HW10

110077443

4/18/2022

Please note that all code in this document is presented in a grey box and the output reflected below each box

- The below code allows lengthy lines of comments to display neatly within the grey box (wrapping it)

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
```

1) Using demo_simple_regression_sq.R, consider and compare the following scenarios:

- Scenario 1: Consider a very narrowly dispersed set of points that have a negative or positive steep slope
- Scenario 2: Consider a widely dispersed set of points that have a negative or positive steep slope
- Scenario 3: Consider a very narrowly dispersed set of points that have a negative or positive shallow slope
- Scenario 4: Consider a widely dispersed set of points that have a negative or positive shallow slope

a) Comparing scenarios 1 and 2, which do we expect to have a stronger R^2?

ANSWER ##

- Scenario 1 may have a stronger R^2 because the variance of y-hat is almost exactly modeled by the regression line.

b) Comparing scenarios 3 and 4, which do we expect to have a stronger R^2?

ANSWER ##

- Scenarios 3 may have a stronger R^2 because narrowly dispersed set of points indicate a more equal variance of y to regression.

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

ANSWER ##

- Scenario 2 should have a bigger SSE, smaller SSR, with a bigger SST intuitively.

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

ANSWER ##

- Scenarios 4 should have a bigger SSE, smaller SSR and bigger SST intuitively.

2) Performing regression ourselves on the programmer_salaries.txt dataset

```
# Importing data
prog <- read.table("programmer_salaries.txt", header = TRUE)

# Importing library
require(knitr)  # For creating tables with kable function
```

a) Using the lm() function to estimate the model Salary ~ Experience + Score + Degree

```
prog_regr <- lm(Salary ~ Experience + Score + Degree, data = prog)  # lm function
prog_lm <- summary(prog_regr)  # Need summary to read regression model output
```

Showing beta coefficients, R^2, first 5 values of y_hat and residuals:

```
# beta coefficients
prog_regr$coefficients
```

```
ANSWER ## -> (Intercept)  Experience       Score      Degree
ANSWER ## ->    7.944849    1.147582    0.196937    2.280424
```

```
# R^2
prog_lm$r.squared
```

```
ANSWER ## -> [1] 0.8467961
```

```
# First 5 values of y_hat
y_hat1 <- prog_regr$fitted.values
head(y_hat1, 5)
```

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

```
# First 5 values of residuals
head(prog_lm$residuals, 5)
```

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

b) Using linear algebra (and the geometric view of regression) to estimate the regression

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

```
X <- cbind(1, prog[1:3])  # First column with 1's
names(X)[1] <- "Intercept"  # Renaming first column
X <- as.matrix(X)  # Converting data frame to matrix
kable(head(X, 5), caption = "X Matrix", align = "c")  # Print table with first 5 observations of matrix
```

Table 1: X Matrix

Intercept	Experience	Score	Degree
1	4	78	0
1	7	100	1
1	1	86	0
1	5	82	1
1	8	86	1

ii) Creating a y vector with the Salary values

```
y_vec <- prog[, 4]  # y vector of salary values
y <- as.matrix(y_vec)  # Converting y vector to matrix
```

iii) Computing the beta_hat vector of estimated regression coefficients

```
b_hat <- solve(t(X) %*% X) %*% (t(X) %*% y)  # applying formula
kable(b_hat, caption = "beta_hat", align = "c")  # Print values
```

Table 2: beta_hat

Intercept	7.944849
Experience	1.147582
Score	0.196937
Degree	2.280424

iv) Computing 'y_hat' vector of estimated y_hat values, and a 'res' vector of residuals

```
y_hat <- X %*% b_hat  # Calculating y_hat
res <- y - y_hat  # Calculating residuals
```

Showing first 5 values of y_hat:

```
kable(head(y_hat, 5), caption = "y_hat", align = "c")  # Print values
```

Table 3: y_hat

27.89626
37.95204
26.02901
32.11201
36.34251

Showing first 5 values of residuals:

```
kable(head(res, 5), caption = "residuals", align = "c")  # Print values
```

Table 4: residuals

-3.8962605
5.0479568
-2.3290112
2.1879860
-0.5425072

v) Computing SSR, SSE and SST - using only the results from (i) – (iv)

```
SSE = sum((y - y_hat)^2)
SSE  # Print
```

```
ANSWER ## -> [1] 91.88949
```

```
SSR = sum((y_hat - mean(y))^2)
SSR  # Print
```

ANSWER ## -> [1] 507.896

```
SST = sum((y - mean(y))^2)
SST  # Print
```

ANSWER ## -> [1] 599.7855

```
# SST
SSR + SSE  # Double checking SST result matches
```

ANSWER ## -> [1] 599.7855

C) Compute R^2 for scenario 2 in two ways, and confirm you get the same results

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

```
r_squared <- SSR/SST
r_squared  # Print
```

ANSWER ## -> [1] 0.8467961

ii) Use the squared correlation of vectors y and y_hat

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

```
ANSWER ## ->            [,1]
ANSWER ## -> [1,] 0.8467961
```

ANSWER ##

- Results for i) and ii) **match**.

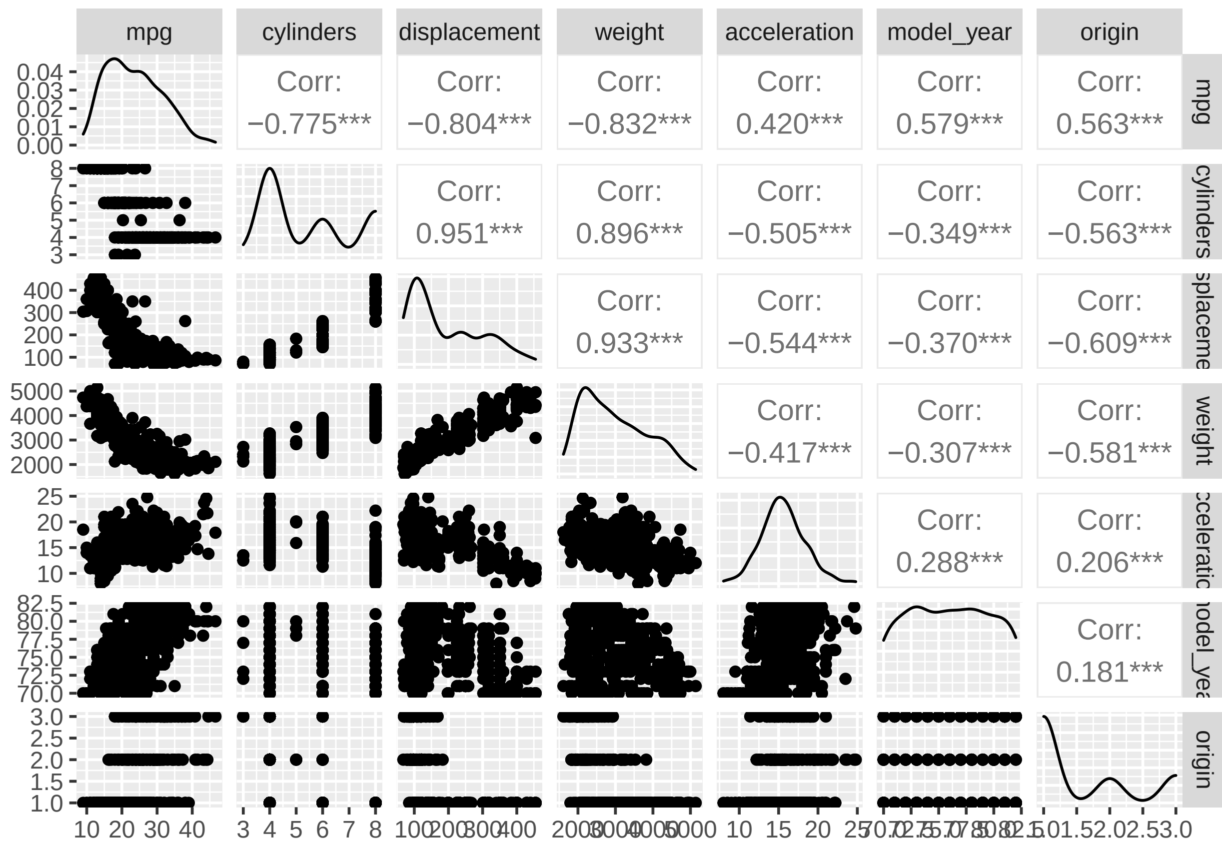
3) What kind of cars have higher fuel efficiency (mpg) from 'auto' data set?

```
# Importing data
auto <- read.table("auto-data.txt", header = FALSE, na.strings = "?")

# Renaming variables
names(auto) <- c("mpg", "cylinders", "displacement", "horsepower",
    "weight", "acceleration", "model_year", "origin", "car_name")
```

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)

```r
# Creating new data frame to include metric variables
auto_df <- data.frame(auto[1:8])

# Visualize
require(GGally)  # Used to visualize relationships between quantitative variables
quant_df <- auto_df[, c("mpg", "cylinders", "displacement", "weight",
    "acceleration", "model_year", "origin")]  # Extracting variables
ggpairs(quant_df)
```



ANSWER ##

- From the above visualization, we can see that weight has the greatest correlation coefficient, followed by displacement, and then cylinders in relation to our dependent variable, 'mpg'.
- The 'horsepower' variable was excluded from this visualization because of missing values.

ii) Reporting a correlation table of all variables, rounding to two decimal places

```r
# Check correlation analysis outputs in relation to
# dependent variable
```

```r
auto_cor <- round(cor(auto_df, use = "pairwise.complete.obs"),
    2)  # Accounting for missing values
auto_cor <- as.data.frame(auto_cor)
kable(auto_cor, caption = "Correlation Analysis Outputs", align = "c")  # Print table
```

Table 5: Correlation Analysis Outputs

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin
mpg	1.00	-0.78	-0.80	-0.78	-0.83	0.42	0.58	0.56
cylinders	-0.78	1.00	0.95	0.84	0.90	-0.51	-0.35	-0.56
displacement	-0.80	0.95	1.00	0.90	0.93	-0.54	-0.37	-0.61
horsepower	-0.78	0.84	0.90	1.00	0.86	-0.69	-0.42	-0.46
weight	-0.83	0.90	0.93	0.86	1.00	-0.42	-0.31	-0.58
acceleration	0.42	-0.51	-0.54	-0.69	-0.42	1.00	0.29	0.21
model_year	0.58	-0.35	-0.37	-0.42	-0.31	0.29	1.00	0.18
origin	0.56	-0.56	-0.61	-0.46	-0.58	0.21	0.18	1.00

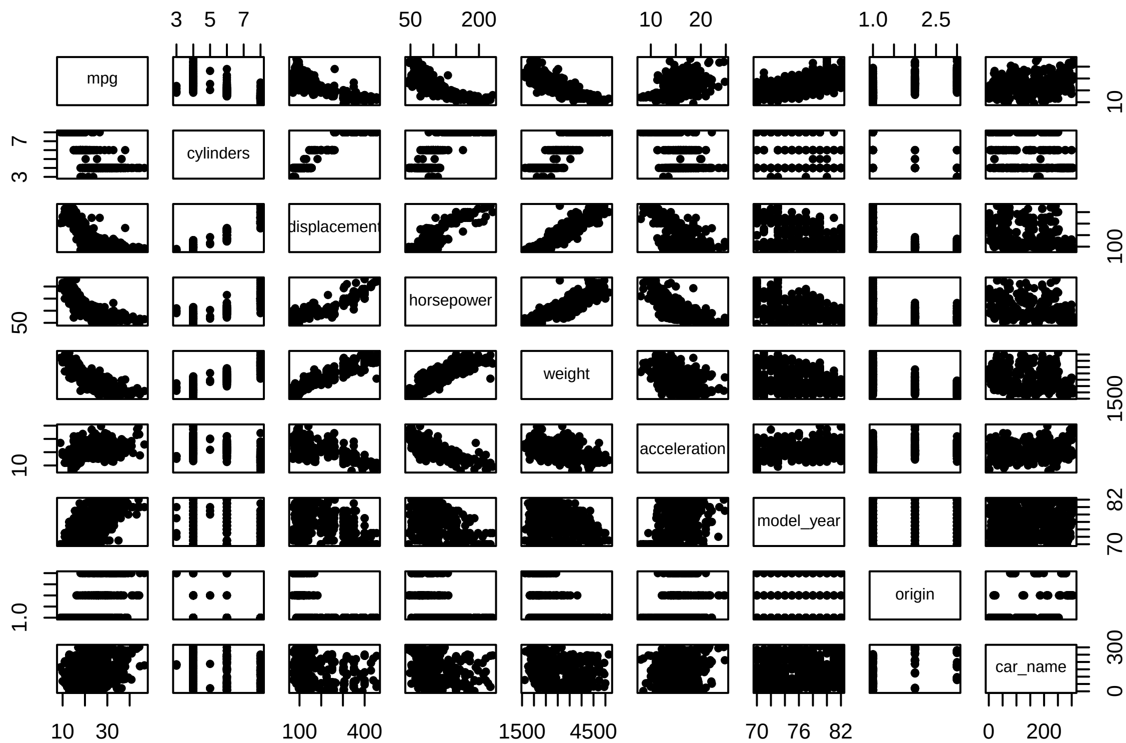
iii) From the visualizations and correlations, which variables seem to relate to mpg?

ANSWER ##

- As evident by the correlation table and visualization, it appears that cylinders, displacement, horsepower, and weight have a negative relationship, while acceleration, model_year, and origin have a positive relationship with mpg. **Weight appears to relate to mpg the most, and acceleration the least**.

iv) Which relationships might not be linear?

```r
plot(auto, pch = 20)  # Scatter plot for all variables
```



ANSWER ##

- From the above plot it appears **displacement, horsepower, and possibly weight do not have a linear relationship with mpg**.

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

```
pairs_IV <- which(abs(auto_cor) > 0.7 & row(auto_cor) < col(auto_cor),
    arr.ind = TRUE)
high_cor <- matrix(colnames(auto_cor)[pairs_IV], ncol = 2)  # reconstruct names from positions
kable(high_cor, caption = "Highly correlated Independant Variables",
    align = "c")  # Print table
```

Table 6: Highly correlated Independant Variables

mpg	cylinders
mpg	displacement
cylinders	displacement
mpg	horsepower
cylinders	horsepower
displacement	horsepower
mpg	weight
cylinders	weight

Table 6: Highly correlated Independant Variables

displacement	weight
horsepower	weight

ANSWER ##

- The pairs of independent variables that are highly correlated are shown in the above table
- Source code for above calculation: https://stackoverflow.com/questions/67708565/extract-pairs-of-variables-with-high-correlation

b) Creating a linear regression model where mpg is dependent upon all other suitable variables

```
# Creating a regression model to plot variables in relation
# to mpg
auto_lm <- lm(mpg ~ cylinders + displacement + horsepower + weight +
    acceleration + model_year + factor(origin), data = auto_df,
    na.action = na.exclude)
```

i) Which independent variables have a 'significant' relationship with mpg at 1% significance?

```
require(stargazer)  # Helps understand results of regression analysis
stargazer(auto_lm, type = "text")  # Print (coefficients)
```

```
ANSWER ## ->
ANSWER ## -> ===============================================
ANSWER ## ->                         Dependent variable:
ANSWER ## ->                     ---------------------------
ANSWER ## ->                                 mpg
ANSWER ## -> -----------------------------------------------
ANSWER ## -> cylinders                     -0.490
ANSWER ## ->                               (0.321)
ANSWER ## ->
ANSWER ## -> displacement                 0.024***
ANSWER ## ->                               (0.008)
ANSWER ## ->
ANSWER ## -> horsepower                    -0.018
ANSWER ## ->                               (0.014)
ANSWER ## ->
ANSWER ## -> weight                       -0.007***
ANSWER ## ->                               (0.001)
ANSWER ## ->
ANSWER ## -> acceleration                   0.079
ANSWER ## ->                               (0.098)
ANSWER ## ->
ANSWER ## -> model_year                    0.777***
ANSWER ## ->                               (0.052)
```

```
ANSWER ## ->
ANSWER ## -> factor(origin)2              2.630***
ANSWER ## ->                               (0.566)
ANSWER ## ->
ANSWER ## -> factor(origin)3              2.853***
ANSWER ## ->                               (0.553)
ANSWER ## ->
ANSWER ## -> Constant                    -17.955***
ANSWER ## ->                               (4.677)
ANSWER ## ->
ANSWER ## -> -----------------------------------------------
ANSWER ## -> Observations                    392
ANSWER ## -> R2                             0.824
ANSWER ## -> Adjusted R2                    0.821
ANSWER ## -> Residual Std. Error      3.307 (df = 383)
ANSWER ## -> F Statistic          224.451*** (df = 8; 383)
ANSWER ## -> ===============================================
ANSWER ## -> Note:               *p<0.1; **p<0.05; ***p<0.01
```

ANSWER ##

- **Displacement, weight, model_year, and origin** have a 'significant' relationship with mpg at 1%

ii) Looking at the coefficients, is it possible to determine which independent variables are the most effective at increasing mpg?

ANSWER ##

- Simply looking at the current coefficients, it is **difficult** to determine the most effective independent variables because the coefficients have different units.
- Example: 'Weight' may be measured in kg/lb, while 'model_year' will use the year as its units. _ Thus, when when comparing with mpg, the results will not be consistent without standardizing first.

c) Resolving some of the issues with our regression model above

i) Creating fully standardized regression results: are these slopes easier to compare?

```r
auto_df_scaled <- as.data.frame(scale(auto_df))  # Standardizing data
auto_lm_std <- lm(mpg ~ cylinders + displacement + horsepower +
    weight + acceleration + model_year + factor(origin), data = auto_df_scaled,
    na.action = na.exclude)  # fully standardized regression
stargazer(auto_lm_std, type = "text")  # Print standardized regression results (coefficients)
```

```
ANSWER ## ->
ANSWER ## -> ============================================================
ANSWER ## ->                                     Dependent variable:
ANSWER ## ->                                 ---------------------------
ANSWER ## ->                                             mpg
ANSWER ## -> ------------------------------------------------------------
ANSWER ## -> cylinders                                 -0.107
```

```
ANSWER ## ->                                              (0.070)
ANSWER ## ->
ANSWER ## -> displacement                                0.320***
ANSWER ## ->                                              (0.102)
ANSWER ## ->
ANSWER ## -> horsepower                                   -0.090
ANSWER ## ->                                              (0.068)
ANSWER ## ->
ANSWER ## -> weight                                     -0.727***
ANSWER ## ->                                              (0.071)
ANSWER ## ->
ANSWER ## -> acceleration                                  0.028
ANSWER ## ->                                              (0.035)
ANSWER ## ->
ANSWER ## -> model_year                                  0.368***
ANSWER ## ->                                              (0.024)
ANSWER ## ->
ANSWER ## -> factor(origin)0.532551687239475             0.336***
ANSWER ## ->                                              (0.072)
ANSWER ## ->
ANSWER ## -> factor(origin)1.7793491667766               0.365***
ANSWER ## ->                                              (0.071)
ANSWER ## ->
ANSWER ## -> Constant                                   -0.133***
ANSWER ## ->                                              (0.032)
ANSWER ## ->
ANSWER ## -> -----------------------------------------------------------
ANSWER ## -> Observations                                   392
ANSWER ## -> R2                                            0.824
ANSWER ## -> Adjusted R2                                   0.821
ANSWER ## -> Residual Std. Error                     0.423 (df = 383)
ANSWER ## -> F Statistic                         224.451*** (df = 8; 383)
ANSWER ## -> ===========================================================
ANSWER ## -> Note:                               *p<0.1; **p<0.05; ***p<0.01
```

ANSWER ##

- These results should be **easier to compare** because the coefficients are expressed in units of standard deviations and not different units as in our previous model.
- We can now see that **weight is the most effective** at increasing mpg.
- Not it has a negative relationship and therefore **as weight decreases, mpg increases**.

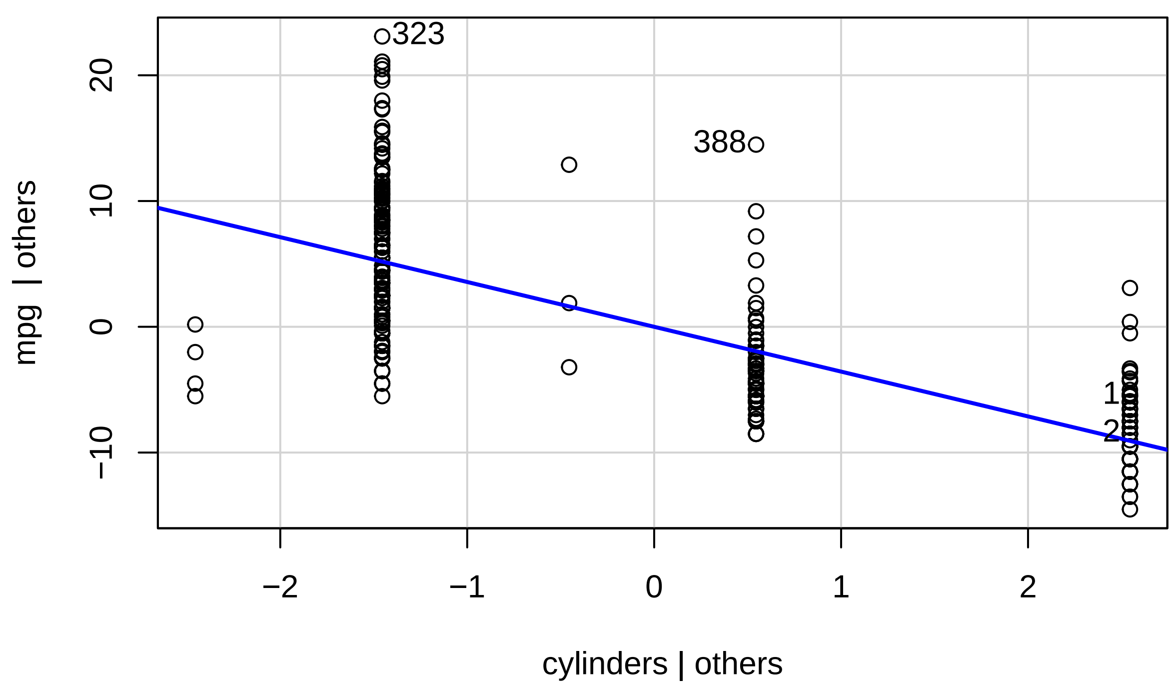
ii) Which 'non-significant' independent variable becomes 'significant' when we regress mpg over them individually?

```
require(car)  # Used to run avPlots function

# Regress mpg over cylinders
cyl_regr <- lm(mpg ~ cylinders, data = auto, na.action = na.exclude)
summary(cyl_regr)  # Print
```

```
##
```

```
## Call:
## lm(formula = mpg ~ cylinders, data = auto, na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2607  -3.3841  -0.6478   2.5538  17.9022
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.9493     0.8330   51.56   <2e-16 ***
## cylinders    -3.5629     0.1458  -24.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.942 on 396 degrees of freedom
## Multiple R-squared:  0.6012, Adjusted R-squared:  0.6002
## F-statistic: 597.1 on 1 and 396 DF,  p-value: < 2.2e-16
```

```
avPlots(cyl_regr)  # Plot
```

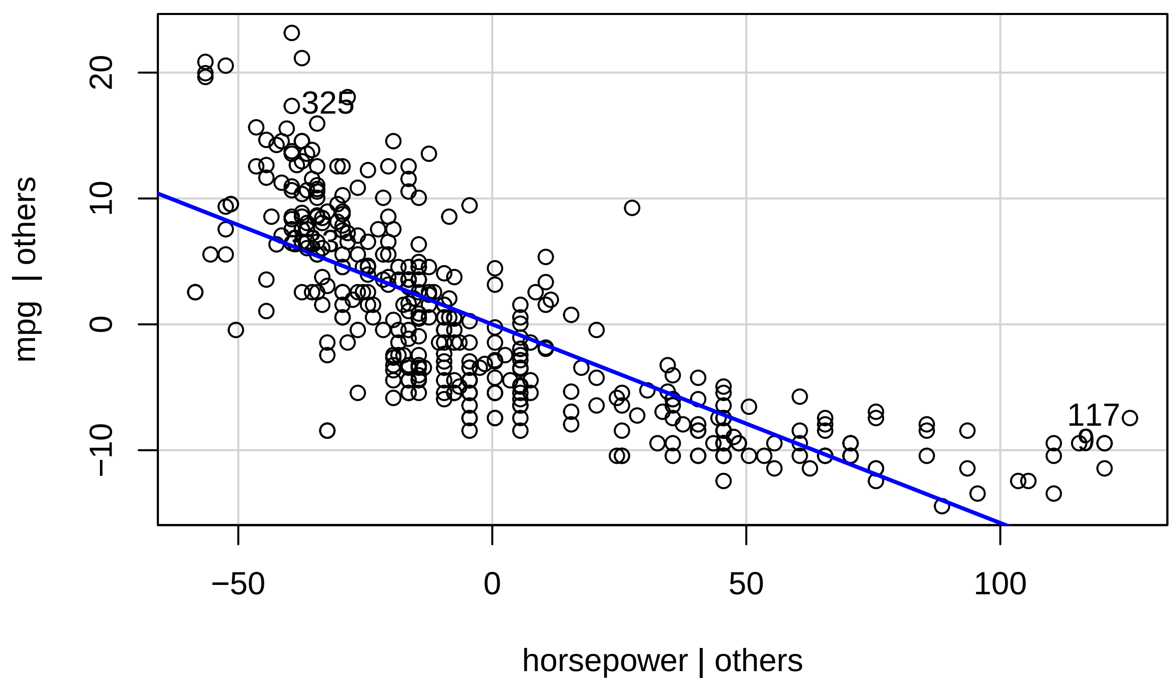



```
# Regress mpg over horsepower
hp_regr <- lm(mpg ~ horsepower, data = auto, na.action = na.exclude)
summary(hp_regr)  # Print
```

```
##
```

```
## Call:
## lm(formula = mpg ~ horsepower, data = auto, na.action = na.exclude)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

```r
avPlots(hp_regr)  # Plot
```

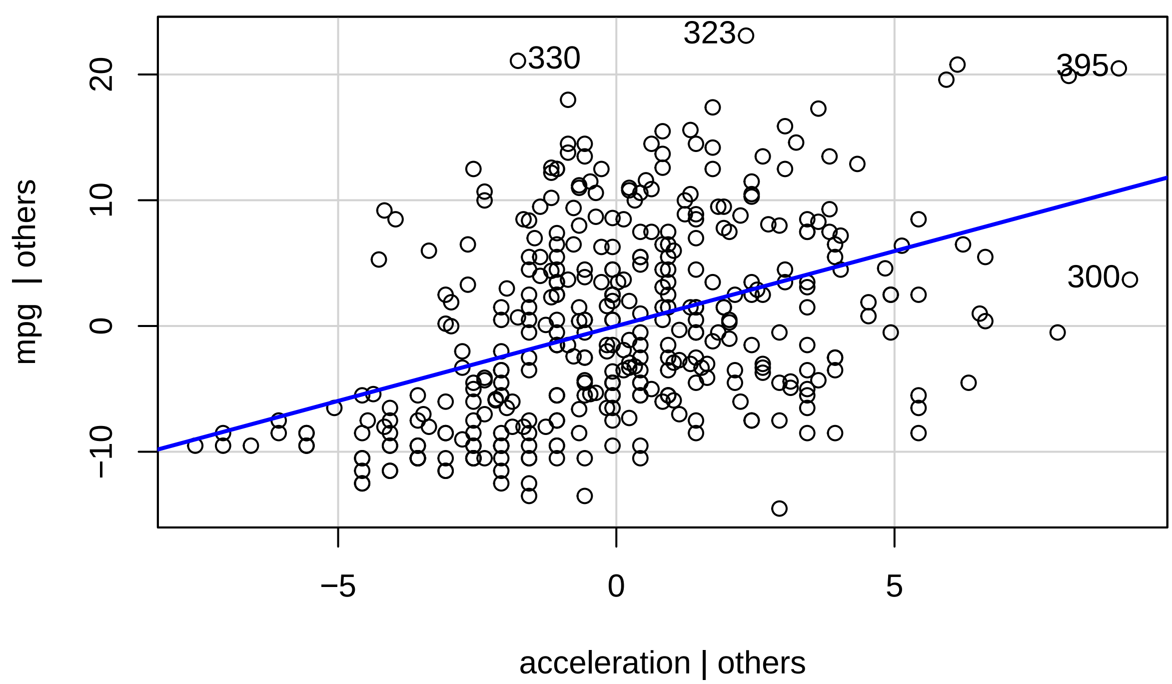



```r
# Regress mpg over acceleration
acc_regr <- lm(mpg ~ acceleration, data = auto, na.action = na.exclude)
summary(acc_regr)  # Print
```

```
##
```

```
## Call:
## lm(formula = mpg ~ acceleration, data = auto, na.action = na.exclude)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.007  -5.636  -1.242   4.758  23.192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.9698     2.0432   2.432   0.0154 *
## acceleration   1.1912     0.1292   9.217   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.101 on 396 degrees of freedom
## Multiple R-squared:  0.1766, Adjusted R-squared:  0.1746
## F-statistic: 84.96 on 1 and 396 DF,  p-value: < 2.2e-16
```

```r
avPlots(acc_regr)  # Plot
```

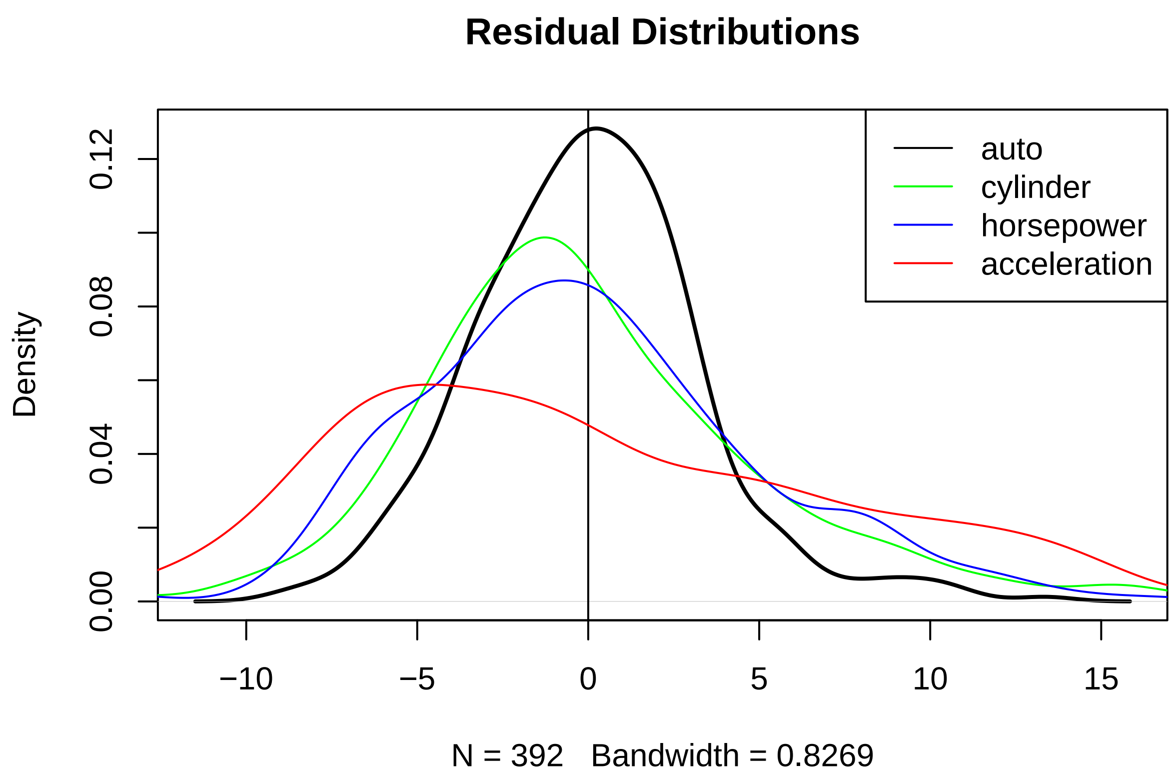


ANSWER ##

- **cylinders, horsepower, and acceleration** were considered 'non significant', but when regressing mpg over them individually, they **all appear to be 'significant'** based on their p-values.

iii) Plot the density of the residuals: are they normally distributed and centered around zero?

```
# Plotting residuals
plot(density(auto_lm$residuals), lwd = 2, main = "Residual Distributions")
abline(v = mean(auto_lm$residuals))
lines(density(cyl_regr$residuals), col = "green")
lines(density(hp_regr$residuals), col = "blue")
lines(density(acc_regr$residuals), col = "red")
legend(x = "topright", lty = 1, c("auto", "cylinder", "horsepower",
    "acceleration"), col = c("black", "green", "blue", "red"))
```



ANSWER ##

- From the density plot, the residuals **do not** appear to be normally distributed.