# Report3

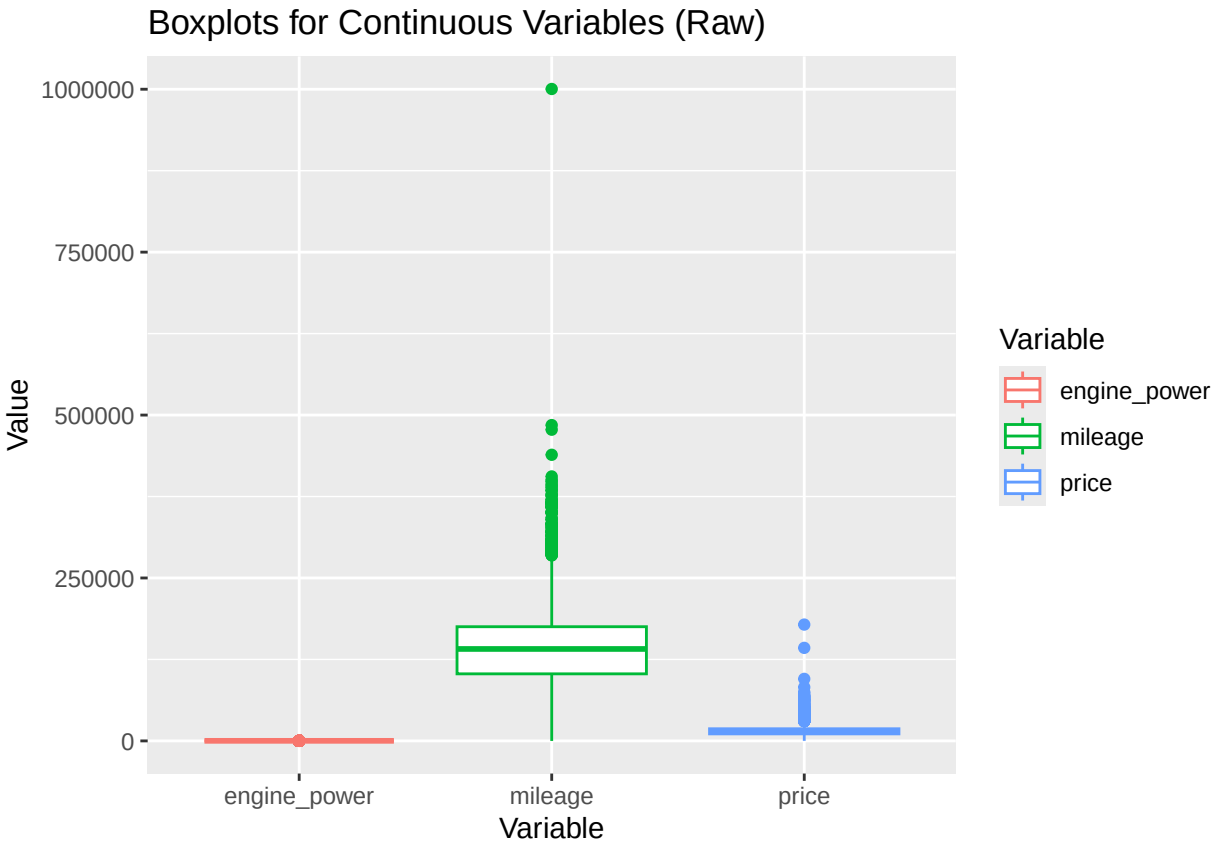
Lab2 Group A: Lee, Joshua; Liu, Kaiyi; Pulsone, Nathaniel; Wang, Mengyao; Xu, Zexian; Yang, Xiaojing

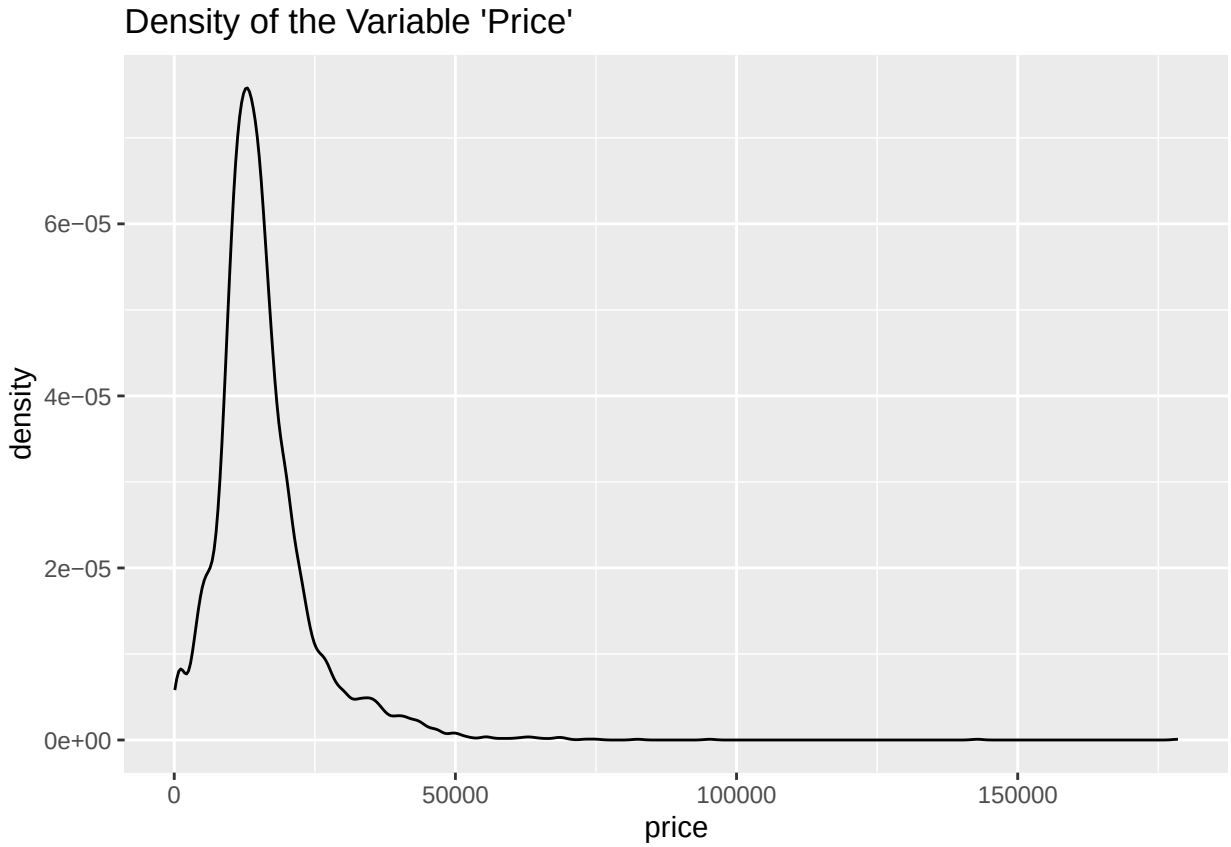
### Loading the Library and DataSet

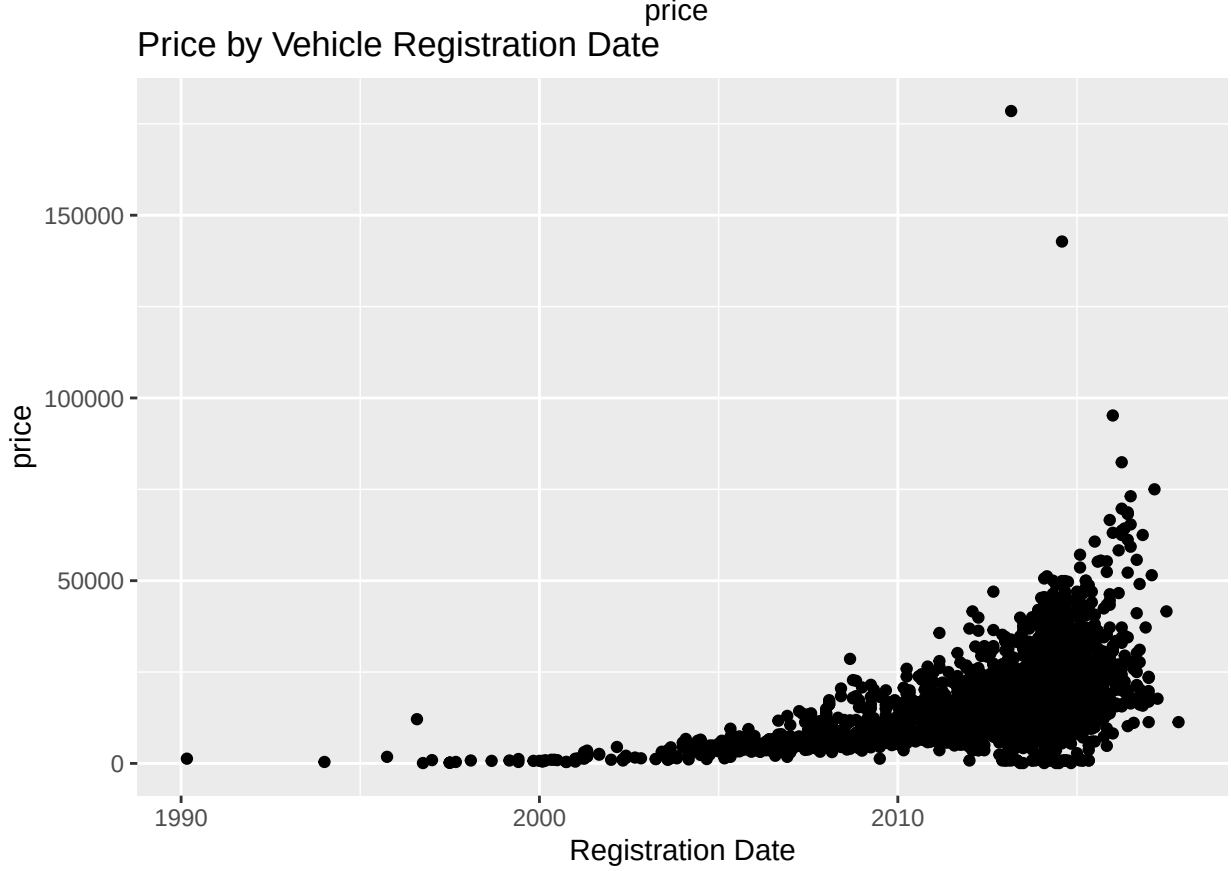
```
mylibrary <- c("tidyverse", "cowplot", "GGally", "MASS", "ggplot2", "glmnet", "data.table", "car")
invisible(lapply(mylibrary, library, character.only = TRUE))
### load the data
dat_bmw <- read.csv("BMWpricing_updated.csv")
```

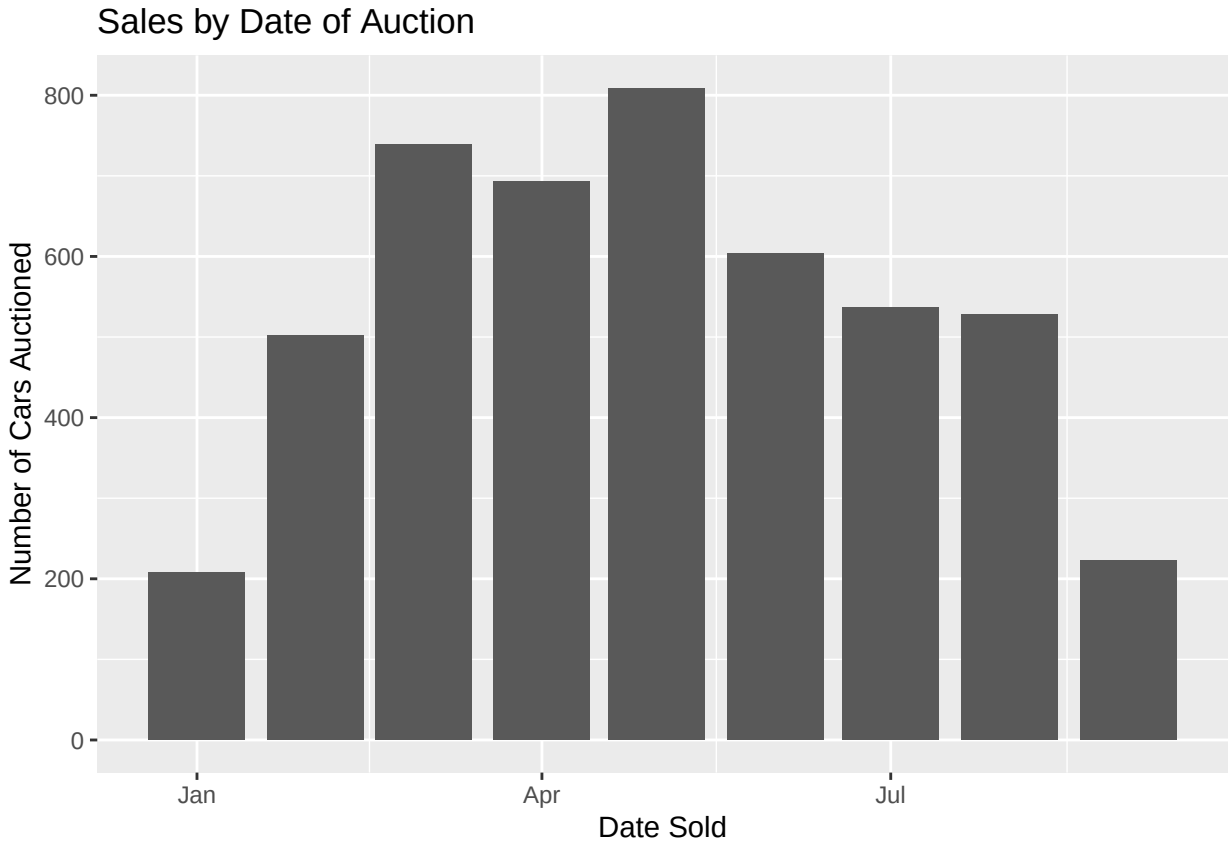
### Data Overview

Sourced from Kaggle, the dataset provides information on roughly 5000 used BMW cars sold in a business-to-business auction. Notable variables included are the price, car model, color, mileage, engine power, and various categorical descriptors.




Density of the Variable 'Price'
density
6e−05
4e−05
2e−05
0e+00
0
50000
100000
150000
price


Price by Vehicle Registration Date
price
150000
100000
50000
0
1990
2000
2010
Registration Date




From the graphs above, along with analysis that can be found in the Appendix section, we can see that the cars from the auction were all registered between March 1990 and November 2017, and the auction took place from January to September 2018. The cars spanned 75 different BMW models, 10 different colors, and 4 different fuel types.

The pricing of the cars is most concentrated around 15000, with a median price of 14200. The distribution of price is skewed heavily to the right. Because price is our main response variable, we will see that this causes the residuals of the constructed models to be right-skewed as well. The skewness of the variables can be somewhat remedied with a log-transformation on price.

Although it is hard to identify the feature variables in the dataset, we can see That most of the vehicles posses features 2 and 7, while most do not have features 3, 4 and 6. Each of the features when present tend to increase the price by `$5000` to `$10000`, except for feature 7, which actually decreases the price by about `$185`

Lastly, there are a few unusual values and outliers to consider in the dataset. There are two cars in the dataset that were sold for more than `$100,000` which is unusual enough to provide a high leverage and skew our constructed models. For this reason, we will remove that observation from the dataset. In addition, there is a single car with over 1 million miles on it, which we speculate was an error in data entry, so we will also remove this from the dataset. Finally, there are a number of observations with a negative mileage, or an engine power of zero. These values are also either errors in data entry, or indicative of a special case, such as scrapped or salvaged car. For this reason, we will not remove the observations, but instead set the negative and zero values to NA.

```
dat_bmw_clean <- dat_bmw|>
  filter(mileage < 500000 & mileage > 0)|>
  filter(price < 100000)|>
  filter(engine_power != 0)

head(dat_bmw_clean)
```

```
##   maker_key model_key mileage engine_power registration_date   fuel paint_color
## 1       BMW       118  140411          100          2/1/2012 diesel       black
## 2       BMW        M4   13929          317          4/1/2016 petrol        grey
## 3       BMW       320  183297          120          4/1/2012 diesel       white
## 4       BMW       420  128035          135          7/1/2014 diesel         red
## 5       BMW       425   97097          160         12/1/2014 diesel      silver
## 6       BMW       335  152352          225          5/1/2011 petrol       black
##      car_type feature_1 feature_2 feature_3 feature_4 feature_5 feature_6
## 1 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
## 2 convertible      TRUE      TRUE     FALSE     FALSE     FALSE      TRUE
## 3 convertible     FALSE     FALSE     FALSE     FALSE      TRUE     FALSE
## 4 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
## 5 convertible      TRUE      TRUE     FALSE     FALSE     FALSE      TRUE
## 6 convertible      TRUE      TRUE     FALSE     FALSE      TRUE      TRUE
##   feature_7 feature_8 price  sold_at obs_type
## 1      TRUE     FALSE 11300 1/1/2018 Training
## 2      TRUE      TRUE 69700 2/1/2018 Training
## 3      TRUE     FALSE 10200 2/1/2018 Training
## 4      TRUE      TRUE 25100 2/1/2018 Training
## 5      TRUE      TRUE 33400 4/1/2018 Training
## 6      TRUE      TRUE 17100 2/1/2018 Training
```

```
### Create Model series variable
dat_bmw_clean <- dat_bmw_clean %>%
  mutate(model_series = case_when(
    grepl("^1", model_key) ~ "1_Series",
    grepl("^2", model_key) ~ "2_Series",
    grepl("^3", model_key) ~ "3_Series",
    grepl("^4", model_key) ~ "4_Series",
    grepl("^5", model_key) ~ "5_Series",
    grepl("^7", model_key) ~ "7_Series",
    grepl("^M|M$", model_key) ~ "M_Power",
    model_key %in% c("X1") ~ "X1",
    model_key %in% c("X3") ~ "X3",
    model_key %in% c("X5") ~ "X5",
    model_key %in% c("X6") ~ "X6",
    TRUE ~ "Other"
  ))
```

### Quesion 1 revised

```
baseline_model <- lm(price ~ mileage + engine_power, data = dat_bmw_clean)
summary(baseline_model)
```

### Fit the baseline model

```
##
## Call:
## lm(formula = price ~ mileage + engine_power, data = dat_bmw_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36415  -2785    -73   2546  65891
##
```

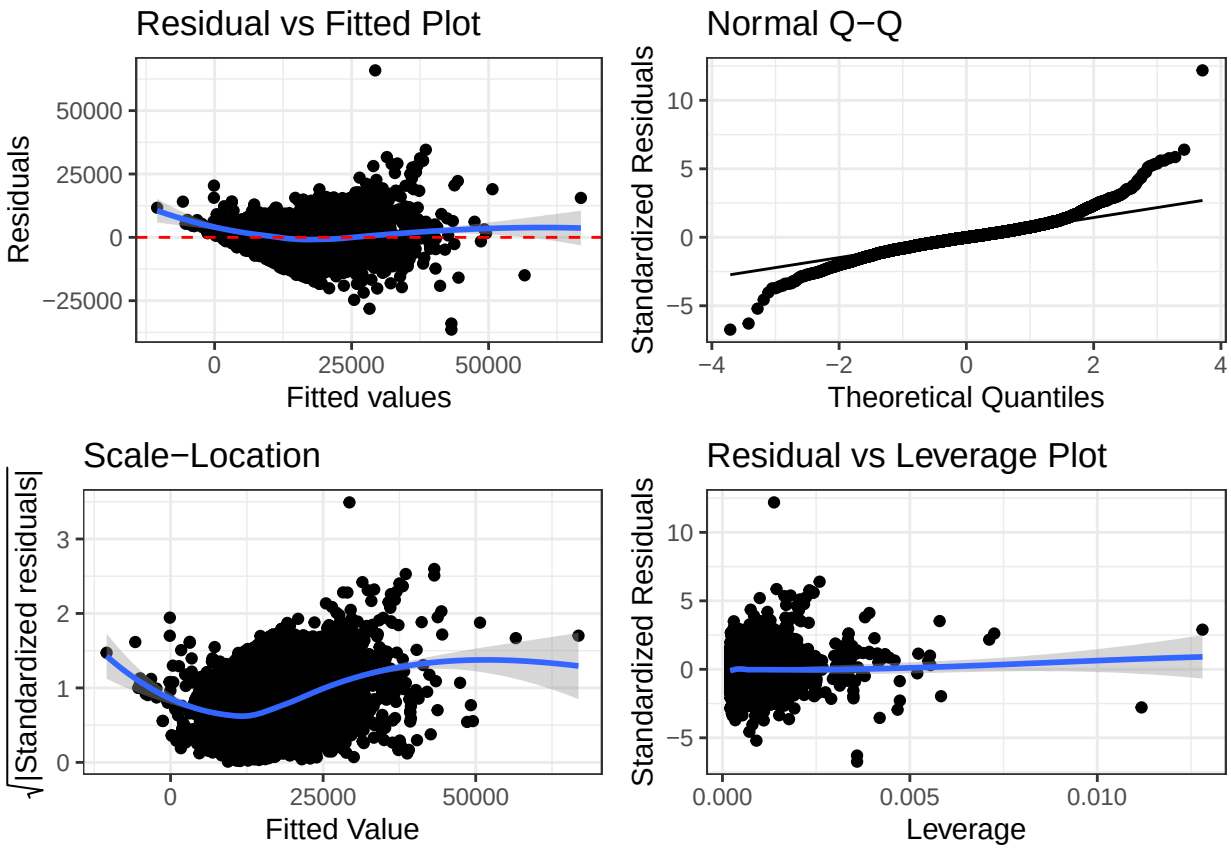
```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.193e+03  3.345e+02   15.53   <2e-16 ***
## mileage      -5.883e-02  1.322e-03  -44.49   <2e-16 ***
## engine_power  1.462e+02  2.000e+00   73.08   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5410 on 4835 degrees of freedom
## Multiple R-squared:  0.6127, Adjusted R-squared:  0.6125
## F-statistic:  3824 on 2 and 4835 DF,  p-value: < 2.2e-16
```

```
### Function to create diagnostic plots
diagPlot<-function(model){
    p1<-ggplot(model, aes(.fitted, .resid))+geom_point()
    p1<-p1+stat_smooth(method="loess")+geom_hline(yintercept=0, col="red", linetype="dashed")
    p1<-p1+xlab("Fitted values")+ylab("Residuals")
    p1<-p1+ggtitle("Residual vs Fitted Plot")+theme_bw()

    p2 <- ggplot(model, aes(sample = .stdresid)) +
      stat_qq() +
      stat_qq_line() +
      xlab("Theoretical Quantiles") +
      ylab("Standardized Residuals") +
      ggtitle("Normal Q-Q") +
      theme_bw()

    p3<-ggplot(model, aes(.fitted, sqrt(abs(.stdresid))))+geom_point(na.rm=TRUE)
    p3<-p3+stat_smooth(method="loess", na.rm = TRUE)+xlab("Fitted Value")
    p3<-p3+ylab(expression(sqrt("|Standardized residuals|")))
    p3<-p3+ggtitle("Scale-Location")+theme_bw()

    p5<-ggplot(model, aes(.hat, .stdresid))+geom_point(na.rm=TRUE)
    p5<-p5+stat_smooth(method="loess", na.rm=TRUE)
    p5<-p5+xlab("Leverage")+ylab("Standardized Residuals")
    p5<-p5+ggtitle("Residual vs Leverage Plot")
    p5<-p5+scale_size_continuous("Cook's Distance", range=c(1,5))
    p5<-p5+theme_bw()+theme(legend.position="bottom")

    #return(list(rvfPlot=p1, qqPlot=p2, sclLocPlot=p3, rvlevPlot=p5))
    plot_grid(p1, p2, p3, p5, align = "h")
}
```

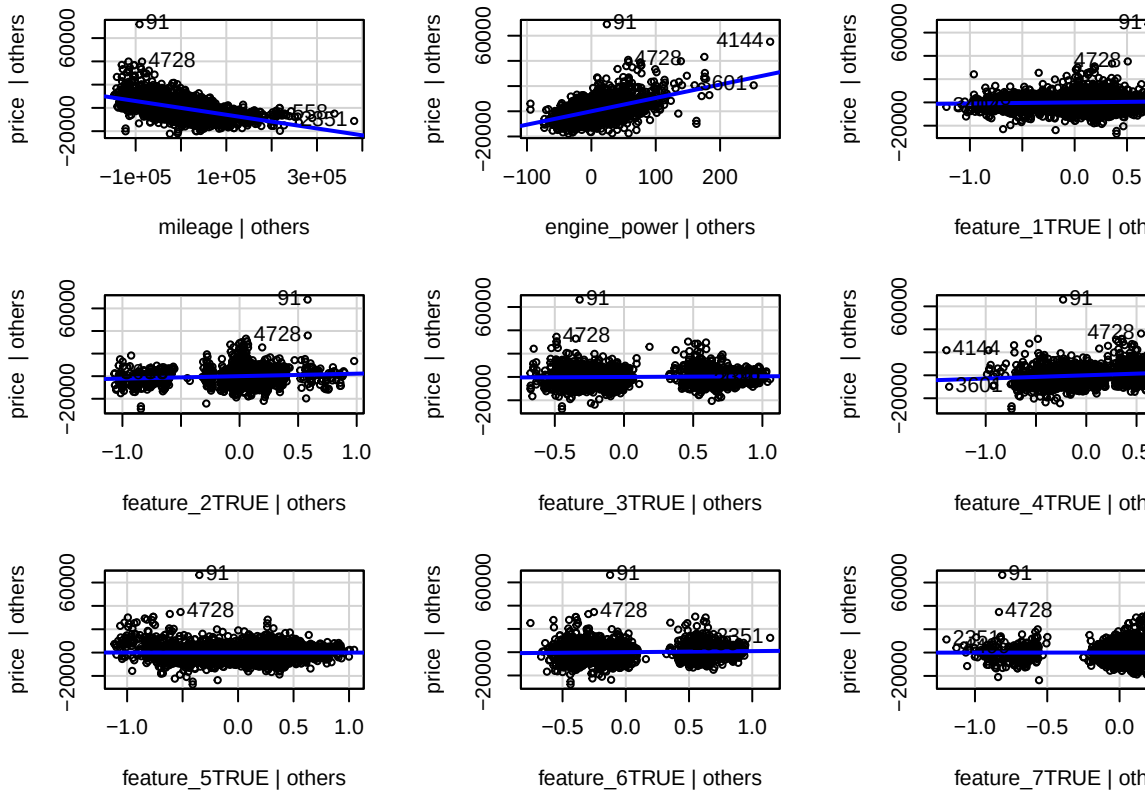
```
diagPlot(baseline_model)
```

### Diagonostic of the baseline model

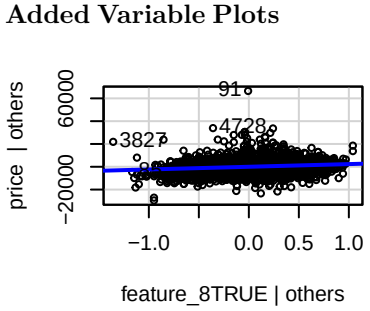
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```
feature_model <- lm(price ~ mileage + engine_power + feature_1 + feature_2 + feature_3 + feature_4 + fea
avPlots(feature_model)
```




**Added Variable Plots**

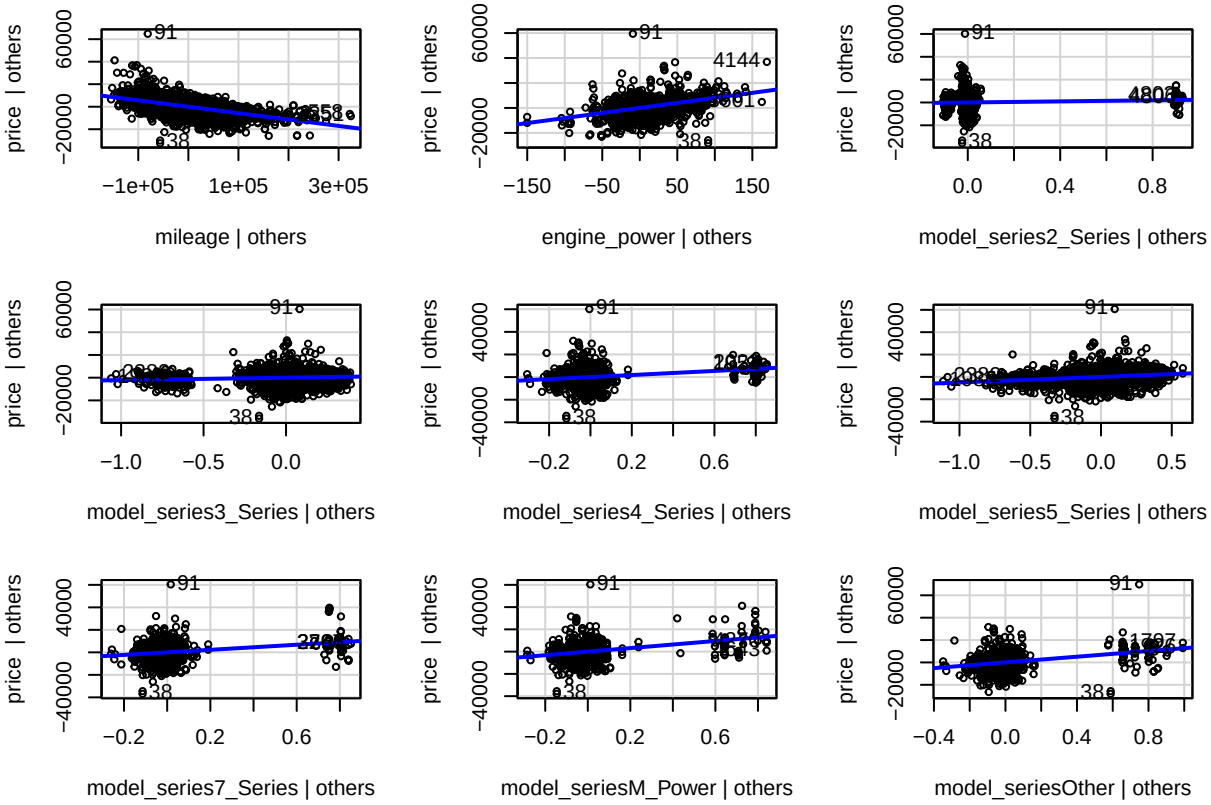


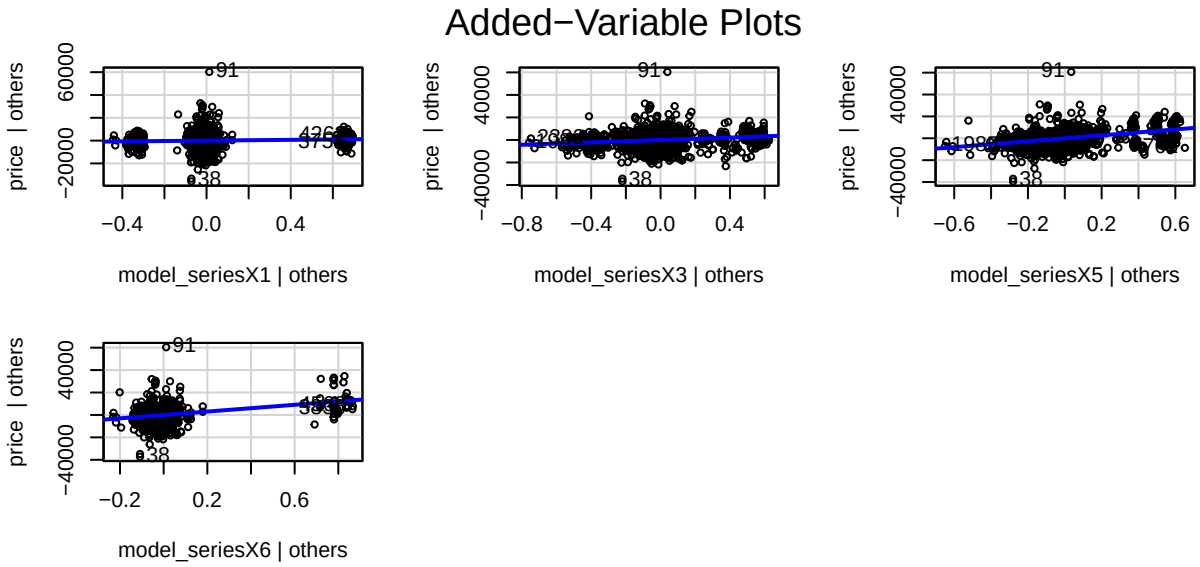

```
Full_model <- lm(price ~ mileage + engine_power + model_series, data = dat_bmw_clean)
summary(Full_model)
```

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series, data = dat_bmw_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35607  -2007    266   2432  60188
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.240e+03  3.521e+02  26.243  < 2e-16 ***
## mileage               -5.681e-02  1.225e-03 -46.380  < 2e-16 ***
## engine_power           8.080e+01  2.543e+00  31.770  < 2e-16 ***
## model_series2_Series   2.389e+03  6.975e+02   3.426 0.000618 ***
## model_series3_Series   2.169e+03  2.377e+02   9.126  < 2e-16 ***
## model_series4_Series   9.068e+03  5.249e+02  17.275  < 2e-16 ***
## model_series5_Series   4.992e+03  2.798e+02  17.841  < 2e-16 ***
```

```
## model_series7_Series  1.128e+04  7.279e+02  15.495  < 2e-16 ***
## model_seriesM_Power   1.596e+04  7.991e+02  19.967  < 2e-16 ***
## model_seriesOther     1.274e+04  5.736e+02  22.213  < 2e-16 ***
## model_seriesX1        1.742e+03  3.520e+02   4.949 7.73e-07 ***
## model_seriesX3        5.334e+03  3.201e+02  16.661  < 2e-16 ***
## model_seriesX5        1.349e+04  4.427e+02  30.482  < 2e-16 ***
## model_seriesX6        1.508e+04  8.018e+02  18.804  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4744 on 4824 degrees of freedom
## Multiple R-squared:  0.7028, Adjusted R-squared:  0.702
## F-statistic: 877.4 on 13 and 4824 DF,  p-value: < 2.2e-16
```

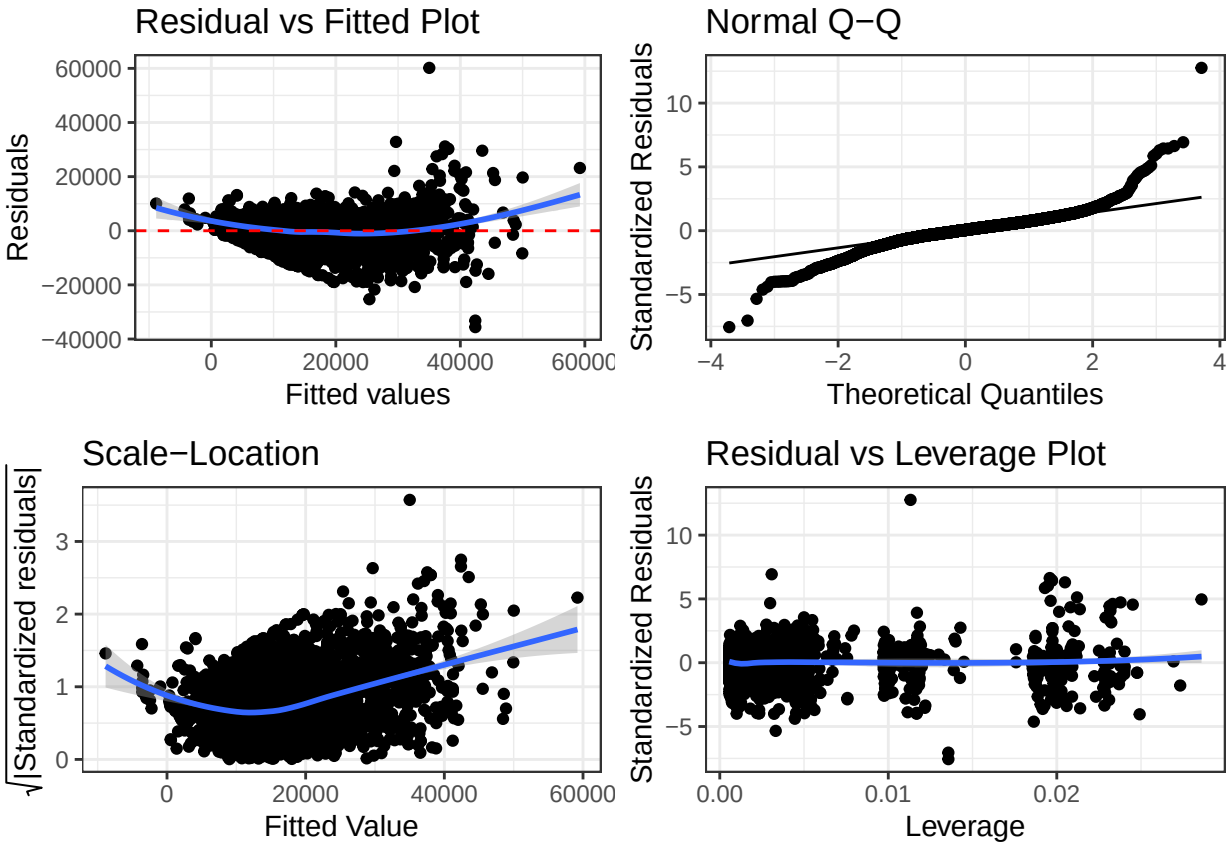
```
avPlots(Full_model)
```






```
diagPlot(Full_model)
```

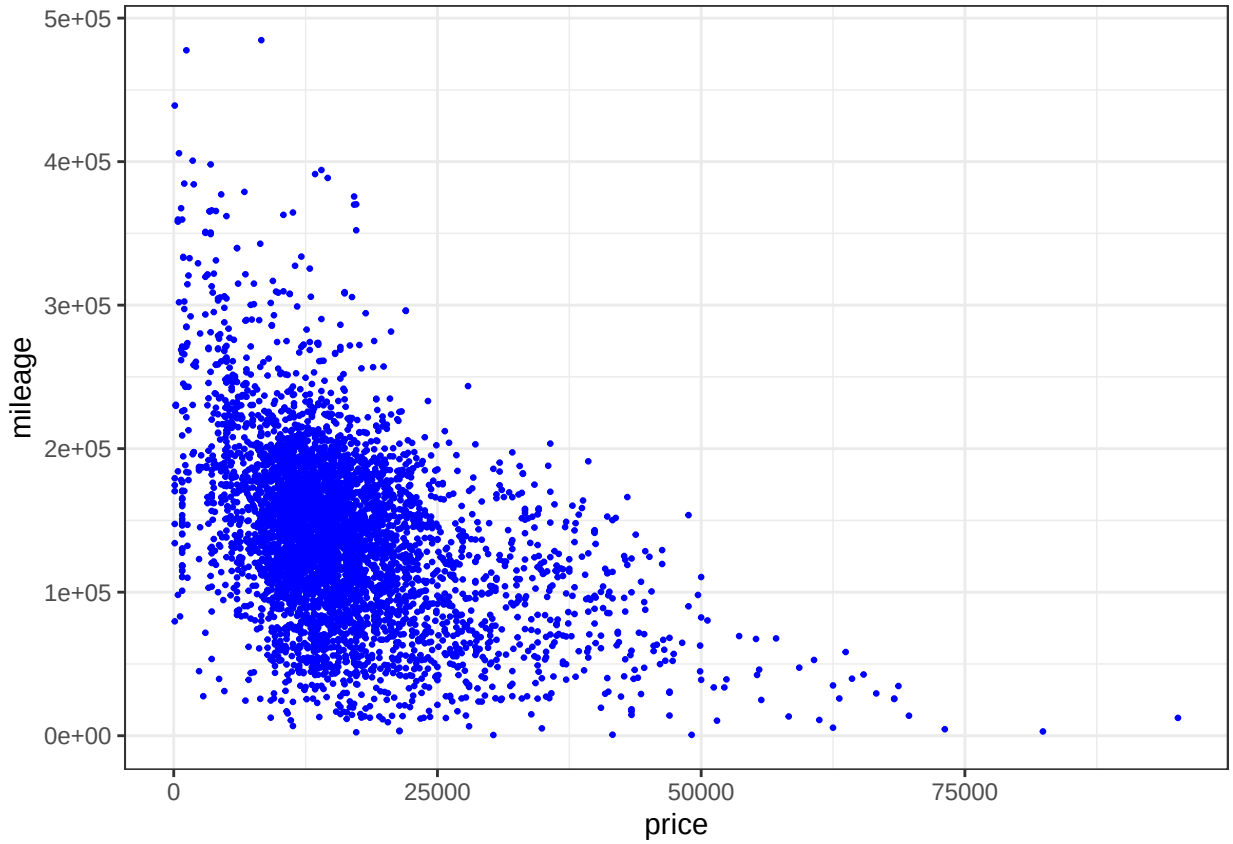
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



**model refinement**

Based on the curved line observed in the "Residual vs Fitted" Plot in the "Full model" and the quadratic relationship between price and mileage, we add mileage^2 to the model.
The fitted line in the "Residual vs Fitted" Plot is not as quadratic as it in the "Full_model". The deviated tail may be caused by potential outliers.

```
ggplot(data = dat_bmw_clean, mapping = aes(x = price, y = mileage)) +
  geom_point(color = "blue", size = 0.5) +
  theme_bw()
```




```
Full_model2 <- lm(price ~ mileage + engine_power + model_series + I(mileage^(1/2)), data = dat_bmw_clean
summary(Full_model2)
```

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series +
##     I(mileage^(1/2)), data = dat_bmw_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37193  -2065    357   2464  57586
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.550e+04  8.191e+02  18.918  < 2e-16 ***
## mileage              -7.259e-03  5.992e-03  -1.211   0.2258
## engine_power          8.105e+01  2.525e+00  32.098  < 2e-16 ***
## model_series2_Series  1.602e+03  6.987e+02   2.292   0.0219 *
```

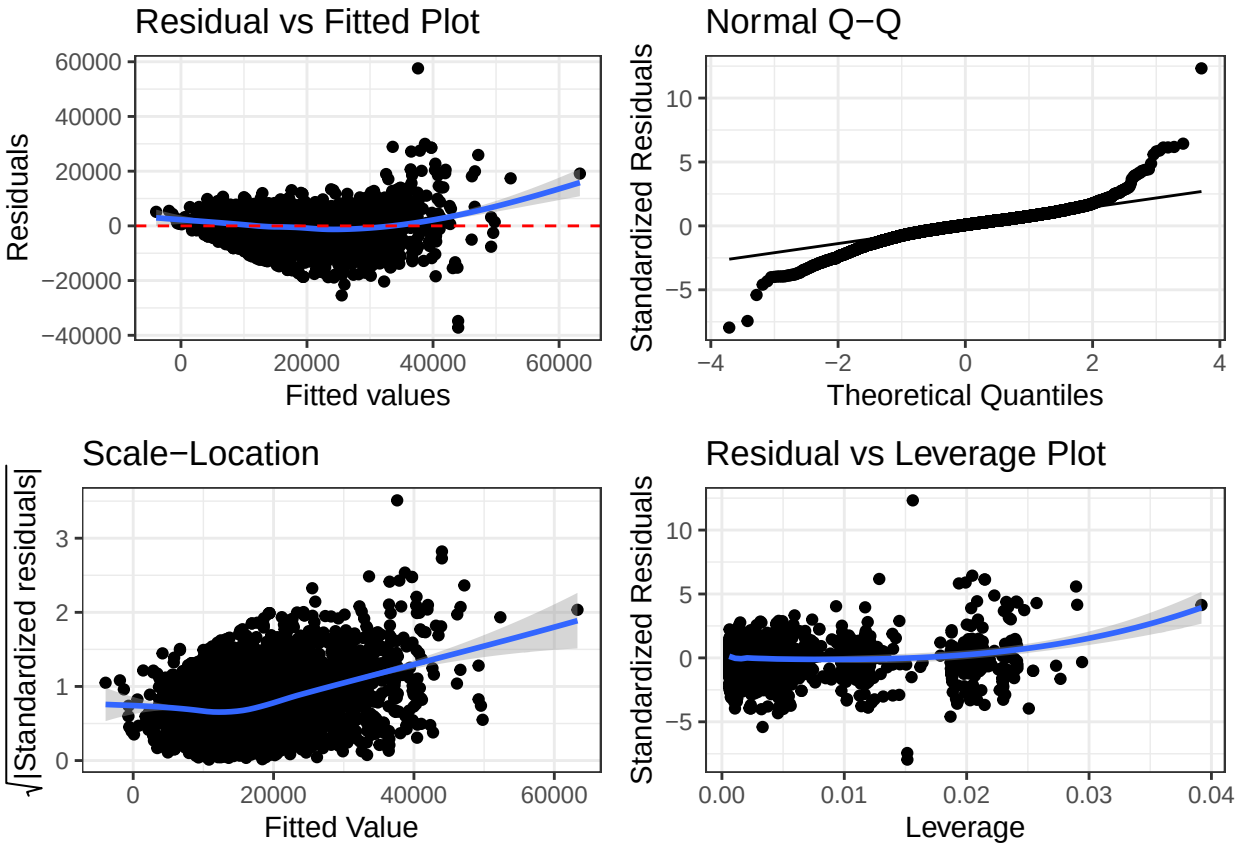
```
## model_series3_Series  2.305e+03  2.366e+02   9.746  < 2e-16 ***
## model_series4_Series  8.727e+03  5.227e+02  16.698  < 2e-16 ***
## model_series5_Series  5.062e+03  2.779e+02  18.213  < 2e-16 ***
## model_series7_Series  1.120e+04  7.227e+02  15.496  < 2e-16 ***
## model_seriesM_Power   1.552e+04  7.950e+02  19.528  < 2e-16 ***
## model_seriesOther     1.248e+04  5.702e+02  21.890  < 2e-16 ***
## model_seriesX1        1.894e+03  3.499e+02   5.411 6.57e-08 ***
## model_seriesX3        5.472e+03  3.182e+02  17.194  < 2e-16 ***
## model_seriesX5        1.342e+04  4.396e+02  30.528  < 2e-16 ***
## model_seriesX6        1.492e+04  7.962e+02  18.740  < 2e-16 ***
## I(mileage^(1/2))     -3.640e+01  4.310e+00  -8.446  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4710 on 4823 degrees of freedom
## Multiple R-squared:  0.7071, Adjusted R-squared:  0.7063
## F-statistic: 831.7 on 14 and 4823 DF,  p-value: < 2.2e-16
```

```
anova(Full_model, Full_model2)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ mileage + engine_power + model_series
## Model 2: price ~ mileage + engine_power + model_series + I(mileage^(1/2))
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1   4824 1.0857e+11
## 2   4823 1.0699e+11  1 1582302175 71.328 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
diagPlot(Full_model2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```
Full_model3 <- lm(price ~ mileage + engine_power + model_series + I(mileage^(1/2)) + mileage:engine_pow
summary(Full_model3)
```

**add interaction term**

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series +
##     I(mileage^(1/2)) + mileage:engine_power, data = dat_bmw_clean)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -43422  -1962    368   2378  56422
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            6.568e+03  9.513e+02   6.904 5.70e-12 ***
## mileage                4.731e-02  6.635e-03   7.129 1.16e-12 ***
## engine_power           1.440e+02  4.419e+00  32.575  < 2e-16 ***
## model_series2_Series   2.289e+03  6.797e+02   3.367 0.000765 ***
## model_series3_Series   1.885e+03  2.310e+02   8.159 4.26e-16 ***
## model_series4_Series   8.160e+03  5.086e+02  16.044  < 2e-16 ***
## model_series5_Series   4.917e+03  2.700e+02  18.213  < 2e-16 ***
## model_series7_Series   1.094e+04  7.020e+02  15.582  < 2e-16 ***
## model_seriesM_Power    1.366e+04  7.796e+02  17.526  < 2e-16 ***
## model_seriesOther      1.161e+04  5.561e+02  20.875  < 2e-16 ***
```

```
## model_seriesX1        1.750e+03  3.399e+02   5.149 2.72e-07 ***
## model_seriesX3        5.241e+03  3.093e+02  16.942  < 2e-16 ***
## model_seriesX5        1.302e+04  4.275e+02  30.465  < 2e-16 ***
## model_seriesX6        1.409e+04  7.747e+02  18.182  < 2e-16 ***
## I(mileage^(1/2))     -3.108e+01  4.197e+00  -7.405 1.54e-13 ***
## mileage:engine_power -4.723e-04  2.761e-05 -17.109  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4574 on 4822 degrees of freedom
## Multiple R-squared:  0.7239, Adjusted R-squared:  0.723
## F-statistic: 842.8 on 15 and 4822 DF,  p-value: < 2.2e-16
```

```
anova(Full_model2, Full_model3)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ mileage + engine_power + model_series + I(mileage^(1/2))
## Model 2: price ~ mileage + engine_power + model_series + I(mileage^(1/2)) +
##     mileage:engine_power
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1   4823 1.0699e+11
## 2   4822 1.0087e+11  1 6123355871 292.73 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(Full_model, Full_model3)
```

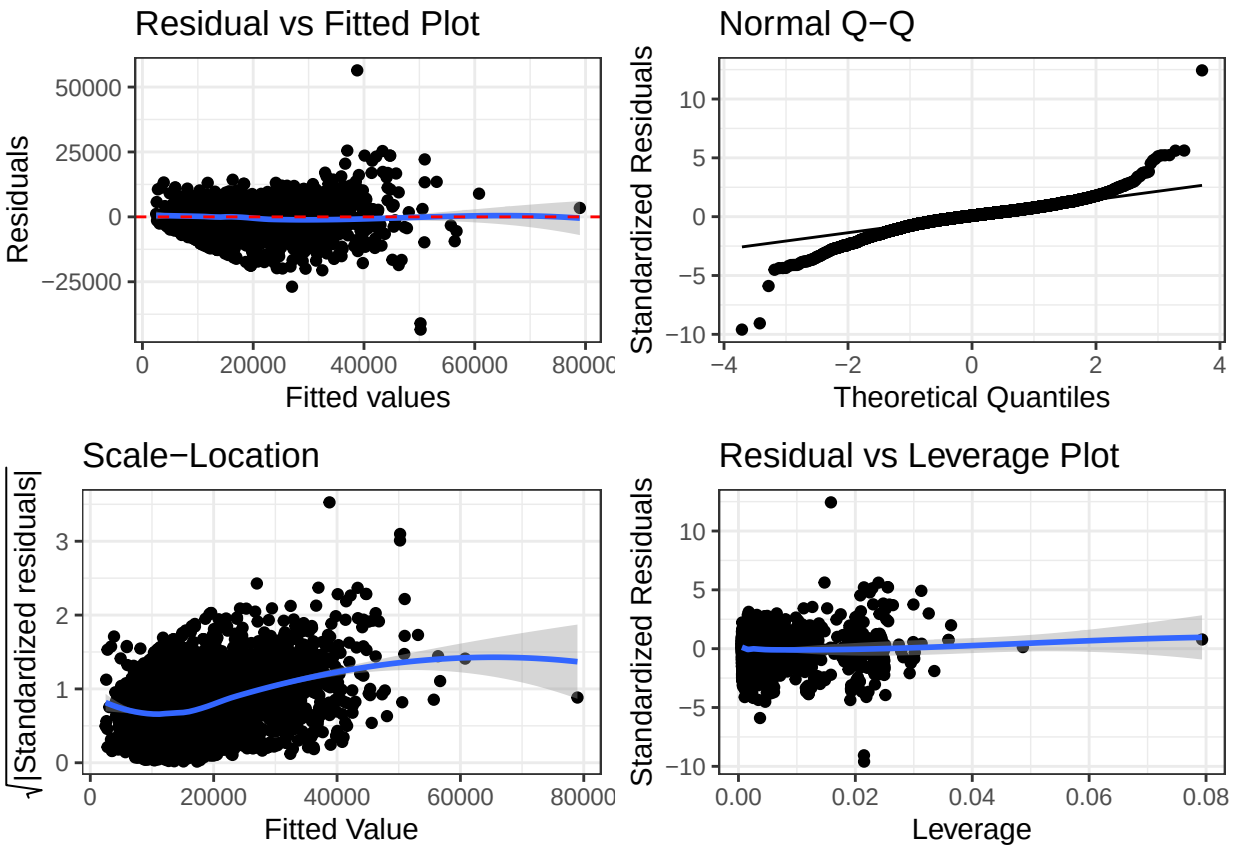
```
## Analysis of Variance Table
##
## Model 1: price ~ mileage + engine_power + model_series
## Model 2: price ~ mileage + engine_power + model_series + I(mileage^(1/2)) +
##     mileage:engine_power
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1   4824 1.0857e+11
## 2   4822 1.0087e+11  2 7705658046 184.19 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(baseline_model, Full_model3)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ mileage + engine_power
## Model 2: price ~ mileage + engine_power + model_series + I(mileage^(1/2)) +
##     mileage:engine_power
##   Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
## 1   4835 1.4149e+11
## 2   4822 1.0087e+11 13 4.0625e+10 149.39 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
diagPlot(Full_model3)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




**predicting performance**

```
set.seed(2025)
group <- split(sample(1:nrow(dat_bmw_clean)),rep(1:2, times=c(nrow(dat_bmw_clean)/2, nrow(dat_bmw_clean)

dat_train <- dat_bmw_clean[group$`1`, ]
dat_test <- dat_bmw_clean[group$`2`, ]

## fit Full_model4 to the training data ##
Full_model3_train <- lm(price ~ mileage + engine_power + model_series + I(mileage^(1/2)) + mileage:engi
summary(Full_model3_train)
```

**randomly divided samples into training and testing data**

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series +
##     I(mileage^(1/2)) + mileage:engine_power, data = dat_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41692  -1942    334   2408  24313
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.016e+03  1.401e+03   2.867 0.004182 **
## mileage               2.881e-02  9.209e-03   3.129 0.001778 **
```

```
## engine_power          1.469e+02  6.252e+00  23.502  < 2e-16 ***
## model_series2_Series  1.474e+03  1.132e+03   1.302 0.193002
## model_series3_Series  1.743e+03  3.307e+02   5.271 1.48e-07 ***
## model_series4_Series  8.639e+03  6.734e+02  12.829  < 2e-16 ***
## model_series5_Series  4.697e+03  3.887e+02  12.083  < 2e-16 ***
## model_series7_Series  1.201e+04  9.376e+02  12.805  < 2e-16 ***
## model_seriesM_Power   1.581e+04  1.289e+03  12.257  < 2e-16 ***
## model_seriesOther     9.999e+03  7.616e+02  13.129  < 2e-16 ***
## model_seriesX1        1.715e+03  4.803e+02   3.570 0.000364 ***
## model_seriesX3        4.770e+03  4.382e+02  10.885  < 2e-16 ***
## model_seriesX5        1.276e+04  6.042e+02  21.116  < 2e-16 ***
## model_seriesX6        1.375e+04  1.150e+03  11.956  < 2e-16 ***
## I(mileage^(1/2))     -1.632e+01  6.023e+00  -2.710 0.006781 **
## mileage:engine_power -5.067e-04  3.986e-05 -12.711  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4613 on 2403 degrees of freedom
## Multiple R-squared:  0.7241, Adjusted R-squared:  0.7223
## F-statistic: 420.4 on 15 and 2403 DF,  p-value: < 2.2e-16
```

```
#### predict new data using model_full_train and model 2 ####
dat_test$price_new <- predict(Full_model3_train, newdata = dat_test[,-17])

#### performance on predicting ####
dat_predict <- dat_test[,c("price", "mileage", "price_new")]
## R^2
fun_r2 <- function(obe, pred){
  rss <- sum((obe-pred)^2)
  sst <- sum((obe-mean(obe))^2)
  r2 <- 1 - rss/sst

  return(r2)
}
fun_r2(dat_predict$price, dat_predict$price_new)
```

**performance on predicting**

```
## [1] 0.7168153
```

```
## MSE
fun_mse <- function(obe, pred){
  mse <- (sum((obe-pred)^2))/length(pred)

  return(mse)
}
fun_mse(dat_predict$price, dat_predict$price_new)
```

```
## [1] 21054602
```

```
## MAE
fun_mae <- function(obe, pred){
  mse <- (sum(abs(obe-pred)))/length(pred)

  return(mse)
```

```
}
fun_mae(dat_predict$price, dat_predict$price_new)
```

```
## [1] 3164.623
```

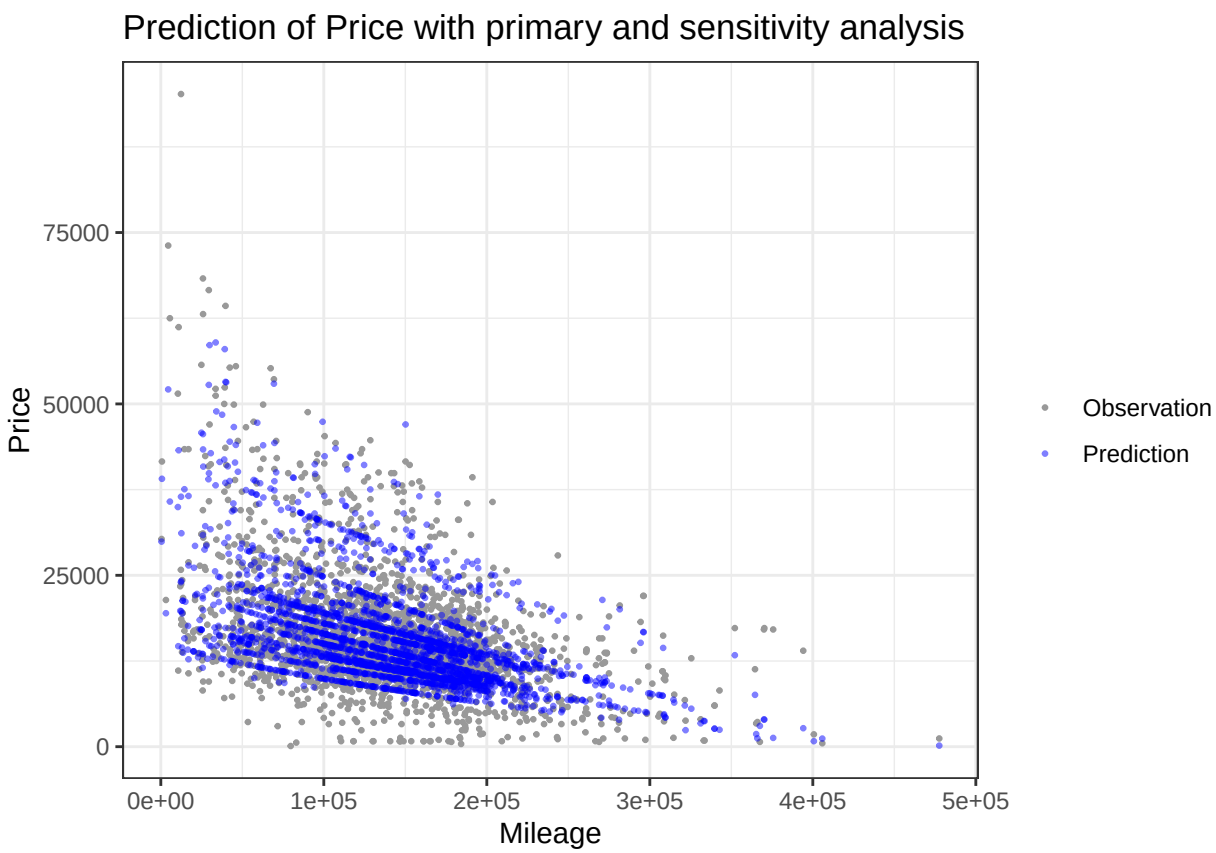
```
summary(dat_predict$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     100   11100   14200   15912   18800   95200
```

```
summary(dat_predict$price_new)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   161.1 10866.4 13835.9 15609.6 18031.3 58963.6
```

```
dat_predict$price <- as.numeric(dat_predict$price)
dat_predict$mileage <- as.numeric(dat_predict$mileage)

ggplot() +
  geom_point(data = dat_predict, aes(x = mileage, y = price, color = "Observation"), size = 0.5) +
  geom_point(data = dat_predict, aes(x = mileage, y = price_new, color = "Prediction"), alpha = 0.5, si
  labs(title = "Prediction of Price with primary and sensitivity analysis") +
  xlab("Mileage") +
  ylab("Price") +
  scale_color_manual(name = element_blank(), values = c("Observation" = "grey60", "Prediction" = "blue"
  theme_bw()
```




**visualization**

**Question 3**

```
dat_age <- dat_bmw_clean|>
  mutate(age = 2018 - year(as.Date(registration_date, format = "%m/%d/%Y")))|>
  mutate(is_116 = ifelse(model_key == "116", 1, 0))|>
  mutate(is_318 = ifelse(model_key == "318", 1, 0))|>
  mutate(is_X1 = ifelse(model_key == "X1", 1, 0))|>
  mutate(is_X3 = ifelse(model_key == "X3", 1, 0))

# determine models with sufficient data
model_candidates <- dat_bmw_clean|>
  group_by(model_key)|>
  summarize(n = n())|>
  filter(n > 100)
model_candidates
```

```
## # A tibble: 11 x 2
##    model_key     n
##    <chr>     <int>
##  1 116         358
##  2 118         142
##  3 316         235
##  4 318         569
##  5 320         752
##  6 520         633
##  7 525         184
##  8 530         157
##  9 X1          274
## 10 X3          437
## 11 X5          231
```

```
model_candidates$model_key
```

```
##  [1] "116" "118" "316" "318" "320" "520" "525" "530" "X1"  "X3"  "X5"
```

```
# narrow down number of models considered
coefs = c()
confintlbs = c()

for(i in seq(1:length(model_candidates$model_key))){
  temp.df <- dat_age|>
    filter(model_key == model_candidates$model_key[i])
  temp.lm <- lm(log(price) ~ age, data = temp.df)
  coefs = c(coefs, coef(temp.lm)[2])
  confintlbs = c(confintlbs, confint(temp.lm)[2])
}

rm(temp.df)
rm(temp.lm)

results = cbind(coefs, confintlbs, model_candidates$model_key)
results
```

```
##     coefs                 confintlbs
## age "-0.12278957659823"   "-0.145500785077517"  "116"
## age "-0.143433541715236"  "-0.162697652058148"  "118"
```

```
## age "-0.149038443543364"  "-0.171057626029924"  "316"
## age "-0.136161010233223"  "-0.149777693349068"  "318"
## age "-0.16563587460071"   "-0.178490627890781"  "320"
## age "-0.183404655126876"  "-0.191612478867974"  "520"
## age "-0.206231690635819"  "-0.224201035913407"  "525"
## age "-0.185604933819817"  "-0.198075674413492"  "530"
## age "-0.0182544527786139" "-0.0676029110385667" "X1"
## age "-0.11571932710388"   "-0.13260525273682"   "X3"
## age "-0.171213304919039"  "-0.177936709330697"  "X5"
```

```
age_model <- lm(formula = price ~ mileage + engine_power + model_series + I(mileage^(1/2)) + mileage:eng
summary(age_model)
```

```
##
## Call:
## lm(formula = price ~ mileage + engine_power + model_series +
##     I(mileage^(1/2)) + mileage:engine_power + age + age:is_318 +
##     age:is_X1 + age:is_X3 + age:is_116, data = dat_age)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35980  -1533    110   1807  53514
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.327e+04  8.165e+02  16.258  < 2e-16 ***
## mileage                  1.126e-01  5.654e-03  19.918  < 2e-16 ***
## engine_power             1.445e+02  3.645e+00  39.643  < 2e-16 ***
## model_series2_Series     4.833e+02  5.894e+02   0.820  0.41221
## model_series3_Series     1.496e+03  2.724e+02   5.492 4.19e-08 ***
## model_series4_Series     7.114e+03  4.545e+02  15.654  < 2e-16 ***
## model_series5_Series     4.903e+03  2.852e+02  17.192  < 2e-16 ***
## model_series7_Series     1.240e+04  6.010e+02  20.631  < 2e-16 ***
## model_seriesM_Power      1.415e+04  6.591e+02  21.473  < 2e-16 ***
## model_seriesOther        1.192e+04  4.873e+02  24.472  < 2e-16 ***
## model_seriesX1          -1.629e+03  1.106e+03  -1.473  0.14094
## model_seriesX3           5.742e+03  5.442e+02  10.552  < 2e-16 ***
## model_seriesX5           1.393e+04  3.895e+02  35.759  < 2e-16 ***
## model_seriesX6           1.539e+04  6.577e+02  23.396  < 2e-16 ***
## I(mileage^(1/2))        -5.259e+01  3.501e+00 -15.020  < 2e-16 ***
## age                     -1.239e+03  2.712e+01 -45.710  < 2e-16 ***
## mileage:engine_power    -5.414e-04  2.286e-05 -23.680  < 2e-16 ***
## age:is_318               7.269e+01  3.008e+01   2.417  0.01569 *
## age:is_X1                6.289e+02  2.028e+02   3.102  0.00193 **
## age:is_X3               -4.629e+01  8.148e+01  -0.568  0.56998
## age:is_116               1.145e+02  5.594e+01   2.047  0.04073 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3761 on 4817 degrees of freedom
## Multiple R-squared:  0.8134, Adjusted R-squared:  0.8127
## F-statistic:  1050 on 20 and 4817 DF,  p-value: < 2.2e-16
```

```
vif(age_model)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##                             GVIF Df GVIF^(1/(2*Df))
## mileage                37.904179  1        6.156637
## engine_power            6.888626  1        2.624619
## model_series          814.487760 11        1.356166
## I(mileage^(1/2))       28.659254  1        5.353434
## age                     1.602662  1        1.265963
## mileage:engine_power   15.477916  1        3.934198
## age:is_318              1.333149  1        1.154621
## age:is_X1              21.331302  1        4.618582
## age:is_X3               6.746347  1        2.597373
## age:is_116              2.314989  1        1.521509
```

```
diagPlot(age_model)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```