HW_Week10_108020033

Che-Wei, Chang

2023-04-19 helped by 108020024, 108020031

Question 1)

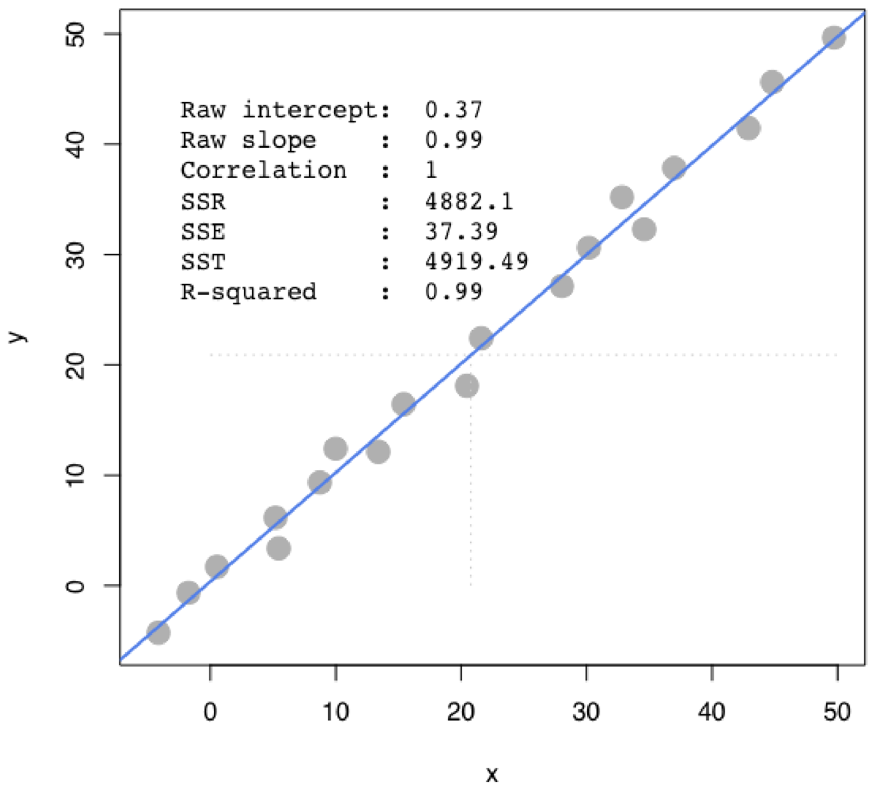
We will use the interactive_regression() function from CompStatsLib again – Windows users please make sure your desktop scaling is set to 100% and RStudio zoom is 100%; alternatively, run R from the Windows Command Prompt.

```
# Import the library
library(compstatslib)
```

To answer the questions below, understand each of these four scenarios by simulating them:

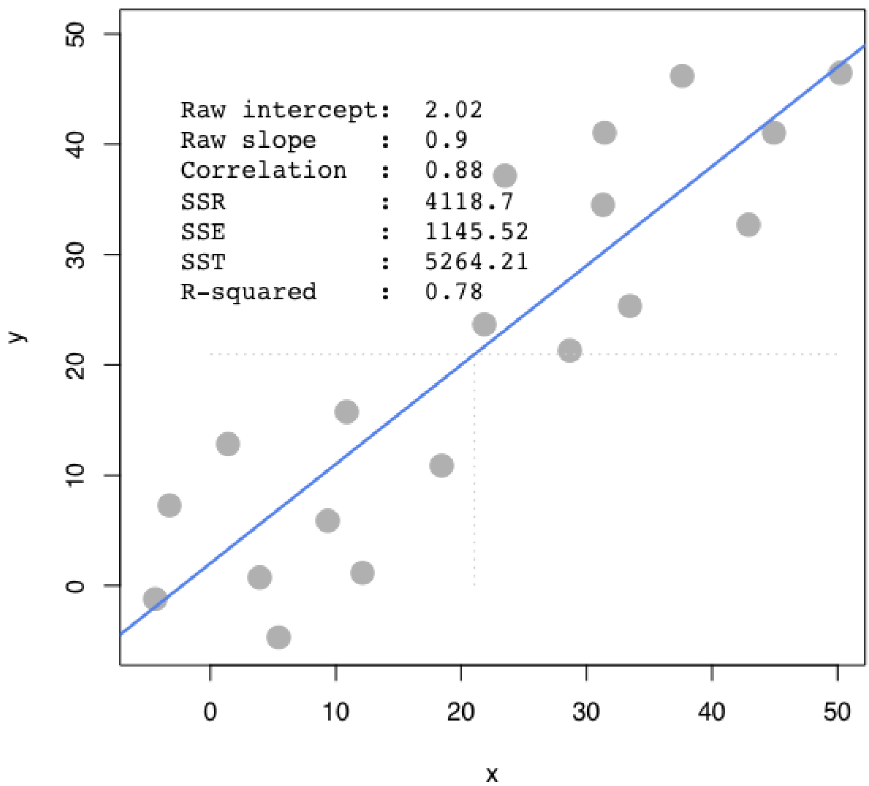
Scenario 1: Consider a very narrowly dispersed set of points that have a negative or positive steep slope

```
knitr::include_graphics("Plot_1.png")
```



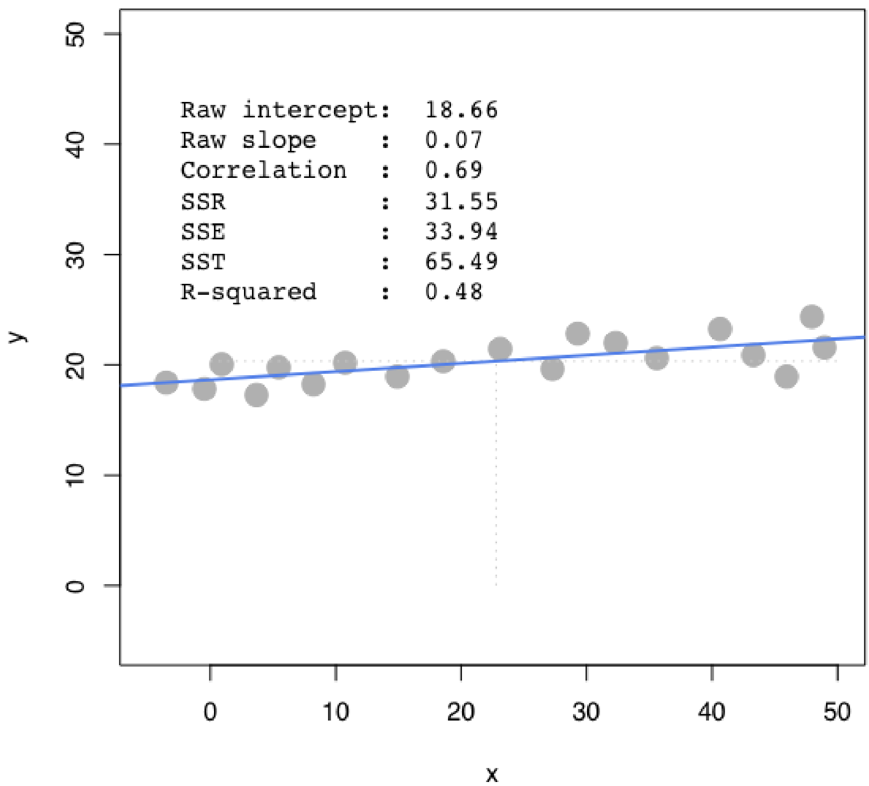
Scenario 2: Consider a widely dispersed set of points that have a negative or positive steep slope

```
knitr::include_graphics("Plot_2.png")
```


Raw intercept: 2.02
Raw slope : 0.9
Correlation : 0.88
SSR : 4118.7
SSE : 1145.52
SST : 5264.21
R-squared : 0.78
y
x
0
10
20
30
40
50

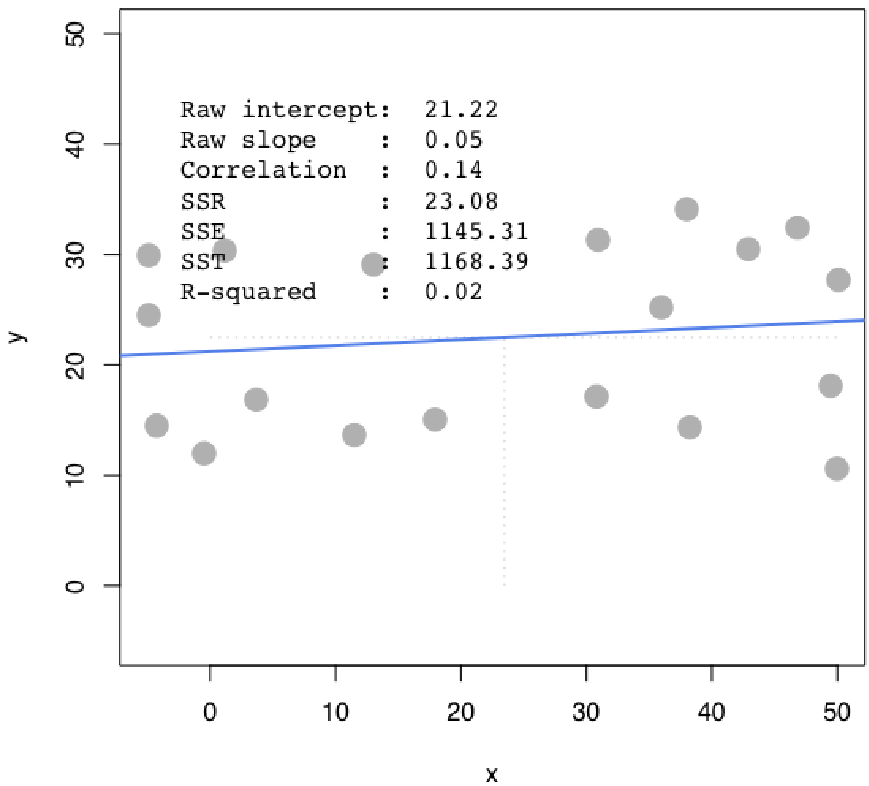
Scenario 3: Consider a very narrowly dispersed set of points that have a negative or positive shallow slope

```
knitr::include_graphics("Plot_3.png")
```



Scenario 4: Consider a widely dispersed set of points that have a negative or positive shallow slope

```
knitr::include_graphics("Plot_4.png")
```



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

```
# By two plots of scenario 1 and scenario 2, we can see that scenario 1 has a stronger R^2
# than scenario 2.This is because in scenario 1, the data points are very narrowly dispersed,
# meaning there is less variation in the data and a stronger linear relationship between the
# x and y variables. In contrast, scenario 2 has widely dispersed data points, meaning there
# is more variation in the data and a weaker linear relationship between the x and y variables.
```

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

```
# By two plots of scenario 3 and scenario 4, we can see that scenario 3 has a stronger R^2
# than scenario 4.This is because in scenario 3, the data points are very narrowly dispersed,
# meaning there is less variation in the data and a stronger linear relationship between the
# x and y variables. In contrast, scenario 4 has widely dispersed data points, meaning there
# is more variation in the data and a weaker linear relationship between the x and y variables.
```

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

```
# In scenarios 1 and 2, we expect scenario 1 to have smaller SSE, SSR, and SST.
#(i.e. scenario 2 has bigger SSE, SSR and SST.) This is because in scenario 1,
# the data points are more tightly clustered around the regression line, leading
# to a smaller error term and a stronger linear relationship. In contrast,
# in scenario 2, the data points are spread out, leading to a larger error
# term and a weaker linear relationship.
```

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

```
# In scenarios 3 and 4, we expect scenario 3 to have smaller SSE, SSR, and SST.
# (i.e. scenario 4 has bigger SSE, SSR, and SST.) This is because in scenario 3,
# the data points are more tightly clustered around the regression line, leading
# to a smaller error term and a stronger linear relationship. In contrast,
# in scenario 4, the data points are spread out, leading to a larger error
# term and a weaker linear relationship.
```

Question 2)

Let's analzye the programmer_salaries.txt dataset we saw in class. Read the file using read.csv("programmer_salaries.txt", sep="") because the columns are separated by tabs ().

```
# Read the data
df <- read.csv("programmer_salaries.txt", sep = "\t")
```

a. Use the lm() function to estimate the regression model Salary ~ Experience + Score + Degree
   Show the beta coefficients, R-square, and the first 5 values of y ($fitted.values)and($residuals)

```
# Fit the regression model
model <- lm(Salary ~Experience + Score + Degree, df)

# extract the beta coefficients and R-square
```

```r
coefficients <- coef(model)
R_square <- summary(model)$r.squared

# extract the first 5 values of y hat and residuals
y_hat <- head(fitted(model), 5)
res <- head(resid(model), 5)

# print the result
cat("beta: ", coefficients, "\n")
```

```
## beta:  7.944849 1.147582 0.196937 2.280424
```

```r
cat("R-square: ", R_square, "\n")
```

```
## R-square:  0.8467961
```

```r
cat("first 5 values of y hat: ", y_hat, "\n")
```

```
## first 5 values of y hat:  27.89626 37.95204 26.02901 32.11201 36.34251
```

```r
cat("residuals: ", res, "\n")
```

```
## residuals:  -3.896261 5.047957 -2.329011 2.187986 -0.5425072
```

b. Use only linear algebra and the geometric view of regression to estimate the regression yourself:

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

```r
# Create the X matrix
X <- cbind(rep(1, nrow(df)), df$Experience, df$Score, df$Degree)
```

   ii. Create a y vector with the Salary values (only show the code)

```r
# put the data into y variable
y <- df$Salary

# Create y vector
y_vec <- as.matrix(y)
```

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

```r
# Compute beta_hat
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y_vec
beta_hat
```

```
##          [,1]
## [1,] 7.944849
## [2,] 1.147582
## [3,] 0.196937
## [4,] 2.280424
```

iv. Compute a y_hat vector of estimated y values, and a res vector of residuals (show the code and the first 5 values of y_hat and res)

```
# Compute y_hat and res
y_hat <- X %*% beta_hat
res <- y - y_hat

# Show the result
head(y_hat, 5)
```

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

```
head(res, 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)

```
# Calculate SSR, SSE, SST
SSR <- sum((y_hat - mean(y))^2)
SSE <- sum(res^2)
SST <- sum((y - mean(y))^2)
cat("SSR: ", SSR, "\n")
```

```
## SSR:  507.896
```

```
cat("SSE: ", SSE, "\n")
```

```
## SSE:  91.88949
```

```
cat("SST: ", SST, "\n")
```

```
## SST:  599.7855
```

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

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

```
# Calculate R square using SSR, SST
R_square_ssr_sst <- SSR / SST
cat("Using SSR, SST\n")
```

```
## Using SSR, SST
```

```
cat("R square: ", R_square_ssr_sst, "\n")
```

```
## R square:  0.8467961
```

```
# Calculate R square using SSE, SST
R_square_sse_sst <- 1 - (SSE/SST)
cat("Using SSE, SST\n")
```

```
## Using SSE, SST
```

```
cat("R square: ", R_square_sse_sst, "\n")
```

```
## R square:  0.8467961
```

ii. Use the squared correlation of vectors y and y

```
# Compute R square using squared correlation of y and y_hat
R_square_corr <- cor(y, y_hat)^2
cat("R square: ", R_square_corr)
```

```
## R square:  0.8467961
```

Question 3)

We're going to take a look back at the early heady days of global car manufacturing, when American, Japanese, and European cars competed to rule the world. Take a look at the data set in file auto-data.txt. We are interested in explaining what kind of cars have higher fuel efficiency (mpg).

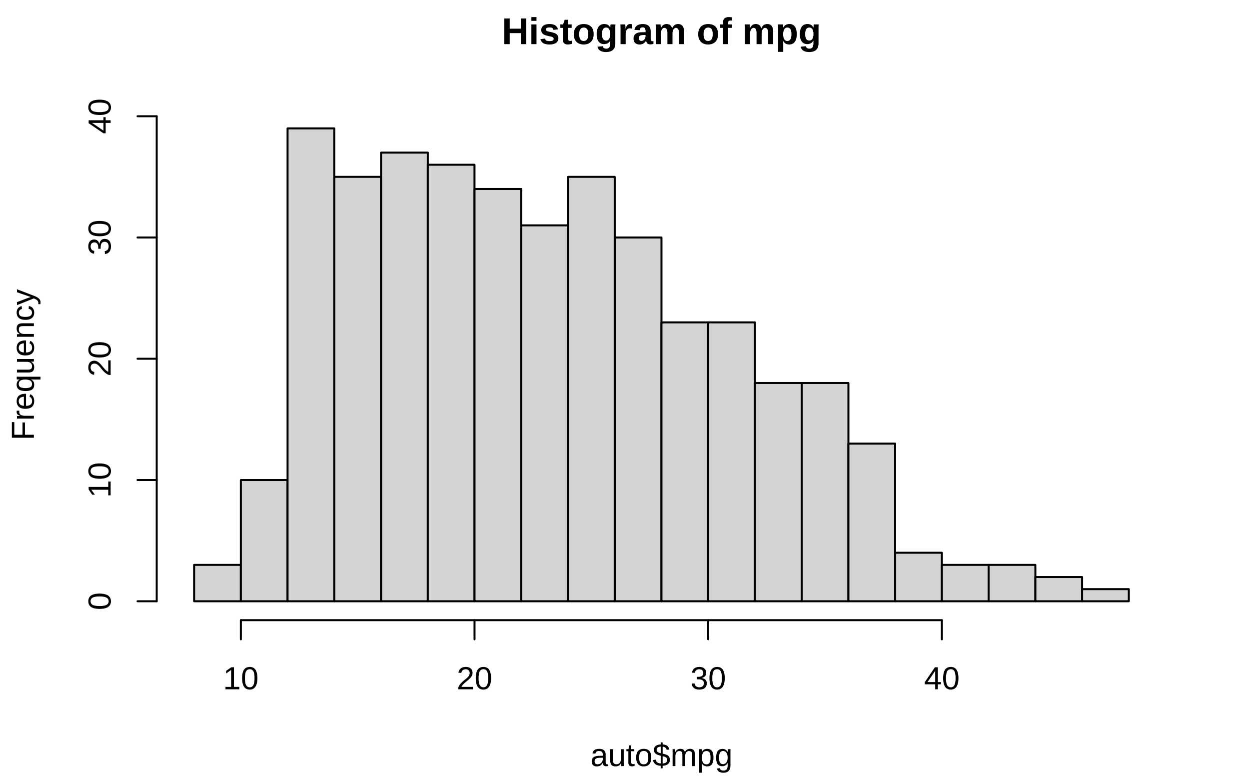
Note that the data has missing values ('?' in data set), and lacks a header row with variable names:

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

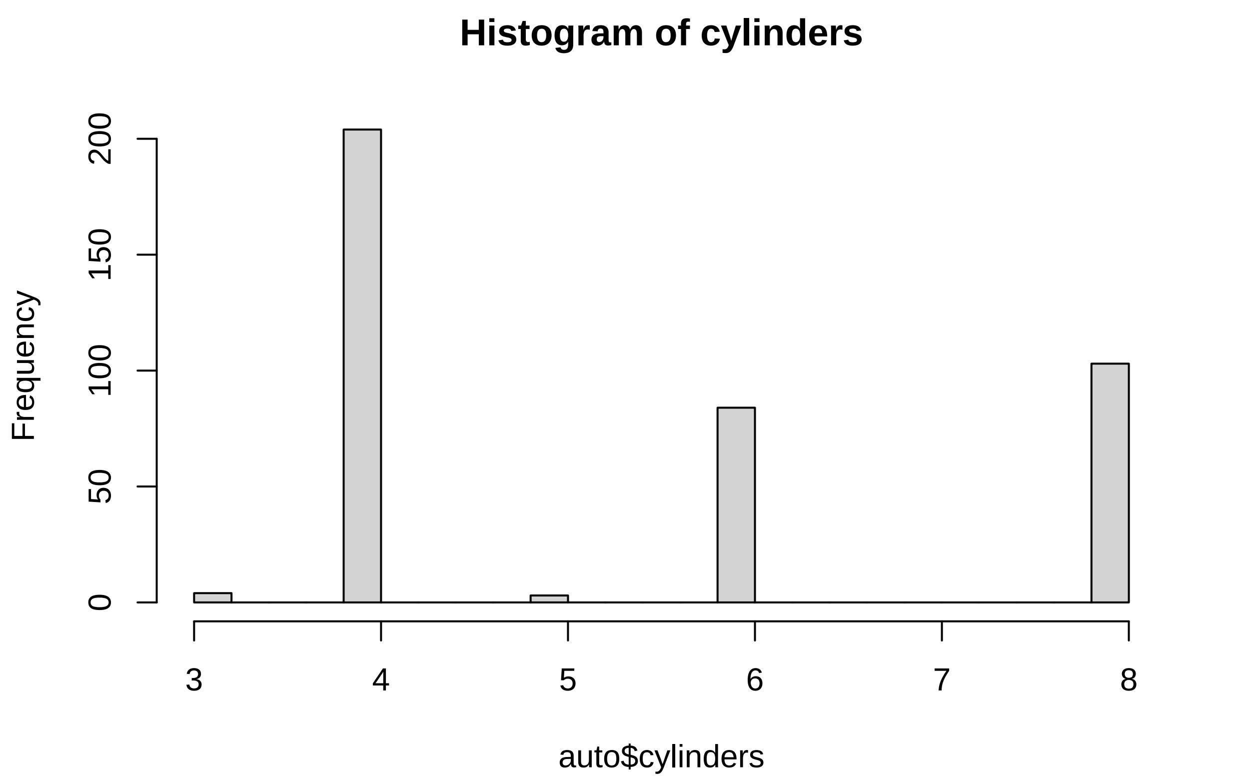
a. Let's first try exploring this data and problem:

i. Visualize the data as you wish (report only relevant/interesting plots)

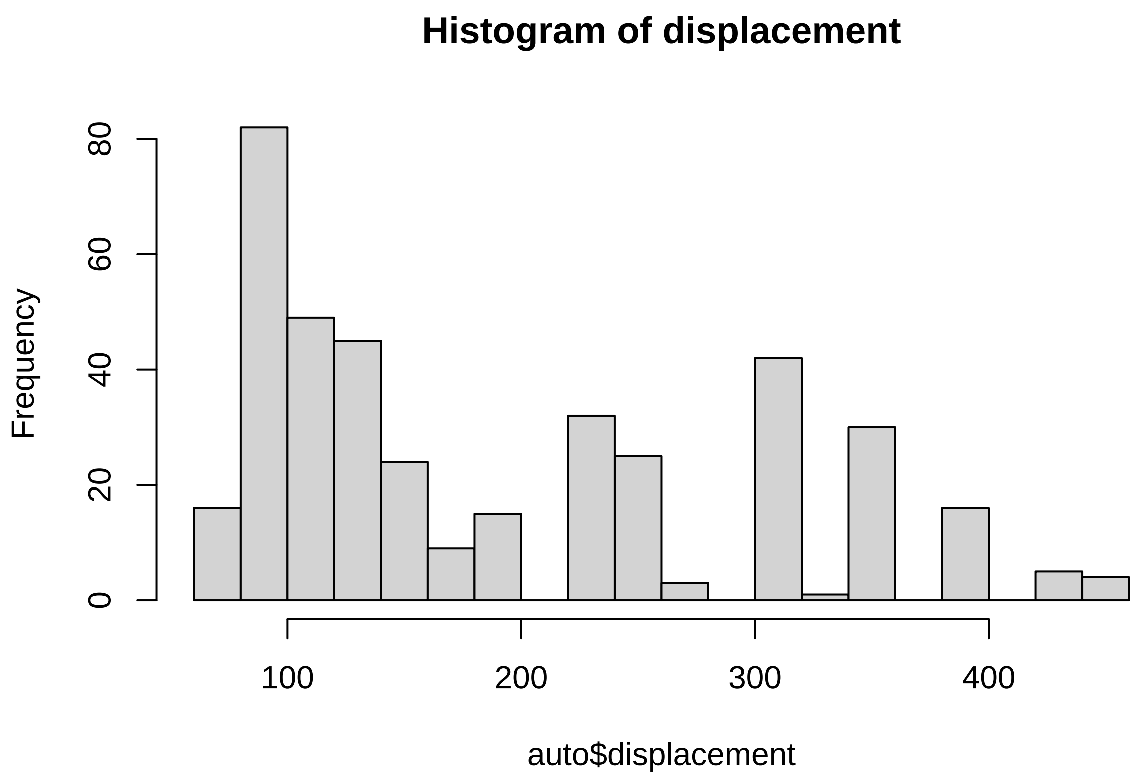
```r
# Draw the histogram of single variable
hist(auto$mpg, breaks = 20, main = "Histogram of mpg")
```



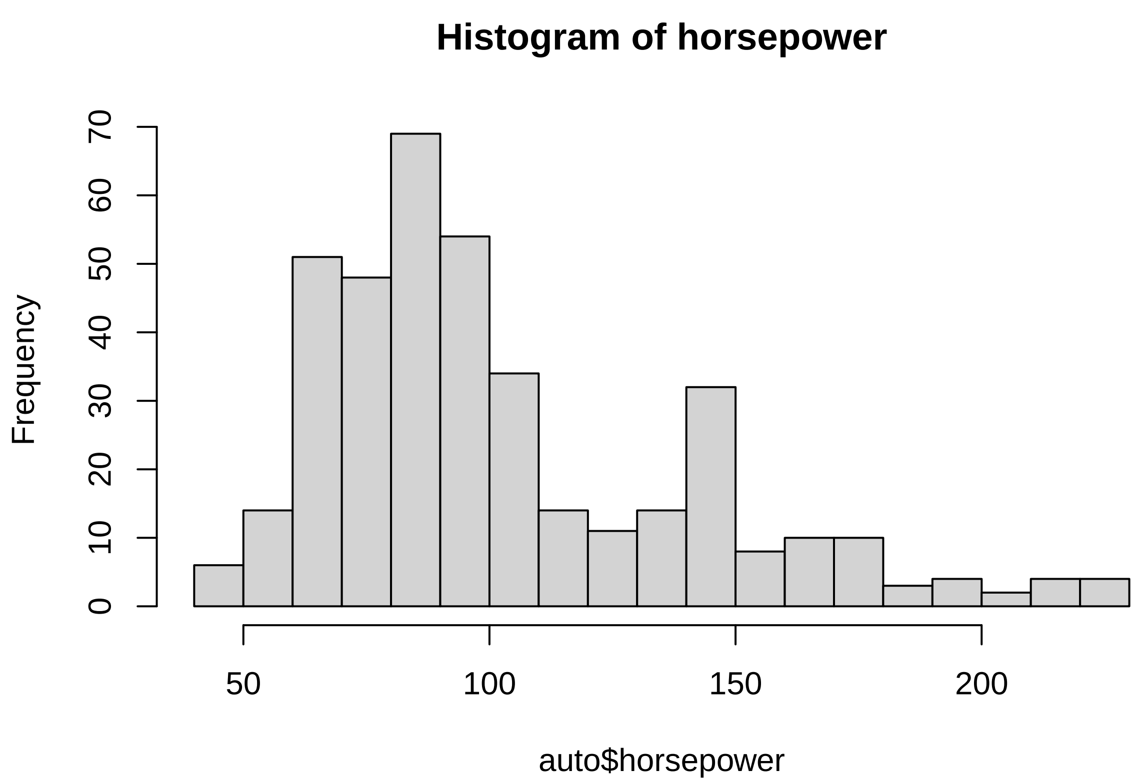

```r
hist(auto$cylinders, breaks = 20, main = "Histogram of cylinders")
```



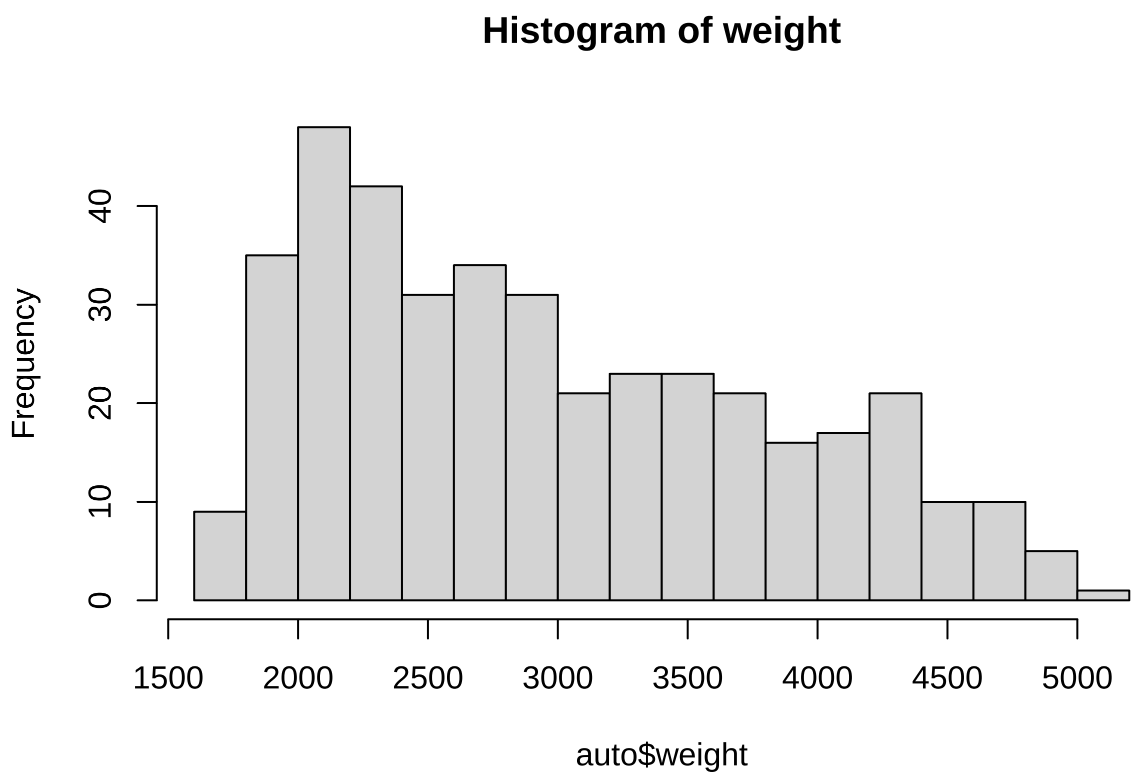
```r
hist(auto$displacement, breaks = 20, main = "Histogram of displacement")
```



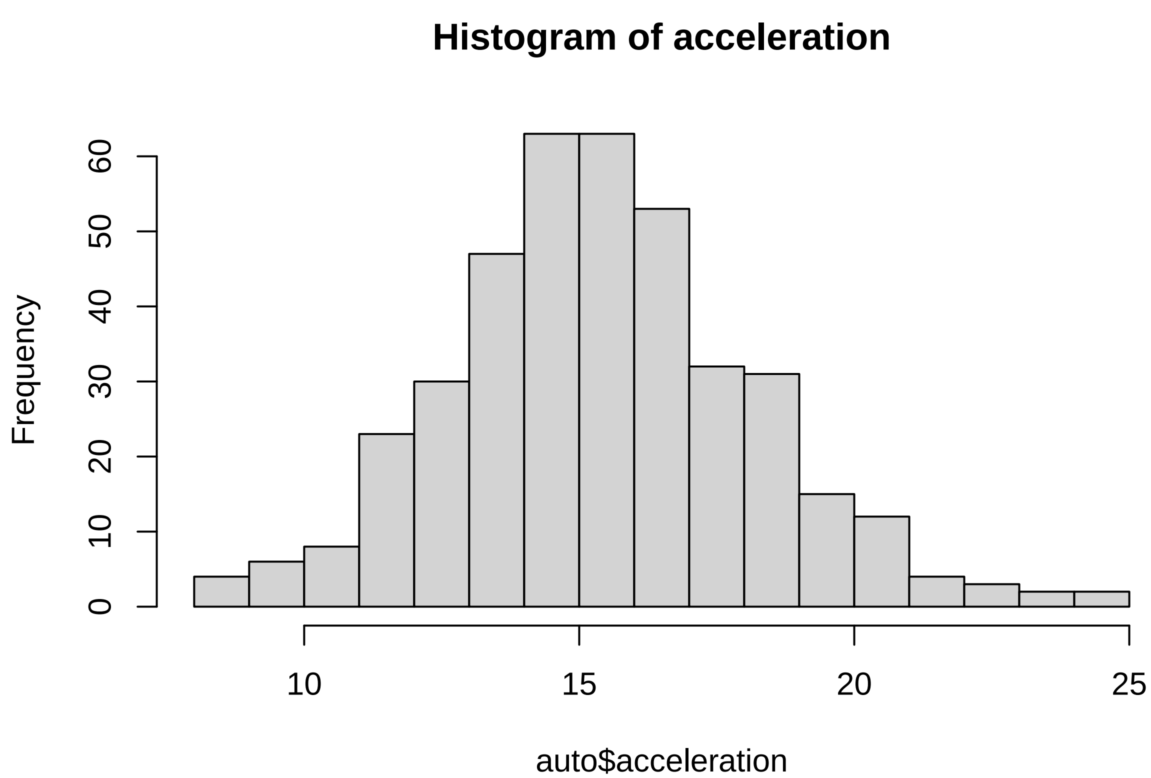

```r
hist(auto$horsepower, breaks = 20, main = "Histogram of horsepower")
```



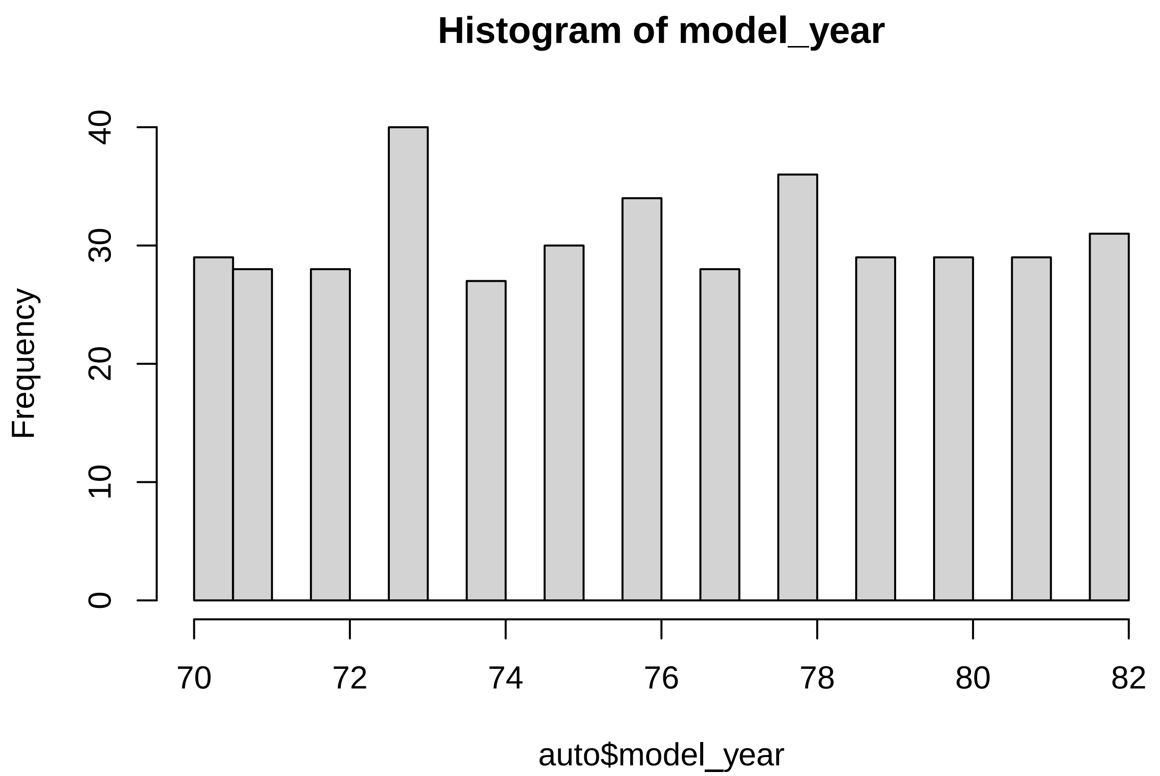
```r
hist(auto$weight, breaks = 20, main = "Histogram of weight")
```



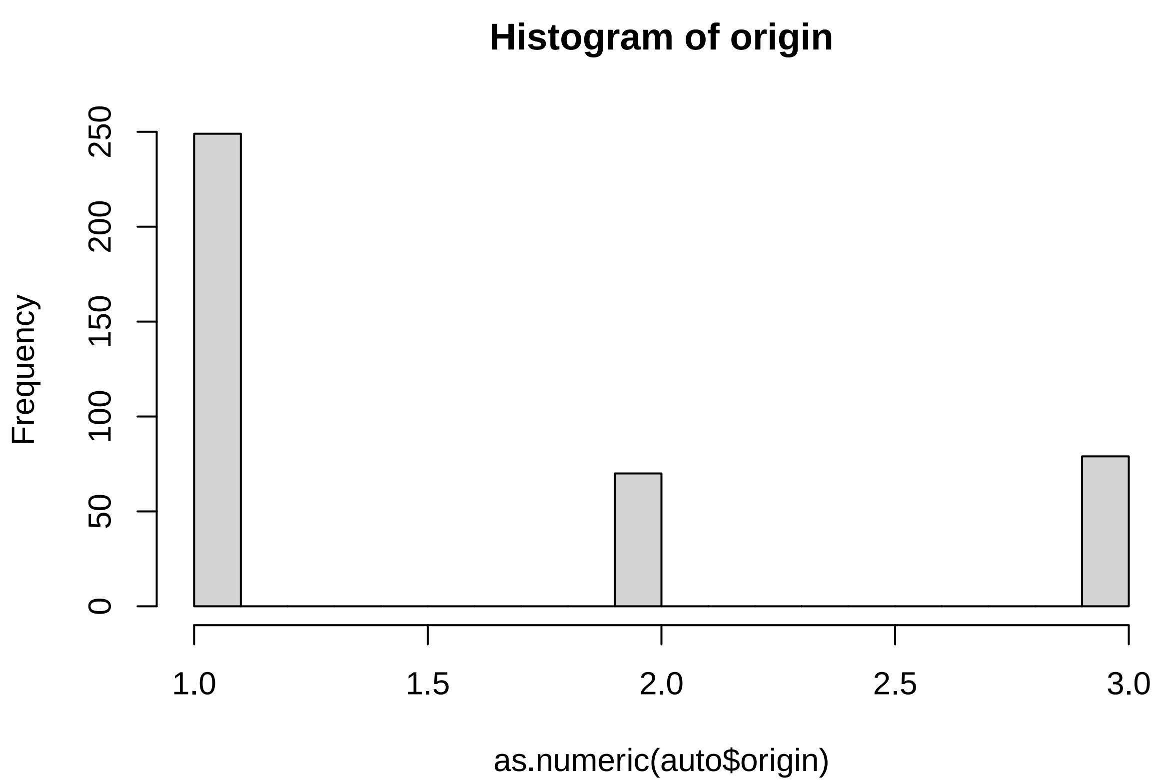

```r
hist(auto$acceleration, breaks = 20, main = "Histogram of acceleration")
```



```r
hist(auto$model_year, breaks = 20, main = "Histogram of model_year")
```




```r
hist(as.numeric(auto$origin), breaks = 20, main = "Histogram of origin")
```



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)

```
# Remove the car_name col since it is not numeric
new_auto <- auto[,-9]
round(cor(new_auto, use = "pairwise.complete.obs"), 2)
```

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

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

```
# From the table, we can see that cylinders, displacement, horsepower, weight are all negatively
# related to mpg.Besides, acceleration, model_year, origin are all positively related to mpg.
```

iv. Which relationships might not be linear? (don't worry about linearity for rest of this HW)

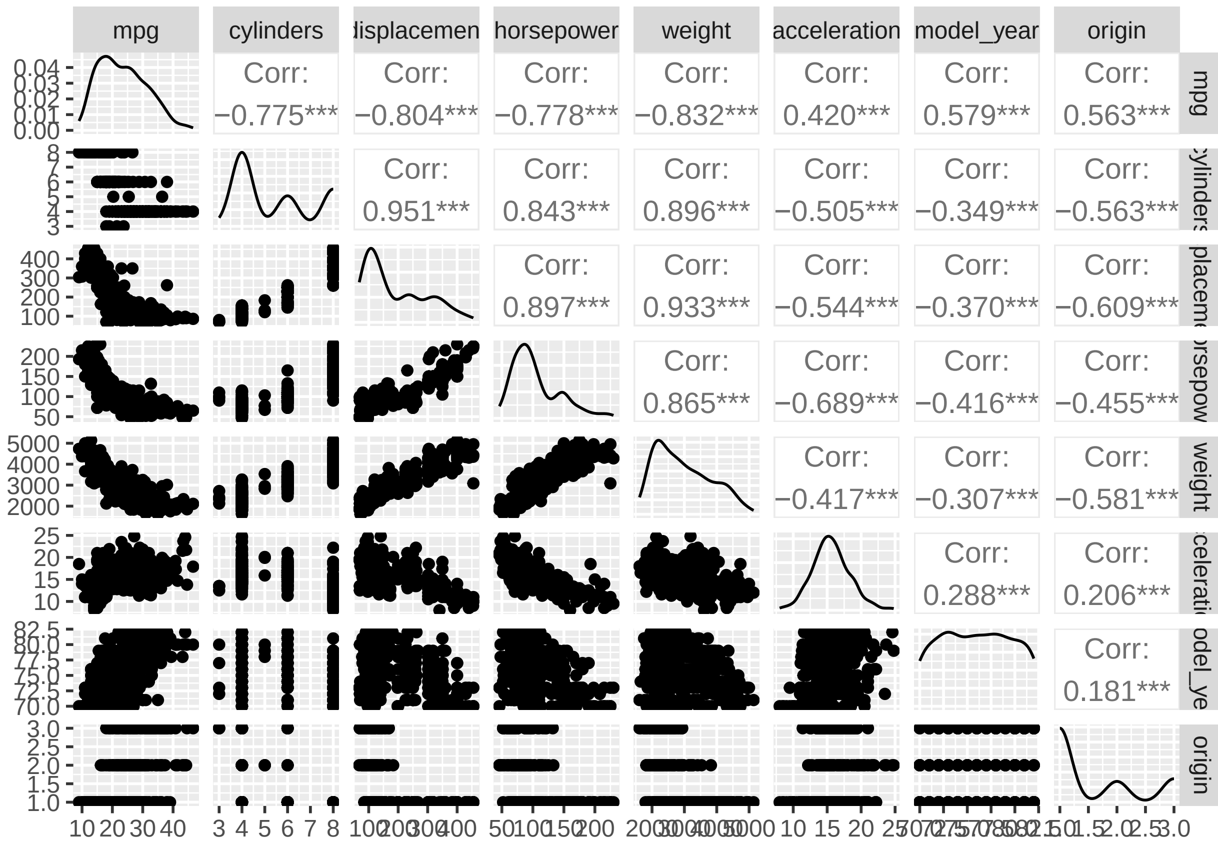
```
# import the library
library(GGally)
```

```
## Loading required package: ggplot2

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

```
# Show the plot
ggpairs(new_auto)
```

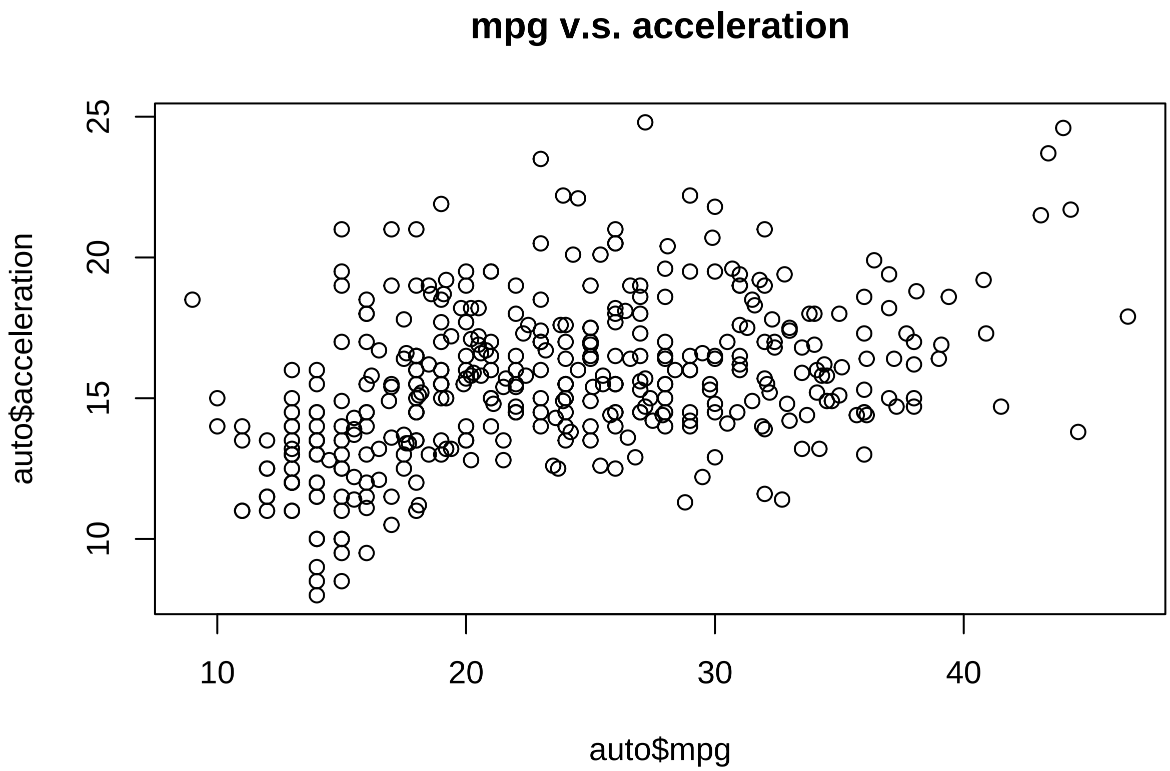
```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 6 rows containing non-finite outside the scale range
## (`stat_density()`).

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 6 rows containing missing values

## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
## Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

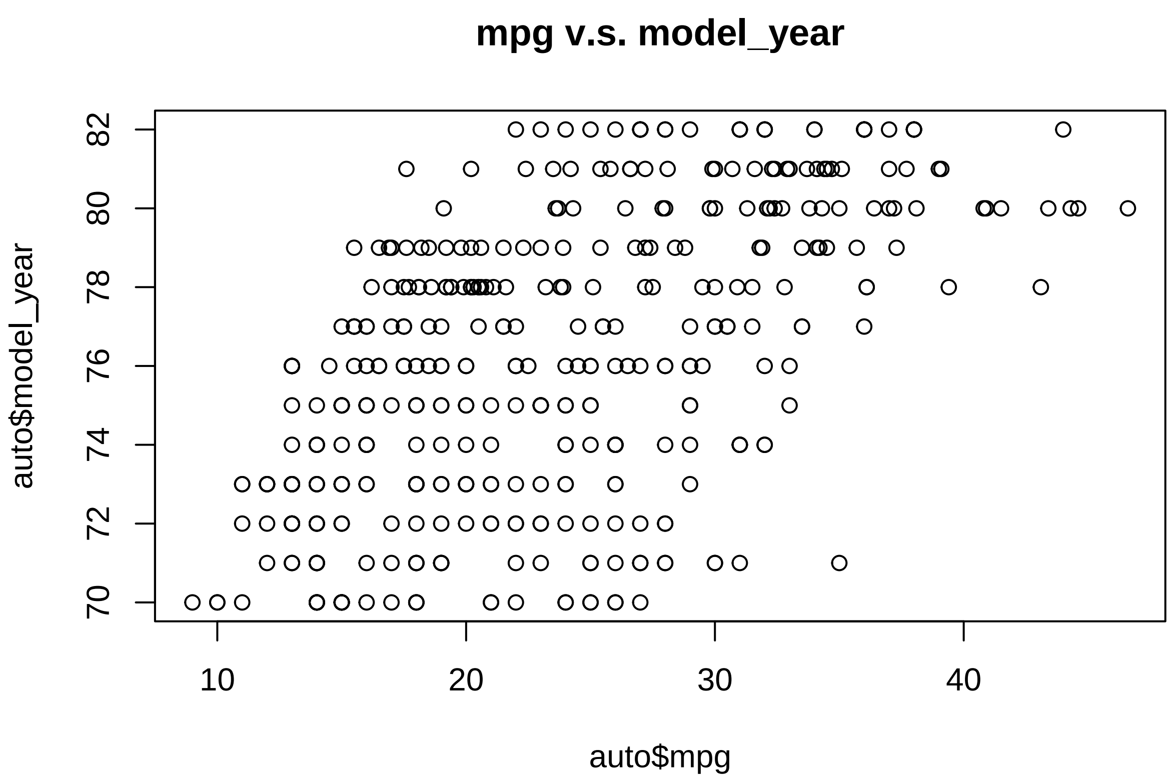



```r
# This table shows that which pairs are linear and which pairs are not linear.
```

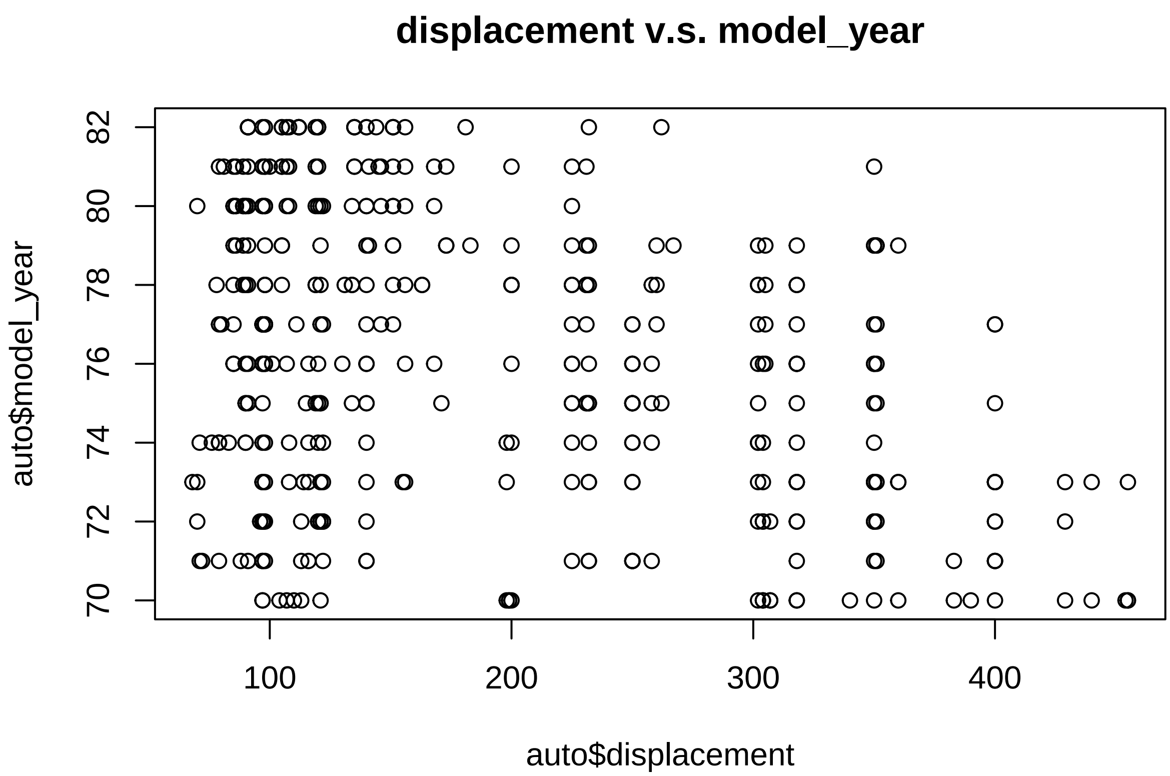
```r
# From the table we create above, we choose some pairs that are not linear to show.
plot(auto$mpg, auto$acceleration, main = "mpg v.s. acceleration")
```



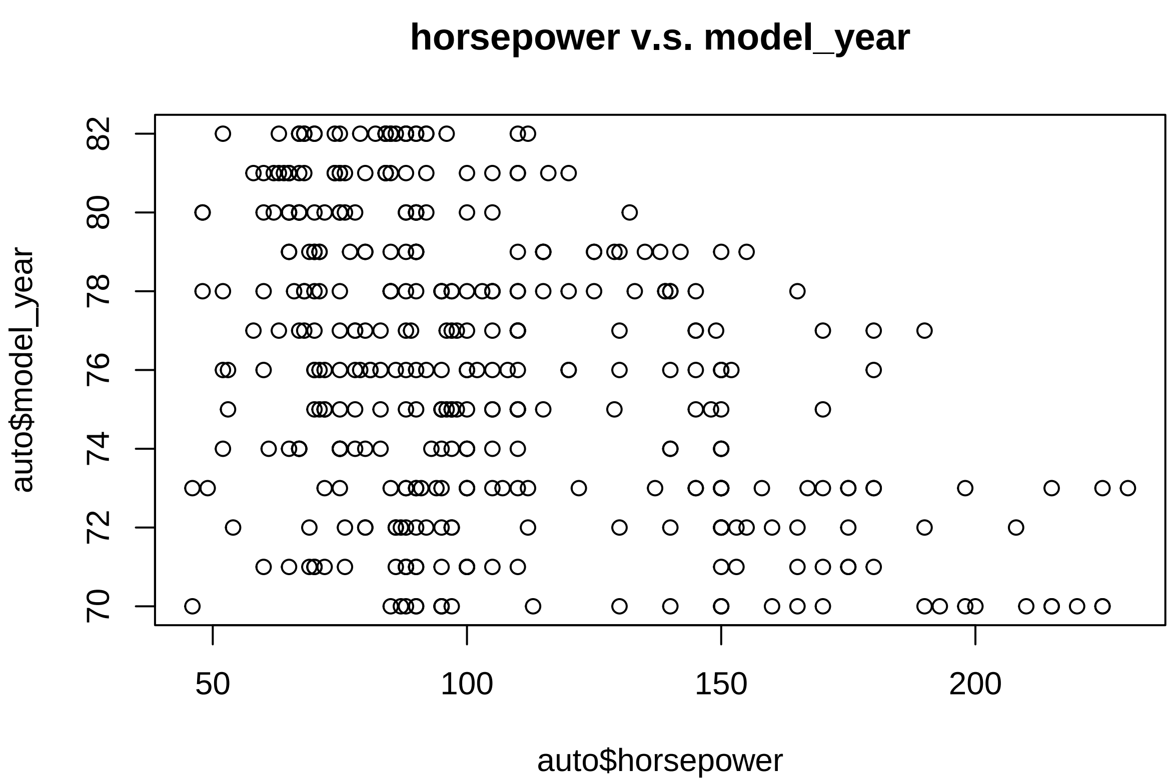

```
plot(auto$mpg, auto$model_year, main = "mpg v.s. model_year")
```



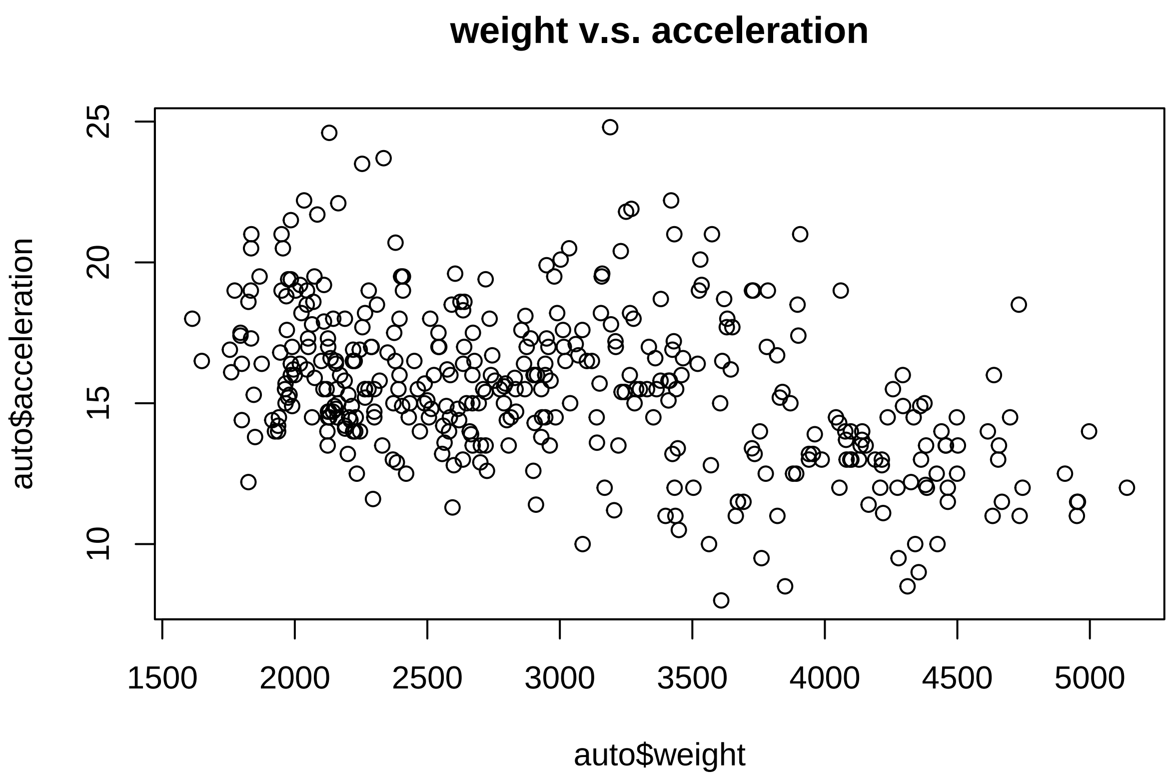
```
plot(auto$displacement, auto$model_year, main = "displacement v.s. model_year")
```



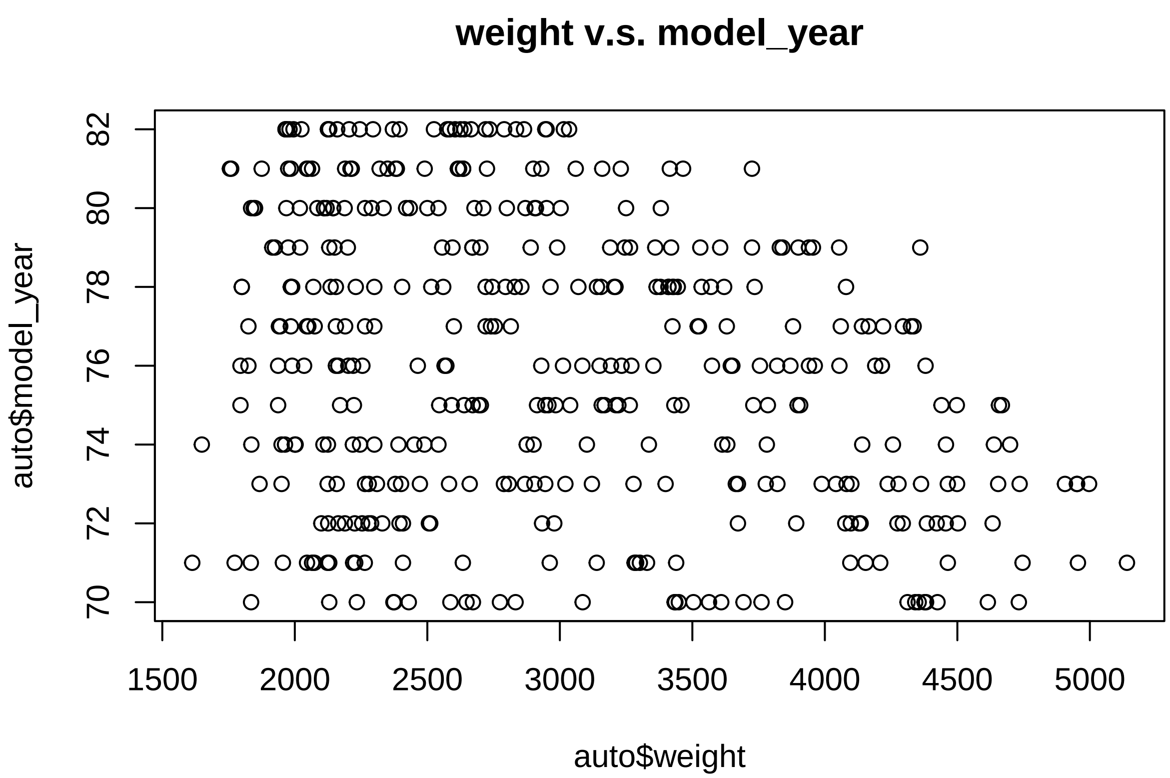

```
plot(auto$horsepower, auto$model_year, main = "horsepower v.s. model_year")
```



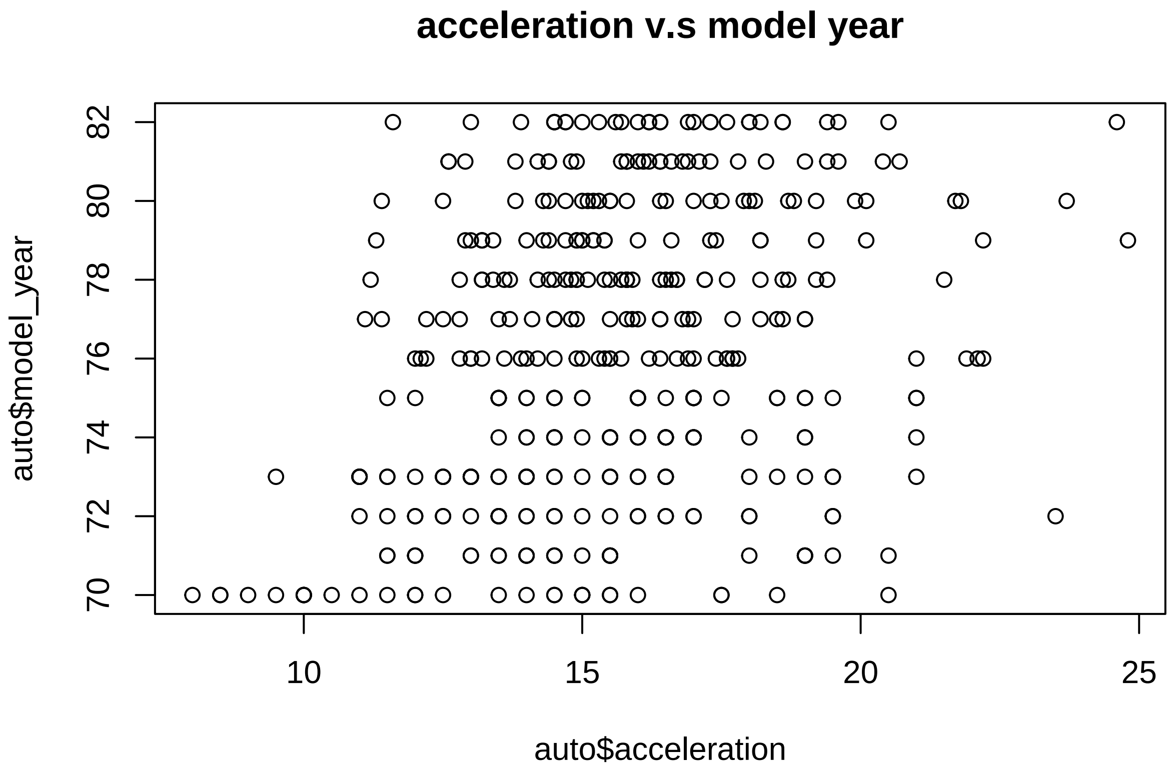
```r
plot(auto$weight, auto$acceleration, main = "weight v.s. acceleration")
```




```r
plot(auto$weight, auto$model_year, main = "weight v.s. model_year")
```



```r
plot(auto$acceleration, auto$model_year, main = "acceleration v.s model year")
```



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

```r
# cylinder and displacement (r = 0.95)
# cylinder and horsepower (r = 0.84)
# cylinder and weight (r = 0.90)
# displacement and horsepower (r = 0.90)
# displacement and weight (r = 0.93)
# horsepower and weight (r = 0.86)
```

b. Let's create a linear regression model where mpg is dependent upon all other suitable variables (Note: origin is categorical with three levels, so use factor(origin) in lm(. . .) to split it into two dummy variables)

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

```r
# Preprocess the data
auto$origin <- factor(auto$origin)
auto <- cbind(auto, model.matrix(~ auto$origin - 1))
names(auto)[10:11] <- c("origin1", "origin2")

# Create model
lm_fit <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration
             + model_year + origin1 + origin2, data = auto)
Table <- summary(lm_fit)
```

```r
# Show the subset of the table that meet 1% significance
sig <- Table$coefficients[row.names(Table$coefficients) != "(Intercept)" &
                          Table$coefficients[, 4] < 0.01, ]
print(sig)
```

```
##                 Estimate   Std. Error    t value     Pr(>|t|)
## displacement  0.023978644 0.0076532690   3.133124 1.862685e-03
## weight       -0.006710384 0.0006551331 -10.242779 6.375633e-22
## model_year    0.777026939 0.0517840867  15.005130 2.332943e-40
## origin1      -2.853228228 0.5527363020  -5.162006 3.933208e-07
```

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!)

```r
# No , because different variables have different scale of units, it is not effective
# that we use these beta estimates that we haven't standardized to determine which
# independent variables are the most effective at increasing mpg.
```

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)

```r
# Since all features are all standardized, these slopes are easier to compare.
auto_std <- data.frame(scale(new_auto))
fit_std <- lm(scale(mpg) ~ ., data = auto_std)
summary(fit_std)
```

```
##
## Call:
## lm(formula = scale(mpg) ~ ., data = auto_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.22701 -0.27591 -0.01496  0.23912  1.67099
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.001748   0.021520  -0.081  0.93532
## cylinders    -0.107374   0.070356  -1.526  0.12780
## displacement  0.265420   0.100256   2.647  0.00844 **
## horsepower   -0.083479   0.067896  -1.230  0.21963
## weight       -0.701446   0.070648  -9.929  < 2e-16 ***
## acceleration  0.028429   0.034875   0.815  0.41548
## model_year    0.355179   0.024115  14.729  < 2e-16 ***
## origin        0.146347   0.028542   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4258 on 384 degrees of freedom
##   (6 observations deleted due to missingness)
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 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?

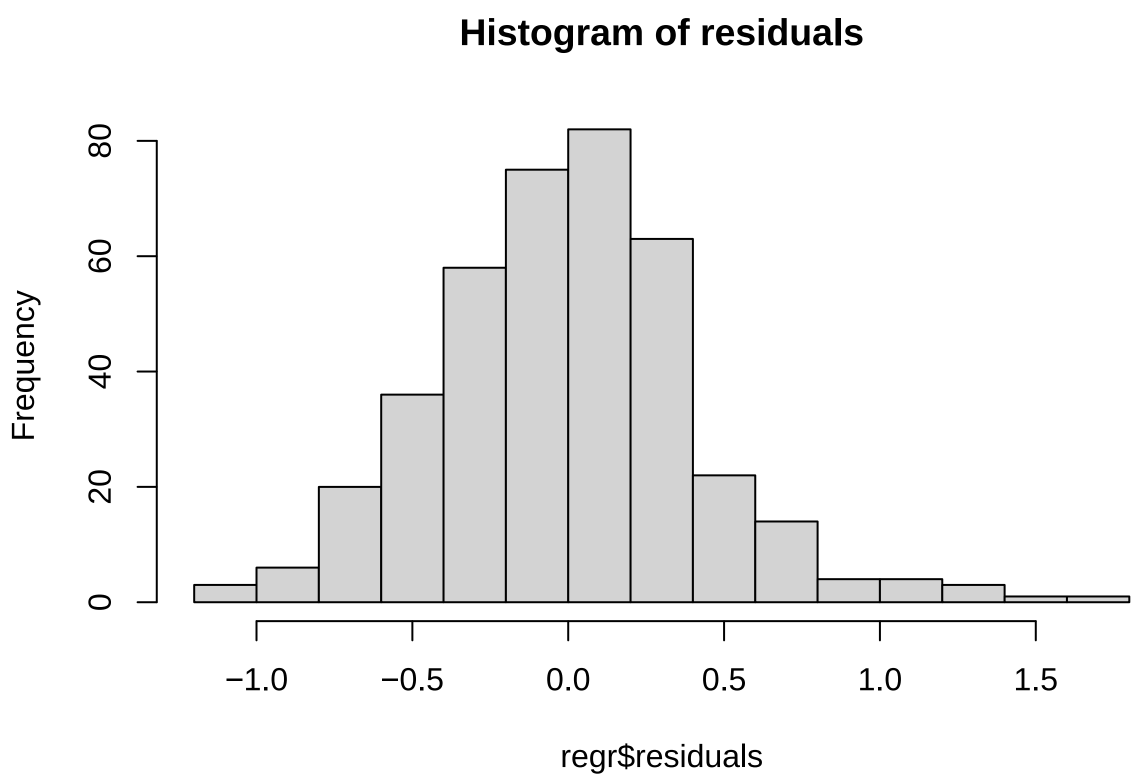
```r
# Extract the list of non-significant independent variables
non_sig_vars <- names(which(summary(fit_std)$coefficients[, 4] >= 0.05))[-1]

# Loop over each non-significant variable and fit a regression model
for (var in non_sig_vars) {
  formula_str <- paste("scale(mpg) ~", var)
  cat("Variable:", var, "\n")
  model <- lm(formula_str, data = auto_std)
  print(summary(model))
  cat("\n")
}
```

```
## Variable: cylinders
##
## Call:
## lm(formula = formula_str, data = auto_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.82455 -0.43297 -0.08288  0.32674  2.29046
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.483e-15  3.169e-02    0.00        1
## cylinders   -7.754e-01  3.173e-02  -24.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6323 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
##
##
## Variable: horsepower
##
## Call:
## lm(formula = formula_str, data = auto_std)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73632 -0.41699 -0.04395  0.35351  2.16531
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.008784   0.031701  -0.277    0.782
## horsepower  -0.777334   0.031742 -24.489   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6277 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
##
##
## Variable: acceleration
##
## Call:
## lm(formula = formula_str, data = auto_std)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3039 -0.7210 -0.1589  0.6087  2.9672
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.025e-17  4.554e-02   0.000        1
## acceleration  4.203e-01  4.560e-02   9.217   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9085 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
```

iii. Plot the distribution 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

```
regr <- lm(mpg ~ . + factor(origin), data = auto_std)
hist(regr$residuals, breaks = 20, main = "Histogram of residuals")
```




```
# By the plot, we think that they are normally distributed and centered around zero.
# But I think that we use normality test to check whether the data is normally distributed
# is better than just look the plot.
```