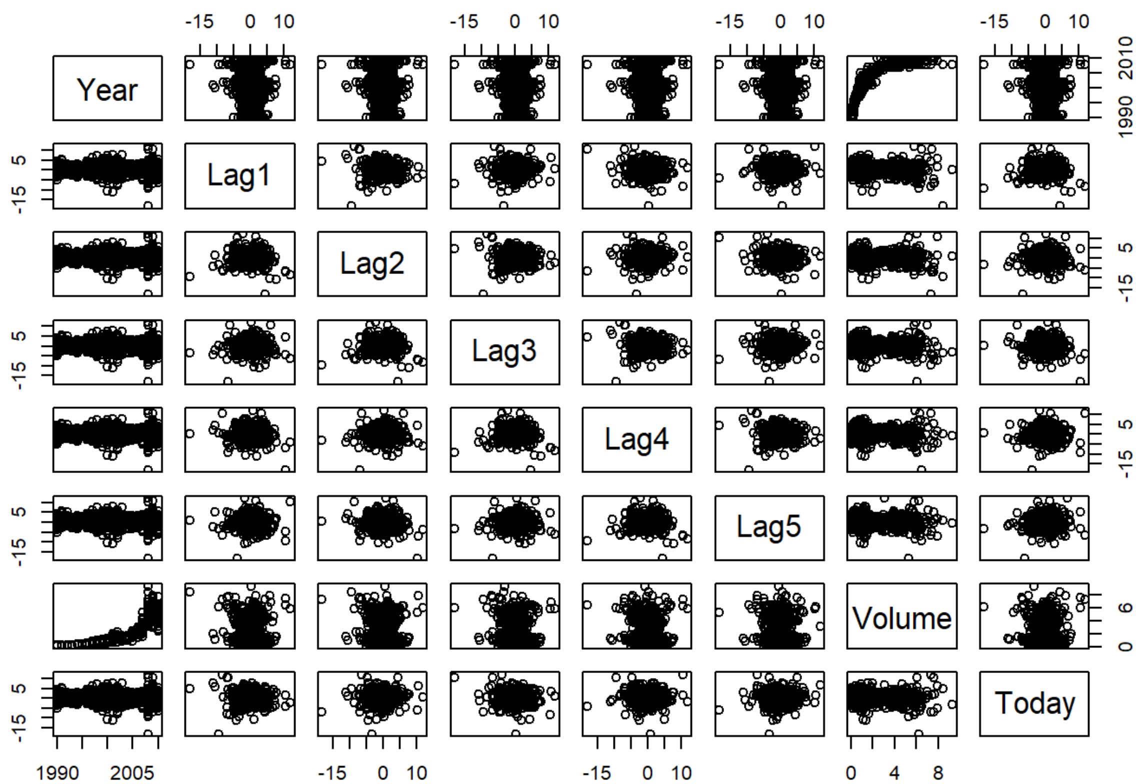
Reihaneh Moghisi
Assignment 2
RMI8300

# Question1-

## a-

```r
pairs(Weekly[ ,-9])
```



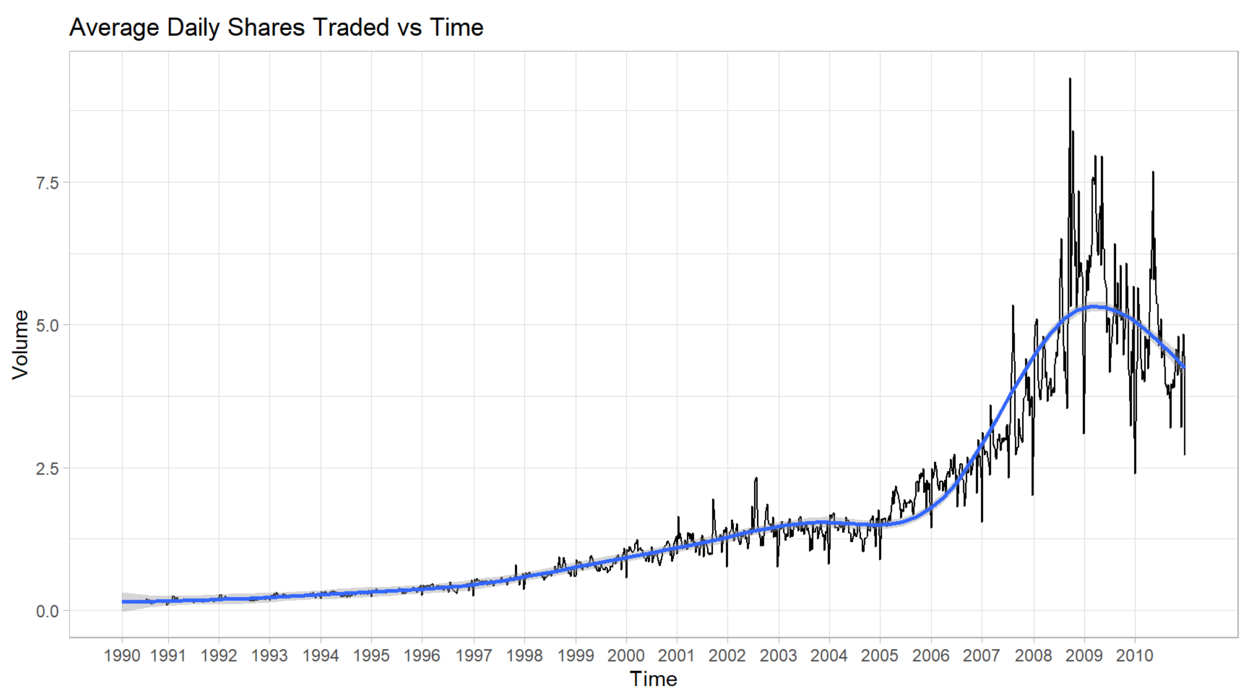

```r
abs(cor(Weekly[ ,-9]))
```

```
##              Year        Lag1       Lag2       Lag3        Lag4        Lag
5
## Year   1.00000000 0.032289274 0.03339001 0.03000649 0.031127923 0.03051910
1
## Lag1   0.03228927 1.000000000 0.07485305 0.05863568 0.071273876 0.00818309
6
## Lag2   0.03339001 0.074853051 1.00000000 0.07572091 0.058381535 0.07249948
2
## Lag3   0.03000649 0.058635682 0.07572091 1.00000000 0.075395865 0.06065717
5
## Lag4   0.03112792 0.071273876 0.05838153 0.07539587 1.000000000 0.07567502
7
```

Reihaneh Moghisi
Assignment 2
RMI8300

```
## Lag5   0.03051910 0.008183096 0.07249948 0.06065717 0.075675027 1.00000000
0
## Volume 0.84194162 0.064951313 0.08551314 0.06928771 0.061074617 0.05851741
4
## Today  0.03245989 0.075031842 0.05916672 0.07124364 0.007825873 0.01101269
8
##            Volume       Today
## Year   0.84194162 0.032459894
## Lag1   0.06495131 0.075031842
## Lag2   0.08551314 0.059166717
## Lag3   0.06928771 0.071243639
## Lag4   0.06107462 0.007825873
## Lag5   0.05851741 0.011012698
## Volume 1.00000000 0.033077783
## Today  0.03307778 1.000000000
```

There are no obvious strong relationships between the `Lag` variables. Although there is some trends of volume over time.




**B –** only Lag 2 is statistically significant.

Reihaneh Moghisi
Assignment 2
RMI8300

```r
glm_dir <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
               data = Weekly,
               family = "binomial")

summary(glm_dir)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
```

```
## Number of Fisher Scoring iterations: 4
```

# C –

The confusion matrix is shown below. The confusion matrix in caret package is used for this. It shows a poor accuracy of 56% . this prediction is based on only training set. It also shows very low specificity which means did not perform well in predicting negative class correctly.

```
predicted <- factor(ifelse(predict(glm_dir, type = "response") < 0.5, "Down",
"Up"))


confusionMatrix(predicted, Weekly$Direction, positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction Down  Up
##       Down   54  48
##       Up    430 557
##
##                Accuracy : 0.5611
##                  95% CI : (0.531, 0.5908)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.369
##
##                   Kappa : 0.035
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9207
##             Specificity : 0.1116
##          Pos Pred Value : 0.5643
##          Neg Pred Value : 0.5294
##              Prevalence : 0.5556
##          Detection Rate : 0.5115
```

```
##    Detection Prevalence : 0.9063
##       Balanced Accuracy : 0.5161
##
##        'Positive' Class : Up
##
```

# D –

The accuracy of model is **0.625**.

`No Information Rate : 0.5865` tells us that the largest class (`Up`) is 58.65% of the `test` observations, this is the baseline. As the `Accuracy : 0.625` > 0.5865 shows a positive outcome of prediction however because the `test` dataset is relatively small so this might not be a meaningful improvement.

The p-value for `P-Value [Acc > NIR] : 0.2439` is a one-sided test to see whether the accuracy is better than the "no information rate. And as this p-value is greater than > 0.05 we conclude that there is no significant evidence that our classifier is better than the baseline strategy.

```
train <- Weekly[Weekly$Year <= 2008, ]
test <- Weekly[Weekly$Year > 2008, ]


glm_dir <- glm(Direction ~ Lag2,
               data = train,
               family = "binomial")


predicted <- factor(ifelse(predict(glm_dir, newdata = test, type = "response"
) < 0.5, "Down", "Up"))


confusionMatrix(predicted, test$Direction, positive = "Up")
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    9  5
```

```
##       Up     34 56
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : Up
##
```

# E – *Repeat (d) using LDA*

```
Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    9  5
##       Up     34 56
##
```

```
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1414
##
##  Mcnemar's Test P-Value : 7.34e-06
##
##             Sensitivity : 0.9180
##             Specificity : 0.2093
##          Pos Pred Value : 0.6222
##          Neg Pred Value : 0.6429
##              Prevalence : 0.5865
##          Detection Rate : 0.5385
##    Detection Prevalence : 0.8654
##       Balanced Accuracy : 0.5637
##
##        'Positive' Class : Up
##
```

The Accuracy of model is **0.625**. Same as above, the `P-Value [Acc > NIR] : 0.2439` > 0.05, so although the accuracy of the classifier is `0.625` > `0.5865`, with the same logic that the `test` sample size is not large enough and therefore the increase in accuracy is not statistically significant enough over the baseline.

# F- *Repeat using QDA.*

```
qda_dir <- qda(Direction ~ Lag2, data = train)

predicted_qda <- predict(qda_dir, newdata = test)
```

Reihaneh Moghisi
Assignment 2
RMI8300

```
confusionMatrix(data = predicted_qda$class,
                reference = test$Direction,
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    0  0
##       Up     43 61
##
##                Accuracy : 0.5865
##                  95% CI : (0.4858, 0.6823)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.5419
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.504e-10
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5865
##          Neg Pred Value :    NaN
##              Prevalence : 0.5865
##          Detection Rate : 0.5865
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Up
##
```

Here we get an Accuracy of **0.5865**. with no-information rate of 58% and p-value 0.54. the accuracy and no-information rate are equal and specificity 0 and sensitivity 1 illustrate that this model follows naïve classifier.

Reihaneh Moghisi
Assignment 2
RMI8300

# G- Repeat using KNN

the confusion matrix:

```
confusionMatrix(data = predicted_knn,
                reference = test$Direction,
                positive = "Up")
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   21 30
##       Up     22 31
##
##                Accuracy : 0.5
##                  95% CI : (0.4003, 0.5997)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9700
##
##                   Kappa : -0.0033
##
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.5082
##             Specificity : 0.4884
##          Pos Pred Value : 0.5849
##          Neg Pred Value : 0.4118
##              Prevalence : 0.5865
##          Detection Rate : 0.2981
##    Detection Prevalence : 0.5096
##       Balanced Accuracy : 0.4983
##
```

```
##        'Positive' Class : Up
##
```

Here we get an accuracy of **0.5**, which is again worse than the baseline.

# H- Which of these methods appears to provide the best results on this data?

Considering the accuracy as the measurement of performance: LDA & Logistic Regression get the same `test` accuracy of **0.625**, so these two are best classifiers in this case.

# I – combined predictors, interactions, transformations

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 5,
                     repeats = 5)
set.seed(111)
knn_train <- train(y = train$Direction,
                   x = train[ ,-8],
                   method = "knn",
                   metric = "Accuracy",
                   preProcess = c("center", "scale"),
                   tuneGrid = expand.grid(k = seq(1, 50, 2)),
                   trControl = ctrl)


caret::varImp(knn_train)
```

```
## ROC curve variable importance
##
##        Importance
## Lag1      100.000
## Lag2       77.256
## Lag5       64.309
```

```
## Year        45.659
## Volume      43.735
## Week        42.513
## Lag4         4.578
## Lag3         0.000
```

```
knn_train
## k-Nearest Neighbors
##
## 985 samples
##   8 predictor
##   2 classes: 'Down', 'Up'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 788, 788, 788, 788, 788, 787, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.4947996  -0.020756148
##    3  0.5232477   0.031373990
##    5  0.5230292   0.028769372
##    7  0.5372435   0.053353378
##    9  0.5372415   0.049612758
##   11  0.5343834   0.040819116
##   13  0.5346081   0.037598994
##   15  0.5368437   0.040363712
##   17  0.5317675   0.026161027
##   19  0.5301555   0.021608137
##   21  0.5370622   0.033488872
##   23  0.5330064   0.024861594
##   25  0.5401182   0.036317004
```

```
##   27  0.5376693   0.029998090
##   29  0.5321890   0.015868622
##   31  0.5338310   0.018687256
##   33  0.5360563   0.021431639
##   35  0.5322138   0.012669022
##   37  0.5283538   0.002535516
##   39  0.5330239   0.011432961
##   41  0.5289589   0.002400751
##   43  0.5295618   0.003081904
##   45  0.5299700   0.003587628
##   47  0.5303854   0.003745293
##   49  0.5320056   0.005552184
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 25.
```

Evaluating the performance of this new model on `test`:

```
knn_pred <- predict(knn_train, newdata = test)

confusionMatrix(data = knn_pred,
                reference = test$Direction,
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   19 20
##       Up     24 41
##
##                Accuracy : 0.5769
##                  95% CI : (0.4761, 0.6732)
##     No Information Rate : 0.5865
```

```
##     P-Value [Acc > NIR] : 0.6193
##
##                   Kappa : 0.1156
##
##  Mcnemar's Test P-Value : 0.6511
##
##             Sensitivity : 0.6721
##             Specificity : 0.4419
##          Pos Pred Value : 0.6308
##          Neg Pred Value : 0.4872
##              Prevalence : 0.5865
##          Detection Rate : 0.3942
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.5570
##
##        'Positive' Class : Up
##
```

Although the accuracy increased from k=1 but we are still below the baseline.

Another combination is using the interaction term of Lag1 and Lag2. The results below shows still very poor accuracy of 55% over baseline.

```
glm(formula = Direction ~ Lag1 * Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = "binomial", data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6176  -1.2590   0.9893   1.0852   1.5367

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.268646   0.086017   3.123  0.00179 **
Lag1        -0.036253   0.028227  -1.284  0.19903
Lag2         0.057800   0.027025   2.139  0.03246 *
Lag3        -0.017376   0.026826  -0.648  0.51716
Lag4        -0.029814   0.026775  -1.113  0.26550
Lag5        -0.014385   0.026380  -0.545  0.58553
Volume      -0.023317   0.036951  -0.631  0.52802
Lag1:Lag2    0.003475   0.006891   0.504  0.61410
---

(Dispersion parameter for binomial family taken to be 1)
```

Reihaneh Moghisi
Assignment 2
RMI8300

```
    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.1  on 1081  degrees of freedom
AIC: 1502.1

Number of Fisher Scoring iterations: 4

Confusion Matrix and Statistics

          Reference
Prediction Down  Up
      Down   51  48
      Up    433 557

               Accuracy : 0.5583
                 95% CI : (0.5282, 0.5881)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 0.4399

                  Kappa : 0.0283

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9207
            Specificity : 0.1054
         Pos Pred Value : 0.5626
         Neg Pred Value : 0.5152
             Prevalence : 0.5556
         Detection Rate : 0.5115
   Detection Prevalence : 0.9091
      Balanced Accuracy : 0.5130

       'Positive' Class : Up
```

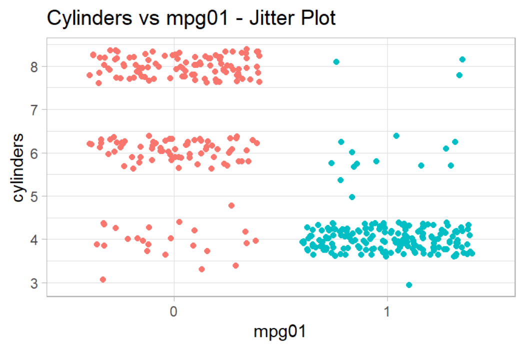
# Question 2-

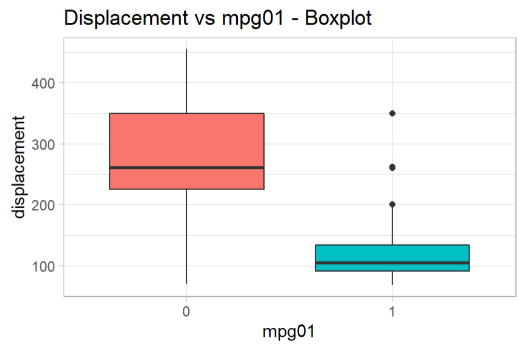
# A –

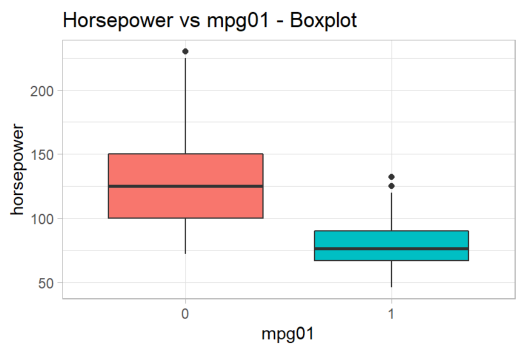
```
Auto$mpg01 <- factor(as.numeric(Auto$mpg > median(Auto$mpg)))
table(Auto$mpg01)
##
##   0   1
## 196 196
```

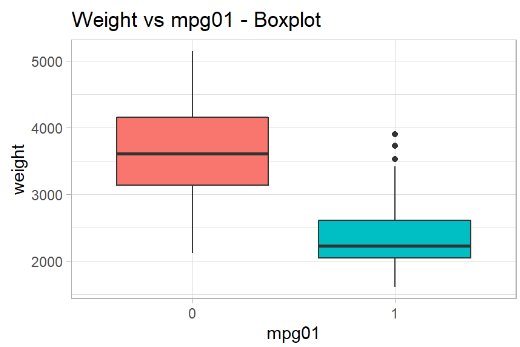
Reihaneh Moghisi
Assignment 2
RMI8300

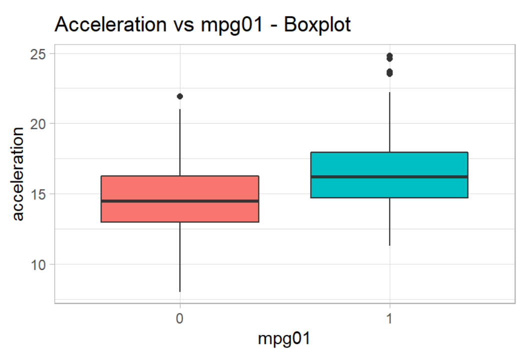
**B** – Although the exact associations cant be extracted from graphical plots but we can poor;y conclude that `cylinders`, `displacement` & `weight` are the three strongest predictors of `mpg01`. Horsepower, year and acceleration look weaker predictors of mpg01.

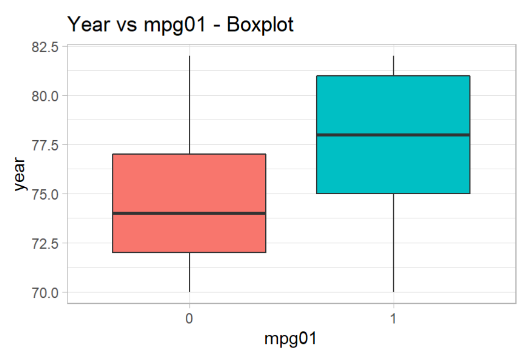


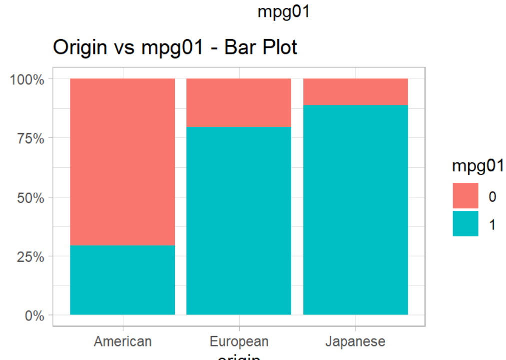



Reihaneh Moghisi
Assignment 2
RMI8300

# C-

```
set.seed(444)
index <- createDataPartition(y = Auto$mpg01, p = 0.5, list = F)

train <- Auto[index, ]
test <- Auto[-index, ]

nrow(train) / nrow(Auto)
## [1] 0.5
```

# D-

We use the top 4 strongest predictors I identified visually (`cylinders`, `displacement`, `weight` & `horsepower`).
It shows that the model is doing great in predicting the dependent variable much better than baseline.
The test error is 0.12244

```
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1)

lda_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "lda",
                 trControl = ctrl)
```

`train` (cross-validation) error:

```
1 - lda_mpg$results$Accuracy # cv error
## [1] 0.09125731
```

`test` error:

Reihaneh Moghisi
Assignment 2
RMI8300

```
predicted_lda <- predict(lda_mpg, newdata = test, type = "raw") # as opposed
to type = "prob"


mean(predicted_lda != test$mpg01)
## [1] 0.122449
```

# E-

In this case, QDA appears to perform better than LDA with respect to `test` error, and slightly better in terms of CV error.

```
set.seed(2)


qda_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "qda",
                 trControl = ctrl)
```

`train` (cross-validation) error:

```
1 - qda_mpg$results$Accuracy
## [1] 0.08991228
```

`test` error:

```
predicted_qda <- predict(qda_mpg, newdata = test, type = "raw")


mean(predicted_qda != test$mpg01)
## [1] 0.1173469
```

# F-

We can see here that Logistic Regression performs better than LDA & QDA with respect to CV error & `test` error.

```
set.seed(3)
```

Reihaneh Moghisi
Assignment 2
RMI8300

```
log_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "glm",
                 family = "binomial",
                 trControl = ctrl)
```

`train` (cross-validation) error:

```
1 - log_mpg$results$Accuracy
## [1] 0.1084405
```

`test` error:

```
predicted_log <- predict(log_mpg, newdata = test, type = "raw")

mean(predicted_log != test$mpg01)
## [1] 0.1173469
```

**G-** The optimal value chosen was K = 7, although this model performed close to Logistic Regression in CV error, and slightly worse in `test` error.

It is observed that all classifiers with larger values for K (from 10 to 73) get the same cross-validation accuracy scores. This can be due to the fact that those KNN models make the same prediction.

```
set.seed(4)

knn_mpg <- train(mpg01 ~ cylinders + displacement + weight + horsepower,
                 data = train,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 trControl = ctrl,
                 tuneGrid = expand.grid(k = seq(1, 85, 3)))

knn_mpg
```

Reihaneh Moghisi
Assignment 2
RMI8300

```
## k-Nearest Neighbors
##
## 196 samples
##   4 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (4), scaled (4)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 176, 177, 176, 176, 176, 176, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8779532  0.7554422
##    4  0.8999805  0.7995964
##    7  0.9033138  0.8062631
##   10  0.9033138  0.8062631
##   13  0.9033138  0.8062631
##   16  0.9033138  0.8062631
##   19  0.9033138  0.8062631
##   22  0.9033138  0.8062631
##   25  0.9033138  0.8062631
##   28  0.9033138  0.8062631
##   31  0.9033138  0.8062631
##   34  0.9033138  0.8062631
##   37  0.9033138  0.8062631
##   40  0.9033138  0.8062631
##   43  0.9033138  0.8062631
##   46  0.9033138  0.8062631
##   49  0.9033138  0.8062631
##   52  0.9033138  0.8062631
##   55  0.9033138  0.8062631
##   58  0.9033138  0.8062631
##   61  0.9033138  0.8062631
```

```
##    64  0.9049805  0.8095964
##    67  0.9049805  0.8095964
##    70  0.9066472  0.8129297
##    73  0.9100682  0.8197621
##    76  0.9083138  0.8162631
##    79  0.9084016  0.8164288
##    82  0.9067349  0.8130955
##    85  0.9034016  0.8065072
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 73.
```

`train` (cross-validation) error:

```
## [1] 0.08993177
```

`test` error:

```
## [1] 0.1122449
```

# Question 3:

# A –

```
Power <- function() {
  2^3
}


Power()
## [1] 8
```

# B –

```
Power2 <- function(x, a) {
  x^a
}
```

Reihaneh Moghisi
Assignment 2
RMI8300

```r
Power2(3, 8)
## [1] 6561
```

# C –

```r
Power2(10, 3)
## [1] 1000
```

8 17:

```r
Power2(8, 17)
## [1] 2.2518e+15
```

131 3:

```r
Power2(131, 3)
## [1] 2248091
```

# D-

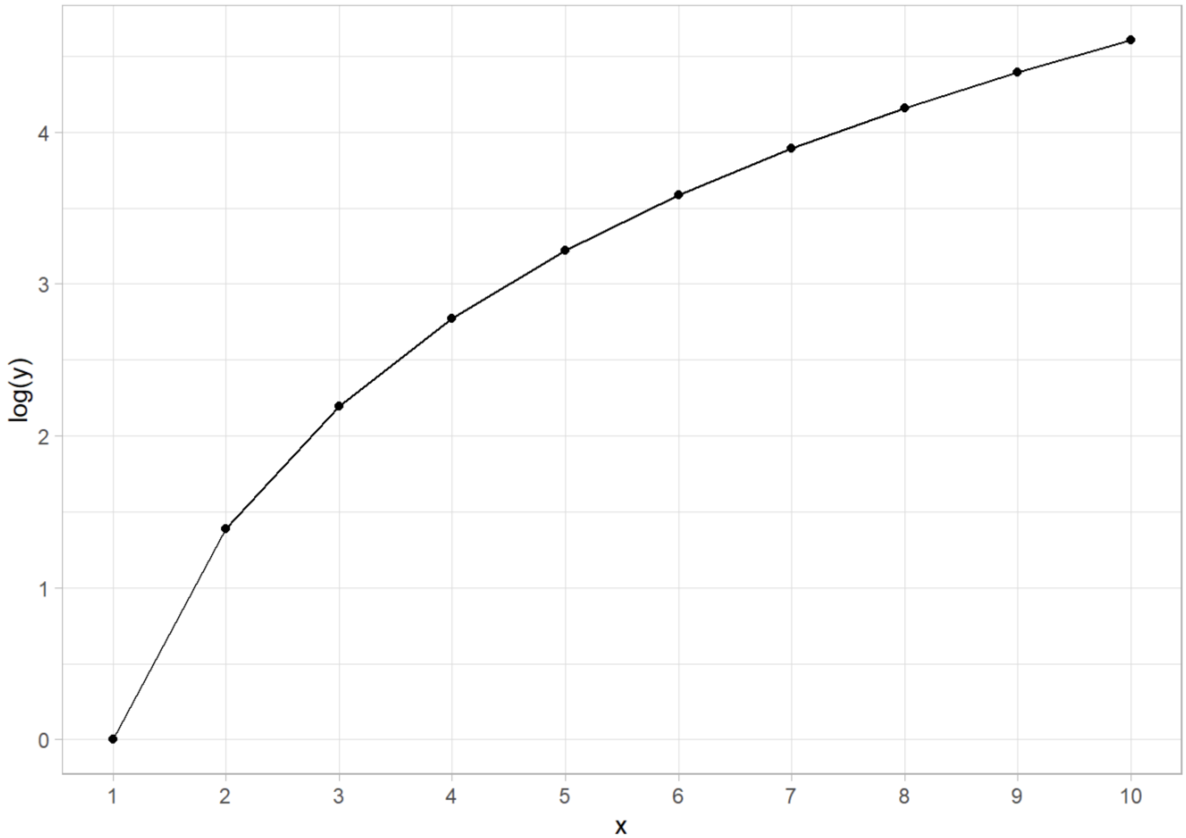
```r
Power3 <- function(x, a) {
  result <- x^a
  result
}

Power3(3, 8)
## [1] 6561
```

# E –

```r
data.frame(x = 1:10, y = Power3(1:10, 2)) %>%
  ggplot(aes(x = x, y = log(y))) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10, minor_breaks = NULL)
```

Reihaneh Moghisi
Assignment 2
RMI8300



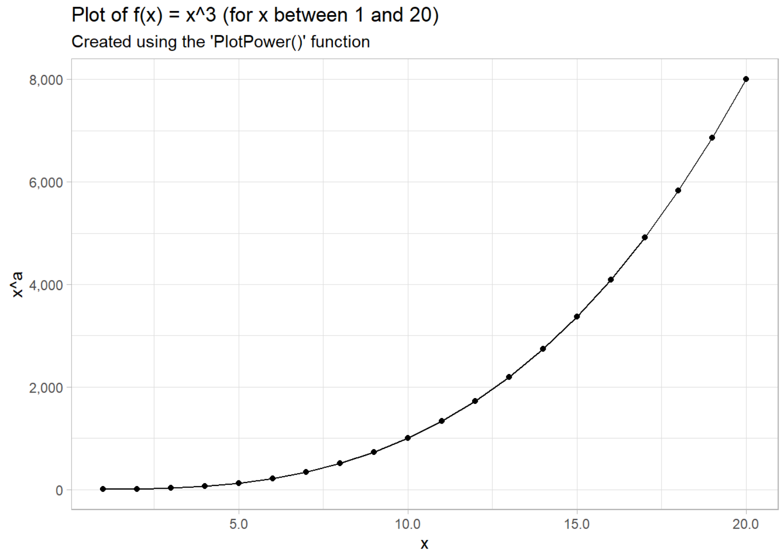

# F-

```
PlotPower <- function(x, a, col = "black") {
  ggplot(mapping = aes(x = x, y = x^a)) +
    geom_point(col = col) +
    geom_line(col = col) +
    scale_x_continuous(labels = scales::comma_format()) +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_light() +
    labs(title = paste0("Plot of f(x) = x^", as.character(a), " (for x betwee
n ", min(x), " and ", max(x), ")"),
         subtitle = "Created using the 'PlotPower()' function")
}
```

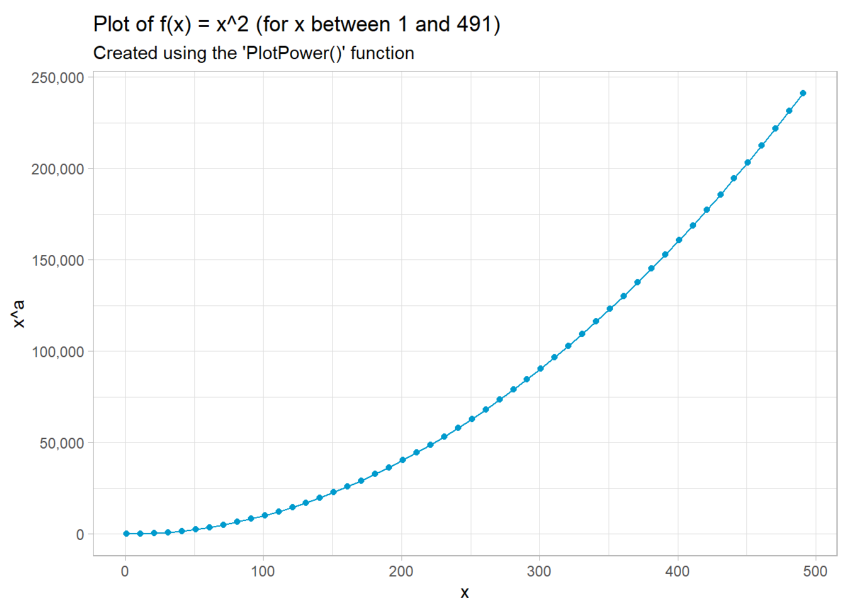
I test the `PlotPower()` function for two examples below. I also added a`col` argument to change the colour, and made the title auto-generate to display the function and range.

Reihaneh Moghisi
Assignment 2
RMI8300

```
PlotPower(1:20, 3)
```




```
PlotPower(x = seq(1, 500, 10),
          a = 2,
          col = "deepskyblue3")
```




# Question 4:

Reihaneh Moghisi
Assignment 2
RMI8300
First, I’ll overwrite the crime rate variable (`crim`) with a binary factor variable, which takes the value `1` when `crim` is above the median, else `0`.

Then I start by fitting all 4 model types, using all predictors each time.

Instead of `train`/`test` split, I used using cross-validation to evaluate performance of accuracy (10-fold, repeated 5 times).

From the results we can see that KNN performed the best. Losgistic regression was the second highest accuracy followed by QDA and LDA .

Changing the crime variable:

```
Boston$crim <- factor(ifelse(Boston$crim > median(Boston$crim), 1, 0))
```

Cross validation:

```
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 5)
```

**Logistic Regression:**

```
set.seed(1)
log_crim <- train(crim ~ .,
                  data = Boston,
                  method = "glm",
                  family = "binomial",
                  trControl = ctrl)


log_crim
## Generalized Linear Model
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 455, 456, 454, 456, 456, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9028549  0.8057423
```

### LDA:

```
set.seed(2)
lda_crim <- train(crim ~ .,
                  data = Boston,
                  method = "lda",
                  trControl = ctrl)

lda_crim
```

```
## Linear Discriminant Analysis
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 455, 456, 456, 455, 454, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8498998  0.6996644
```

Reihaneh Moghisi
Assignment 2
RMI8300

## QDA:

```
set.seed(3)
qda_crim <- train(crim ~ .,
                  data = Boston,
                  method = "qda",
                  trControl = ctrl)


qda_crim
## Quadratic Discriminant Analysis
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 456, 456, 455, 455, 455, 456, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8894561  0.7788975
```

## KNN:

```
set.seed(4)
knn_crim <- train(crim ~ .,
                  data = Boston,
                  method = "knn",
                  preProcess = c("center", "scale"),
```

```
                  trControl = ctrl,
                  tuneGrid = expand.grid(k = seq(1, 20, 2)))

knn_crim
```

```
## k-Nearest Neighbors
##
## 506 samples
##  13 predictor
##   2 classes: '0', '1'
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 455, 456, 456, 455, 455, 456, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9145970  0.8292052
##    3  0.9181104  0.8362439
##    5  0.9153345  0.8306748
##    7  0.9047128  0.8094000
##    9  0.8779164  0.7557756
##   11  0.8664404  0.7327928
##   13  0.8664323  0.7327720
##   15  0.8703940  0.7407136
##   17  0.8684169  0.7367859
##   19  0.8648404  0.7296460
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```