of the price distribution. Furthermore, the data points are widely dispersed from the mean, suggesting a significant variance in car prices. Specifically, approximately 85% of the prices fall below $18,500, while the remaining 15% range between $18,500 and $45,400.

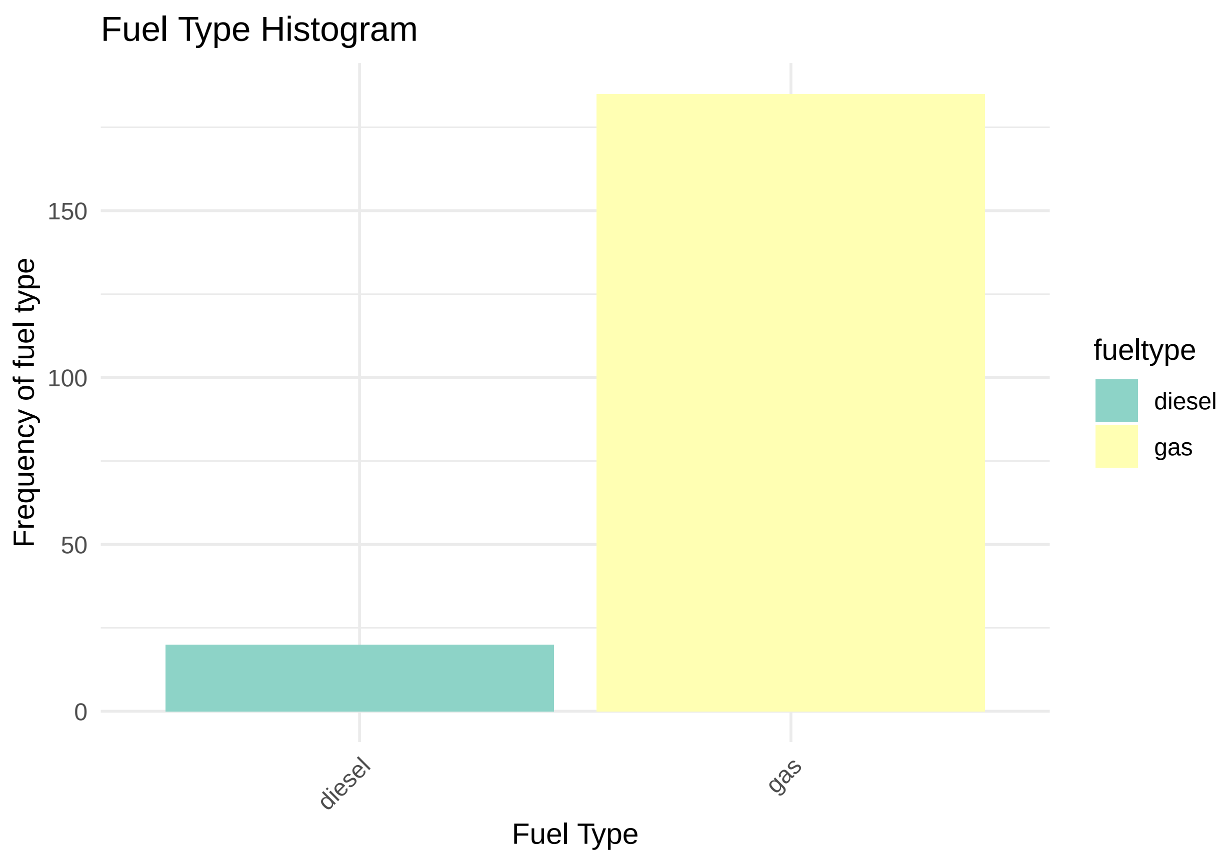
Visualising Categorical data

For Categorical Data: - CompanyName - Symboling - fueltype - enginetype - carbody - doornumber - enginelocation - fuelsystem - cylindernumber - aspiration - drivewheel

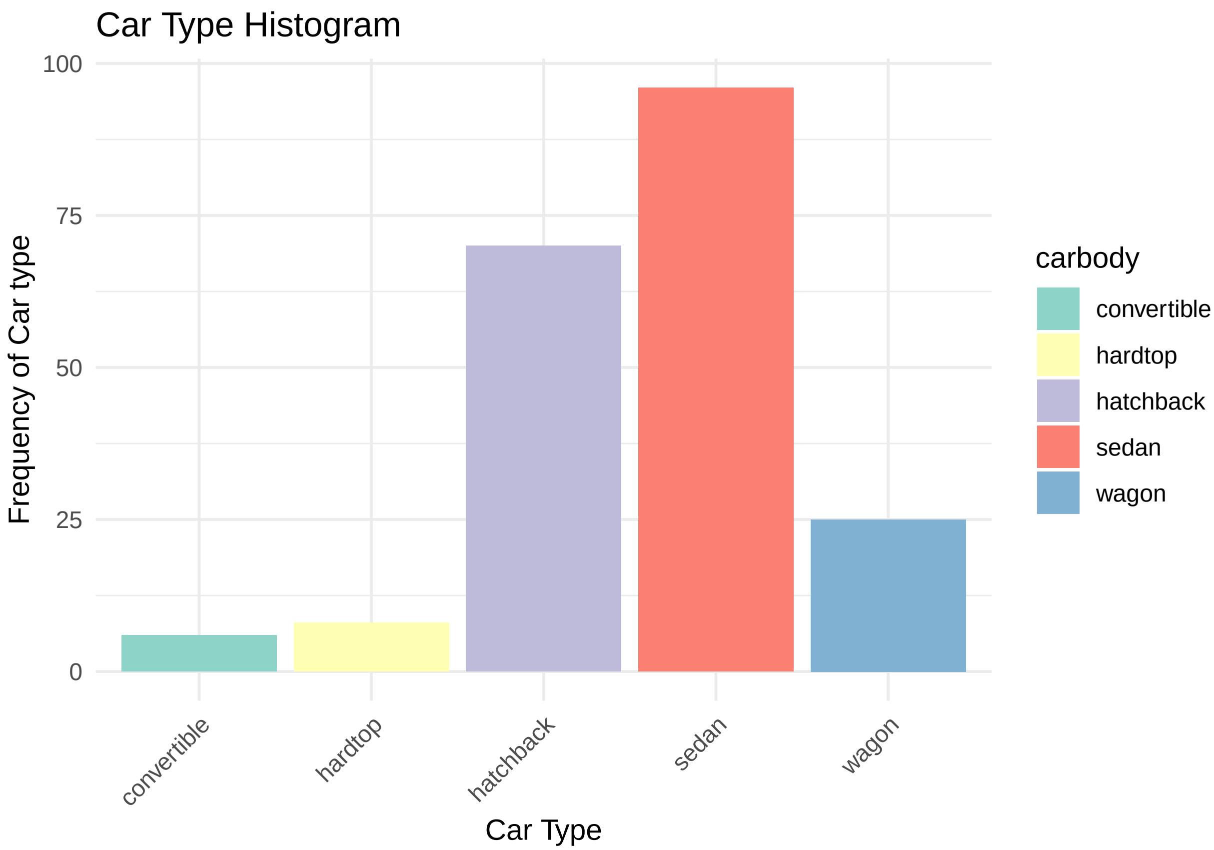
```
# Companies Histogram
p3 <- ggplot(cars, aes(x = CompanyName, fill = CompanyName)) +
  geom_bar() +
  xlab("Car company") + ylab("Frequency of company") +
  ggtitle("Companies Histogram") +
  theme_minimal() +
  theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
print(p3)
```




```
# Fuel Type Histogram
p4 <- ggplot(cars, aes(x = fueltype, fill = fueltype)) +
  geom_bar() +
  xlab("Fuel Type") + ylab("Frequency of fuel type") +
  ggtitle("Fuel Type Histogram") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette="Set3")
print(p4)
```

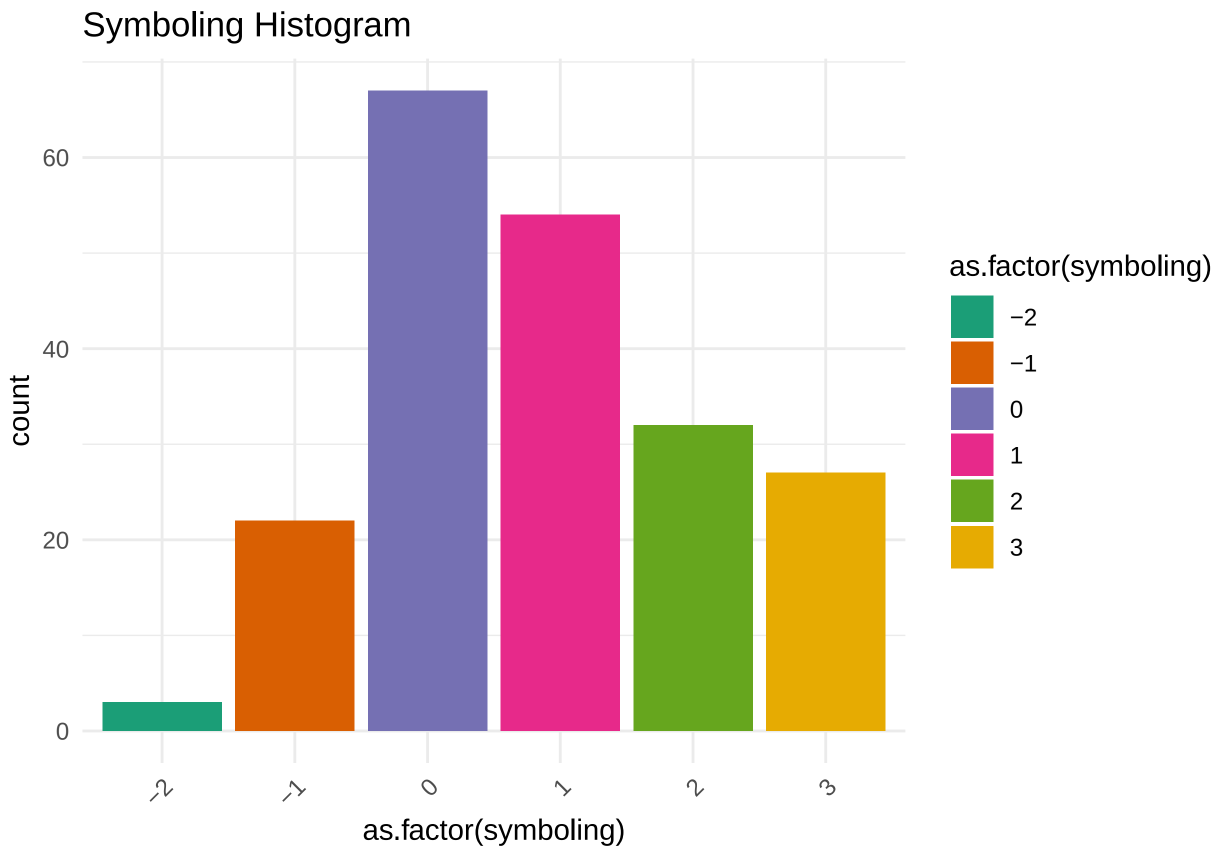



```r
# Car Type Histogram
p5 <- ggplot(cars, aes(x = carbody, fill = carbody)) +
  geom_bar() +
  xlab("Car Type") + ylab("Frequency of Car type") +
  ggtitle("Car Type Histogram") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette="Set3")

print(p5)
```

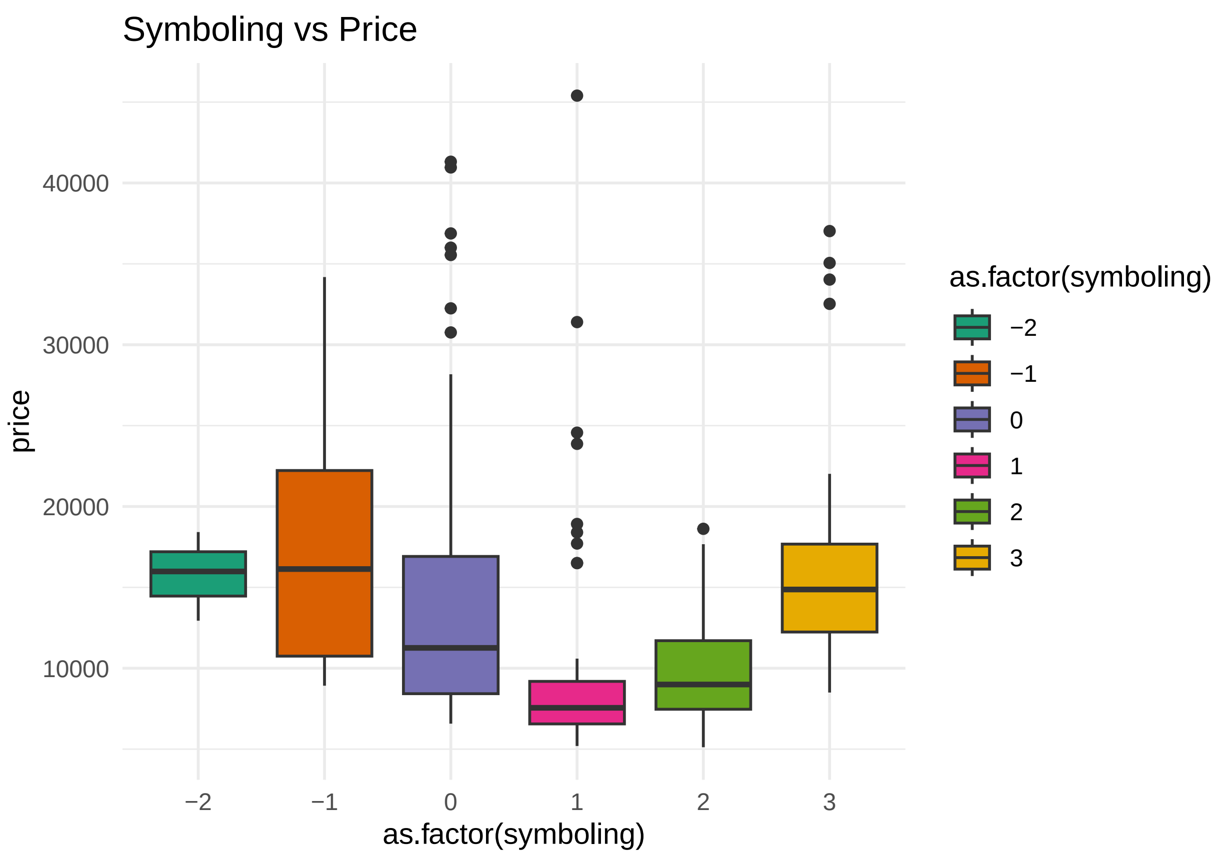


1. Toyota appears to be the most popular car company among the available options.
2. There is a higher number of cars fueled by gas compared to diesel.
3. Sedan is the most preferred car type.

```r
# Symboling Histogram
p1 <- ggplot(cars, aes(x = as.factor(symboling), fill = as.factor(symboling))) +
  geom_bar() +
  ggtitle("Symboling Histogram") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p1)
```




```r
# Symboling vs Price
p2 <- ggplot(cars, aes(x = as.factor(symboling), y = price, fill = as.factor(symboling))) +
  geom_boxplot() +
  ggtitle("Symboling vs Price") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p2)
```

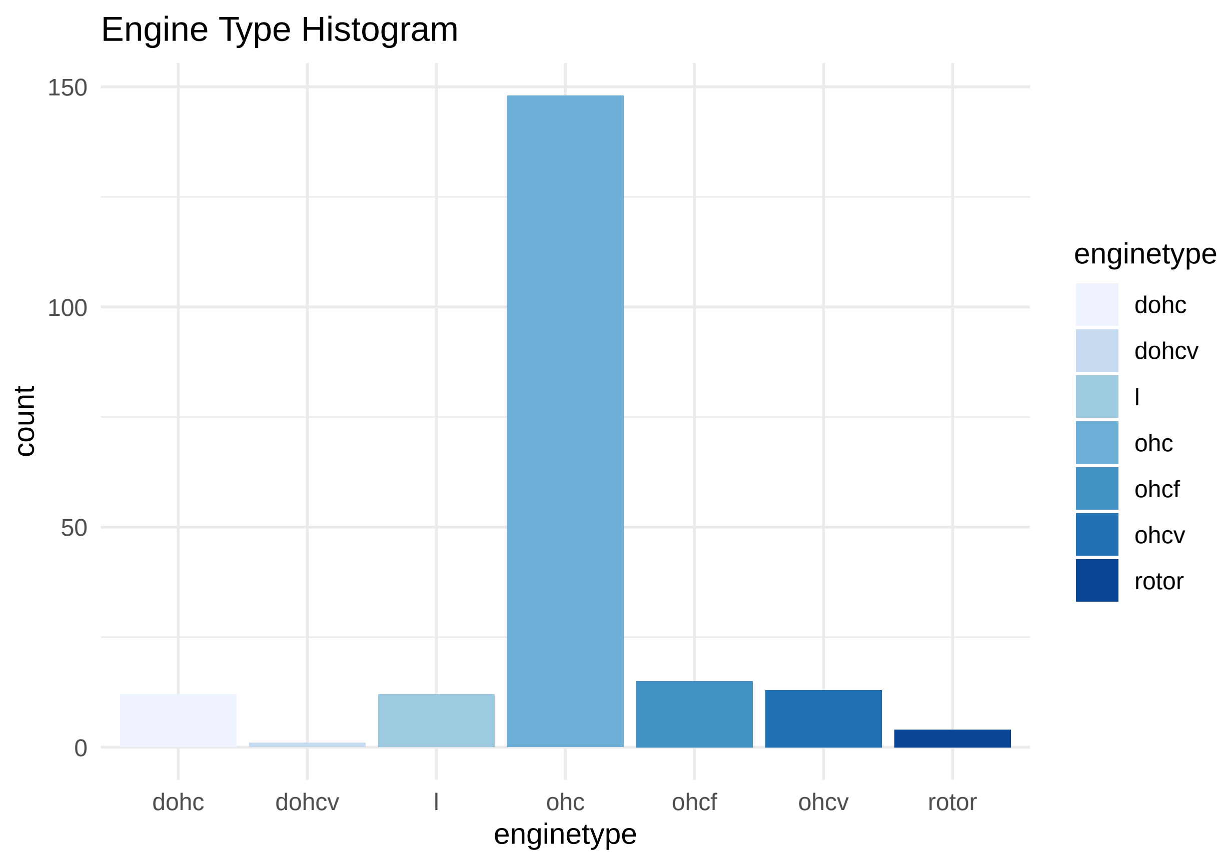


Upon analyzing the data, it appears that cars with symboling values of 0 and 1 have a higher number of rows, indicating that they are the most commonly sold cars in the dataset.

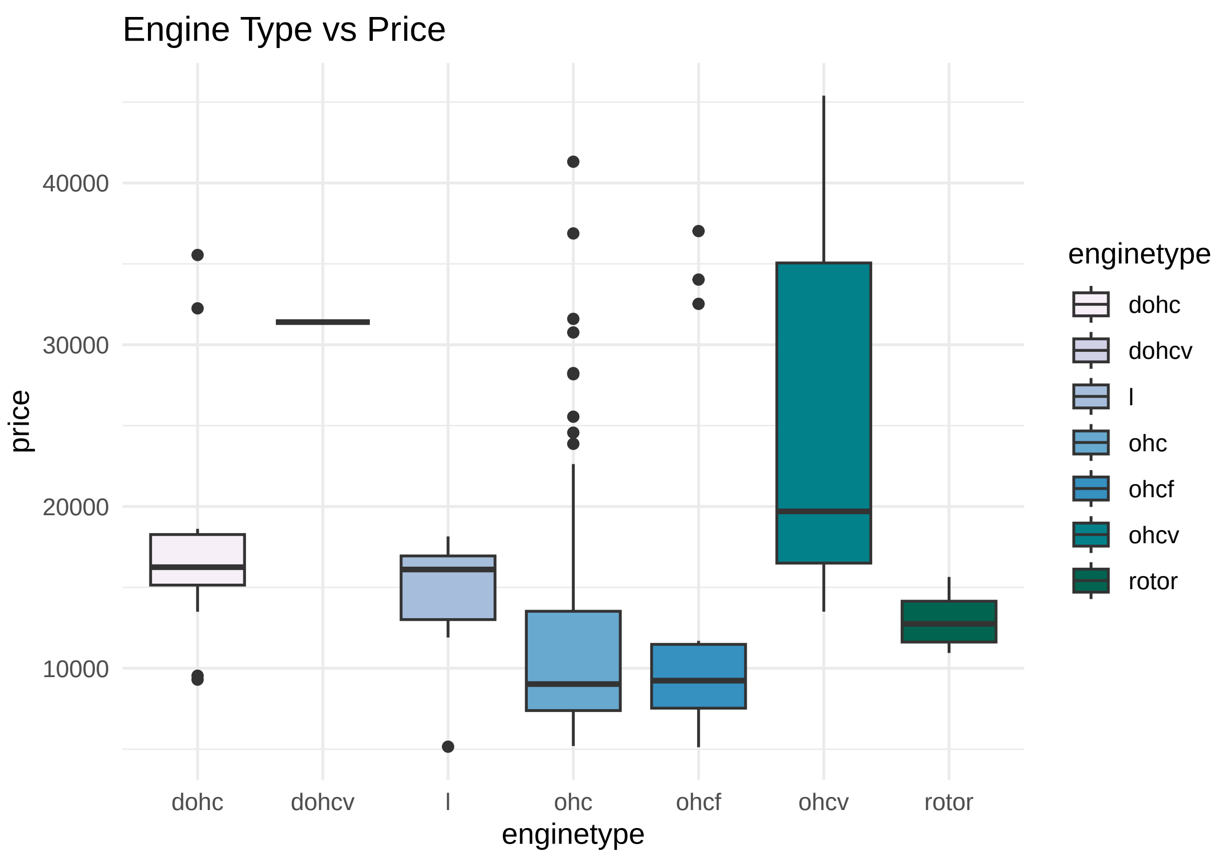
Interestingly, cars with a symboling value of -1, which indicates a favorable insurance risk rating, tend to have higher prices. This observation aligns with expectations, as a lower risk rating usually corresponds to higher prices.

Surprisingly, cars with a symboling value of 3 exhibit a price range similar to cars with a symboling value of -2. This suggests that despite the significant difference in risk ratings, these cars have comparable pricing. Notably, there is a price dip observed for cars with a symboling value of 1, indicating a deviation from the expected pricing pattern based on risk ratings.

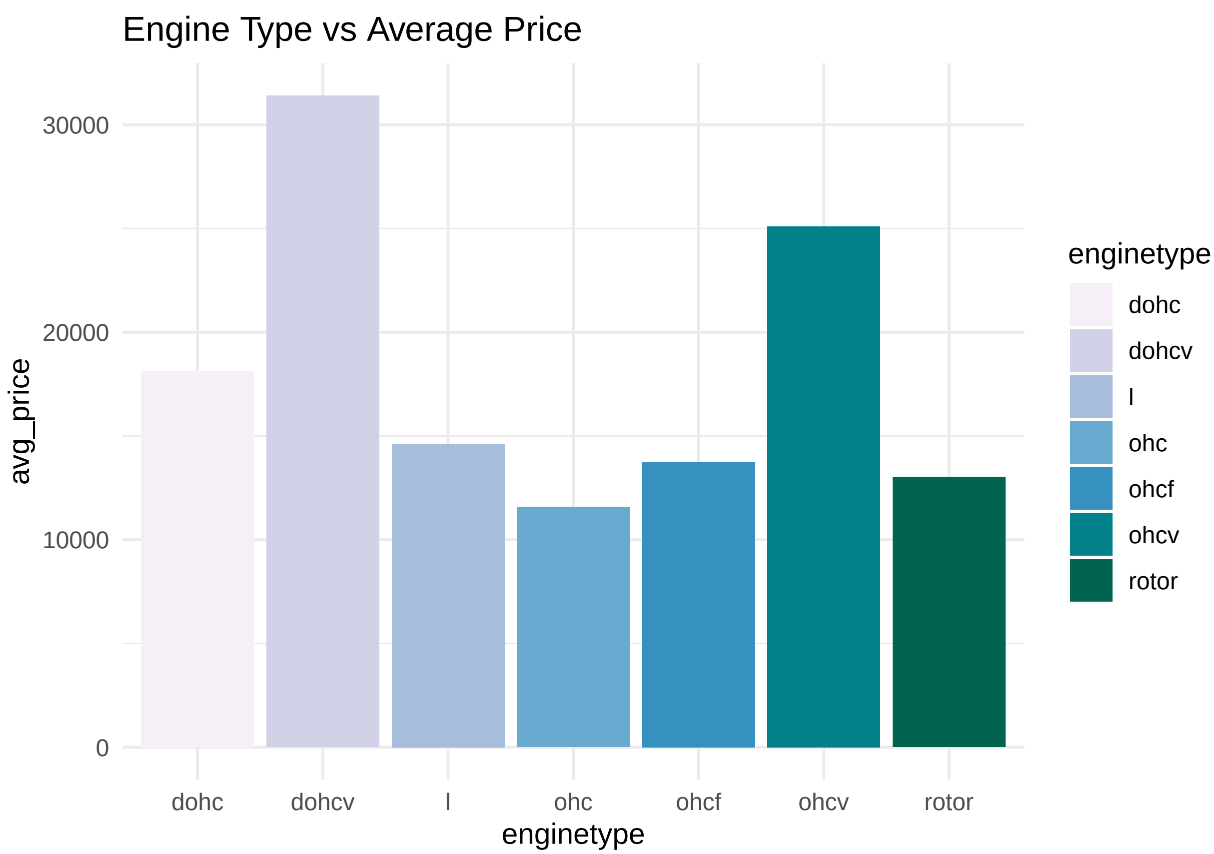
```r
# Engine Type Histogram
p3 <- ggplot(cars, aes(x = enginetype, fill = enginetype)) +
  geom_bar() +
  ggtitle("Engine Type Histogram") +
  theme_minimal() +
  scale_fill_brewer(palette="Blues")
print(p3)
```




```r
# Engine Type vs Price
p4 <- ggplot(cars, aes(x = enginetype, y = price, fill = enginetype)) +
  geom_boxplot() +
  ggtitle("Engine Type vs Price") +
  theme_minimal() +
  scale_fill_brewer(palette="PuBuGn")

print(p4)
```

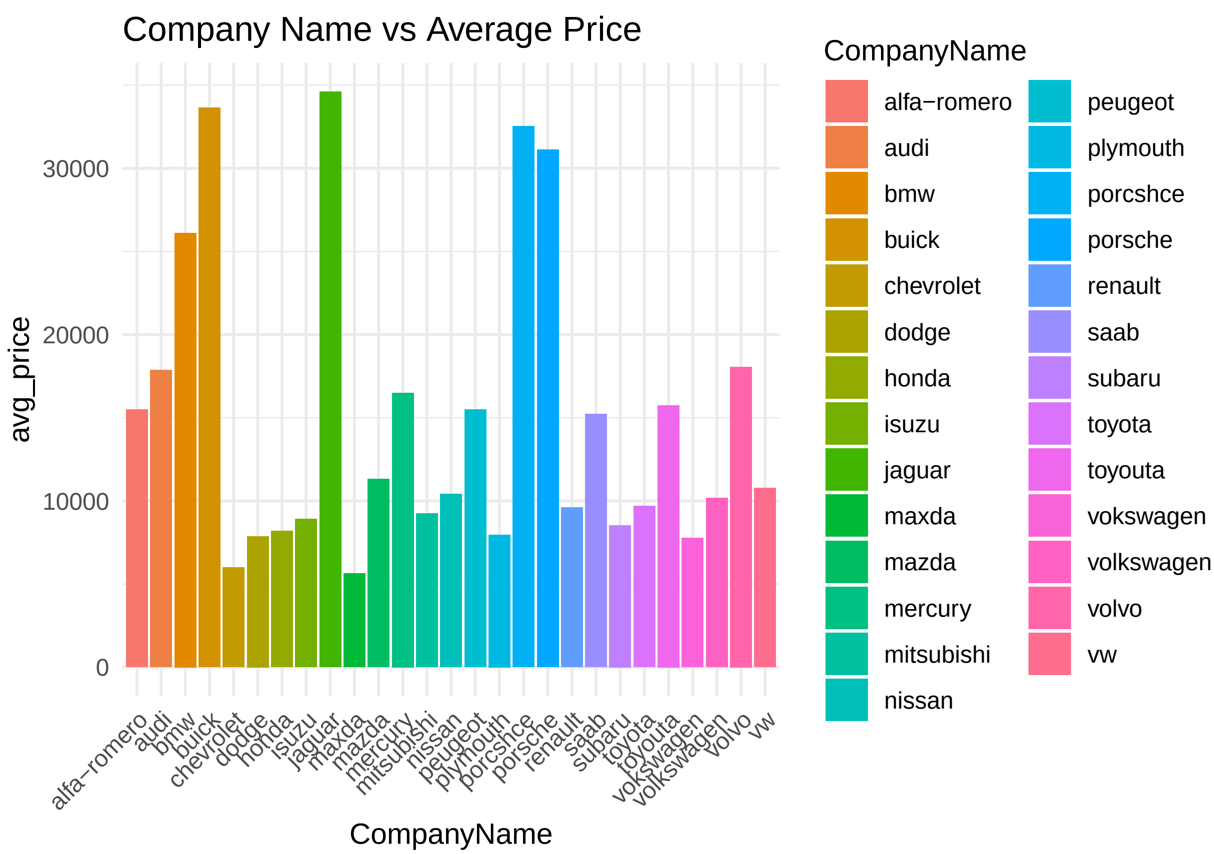



```
# Engine Type vs Average Price
df <- cars %>%
  group_by(enginetype) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(avg_price))

p5 <- ggplot(df, aes(x = enginetype, y = avg_price, fill = enginetype)) +
  geom_bar(stat = "identity") +
  ggtitle("Engine Type vs Average Price") +
  theme_minimal() +
  scale_fill_brewer(palette="PuBuGn")

print(p5)
```

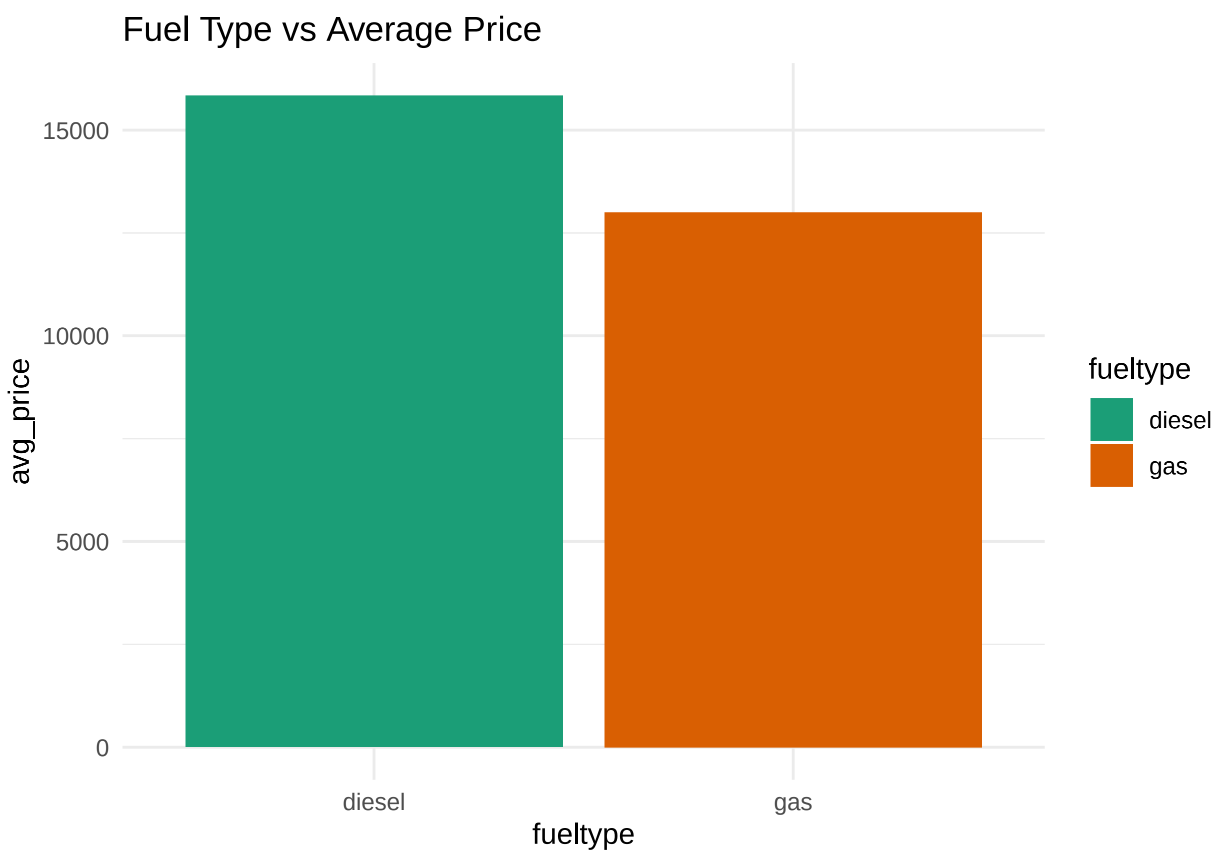


Based on the data provided, it appears that the ohc (Overhead Camshaft) engine type is the most preferred among the car models. On the other hand, cars with ohcv (Overhead Camshaft with Variable Valve Timing) engine type have the highest price range. It is worth noting that the dohcv (Double Overhead Camshaft with Variable Valve Timing) engine type has only one entry in the dataset. Furthermore, cars with ohc (Overhead Camshaft) and ohcf (Overhead Camshaft with Carburetor and Variable Valve Timing) engine types tend to have a lower price range compared to the others.

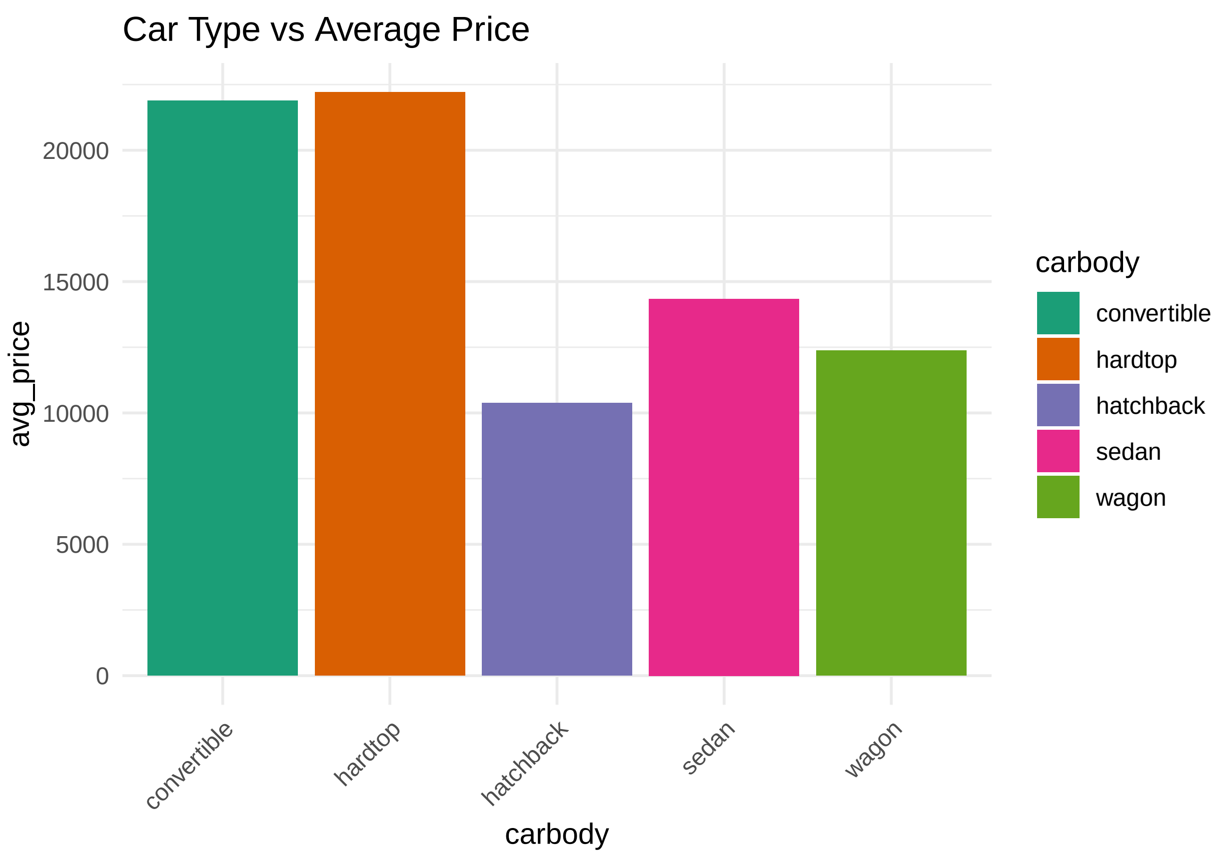
```r
library(dplyr)

# Company Name vs Average Price
df <- cars %>%
  group_by(CompanyName) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(avg_price))

p1 <- ggplot(df, aes(x = CompanyName, y = avg_price, fill = CompanyName)) +
  geom_bar(stat = "identity") +
  ggtitle("Company Name vs Average Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p1)
```




```r
# Fuel Type vs Average Price
df <- cars %>%
  group_by(fueltype) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(avg_price))

p2 <- ggplot(df, aes(x = fueltype, y = avg_price, fill = fueltype)) +
  geom_bar(stat = "identity") +
  ggtitle("Fuel Type vs Average Price") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p2)
```

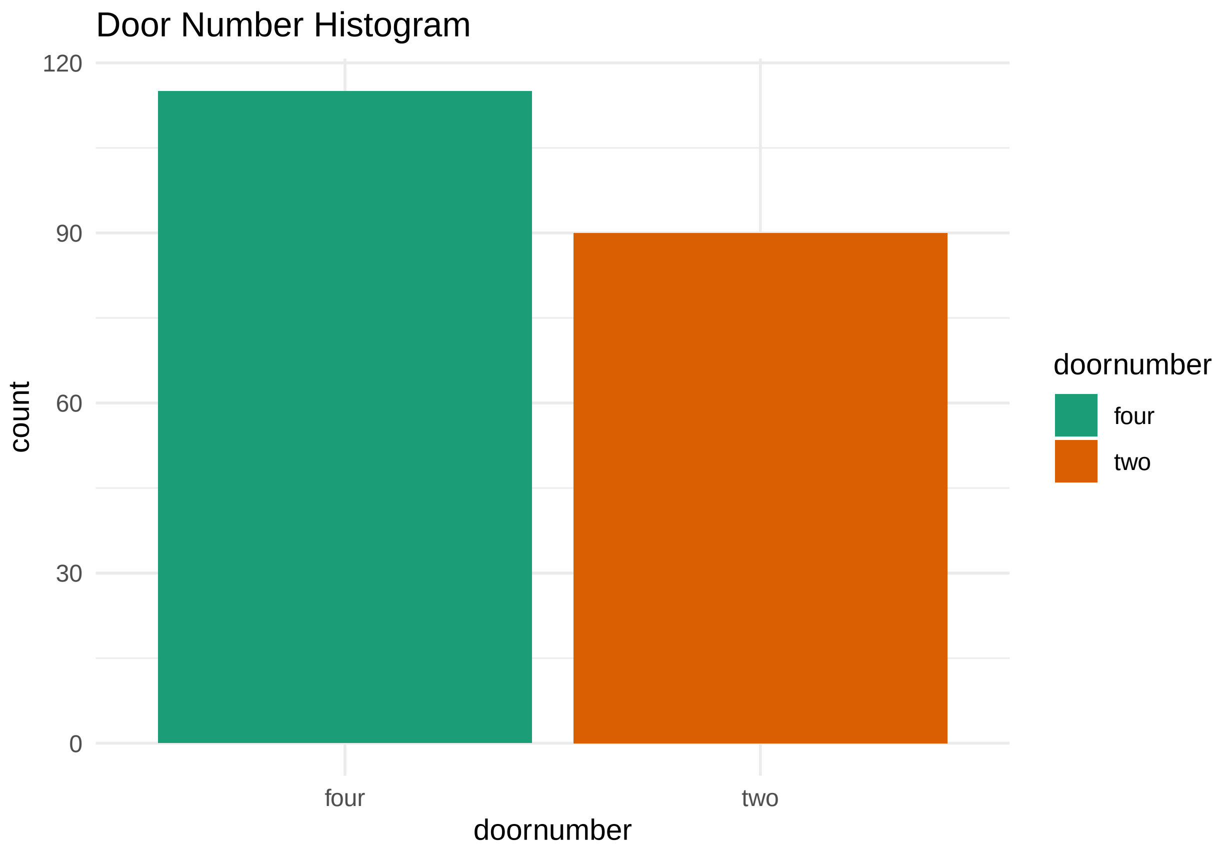



```r
# Car Type vs Average Price
df <- cars %>%
  group_by(carbody) %>%
  summarise(avg_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(avg_price))

p3 <- ggplot(df, aes(x = carbody, y = avg_price, fill = carbody)) +
  geom_bar(stat = "identity") +
  ggtitle("Car Type vs Average Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_brewer(palette="Dark2")

print(p3)
```

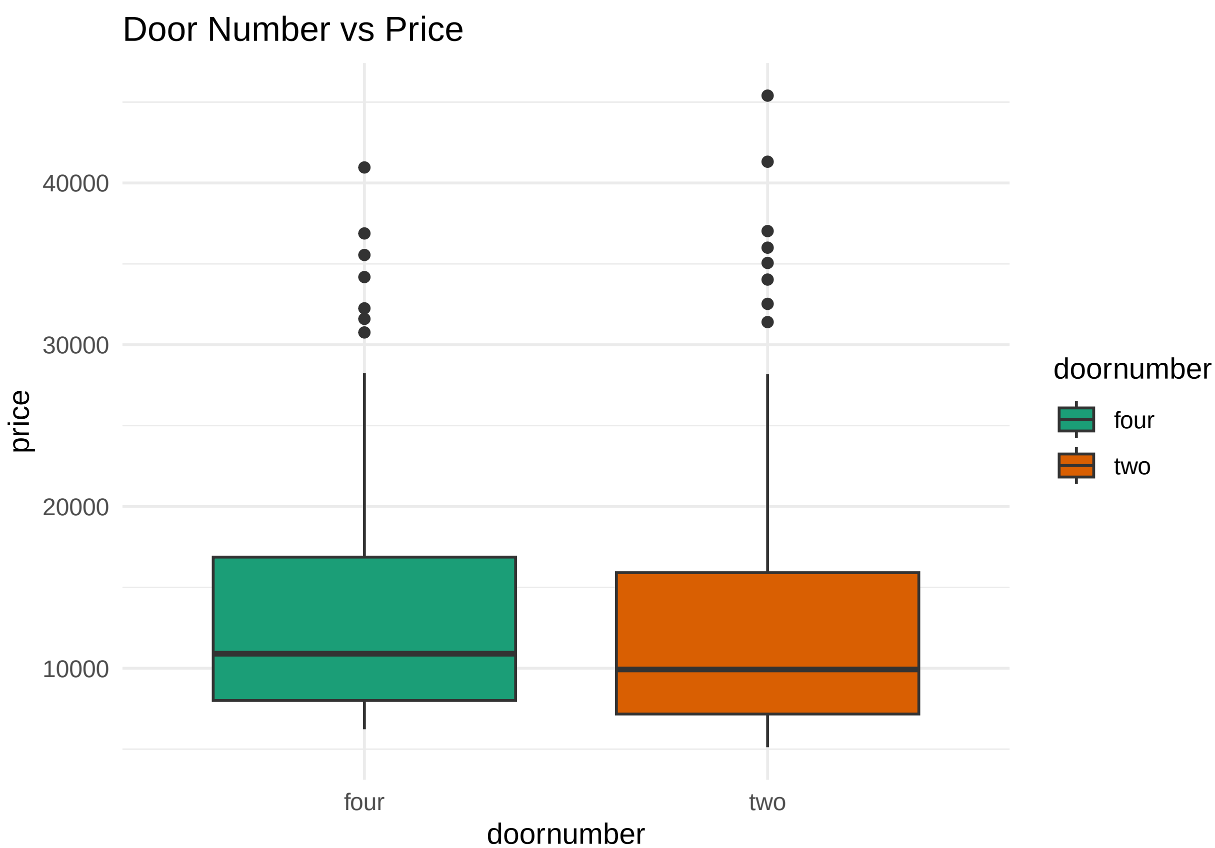


Based on the data analysis, it appears that Jaguar and Buick are the car companies with the highest average prices. Additionally, cars powered by diesel fuel tend to have higher average prices compared to those running on gas. Furthermore, the car body types classified as hardtop and convertible generally exhibit higher average prices.

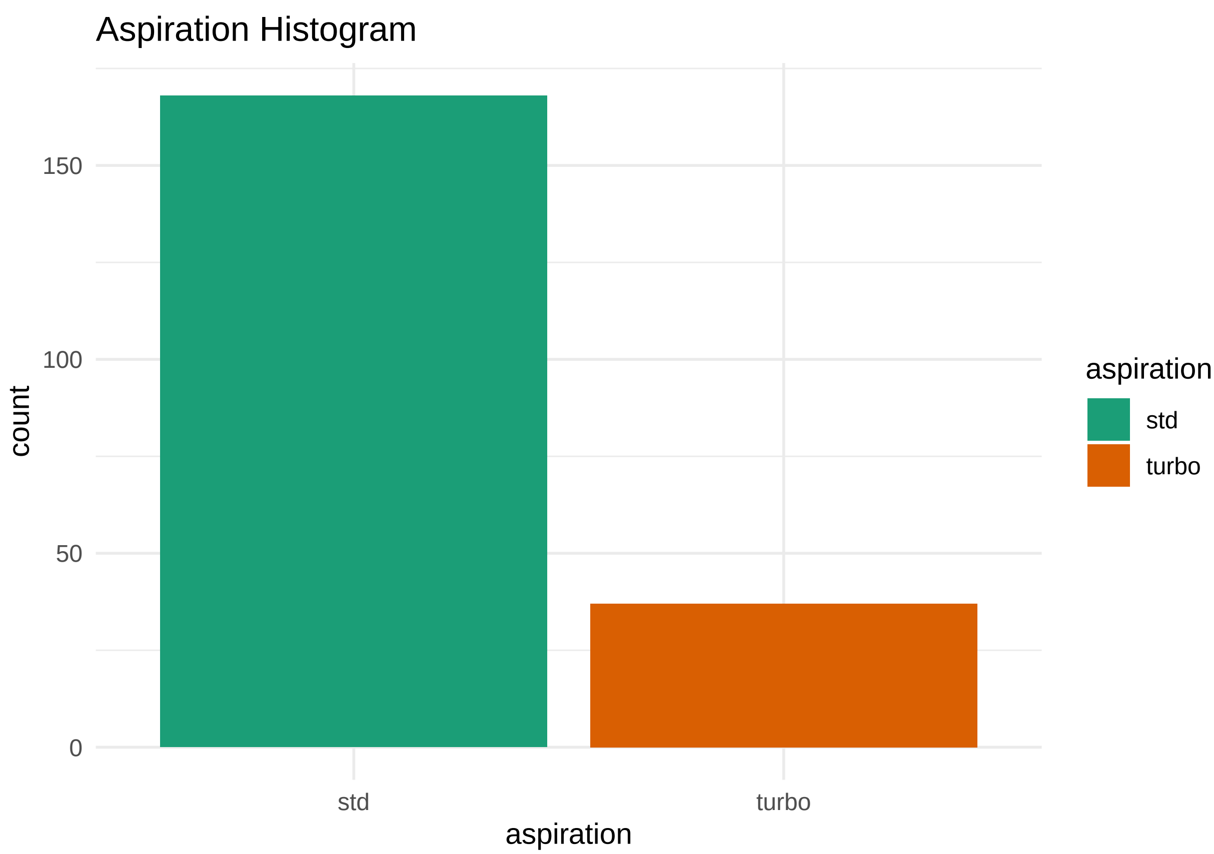
```r
# Door Number Histogram
p4 <- ggplot(cars, aes(x = doornumber, fill = doornumber)) +
  geom_bar() +
  ggtitle("Door Number Histogram") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p4)
```



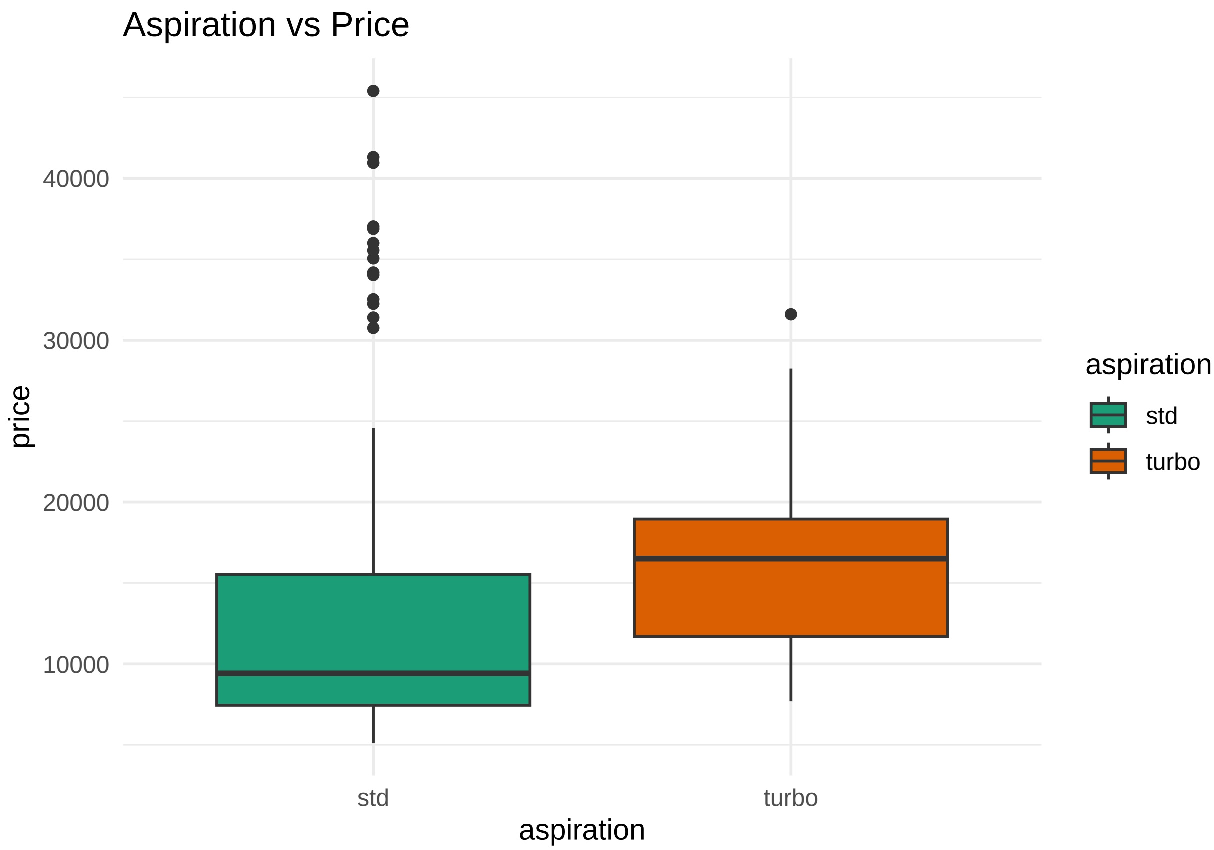

```r
# Door Number vs Price
p5 <- ggplot(cars, aes(x = doornumber, y = price, fill = doornumber)) +
  geom_boxplot() +
  ggtitle("Door Number vs Price") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p5)
```




```
# Aspiration Histogram
p6 <- ggplot(cars, aes(x = aspiration, fill = aspiration)) +
  geom_bar() +
  ggtitle("Aspiration Histogram") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p6)
```



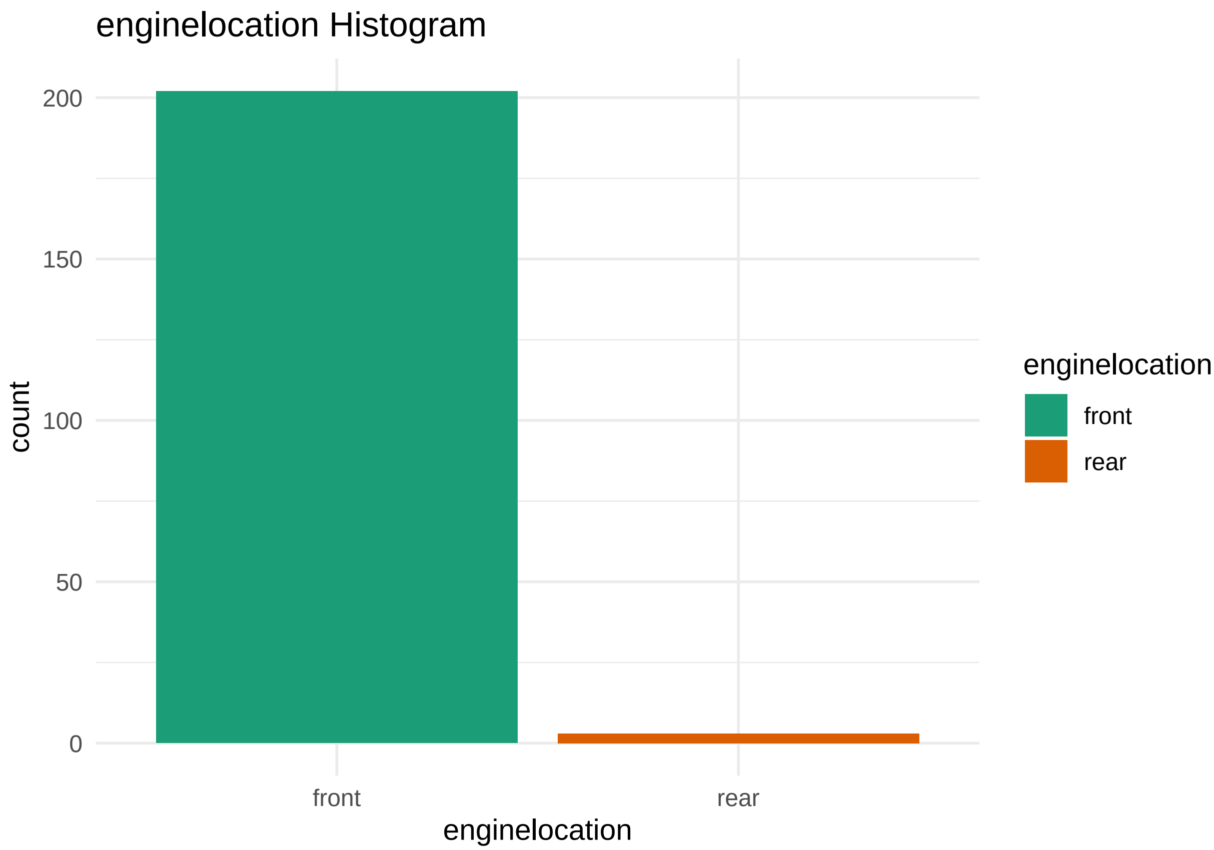

```
# Aspiration vs Price
p7 <- ggplot(cars, aes(x = aspiration, y = price, fill = aspiration)) +
  geom_boxplot() +
  ggtitle("Aspiration vs Price") +
  theme_minimal() +
  scale_fill_brewer(palette="Dark2")

print(p7)
```

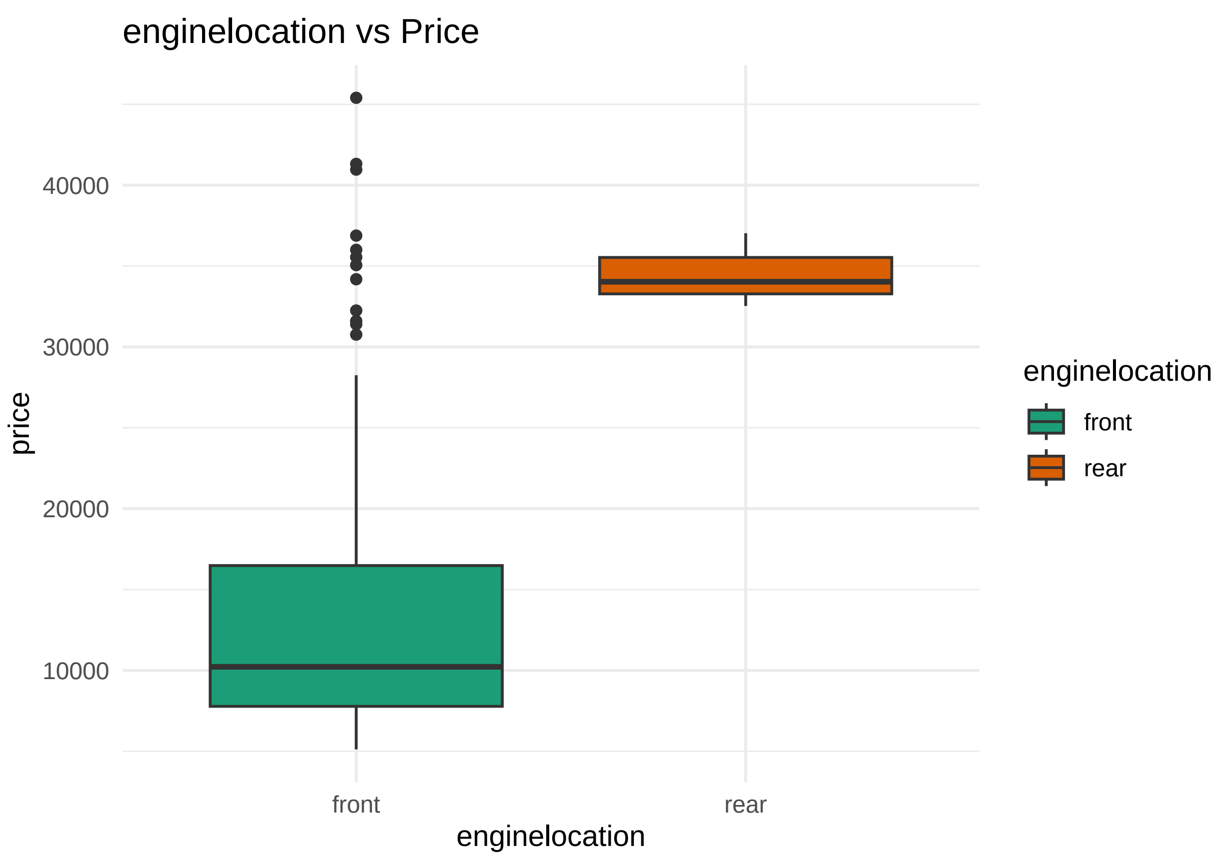


Based on the analysis, it appears that the "doornumber" variable does not have a significant impact on the price of the cars. There is minimal difference observed between the different categories of this variable.

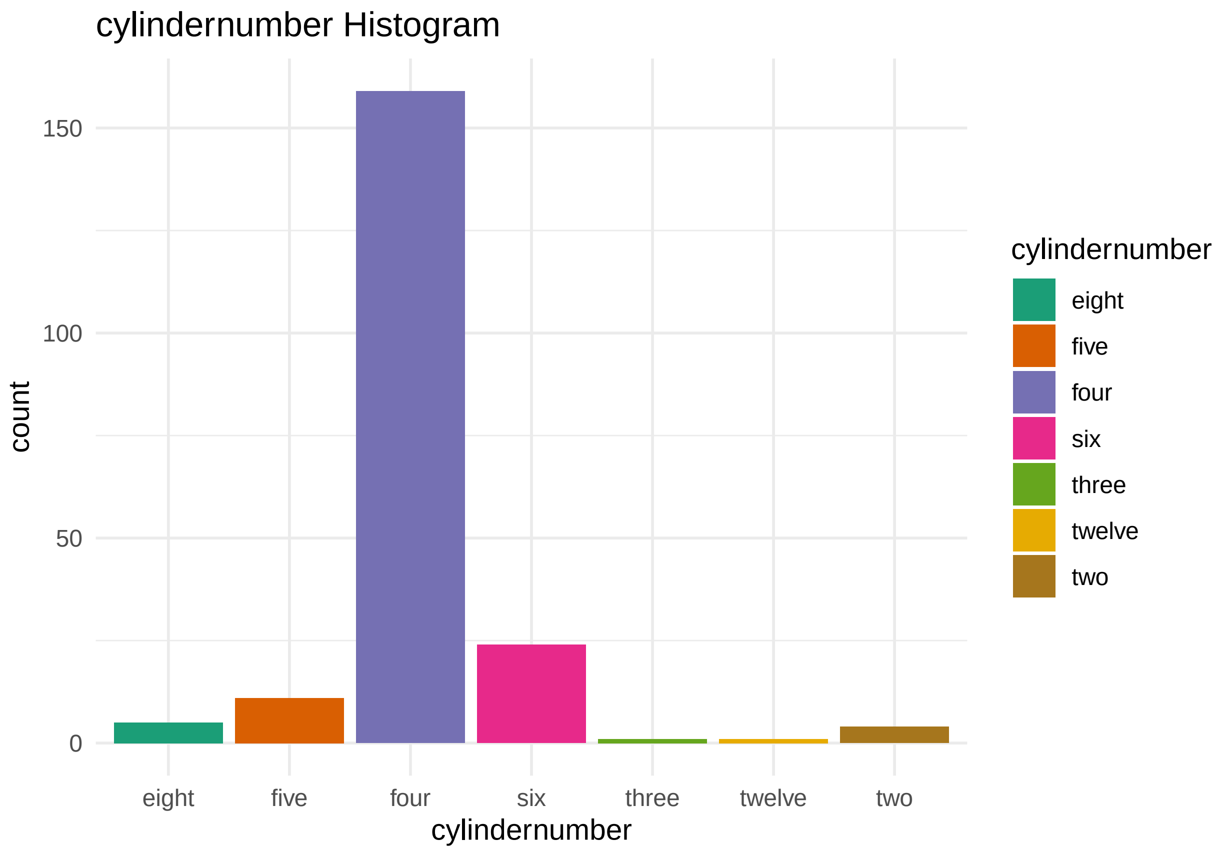
Furthermore, it seems that cars with a turbo aspiration have a higher price range compared to those with a standard aspiration. However, it should be noted that there are some outliers with unusually high values outside the typical range.

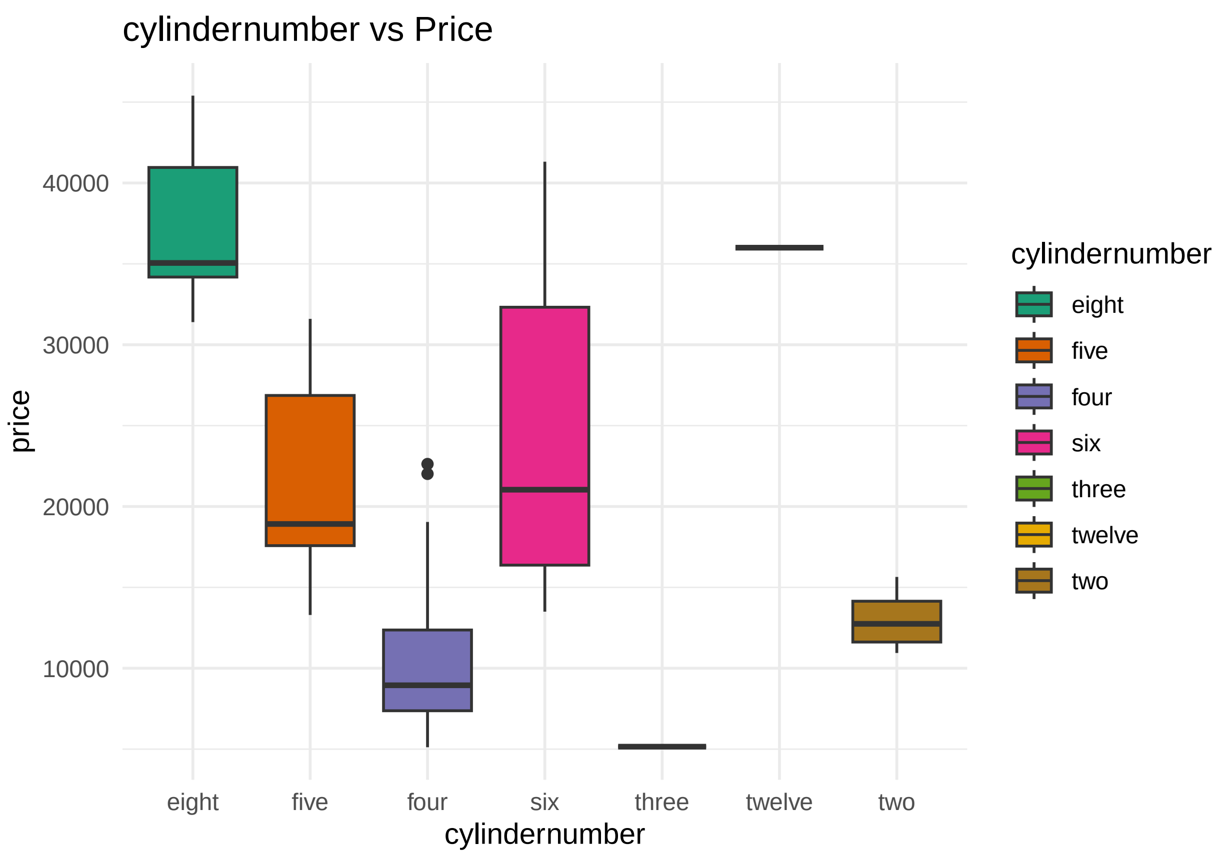
```
# Create a function for plotting histograms and boxplots
plot_count <- function(x, fig) {
  p1 <- ggplot(cars, aes_string(x = x, fill = x)) +
    geom_bar() +
    ggtitle(paste(x, "Histogram")) +
    theme_minimal() +
    scale_fill_brewer(palette="Dark2")

  print(p1)

  p2 <- ggplot(cars, aes_string(x = x, y = "price", fill = x)) +
    geom_boxplot() +
    ggtitle(paste(x, "vs Price")) +
    theme_minimal() +
    scale_fill_brewer(palette="Dark2")

  print(p2)
}

# Plot countplots and boxplots
plot_count('enginelocation', 1)
```


enginelocation Histogram
count
200
150
100
50
0
front
rear
enginelocation
enginelocation
front
rear

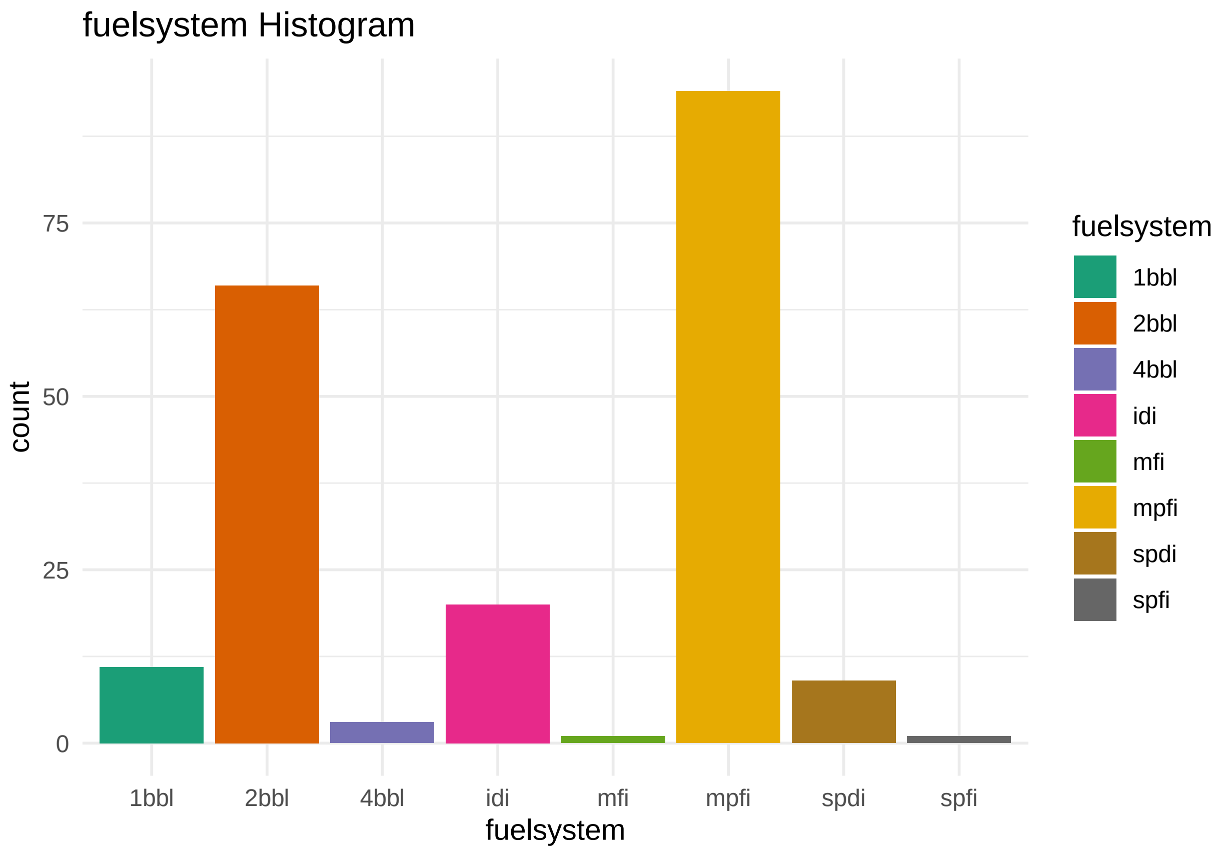

enginelocation vs Price
price
40000
30000
20000
10000
front
rear
enginelocation
enginelocation
front
rear

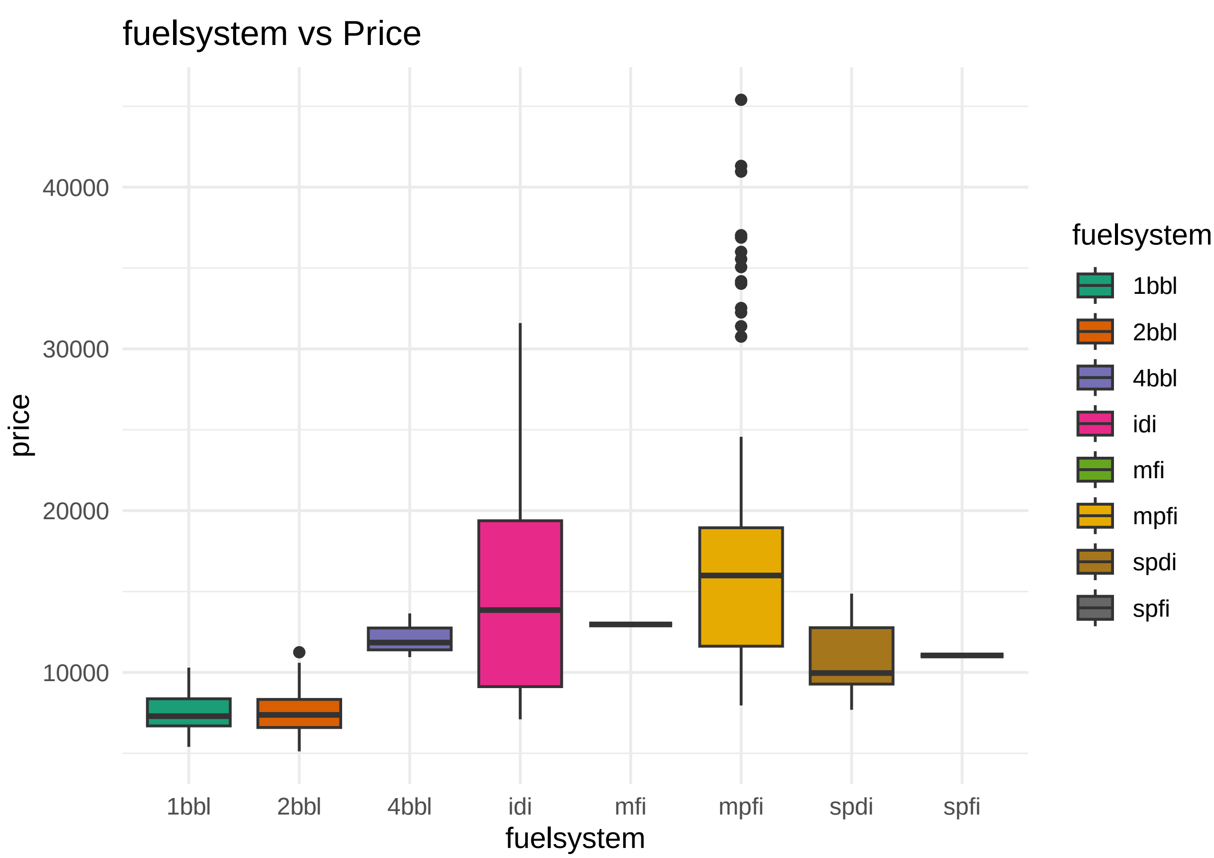
```
plot_count('cylindernumber', 3)
```



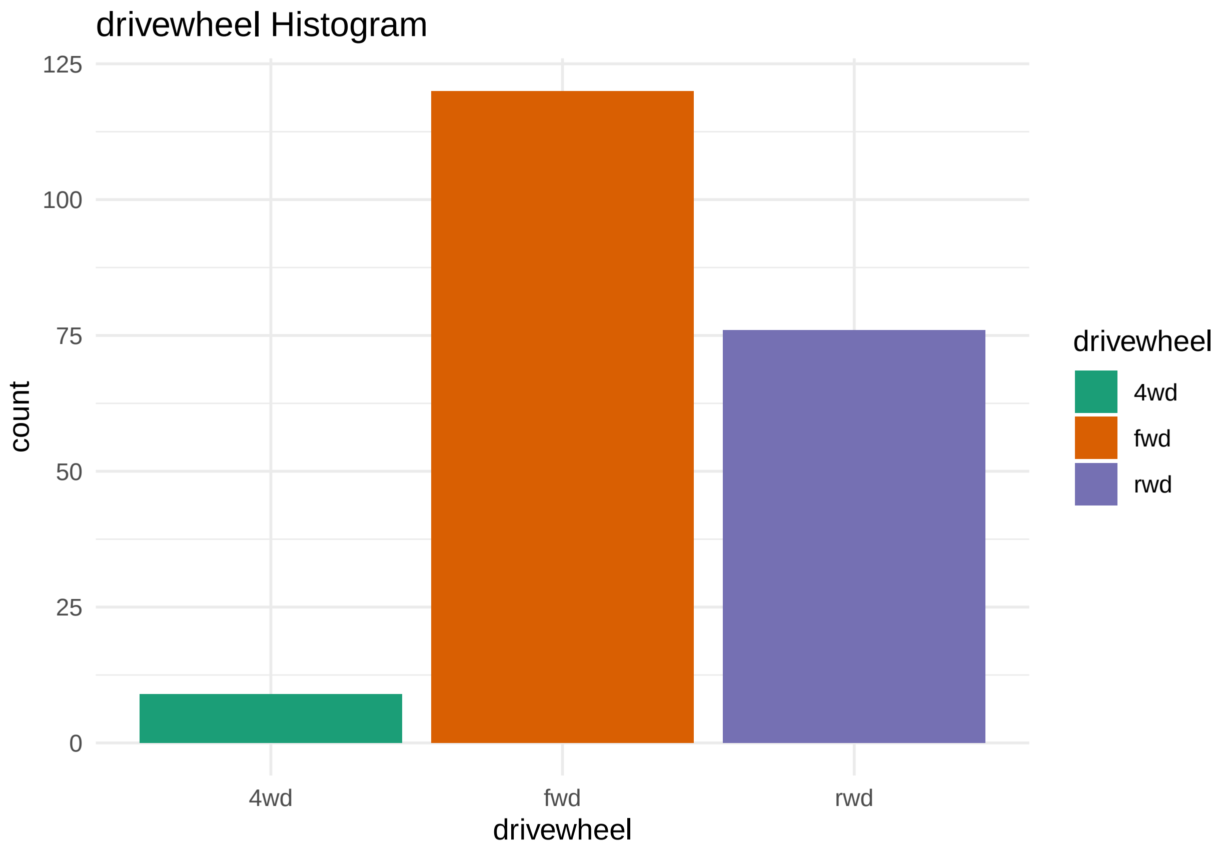


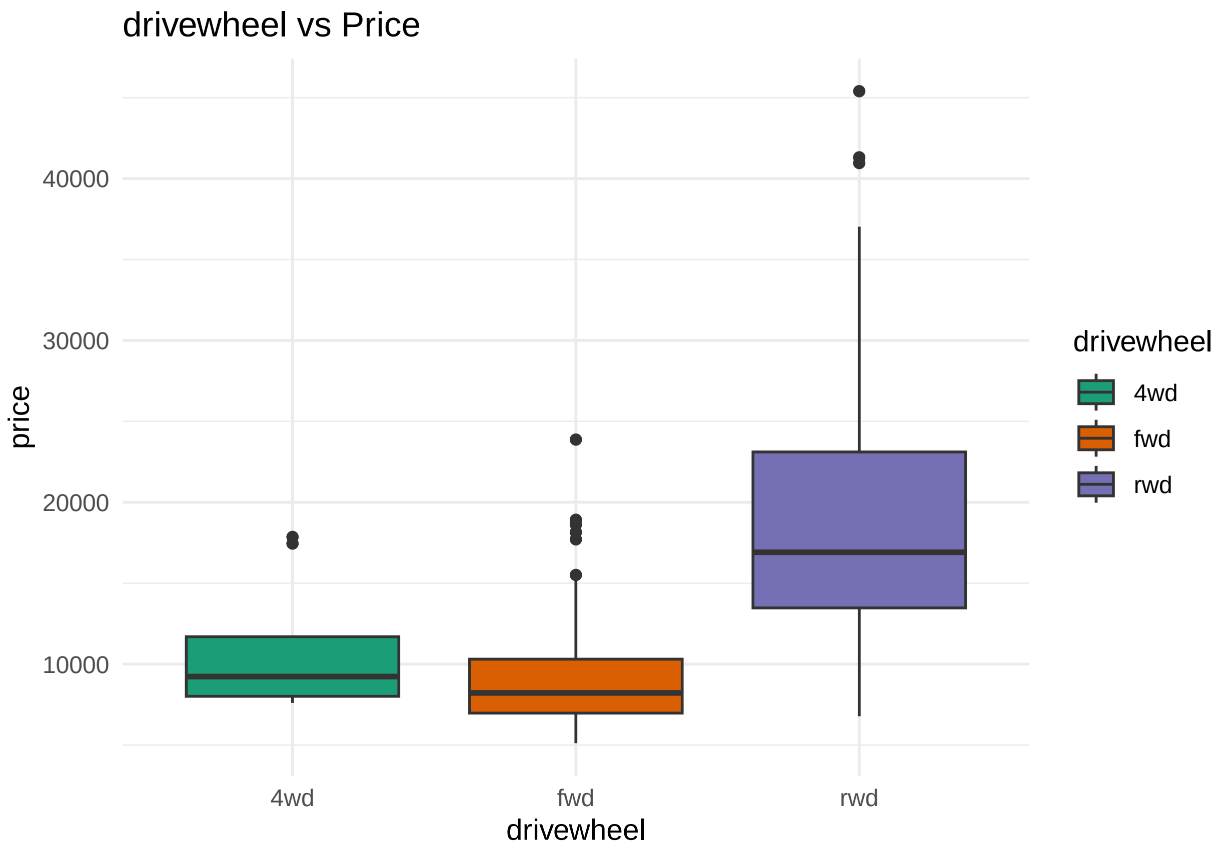

```
plot_count('fuelsystem', 5)
```


fuelsystem Histogram
count
0
25
50
75
1bbl
2bbl
4bbl
idi
mfi
mpfi
spdi
spfi
fuelsystem
fuelsystem
1bbl
2bbl
4bbl
idi
mfi
mpfi
spdi
spfi


fuelsystem vs Price
price
10000
20000
30000
40000
1bbl
2bbl
4bbl
idi
mfi
mpfi
spdi
spfi
fuelsystem
fuelsystem
1bbl
2bbl
4bbl
idi
mfi
mpfi
spdi
spfi

```
plot_count('drivewheel', 7)
```





The dataset provides information on various car attributes, but there are some observations to note:

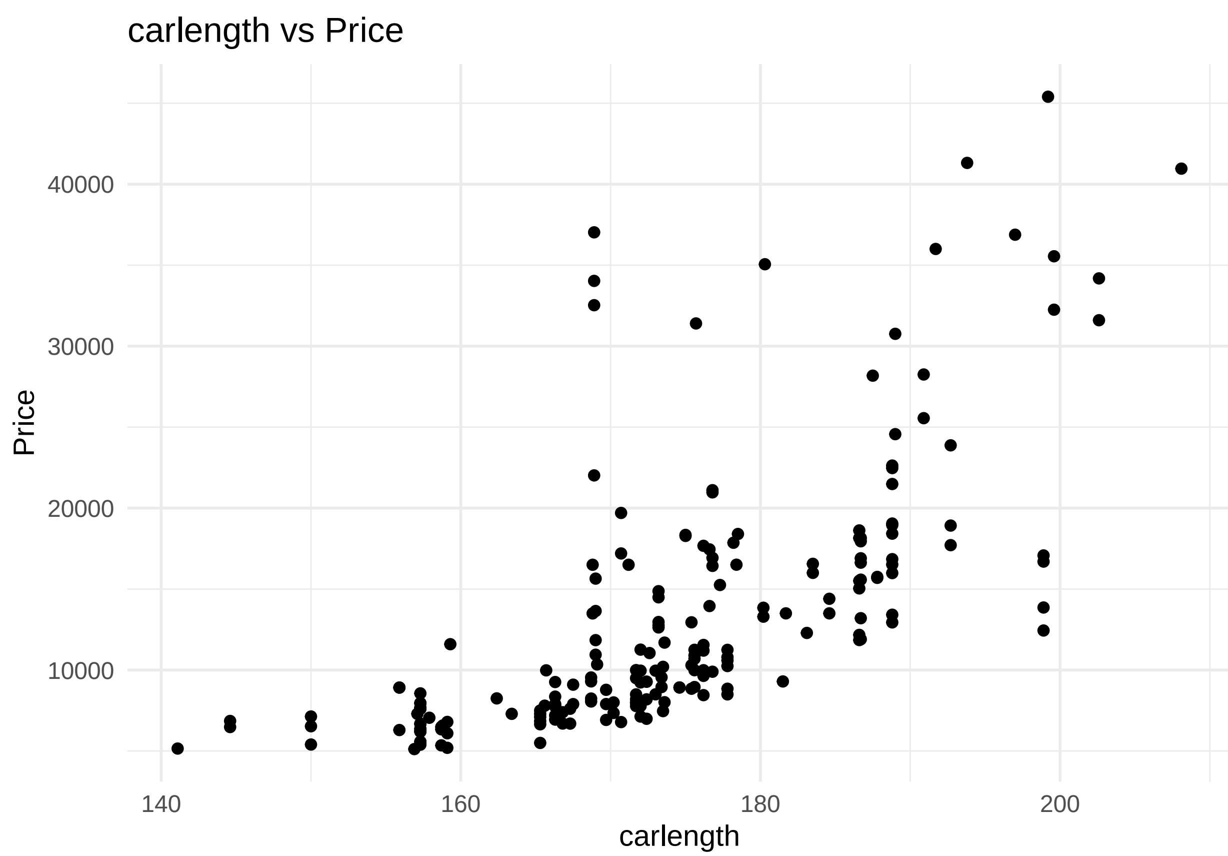
1. Engine Location: There are very few data points available for the engine location categories. Therefore, it is challenging to draw meaningful inferences from this attribute.
2. Cylinder Number: The most common number of cylinders found in cars is four, followed by six and five. However, cars with eight cylinders tend to have the highest price range.
3. Fuel System: The majority of cars in the dataset have the mpfi and 2bbl fuel systems. Among them, cars with the mpfi and idi fuel systems tend to have the highest price range. It is important to note that there are limited data points available for other fuel system categories, making it difficult to derive significant insights from those categories.
4. Drivewheel: There is a significant difference observed in the drivewheel category. Most high-priced cars seem to prefer the rear-wheel drive (rwd) drivewheel configuration.

These observations provide some insights into the relationships between certain car attributes and their pricing, but further analysis and consideration of other variables are necessary for a more comprehensive understanding.

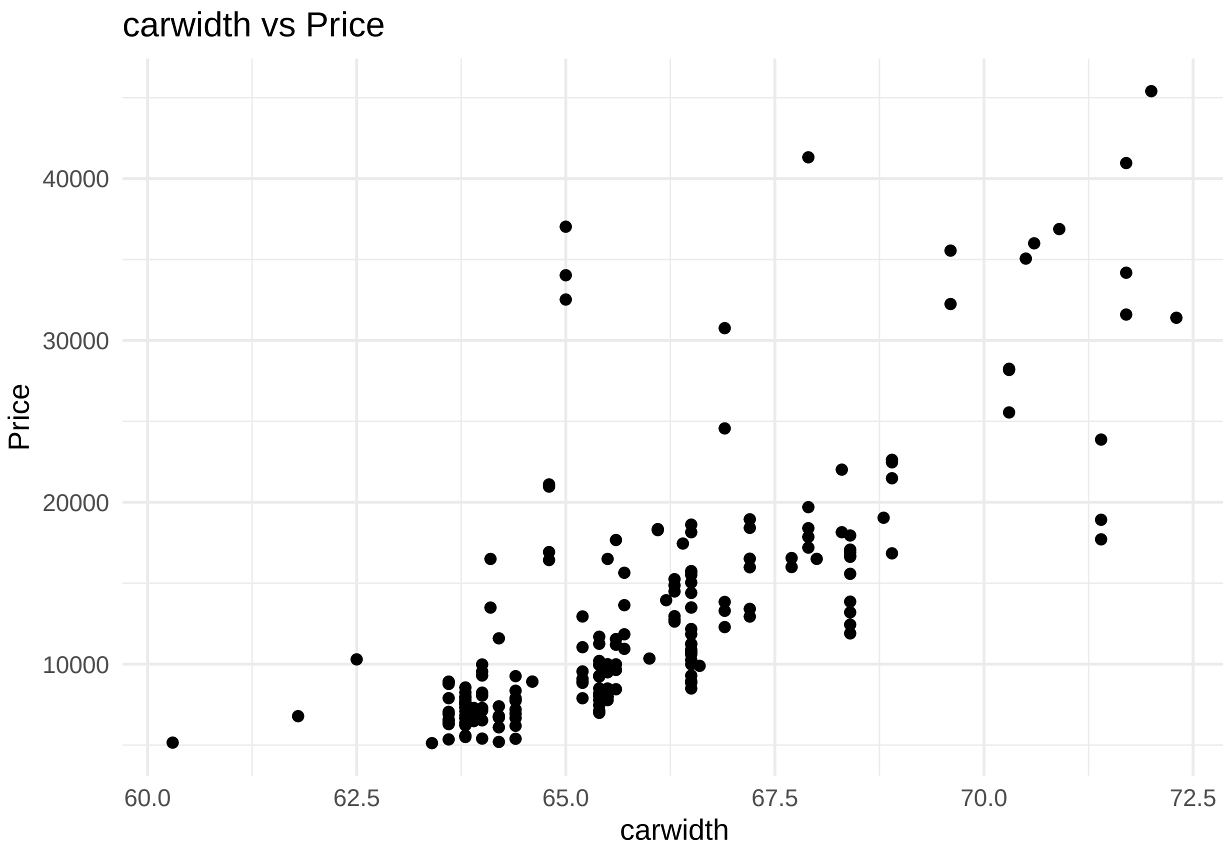
Visualising numerical data

```
# Create a function for scatter plots
scatter <- function(x, fig) {
  p <- ggplot(cars, aes_string(x = x, y = "price")) +
    geom_point() +
    ggtitle(paste(x, "vs Price")) +
```

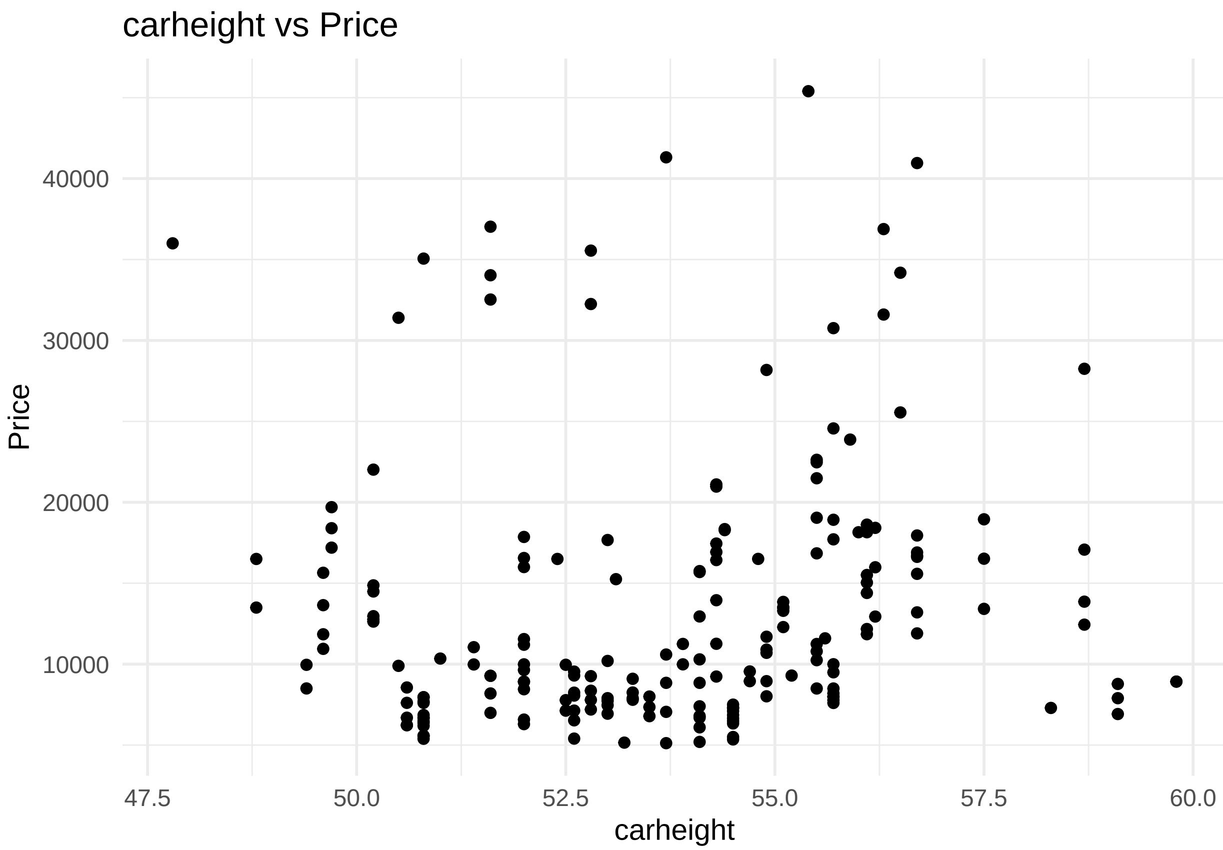
```r
    ylab("Price") +
    xlab(x) +
    theme_minimal()

  print(p)
}

# Plot scatter plots
scatter('carlength', 1)
```



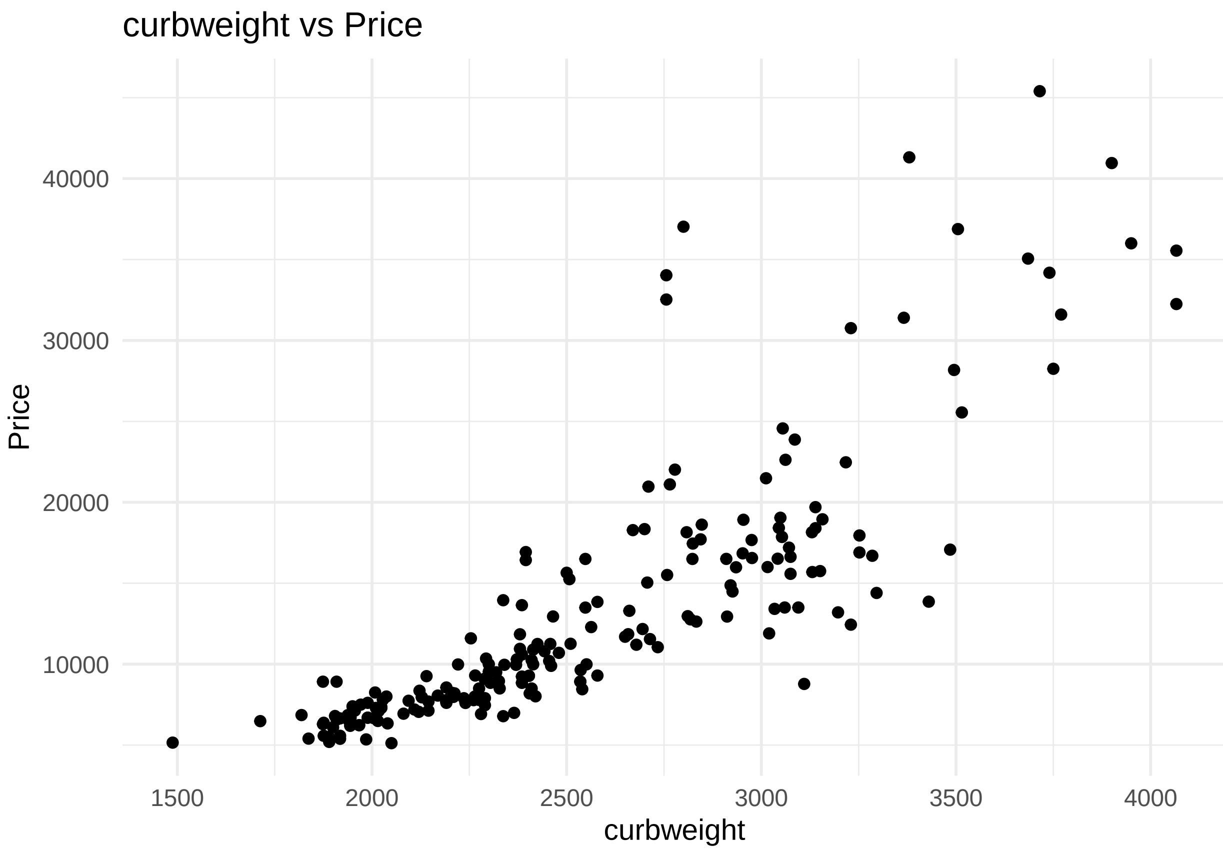

```r
scatter('carwidth', 2)
```




```
scatter('carheight', 3)
```




```
scatter('curbweight', 4)
```

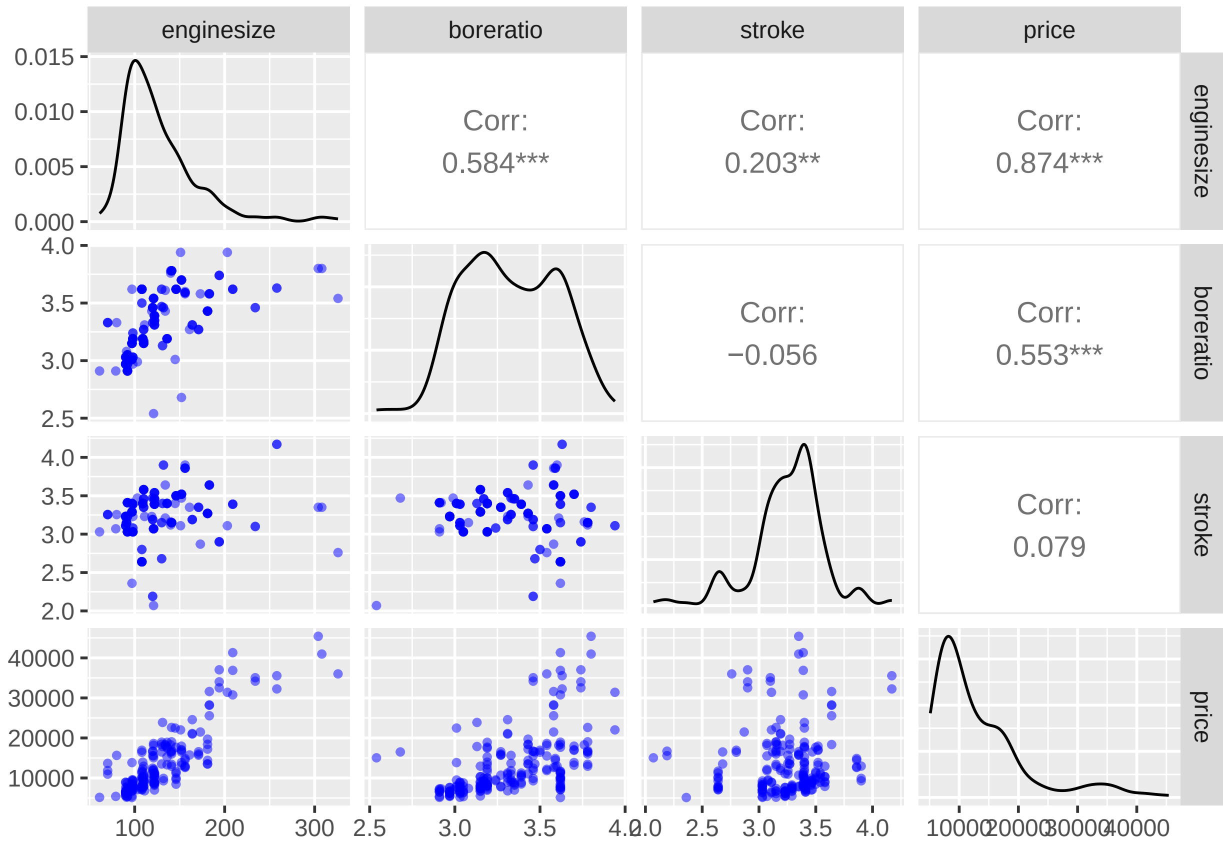


Based on the data analysis, it appears that carwidth, carlength, and curbweight exhibit a positive correlation with price. This implies that as these variables increase, the price of the car tends to increase as well. On the other hand, carheight does not show a significant trend or correlation with price, suggesting that changes in carheight do not strongly influence the pricing of the car.

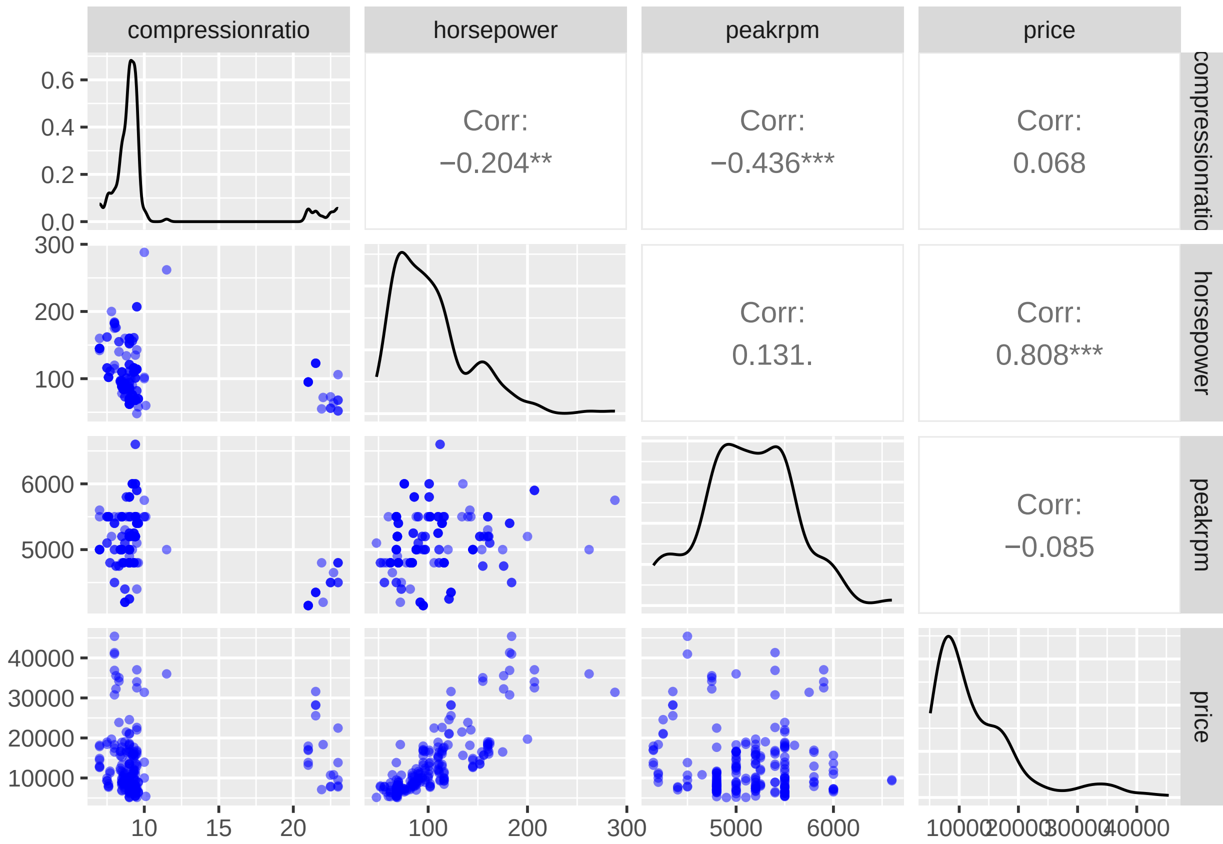
```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

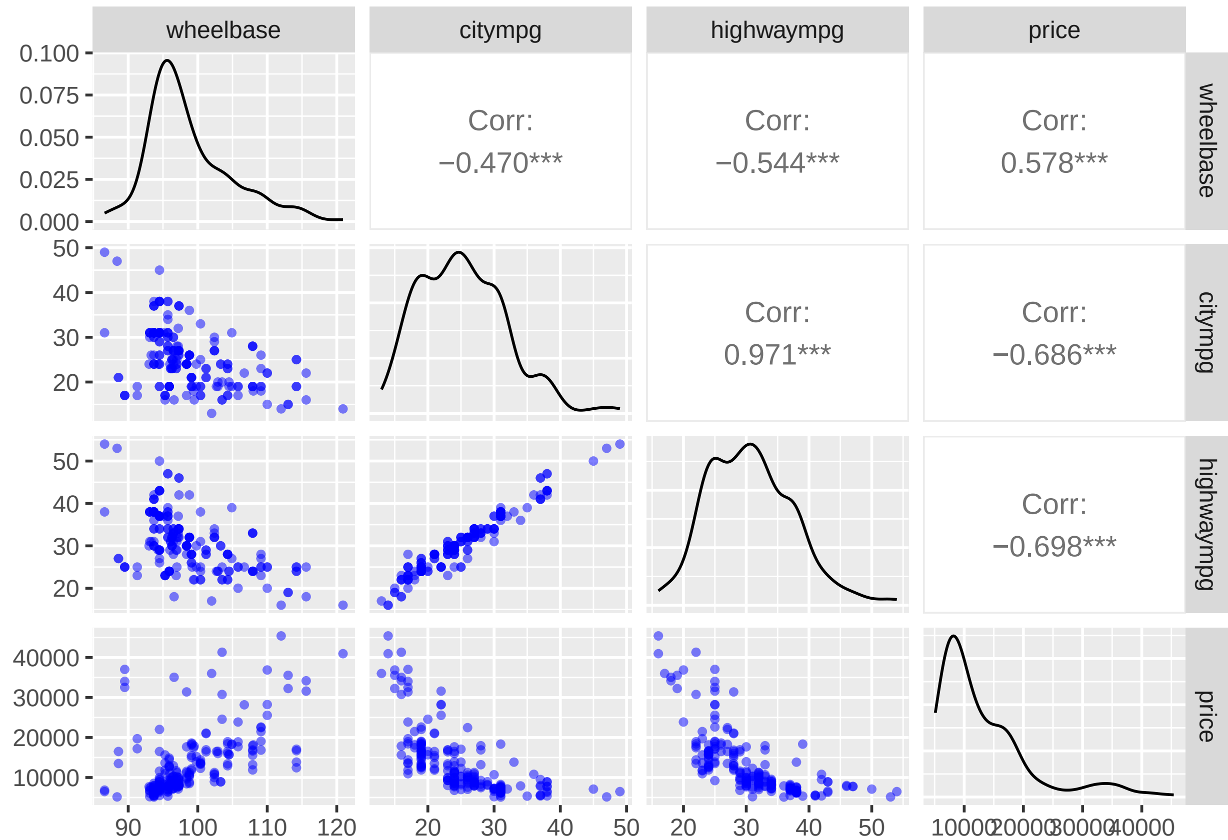
```
# Create a function for pair plots
pp <- function(x,y,z) {
  p <- ggpairs(cars, columns = c(x,y,z, "price"),
               lower = list(continuous = wrap("points", alpha = 0.5, size = 1, color = "blue")))
  print(p)
}

# Plot pair plots
pp('enginesize', 'boreratio', 'stroke')
```




```
pp('compressionratio', 'horsepower', 'peakrpm')
```




```
pp('wheelbase', 'citympg', 'highwaympg')
```



Based on the analysis of the data, it appears that certain attributes show a significant correlation with the price of the car:

Engine Size, Bore Ratio, Horsepower, and Wheelbase: These attributes exhibit a notable positive correlation with the price of the car. As these values increase, the price of the car tends to increase as well.

City MPG and Highway MPG: These attributes display a significant negative correlation with the price of the car. Higher city and highway mileage (measured in miles per gallon) are associated with lower car prices.

It is important to note that correlation does not imply causation, and further analysis is required to understand the precise relationships between these attributes and car prices.

```
# Correlation
cor(cars$carlength, cars$carwidth)
```

```
## [1] 0.8411183
```

Feature Engineering

```
# Fuel economy
cars$fueleconomy <- (0.55 * cars$citympg) + (0.45 * cars$highwaympg)

# Binning the Car Companies based on avg prices of each Company
cars$price <- as.integer(cars$price)
temp <- cars
table <- temp %>%
```

```r
  group_by(CompanyName) %>%
  summarise(mean_price = mean(price, na.rm = TRUE))

temp <- merge(temp, table, by = "CompanyName")
bins <- c(0,10000,20000,40000)
cars_bin <- c('Budget','Medium','Highend')
cars$carsrange <- cut(temp$mean_price, breaks = bins, labels = cars_bin, include.lowest = TRUE)

head(cars)
```

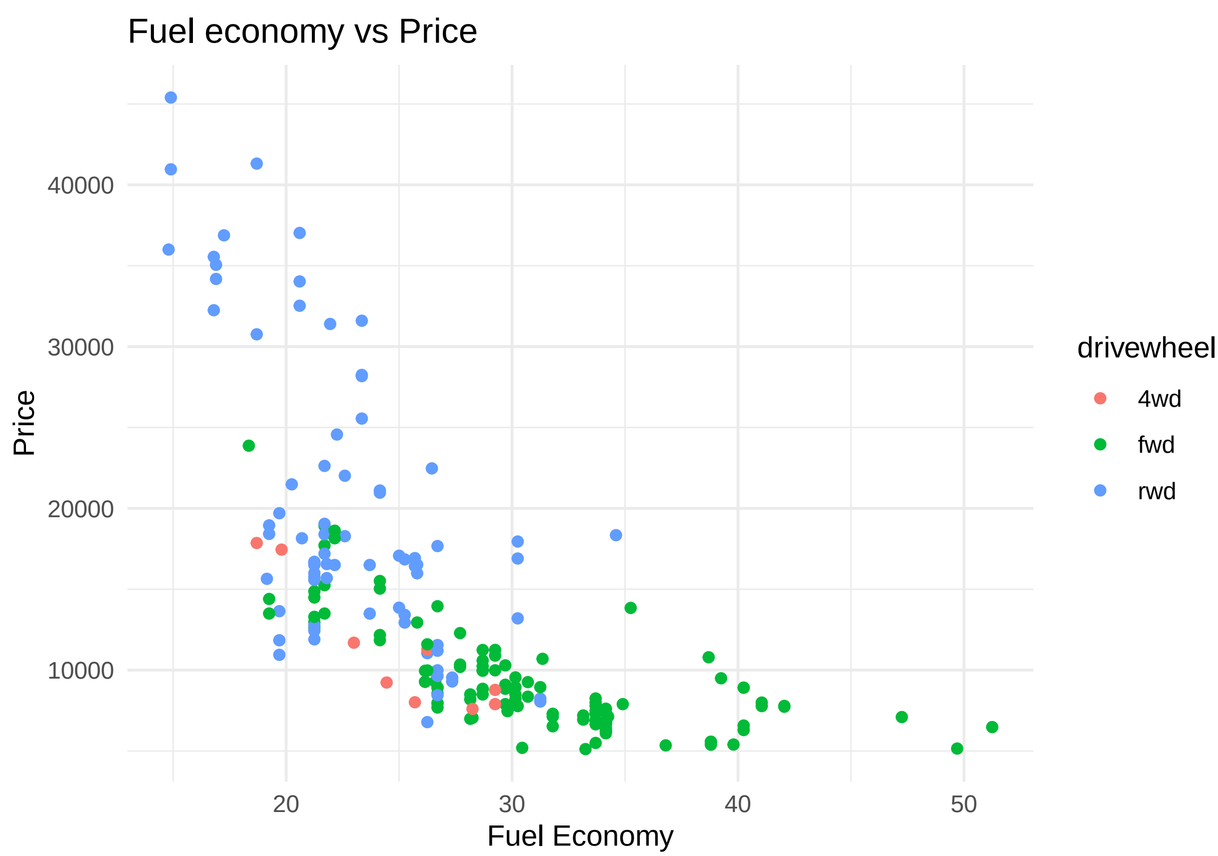
```
##   car_ID symboling fueltype aspiration doornumber     carbody drivewheel
## 1      1         3      gas        std        two convertible        rwd
## 2      2         3      gas        std        two convertible        rwd
## 3      3         1      gas        std        two   hatchback        rwd
## 4      4         2      gas        std       four       sedan        fwd
## 5      5         2      gas        std       four       sedan        4wd
## 6      6         2      gas        std        two       sedan        fwd
##   enginelocation wheelbase carlength carwidth carheight curbweight enginetype
## 1          front      88.6     168.8     64.1      48.8       2548       dohc
## 2          front      88.6     168.8     64.1      48.8       2548       dohc
## 3          front      94.5     171.2     65.5      52.4       2823       ohcv
## 4          front      99.8     176.6     66.2      54.3       2337        ohc
## 5          front      99.4     176.6     66.4      54.3       2824        ohc
## 6          front      99.8     177.3     66.3      53.1       2507        ohc
##   cylindernumber enginesize fuelsystem boreratio stroke compressionratio
## 1           four        130       mpfi      3.47   2.68              9.0
## 2           four        130       mpfi      3.47   2.68              9.0
## 3            six        152       mpfi      2.68   3.47              9.0
## 4           four        109       mpfi      3.19   3.40             10.0
## 5           five        136       mpfi      3.19   3.40              8.0
## 6           five        136       mpfi      3.19   3.40              8.5
##   horsepower peakrpm citympg highwaympg price CompanyName fueleconomy carsrange
## 1        111    5000      21         27 13495 alfa-romero       23.70    Medium
## 2        111    5000      21         27 16500 alfa-romero       23.70    Medium
## 3        154    5000      19         26 16500 alfa-romero       22.15    Medium
## 4        102    5500      24         30 13950        audi       26.70    Medium
## 5        115    5500      18         22 17450        audi       19.80    Medium
## 6        110    5500      19         25 15250        audi       21.70    Medium
```

- Fuel Economy: A new feature called "fueleconomy" is created by combining the city and highway mileage of the cars. The fuel economy is calculated using a weighted average formula, where 55% weight is given to city mileage (cars$citympg) and 45$highwaympg).
- Binning Car Companies: The car companies are grouped based on the average prices of their cars. The "price" column is converted to integer format. The dataset "temp" is created as a copy of the original dataset "cars". Then, a table is generated by grouping "temp" by "CompanyName" and calculating the mean price for each company. The resulting table is merged with "temp" based on the "CompanyName" column. Bins are defined as 0-10,000, 10,000-20,000, and above 20,000. The labels "Budget," "Medium," and "Highend" are assigned to the corresponding price ranges. The new column "carsrange" is created to store the bin labels.

These feature engineering steps aim to enhance the dataset by incorporating additional information such as fuel economy and categorizing car companies based on their average prices. These engineered features can provide valuable insights and potentially improve the performance of predictive models or analysis conducted on the dataset.

Bivariate Analysis

```
# Scatter plot with hue
p <- ggplot(cars, aes(x = fueleconomy, y = price, color = drivewheel)) +
  geom_point() +
  labs(title = "Fuel economy vs Price",
       x = "Fuel Economy",
       y = "Price") +
  theme_minimal()

print(p)
```



Based on the feature engineering performed, the newly created feature "fueleconomy" shows a clear and significant negative correlation with the price of the cars. This implies that as the fuel economy (measured by the combined city and highway mileage) improves, the price of the car tends to decrease. This relationship between fuel economy and price can provide valuable insights for analyzing the pricing dynamics of the cars in the dataset.

```
library(reshape2)
```

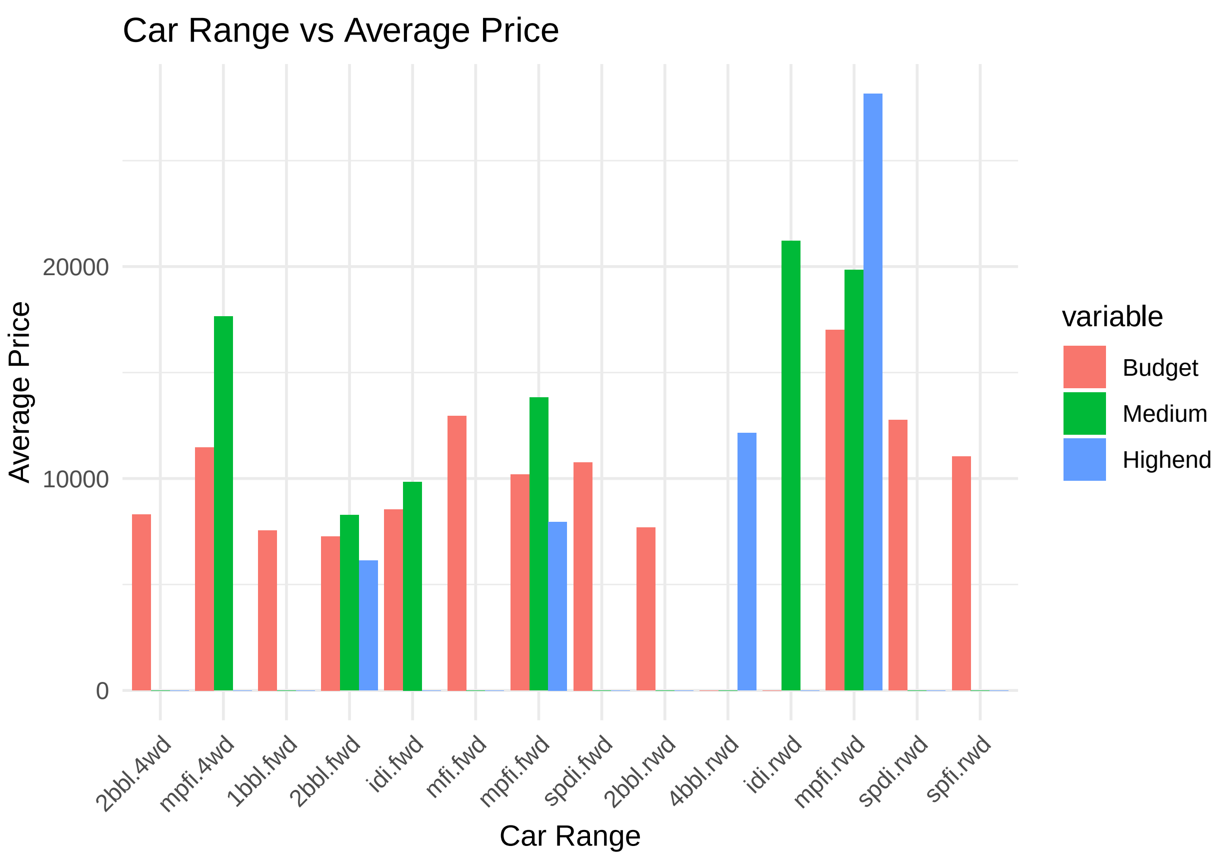
```
##
##    'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
# Group by multiple columns and bar plot
df <- cars %>%
  group_by(fuelsystem, drivewheel, carsrange) %>%
  summarise(mean_price = mean(price, na.rm = TRUE)) %>%
  ungroup() %>%
  spread(key = carsrange, value = mean_price, fill = 0)
```

```
## `summarise()` has grouped output by 'fuelsystem', 'drivewheel'. You can
## override using the `.groups` argument.
```

```r
df_long <- melt(df, id.vars = c("fuelsystem", "drivewheel"))
p <- ggplot(df_long, aes(x = interaction(fuelsystem, drivewheel), y = value, fill = variable)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Car Range vs Average Price",
       x = "Car Range",
       y = "Average Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

print(p)
```



Cars in the higher price range tend to have a preference for rear-wheel drive (rwd) as the drivewheel type. Additionally, they are more likely to feature indirect injection (idi) or multi-point fuel injection (mpfi) as the fuel system.

After conducting a visual analysis, the following variables have been identified as significant:

- Car Range
- Engine Type
- Fuel Type
- Car Body
- Aspiration
- Cylinder Number
- Drivewheel
- Curbweight
- Car Length
- Car Width
- Engine Size
- Boreratio
- Horsepower
- Wheelbase
- Fuel Economy

These variables have shown meaningful patterns, relationships, or variations that are visually apparent and may have an impact on the pricing or other aspects of the cars. Further analysis can be conducted on these variables to gain deeper insights and understand their influence on the target variable or other relevant factors in the dataset.

```
# Subset of cars dataframe
cars_lr <- cars[c('price', 'fueltype', 'aspiration','carbody', 'drivewheel','wheelbase',
                  'curbweight', 'enginetype', 'cylindernumber', 'enginesize', 'boreratio','horsepower',
                    'fueleconomy', 'carlength','carwidth', 'carsrange')]
```

Encoding

Create Dummy Variable

Next, to facilitate regression, we create dummy variables for the categorical variables

```
colnames(cars)# Defining the function to create dummy variables
```

```
##  [1] "car_ID"           "symboling"        "fueltype"         "aspiration"
##  [5] "doornumber"       "carbody"          "drivewheel"       "enginelocation"
##  [9] "wheelbase"        "carlength"        "carwidth"         "carheight"
## [13] "curbweight"       "enginetype"       "cylindernumber"   "enginesize"
## [17] "fuelsystem"       "boreratio"        "stroke"           "compressionratio"
## [21] "horsepower"       "peakrpm"          "citympg"          "highwaympg"
## [25] "price"            "CompanyName"      "fueleconomy"      "carsrange"
```

```
# Defining the function to create dummy variables
dummies <- function(x, df) {
  temp <- model.matrix(~0 + df[[x]])
  temp <- temp[, -1] # drop the first column to avoid dummy variable trap
  names(temp) <- paste(x, names(temp), sep = "_")
  df <- cbind(df, temp)
  df[[x]] <- NULL
  return(df)
}

# Applying the function to the cars_lr
```

```r
cars_lr <- dummies('fueltype', cars_lr)
cars_lr <- dummies('aspiration', cars_lr)
cars_lr <- dummies('carbody', cars_lr)
cars_lr <- dummies('drivewheel', cars_lr)
cars_lr <- dummies('enginetype', cars_lr)
cars_lr <- dummies('cylindernumber', cars_lr)
cars_lr <- dummies('carsrange', cars_lr)

# Checking the first few rows and the dimensions of the dataframe
head(cars_lr)
```

```
##   price wheelbase curbweight enginesize boreratio horsepower fueleconomy
## 1 13495      88.6       2548        130      3.47        111       23.70
## 2 16500      88.6       2548        130      3.47        111       23.70
## 3 16500      94.5       2823        152      2.68        154       22.15
## 4 13950      99.8       2337        109      3.19        102       26.70
## 5 17450      99.4       2824        136      3.19        115       19.80
## 6 15250      99.8       2507        136      3.19        110       21.70
##   carlength carwidth temp temp df[[x]]hardtop df[[x]]hatchback df[[x]]sedan
## 1     168.8     64.1    1    0              0                0            0
## 2     168.8     64.1    1    0              0                0            0
## 3     171.2     65.5    1    0              0                1            0
## 4     176.6     66.2    1    0              0                0            1
## 5     176.6     66.4    1    0              0                0            1
## 6     177.3     66.3    1    0              0                0            1
##   df[[x]]wagon df[[x]]fwd df[[x]]rwd df[[x]]dohcv df[[x]]l df[[x]]ohc
## 1            0          0          1            0        0          0
## 2            0          0          1            0        0          0
## 3            0          0          1            0        0          0
## 4            0          1          0            0        0          1
## 5            0          0          0            0        0          1
## 6            0          1          0            0        0          1
##   df[[x]]ohcf df[[x]]ohcv df[[x]]rotor df[[x]]five df[[x]]four df[[x]]six
## 1           0           0            0           0           1          0
## 2           0           0            0           0           1          0
## 3           0           1            0           0           0          1
## 4           0           0            0           0           1          0
## 5           0           0            0           1           0          0
## 6           0           0            0           1           0          0
##   df[[x]]three df[[x]]twelve df[[x]]two df[[x]]Medium df[[x]]Highend
## 1            0             0          0             1              0
## 2            0             0          0             1              0
## 3            0             0          0             1              0
## 4            0             0          0             1              0
## 5            0             0          0             1              0
## 6            0             0          0             1              0
```

```r
dim(cars_lr)
```

```
## [1] 205  31
```

Train-Test Split and feature scaling

```
library(caret)
```

```
##      lattice

##
##    'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(corrr)

# Setting the seed to make the partition reproducible
set.seed(100)

temp_positions <- which(names(cars_lr) == "temp")
names(cars_lr)[temp_positions[1]] <- "gas"
names(cars_lr)[temp_positions[2]] <- "turbo"

# Splitting the data into training set and test set
trainIndex <- createDataPartition(cars_lr$price, p = .7, list = FALSE, times = 1)
df_train <- cars_lr[ trainIndex,]
df_test  <- cars_lr[-trainIndex,]

# Rename the columns of the dataframe
names(df_train) <- make.names(names(df_train))
names(df_test) <- make.names(names(df_test))

# Scaling the numeric variables
num_vars <- c('wheelbase', 'curbweight', 'enginesize', 'boreratio', 'horsepower','fueleconomy','carlength'
preProc <- preProcess(df_train[,num_vars], method = c("scale"), range = c(0, 1))
df_train[,num_vars] <- predict(preProc, df_train[,num_vars])

# Checking the first few rows and the summary of the training set
head(df_train)
```

```
##      price wheelbase curbweight enginesize boreratio horsepower fueleconomy
## 1 1.639889  15.84834   5.054504   3.143711 12.746232   2.637355    3.469641
## 2 2.005051  15.84834   5.054504   3.143711 12.746232   2.637355    3.469641
## 3 2.005051  16.90371   5.600025   3.675724  9.844352   3.659033    3.242723
## 4 1.695180  17.85175   4.635940   2.635881 11.717718   2.423515    3.908836
## 5 2.120494  17.78020   5.602009   3.288806 11.717718   2.732395    2.898687
## 6 1.853153  17.85175   4.973172   3.288806 11.717718   2.613595    3.176844
##   carlength carwidth gas turbo df..x..hardtop df..x..hatchback df..x..sedan
## 1  14.00992 30.62981   1     0              0                0            0
## 2  14.00992 30.62981   1     0              0                0            0
## 3  14.20911 31.29879   1     0              0                1            0
## 4  14.65729 31.63328   1     0              0                0            1
## 5  14.65729 31.72885   1     0              0                0            1
```

```
## 6  14.71539 31.68106   1     0              0                0            1
##   df..x..wagon df..x..fwd df..x..rwd df..x..dohcv df..x..l df..x..ohc
## 1            0          0          1            0        0          0
## 2            0          0          1            0        0          0
## 3            0          0          1            0        0          0
## 4            0          1          0            0        0          1
## 5            0          0          0            0        0          1
## 6            0          1          0            0        0          1
##   df..x..ohcf df..x..ohcv df..x..rotor df..x..five df..x..four df..x..six
## 1           0           0            0           0           1          0
## 2           0           0            0           0           1          0
## 3           0           1            0           0           0          1
## 4           0           0            0           0           1          0
## 5           0           0            0           1           0          0
## 6           0           0            0           1           0          0
##   df..x..three df..x..twelve df..x..two df..x..Medium df..x..Highend
## 1            0             0          0             1              0
## 2            0             0          0             1              0
## 3            0             0          0             1              0
## 4            0             0          0             1              0
## 5            0             0          0             1              0
## 6            0             0          0             1              0
```

```
summary(df_train)
```

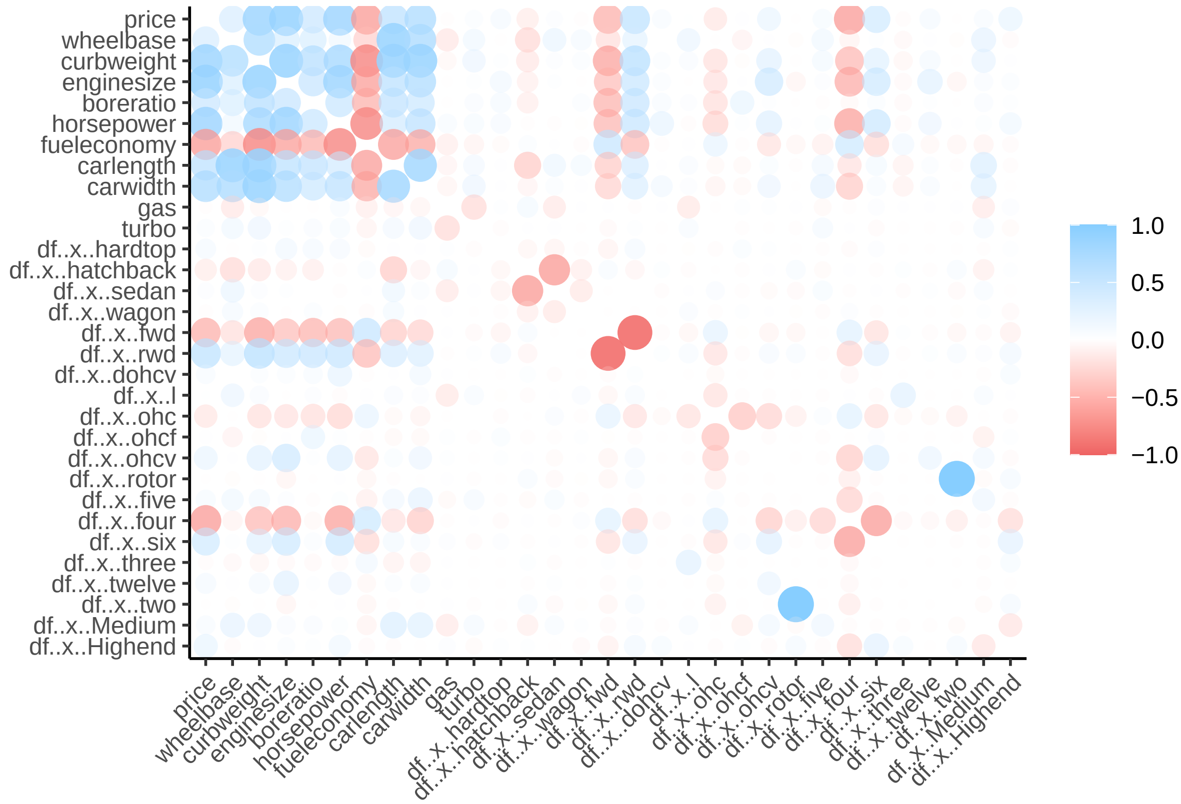
```
##      price          wheelbase       curbweight      enginesize
##  Min.   :0.6219   Min.   :15.49   Min.   :2.952   Min.   :1.475
##  1st Qu.:0.9448   1st Qu.:16.90   1st Qu.:4.255   1st Qu.:2.346
##  Median :1.2510   Median :17.35   Median :4.811   Median :2.902
##  Mean   :1.6244   Mean   :17.56   Mean   :5.025   Mean   :3.071
##  3rd Qu.:2.0054   3rd Qu.:18.10   3rd Qu.:5.642   3rd Qu.:3.410
##  Max.   :5.5169   Max.   :20.68   Max.   :8.066   Max.   :7.883
##    boreratio        horsepower      fueleconomy      carlength
##  Min.   : 9.844   Min.   :1.140   Min.   :2.167   Min.   :11.71
##  1st Qu.:11.571   1st Qu.:1.663   1st Qu.:3.243   1st Qu.:13.80
##  Median :12.232   Median :2.233   Median :3.909   Median :14.36
##  Mean   :12.261   Mean   :2.493   Mean   :4.081   Mean   :14.38
##  3rd Qu.:13.187   3rd Qu.:2.756   3rd Qu.:4.655   3rd Qu.:14.81
##  Max.   :14.473   Max.   :6.843   Max.   :7.503   Max.   :16.82
##     carwidth          gas              turbo        df..x..hardtop
##  Min.   :28.81   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:30.63   1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :31.30   Median :1.0000   Median :0.0000   Median :0.00000
##  Mean   :31.45   Mean   :0.9103   Mean   :0.1586   Mean   :0.04828
##  3rd Qu.:31.78   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :34.55   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  df..x..hatchback  df..x..sedan     df..x..wagon       df..x..fwd
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.00000   Median :1.0000
##  Mean   :0.3655   Mean   :0.4552   Mean   :0.09655   Mean   :0.5793
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##    df..x..rwd      df..x..dohcv         df..x..l          df..x..ohc
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.00000   Min.   :0.0000
```

```
##  1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.000000   Median :0.00000   Median :1.0000
##  Mean   :0.3793   Mean   :0.006897   Mean   :0.04138   Mean   :0.7379
##  3rd Qu.:1.0000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.000000   Max.   :1.00000   Max.   :1.0000
##   df..x..ohcf       df..x..ohcv       df..x..rotor      df..x..five
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.08276   Mean   :0.06207   Mean   :0.02069   Mean   :0.05517
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   df..x..four       df..x..six      df..x..three      df..x..twelve
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:1.0000   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :1.0000   Median :0.0000   Median :0.000000   Median :0.000000
##  Mean   :0.7655   Mean   :0.1241   Mean   :0.006897   Mean   :0.006897
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.000000
##    df..x..two      df..x..Medium df..x..Highend
##  Min.   :0.00000   Min.   :0.0   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0   1st Qu.:0.0000
##  Median :0.00000   Median :0.0   Median :0.0000
##  Mean   :0.02069   Mean   :0.4   Mean   :0.1379
##  3rd Qu.:0.00000   3rd Qu.:1.0   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.0   Max.   :1.0000
```

```r
# Correlation using heatmap
corr_df <- df_train %>% correlate()
```

```
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'
```

```r
corr_df %>% rplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



There are perfectly correlated features that delete to prevent perfect multicollinearity.

```
df_train <- df_train[, -which(names(df_train) == "df..x..two")]
df_test <- df_test[, -which(names(df_test) == "df..x..two")]
rownames(df_train) <- NULL
```

```
# Dividing data into X and y variables
y_train <- df_train$price
X_train <- df_train[, !names(df_train) %in% 'price']
```

Regression model Building

Feature Selection

Validation Set Approach

Performance on unknown data is an important metric for selecting a model, and we do this here using the Validation Set Approach. I started by using regsubsets to find the best model with the number of different predictors. Then find the model with the best generalization ability among these models.

```
#### Load necessary libraries
library(ISLR)
library(leaps)
library(glmnet)
```

```
##      Matrix
```

```
##
##    'Matrix'
```

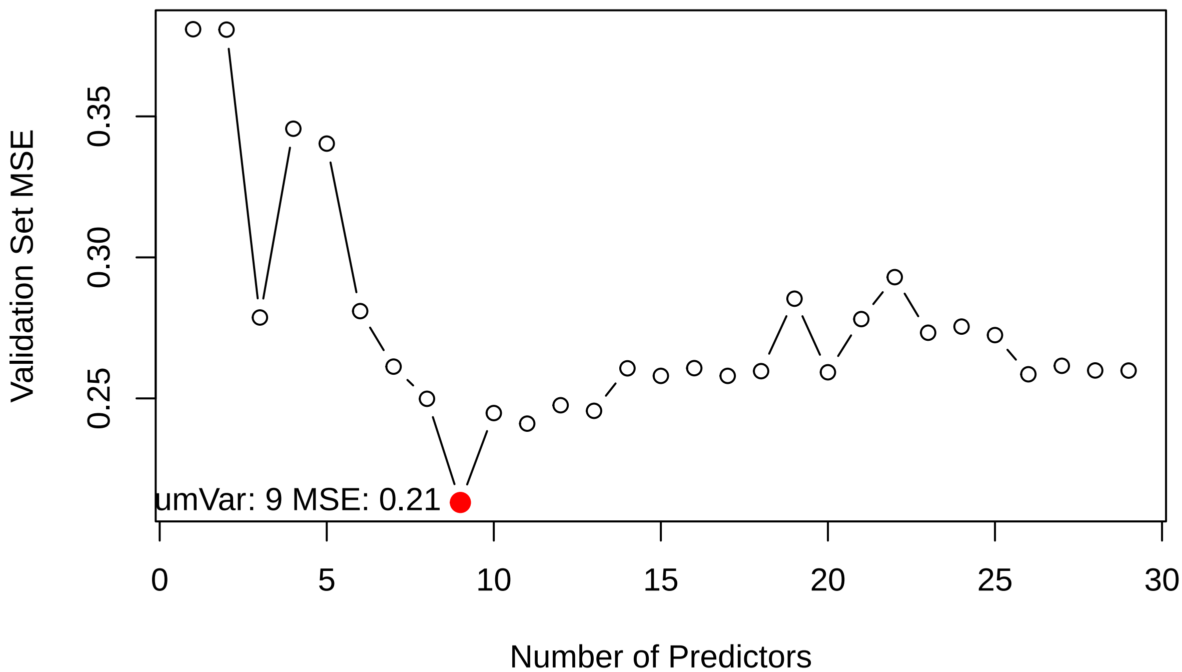
```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-3
```

```r
nvmax <- ncol(df_train)

# Split data into training and validation set
set.seed(0)

# Determine the number of rows to include in the training set (80% of the total)
train_size <- floor(0.8 * nrow(df_train))

train <- sample(seq_len(nrow(df_train)), size = train_size)
test <- (-train)

# Fit a model on the training set, and evaluate its MSE on the test set
regfit.best <- regsubsets(price~., data = df_train[train, ], nvmax=nvmax)
test.df <- df_train[test, ]

# Function to get predictions
get.regsubsets.predictions <- function(object, newdata, id, ...){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[,xvars] %*% coefi
}

# Get the best model's predictions
val.errors <- rep(NA, (nvmax-1))
for(i in 1:(nvmax-1)){
  pred <- get.regsubsets.predictions(regfit.best, test.df, i)
  val.errors[i] <- mean((df_train$price[test]-pred)^2)
}

# Find the model with the lowest validation set error
print(which.min(val.errors))
```

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

```r
# Plot MSE against number of predictors
plot(val.errors, type = "b", xlab = "Number of Predictors", ylab = "Validation Set MSE")
points(which.min(val.errors), val.errors[which.min(val.errors)], col = "red", cex = 2, pch = 20)

text(which.min(val.errors), val.errors[which.min(val.errors)], labels = paste("NumVar:", which.min(val.err
```



We can find that the regression model with 9 features performs best on the verification set

But the Validation Set Approach has drawbacks:

Low data utilization: The Validation Set Approach uses only a portion of the data as the validation set, while the rest of the data is used to train the model. This can lead to inaccurate estimates of model performance on validation sets, especially when the data sets are small.

There may be large variance: Because the way the Validation Set is divided may lead to different model performance evaluation results, the evaluation results of the Validation Set Approach may have large variance.

Cross-Validation

Therefore, using cross validation may be a better approach

```
library(leaps)
library(boot)
```

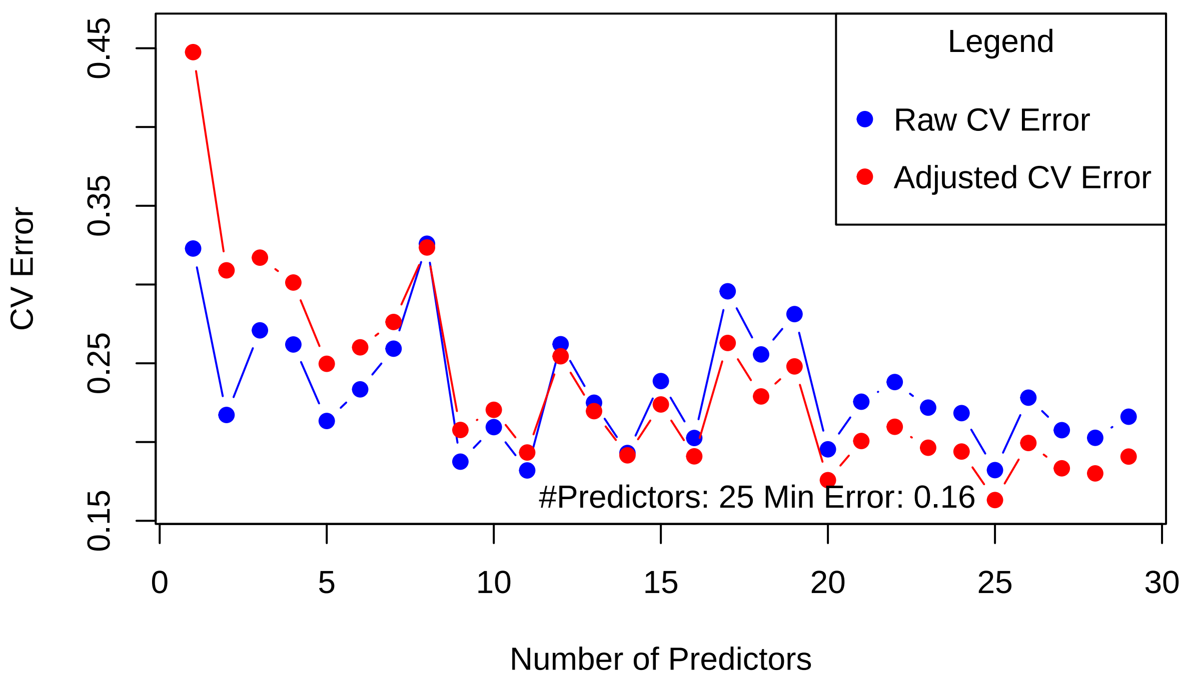
```
##
##    'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
set.seed(0)
# Fit the model
regfit.best <- regsubsets(price~., data = df_train, nvmax = nvmax)

# Create a matrix to store the results
cv.errors <- matrix(NA, nvmax-1, 2)

# Extract the column names of the data
colnames <- colnames(df_train[,names(df_train) != "price"])

for(i in 1:(nvmax-1)){
  # Get the best model of size i
  coefi <- coef(regfit.best, id = i)

  # Get the names of the predictors for the best model of size i
  predictors <- names(coefi)[2:length(coefi)]

  # Fit the GLM model using these predictors
  glm.fit <- glm(price ~ ., data = df_train[, c(predictors, "price")])

  # Perform k-fold cross-validation
  cv.errors[i,] <- cv.glm(df_train, glm.fit, K = 5)$delta
}

# The MSE for the best models of each size
cv.errors
```

```
##            [,1]      [,2]
##  [1,] 0.3228558 0.4475433
##  [2,] 0.2171872 0.3090003
##  [3,] 0.2709069 0.3170641
##  [4,] 0.2619464 0.3012703
##  [5,] 0.2133746 0.2496796
##  [6,] 0.2334430 0.2601299
##  [7,] 0.2592687 0.2762141
##  [8,] 0.3259058 0.3235116
##  [9,] 0.1875377 0.2076653
## [10,] 0.2094833 0.2204615
## [11,] 0.1819783 0.1933615
## [12,] 0.2621109 0.2544536
## [13,] 0.2249598 0.2196138
## [14,] 0.1930282 0.1915646
## [15,] 0.2387004 0.2238613
## [16,] 0.2025715 0.1909230
## [17,] 0.2957049 0.2629006
## [18,] 0.2556204 0.2289536
## [19,] 0.2812240 0.2480194
## [20,] 0.1953809 0.1758017
## [21,] 0.2255974 0.2006532
## [22,] 0.2380921 0.2096832
## [23,] 0.2218365 0.1963915
## [24,] 0.2183639 0.1939588
## [25,] 0.1821665 0.1631722
## [26,] 0.2281643 0.1993989
## [27,] 0.2075274 0.1833171
```

```
## [28,] 0.2026842 0.1800942
## [29,] 0.2160730 0.1907592
```

```r
# Plot the raw CV errors
plot(1:(nvmax-1), cv.errors[,1], xlab = "Number of Predictors", ylab = "CV Error", type = "b", pch = 19, c

# Add points for the adjusted CV errors
points(1:(nvmax-1), cv.errors[,2], col = "red", pch = 19, type = "b")

# Find the minimum CV error and its corresponding index
min_error <- min(cv.errors[, 2])
min_index <- which.min(cv.errors[, 2])

# Add label for the minimum point
text(min_index, min_error, labels = paste("#Predictors:", min_index,"Min Error:", round(min_error,2)), pos

# Add a legend
# Set the coordinates for the legend (outside the plot area)
legend_x <- "topright"
legend_y <- max(cv.errors) + 0.02

# Add a legend outside the plot
legend(legend_x, legend = c("Raw CV Error", "Adjusted CV Error"), col = c("blue", "red"), pch = 19, inset =
```



In this diagram we can see how the MSE error estimate from cross-validation varies with the number of Predictor counts

We pay more attention to the generalization ability of the model, so we prefer to use Validation Set Approach and

Cross-Validation to select the model. In addition, we wish we could get more robust regression results, but our sample size is small (n=205). As a rule of thumb, regression results are generally reliable when the sample size is 15 or more times the number of variables, so only use models with less than 14 Predictor numbers. So let's choose the model with 11 Predictor. The model has performed well in Validation Set Approach and Cross-Validation, and has shown reliable performance in Cp, BIC, and adjusted R^2.

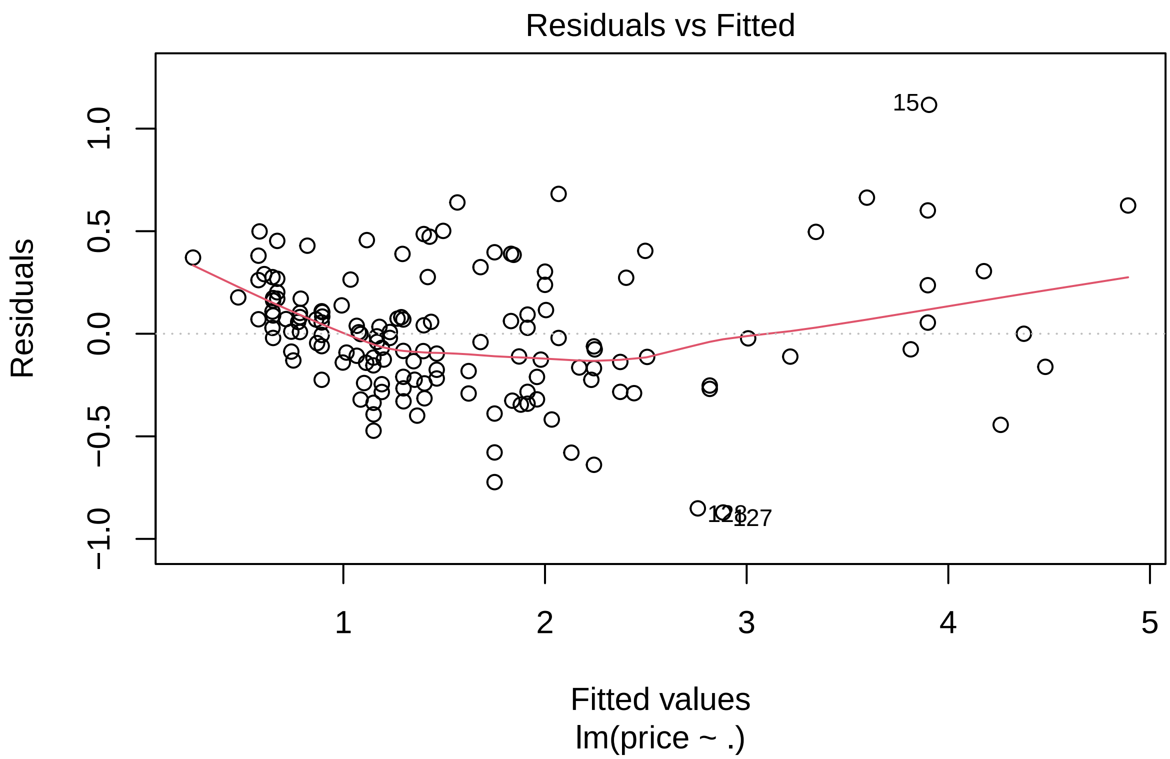
```r
# Get the coefficients of the best model of size 9
coefi <- coef(regfit.best, id = 11)

# Get the names of the predictors for the best model of size 9
predictors <- names(coefi)[2:length(coefi)]

# Fit the LM model using these predictors
lm.fit <- lm(price ~ ., data = df_train[, c(predictors, "price")])

# Print the summary of the model
summary(lm.fit)
```

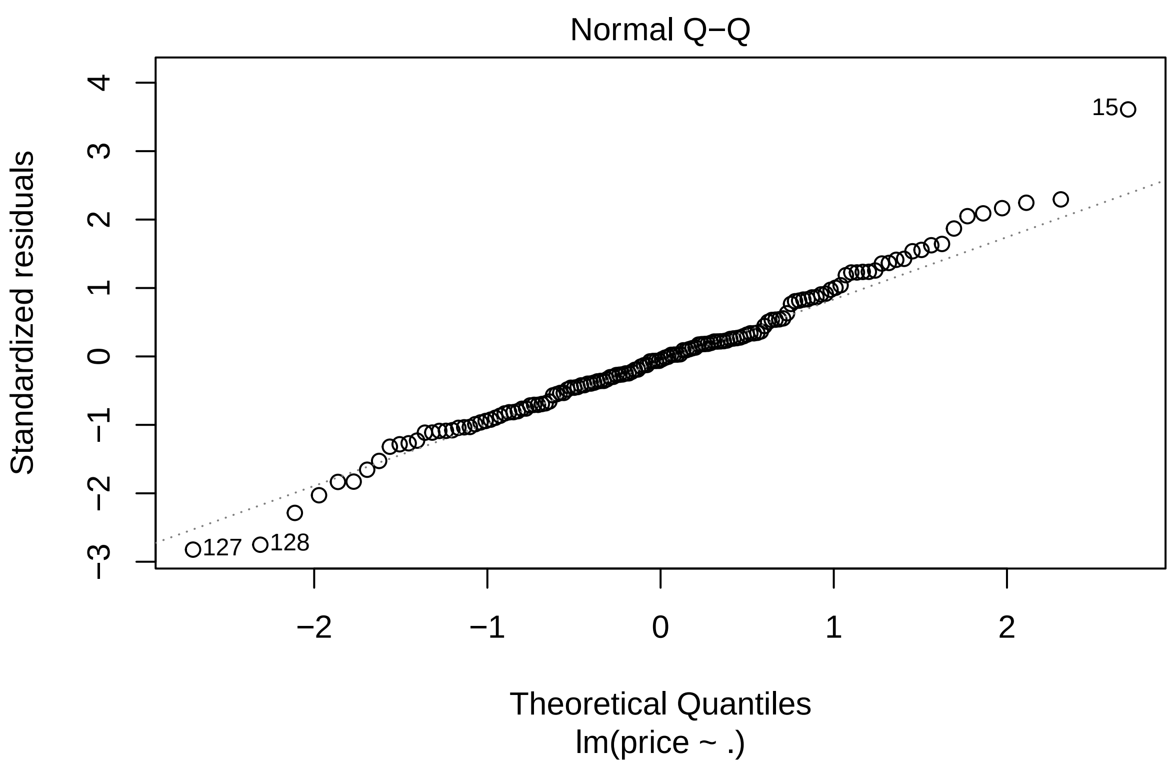
```
##
## Call:
## lm(formula = price ~ ., data = df_train[, c(predictors, "price")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.87130 -0.21007 -0.01253  0.17102  1.11605
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.50918    1.37420  -3.281 0.001320 **
## enginesize      0.80386    0.07313  10.992  < 2e-16 ***
## boreratio      -0.23876    0.04950  -4.823 3.81e-06 ***
## horsepower      0.14034    0.05414   2.592 0.010602 *
## carwidth        0.18980    0.04825   3.934 0.000134 ***
## df..x..rwd      0.25946    0.08539   3.038 0.002863 **
## df..x..ohcf     0.57044    0.12744   4.476 1.62e-05 ***
## df..x..ohcv    -0.84147    0.15919  -5.286 4.99e-07 ***
## df..x..rotor    0.63419    0.23193   2.734 0.007102 **
## df..x..twelve  -1.04336    0.38094  -2.739 0.007010 **
## df..x..Medium   0.26252    0.07023   3.738 0.000274 ***
## df..x..Highend  0.50118    0.09589   5.227 6.51e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3241 on 133 degrees of freedom
## Multiple R-squared:  0.903,  Adjusted R-squared:  0.895
## F-statistic: 112.6 on 11 and 133 DF,  p-value: < 2.2e-16
```

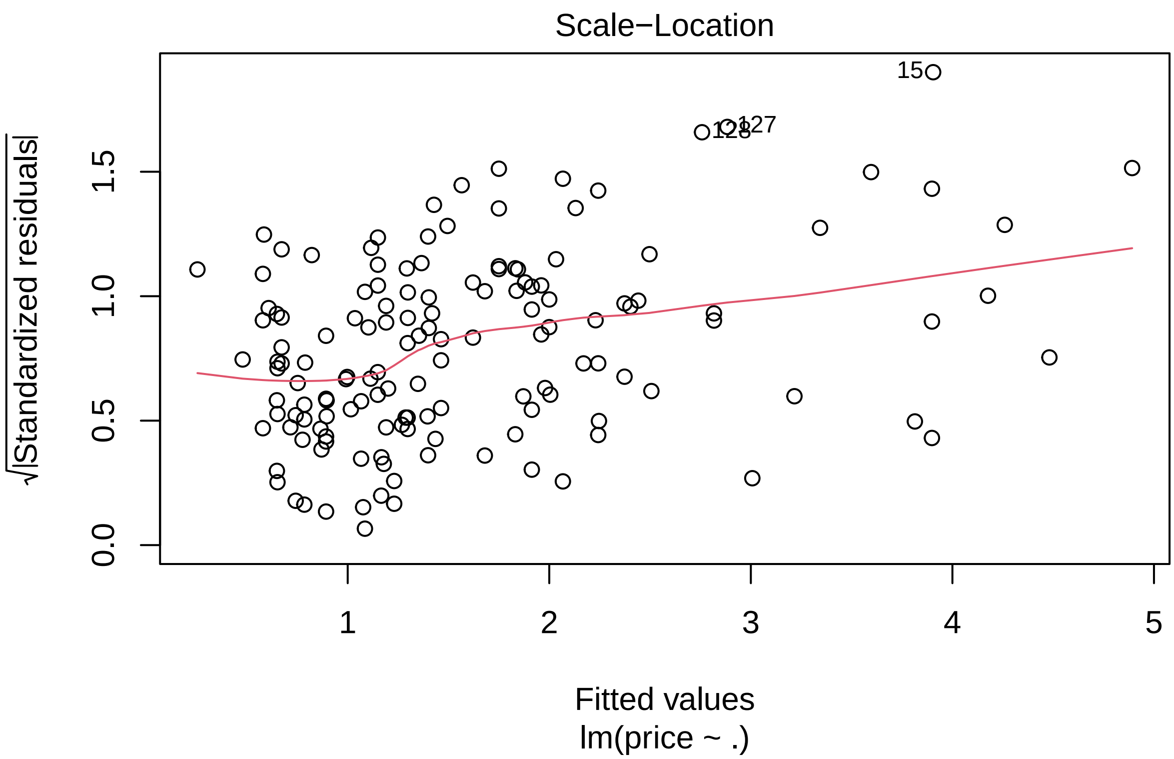
Diagnsitics Test

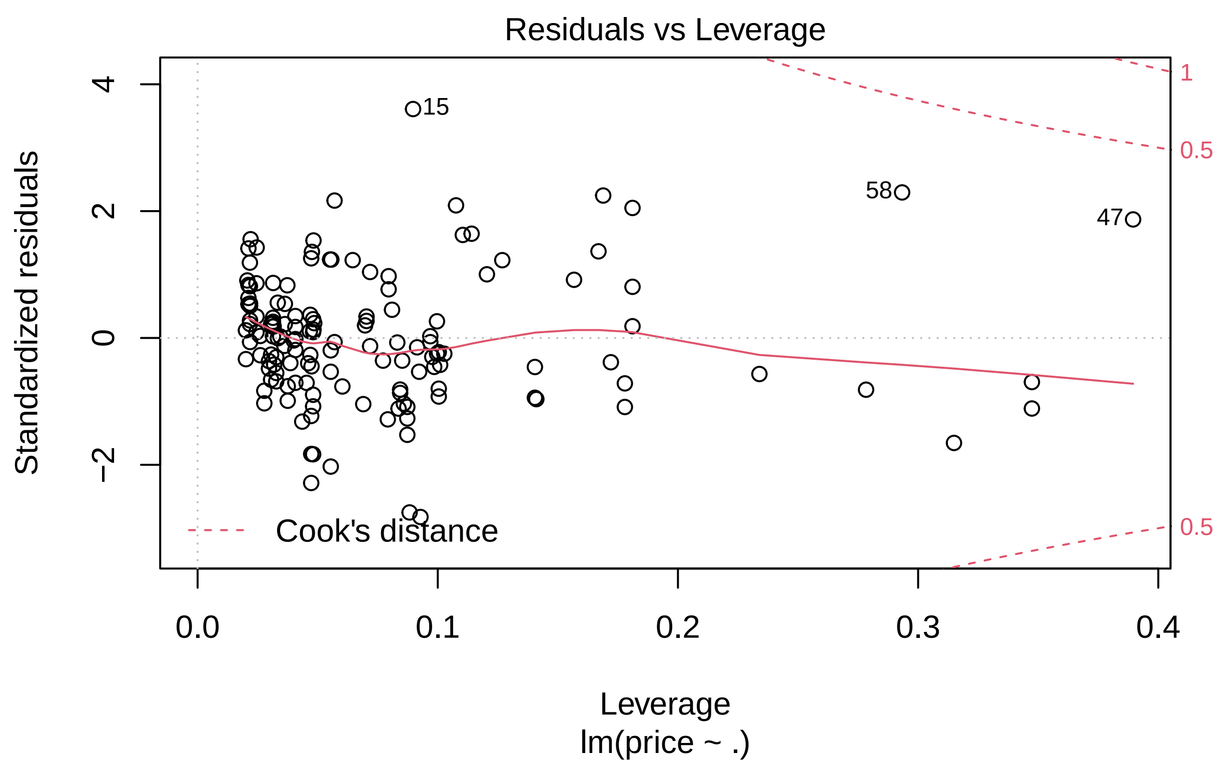
Let's diagnose the regression:

```r
plot(lm.fit)
```


Residuals vs Fitted
Residuals
Fitted values
lm(price ~ .)
15
127
-1.0
-0.5
0.0
0.5
1.0
1
2
3
4
5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(price ~ .)
15
127
128
-3
-2
-1
0
1
2
3
4
-2
-1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
1
2
3
4
5
15
128
127
Fitted values
lm(price ~ .)


Residuals vs Leverage
Standardized residuals
−2
0
2
4
0.0
0.1
0.2
0.3
0.4
15
58
47
1
0.5
0.5
Cook's distance
Leverage
lm(price ~ .)

There are several issues with the model that can be observed:

- Residuals vs Fitted: This plot examines whether the residuals exhibit non-linear patterns, indicating a potential non-linear relationship between predictor variables and the outcome variable. Such patterns suggest that the linear hypothesis is not satisfied, and there may be non-linear components present in the residuals.
- Normal Q-Q: This plot assesses the normality of the residuals. Ideally, the residuals should approximately follow a straight dashed line, indicating a normal distribution.
- Scale-Location (Spread-Location): This plot investigates whether the residuals are equally spread across the ranges of the predictor variables. It helps verify the assumption of equal variance (homoscedasticity). If the spread of residuals varies across predictor ranges, it may imply heteroscedasticity or the failure to capture a non-linear relationship.
- Residuals vs Leverage: This plot helps identify influential cases (i.e., subjects) within the data. Not all outliers have a significant impact on linear regression analysis. While some extreme values may not substantially alter the results if included or excluded from the analysis, as they align with the overall trend, others can greatly influence the regression line, even if they fall within a reasonable value range. These influential cases deviate from the majority trend and may warrant special attention in the analysis.

In addition, we also need to test whether the model has multicollinearity. To assess the presence of multicollinearity, Variance Inflation Factor (VIF) values were calculated for each predictor. The VIF measures the correlation between a predictor and the other predictors in the model.

The Variance Inflation Factor (VIF) measures the inflation of the variance of the estimated regression coefficients due to multicollinearity. The formula for calculating the VIF for a predictor variable is as follows:

$VIF = \frac{1}{1-R^2}$

Where:

R^2 is the coefficient of determination of a regression model in which the predictor variable is regressed against all other predictor variables.

According to James et al. (2013), a VIF value less than 5 indicates a low correlation, a value between 5 and 10 indicates a moderate correlation, and VIF values larger than 10 suggest a high correlation that is not tolerable.

```
library(car)
```

```
##      carData
```

```
##
##    'car'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

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

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
vif(lm.fit)
```

```
##     enginesize      boreratio     horsepower       carwidth     df..x..rwd
##       7.333327       3.359750       4.018803       3.191973       2.370150
##    df..x..ohcf    df..x..ohcv   df..x..rotor  df..x..twelve df..x..Medium
##       1.702170       2.036902       1.504752       1.372185       1.634178
## df..x..Highend
##       1.509310
```

Based on the VIF values, it can be observed that most predictors exhibit a low to moderate correlation with other predictors. It can be observed that most predictors exhibit a low to moderate correlation with other predictors. the variable "enginesize" shows a moderate correlation with other predictors with a VIF value of 7.33. The variable "Enginesize" shows a moderate correlation with other predictors with a vif value of 7.33. So we can say that there is no obvious multicollinearity problem in the model.

Remedial measures

Deal with heteroscedasticity

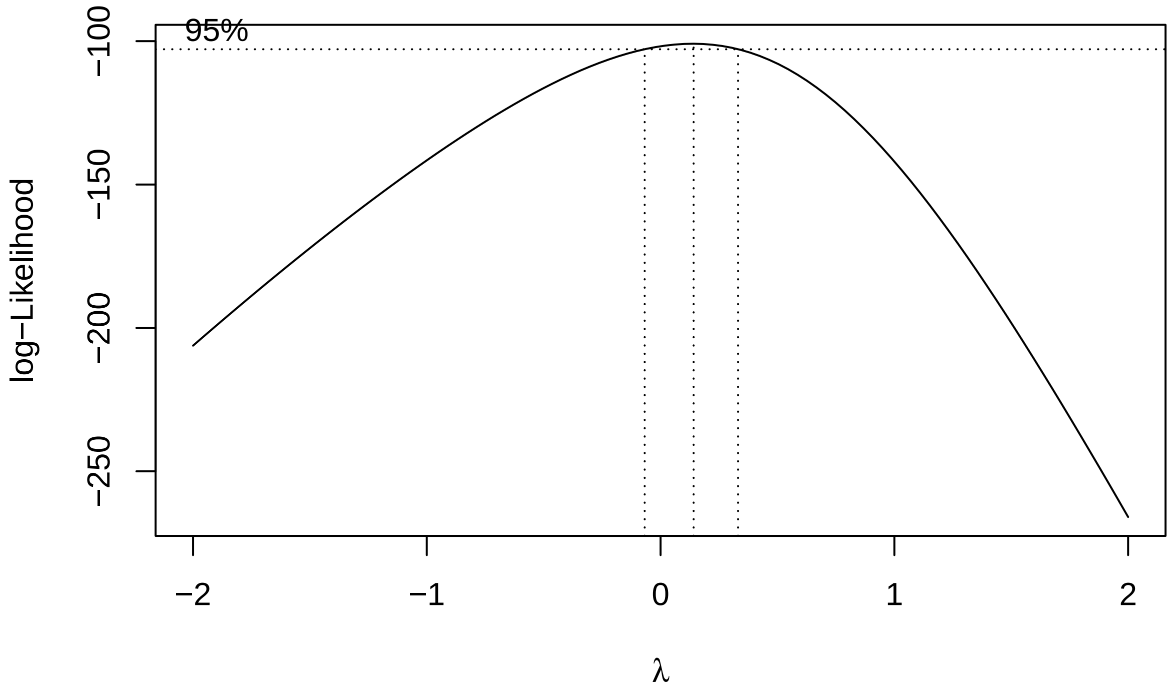
Noting heteroscedasticity, we use Box-Cox transformation to process the data.

```r
library(MASS)
```

```
##
##    'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
# Use Box-Cox transformation to choose parameter
boxcox_results <- boxcox(lm.fit)
```




```r
lambda <- boxcox_results$x[which.max(boxcox_results$y)]

# Print the chosen lambda parameter
print(lambda)
```

```
## [1] 0.1414141
```

So we use the λ that makes log-likelihood maximum, which is $\lambda = 0.1414141$

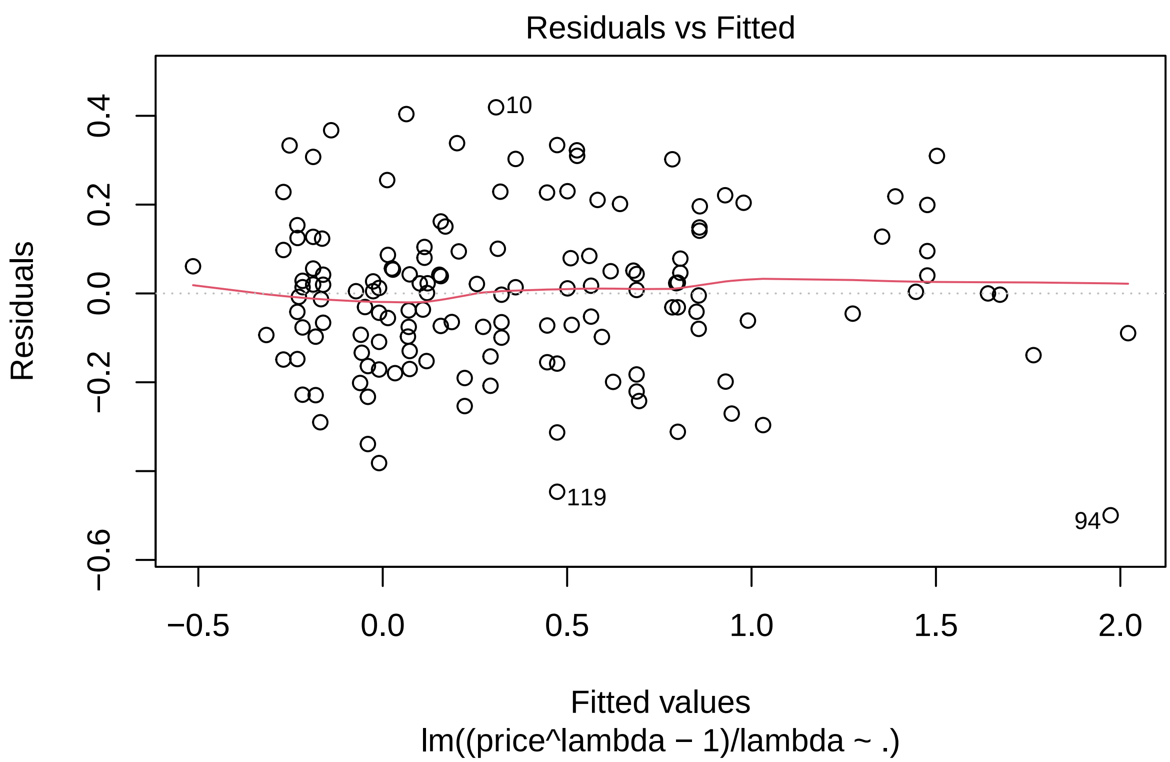
Regression again:

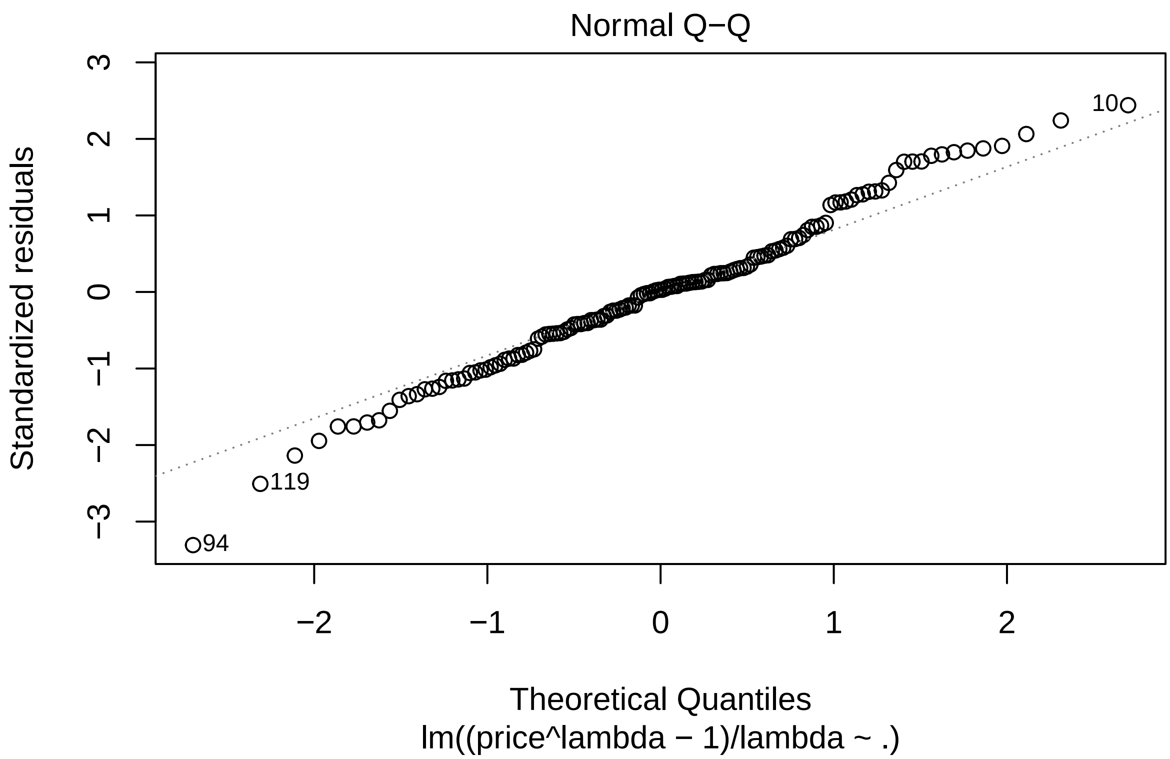
```r
# Fit a new LM model using the transformed response variable and predictors
lm.fit_transformed <- lm((price ^ lambda - 1) / lambda ~ ., data = df_train[, c(predictors, "price")])

# Print the final regression results
summary(lm.fit_transformed)
```

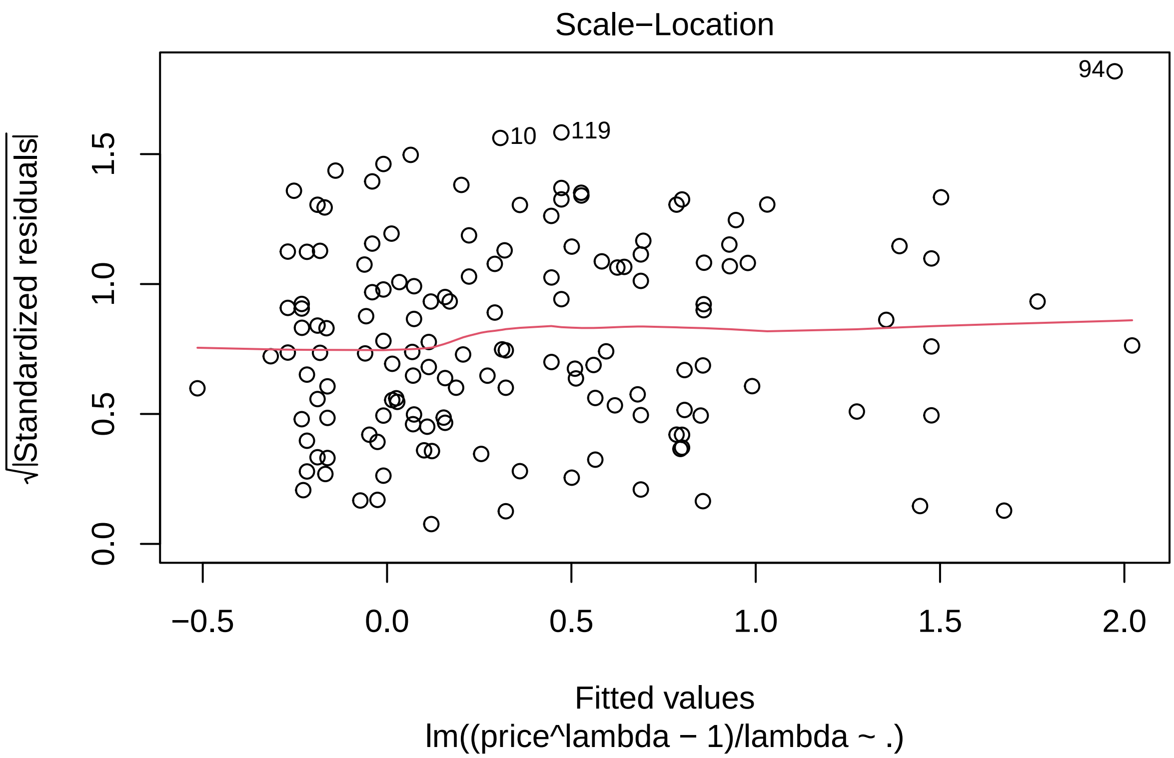
```
##
## Call:
## lm(formula = (price^lambda - 1)/lambda ~ ., data = df_train[,
##     c(predictors, "price")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49942 -0.09797  0.00500  0.09532  0.41906
##
```

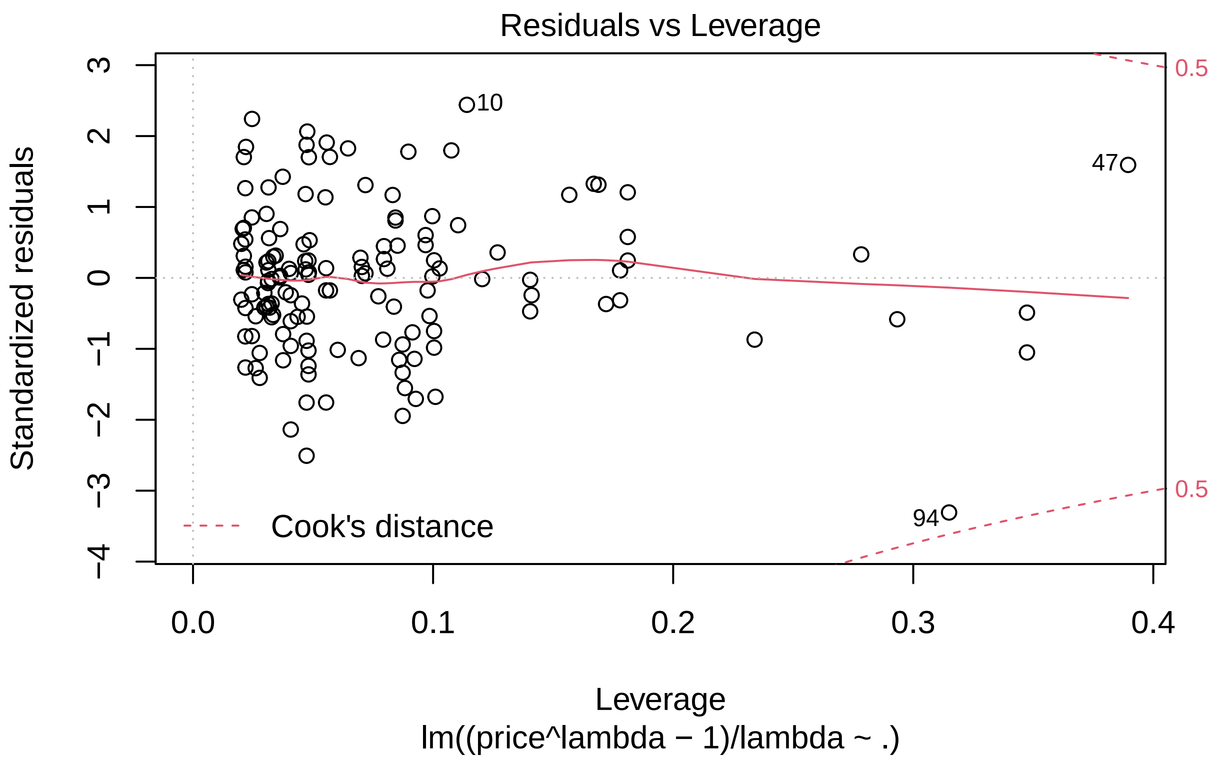
```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      -3.98235    0.77363  -5.148 9.27e-07 ***
## enginesize        0.31985    0.04117   7.769 1.90e-12 ***
## boreratio        -0.07407    0.02787  -2.658 0.008823 ** 
## horsepower        0.14043    0.03048   4.608 9.43e-06 ***
## carwidth          0.11926    0.02716   4.391 2.29e-05 ***
## df..x..rwd        0.18544    0.04807   3.858 0.000178 ***
## df..x..ohcf       0.20442    0.07175   2.849 0.005081 ** 
## df..x..ohcv      -0.42959    0.08962  -4.793 4.32e-06 ***
## df..x..rotor      0.33503    0.13057   2.566 0.011400 *  
## df..x..twelve    -0.58827    0.21446  -2.743 0.006927 ** 
## df..x..Medium     0.21314    0.03954   5.391 3.09e-07 ***
## df..x..Highend    0.19130    0.05398   3.544 0.000545 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1824 on 133 degrees of freedom
## Multiple R-squared:  0.901,  Adjusted R-squared:  0.8928
## F-statistic: 110.1 on 11 and 133 DF,  p-value: < 2.2e-16
```

```
plot(lm.fit_transformed)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm((price^lambda − 1)/lambda ~ .)
10
119
94


Scale−Location
√|Standardized residuals|
Fitted values
lm((price^lambda − 1)/lambda ~ .)
94
10
119



After applying the Box-Cox transformation, we were able to address two issues: the presence of non-linear patterns in the residuals and heteroscedasticity. The plots for Residuals vs Fitted, Normal Q-Q, and Scale-Location now indicate a good fit to the regression hypothesis. However, we have identified a remaining concern related to a high leverage point. Despite the improvements in the other aspects, this particular data point seems to have a significant impact on the regression line. Its influence could potentially distort the results if it is included in the analysis. Therefore, it is important to carefully consider the implications of this high leverage point and determine whether it should be included or excluded from the analysis, based on its alignment with the overall trend and the desired outcome of the regression.

Deal with high leverage points

Calculation of Cook's Distance: Cook's distance measures the influence of each observation on the regression model. It is calculated as the scaled change in predicted values when a particular observation is omitted from the model. Mathematically, Cook's distance for the i-th observation can be expressed as:

$$D_i = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_{j(i)})^2}{p \cdot MSE}$$

where y_j is the observed response value,$\hat{y}_{j(i)}$is the predicted value when the i-th observation is omitted, p is the number of predictors in the model, and MSE is the mean squared error.

Determining High Leverage Points: To identify high leverage points, Cook's distance is compared to a threshold value. The threshold is often chosen as $\frac{4}{n-p-1}$, where n is the sample size and p is the number of predictors. If Cook's distance for an observation exceeds this threshold, it is considered a high leverage point.

```r
# Calculate the studized residuals and Cook's distance
studres <- rstudent(lm.fit_transformed)
cooksd <- cooks.distance(lm.fit_transformed)

# Combine student residuals and Cook's distance
influential_points <- which(cooksd > 4 / (nrow(df_train) - length(lm.fit_transformed$model) - 1) & abs(stu

print(influential_points)
```

```
## 10 94
## 10 94
```

We were surprised to find that there was no sample with index 39 in the prompt. Let's examine what's different about this sample point

```r
df_train[df_train$df..x..twelve == 1,c(predictors, "price")]
```

```
##    enginesize boreratio horsepower carwidth df..x..rwd df..x..ohcf df..x..ohcv
## 39   7.883461  13.00336   6.225108 33.73579          1           0           1
##    df..x..rotor df..x..twelve df..x..Medium df..x..Highend    price
## 39            0             1             0              0 4.374657
```

It turns out that this sample is the only df.. x.. twelve is a sample of one, which is why it has a very, very high leverage. In the case of limited data, we keep it.

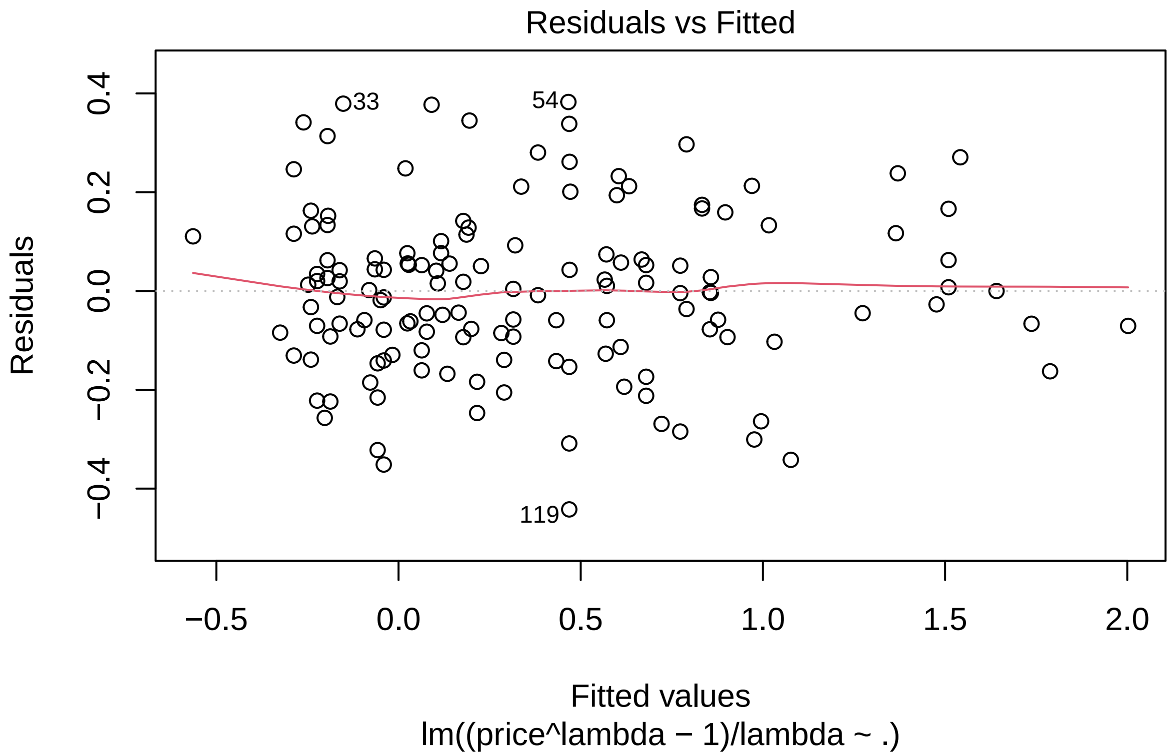
The Cook distance was only two points. Since we have no prior knowledge, we delete them again to fit the regression:

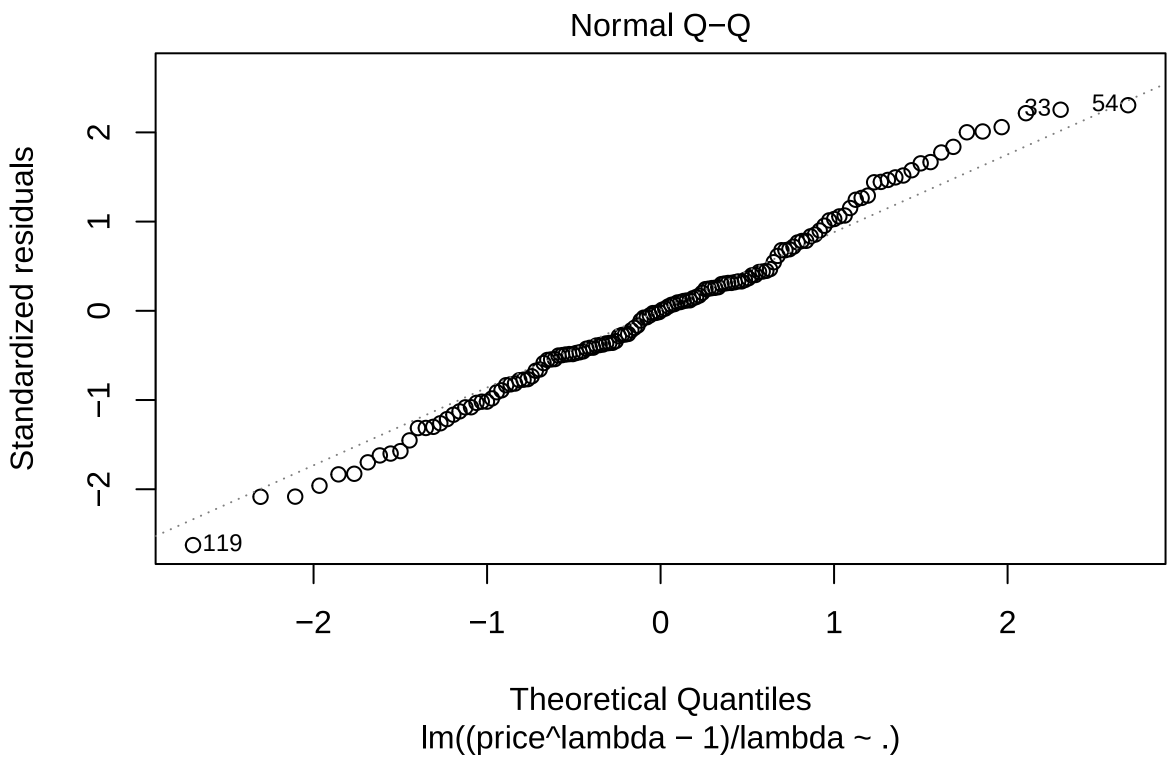
```r
# Fit a new LM model using the transformed response variable and predictors
lm.fit_transformed <- lm((price ^ lambda - 1) / lambda ~ ., data = df_train[-c(influential_points), c(pred

# Print the final regression results
summary(lm.fit_transformed)
```

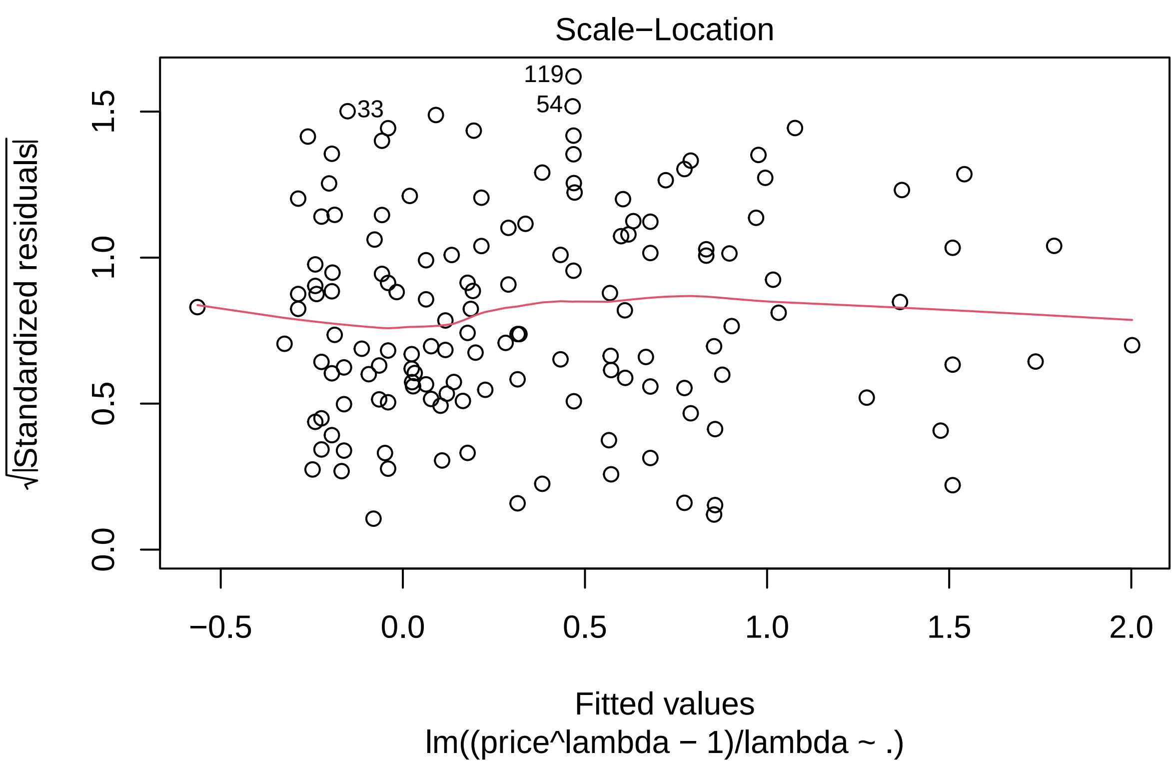
```
##
## Call:
## lm(formula = (price^lambda - 1)/lambda ~ ., data = df_train[-c(influential_points),
##     c(predictors, "price")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44212 -0.09343  0.00000  0.09671  0.38266
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -4.54010    0.76392  -5.943 2.38e-08 ***
## enginesize      0.29379    0.04170   7.045 9.48e-11 ***
## boreratio      -0.07327    0.02678  -2.736 0.007089 **
## horsepower      0.18658    0.03163   5.899 2.94e-08 ***
## carwidth        0.13639    0.02641   5.165 8.74e-07 ***
## df..x..rwd      0.15597    0.04600   3.391 0.000921 ***
## df..x..ohcf     0.18481    0.06895   2.680 0.008303 **
## df..x..ohcv    -0.45301    0.08561  -5.292 4.95e-07 ***
## df..x..rotor    0.30311    0.12770   2.374 0.019068 *
```

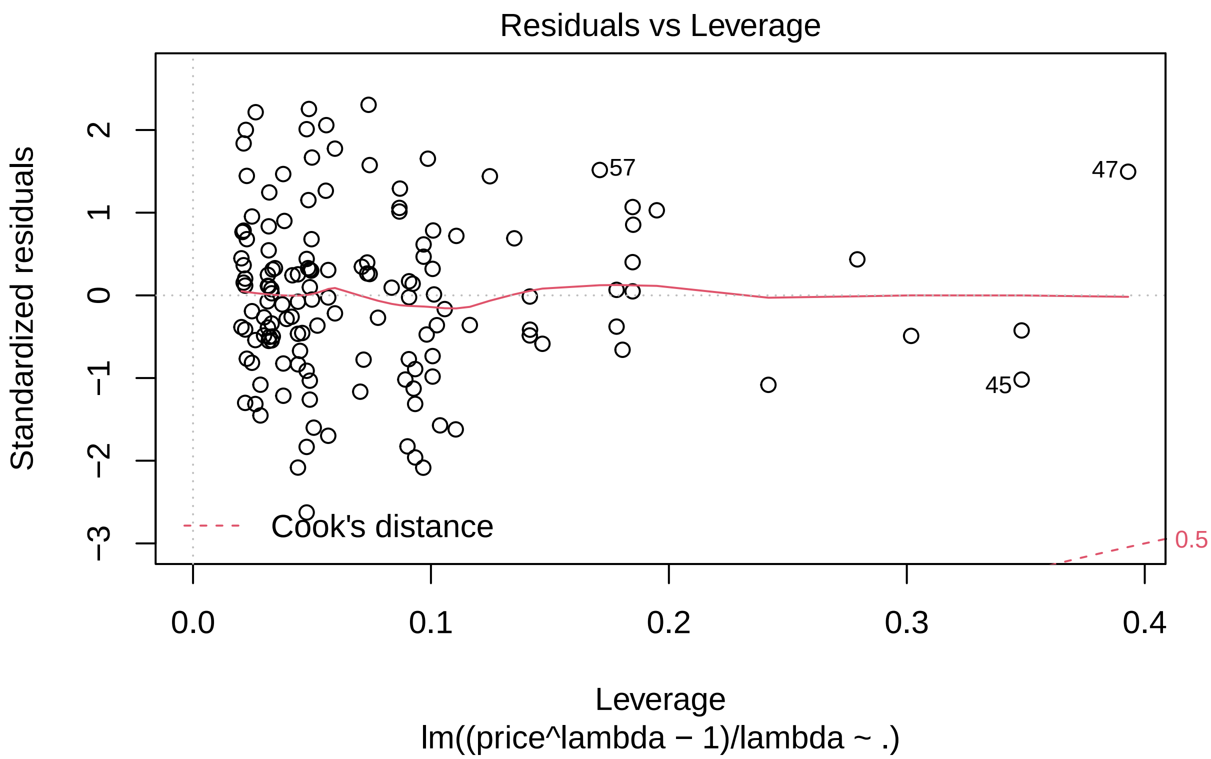
```
## df..x..twelve  -0.64783    0.20358  -3.182 0.001826 **
## df..x..Medium   0.19421    0.03816   5.090 1.22e-06 ***
## df..x..Highend  0.18312    0.05312   3.448 0.000761 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1725 on 131 degrees of freedom
## Multiple R-squared:  0.9101, Adjusted R-squared:  0.9025
## F-statistic: 120.5 on 11 and 131 DF,  p-value: < 2.2e-16
```

```
plot(lm.fit_transformed)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm((price^lambda − 1)/lambda ~ .)
119
33
54


Scale−Location
√|Standardized residuals|
Fitted values
lm((price^lambda − 1)/lambda ~ .)
119
54
33



It looks like the results are all very consistent with the regression hypothesis. We did well.

t-Test and F-test

Look at the final model:

```
summary(lm.fit_transformed)
```

```
##
## Call:
## lm(formula = (price^lambda - 1)/lambda ~ ., data = df_train[-c(influential_points),
##     c(predictors, "price")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44212 -0.09343  0.00000  0.09671  0.38266
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -4.54010    0.76392  -5.943 2.38e-08 ***
## enginesize       0.29379    0.04170   7.045 9.48e-11 ***
## boreratio       -0.07327    0.02678  -2.736 0.007089 **
## horsepower       0.18658    0.03163   5.899 2.94e-08 ***
## carwidth         0.13639    0.02641   5.165 8.74e-07 ***
## df..x..rwd       0.15597    0.04600   3.391 0.000921 ***
```

```
## df..x..ohcf     0.18481    0.06895   2.680 0.008303 **
## df..x..ohcv    -0.45301    0.08561  -5.292 4.95e-07 ***
## df..x..rotor    0.30311    0.12770   2.374 0.019068 *
## df..x..twelve  -0.64783    0.20358  -3.182 0.001826 **
## df..x..Medium   0.19421    0.03816   5.090 1.22e-06 ***
## df..x..Highend  0.18312    0.05312   3.448 0.000761 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1725 on 131 degrees of freedom
## Multiple R-squared:  0.9101, Adjusted R-squared:  0.9025
## F-statistic: 120.5 on 11 and 131 DF,  p-value: < 2.2e-16
```

The coefficients in the analysis indicate the estimated effect of each predictor variable on the dependent variable. The estimated coefficients, their standard errors, t-values, and corresponding p-values are provided in the table. The intercept term, represented as (Intercept), is estimated to be -4.54010 with a standard error of 0.76392. The negative sign suggests that when all predictor variables are zero, there is a negative offset in the dependent variable.

Among the predictor variables, enginesize, boreratio, horsepower, carwidth, df.. x.. rwd, df.. x.. ohcf, df.. x.. ohcv, df.. x.. rotor, df.. x.. twelve, df.. x.. Medium, and df.. x.. Highend are found to have significant effects on the dependent variable.

Enginesize has a positive coefficient estimate of 0.29379 (std. error = 0.04170), indicating that an increase in enginesize leads to a higher value of the dependent variable.

Boreratio, on the other hand, has a negative coefficient estimate of -0.07327 (std. error = 0.02678). This suggests that as boreratio increases, the value of the dependent variable decreases.

Horsepower has a positive coefficient estimate of 0.18658 (std. error = 0.03163), implying that an increase in horsepower is associated with a higher value of the dependent variable.

Carwidth also has a positive coefficient estimate of 0.13639 (std. error = 0.02641), indicating that wider cars tend to have higher values of the dependent variable.

The dummy variables, df.. x.. rwd, df.. x.. ohcf, df.. x.. ohcv, df.. x.. rotor, df.. x.. twelve, df.. x.. Medium, and df.. x.. Highend, represent different categories or levels within their respective variables. Each of these variables has a coefficient estimate indicating the difference in the dependent variable compared to the reference category. The positive coefficients (df.. x.. Medium, df.. x.. Highend) suggest that being in these categories is associated with higher values of the dependent variable, while the negative coefficients (df.. x.. ohcv, df.. x.. twelve) indicate lower values compared to the reference category.

The analysis also provides information on the significance of the coefficients. The significance is determined based on the t-values and their corresponding p-values. Significance is denoted using asterisks, with more asterisks indicating higher significance. For instance, ”” represents high significance (p-value < 0.001), ”” represents moderate significance (p-value < 0.01), and ”” represents lower significance (p-value < 0.05). In this analysis, several predictors have high significance, including enginesize, boreratio, horsepower, carwidth, df.. x.. rwd, df.. x.. ohcf, df.. x.. ohcv, and df.. x.. twelve.

The regression model's performance is assessed using multiple measures. The residual standard error (0.1725) indicates the average magnitude of the residuals, which represents the unexplained variation in the dependent variable after accounting for the predictor variables. A smaller residual standard error suggests a better fit of the model to the data.

The multiple R-squared value is 0.9101, indicating that approximately 91.01% of the variation in the dependent variable can be explained by the predictor variables included in the model. This suggests a strong overall relationship between the predictors and the dependent variable.

The adjusted R-squared value is 0.9025, which takes into account the number of predictors and adjusts the R-squared value accordingly. It penalizes the inclusion of unnecessary predictors and provides a more conservative estimate of the model's explanatory power. The adjusted R-squared is slightly lower than the multiple R-squared, suggesting that the included predictors are relevant.

The F-statistic (120.5) with its associated p-value (< 2.2e-16) tests the overall significance of the model. In this case, the extremely low p-value suggests that the overall model is statistically significant, meaning that at least one of the predictor variables has a significant effect on the dependent variable.

In summary, the regression analysis indicates that the selected predictor variables collectively have a strong relationship with the dependent variable. Enginesize, horsepower, carwidth, and the included dummy variables (df.. x.. rwd, df.. x.. ohcf, df.. x.. ohcv, df.. x.. rotor, df.. x.. twelve, df.. x.. Medium, df.. x.. Highend) all have significant effects on the dependent variable.

Performance on Test

```r
df_test[,num_vars]<- predict(preProc, df_test[,num_vars])
df_test_transformed <- df_test
df_test_transformed$price <- (df_test_transformed$price ^ lambda - 1) / lambda

predicted <- predict(lm.fit_transformed, newdata = df_test_transformed)


rmse <- sqrt(mean((predicted - df_test_transformed$price)^2))

y_mean <- mean(df_test_transformed$price)
ss_total <- sum((df_test_transformed$price - y_mean)^2)
ss_residual <- sum((df_test_transformed$price - predicted)^2)
r_squared <- 1 - (ss_residual / ss_total)

cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.178019
```

```r
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.8763747
```

The test set evaluation provides additional insights into the performance of the regression model. The root mean squared error (RMSE) is a measure of the average magnitude of the residuals in the test set. In this case, the RMSE is 0.178019, indicating that, on average, the predicted values deviate from the actual values by approximately 0.178019 units of the dependent variable. A smaller RMSE suggests better predictive accuracy.

The R-squared value for the test set is 0.8763747. This metric represents the proportion of the variation in the dependent variable that can be explained by the predictor variables in the model. An R-squared of 0.8763747 indicates that approximately 87.64% of the variation in the dependent variable is accounted for by the predictor variables in the test set.

Overall, the model demonstrates a good level of performance on the test set, as evidenced by the relatively small RMSE and high R-squared value. This shows that our model not only has high fitting accuracy, but also has good generalization ability.

Conclusion

In this analysis, a comprehensive regression modeling process was followed, including extensive data exploration, visualization, feature selection using various methods such as best subset selection, forward and backward stepwise selection, and Cross-Validation. Model diagnostics were performed to assess linearity, residual normality, homoscedasticity, and identify influential points such as high leverage points. Additionally, data preprocessing techniques such as Box-Cox transform and the Cook distance test were employed to address nonlinearity and evaluate influential observations, respectively. The final regression model was analyzed, and its performance was evaluated using test set metrics.

The regression analysis reveals a strong relationship between the selected predictor variables and the dependent variable. Enginesize, horsepower, carwidth, and several categorical variables (df.. x.. rwd, df.. x.. ohcf, df.. x.. ohcv, df.. x.. rotor, df.. x.. twelve, df.. x.. Medium, df.. x.. Highend) significantly influence the value of the dependent variable, indicating their importance in predicting the outcome.

The model's high multiple R-squared value of 0.9101 suggests that approximately 91.01% of the variation in the dependent variable can be explained by the included predictors. This indicates a good fit of the model to the data and highlights the relevance of the chosen variables. The adjusted R-squared value of 0.9025 provides a conservative estimate of the model's explanatory power, considering the number of predictors.

The test set performance, as indicated by the RMSE and R-squared, demonstrated the model's ability to accurately predict the dependent variable in unseen data.

However, further investigation is necessary to validate the model's robustness and generalize the findings. Several directions for future research can be considered:

1. High Leverage Points: It is important to examine the reasons for high leverage points, which are observations that have a significant impact on the regression results due to their extreme values or unique characteristics. Investigating the potential outliers and influential observations can help identify data quality issues, anomalies, or other factors that might influence the model's performance.

2. Additional Predictor Variables: Exploring the inclusion of additional relevant predictor variables that were not considered in the current analysis might enhance the model's explanatory power. It is important to conduct thorough research to identify and incorporate other variables that could have a meaningful impact on the dependent variable.

3. External Validation: Validating the regression model on an independent dataset can assess its predictive performance and generalizability. Collecting new data and applying the model to unseen observations can provide insights into its reliability and applicability in real-world scenarios.

4. Robustness Analysis: It is crucial to evaluate the model's robustness by testing it on different subsets of the data or using alternative modeling techniques. This can help determine if the selected variables and model structure consistently yield reliable results. Robustness analysis provides insights into the model's stability and can enhance its credibility.

In summary, the regression analysis demonstrates the significance of the selected predictor variables in explaining the variation in the dependent variable. However, further investigation into high leverage points, considering additional predictor variables, and conducting external validation is necessary to strengthen the findings and enhance the model's reliability and applicability.

References

- Francoeur, R. B. (2013). Could Sequential Residual Centering Resolve Low Sensitivity in Moderated Regression? Simulations and Cancer Symptom Clusters. Open Journal of Statistics, 03(06), 24-44.

- James, G., Witten, D., Hastie, T., and Tibshirani, R. (eds.). (2013). An introduction to statistical learning: with applications in R. New York: Springer.

- Marcoulides, K. M., and Raykov, T. (2019). Evaluation of Variance Inflation Factors in Regression Models Using Latent Variable Modeling Methods. Educational and Psychological Measurement, 79(5), 874–882.
- McElreath, R. (2020). Statistical rethinking: A Bayesian course with examples in R and Stan. 2nd edition. Chapman and Hall/CRC.
- Zuur AF, Ieno EN, Elphick CS. A protocol for data exploration to avoid common statistical problems: Data exploration. Methods in Ecology and Evolution (2010) 1:3–14.