Linear Models with Model Selection

Seungheon Han

1. generate simulated data, and then use this data to perform model selection.

(a) Use the `rnorm` function to generate a predictor $\boldsymbol{X}$ of length $n = 100$, as well as a noise vector $\boldsymbol{\epsilon}$ of length $n = 100$.

```
set.seed(123)
X <- rnorm(100)
error <- rnorm(100)
```

(b) Generate a response vector $\boldsymbol{Y}$ of length $n = 100$ according to the model

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon,$$

where $\beta_0 = 3$, $\beta_1 = 2$, $\beta_2 = -3$, $\beta_3 = 0.3$.

```
beta <- c(3, 2, -3, 0.3)
X.mat <- matrix(c(rep(1, 100), X, X^2, X^3), ncol = 4)
Y <- X.mat%*%beta + error
```

(c) perform best subset selection in order to choose the best model containing the predictors $(X, X^2, \cdots, X^{10})$.

```
predictor <- matrix(c(X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10), ncol = 10)
simul.df <- data.frame(Y, predictor)

library(leaps)
reg.fit <- regsubsets(Y~., data = simul.df, nvmax = 10)
reg.summary <- summary(reg.fit)
print(reg.summary)
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = simul.df, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
```

```
## X10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" " " " " " " " " "*" " " "*" " "
## 5  ( 1 )  "*" "*" " " " " " "*" "*" " " " " "*" " "
## 6  ( 1 )  "*" " " "*" "*" " " "*" " " "*" " " "*"
## 7  ( 1 )  "*" " " " " " " "*" "*" "*" " " "*" "*" "*"
## 8  ( 1 )  "*" "*" " " " " "*" "*" "*" " " "*" "*" "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" " " "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
# Cp
reg.summary$cp
```

```
##  [1] 608.107369  22.585603   2.184588   2.713288   4.308056   3.881940
##  [7]   5.384989   7.076427   9.022299  11.000000
```

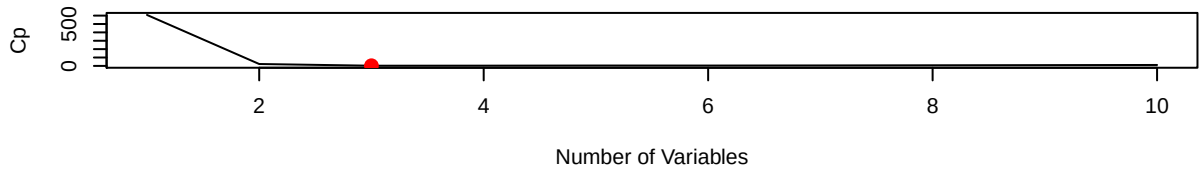
```
# BIC
reg.summary$bic
```

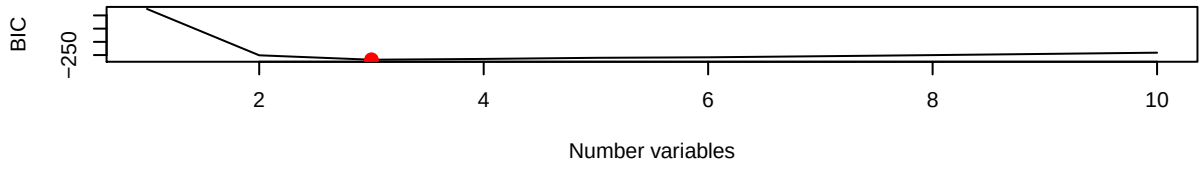
```
##  [1]  -75.61892 -250.84425 -267.57600 -264.54531 -260.37818 -258.43645
##  [7] -254.38570 -250.12634 -245.58195 -241.00183
```

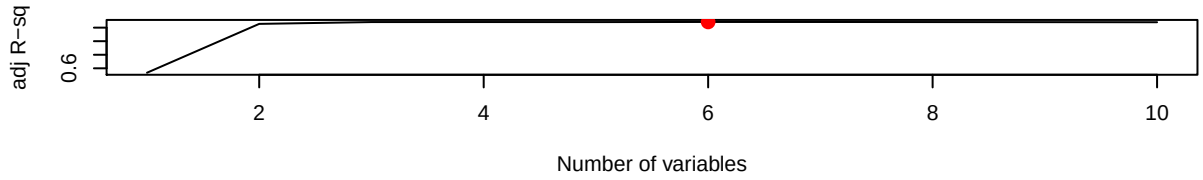
```
# adj R-sq
reg.summary$adjr2
```

```
##  [1] 0.5674858 0.9276463 0.9409396 0.9412503 0.9408848 0.9418196 0.9415123
##  [8] 0.9410737 0.9404552 0.9398012
```

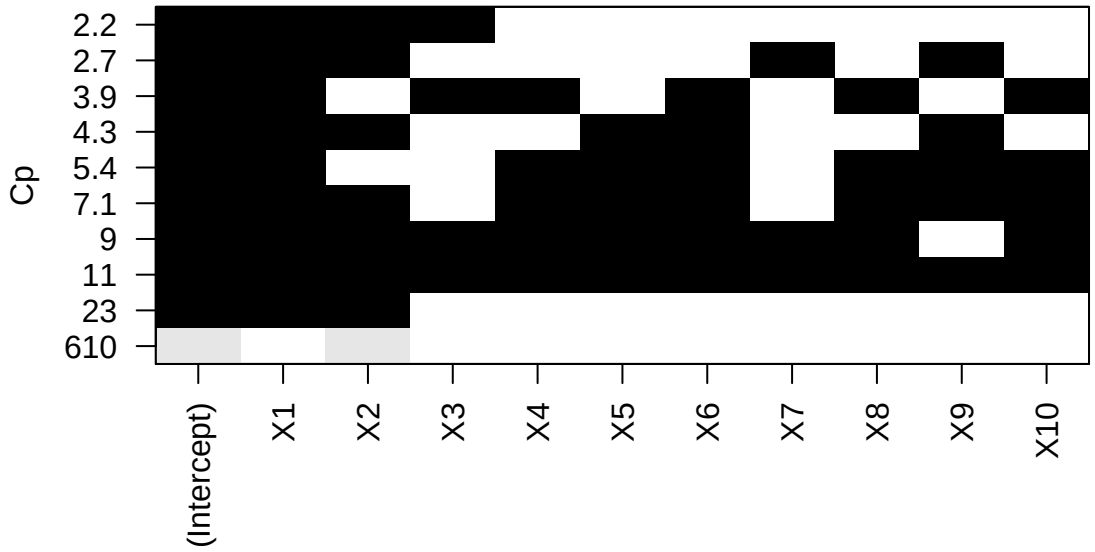
```
# Plot
par(mfrow=c(3,1))

# Cp -> model with 3 features
plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
best.min.cp <- which.min(reg.summary$cp)
points(best.min.cp, reg.summary$cp[best.min.cp], col = "red", cex=2, pch=20)

# BIC -> model with 3 features
plot(reg.summary$bic, xlab = "Number variables", ylab = "BIC", type = "l")
best.min.bic <- which.min(reg.summary$bic)
points(best.min.bic, reg.summary$bic[best.min.bic], col = "red", cex=2, pch=20)

# adj R-sq -> model with 6 features
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "adj R-sq", type = "l")
best.max.adjr2 <- which.max(reg.summary$adjr2)
points(best.max.adjr2, reg.summary$adjr2[best.max.adjr2], col = "red", cex=2, pch=20)
```








```
# Selected Predictors

plot(reg.fit, scale = "Cp")
```




```
plot(reg.fit, scale = "bic")
```


```
plot(reg.fit, scale = "adjr2")
```


```
# Coefficients

# Best subset based on Cp
coef(reg.fit, best.min.cp)
```

```
## (Intercept)          X1          X2          X3
##   2.9703939   1.9204462  -3.0915431   0.3204363
```

```
# Best subset based on BIC
coef(reg.fit, best.min.bic)
```

```
## (Intercept)          X1          X2          X3
##   2.9703939   1.9204462  -3.0915431   0.3204363
```

```r
# Best subset based on adj R-sq
coef(reg.fit, best.max.adjr2)
```

```
## (Intercept)          X1          X3          X4          X6          X8
##  2.71346671  1.87575532  0.36381032 -5.11742226  2.80269053 -0.61718295
##         X10
##  0.04740728
```

(d) Repeating (c), using forward stepwise selection and also using backwards stepwise selection.

```r
# Forward Stepwise Selectioin
regfit.fwd <- regsubsets(Y~., data = simul.df, nvmax = 10, method = "forward")
fwd.summary <- summary(regfit.fwd)
print(fwd.summary)
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = simul.df, nvmax = 10, method = "forward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 )  " " "*" " " " " " " " " " " " " " " " "
## 2  ( 1 )  "*" "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  "*" "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  "*" "*" "*" " " " " " " "*" " " " " " "
## 5  ( 1 )  "*" "*" "*" " " " " " " "*" " " " " "*" " "
## 6  ( 1 )  "*" "*" "*" " " "*" "*" " " " " " "*" " "
## 7  ( 1 )  "*" "*" "*" " " "*" "*" " " " " " "*" "*"
## 8  ( 1 )  "*" "*" "*" "*" "*" "*" " " " " "*" "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" " " "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
# Cp
fwd.summary$cp
```

```
##  [1] 608.107369  22.585603   2.184588   2.928712   4.852545   6.297576
##  [7]   8.025831   8.240341   9.057428  11.000000
```

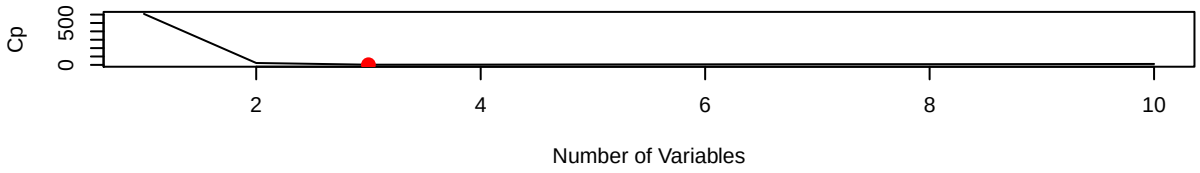
```r
# BIC
fwd.summary$bic
```

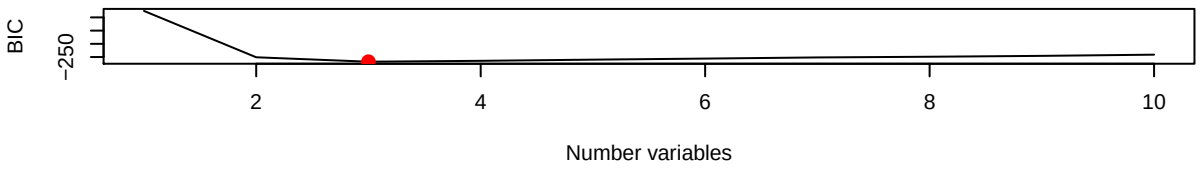
```
##  [1]  -75.61892 -250.84425 -267.57600 -264.31322 -259.79005 -255.78436
##  [7] -251.47405 -248.82815 -245.54250 -241.00183
```

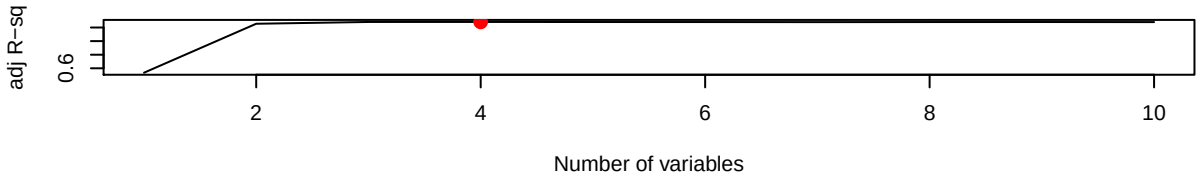
```r
# adj R-sq
fwd.summary$adjr2
```

```
##  [1] 0.5674858 0.9276463 0.9409396 0.9411138 0.9405361 0.9402559 0.9397843
##  [8] 0.9403038 0.9404317 0.9398012
```

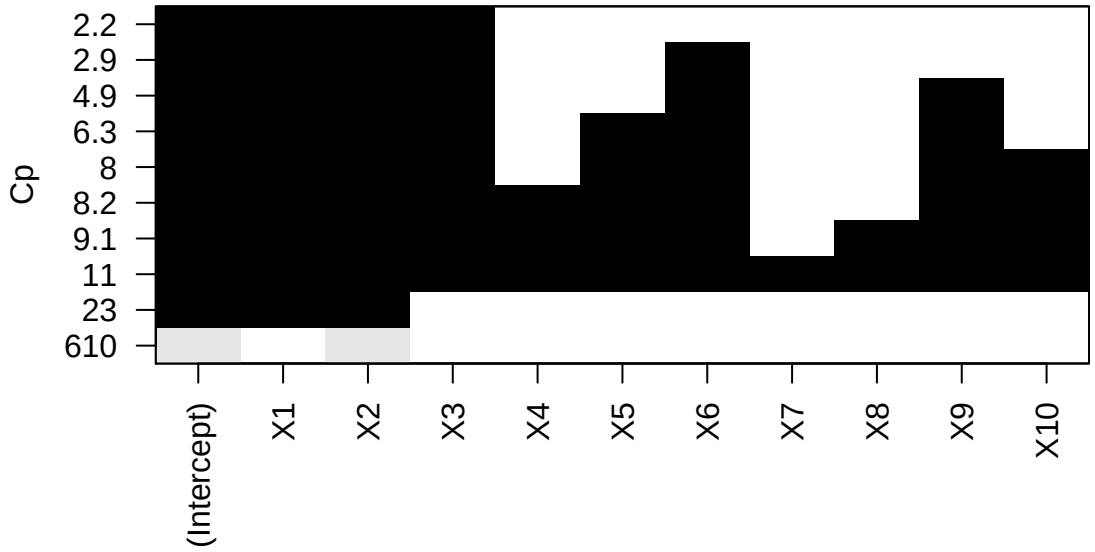
```r
# Plot (Forward Stepwise)
par(mfrow=c(3,1))

# Cp -> model with 3 features
plot(fwd.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
fwd.min.cp <- which.min(fwd.summary$cp)
points(fwd.min.cp , fwd.summary$cp[fwd.min.cp], col = "red", cex=2, pch=20)

# BIC -> model with 3 features
plot(fwd.summary$bic, xlab = "Number variables", ylab = "BIC", type = "l")
fwd.min.bic <- which.min(fwd.summary$bic)
points(fwd.min.bic, fwd.summary$bic[fwd.min.bic], col = "red", cex=2, pch=20)

# adj R-sq -> model with 6 features
plot(fwd.summary$adjr2, xlab = "Number of variables", ylab = "adj R-sq", type = "l")
fwd.max.adjr2 <- which.max(fwd.summary$adjr2)
points(fwd.max.adjr2, fwd.summary$adjr2[fwd.max.adjr2], col = "red", cex=2, pch=20)
```



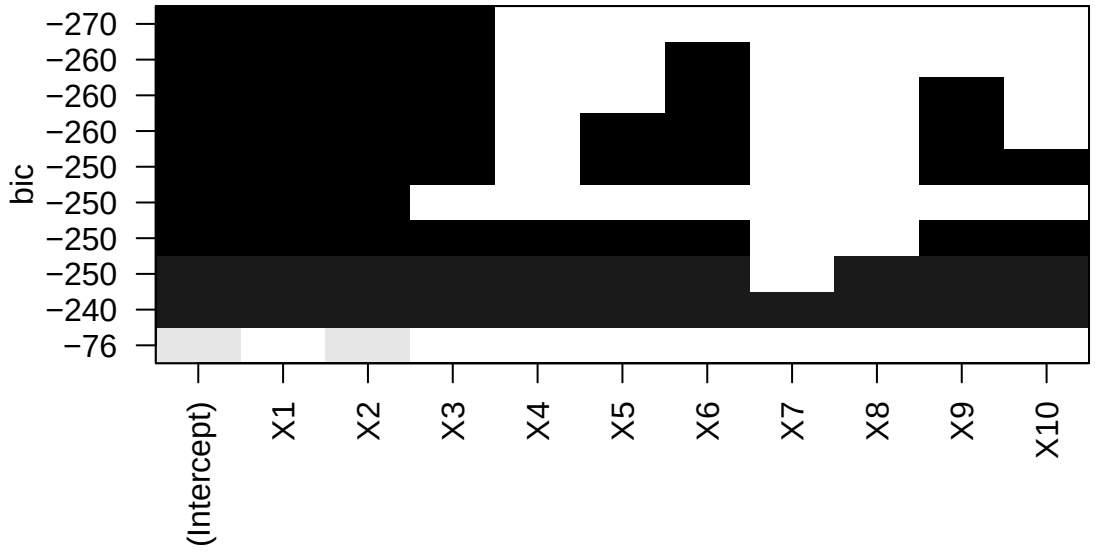




```
# Selected Predictors
plot(regfit.fwd, scale = "Cp")
```




```
plot(regfit.fwd, scale = "bic")
```




```
plot(regfit.fwd, scale = "adjr2")
```


```
# Coefficients
# Best subset based on Cp
coef(regfit.fwd, fwd.min.cp)
```

```
## (Intercept)          X1          X2          X3
##   2.9703939   1.9204462  -3.0915431   0.3204363
```

```
# # Best subset based on BIC
coef(regfit.fwd, fwd.min.bic)
```

```
## (Intercept)          X1          X2          X3
##   2.9703939   1.9204462  -3.0915431   0.3204363
```

```
# Best subset based on adj R-sq
coef(regfit.fwd, fwd.max.adjr2)
```

```
##  (Intercept)           X1           X2           X3           X6
##  3.040475559  1.928367169 -3.253869916  0.323397035  0.009248115
```

```
Comment:
In the cases of Cp and BIC, Forward Stepwise method selects the same variables
as the ones Best Subset selects. However, for the adj R-sq case,
the selected models are different.
We can see a difference between the two methods here:
Best subset selection is not restricted to making pairs of variables
but the pairs of FWD Stepwise
must include previous pairs as the number of variables increase.
```

```
# Backward Stepwise Selectioin
regfit.bwd <- regsubsets(Y~., data = simul.df, nvmax = 10, method = "backward")
bwd.summary <- summary(regfit.bwd)
print(bwd.summary)
```

```
## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = simul.df, nvmax = 10, method = "backward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: backward
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 )  " " " " " " "*" " " " " " " " " " " " "
## 2  ( 1 )  "*" " " " " "*" " " " " " " " " " " " "
## 3  ( 1 )  "*" " " " " "*" " " "*" " " " " " " " "
## 4  ( 1 )  "*" " " " " "*" " " "*" " " "*" " " " "
## 5  ( 1 )  "*" " " " " "*" "*" "*" " " "*" " " " "
## 6  ( 1 )  "*" " " " " "*" "*" "*" " " "*" " " "*"
## 7  ( 1 )  "*" " " " " "*" "*" "*" "*" "*" " " "*"
## 8  ( 1 )  "*" "*" " " "*" "*" "*" "*" "*" " " "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" " " "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
# Cp
bwd.summary$cp
```

```
##  [1] 707.825918 203.249864  76.623982  18.559936  13.292162   4.260994
##  [7]   5.396602   7.109673   9.022299  11.000000
```

```
# BIC
bwd.summary$bic
```

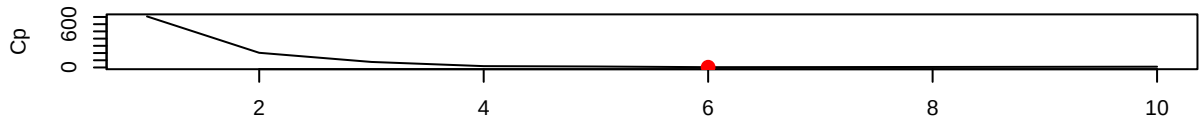
```
##  [1]  -62.37373 -157.24952 -209.33453 -248.76625 -251.09042 -258.01561
##  [7] -254.37271 -250.08902 -245.58195 -241.00183
```

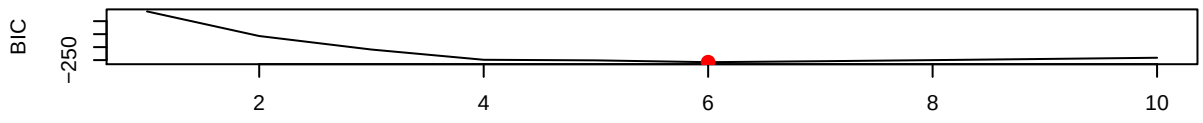
```
# adj R-sq
bwd.summary$adjr2
```

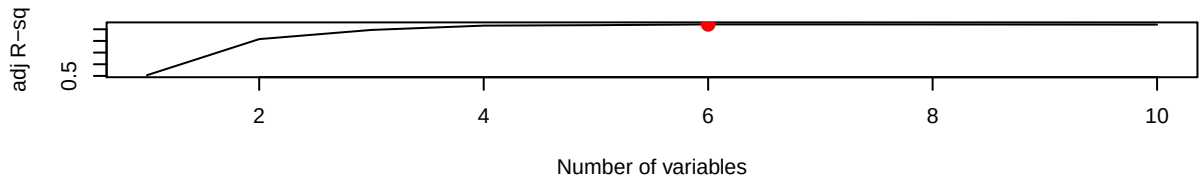
```
##  [1] 0.5062314 0.8155250 0.8942609 0.9312087 0.9351313 0.9415742 0.9415047
##  [8] 0.9410517 0.9404552 0.9398012
```

```
# Plot (Backward Stepwise)

par(mfrow=c(3,1))

# Cp -> model with 6 features
plot(bwd.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
```

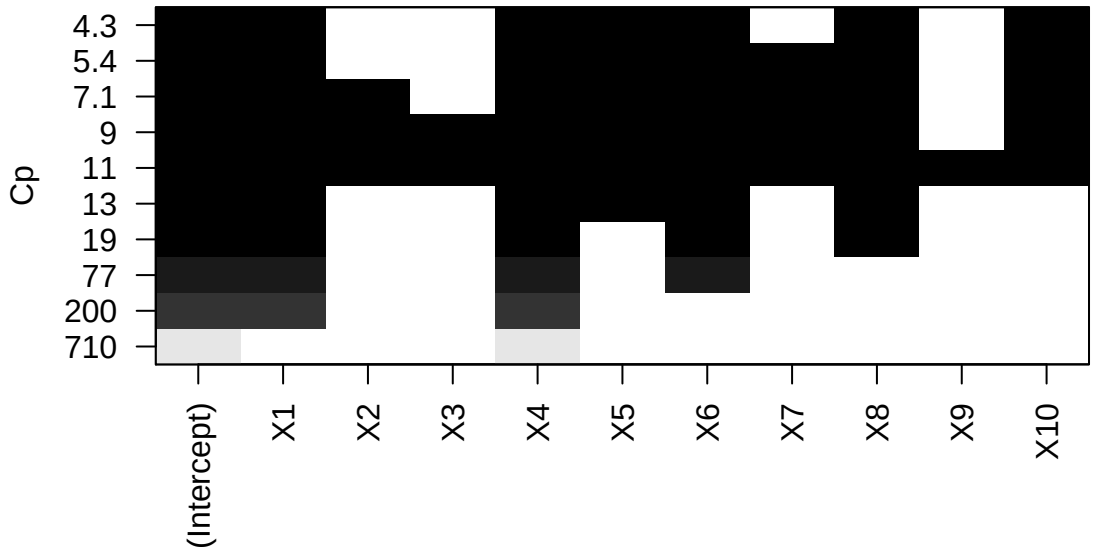
```r
bwd.min.cp <- which.min(bwd.summary$cp)
points(bwd.min.cp, bwd.summary$cp[bwd.min.cp], col = "red", cex=2, pch=20)

# BIC -> model with 6 features
plot(bwd.summary$bic, xlab = "Number variables", ylab = "BIC", type = "l")
bwd.min.bic <- which.min(bwd.summary$bic)
points(bwd.min.bic, bwd.summary$bic[bwd.min.bic], col = "red", cex=2, pch=20)

# adj R-sq -> model with 6 features
plot(bwd.summary$adjr2, xlab = "Number of variables", ylab = "adj R-sq", type = "l")
bwd.max.adjr2 <- which.max(bwd.summary$adjr2)
points(bwd.max.adjr2 , bwd.summary$adjr2[bwd.max.adjr2 ], col = "red", cex=2, pch=20)
```



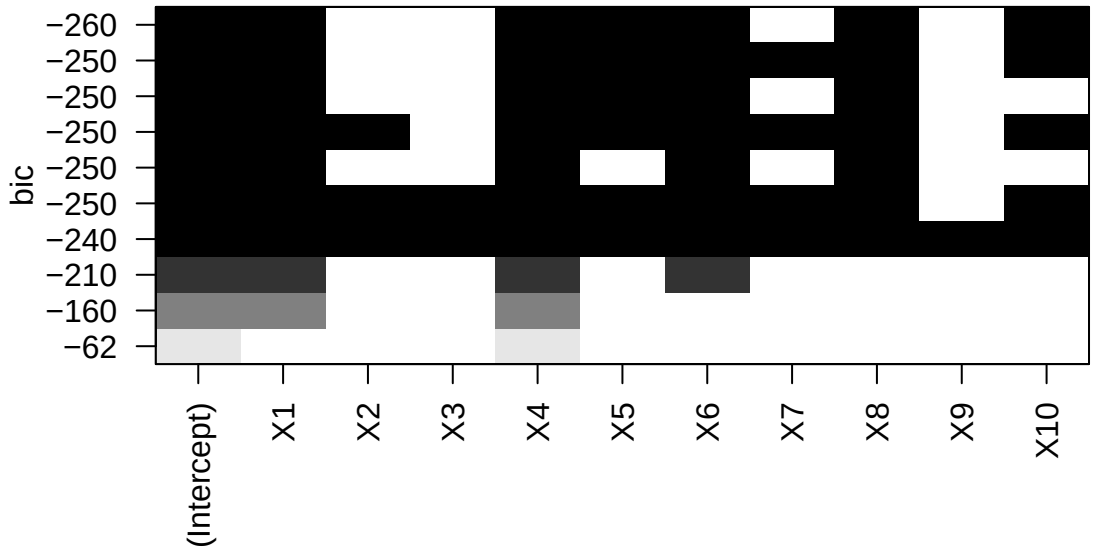




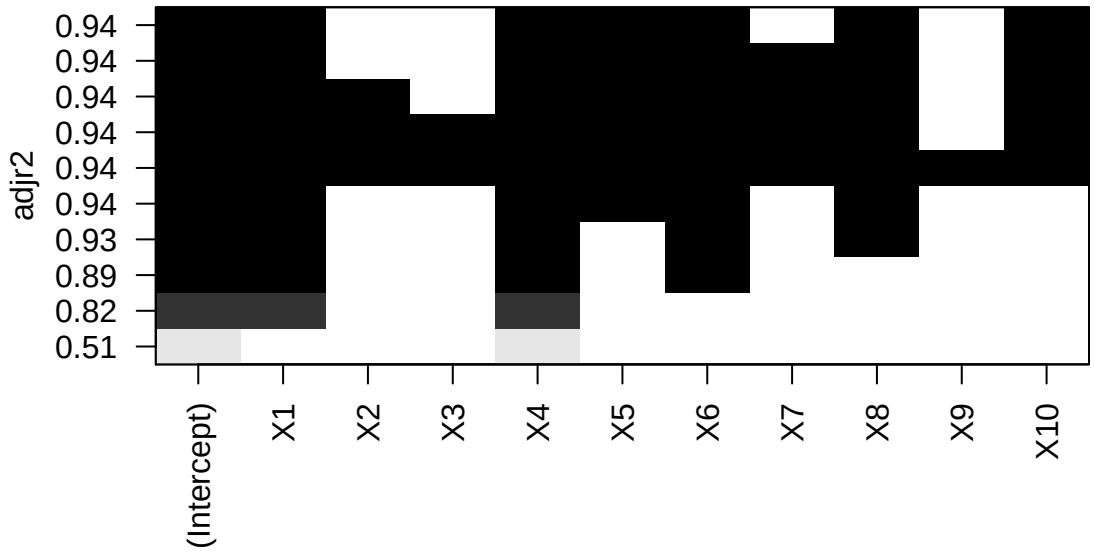

```r
# Selected Predictors

plot(regfit.bwd, scale = "Cp")
```




```
plot(regfit.bwd, scale = "bic")
```




```
plot(regfit.bwd, scale = "adjr2")
```




```
# Coefficients
# Best subset based on Cp
coef(regfit.bwd, bwd.min.cp)
```

```
## (Intercept)          X1          X4          X5          X6          X8
##  2.73249039  2.17835190 -5.40990489  0.07249919  3.14030764 -0.73034440
##         X10
##  0.05888058
```

```
# # Best subset based on BIC
coef(regfit.bwd, bwd.min.bic)
```

```
## (Intercept)          X1          X4          X5          X6          X8
##  2.73249039  2.17835190 -5.40990489  0.07249919  3.14030764 -0.73034440
##         X10
##  0.05888058
```

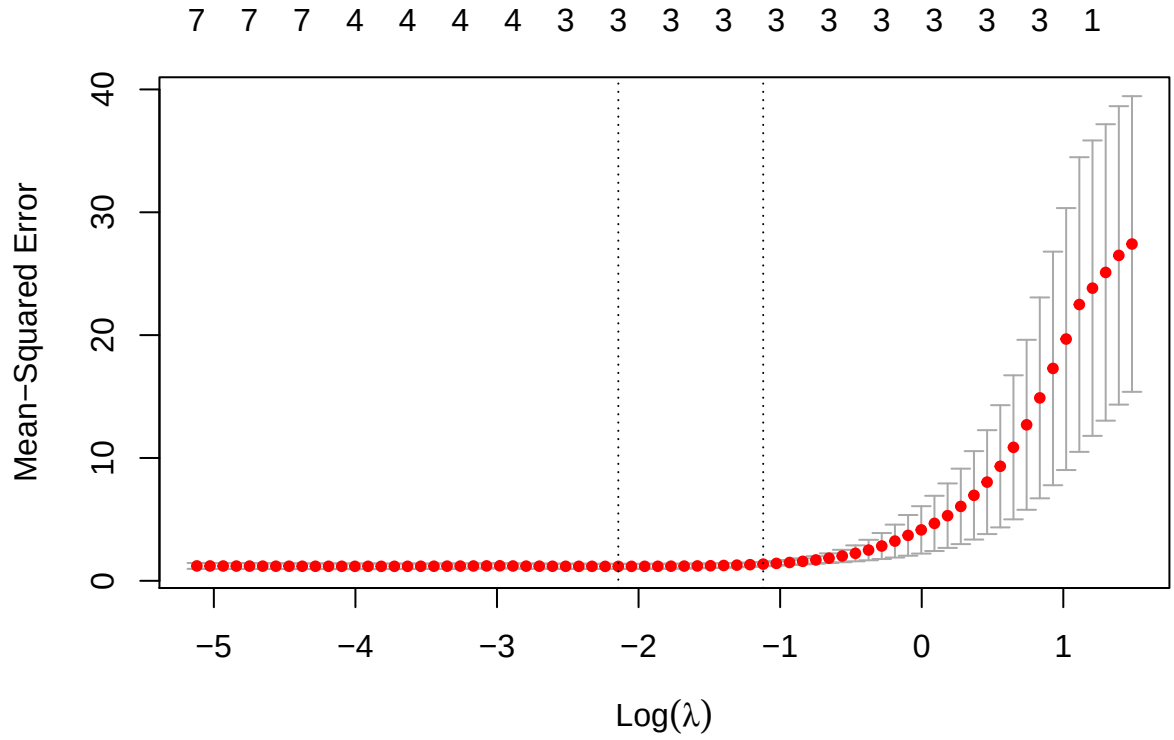
```
# Best subset based on adj R-sq
coef(regfit.bwd, bwd.max.adjr2)
```

```
## (Intercept)          X1          X4          X5          X6          X8
##  2.73249039  2.17835190 -5.40990489  0.07249919  3.14030764 -0.73034440
##         X10
##  0.05888058
```

```
Comment:
In comparison to the best models selected by Best Subset and FWD Stepwise methods,
BWD Stepwise selects different variables across Cp, BIC and adj R-sq.
Unlike Best Subset & FWD, BWD deos not generate the expected model which is X, X^2, X^3.
```

(e) Fit a LASSO model with `glmnet` function from `glmnet` package to the simulated data, again using $(X, X^2, \cdots, X^{10})$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ.

```r
library(glmnet)
x <- model.matrix(Y~.,simul.df)[,-1]
y = simul.df$Y

# Split the data frame into train and test sets
set.seed(123)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
y.test = y[test]

# fit Lasso Regression (Cross Validation)
set.seed(123)
cv.out = cv.glmnet(x[train,], y[train], alpha = 1) # To find the optimal tuning parameter
plot(cv.out)
```




```r
bestlam = cv.out$lambda.min # the lambda minimizing RSS

lasso.mod <- glmnet(x[train,], y[train], alpha=1, lambda = bestlam)
coef(lasso.mod)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept)  3.0304942
## X1           1.8928533
## X2          -2.9946425
## X3           0.3232353
## X4           .
## X5           .
## X6           .
```

```
## X7           .
## X8           .
## X9           .
## X10          .
```

```
lasso.pred <- predict(lasso.mod, s=bestlam, newx = x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 1.014784
```

```
out <- glmnet(x, y, alpha=1, lambda = bestlam)
lasso.coef = predict(out, type="coefficients", s = bestlam)[1:11]
lasso.coef[lasso.coef != 0]
```

```
## [1]  2.8777952  1.8380125 -2.9608165  0.2876199
```

```
Comment:
Based on the lambda obtained by cross validation, Lasso Regression selects
the best model by dropping some variables to zero. As a result,
the X, X^2, and X^3 are selected that were used to create the Y variable.
By means of this feature of Lasso that drops some features to zero,
it gives better interpretability.
```

(f) Now generate a response vector Y according to the model

$$Y = \beta_0 + \beta_7 X^7 + \epsilon,$$

where $\beta_7 = 7$, and perform best subset selection and the LASSO.

```
X.mat.new <- matrix(c(rep(1,100), X^7), ncol = 2)
beta.new <- c(3, 7)
Y.new <- X.mat.new%*%beta.new + error
predictor <- matrix(c(X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10), ncol = 10)
simul.df.new <- data.frame(Y.new, predictor)
```

Best Subset Selection (Cp)

```
reg.fit.new <- regsubsets(Y.new~., data = simul.df.new, nvmax = 10)
reg.summary.new <- summary(reg.fit.new)
reg.summary.new
```

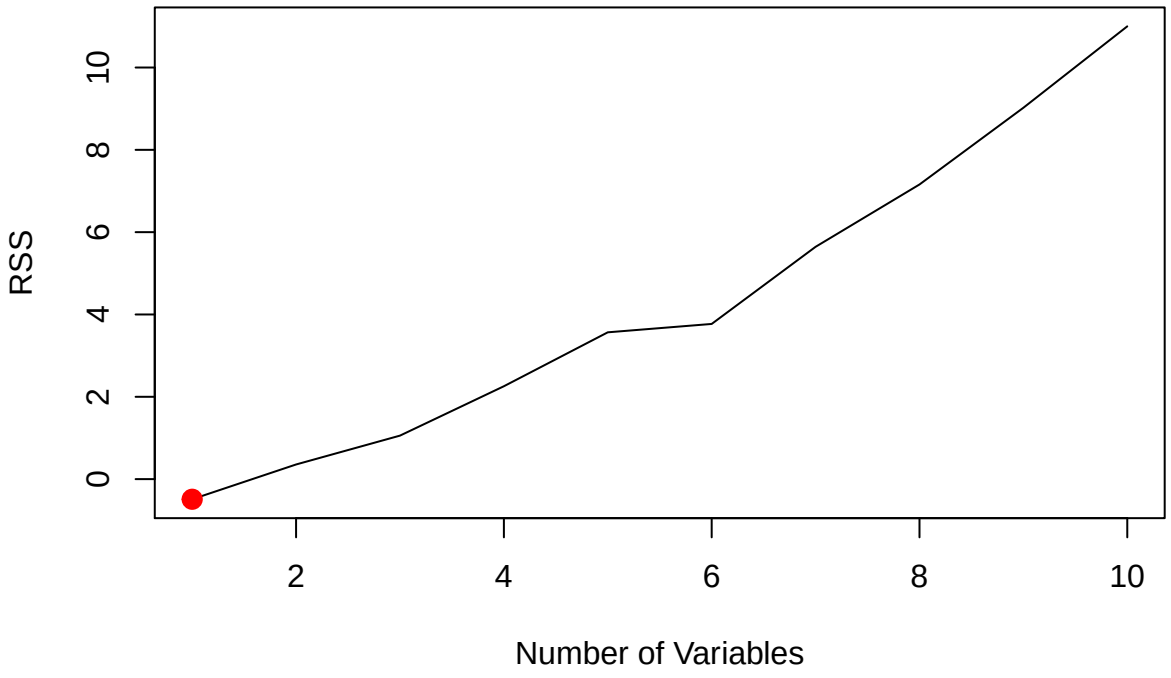
```
## Subset selection object
## Call: regsubsets.formula(Y.new ~ ., data = simul.df.new, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
```

```
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 )  " " " " " " " " " " " " "*" " " " " " "
## 2  ( 1 )  " " "*" " " " " " " " " "*" " " " " " "
## 3  ( 1 )  " " "*" " " " " " " " " "*" "*" " " " " " "
## 4  ( 1 )  " " " " " " "*" " " " " "*" "*" "*" " " " " "
## 5  ( 1 )  " " "*" " " "*" " " " " "*" "*" "*" " " " " "
## 6  ( 1 )  " " "*" " " "*" " " " " "*" "*" "*" " " " " "*"
## 7  ( 1 )  "*" "*" " " "*" " " " " "*" "*" "*" " " " " "*"
## 8  ( 1 )  " " "*" "*" "*" " " " " "*" "*" "*" "*" "*"
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" " " " " "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

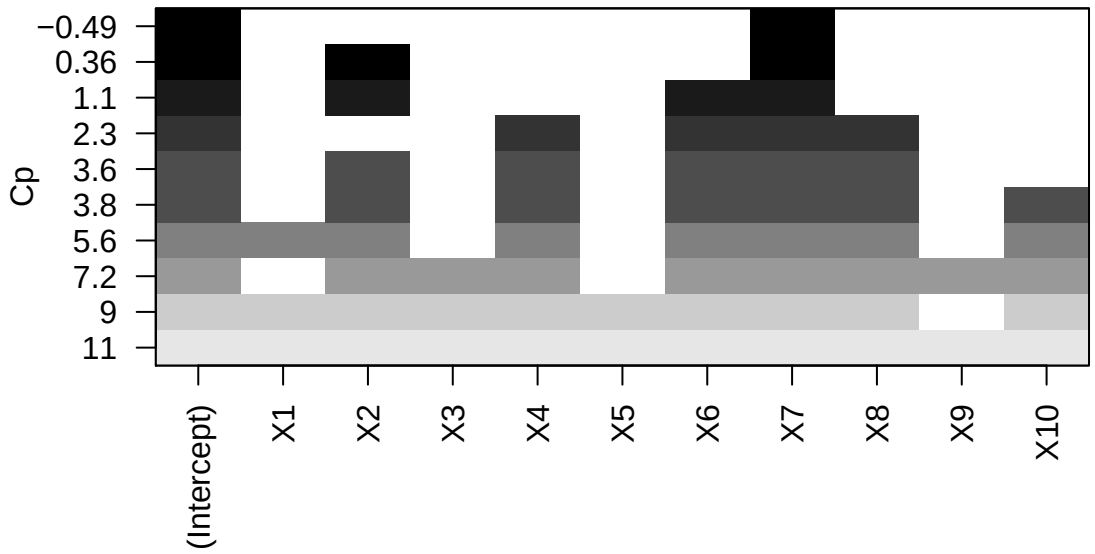
```r
# Cp
reg.summary.new$cp
```

```
##  [1] -0.4892301  0.3551574  1.0565357  2.2565990  3.5665229  3.7694845
##  [7]  5.6431661  7.1581102  9.0222992 11.0000000
```

```r
# Plot

# Cp -> model with 1 feature
plot(reg.summary.new$cp, xlab = "Number of Variables", ylab = "RSS", type = "l")
best.min.cp <- which.min(reg.summary.new$cp)
points(best.min.cp, reg.summary.new$cp[best.min.cp], col = "red", cex=2, pch=20)
```



```
# Selected Predictors
plot(reg.fit.new, scale = "Cp") # Cp -> model with 1 feature: X~7
```

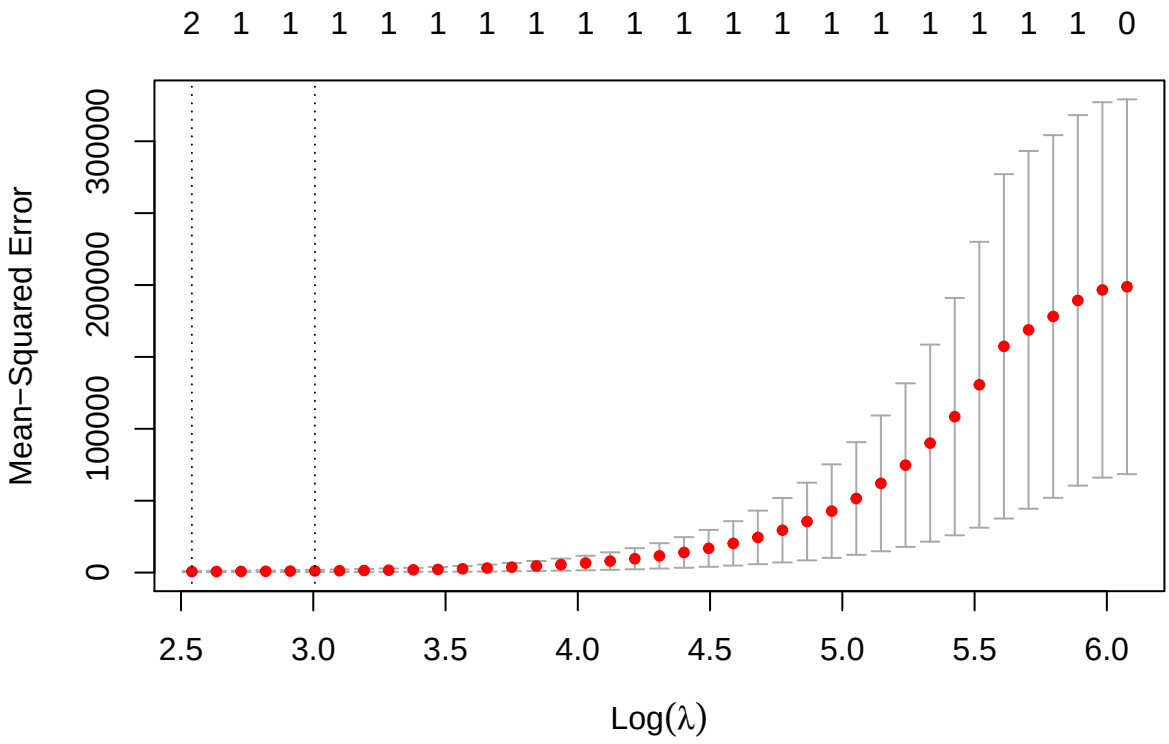



```
# Coefficients
coef(reg.fit.new, best.min.cp)
```

```
## (Intercept)          X7
##    2.893251    6.999704
```

Lasso

```
x.new <- model.matrix(Y.new~.,simul.df.new)[,-1]
y.new = simul.df.new$Y.new

set.seed(123)
train = sample(1:nrow(x.new), nrow(x.new)/2)
test = (-train)
y.test = y.new[test]

# fit Lasso Regression (Cross Validation)
set.seed(123)
cv.out = cv.glmnet(x.new[train,], y.new[train], alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min

lasso.mod.new <- glmnet(x.new[train,], y.new[train], alpha=1, lambda = bestlam)
coef(lasso.mod.new)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 2.03162737
## X1          .
## X2          .
## X3          .
## X4          .
## X5          1.65704384
## X6          .
## X7          5.97297572
## X8          .
## X9          0.09720614
## X10         .
```

```
lasso.pred.new <- predict(lasso.mod.new, s=bestlam, newx = x.new[test,])
mean((lasso.pred.new-y.test)^2)
```

```
## [1] 78.86932
```

```
out.new <- glmnet(x.new, y.new, alpha=1, lambda = bestlam)
lasso.coef.new = predict(out.new, type="coefficients", s = bestlam)[1:11]
lasso.coef.new[lasso.coef.new != 0]
```

```
## [1] 3.56126923 6.74527146 0.00220135
```

```
Comment:
The resulting best model by Best Subset selection includes only X^7 in the model
which was used to create Y variable. But the Lasso Regression
selects three features one of which is X^7.
```

2. I will predict the number of applications received using the other variables in the `College` data set from `ISLR` package.

(a) Randomly split the data set into a training set and a test set (1:1).

```
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

```
x <- model.matrix(Apps~., data = College)[, -1]
y <- College$Apps

set.seed(123)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
college.lm <- lm(Apps~., data = College[train,]) # Fit LS
print(summary(college.lm))
```

```
##
## Call:
## lm(formula = Apps ~ ., data = College[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

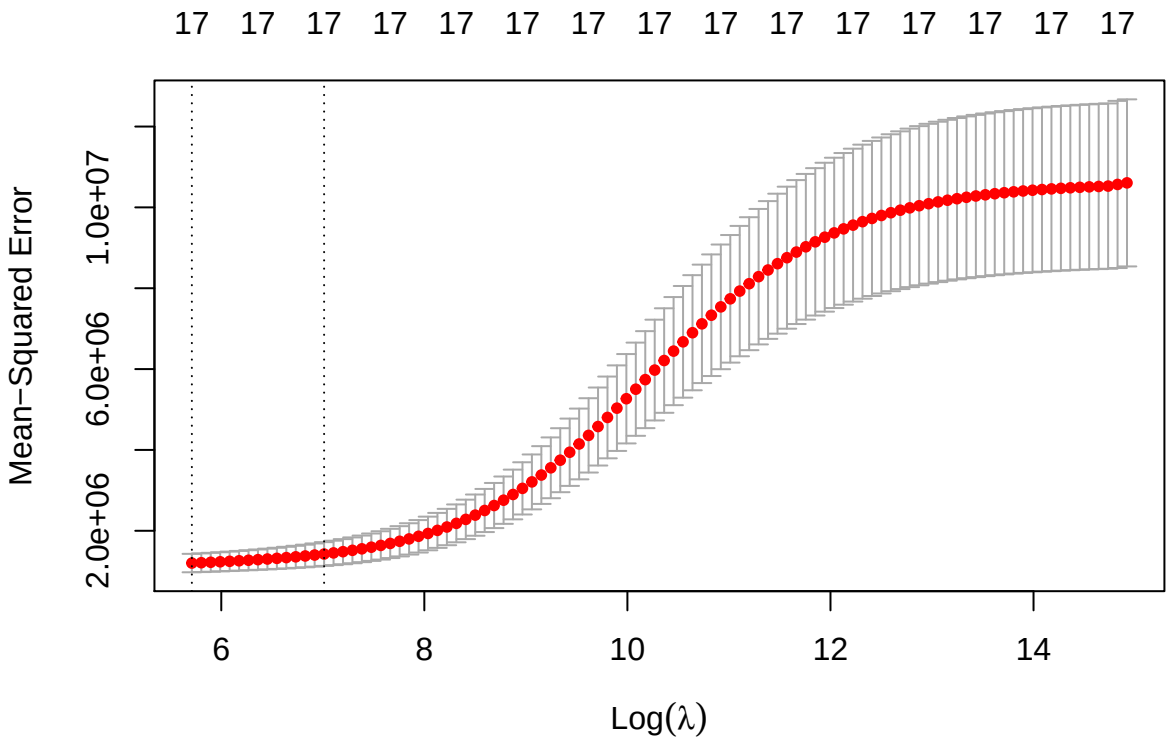
```
## -2623.9  -472.8   -64.4   319.2  6042.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.778e+01  5.815e+02   0.134  0.89366
## PrivateYes  -8.146e+02  2.002e+02  -4.069 5.77e-05 ***
## Accept       1.350e+00  7.318e-02  18.448  < 2e-16 ***
## Enroll      -1.455e-01  2.627e-01  -0.554  0.58006
## Top10perc    3.784e+01  7.111e+00   5.322 1.79e-07 ***
## Top25perc   -8.680e+00  5.646e+00  -1.538  0.12502
## F.Undergrad  2.157e-02  4.778e-02   0.451  0.65192
## P.Undergrad -2.767e-03  5.049e-02  -0.055  0.95632
## Outstate    -5.351e-02  2.467e-02  -2.169  0.03075 *
## Room.Board   1.709e-01  6.102e-02   2.801  0.00536 **
## Books        5.341e-02  3.183e-01   0.168  0.86682
## Personal    -9.869e-02  8.594e-02  -1.148  0.25157
## PhD         -6.503e+00  6.215e+00  -1.046  0.29608
## Terminal    -6.148e+00  7.156e+00  -0.859  0.39083
## S.F.Ratio   -8.663e+00  1.953e+01  -0.443  0.65769
## perc.alumni -8.923e+00  5.544e+00  -1.610  0.10835
## Expend       7.938e-02  1.951e-02   4.068 5.80e-05 ***
## Grad.Rate    1.115e+01  3.966e+00   2.811  0.00520 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 965.7 on 370 degrees of freedom
## Multiple R-squared:  0.9158, Adjusted R-squared:  0.9119
## F-statistic: 236.6 on 17 and 370 DF,  p-value: < 2.2e-16
```

```
pred.test <- predict(college.lm, newdata = College[test,])

# Test Error (MSE based on the test set)
paste("Test error is", round(mean((pred.test - y.test)^2), 3))
```

```
## [1] "Test error is 1373994.683"
```

(c) Fit a ridge regression model on the training set, with λ chosen by 5-fold cross-validation. Report the test error obtained.

```
# Optimal tuning parameter (5-Folds CV)
cv.ridge <- cv.glmnet(x[train,], y[train], alpha = 0, nfolds=5)
plot(cv.ridge)
```




```
bestlamb.ridge <- cv.ridge$lambda.min
paste("Best Lambda =", round(bestlamb.ridge,4))
```

```
## [1] "Best Lambda = 301.9518"
```

```
# Fit a Ridge model
ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = bestlamb.ridge)

# Ridge coefficient estimates
coef(ridge.mod)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -823.94709397
## PrivateYes  -760.74446434
## Accept         0.84954596
## Enroll         0.64009139
## Top10perc     23.40725492
## Top25perc      0.63260565
## F.Undergrad    0.09419837
## P.Undergrad   -0.04171499
## Outstate      -0.01153920
## Room.Board     0.22216845
## Books          0.18546886
## Personal      -0.12147801
## PhD           -2.79005989
## Terminal      -6.14187264
## S.F.Ratio     -6.23981722
## perc.alumni  -13.93570597
```

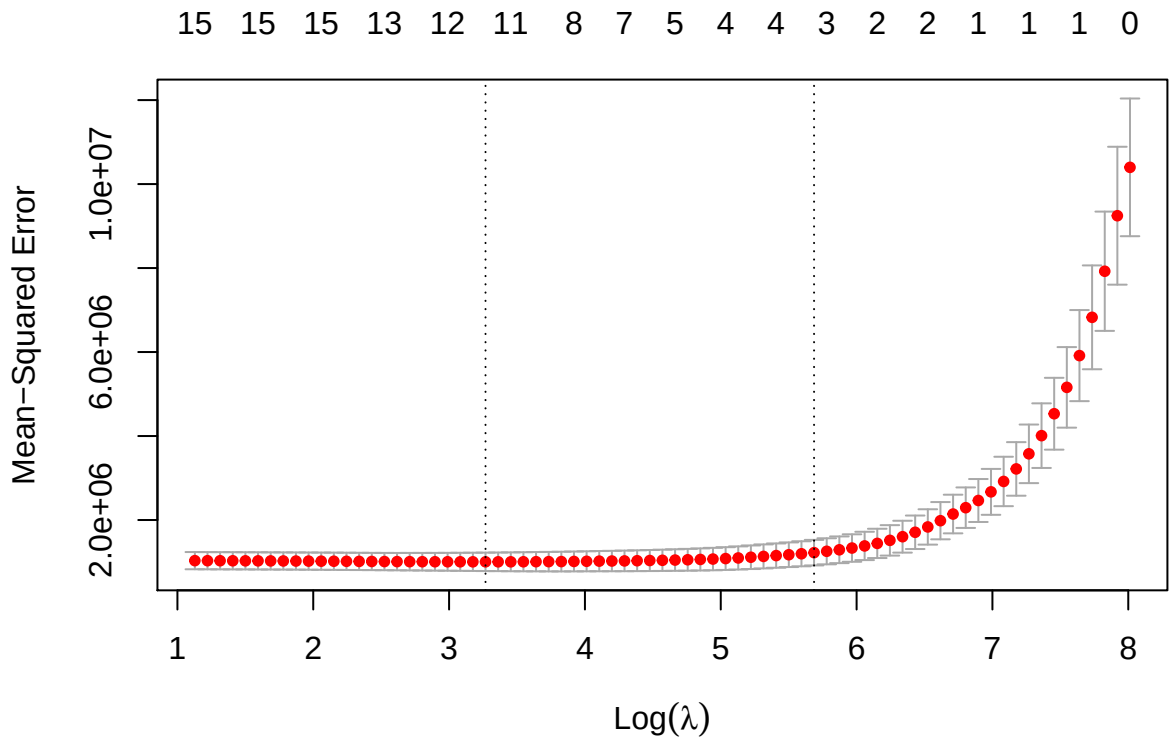
```
## Expend          0.07591727
## Grad.Rate      11.64402849
```

```
# Predicted values on the test set
ridge.pred <- predict(ridge.mod, lambda = bestlamb.ridge, newx = x[test,])

# Test Error (MSE)
paste("Test error is", round(mean((ridge.pred-y.test)^2), 3))
```

```
## [1] "Test error is 2092402.187"
```

(d) Fit a LASSO model on the training set, with λ chosen by 5-fold cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
# Optimal tuning parameter (5-Folds CV)
cv.lasso <- cv.glmnet(x[train,], y[train], alpha = 1, nfolds=5)
plot(cv.lasso)
```




```
bestlamb.lasso <- cv.lasso$lambda.min
paste("Best Lambda =", round(bestlamb.lasso,4))
```

```
## [1] "Best Lambda = 26.2622"
```

```
# Fit a Lasso model
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = bestlamb.lasso)

# Predicted values on the test set
```

```
lasso.pred <- predict(lasso.mod, lambda = bestlamb.lasso, newx = x[test,])

# Test Error (MSE)
paste("Test error is", round(mean((lasso.pred-y.test)^2), 3))
```

```
## [1] "Test error is 1400811.376"
```

```
# coefficient estimates
coef(lasso.mod)
```

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -463.77291923
## PrivateYes  -706.17010334
## Accept         1.33062226
## Enroll         .
## Top10perc     26.51145228
## Top25perc      .
## F.Undergrad    .
## P.Undergrad    .
## Outstate      -0.02502523
## Room.Board     0.11904671
## Books          .
## Personal      -0.03165470
## PhD           -2.70628028
## Terminal      -5.64060257
## S.F.Ratio      .
## perc.alumni   -6.52803867
## Expend         0.07333110
## Grad.Rate      6.95696564
```

```
# non-zero coefficient estimates
lasso.coef = predict(lasso.mod, type="coefficients", s = bestlamb.lasso)[1:17]
lasso.coef[lasso.coef != 0]
```

```
##  [1] -463.77291923 -706.17010334    1.33062226   26.51145228   -0.02502523
##  [6]    0.11904671   -0.03165470   -2.70628028   -5.64060257   -6.52803867
## [11]    0.07333110
```

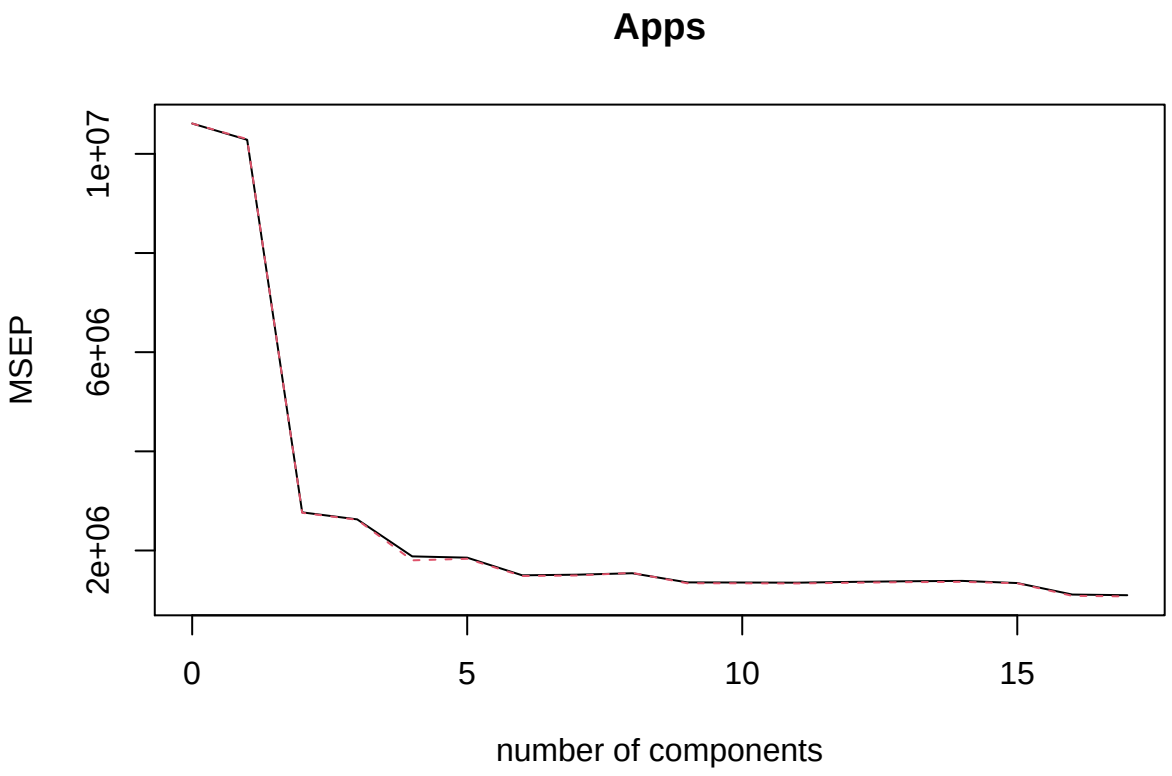
(e) Fit a PCR model on the training set, with M chosen by 5-fold cross-validation.

```
set.seed(123)
pcr.fit <- pcr(Apps~., data = College[train,], scale = TRUE, validation = "CV", segments = 5)
# summary of the fitted PCR
summary(pcr.fit)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 1
## Fit method: svdpc
## Number of components considered: 17
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 5 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3258     3207     1664     1621     1371     1362     1224
## adjCV         3258     3209     1661     1619     1342     1354     1219
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1230     1241     1165      1164      1162      1169      1174
## adjCV     1223     1244     1157      1157      1156      1162      1167
##        14 comps  15 comps  16 comps  17 comps
## CV         1178      1159      1054      1048
## adjCV      1169      1157      1042      1036
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      32.969    59.69    66.68    72.25    77.31    81.82    85.18    88.34
## Apps    3.259    74.43    76.35    84.40    84.42    86.69    86.79    87.01
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       91.32     93.61     95.48     97.05     98.11     98.90     99.40
## Apps    88.38     88.39     88.47     88.49     88.50     88.51     88.74
##       16 comps  17 comps
## X        99.81    100.00
## Apps     91.48     91.58
```

```r
# Plot MSEP by M
validationplot(pcr.fit, val.type = "MSEP")
```




```r
print("The lowest CV error occurs at M = 17")
```

```
## [1] "The lowest CV error occurs at M = 17"
```

```r
# Test MSE
pcr.pred <- predict(pcr.fit, x[test,], ncomp = 17)
paste("Test error is", round(mean((pcr.pred-y.test)^2), 4), "with M = 17")
```

```
## [1] "Test error is 1373994.6833 with M = 17"
```

```r
print("No dimensionality reduction occurs")
```

```
## [1] "No dimensionality reduction occurs"
```

```
Comment:
The Test errors are significantly big no matter which model we use out of the four.
The test errors from the LS, Lasso, and PCR models are similar.
The Ridge model produces the biggest test MSE.
So, in order to raise its accuracy, thorough preprocessing seems to be needed.
```

3. using the `Weekly` data set, which is part of the `ISLR` package.

(a) Produce some numerical and graphical summaries of the `Weekly` data.

```r
# column names
names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```r
# number of row & col
dim(Weekly)
```

```
## [1] 1089    9
```

```r
# summary of the dataset
summary(Weekly)
```

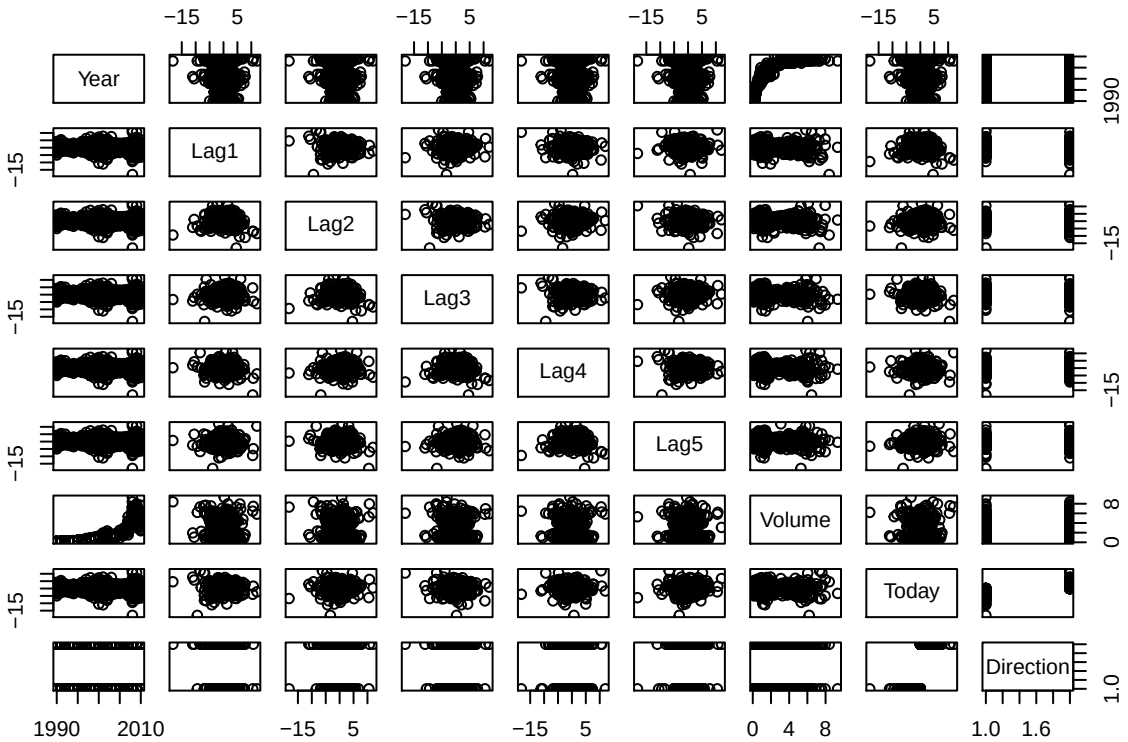
```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
```

```
##  Down:484
##  Up  :605
##
##
##
##
```

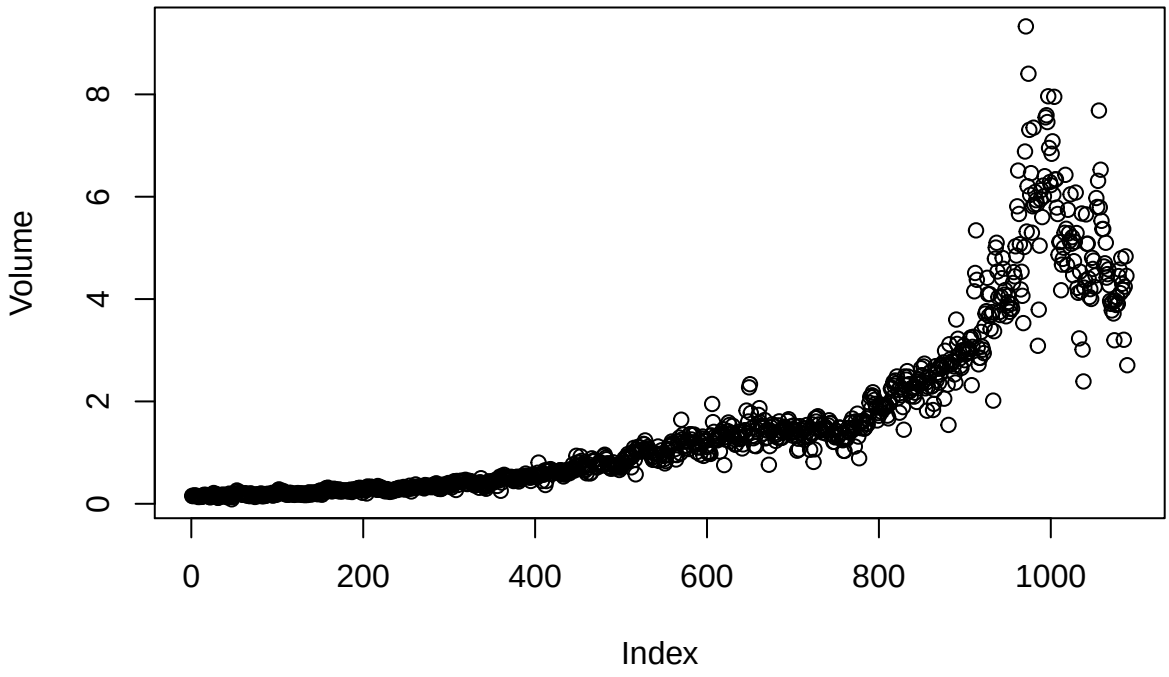
```
# Correlations
cor(Weekly[, -9]) # Year & Volume are highly correlated
```

```
##                Year         Lag1        Lag2        Lag3         Lag4
## Year     1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1    -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2    -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3    -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4    -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5    -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume   0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today   -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year    -0.030519101  0.84194162 -0.032459894
## Lag1    -0.008183096 -0.06495131 -0.075031842
## Lag2    -0.072499482 -0.08551314  0.059166717
## Lag3     0.060657175 -0.06928771 -0.071243639
## Lag4    -0.075675027 -0.06107462 -0.007825873
## Lag5     1.000000000 -0.05851741  0.011012698
## Volume  -0.058517414  1.00000000 -0.033077783
## Today    0.011012698 -0.03307778  1.000000000
```

```
pairs(Weekly)
```



```r
# Pattern of Volume
attach(Weekly)
plot(Volume)
```



(b) Use the full data set to perform a logistic regression with `Direction` as the response and the five lag variables plus `Volumn` as predictors.

```r
weekly.df <- Weekly[c(2,3,4,5,6,7,9)]

# Fit logistic model
weekly.logit <- glm(Direction~., data = weekly.df, family = binomial)
summary(weekly.logit)
```

```
##
## Call:
## glm(formula = Direction ~ ., family = binomial, data = weekly.df)
##
## 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
```

Comment:
the p-value of the lag2 is smaller than 0.05 which is statistically significant.

(c) Compute the confusion matrix and overall fraction of correct predictions.

```
# Compute the probability of Direction being "up" given X
glm.probs <- predict(weekly.logit, type = "response")

# Convert the probabilities into class labels
glm.pred <- rep("Down", 1089)
glm.pred[glm.probs > .5] = "Up"

# Confusion matrix
confusion.mat <- table(glm.pred, Direction)
print(confusion.mat)
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
# Diagonal elements: Correct predictions
mean(glm.pred == Direction) # Approximately 56% fraction of correct prediction (based on train set)
```

```
## [1] 0.5610652
```

```
paste("Train error:", round(1-mean(glm.pred == Direction), 3))
```

```
## [1] "Train error: 0.439"
```

```
# Mistakes made by the logistic regression
confusion.mat[2,1] # False Positive:
```

```
## [1] 430
```

```
confusion.mat[1,2] # False Negative
```

```
## [1] 48
```

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with `Lag2` as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
# Make a hold-out sample
train <- (Year <= 2008)

# Fit the logistic model with only the significant variable on the train set
logit.new <- glm(Direction~Lag2, data = Weekly, family = binomial, subset = train)
summary(logit.new)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```
# Predict the Direction on the hold-out set
prob.new <- predict(logit.new, Weekly[!train,], type = "response")
pred.new <- rep("Down", 104)
pred.new[prob.new > 0.5] = "Up"

# Confusion matrix
confusion.new <- table(pred.new, Direction[!train])
print(confusion.new)
```

```
##
## pred.new Down Up
##     Down    9  5
##     Up     34 56
```

```
# Fraction of correct predictions
mean(pred.new == Direction[!train]) # Approximately 62.5%
```

```
## [1] 0.625
```

(e) Repeat (d) using LDA.

```
# Fit LDA
library(MASS)
lda.fit <- lda(Direction~Lag2, data = Weekly, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
# Prediction on the hold-out sample
lda.pred <- predict(lda.fit, Weekly[!train,])
lda.table <- table(lda.pred$class, Direction[!train])
print(lda.table)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

```
# Fraction of correct predictions
mean(lda.pred$class == Direction[!train])
```

```
## [1] 0.625
```

(f) Repeat (d) using QDA.

```
# Fit QDA
qda.fit <- qda(Direction~Lag2, data = Weekly, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
```

```
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```
# Prediction on the hold-out sample
qda.pred <- predict(qda.fit, Weekly[!train,])
qda.table <- table(qda.pred$class, Direction[!train])
print(qda.table)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

```
# Fraction of correct predictions
mean(qda.pred$class == Direction[!train])
```

```
## [1] 0.5865385
```

Commnet:
Logistic regression and Linear Discriminant Regression (LDA) yield the best accuracy on the basis of their test error which is 0.375.
QDA's test error is bigger than the two:

```
# Test error
# Logistic Model
print(1-mean(pred.new == Direction[!train]))
```

```
## [1] 0.375
```

```
# LDA
print(1-mean(lda.pred$class == Direction[!train]))
```

```
## [1] 0.375
```

```
# QDA
print(1-mean(qda.pred$class == Direction[!train]))
```

```
## [1] 0.4134615
```

(g) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data.

```
# Logistic with different variable set
glm.fit <- glm(Direction~Lag1 + Lag2 + log(Volume) + Lag1*log(Volume), data = Weekly, family = binomial
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + log(Volume) + Lag1 *
##     log(Volume), family = binomial, data = Weekly, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5401  -1.2585   0.9951   1.0845   1.5621
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.184654   0.067388   2.740  0.00614 **
## Lag1             -0.055926   0.029750  -1.880  0.06013 .
## Lag2              0.050193   0.029265   1.715  0.08632 .
## log(Volume)      -0.089513   0.065200  -1.373  0.16978
## Lag1:log(Volume) -0.006391   0.026500  -0.241  0.80944
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1345.0  on 980  degrees of freedom
## AIC: 1355
##
## Number of Fisher Scoring iterations: 4
```

```
# Prediction
glm.prob <- predict(glm.fit, Weekly[!train,], type = "response")
glm.pred <- c(rep("Down", 104))
glm.pred[glm.prob > 0.5] = "Up"

# Confusion matrix for Logistic Model
print(table(glm.pred, Direction[!train]))
```

```
##
## glm.pred Down Up
##     Down   22 19
##     Up     21 42
```

```
# Test error of logistic model
paste("test error is", round(1-mean(glm.pred == Direction[!train]),3))
```

```
## [1] "test error is 0.385"
```

```
# LDA with different variable set
lda.fit <- lda(Direction~Lag1 + Lag2 + log(Volume) + Lag1*log(Volume), data = Weekly, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2 + log(Volume) + Lag1 * log(Volume),
##     data = Weekly, subset = train)
```

```
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag1        Lag2 log(Volume) Lag1:log(Volume)
## Down  0.289444444 -0.03568254  -0.2386605       -0.1419234
## Up   -0.009213235  0.26036581  -0.3281402       -0.2510587
##
## Coefficients of linear discriminants:
##                            LD1
## Lag1              -0.27799059
## Lag2               0.24759651
## log(Volume)       -0.44886364
## Lag1:log(Volume) -0.02864704
```

```
# Confusion matrix for LDA
lda.pred <- predict(lda.fit, Weekly[!train,])
lda.table <- table(lda.pred$class, Direction[!train])

# Test error of LDA
paste("test error is", round(1-mean(lda.pred$class == Direction[!train]),3))
```

```
## [1] "test error is 0.385"
```

Comment:
QDA produces the best result in terms of its test error that is 0.375.
The variable used for the QDA model is Lag2^2 which is the transformed Lag2

```
# QDA with different variable set
qda.fit <- qda(Direction ~ poly(Lag2,2), data = Weekly, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ poly(Lag2, 2), data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##      poly(Lag2, 2)1 poly(Lag2, 2)2
## Down   -0.002401964  -0.0018033874
## Up      0.001405552  -0.0001094245
```

```
# Confusion matrix for QDA
qda.pred <- predict(qda.fit, Weekly[!train,])
qda.table <- table(qda.pred$class, Direction[!train])
print(qda.table)
```

```
##
```

```
##        Down Up
##   Down    7  3
##   Up     36 58
```

```
# Test error of QDA
paste("test error is", round(1-mean(qda.pred$class == Direction[!train]),3))
```

```
## [1] "test error is 0.375"
```

---

4. I will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

(a) Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median.

```
auto <- Auto[, -1]
auto <- auto[, c(8,1,2,3,4,5,6,7)]
auto$mpg01 <- (Auto$mpg > median(Auto$mpg))+0 # mpg > median = 1 & otherwise = 0
str(auto)
```

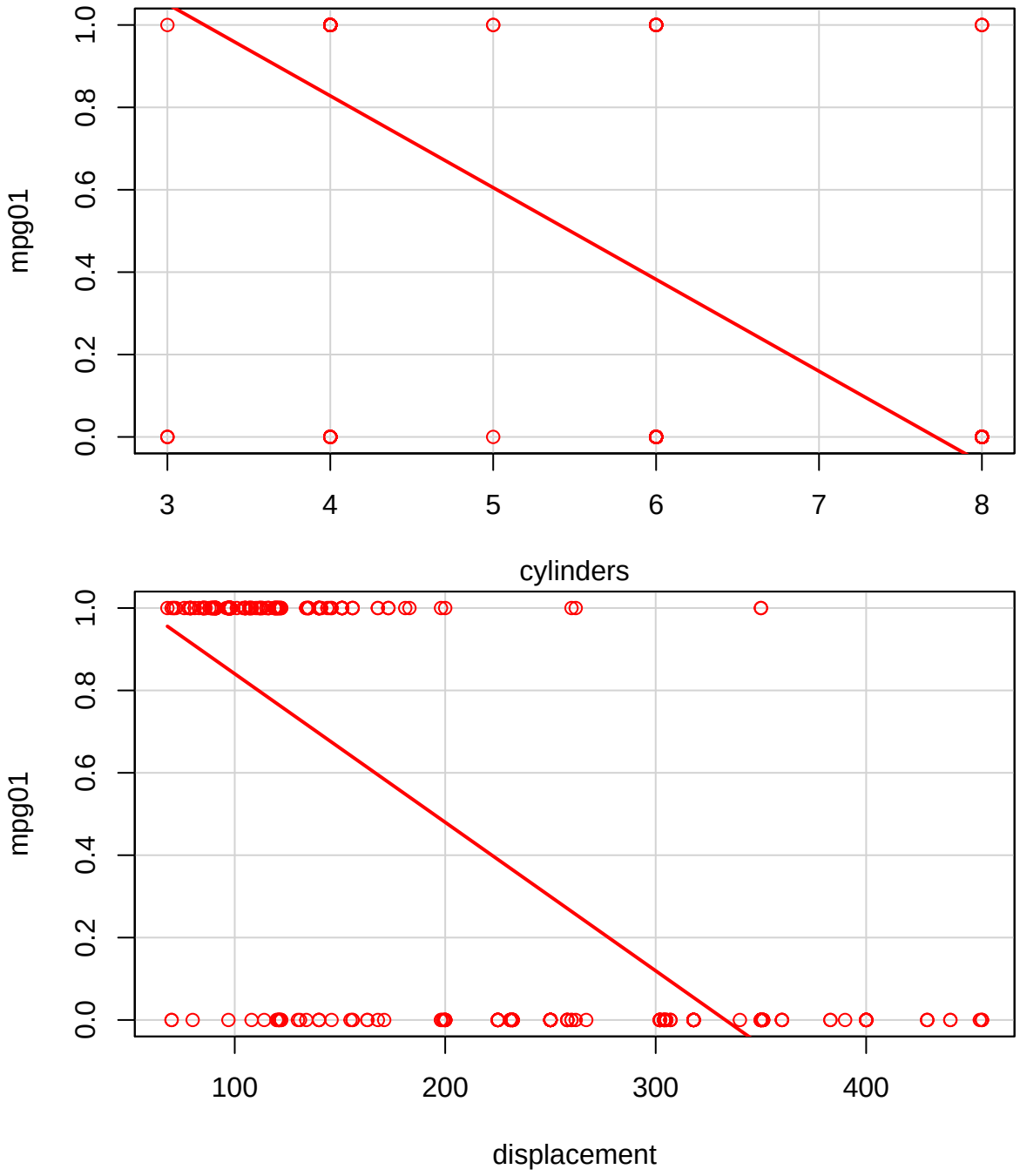
```
## 'data.frame':    392 obs. of  9 variables:
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ mpg01       : num  0 0 0 0 0 0 0 0 0 0 ...
```

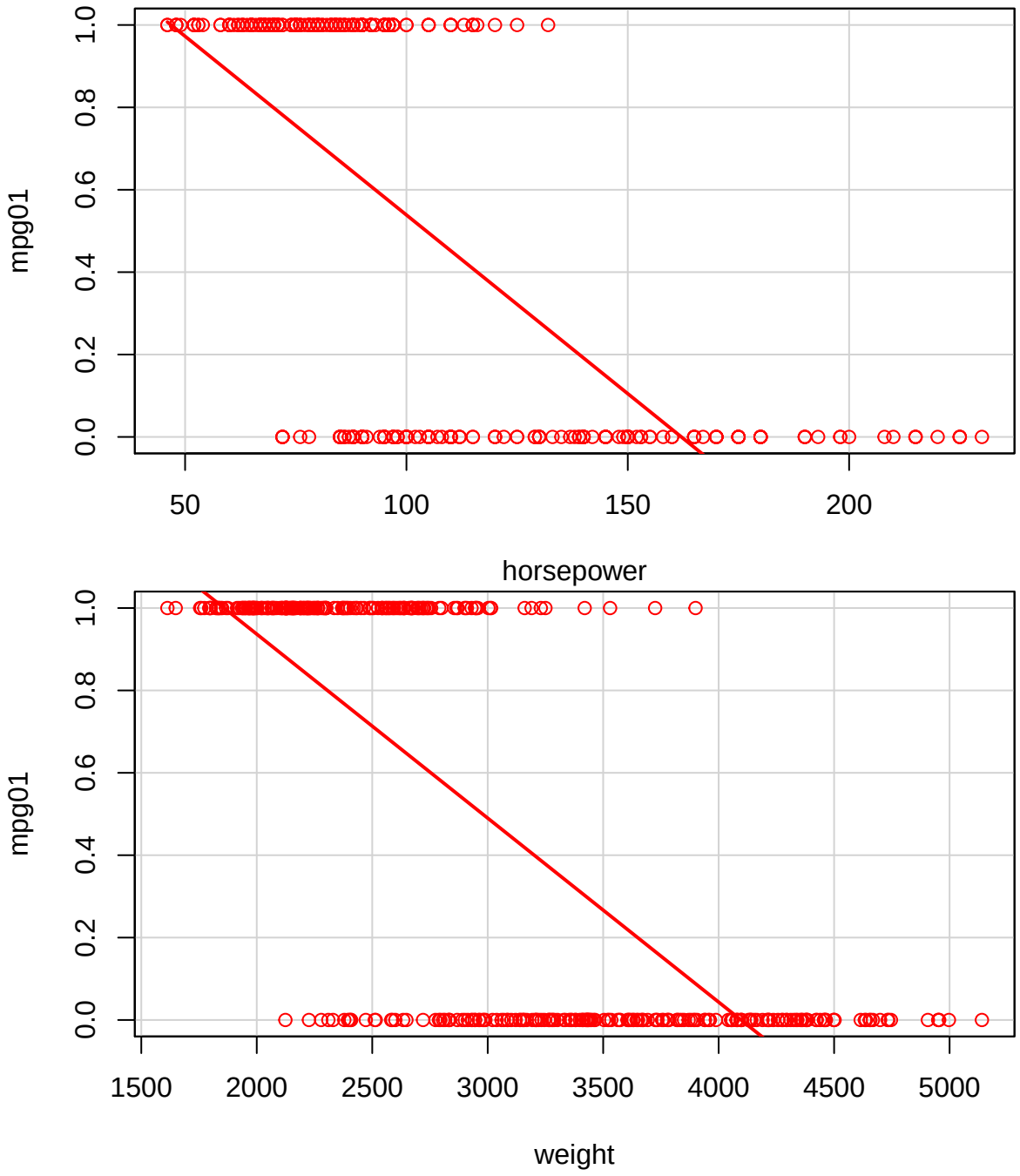
(b) Explore the data graphically in order to investigate the association between `mgp01` and the other features.

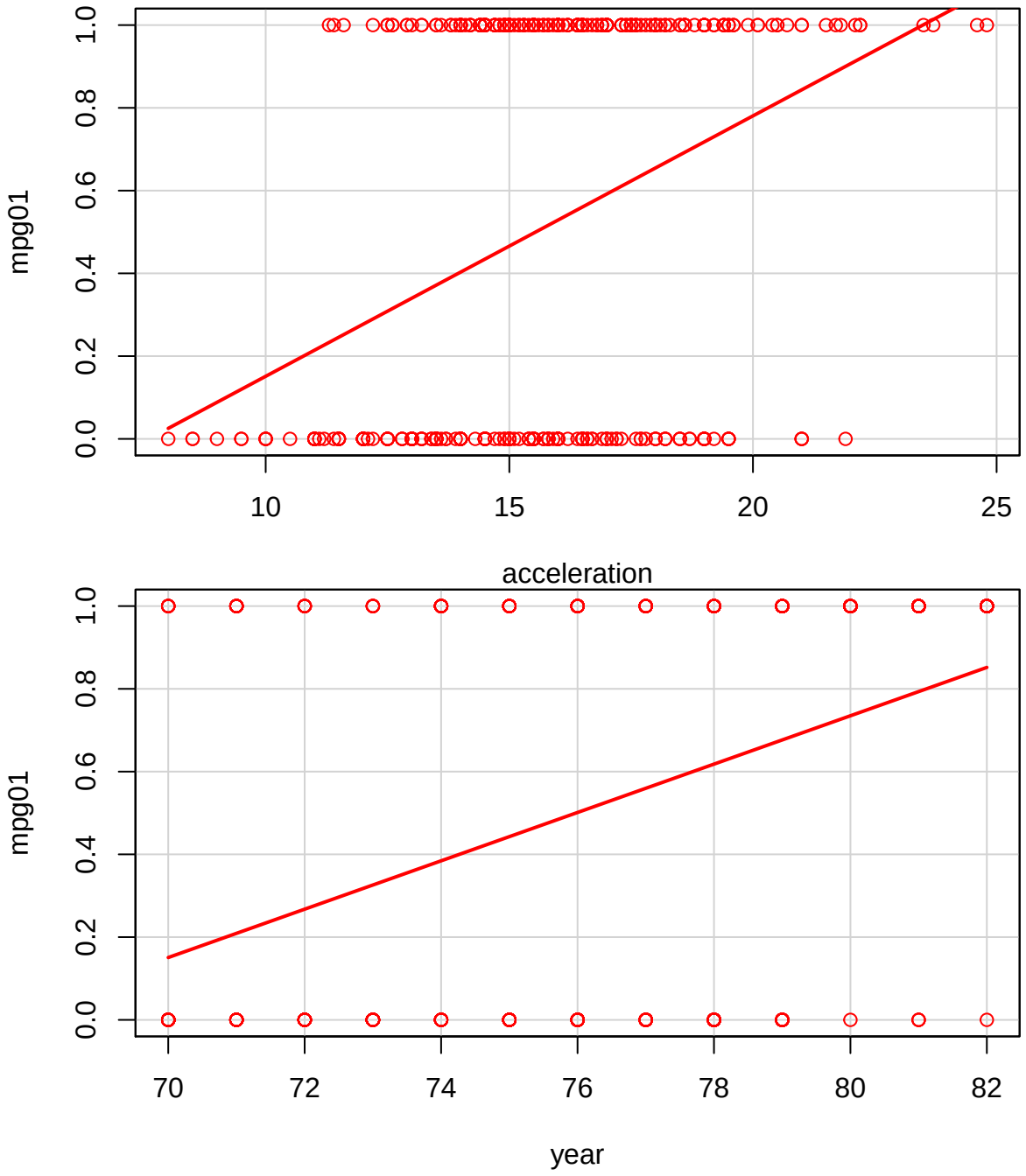
```
# Scatterplots of each variable with mpg01
library(car)
```

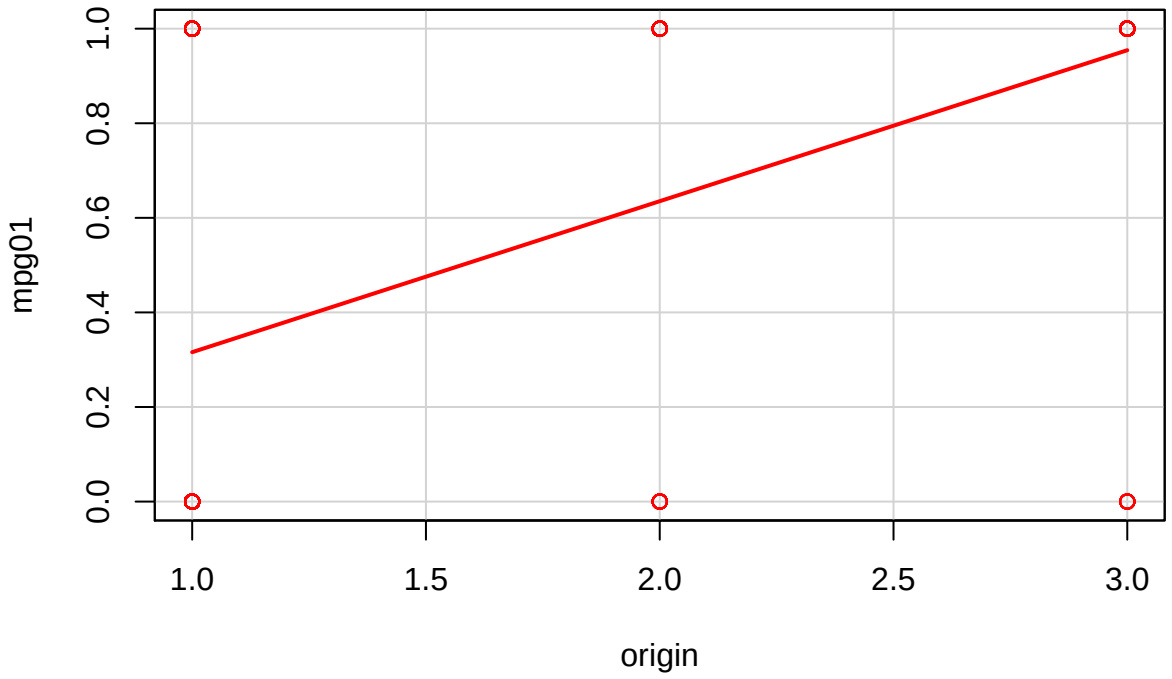
```
## Loading required package: carData
```

```
for (i in 2:8){
  scatterplot(mpg01~., regLine = T, smooth = F, boxplot = F, col= "red", data = auto[c(i,9)])
}
```


mpg01
0.0
0.2
0.4
0.6
0.8
1.0
3
4
5
6
7
8
cylinders
mpg01
0.0
0.2
0.4
0.6
0.8
1.0
100
200
300
400
displacement


mpg01
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
horsepower
mpg01
0.0
0.2
0.4
0.6
0.8
1.0
1500
2000
2500
3000
3500
4000
4500
5000
weight


mpg01
0.0
0.2
0.4
0.6
0.8
1.0
10
15
20
25
acceleration
mpg01
0.0
0.2
0.4
0.6
0.8
1.0
70
72
74
76
78
80
82
year



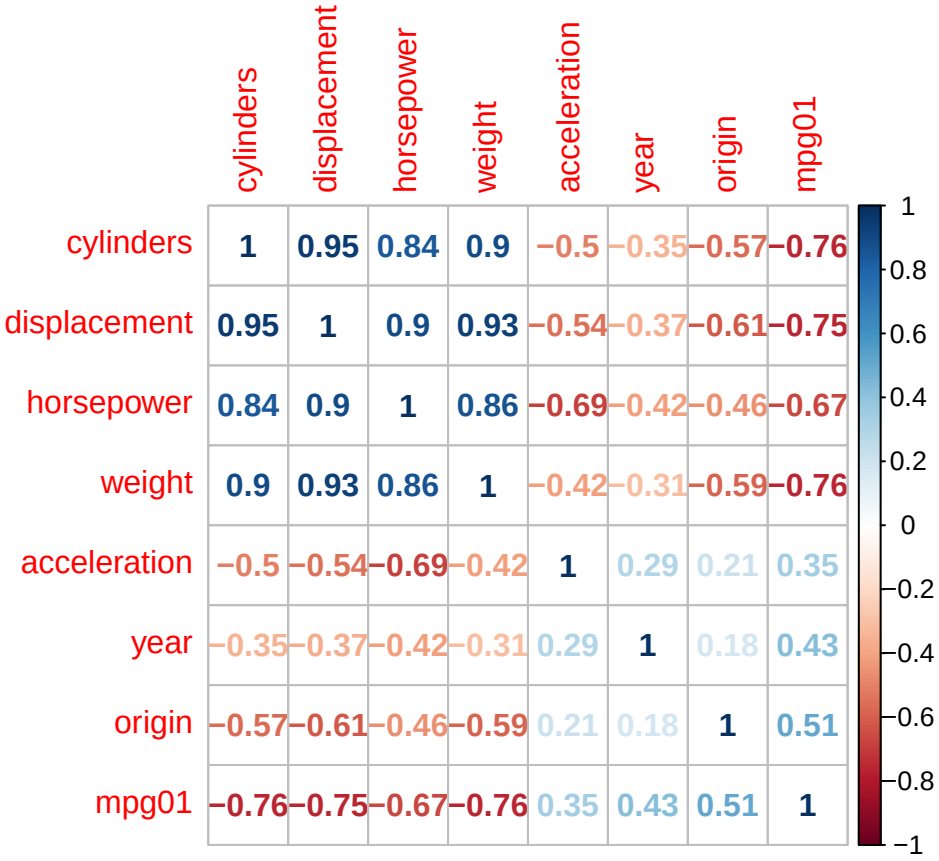

```
# Correlations
library(corrplot)
```

```
## corrplot 0.84 loaded
```

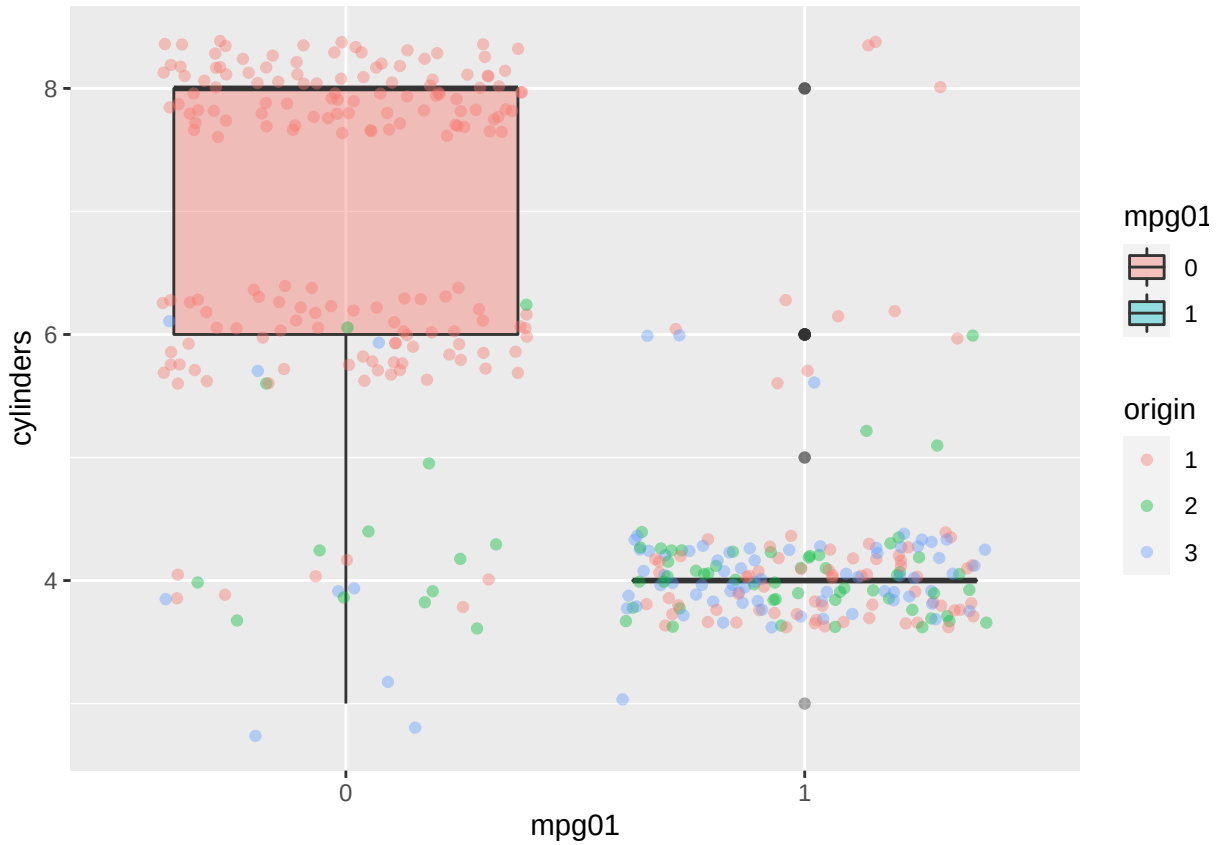
```
##
## Attaching package: 'corrplot'
```

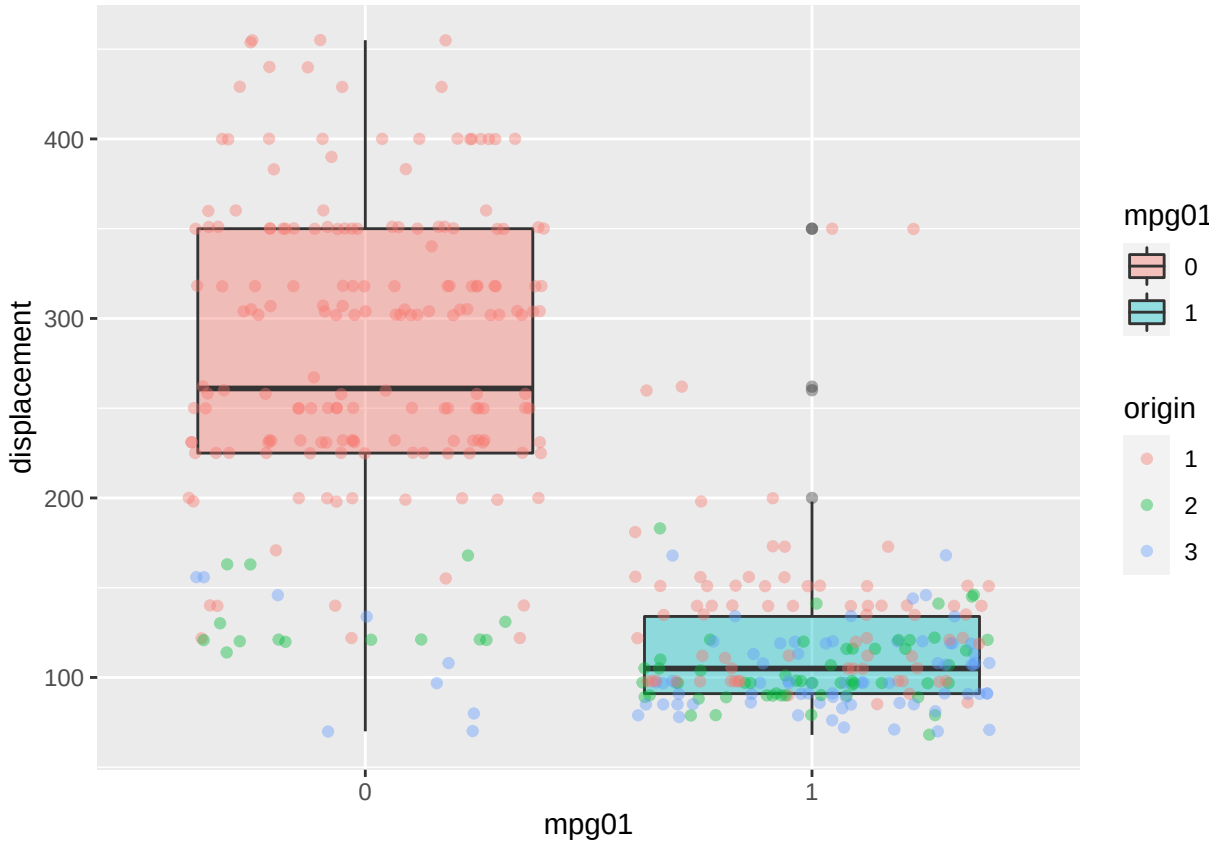
```
## The following object is masked from 'package:pls':
##
##     corrplot
```

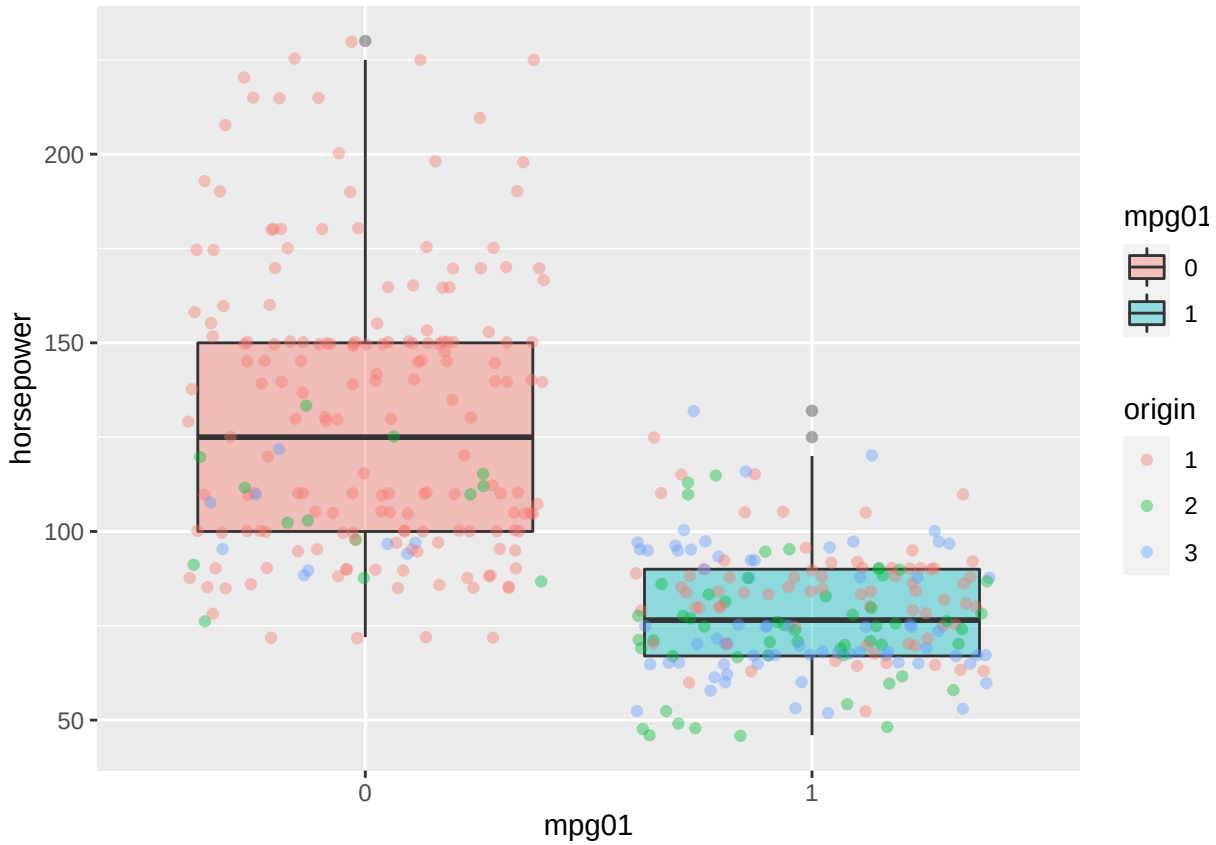
```
corrplot(cor(auto[-1]), method = "number")
```

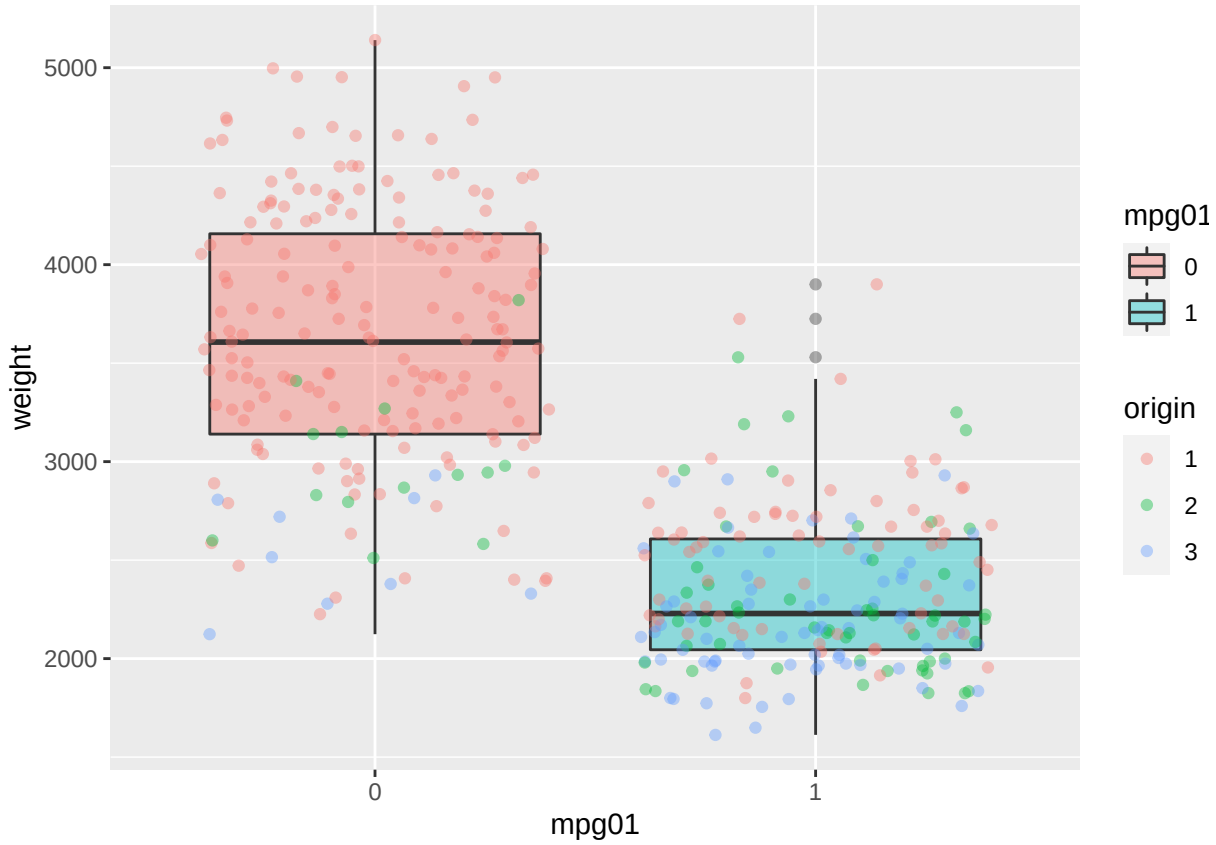


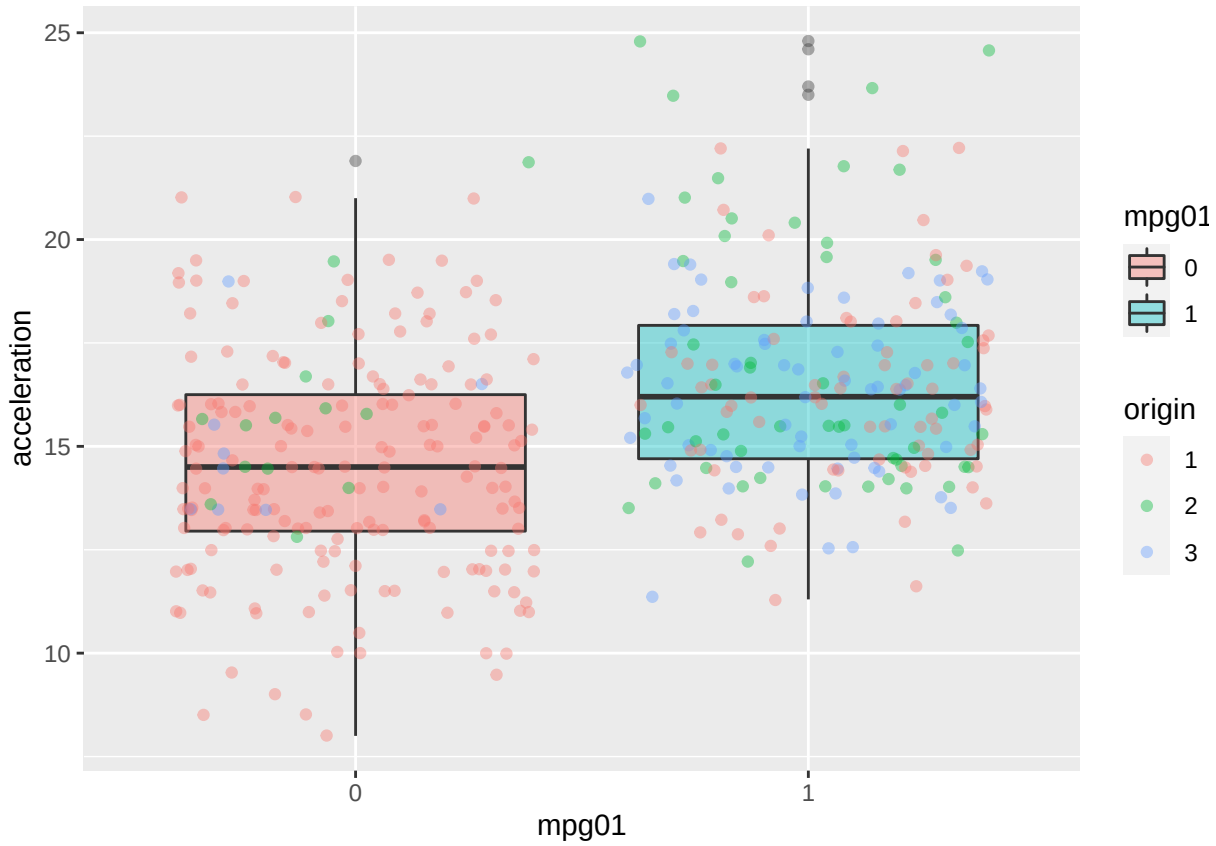

```
# Boxplots

library(ggplot2)
# First, convert the discrete variable to factor type
auto$mpg01 <- as.factor(auto$mpg01)
auto$origin <- as.factor(auto$origin)

# Boxplots with Jittered points
for(i in 2:7){
  print(ggplot(data = auto, aes_string("mpg01", colnames(auto)[i]))+
          geom_boxplot(aes(fill = mpg01), alpha = 0.4) +
          geom_jitter(aes(col=origin), alpha = 0.4))
}
```

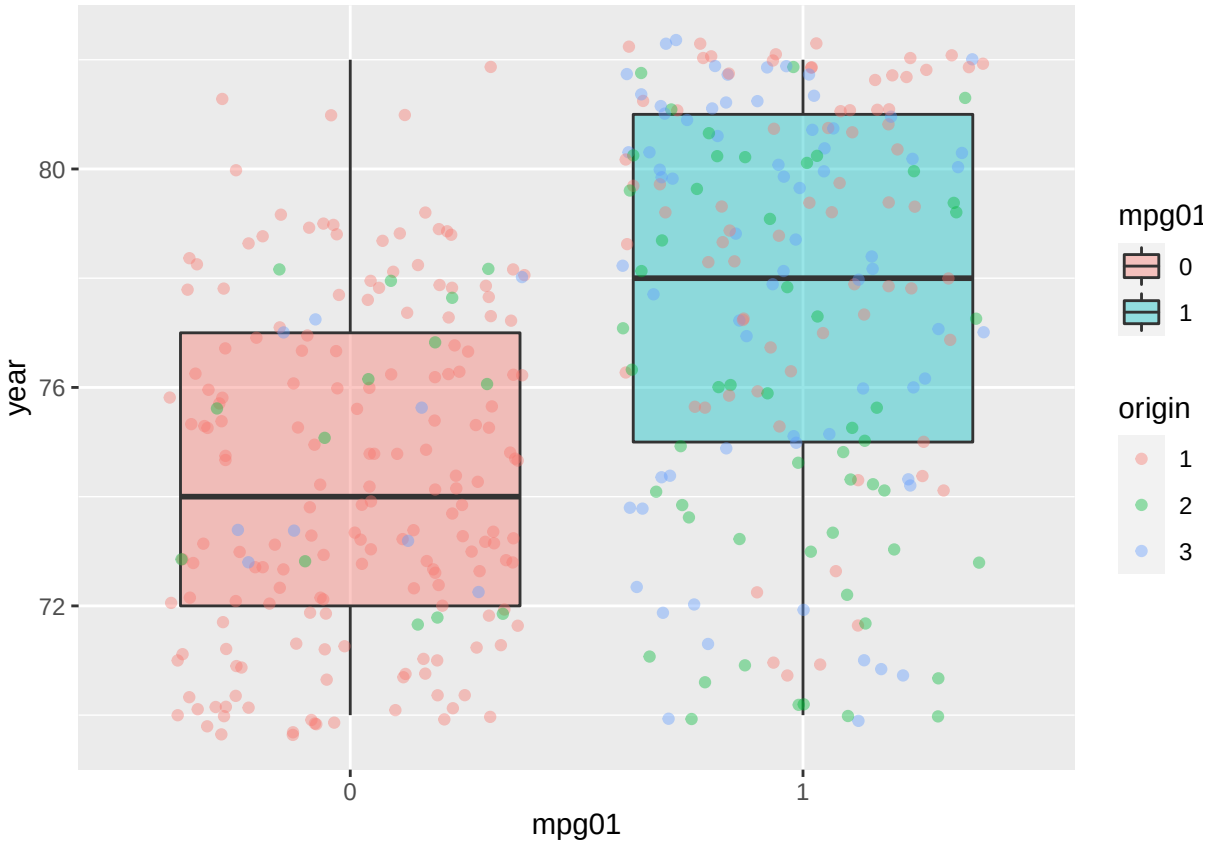

8
6
4
cylinders
0
1
mpg01
mpg01
0
1
origin
1
2
3


400
300
200
100
displacement
0
1
mpg01
mpg01
0
1
origin
1
2
3


horsepower
200
150
100
50
0
1
mpg01
mpg01
0
1
origin
1
2
3


weight
5000
4000
3000
2000
0
1
mpg01
mpg01
0
1
origin
1
2
3


25
20
15
10
acceleration
0
1
mpg01
mpg01
0
1
origin
1
2
3


80
76
72
year
0
1
mpg01
mpg01
0
1
origin
1
2
3

(c) Split the data into a training set and a test set.

```
set.seed(123)
train <- sample(1:nrow(auto), nrow(auto)/1.5)
test <- (-train)
```

(d) Perform LDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b).

```
# Fit LDA model
auto.lda <- lda(mpg01~., data = auto[train,-c(1,2,3,6)])
auto.lda
```

```
## Call:
## lda(mpg01 ~ ., data = auto[train, -c(1, 2, 3, 6)])
##
## Prior probabilities of groups:
##         0         1
## 0.4904215 0.5095785
##
## Group means:
##   horsepower  weight     year   origin2   origin3
## 0  131.03906 3621.82 74.33594 0.0625000 0.0390625
## 1   77.90226 2309.91 77.85714 0.2857143 0.3458647
##
## Coefficients of linear discriminants:
##                      LD1
## horsepower  0.004734674
## weight     -0.001600074
## year        0.157034808
## origin2     0.800377619
## origin3     0.546303105
```

```
# confusion matrix
lda.pred <- predict(auto.lda, auto[test, -c(1,2,3,6)])
lda.table <- table(lda.pred$class, auto[test, 9])
print(lda.table)
```

```
##
##      0  1
##   0 58  5
##   1 10 58
```

```
# Test error
paste("Test error is", round(1-mean(lda.pred$class == auto$mpg01[test]),3))
```

```
## [1] "Test error is 0.115"
```

(e) Perform QDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b).

```r
# Fit QDA model
auto.qda <- qda(mpg01~., data = auto[train,-c(1,2,3,6)])
auto.qda
```

```
## Call:
## qda(mpg01 ~ ., data = auto[train, -c(1, 2, 3, 6)])
##
## Prior probabilities of groups:
##         0         1
## 0.4904215 0.5095785
##
## Group means:
##   horsepower  weight     year   origin2   origin3
## 0  131.03906 3621.82 74.33594 0.0625000 0.0390625
## 1   77.90226 2309.91 77.85714 0.2857143 0.3458647
```

```r
# confusion matrix
qda.pred <- predict(auto.qda, auto[test, -c(1,2,3,6)])
qda.table <- table(qda.pred$class, auto[test, 9])
print(qda.table)
```

```
##
##      0  1
##   0 58  8
##   1 10 55
```

```r
# Test error
paste("Test error is", round(1-mean(qda.pred$class == auto$mpg01[test]),3))
```

```
## [1] "Test error is 0.137"
```

(f) Perform logistic regression on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b).

```r
# Fit Logistic Model
auto.logit <- glm(mpg01~., data = auto[train,-c(1,2,3,6)], family = binomial)
summary(auto.logit)
```

```
##
## Call:
## glm(formula = mpg01 ~ ., family = binomial, data = auto[train,
##     -c(1, 2, 3, 6)])
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.02397  -0.06329   0.01477   0.17449   2.35819
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -24.112933   6.693774  -3.602 0.000315 ***
## horsepower   -0.048414   0.022375  -2.164 0.030479 *
```

```
## weight       -0.004665   0.001004  -4.645 3.41e-06 ***
## year          0.541667   0.104892   5.164 2.42e-07 ***
## origin2       1.511329   0.720363   2.098 0.035904 *
## origin3       0.746254   0.752296   0.992 0.321213
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 361.727  on 260  degrees of freedom
## Residual deviance:  90.849  on 255  degrees of freedom
## AIC: 102.85
##
## Number of Fisher Scoring iterations: 8
```

```
# Compute the probability of mpg being bigger than its median given X
auto.probs <- predict(auto.logit, auto[test,], type = "response")

# Convert the probabilities into class labels
auto.pred <- rep("0", 131)
auto.pred[auto.probs > .5] = "1"

# Confusion matrix
logit.table <- table(auto.pred, auto$mpg01[test])
print(logit.table)
```

```
##
## auto.pred  0  1
##         0 63  9
##         1  5 54
```

```
# Test error
mean(auto.pred == auto$mpg01[test])
```

```
## [1] 0.8931298
```

```
paste("Test error:", round(1-mean(auto.pred == auto$mpg01[test]), 3))
```

```
## [1] "Test error: 0.107"
```

Comment:
I selected the variables based on the scatteplots, box and jitter plots, and correlations:
"horsepower", "weight", "year," and "origin"
I dropped "Cylinder" is highly correlated with most of the other variables
and it does not seem to be significantly associated with the response on the basis of
the scatter plot. The boxplot also shows it has outliers.
The boxplot presents that "accelaration" is not very differentiated
by the different class of mpg01.
In the case of "horsepower" and "displacement" which are the variables highly correlated
with each other, I decided to drop the displacement variable because it is not notably
different by the dichotomous class according to the scatterplot.
As a result, the logistic model generates the best result in terms of the smallest test error
among the three models.