HW10

110078509 & I discussed with 109065707

20220423

Question 1

a. Comparing scenarios 1 and 2, which do we expect to have a stronger R2 ?

- Ans: Scenarios 1

b. Comparing scenarios 3 and 4, which do we expect to have a stronger R2 ?

- Ans: Scenarios 3

c. Comparing scenarios 1 and 2, which do we expect has bigger/smaller SSE, SSR, and SST? (intuitively)

- Ans: Intuitively, scenarios 2 have bigger SSR and SST. For the part of SSR, I think scenarios 1 having slighly bigger one.

d. Comparing scenarios 3 and 4, which do we expect has bigger/smaller SSE, SSR, and SST? (intuitively)

- Ans: Intuitively,scenarios 4 has a bigger SST, SSE. For the part of SSR, it's not obvious to tell.

Question 2

a.

```
prog <- read.csv("programmer_salaries.txt", sep="\t", header=TRUE)
head(prog,3)
```

```
##   Experience Score Degree Salary
## 1          4    78      0   24.0
## 2          7   100      1   43.0
## 3          1    86      0   23.7
```

- Coefficients:

```
# Raw data regression
prog_regr <- lm( Salary ~ Experience + Score + Degree , data =prog )
summary(prog_regr)
```

```
##
## Call:
## lm(formula = Salary ~ Experience + Score + Degree, data = prog)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8963 -1.7290 -0.3375  1.9699  5.0480
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.9448     7.3808   1.076   0.2977
## Experience    1.1476     0.2976   3.856   0.0014 **
## Score         0.1969     0.0899   2.191   0.0436 *
## Degree        2.2804     1.9866   1.148   0.2679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.396 on 16 degrees of freedom
## Multiple R-squared:  0.8468, Adjusted R-squared:  0.8181
## F-statistic: 29.48 on 3 and 16 DF,  p-value: 9.417e-07
```

-R Square: Multiple R-squared: 0.8468, Adjusted R-squared: 0.8181

(1). First 5 values

```
#  (1). First 5 values of y HAT
head(prog_regr$fitted.values,5)
```

```
##        1        2        3        4        5
## 27.89626 37.95204 26.02901 32.11201 36.34251
```

```
# (2). epsilon ε
head(prog_regr$residuals,5)
```

```
##          1          2          3          4          5
## -3.8962605  5.0479568 -2.3290112  2.1879860 -0.5425072
```

---

b. Linear algebra

i. Create an X matrix that has a first column of 1s followed by columns of the independent variables

(only show the code)

```
X <- cbind(Intercept =1,as.matrix(prog[1:3]))
```

---

ii. Create a y vector with the Salary values

(only show the code)

```
mat.data4 <- prog$Salary
y <- matrix(mat.data4,ncol = 1)
```

iii. Compute the beta_hat vector of estimated regression coefficients (show the code and values)

```
beta_hat <- (solve(t(X)%*%X)) %*% t(X) %*%y
beta_hat
```

```
##                [,1]
## Intercept  7.944849
## Experience 1.147582
## Score      0.196937
## Degree     2.280424
```

iv. Compute a y_hat vector of estimated y hat values, and a res vector of residuals

(show the code and the first 5 values of y_hat and res)

- y hat

```
y_hat <- X %*% beta_hat
head(y_hat,5)
```

```
##          [,1]
## [1,] 27.89626
## [2,] 37.95204
## [3,] 26.02901
## [4,] 32.11201
## [5,] 36.34251
```

- res vector of residuals

```
residual <- y -y_hat;
head(residual,5)
```

```
##            [,1]
## [1,] -3.8962605
## [2,]  5.0479568
## [3,] -2.3290112
## [4,]  2.1879860
## [5,] -0.5425072
```

v. Using only the results from (i) – (iv), compute SSR, SSE and SST (show the code and values)

```
SSE <- sum((y -y_hat)^2)
SSR <- sum(( y_hat - mean(y))^2)
SST <- SSE + SSR

sprintf('SSE: %f ; SSR: %f; SST: %f', SSE,SSR, SST)
```

```
## [1] "SSE: 91.889487 ; SSR: 507.896013; SST: 599.785500"
```

---

c. Compute R2 for in two ways, and confirm you get the same results (show code and values):

i. Use any combination of SSR, SSE, and SST

```
r.square = SSR/SST ;
r.square
```

```
## [1] 0.8467961
```

ii. Use the squared correlation of vectors y and y hat

```
cor(y, y_hat) ^2
```

```
##           [,1]
## [1,] 0.8467961
```

- Ans: Yes. They're the same.

Question 3

- Data preprocessing

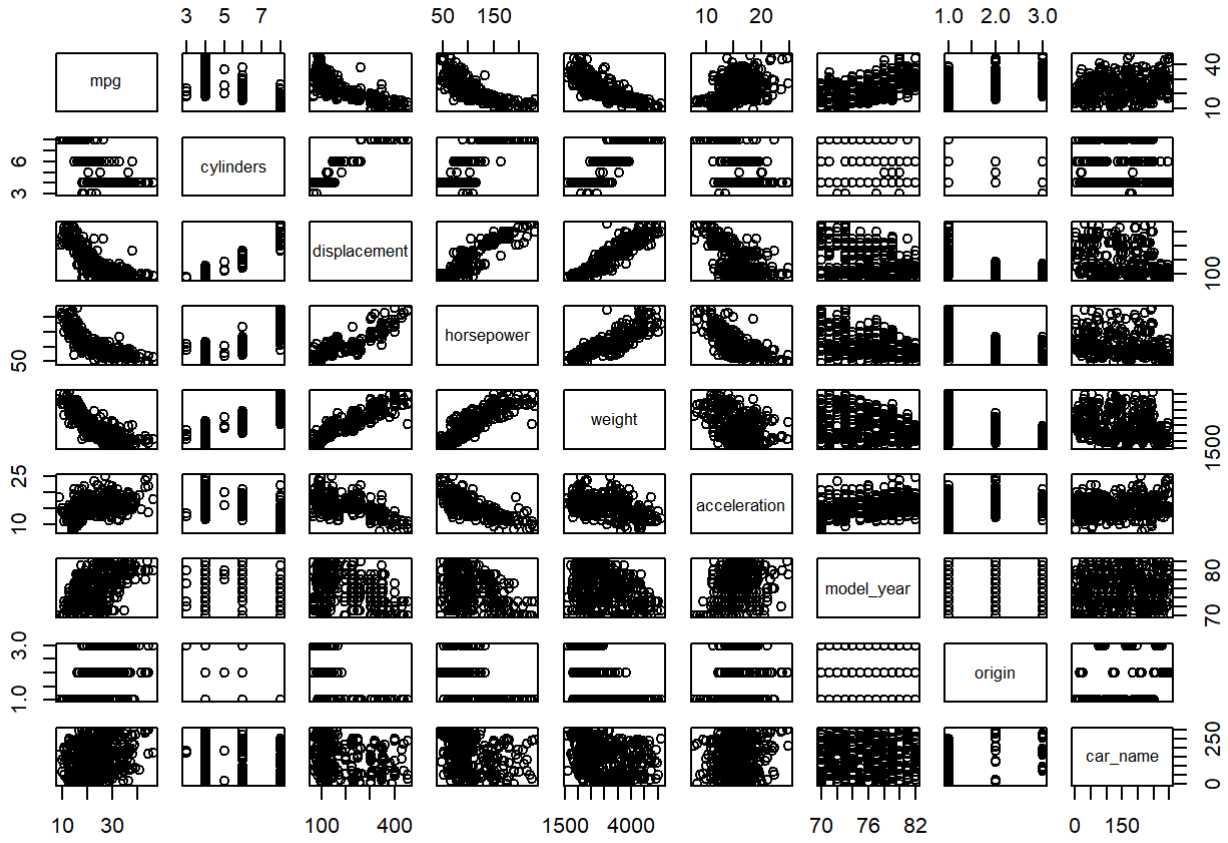
```
auto <- read.table("auto-data.txt", header=FALSE, na.strings = "?")
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower", "weight",
                 "acceleration", "model_year", "origin", "car_name")
auto$origin <-as.factor(auto$origin)
auto<-na.omit(auto)
head(auto,10)
```

```
##    mpg cylinders displacement horsepower weight acceleration model_year origin
## 1   18         8          307        130   3504         12.0         70      1
## 2   15         8          350        165   3693         11.5         70      1
## 3   18         8          318        150   3436         11.0         70      1
## 4   16         8          304        150   3433         12.0         70      1
## 5   17         8          302        140   3449         10.5         70      1
## 6   15         8          429        198   4341         10.0         70      1
## 7   14         8          454        220   4354          9.0         70      1
## 8   14         8          440        215   4312          8.5         70      1
## 9   14         8          455        225   4425         10.0         70      1
## 10  15         8          390        190   3850          8.5         70      1
##                       car_name
## 1  chevrolet chevelle malibu
## 2          buick skylark 320
## 3         plymouth satellite
## 4              amc rebel sst
## 5                ford torino
## 6           ford galaxie 500
## 7           chevrolet impala
## 8          plymouth fury iii
## 9           pontiac catalina
## 10        amc ambassador dpl
```

See whether there is any strong cor first

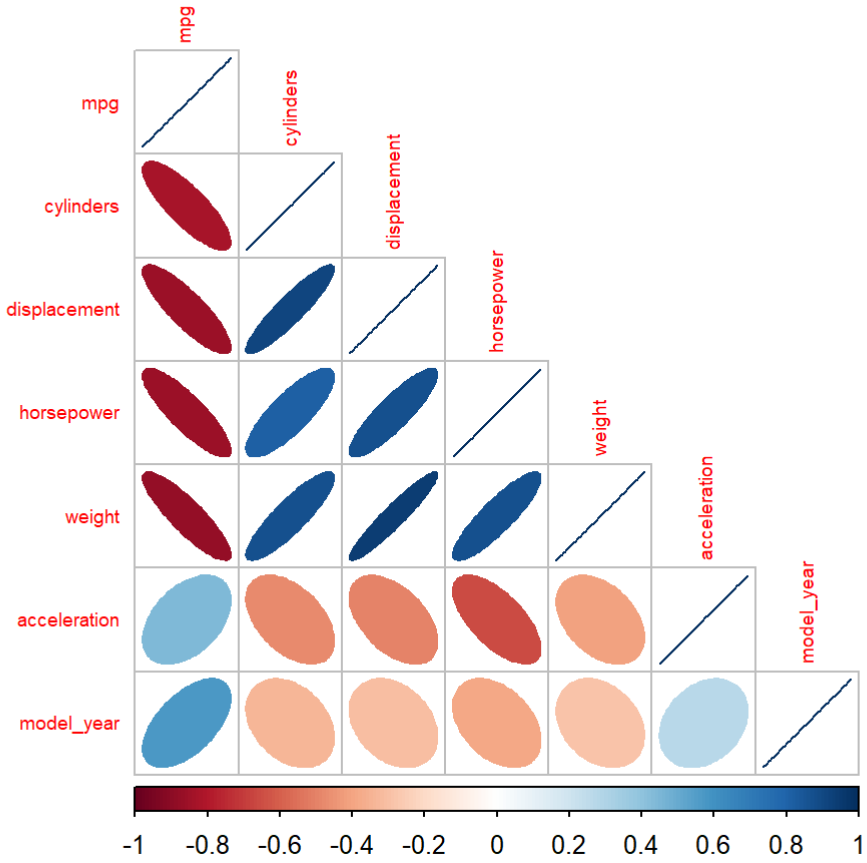
- Exploratory analysis:

```
plot(auto)
```



Except origin & car_name, the rest of column are numeric. Hence I plot it as corrplot to have a basic understanding of the dataset.

```
options(repr.plot.width = 14, repr.plot.height = 8)
cor_features <- cor(auto[,1:7], method='spearman')
corrplot::corrplot(cor_features, tl.cex=0.6, type='lower', method="ellipse")
```



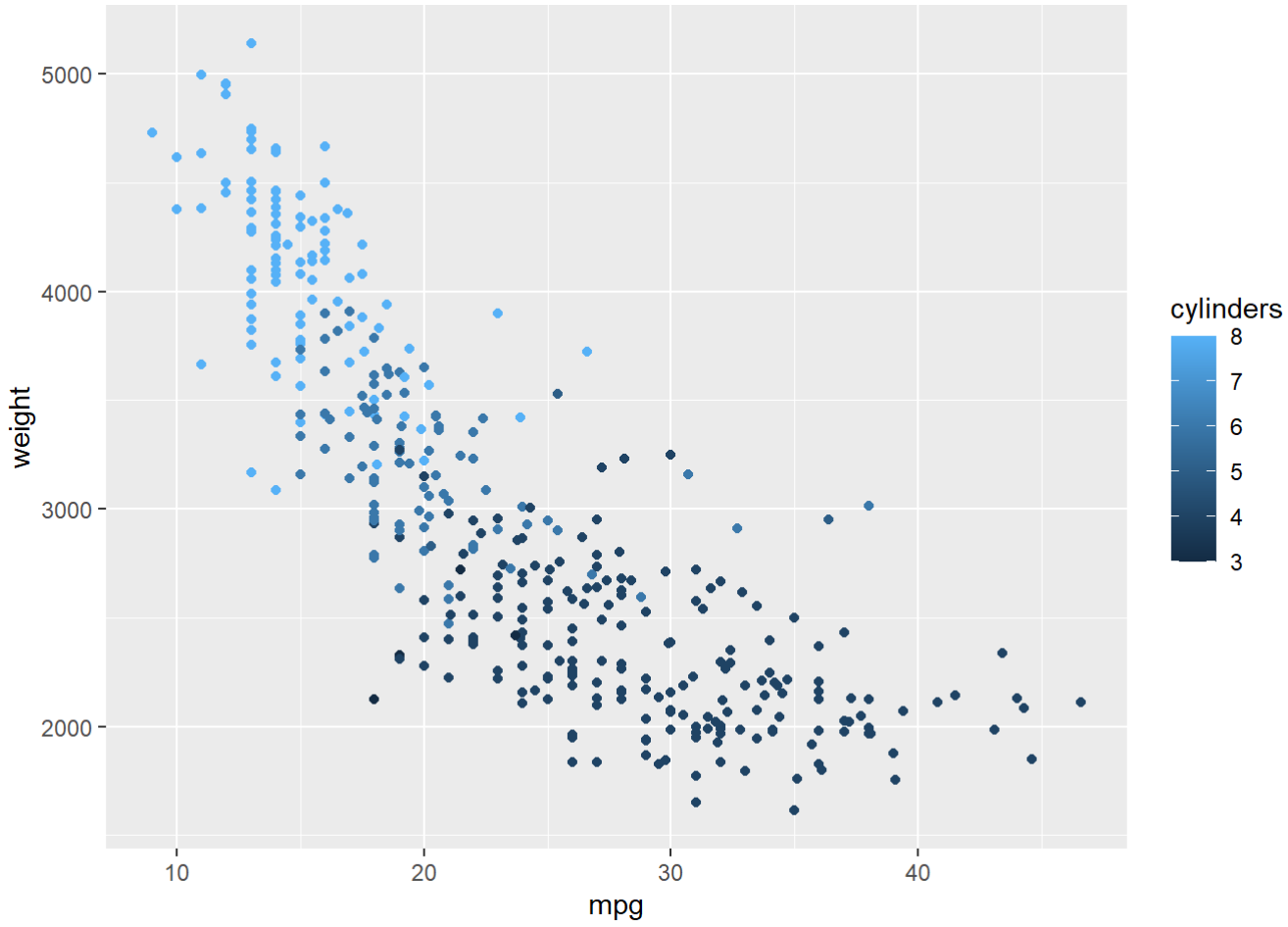

```
summary(auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration     model_year    origin    car_name
##  Min.   : 8.00   Min.   :70.00   1:245   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   2: 68   Class :character
##  Median :15.50   Median :76.00   3: 79   Mode  :character
##  Mean   :15.54   Mean   :75.98
##  3rd Qu.:17.02   3rd Qu.:79.00
##  Max.   :24.80   Max.   :82.00
```

a. Let's first try exploring this data and problem:

i. Visualize the data in any way you feel relevant (report only relevant/interesting ones)

```
ggplot(auto, aes(x = mpg, y = weight)) + geom_point(aes(color = cylinders))
```



- *Ans:* This one is pretty interesting, isn't it?

The insight is :

In general, the mpg has a negative relation with the weight of car, which is quite intuitively. However, after grouping, even when the car weight are close, we can tell that there is a huge variance among the mpg performance when the cylinder numbers is limited (The dark blue data points).

Comparing to the former , the mpg performance of the multi-cylinder car is not that disperse as their car weight are alike (the light blue data points).

My friend told me the reason behind it is because as numbers of cylinder decreases, the impact of the 'cylinder design' increases based on the functional positioning.

Notice that: cylinder is multi-valued discrete data type, which is used for grouping as above.

---

ii. Report a correlation table of all variables, rounding to two decimal places (in the cor() function, set use="pairwise.complete.obs" to handle missing values)

- *Ans:* Examine the correlations between variables and then examine relationships 1 by 1.
- ps: We skipped the last 2 columns because they are not continuous variable.

```
round(cor(auto[,c(1:7)], use="pairwise.complete.obs"),2)
```

```
##                mpg cylinders displacement horsepower weight acceleration
## mpg           1.00     -0.78        -0.81      -0.78  -0.83         0.42
## cylinders    -0.78      1.00         0.95       0.84   0.90        -0.50
## displacement -0.81      0.95         1.00       0.90   0.93        -0.54
## horsepower   -0.78      0.84         0.90       1.00   0.86        -0.69
## weight       -0.83      0.90         0.93       0.86   1.00        -0.42
## acceleration  0.42     -0.50        -0.54      -0.69  -0.42         1.00
## model_year    0.58     -0.35        -0.37      -0.42  -0.31         0.29
##              model_year
## mpg                0.58
## cylinders         -0.35
## displacement      -0.37
## horsepower        -0.42
## weight            -0.31
## acceleration       0.29
## model_year         1.00
```

---

iii. From the visualizations and correlations, which variables seem to relate to mpg?

- *Ans:*

cylinders , displacement , horsepower, weight are negative to mpg. Other features are positive relate to mpg.

---

iv. Which relationships might not be linear?

- *Ans:*

Any relation between the rest of the column to the colname of rigin & car_name will not be linear.

Because they are not continuous variable. Therefore, I did not show them in the table above as it's redundant information.

In the table above, as the the data points format into symmetrical but non-linear shape, the correlation will getting close to 0. Therefore, I guessed the correlation(0.29) between model_year and acceleration will be non-linear.

---

v. Are there any pairs of independent variables that are highly correlated (r > 0.7)?

- *Ans:* Yes, for example, cylinders & displacement (0.95), weight & displacement (0.93) etc.

---

b. Let's create a linear regression model where mpg is dependent upon all other suitable variables

```
summary(lm(mpg ~ cylinders+displacement+horsepower+weight+acceleration+model_year+origin, dat
a = auto))
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + origin, data = auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders    -4.897e-01  3.212e-01  -1.524 0.128215
## displacement  2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower   -1.818e-02  1.371e-02  -1.326 0.185488
## weight       -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
## acceleration  7.910e-02  9.822e-02   0.805 0.421101
## model_year    7.770e-01  5.178e-02  15.005  < 2e-16 ***
## origin2       2.630e+00  5.664e-01   4.643 4.72e-06 ***
## origin3       2.853e+00  5.527e-01   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.307 on 383 degrees of freedom
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

i.Which independent variables have a ‘significant’ relationship with mpg at 1% significance?

- Ans:

displacement , weight , model_year, origin2 , origin3

---

ii. Looking at the coefficients, is it possible to determine which independent variables are the most effective at increasing mpg? If so, which ones, and if not, why not? (hint: units!)

First thing first, we select the positive correlation one. And see the R Square individually.

```
summary(lm(mpg~acceleration , data = auto))[8]
```

```
## $r.squared
## [1] 0.1792071
```

```
summary(lm(mpg~model_year, data = auto))[8]
```

```
## $r.squared
## [1] 0.3370278
```

Explain:

For these 2 positive correlation to mpg columns (acceleration,model_year), the r square of model_year is higher than acceleration. Which indicates that SSE/SST is smaller in the individual ratio. If the sum square of error(residual error sum) is smaller, means the predictive linear line fit the y data points(mpg) much better as the x-axis is model_year.

---

c.Let's try to resolve some of the issues with our regression model above.

i. Create fully standardized regression results: are these slopes easier to compare?

(note: consider if you should standardize origin)

- *Ans.* :Sure! The slopes of standard regression are easier to compare because they are in the same scale. Additionaly, we should not standardize origin cuz it's categorical variable.

```
# [1:7] mpg to model_year (without origin and car_names)

auto_std<-data.frame(scale(auto[,1:7]))

auto_std<-cbind(auto_std, origin = auto$origin)

lm_std <- lm(mpg ~ cylinders+displacement+horsepower+weight+acceleration+model_year+origin, d
ata = auto_std)
summary(lm_std)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + model_year + origin, data = auto_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.15432 -0.26630 -0.01259  0.25440  1.71182
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.13213    0.03155  -4.187 3.50e-05 ***
## cylinders    -0.10703    0.07020  -1.524  0.12821
## displacement  0.32149    0.10261   3.133  0.00186 **
## horsepower   -0.08967    0.06761  -1.326  0.18549
## weight       -0.73028    0.07130 -10.243  < 2e-16 ***
## acceleration  0.02796    0.03472   0.805  0.42110
## model_year    0.36673    0.02444  15.005  < 2e-16 ***
## origin2       0.33696    0.07257   4.643 4.72e-06 ***
## origin3       0.36556    0.07082   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4236 on 383 degrees of freedom
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

ii. Regress mpg over each non-significant independent variable, individually.

Which ones become significant when we regress mpg over them individually?

- *Ans.* : First, we pick the non-significant features : cylinders , horsepower and acceleration.

```
summary(lm(mpg~ cylinders , data = auto))[4]
```

```
## $coefficients
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 42.915505  0.8348668  51.40401 8.828025e-176
## cylinders   -3.558078  0.1456755 -24.42468  1.311384e-80
```

```
summary(lm(mpg~ horsepower, data = auto))[4]
```

```
## $coefficients
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) 39.9358610 0.717498656  55.65984 1.220362e-187
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-81
```

```
summary(lm(mpg ~ acceleration, data = auto))[4]
```

```
## $coefficients
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept)  4.833250  2.0484999 2.359409 1.879614e-02
## acceleration 1.197624  0.1297859 9.227691 1.778576e-18
```

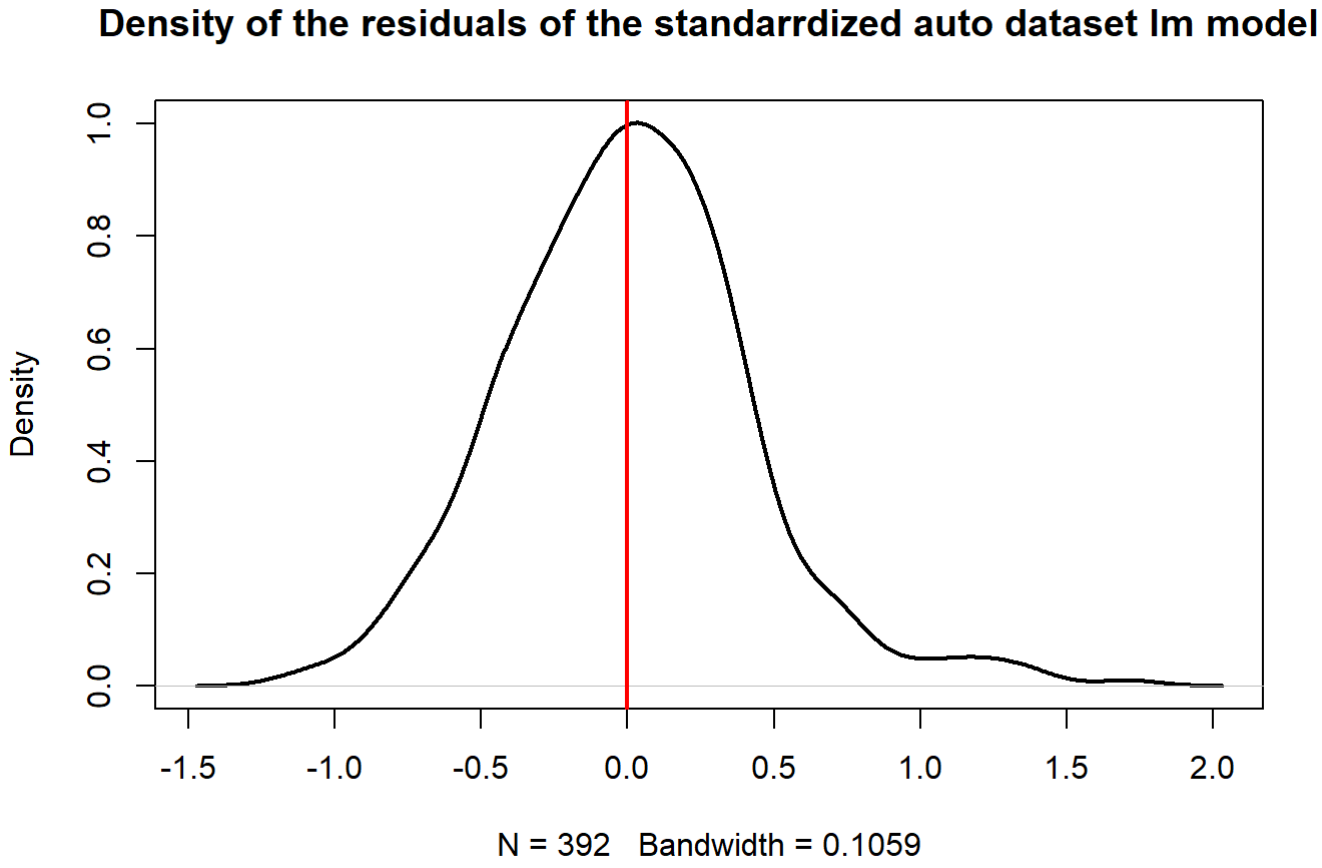
Due to the results above, all three (the non-significant features) become significant after standardization.

iii. Plot the density of the residuals: are they normally distributed and centered around zero? (get the residuals of a fitted linear model, e.g. regr <- lm(…), using regr$residuals

- *Ans.* :

About the question of whether it's normally distributed (centered at 0), we supposed to do the further QQ plot & shapiro.test as below:

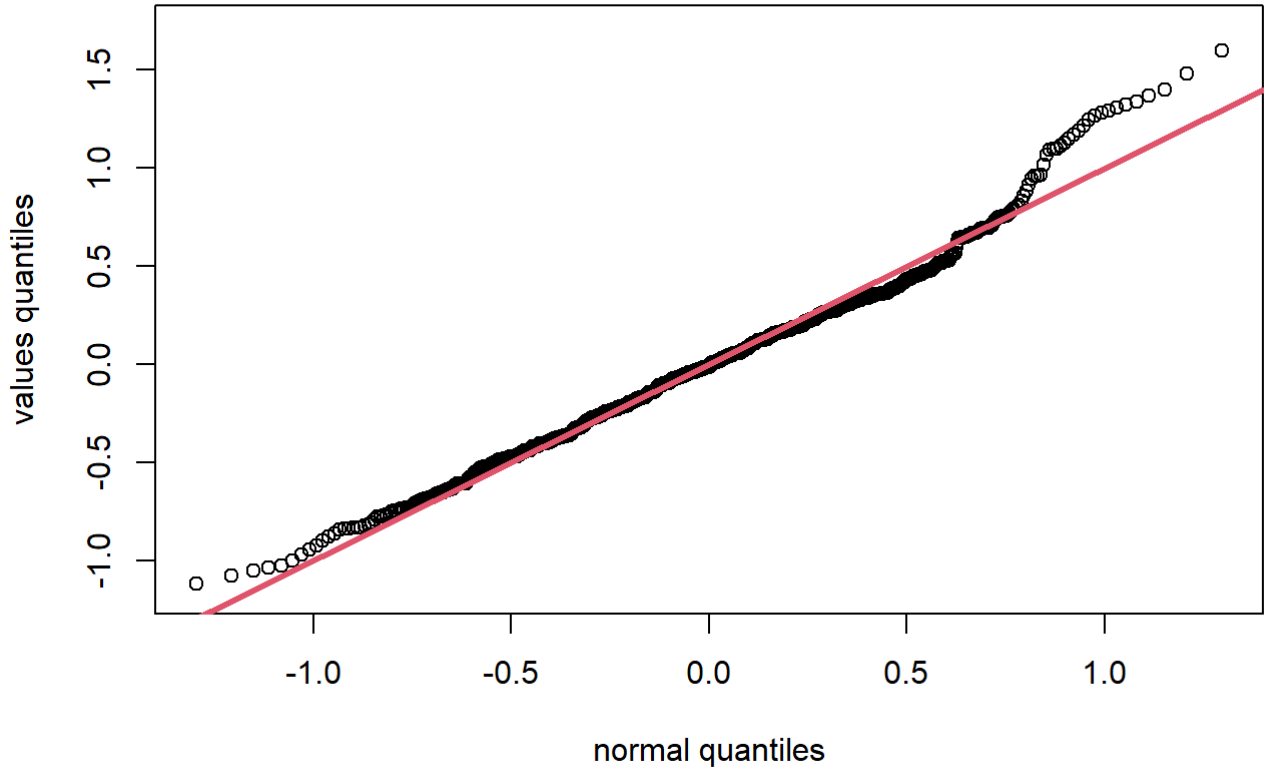
```
plot(density(lm_std$residuals), main='Density of the residuals of the standarrdized auto data
set lm model', lwd=2) + abline(v= mean(lm_std$residuals), col = "red", lwd = 2)
```




```
## integer(0)
```

For the plot of QQ- plot, we can tell that sth strange in the right tail of the data points.

```
norm_qq_plot(lm_std$residuals)
```



Hence, we do the further shapiro.test. As the p-value below, the p-value < 0.01, we can reject the default H0. Which means that residuals is not following normal distribution.

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

```
##
##  Shapiro-Wilk normality test
##
## data:  lm_std$residuals
## W = 0.98243, p-value = 0.0001061
```