Coursework 2

Baran Buluttekin

Big Data Analytics using R

- **Programme:** MSc Data Science
- **Student ID:** 13153116

1. Decision Trees

(a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of the figure above. The numbers inside the boxes indicate the mean of Y within each region.

Please check the pdf file for the first diagram!

(b) Create a diagram similar to the left-hand panel of the figure, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.

```r
plot(NA, NA, type = "n", xlim = c(-1,4), ylim = c(-1,4), xlab = "X1", ylab = "X2")
# X2 < 1
lines(x = c(-1,4), y = c(1,1))
text(x = 4.1, y = 1, labels = c("1"), col = "red")
# X1 <1
lines(x = c(1,1), y = c(-1,1))
text(x = 1, y = 1.5, labels = c("1"), col = "red")
# X2 < 2
lines(x = c(-1,4), y = c(2,2))
text(x = 4.1, y = 2, labels = c("2"), col = "red")
# X1 < 0
lines(x = c(0,0), y = c(1,2))
text(x = 0, y = 2.5, labels = c("0"), col = "red")
# Labelling regions
text(x = 0, y = 0, labels = c("-1.80"))
text(x = 2.5, y = 0, labels = c("0.63"))
text(x = -0.5, y = 1.5, labels = c("-1.06"))
text(x = 2, y = 1.5, labels = c("0.21"))
text(x = 1.5, y = 3, labels = c("2.49"))
```

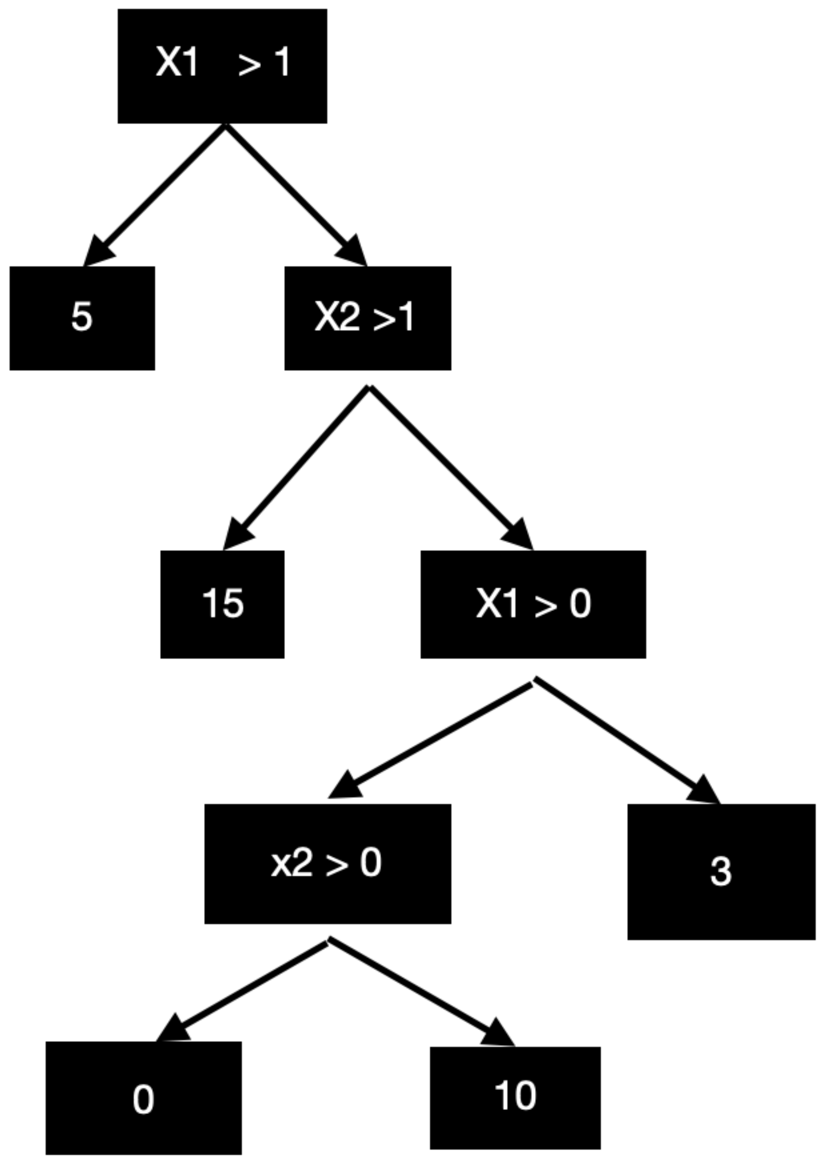
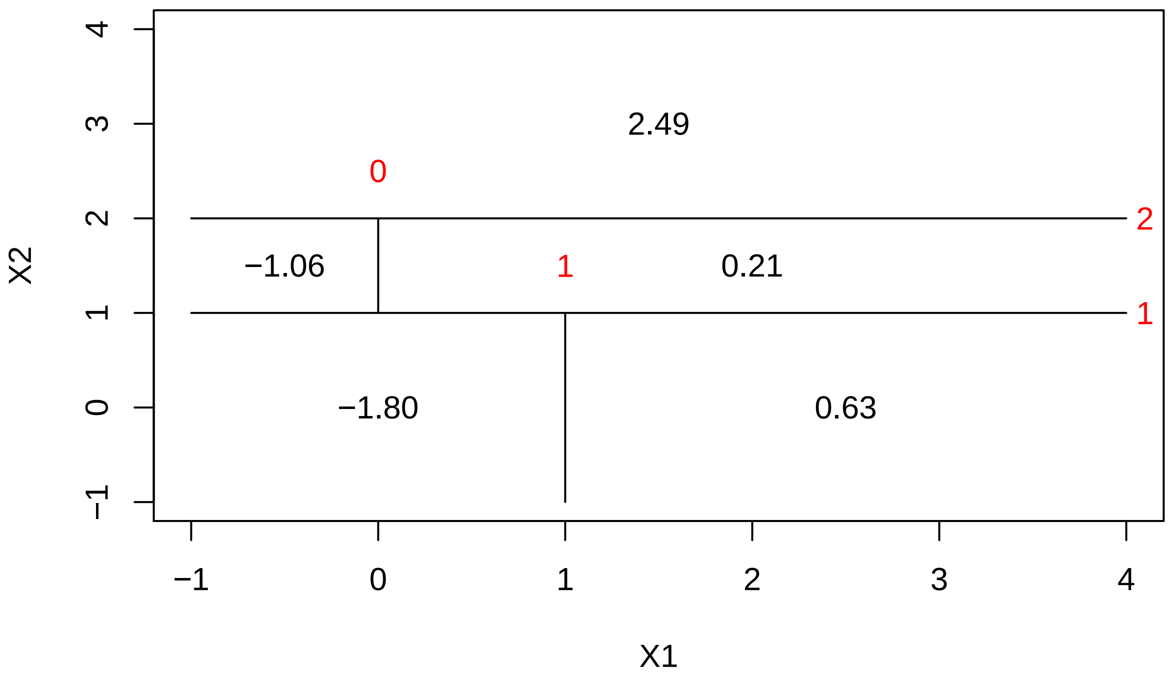


Figure 1:



2. Regression Trees

In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

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

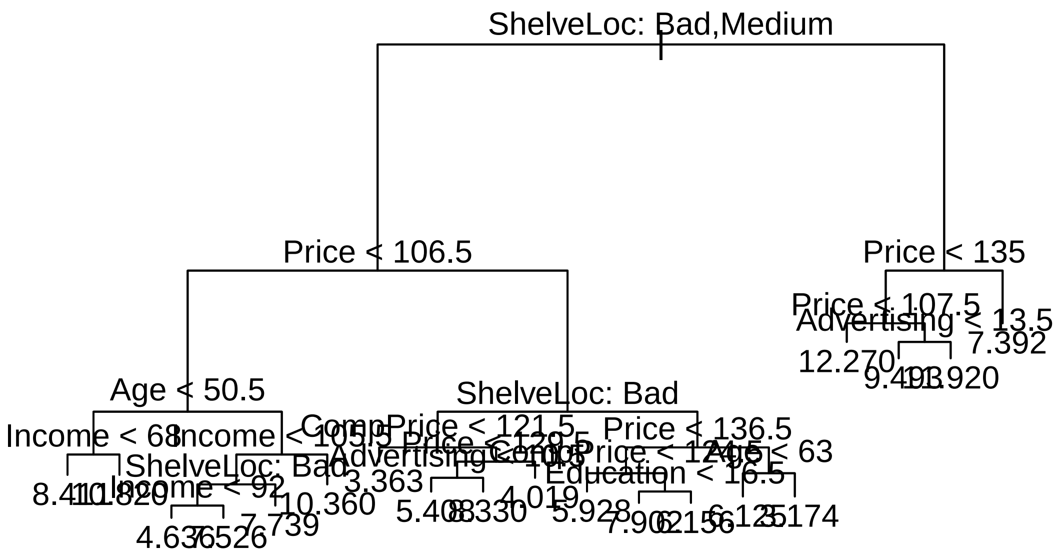
```
library(ISLR)
library(tree)
set.seed(123)
train <- sample(1:nrow(Carseats), nrow(Carseats) * 3 / 4)
df.train <- Carseats[train, ]
df.test <- Carseats[-train, ]
```

(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test error rate do you obtain?

```
tree.carseats <- tree(Sales ~ ., data = df.train)
summary(tree.carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = df.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising" "Education"
## Number of terminal nodes:  19
## Residual mean deviance:  2.306 = 647.9 / 281
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.17800 -1.02800 -0.03362  0.00000  0.96300  4.22800
```

```r
plot(tree.carseats)
text(tree.carseats, pretty = 0)
```



```r
tree.pred <- predict(tree.carseats, newdata = df.test)
mean((tree.pred - df.test$Sales)^2)
```

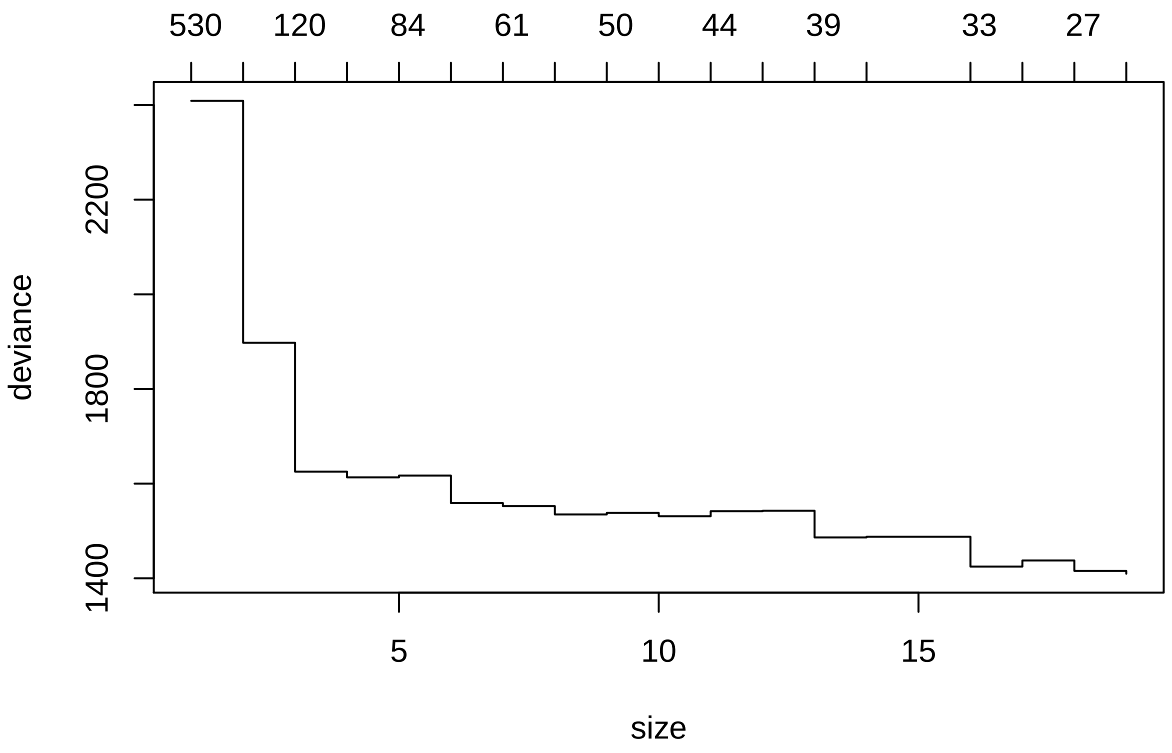
```
## [1] 4.276773
```

```r
print(paste("MSE for the train set is around",
            round(mean((tree.pred - df.test$Sales)^2), digits = 2)))
```

```
## [1] "MSE for the train set is around 4.28"
```

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test error rate?

```r
cv.carseats <- cv.tree(tree.carseats)
plot(cv.carseats)
```

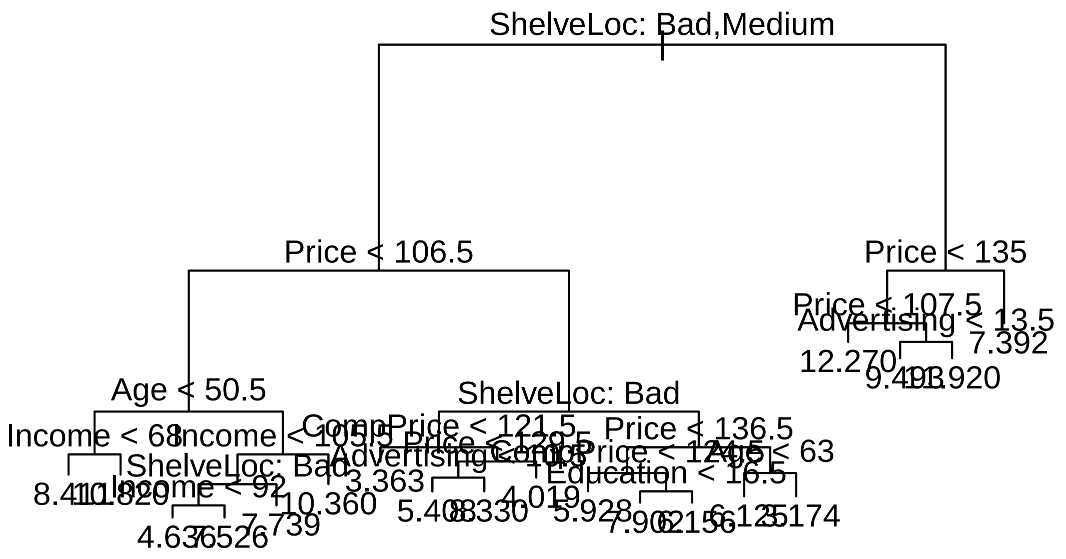


```
min <- which.min(cv.carseats$dev)
cv.carseats$dev[min]
```

```
## [1] 1409.598
```

We can choose min size from cross validation to prune the tree.

```
prune.carseats <- prune.tree(tree.carseats, best = cv.carseats$size[min])
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```



```
cv.pred <- predict(prune.carseats, newdata = df.test)
mean((cv.pred - df.test$Sales)^2)
```

```
## [1] 4.276773
```

In this case pruning the tree increased the MSE to 4.27.

(d) Use the bagging approach in order to analyze this data. What test error rate do you obtain? Use the importance() function to determine which variables are most important.

```r
require(randomForest)
```

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
bag.carseats <- randomForest(Sales ~ ., data = df.train,
                             mtry = 10, ntree = 500, importance = TRUE)
bag.pred <- predict(bag.carseats, newdata = df.test)
mean((bag.pred - df.test$Sales)^2)
```

```
## [1] 2.24857
```

MSE is decreased to as low as 2.24 after we applied random forest algorithm.

```r
importance(bag.carseats)
```

```
##               %IncMSE IncNodePurity
## CompPrice   31.216418     231.68911
## Income      12.515537     130.18293
## Advertising 18.583823     145.55527
## Population  -2.476697      70.33574
## Price       71.719968     728.65376
## ShelveLoc   71.982774     676.78616
## Age         26.414235     262.59519
## Education    2.835862      66.53615
## Urban        1.321304      10.69032
## US           3.309603      10.12666
```

From the table above we can clearly observe that `Price` and `ShelveLoc` are by far most important variables.

(e) Use random forests to analyze this data. What test error rate do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```r
rf.carseats <- randomForest(Sales ~ ., data = df.train,
                            mtry = 3, ntree = 500, importance = TRUE)
rf.pred <- predict(rf.carseats, newdata = df.test)
mean((rf.pred - df.test$Sales)^2)
```

```
## [1] 2.691865
```

By selecting m = $\sqrt{p}$, we obtained 2.69 MSE.

```r
importance(rf.carseats)
```

```
##               %IncMSE IncNodePurity
## CompPrice   14.165434     209.92409
## Income       4.399950     175.59871
## Advertising 15.692027     195.01607
## Population  -2.320700     139.24996
## Price       45.568124     576.29642
## ShelveLoc   47.579744     519.22724
## Age          16.041087     284.60814
## Education     1.413292     102.35397
## Urban        -1.149342      19.82060
## US            3.788742      26.92051
```

Similar to subsection (d) `Price` and `ShelveLoc` are the most important variables.

3. Classification Trees

This problem involves the OJ data set which is part of the ISLR package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
set.seed(123)
train <- sample(1:nrow(OJ), 800)
train.oj <- OJ[train, ]
test.oj <- OJ[-train, ]
```

(b) Fit a tree to the training data, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

```
tree.oj <- tree(Purchase ~ ., data = train.oj)
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train.oj)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff" "SpecialCH" "PctDiscMM"
## Number of terminal nodes:  10
## Residual mean deviance:  0.7289 = 575.8 / 790
## Misclassification error rate: 0.1612 = 129 / 800
```

Misclassification train error rate is 0.161 Number of terminal nodes:10

(c) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

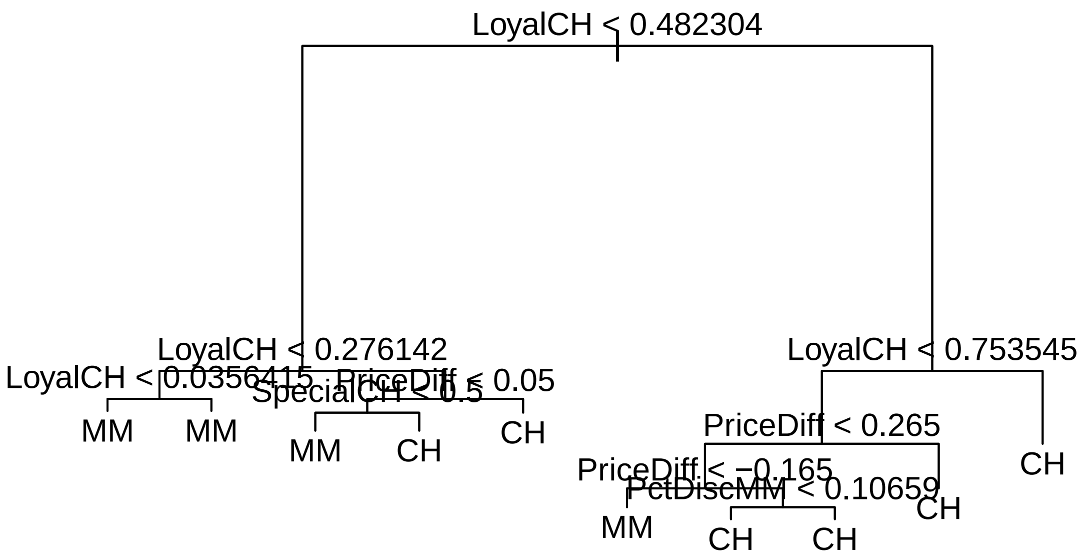
```
tree.oj
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.000 CH ( 0.60500 0.39500 )
##    2) LoyalCH < 0.482304 299  320.600 MM ( 0.22742 0.77258 )
##      4) LoyalCH < 0.276142 172  127.600 MM ( 0.12209 0.87791 )
##        8) LoyalCH < 0.0356415 56   10.030 MM ( 0.01786 0.98214 ) *
##        9) LoyalCH > 0.0356415 116  106.600 MM ( 0.17241 0.82759 ) *
##      5) LoyalCH > 0.276142 127  167.400 MM ( 0.37008 0.62992 )
##       10) PriceDiff < 0.05 58   59.140 MM ( 0.20690 0.79310 )
##         20) SpecialCH < 0.5 51   36.950 MM ( 0.11765 0.88235 ) *
##         21) SpecialCH > 0.5 7    5.742 CH ( 0.85714 0.14286 ) *
##       11) PriceDiff > 0.05 69   95.640 CH ( 0.50725 0.49275 ) *
##    3) LoyalCH > 0.482304 501  456.300 CH ( 0.83034 0.16966 )
##      6) LoyalCH < 0.753545 236  292.000 CH ( 0.69068 0.30932 )
##       12) PriceDiff < 0.265 147  202.300 CH ( 0.55102 0.44898 )
##         24) PriceDiff < -0.165 40   47.050 MM ( 0.27500 0.72500 ) *
##         25) PriceDiff > -0.165 107  138.000 CH ( 0.65421 0.34579 )
##           50) PctDiscMM < 0.10659 75  102.900 CH ( 0.56000 0.44000 ) *
##           51) PctDiscMM > 0.10659 32   24.110 CH ( 0.87500 0.12500 ) *
##       13) PriceDiff > 0.265 89   49.030 CH ( 0.92135 0.07865 ) *
##      7) LoyalCH > 0.753545 265   97.720 CH ( 0.95472 0.04528 ) *
```

Terminal nodes are denoted with asterisk (*). I pick the number 9 node, which splitted in LoyalCH > 0.036, there are 116 observations in this branch with deviance of 106.600. Around 17% of the observations in that branch belong to CH and the remaining observations (83%) takes MM value.

(d) Create a plot of the tree, and interpret the results.

```r
plot(tree.oj)
text(tree.oj, pretty = 0)
```



It is clear that the brand loyalty to citrus hill `LoyalCH` is most important predictor. This can be seen from the fact its the deciding factor in root branch as wel as left and right branches after the root branch.

(e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

```r
tree.pred.oj <- predict(tree.oj, test.oj, type = "class")
table(tree.pred.oj, test.oj$Purchase)
```

```
##
## tree.pred.oj  CH  MM
##           CH 158  37
##           MM  11  64
```

```r
1 - (158 + 64) /nrow(test.oj)
```

```
## [1] 0.1777778
```

From the calculation above misclassification error rate is around 18%.

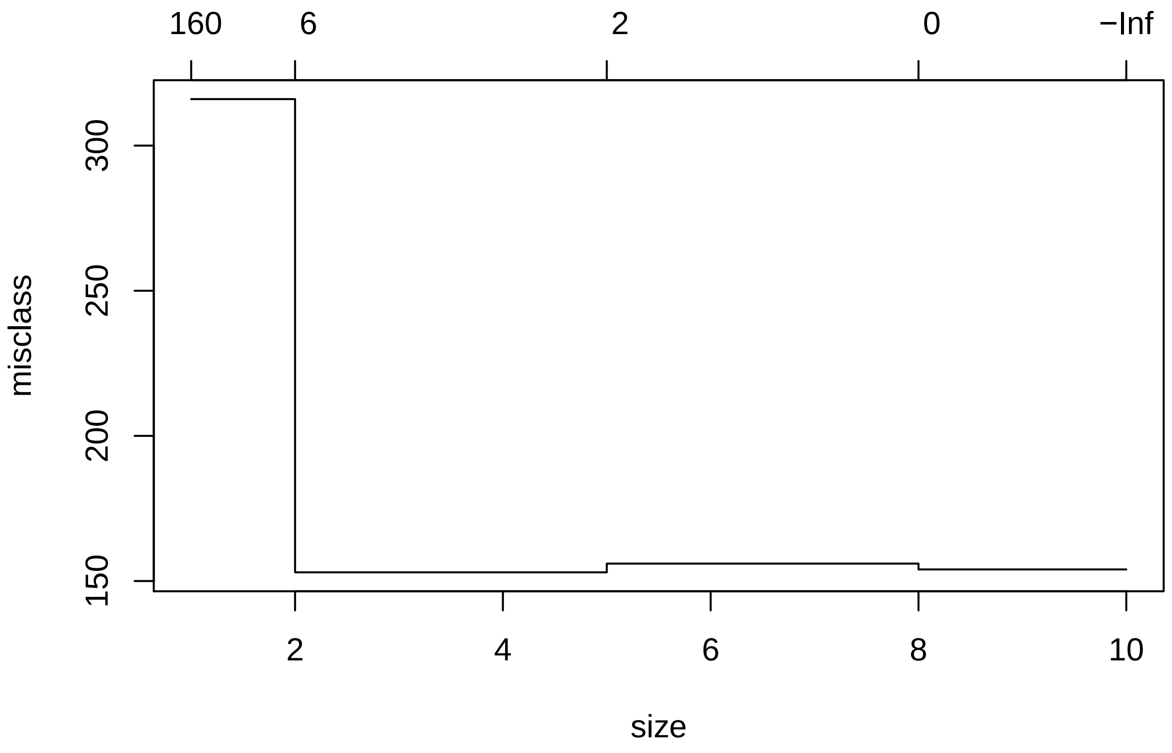
(f) Apply the `cv.tree()` function to the training set in order to determine the optimal tree size.

```r
cv.oj <- cv.tree(tree.oj, FUN = prune.misclass)
cv.oj
```

```
## $size
## [1] 10  8  5  2  1
##
## $dev
## [1] 154 154 156 153 316
##
## $k
## [1] -Inf    0    2    6  163
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

(g) Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

```
plot(cv.oj)
```

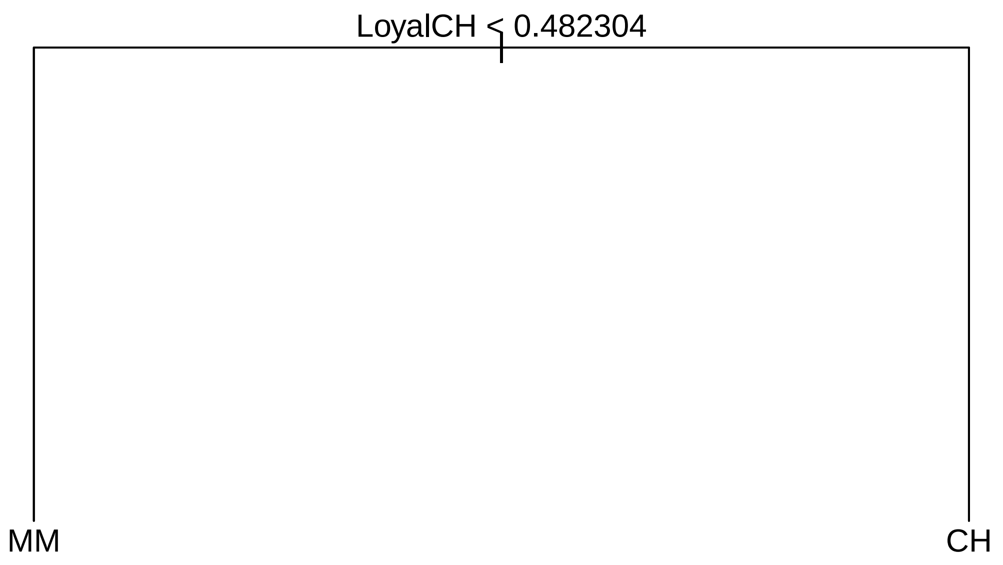


(h) Which tree size corresponds to the lowest cross-validated classification error rate?

From size 2 onwards misclassification rate is flat (with exception of 5 to 8) and we can observe that first instance of the lowest rate starts at 2.

(i) Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

```
prune.oj <- prune.misclass(tree.oj, best = 2)
plot(prune.oj)
text(prune.oj, pretty = 0)
```



(j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

```
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = train.oj)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff" "SpecialCH" "PctDiscMM"
## Number of terminal nodes:  10
## Residual mean deviance:  0.7289 = 575.8 / 790
## Misclassification error rate: 0.1612 = 129 / 800
```

```
summary(prune.oj)
```

```
##
## Classification tree:
## snip.tree(tree = tree.oj, nodes = 2:3)
## Variables actually used in tree construction:
## [1] "LoyalCH"
## Number of terminal nodes:  2
## Residual mean deviance:  0.9735 = 776.9 / 798
## Misclassification error rate: 0.1912 = 153 / 800
```

Pruning resulted in less accurate prediction in this case by incresing error rate slightly from 16% in full tree to 19% in pruned tree.

(k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

```
prune.pred <- predict(prune.oj, test.oj, type = "class")
table(prune.pred, test.oj$Purchase)
```

```
##
## prune.pred  CH  MM
##         CH 143  25
##         MM  26  76
```

```
1 - (143 + 76) / nrow(test.oj)
```

```
## [1] 0.1888889
```

Error rate in pruned tree is slightly higher (approximately 19%) then the unpruned tree (approximately 18%).

4. SVM

In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

(a) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```
Bvar <- ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
Auto$mpglevel <- as.factor(Bvar)
```

(b) Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```
library(e1071)
set.seed(123)
grid <- c(seq(0.001,0.01, length.out = 5),
          seq(0.01, 1, length.out = 5),
          seq(1, 10, length.out = 5),
          seq(10, 100, length.out = 5))
tune.svm <- tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = grid))
summary(tune.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.505
##
## - best performance: 0.01269231
##
## - Detailed performance results:
##         cost      error dispersion
## 1    0.00100 0.09666667 0.07263311
## 2    0.00325 0.08153846 0.07006319
## 3    0.00550 0.07903846 0.07076493
## 4    0.00775 0.07647436 0.06373975
## 5    0.01000 0.07647436 0.06373975
## 6    0.01000 0.07647436 0.06373975
## 7    0.25750 0.01775641 0.01700310
## 8    0.50500 0.01269231 0.01783081
## 9    0.75250 0.01525641 0.01764548
## 10   1.00000 0.01275641 0.01344780
## 11   1.00000 0.01275641 0.01344780
## 12   3.25000 0.02032051 0.01999445
## 13   5.50000 0.02032051 0.01999445
## 14   7.75000 0.02032051 0.01999445
## 15  10.00000 0.02032051 0.01999445
## 16  10.00000 0.02032051 0.01999445
## 17  32.50000 0.03307692 0.02397013
## 18  55.00000 0.03307692 0.02397013
## 19  77.50000 0.03307692 0.02397013
## 20 100.00000 0.03307692 0.02397013
```

Tuning returns us best parameter cost = 0.505

(c) Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```r
set.seed(123)
tune.svm <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = grid), gamma =
summary(tune.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.03051282
##
## - Detailed performance results:
##         cost      error dispersion
## 1    0.00100 0.53814103 0.02964970
## 2    0.00325 0.53814103 0.02964970
## 3    0.00550 0.53814103 0.02964970
## 4    0.00775 0.53814103 0.02964970
## 5    0.01000 0.53814103 0.02964970
## 6    0.01000 0.53814103 0.02964970
## 7    0.25750 0.11435897 0.08773453
## 8    0.50500 0.09666667 0.07263311
## 9    0.75250 0.09416667 0.07002059
## 10   1.00000 0.08916667 0.06732394
## 11   1.00000 0.08916667 0.06732394
## 12   3.25000 0.07903846 0.07076493
## 13   5.50000 0.07647436 0.06373975
## 14   7.75000 0.07647436 0.06373975
## 15  10.00000 0.07647436 0.06373975
## 16  10.00000 0.07647436 0.06373975
## 17  32.50000 0.05608974 0.03963690
## 18  55.00000 0.04339744 0.04018691
## 19  77.50000 0.03820513 0.03461697
## 20 100.00000 0.03051282 0.02626589
```

Returned best parameters: cost=100

```r
set.seed(123)
tune.svm <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = grid), deg
summary(tune.svm)
```

```
##
## Parameter tuning of 'svm':
##
```

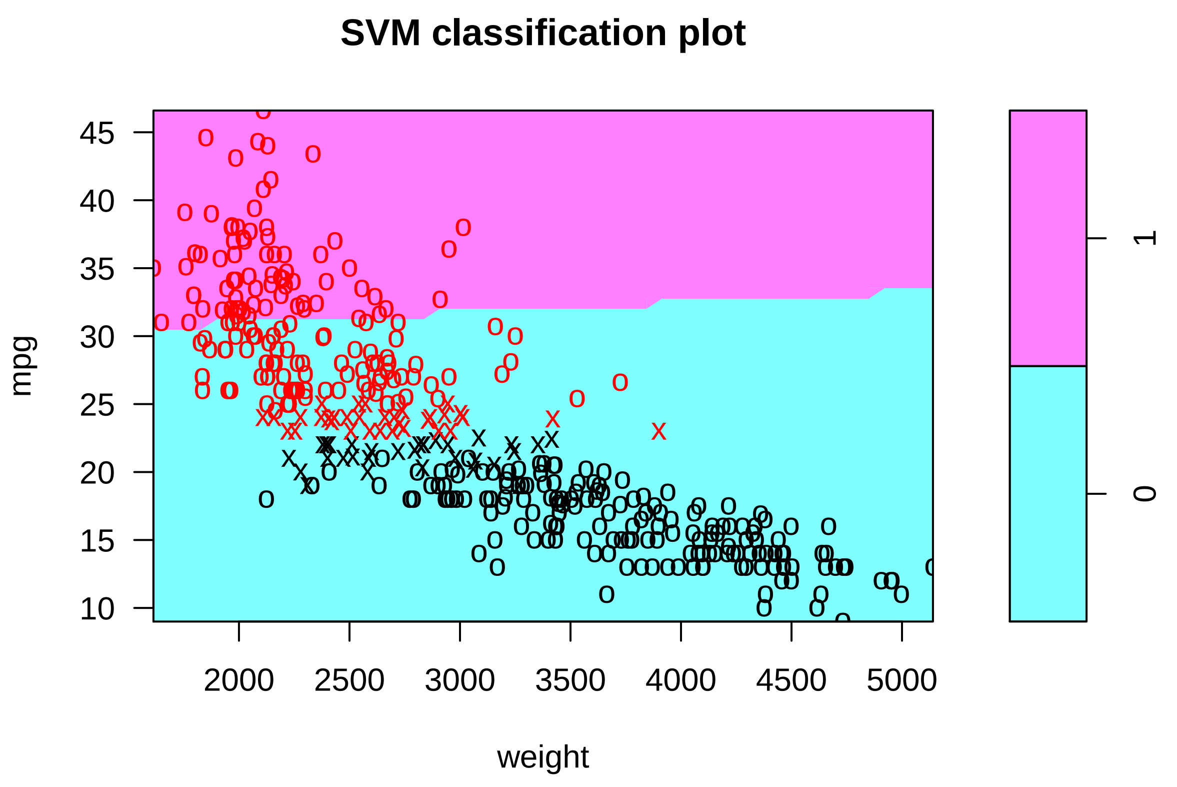
```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  77.5
##
## - best performance: 0.01775641
##
## - Detailed performance results:
##         cost      error dispersion
## 1    0.00100 0.53814103 0.02964970
## 2    0.00325 0.53814103 0.02964970
## 3    0.00550 0.53814103 0.02964970
## 4    0.00775 0.53814103 0.02964970
## 5    0.01000 0.53814103 0.02964970
## 6    0.01000 0.53814103 0.02964970
## 7    0.25750 0.10173077 0.07557946
## 8    0.50500 0.09416667 0.07002059
## 9    0.75250 0.08916667 0.06732394
## 10   1.00000 0.08153846 0.07006319
## 11   1.00000 0.08153846 0.07006319
## 12   3.25000 0.07647436 0.06373975
## 13   5.50000 0.07647436 0.06373975
## 14   7.75000 0.07647436 0.06373975
## 15  10.00000 0.07134615 0.06791784
## 16  10.00000 0.07134615 0.06791784
## 17  32.50000 0.04852564 0.04268164
## 18  55.00000 0.03564103 0.03230272
## 19  77.50000 0.01775641 0.01700310
## 20 100.00000 0.01775641 0.01700310
```

Returned best parameters: cost=77.5 and performance of polinomial kernel was better than the rbf kernel.

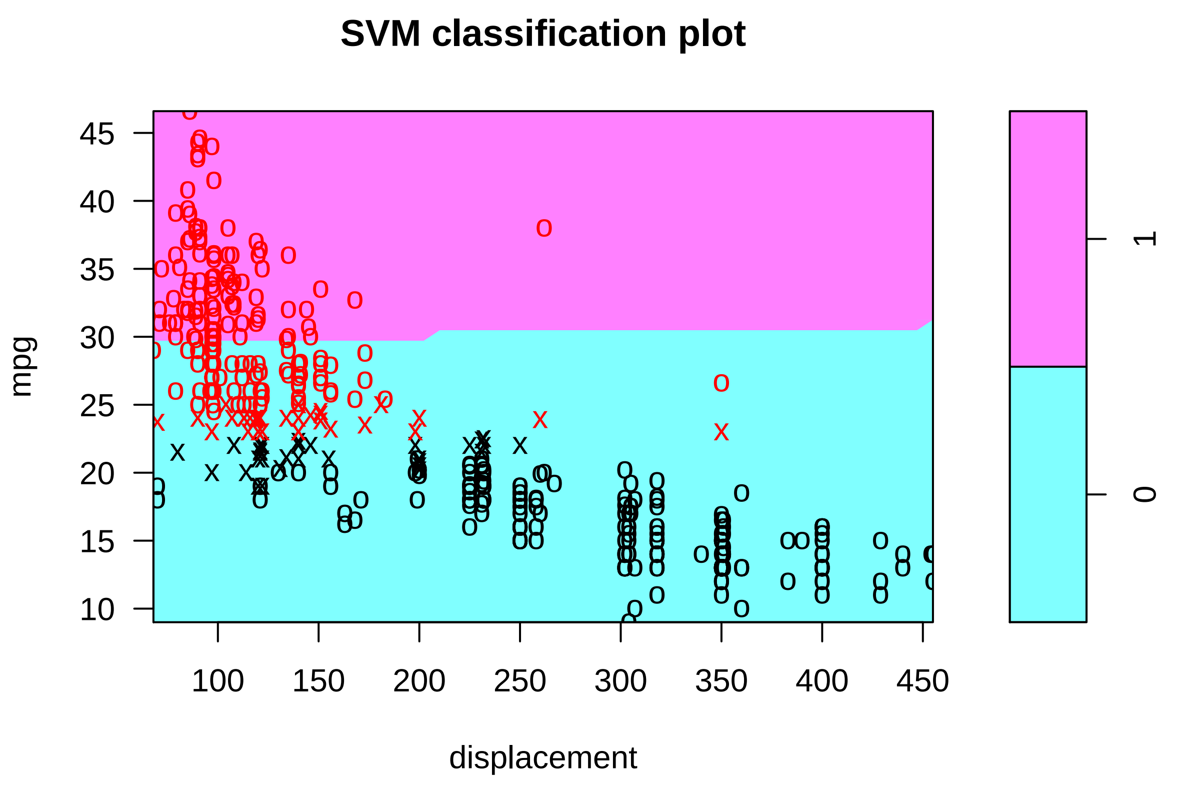
(d) Make some plots to back up your assertions in (b) and (c).

Hint: In the lab, we used the `plot()` function for svm objects only in cases with p = 2. When p > 2, you can use the `plot()` function to create plots displaying pairs of variables at a time. Essentially, instead of typing `plot(svmfit , dat)` where svmfit contains your fitted model and dat is a data frame containing your data, you can type `plot(svmfit , dat , x1~x4)` in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type `?plot.svm`.

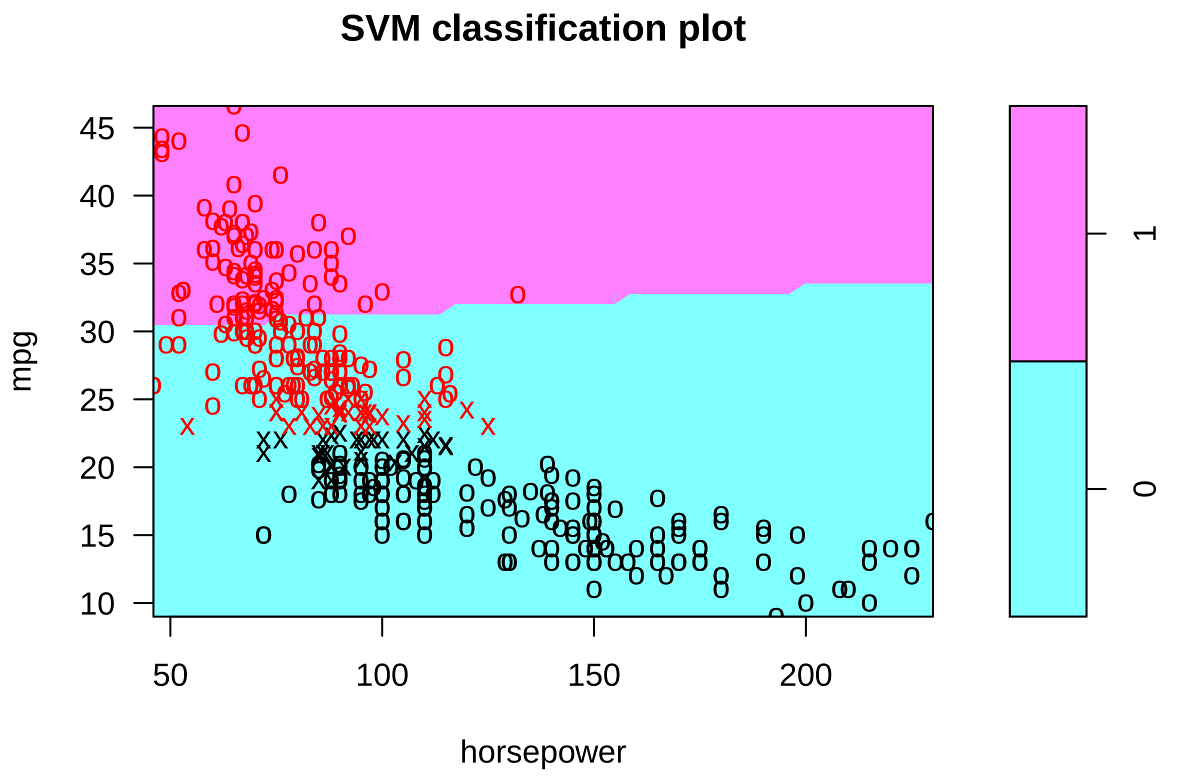
```r
svm.linear <- svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 0.505)
svm.polynomial <- svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 77.5)
svm.radial <- svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 100)
# Plots for linear kernel
plot(svm.linear, data = Auto, mpg ~ weight)
```



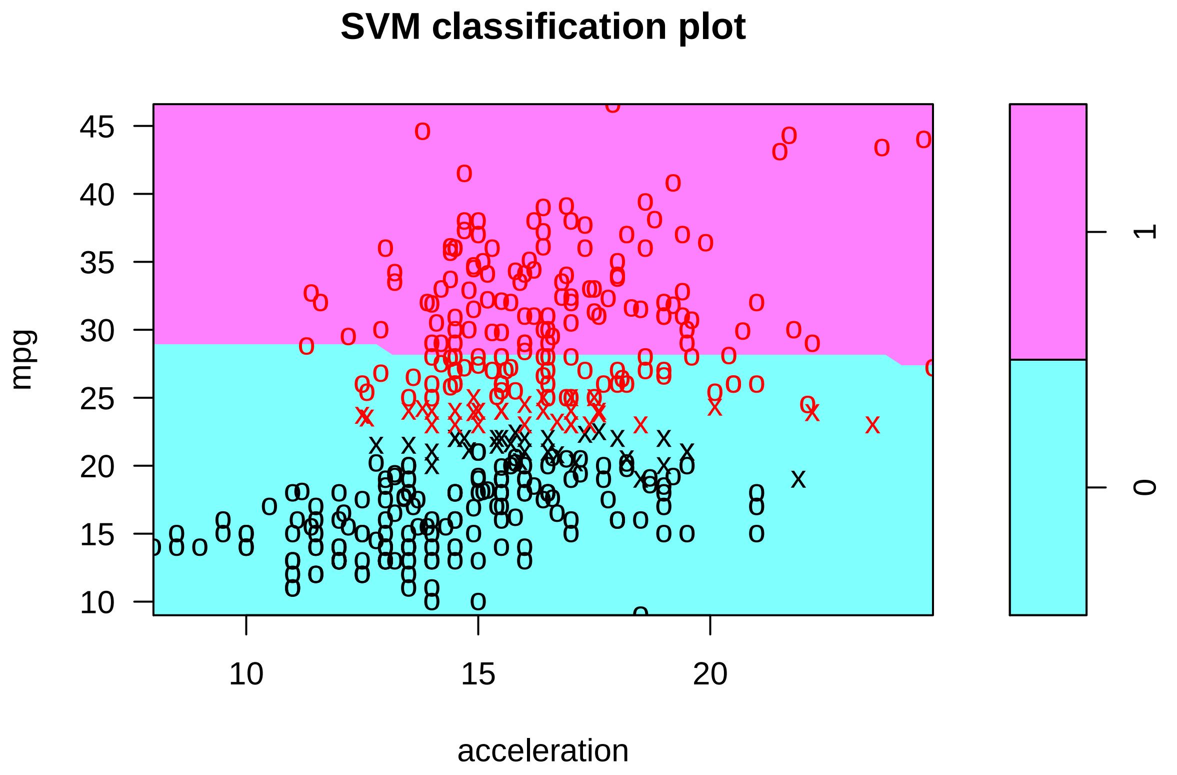
```
plot(svm.linear, Auto, mpg ~ displacement)
```



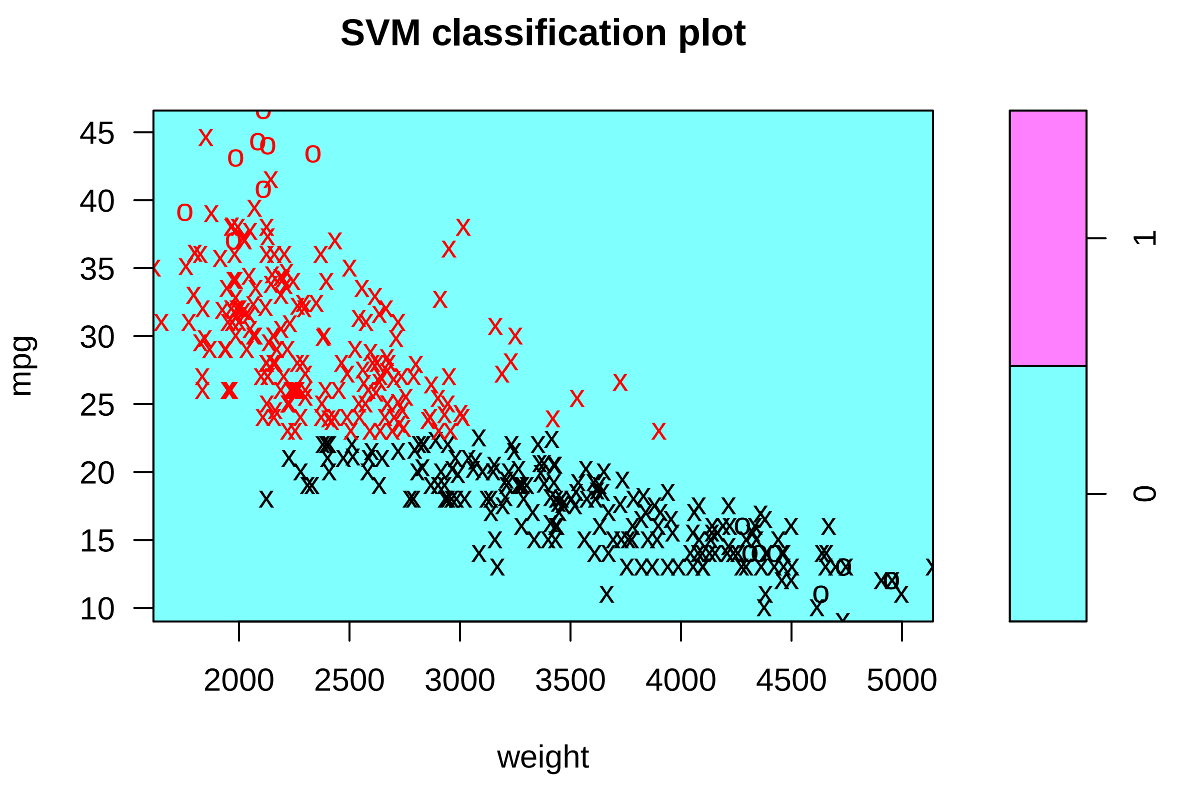
```
plot(svm.linear, Auto, mpg ~ horsepower)
```



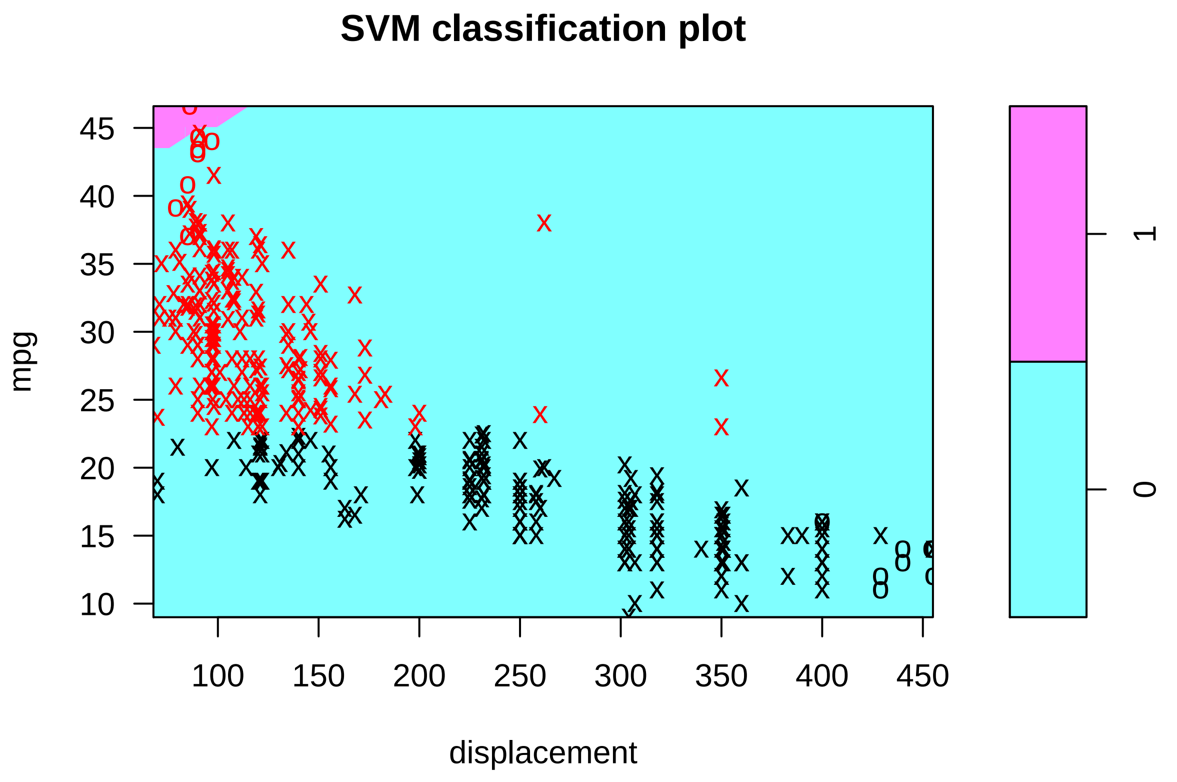
```r
plot(svm.linear, Auto, mpg ~ acceleration)
```



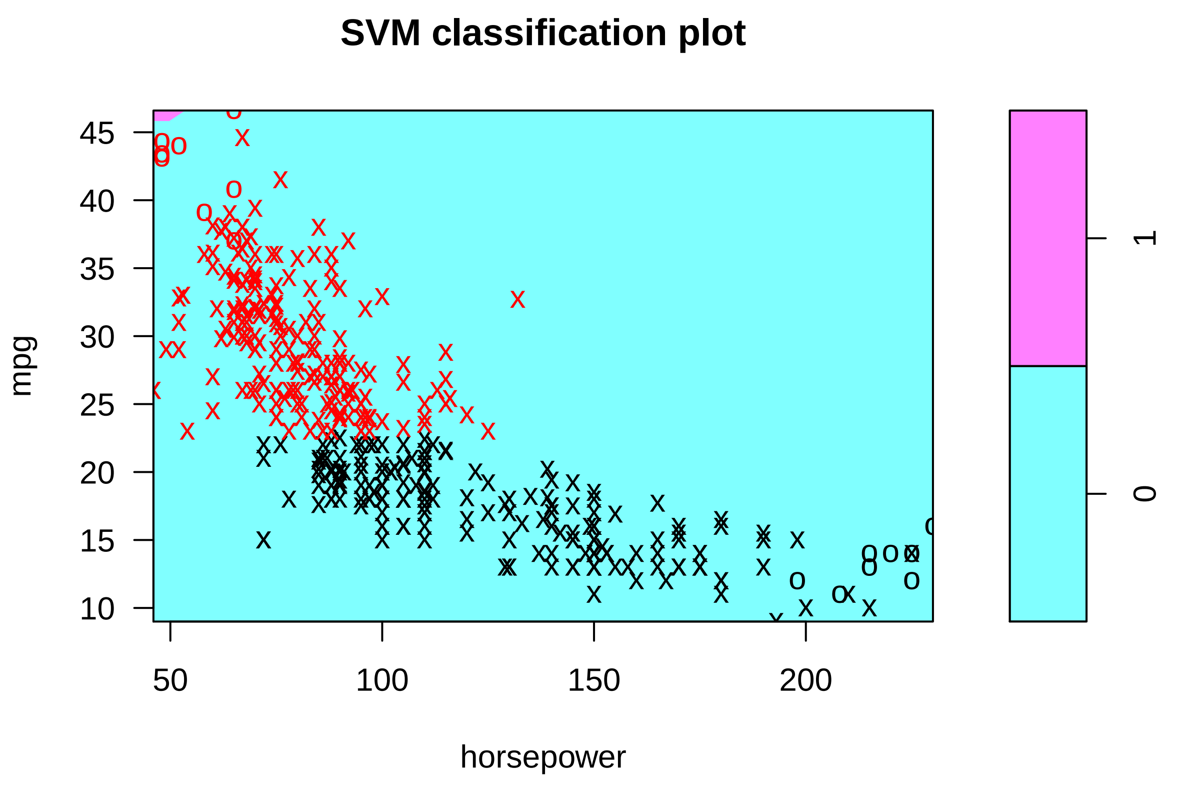
```r
# Plots for polynomial kernel
plot(svm.polynomial, data = Auto, mpg ~ weight)
```



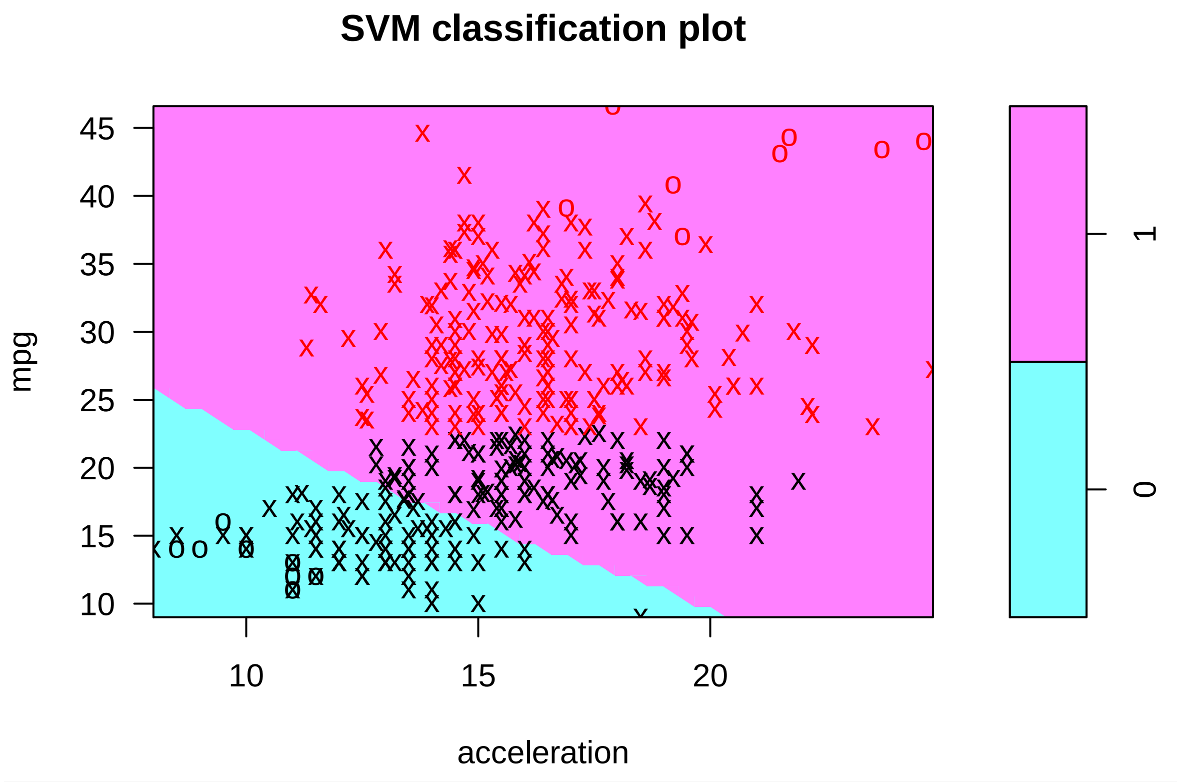
```r
plot(svm.polynomial, Auto, mpg ~ displacement)
```



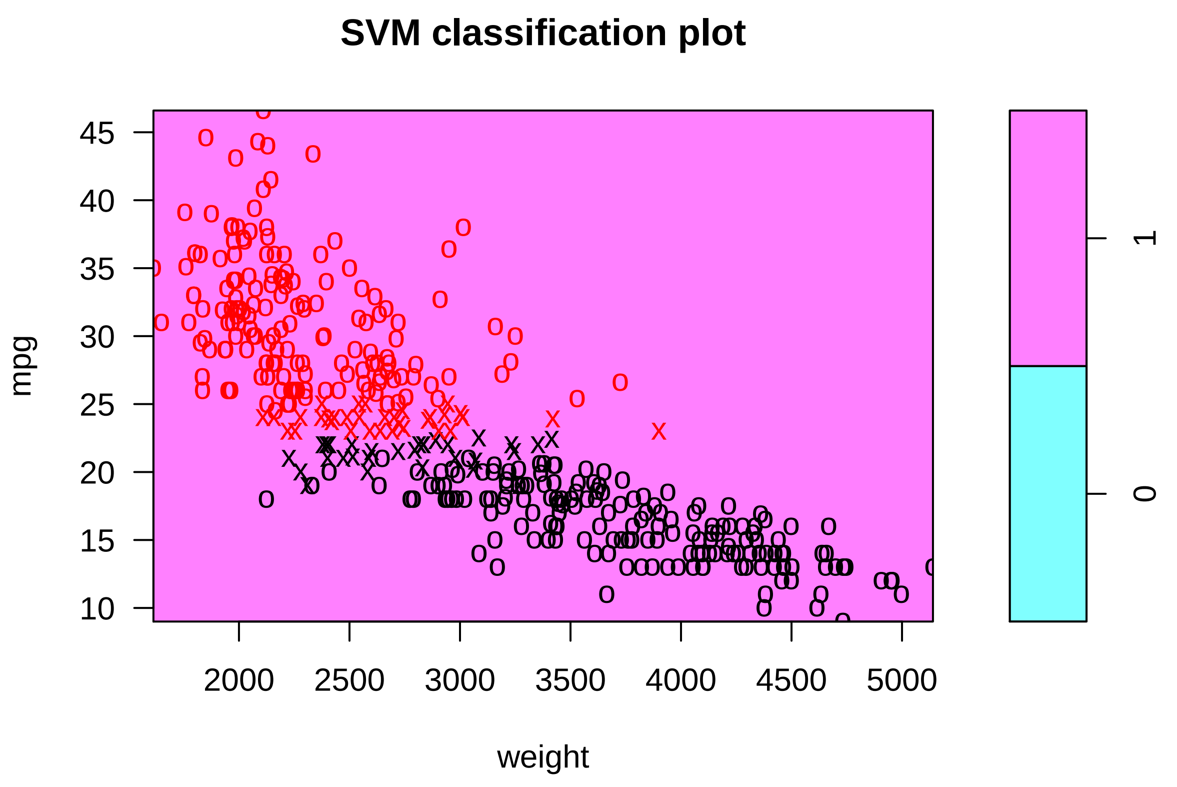
```r
plot(svm.polynomial, Auto, mpg ~ horsepower)
```



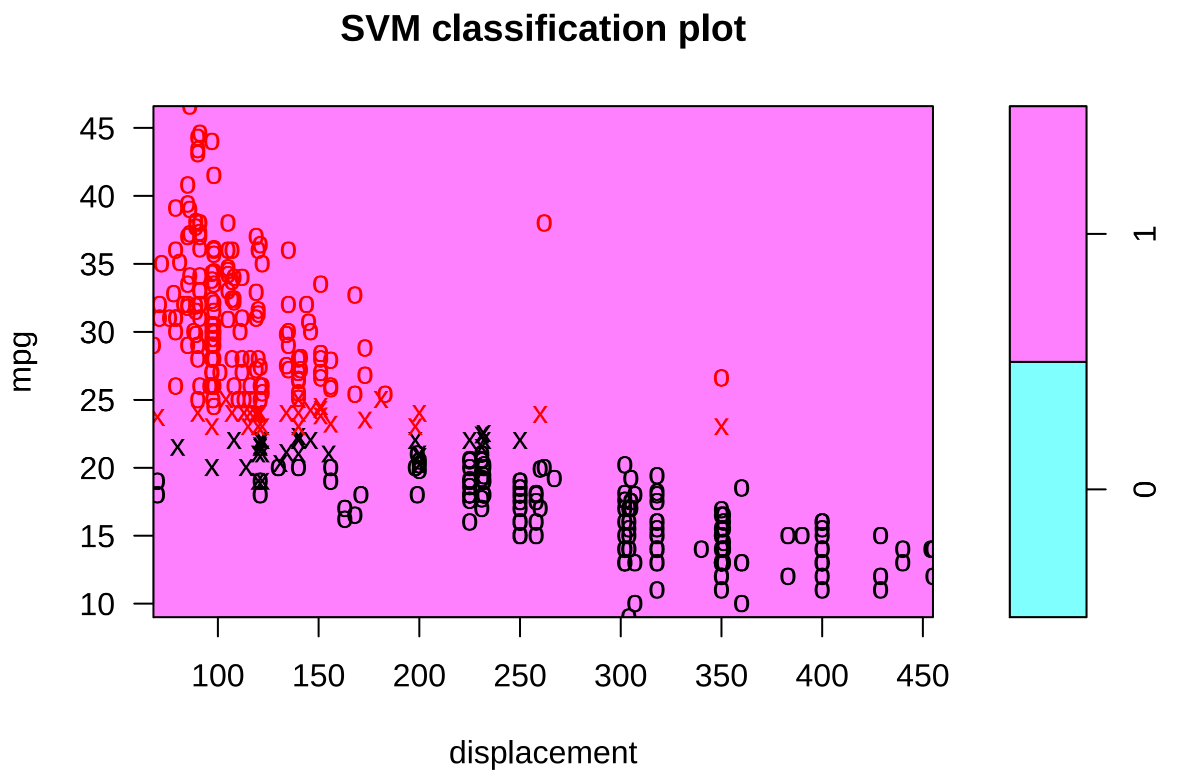
```r
plot(svm.polynomial, Auto, mpg ~ acceleration)
```



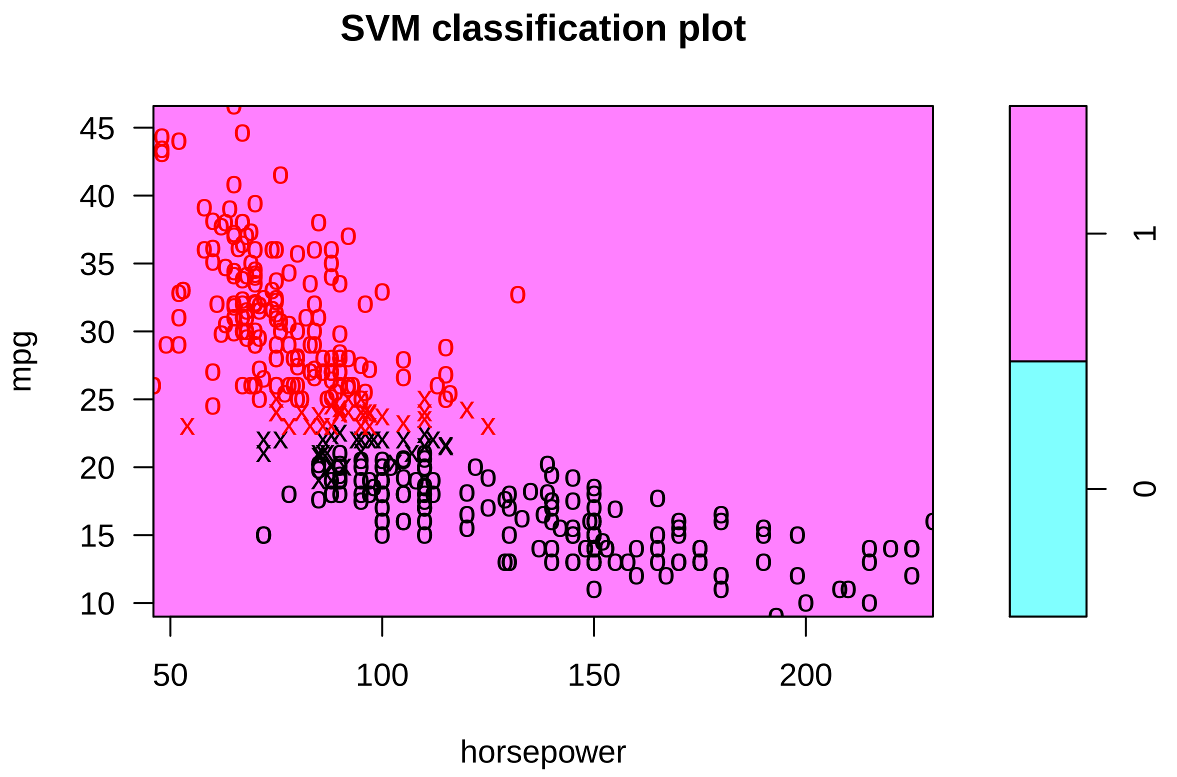
```r
# Plots for radial kernel
plot(svm.radial, data = Auto, mpg ~ weight)
```



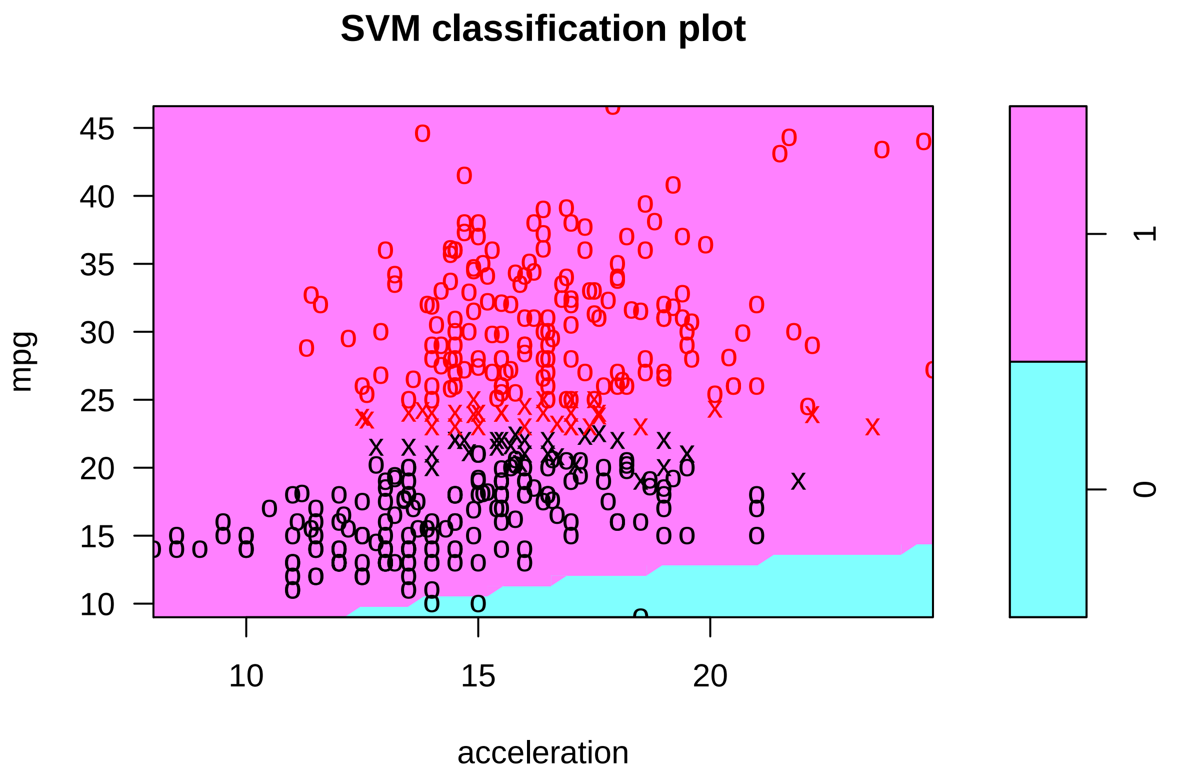
```r
plot(svm.radial, Auto, mpg ~ displacement)
```



```r
plot(svm.radial, Auto, mpg ~ horsepower)
```



```r
plot(svm.radial, Auto, mpg ~ acceleration)
```

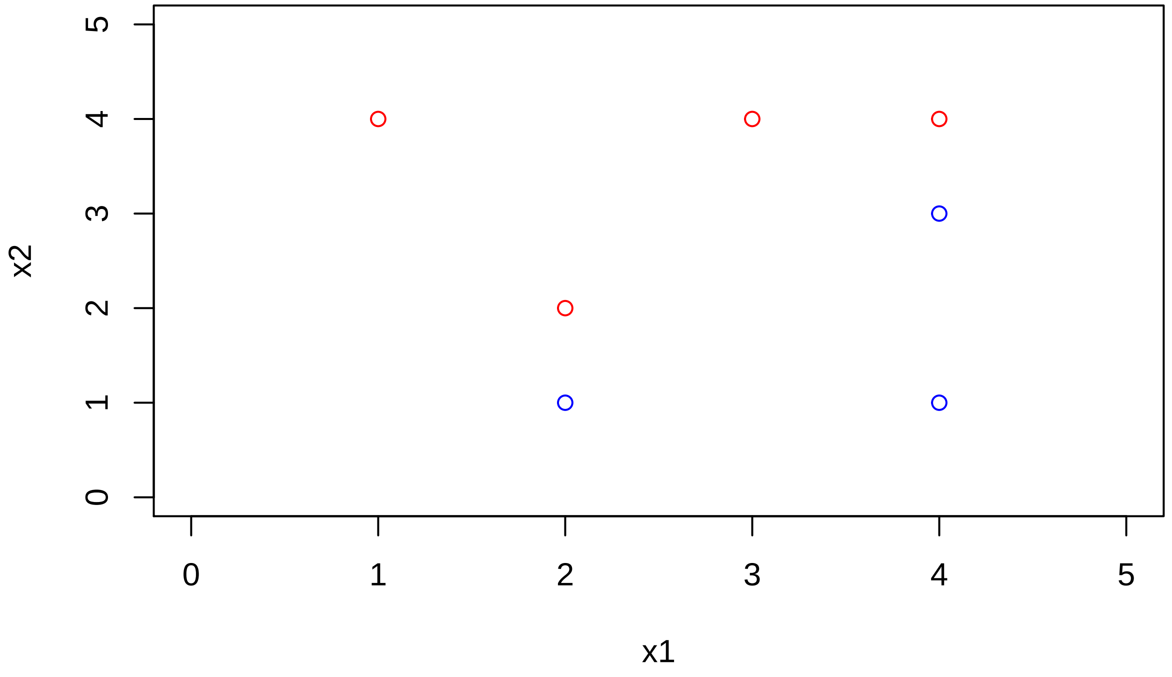


5. SVM

Here we explore the maximal margin classifier on a toy data set. (a) We are given n = 7 observations in p = 2 dimensions. For each observation, there is an associated class label.

Sketch the observations.

```r
x1 = c(3, 2, 4, 1, 2, 4, 4)
x2 = c(4, 2, 4, 4, 1, 3, 1)
cols = c("red", "red", "red", "red", "blue", "blue", "blue")
plot(x1, x2, col = cols, xlim = c(0, 5), ylim = c(0, 5))
```



(b) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane of the following form.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

Best hyperplane that would separate two color should pass mid way between points of (2, 1), (2, 2) and (4,3), (4, 4). Coordinate wise now we know that line passes through (2, 1.5) and (4, 3.5). This will give us two equations to workout the β_0, β_1 and β_2 which can be written as:

$$\beta_0 + \beta_1 2 + \beta_2 1.5 = 0$$

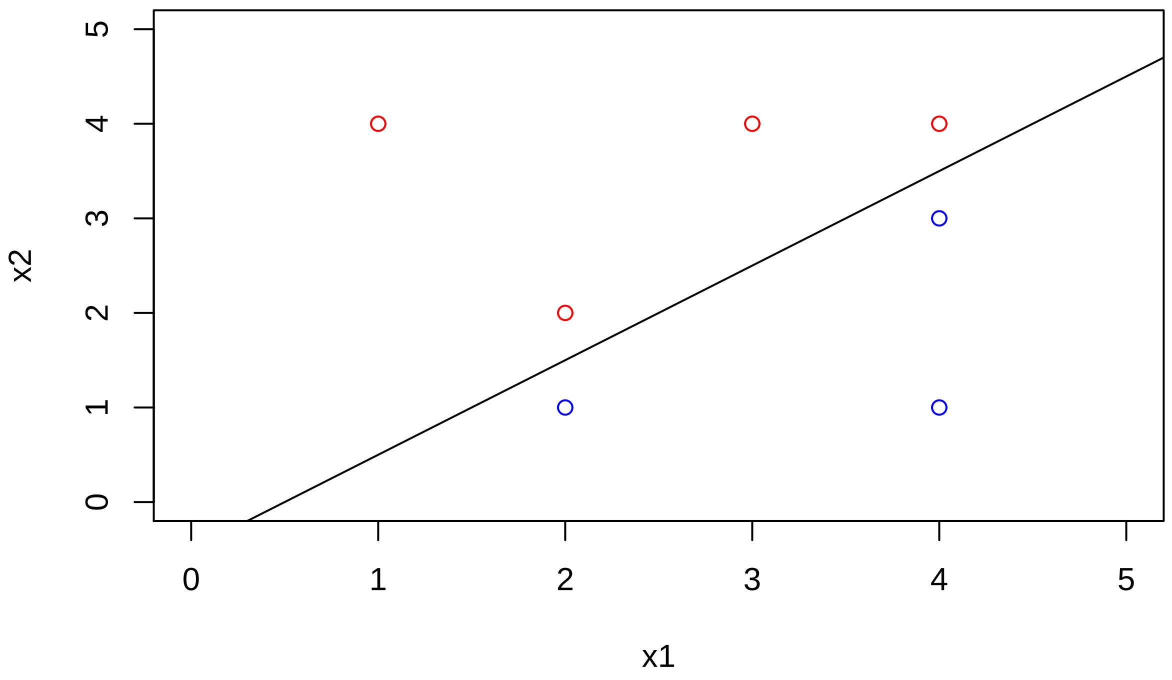
$$\beta_0 + \beta_1 4 + \beta_2 3.5 = 0$$

Slope of the function (β_1) from this points calculates as 1. which will give us β_2 = -1 and β_0 = -0.5. Given the values function to separate these points is:

$$X_1 - X_2 - 0.5 = 0$$

We can plot the hyperplane to visualise how it fits

```r
plot(x1, x2, col = cols, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
```



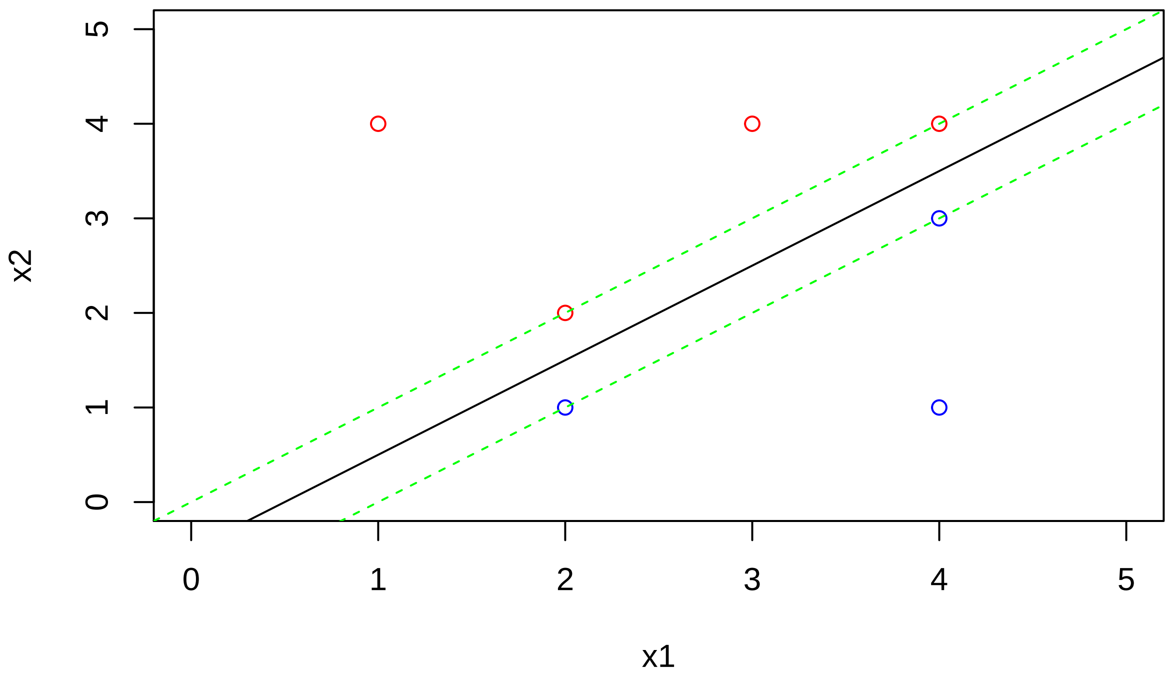
(c) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2.

In the graph red poins are above the hyperplane and blue points are below it. As we know that the point to be above the hyperplane it shoud have a higher X_2 value that what satisfies the equation we found above. As we found the coefficent of X_2 as -1 in the equation above we can conclude that the any point above the line will produce negative result in that equation which will be less than 0. Given the intuition above we can write the rule as:

Classify to Red if $X_1 - X_2 - 0.5 < 0$, and classify to Blue otherwise.

(d) On your sketch, indicate the margin for the maximal margin hyperplane.

```r
plot(x1, x2, col = cols, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.5, 1)
abline(-1, 1, lty = 2, col="green")
abline(0, 1, lty = 2, col="green")
```



(e) Indicate the support vectors for the maximal margin classifier.

Support vectors in red points are, (2, 2) and (4, 4). For the blue points, (2, 1) and (4, 3).

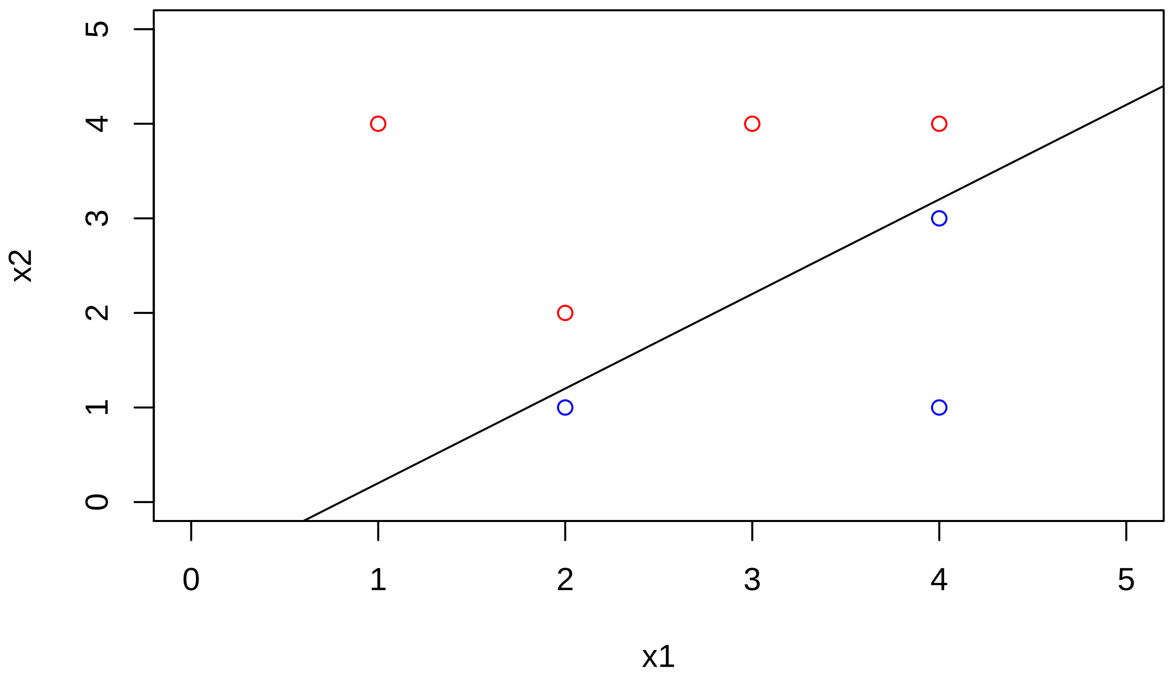
(f) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

Seventh vector have coordinates (4, 1). This point is not the support vector hence changing it will not result in change of maximal margin hyperplane. But same cannot be said for fifth and sixth point and changing them will alter the maximal margin hyperplane.

(g) Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane.

In this graph changing the intercept (β_0) in anywhere between 0 to -1 will give us not optimal yet accurately separable hyperplane. Let's choose β_0 = -0.8 and draw the hyperplane.

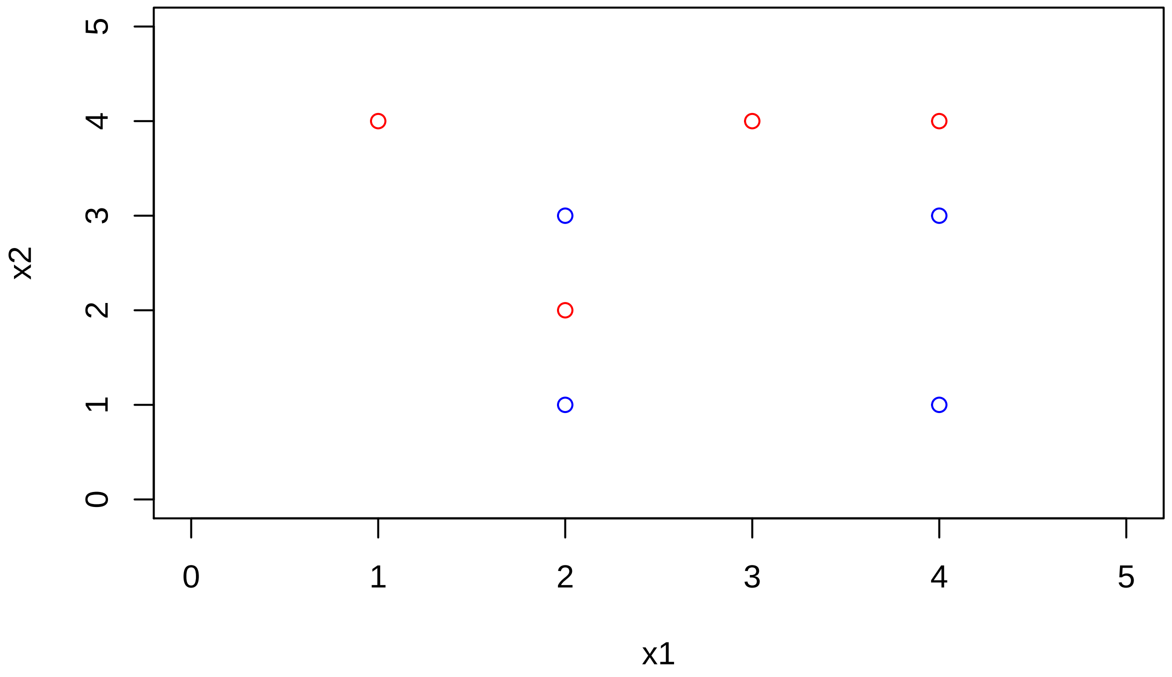
```
plot(x1, x2, col = cols, xlim = c(0, 5), ylim = c(0, 5))
abline(-0.8, 1)
```



As we can see hpyperplane still separates the blue and red points but have a very narrow margin towards to blue points.

(h) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

```r
plot(x1, x2, col = cols, xlim = c(0, 5), ylim = c(0, 5))
points(c(2), c(3), col = c("blue"))
```



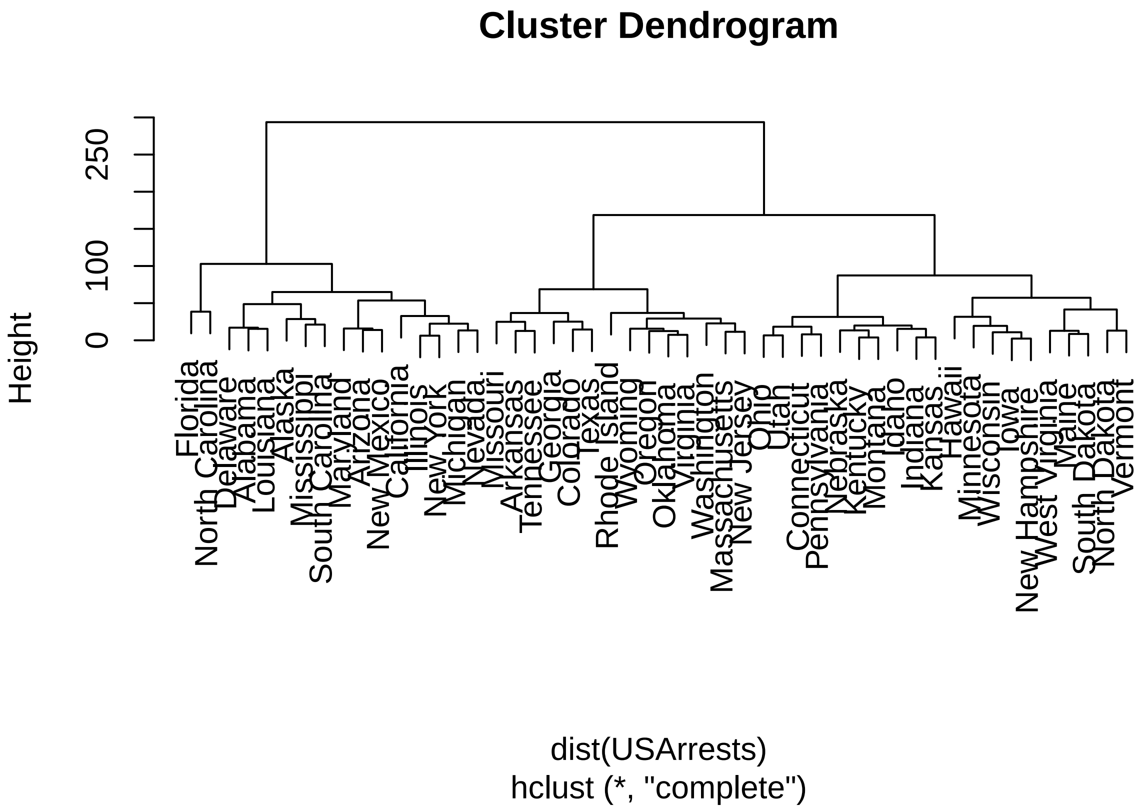
Adding that new point, two groups no longer linearly seprable.

6. Hierarchical clustering

Consider the USArrests data. We will now perform hierarchical clustering on the states.

(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```r
set.seed(123)
hc.us <- hclust(dist(USArrests), method = "complete")
plot(hc.us)
```



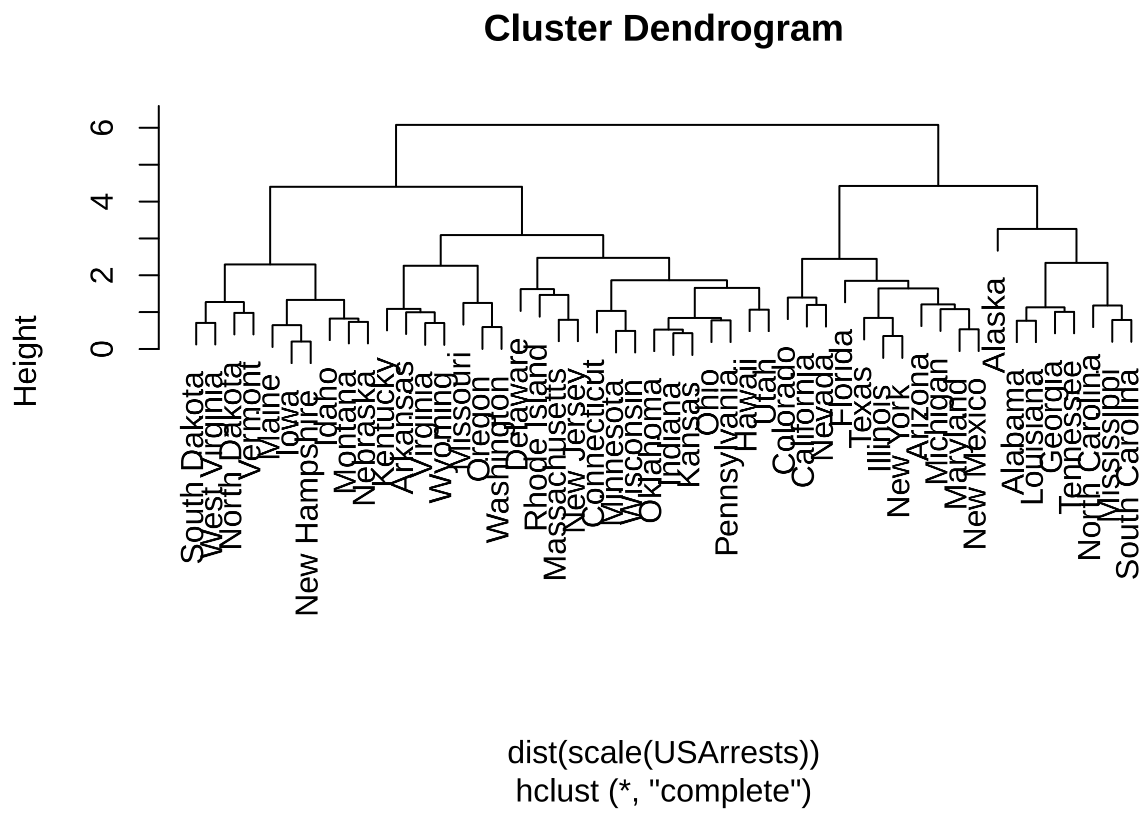
(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```r
hc.us.cut <- cutree(hc.us, 3)
split(data.frame(names(hc.us.cut), hc.us.cut), as.factor(hc.us.cut))
```

```
## $`1`
##                 names.hc.us.cut. hc.us.cut
## Alabama                  Alabama         1
## Alaska                    Alaska         1
## Arizona                  Arizona         1
## California            California         1
## Delaware                Delaware         1
## Florida                  Florida         1
## Illinois                Illinois         1
## Louisiana              Louisiana         1
## Maryland                Maryland         1
## Michigan                Michigan         1
## Mississippi          Mississippi         1
## Nevada                    Nevada         1
## New Mexico           New Mexico         1
## New York               New York         1
## North Carolina   North Carolina         1
## South Carolina   South Carolina         1
##
## $`2`
##                names.hc.us.cut. hc.us.cut
## Arkansas               Arkansas         2
## Colorado               Colorado         2
## Georgia                 Georgia         2
## Massachusetts     Massachusetts         2
## Missouri               Missouri         2
## New Jersey           New Jersey         2
## Oklahoma               Oklahoma         2
## Oregon                   Oregon         2
## Rhode Island       Rhode Island         2
## Tennessee             Tennessee         2
## Texas                     Texas         2
## Virginia               Virginia         2
## Washington           Washington         2
## Wyoming                 Wyoming         2
##
## $`3`
##                names.hc.us.cut. hc.us.cut
## Connecticut         Connecticut         3
## Hawaii                   Hawaii         3
## Idaho                     Idaho         3
## Indiana                 Indiana         3
## Iowa                       Iowa         3
## Kansas                   Kansas         3
## Kentucky               Kentucky         3
## Maine                     Maine         3
## Minnesota             Minnesota         3
## Montana                 Montana         3
## Nebraska               Nebraska         3
## New Hampshire     New Hampshire         3
## North Dakota       North Dakota         3
## Ohio                       Ohio         3
## Pennsylvania       Pennsylvania         3
## South Dakota       South Dakota         3
## Utah                       Utah         3
## Vermont                 Vermont         3
## West Virginia     West Virginia         3
## Wisconsin             Wisconsin         3
```

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
hc.us.scale = hclust(dist(scale(USArrests)), method='complete')
plot(hc.us.scale)
```



(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
hc.cut.sc <- cutree(hc.us.scale, 3)
split(data.frame(names(hc.cut.sc), hc.cut.sc), as.factor(hc.cut.sc))
```

```
## $`1`
##                 names.hc.cut.sc. hc.cut.sc
## Alabama                  Alabama         1
## Alaska                    Alaska         1
## Georgia                  Georgia         1
## Louisiana              Louisiana         1
## Mississippi          Mississippi         1
## North Carolina    North Carolina         1
## South Carolina    South Carolina         1
## Tennessee              Tennessee         1
##
## $`2`
##             names.hc.cut.sc. hc.cut.sc
## Arizona              Arizona         2
## California        California         2
## Colorado            Colorado         2
## Florida              Florida         2
## Illinois            Illinois         2
## Maryland            Maryland         2
## Michigan            Michigan         2
## Nevada                Nevada         2
## New Mexico        New Mexico         2
## New York           New York         2
## Texas                 Texas         2
##
## $`3`
##               names.hc.cut.sc. hc.cut.sc
## Arkansas              Arkansas         3
## Connecticut        Connecticut         3
## Delaware              Delaware         3
## Hawaii                  Hawaii         3
## Idaho                    Idaho         3
## Indiana                Indiana         3
## Iowa                      Iowa         3
## Kansas                  Kansas         3
## Kentucky              Kentucky         3
## Maine                    Maine         3
## Massachusetts    Massachusetts         3
## Minnesota            Minnesota         3
## Missouri              Missouri         3
## Montana                Montana         3
## Nebraska              Nebraska         3
## New Hampshire    New Hampshire         3
## New Jersey          New Jersey         3
## North Dakota      North Dakota         3
## Ohio                      Ohio         3
## Oklahoma              Oklahoma         3
## Oregon                  Oregon         3
## Pennsylvania      Pennsylvania         3
## Rhode Island      Rhode Island         3
## South Dakota      South Dakota         3
## Utah                      Utah         3
## Vermont                Vermont         3
## Virginia              Virginia         3
## Washington          Washington         3
## West Virginia    West Virginia         3
## Wisconsin            Wisconsin         3
## Wyoming                Wyoming         3
```

```
table(cutree(hc.us, 3), cutree(hc.us.scale, 3))
```

```
##
##      1  2  3
##   1  6  9  1
##   2  2  2 10
##   3  0  0 20
```

Scaling the variables in this case is appropriate because of the range and the unit differences in the data.

7. PCA and K-Means Clustering

In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

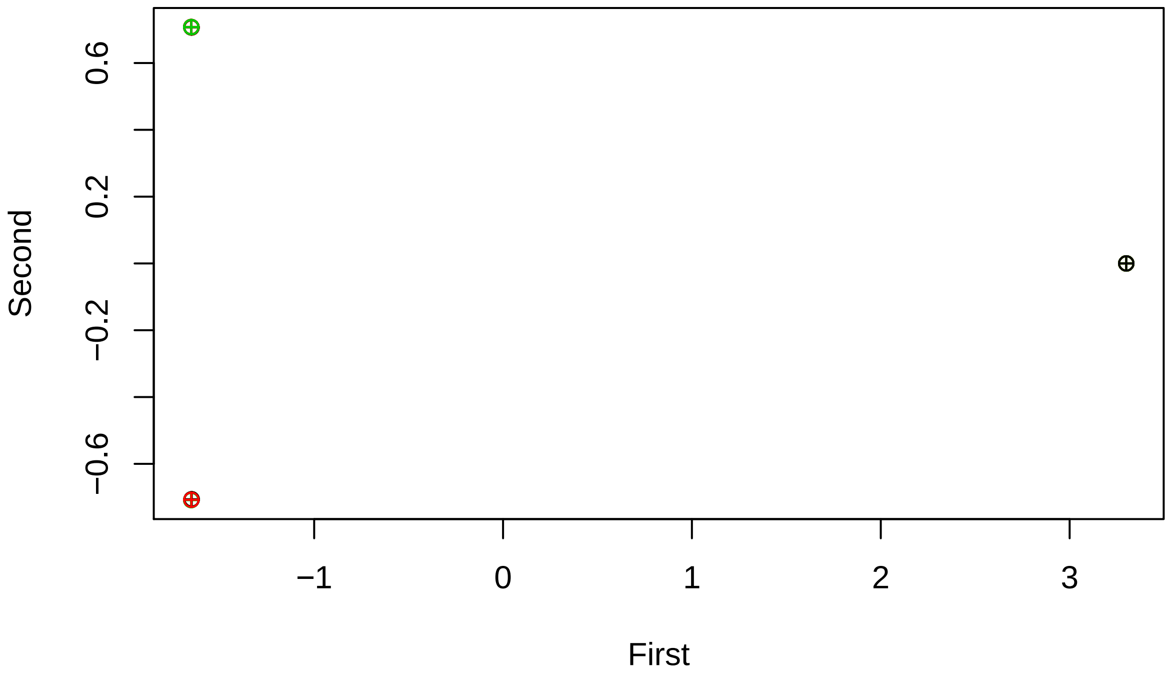
(a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(123)
df <- matrix(rnorm(20 * 3 * 50, mean = 0, sd = 0.001), ncol = 50)
df[1:20, 2] <- 1
df[21:40, 1] <- 4
df[21:40, 2] <- 4
df[41:60, 1] <- 1
y <- c(rep(1, 20), rep(2, 20), rep(3, 20))
```

Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

(b) Perform PCA on the 60 observations and plot the first two principal components' eigenvector. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component eigenvectors.

```
pr.out <- prcomp(df)
plot(pr.out$x[, 1:2], col = 1:3, xlab = "First", ylab = "Second", pch = 10)
```



(c) Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels?

```
set.seed(123)
kmean.out <- kmeans(df, 3, nstart = 20)
table(y, kmean.out$cluster)
```

```
##
## y    1  2  3
##   1 20  0  0
##   2  0  0 20
##   3  0 20  0
```

Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

(d) Perform K-means clustering with K = 2. Describe your results.

```r
km2.out <- kmeans(df, 2, nstart = 20)
table(y, km2.out$cluster)
```

```
##
## y    1  2
##   1  0 20
##   2 20  0
##   3  0 20
```

All three classes now classified in two class. Where class 1 and 2 share the same class.

(e) Now perform K-means clustering with K = 4, and describe your results.

```r
km4.out <- kmeans(df, 4, nstart = 20)
table(y, km4.out$cluster)
```

```
##
## y    1  2  3  4
##   1 20  0  0  0
##   2  0  0  0 20
##   3  0  8 12  0
```

First 2 class correctly classified in one class but the class 3 is divided into 2 classes.

(f) Now perform K-means clustering with K = 3 on the first two principal components, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component's corresponding eigenvector, and the second column is the second principal component's corresponding eigenvector. Comment on the results.

```r
km3.out <- kmeans(pr.out$x[, 1:2], 3, nstart = 20)
table(y, km3.out$cluster)
```

```
##
## y    1  2  3
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0
```

Algorithm successfully made 3 classification.

(g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

```r
km3s.out <- kmeans(scale(df), 3, nstart = 20)
table(y, km3s.out$cluster)
```

```
##
## y    1  2  3
##   1 11  9  0
##   2  4  0 16
##   3 12  6  2
```

Algorithm performs very poorly as scaling distord the information such as distance between obsevations.