exam

mares480

6 januari 2017

```
## Warning: package 'tree' was built under R version 3.3.2
```

```
## Warning: package 'pls' was built under R version 3.3.2
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

Assignment 1

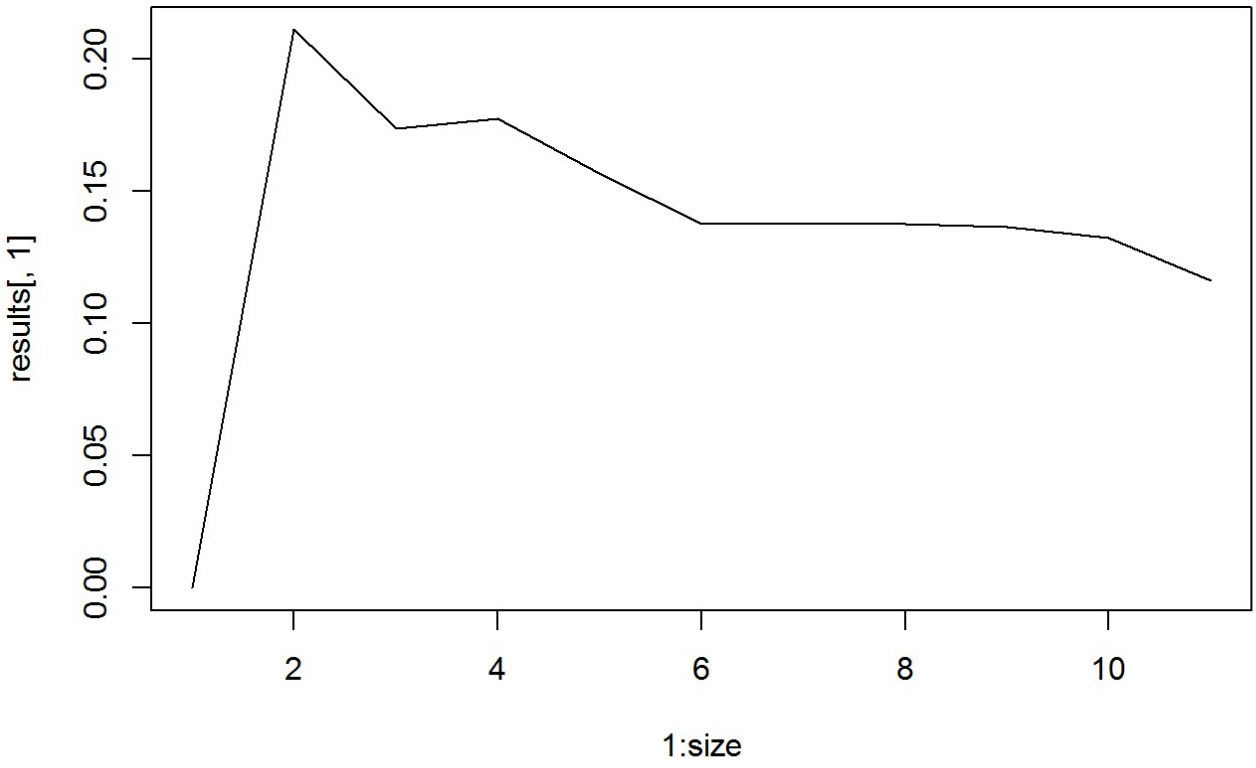
```
data = read.csv2("glass.csv",stringsAsFactors = TRUE)
n = nrow(data)
set.seed(12345)
data = data[sample(1:n,n),]
training = data[0:floor(n*0.5),]
validation = data[floor(n*0.5):floor(n*0.75),]
test = data[floor(n*0.75):n,]
```

Generate tree models for the testing and validation data and show the error plots

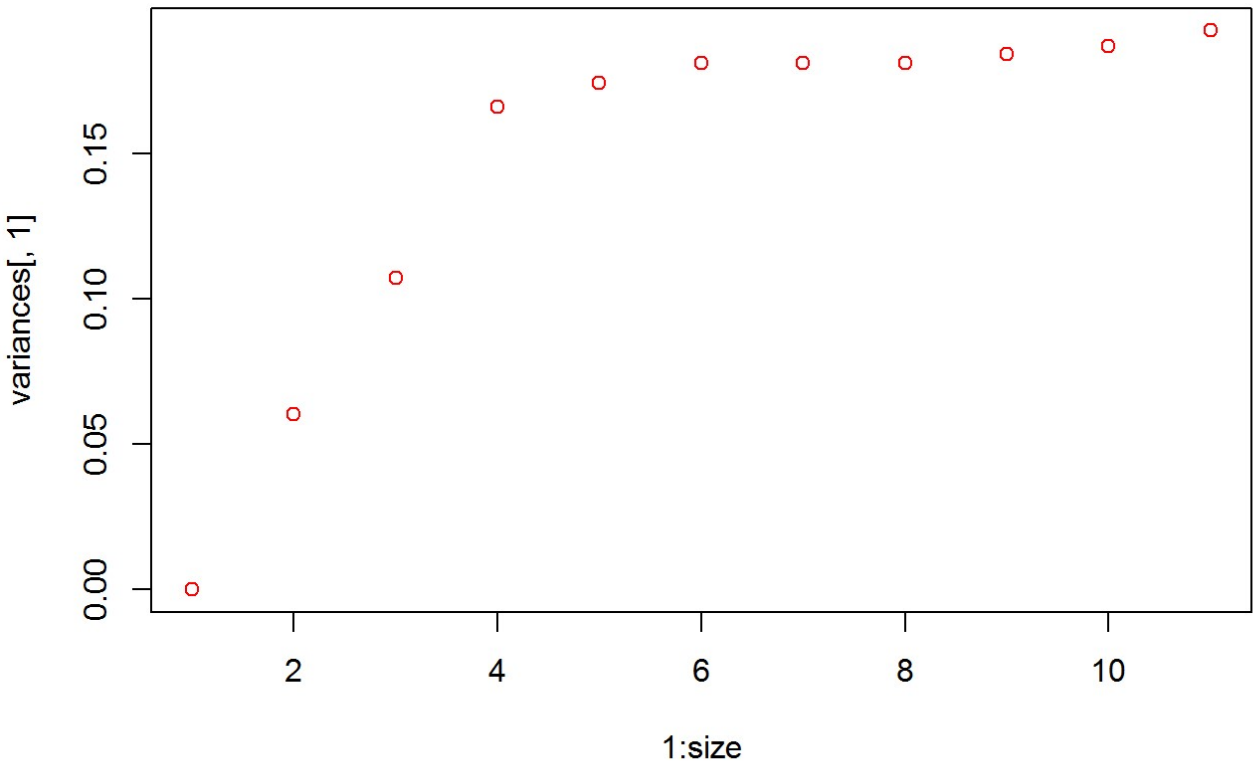
```
n = nrow(training)
control = tree.control(n, minsize=1)
fit = tree(Al ~ ., data = training, control = control)
fit.cv = cv.tree(fit)
size = summary(fit)$size[1]
print(size)
```

```
## [1] 11
```

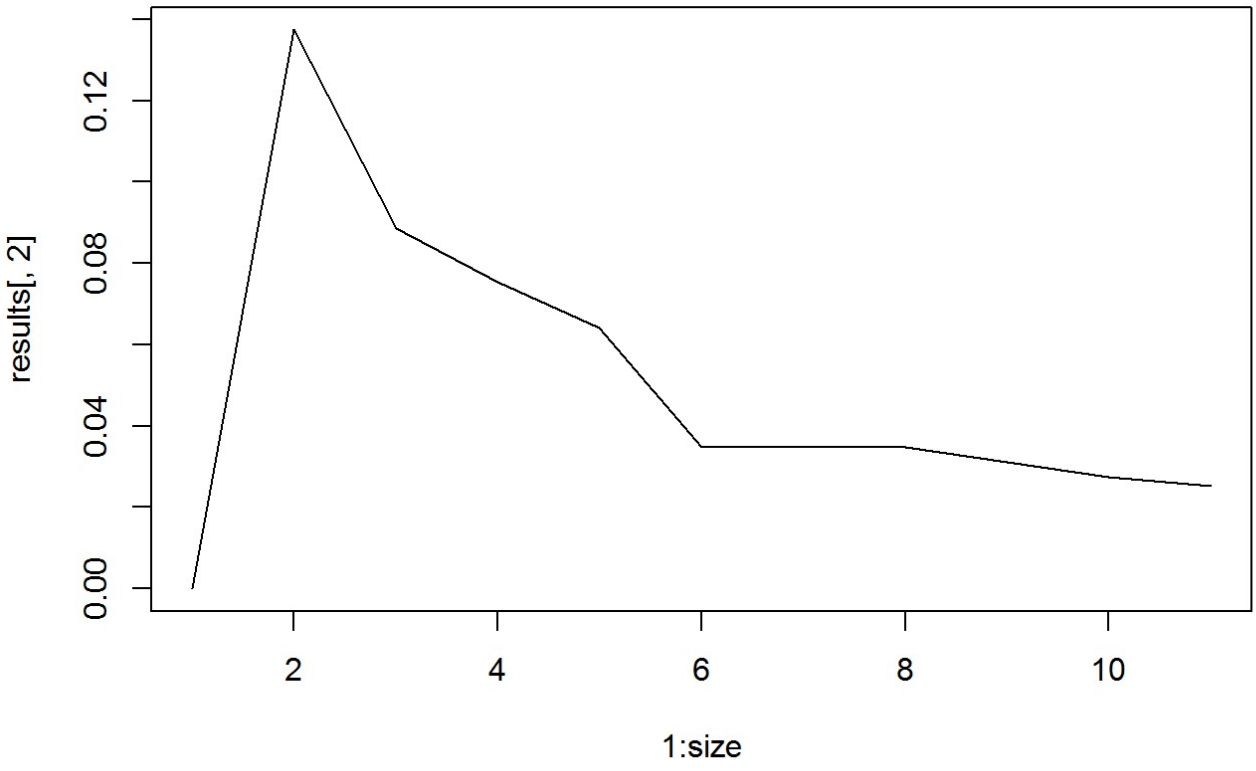
```
results = matrix(0,size,2)
variances = matrix(0,size,2)

for(i in 2:size){
  results[i,1] = mean((predict(prune.tree(fit,best = i), newdata=validation) - vali
dation$Al)^2)
  results[i,2] = mean((predict(prune.tree(fit,best = i), newdata=training) - traini
ng$Al)^2)
  variances[i,1] = var(predict(prune.tree(fit,best = i), newdata=validation))
  variances[i,2] = var(predict(prune.tree(fit,best = i), newdata=training))
}

plot(1:size, results[,1], type = "l")
```



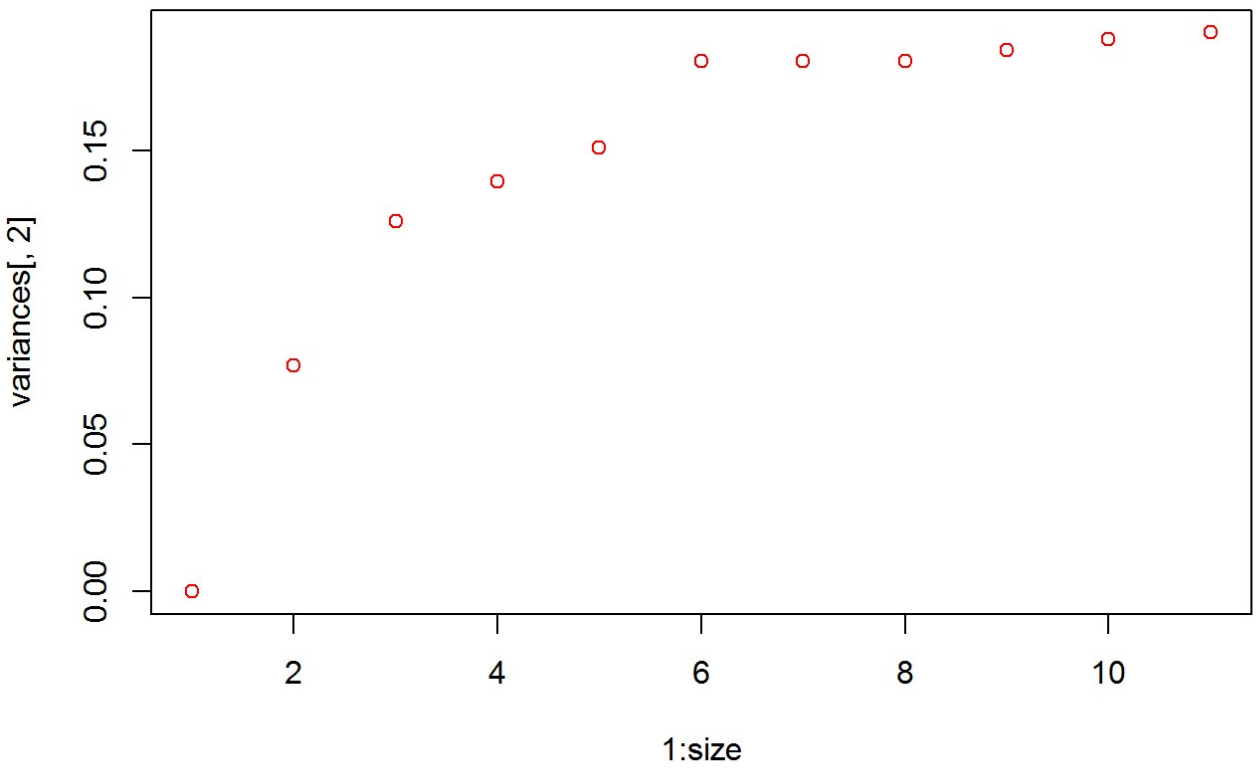

```
plot(1:size,variances[,1],col="Red")
```




```
plot(1:size, results[,2], type = "l")
```



```
plot(1:size,variances[,2],col="Red")
```



Best tree size

```
best_size = fit.cv$size[which.min(fit.cv$dev)]
optimal_tree = prune.tree(fit,best=best_size)
print(best_size)
```

```
## [1] 5
```

```
summary(optimal_tree)
```

```
##
## Regression tree:
## snip.tree(tree = fit, nodes = c(11L, 6L, 4L))
## Variables actually used in tree construction:
## [1] "Ba" "K"  "Ca"
## Number of terminal nodes:  5
## Residual mean deviance:  0.06715 = 6.85 / 102
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.61560 -0.13290 -0.03286  0.00000  0.12270  1.18400
```

test error

```
result = mean((predict(optimal_tree, newdata = test) - test$Al)^2)
print(result)
```

```
## [1] 0.1387066
```

PLS regression model

```
fit = plsr(Al ~ ., data = training, validation = "CV")
summary(fit)
```

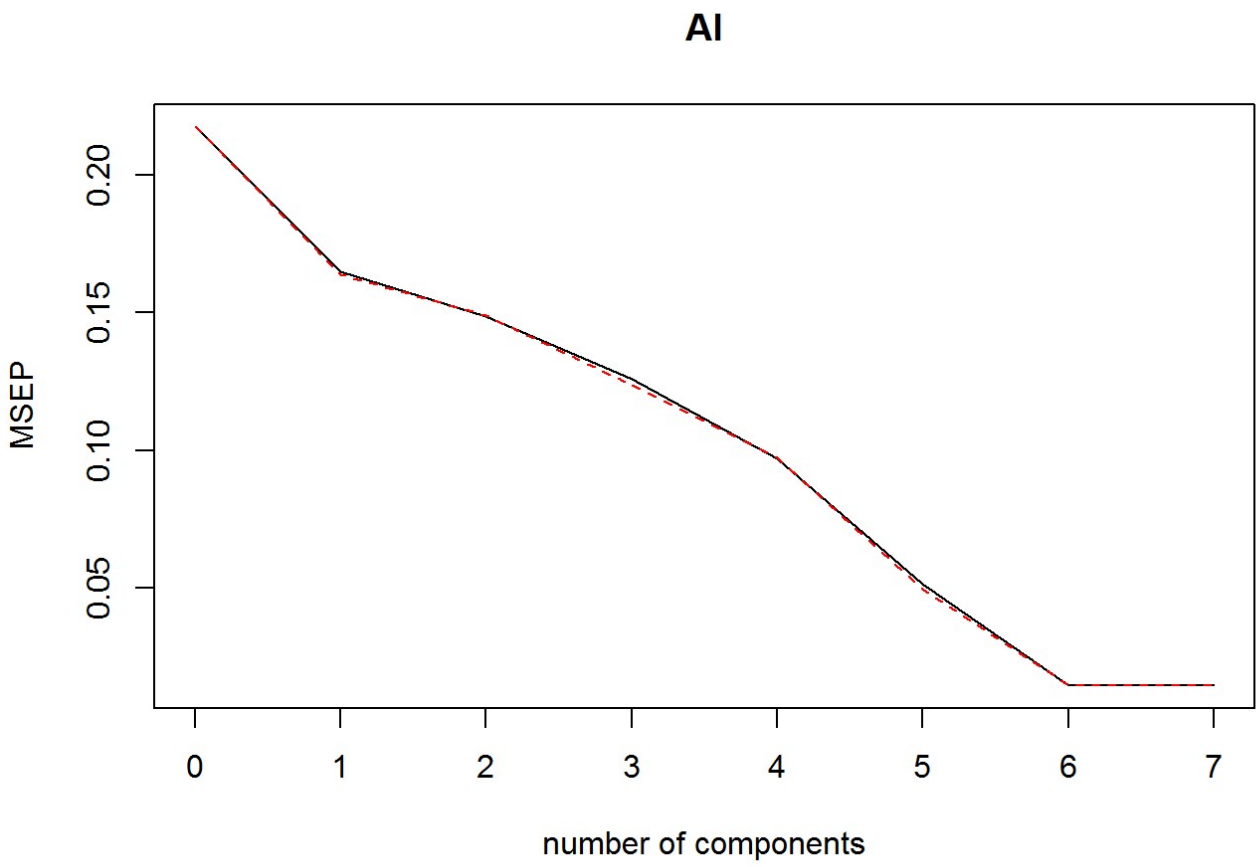
```
## Data:    X dimension: 107 7
##  Y dimension: 107 1
## Fit method: kernelpls
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.4665   0.4062   0.3855   0.3548   0.3117   0.2271   0.1215
## adjCV       0.4665   0.4049   0.3860   0.3519   0.3123   0.2225   0.1208
##        7 comps
## CV      0.1212
## adjCV   0.1205
##
## TRAINING: % variance explained
##     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X     35.86    72.19    91.13    97.80    99.09    99.87   100.00
## Al    31.98    43.73    56.75    64.93    86.86    94.41    94.43
```

We can observe how 3 components are enough to explain 90% of the variance of the data and 6 for the target. According to the CV 7 is the optimal number of varaibles to consider.

```
print(names(data)[which.max((fit$coefficients)^2)])
```

```
## [1] NA
```

```
validationplot(fit,val.type = "MS")
```



shows that the most significant component was the “Channel29” component. The function from the components to the result can be seen as the following coefficients (Al.6comps)

```
fit$coefficients
```

```
## , , 1 comps
##
##              Al
## Na  0.008835006
## Mg -0.164090892
## Si  0.031871940
## K   0.021049512
## Ca -0.099759609
## Ba  0.070091664
## Fe -0.002373527
##
## , , 2 comps
##
##              Al
## Na -0.105885813
## Mg -0.182492794
## Si  0.002343184
## K   0.072212317
## Ca -0.217019847
## Ba  0.099860617
## Fe -0.002280949
##
## , , 3 comps
##
##              Al
## Na -0.319668176
## Mg -0.291170650
## Si -0.089021787
## K   0.130519805
## Ca -0.235569000
## Ba  0.121716799
## Fe -0.002092571
##
## , , 4 comps
##
##             Al
## Na -0.35680800
## Mg -0.39452422
## Si -0.37771865
## K   0.10196171
## Ca -0.32887585
## Ba  0.04919988
## Fe -0.01411537
##
## , , 5 comps
##
##             Al
## Na -0.67225793
## Mg -0.84586015
## Si -0.79251966
## K  -0.32352053
## Ca -0.76055484
## Ba -0.84334600
## Fe -0.06466132
```

```
##
## , , 6 comps
##
##            Al
## Na -0.9174895
## Mg -0.9381058
## Si -0.9508007
## K  -0.9950250
## Ca -0.9371439
## Ba -0.8615871
## Fe -0.1062298
##
## , , 7 comps
##
##            Al
## Na -0.9189207
## Mg -0.9370586
## Si -0.9492225
## K  -0.9928497
## Ca -0.9359308
## Ba -0.8611291
## Fe -0.1963429
```

and the predicted error of the PLS model is

```
print( mean((predict(fit,newdata=test) - test$Al)^2))
```

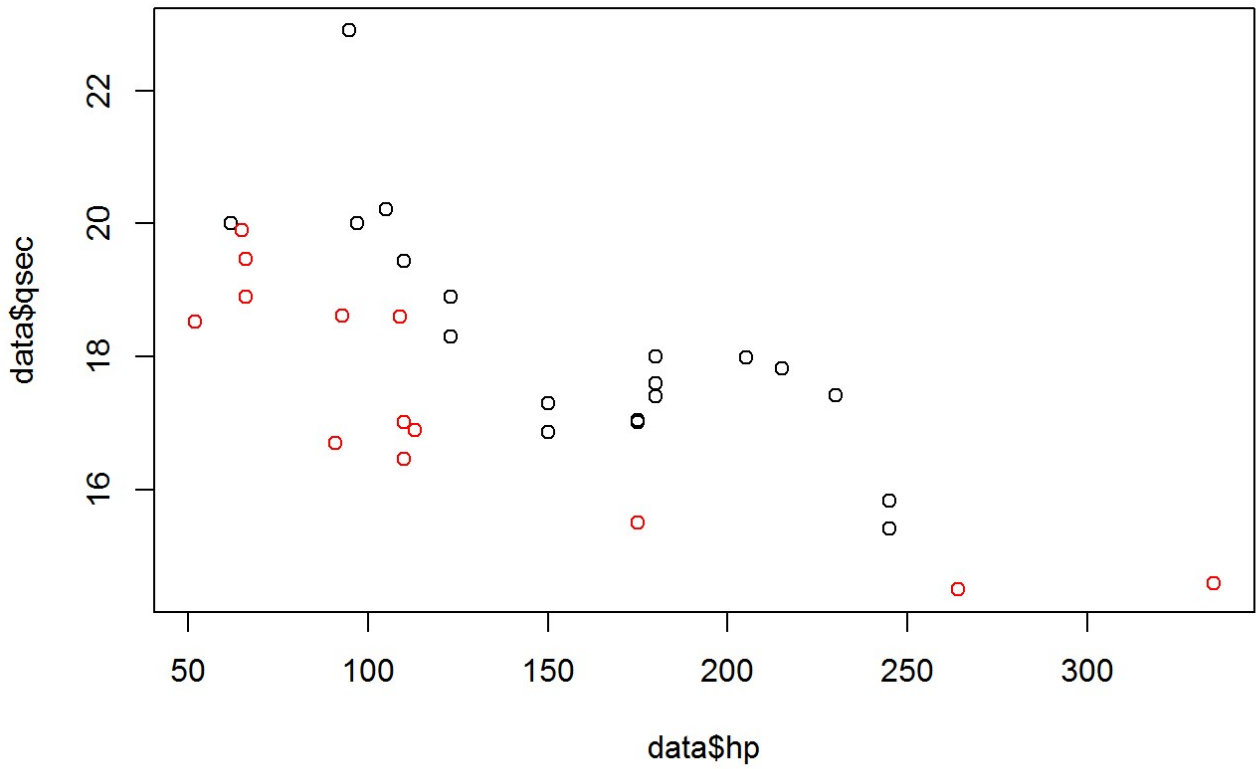
```
## [1] 0.09987647
```

Comparing the reggresion tree model and the PLS we can see how the PLS model had a lower MSE.

Assignment 2

Plot the data in the coordinates hp vs qsec

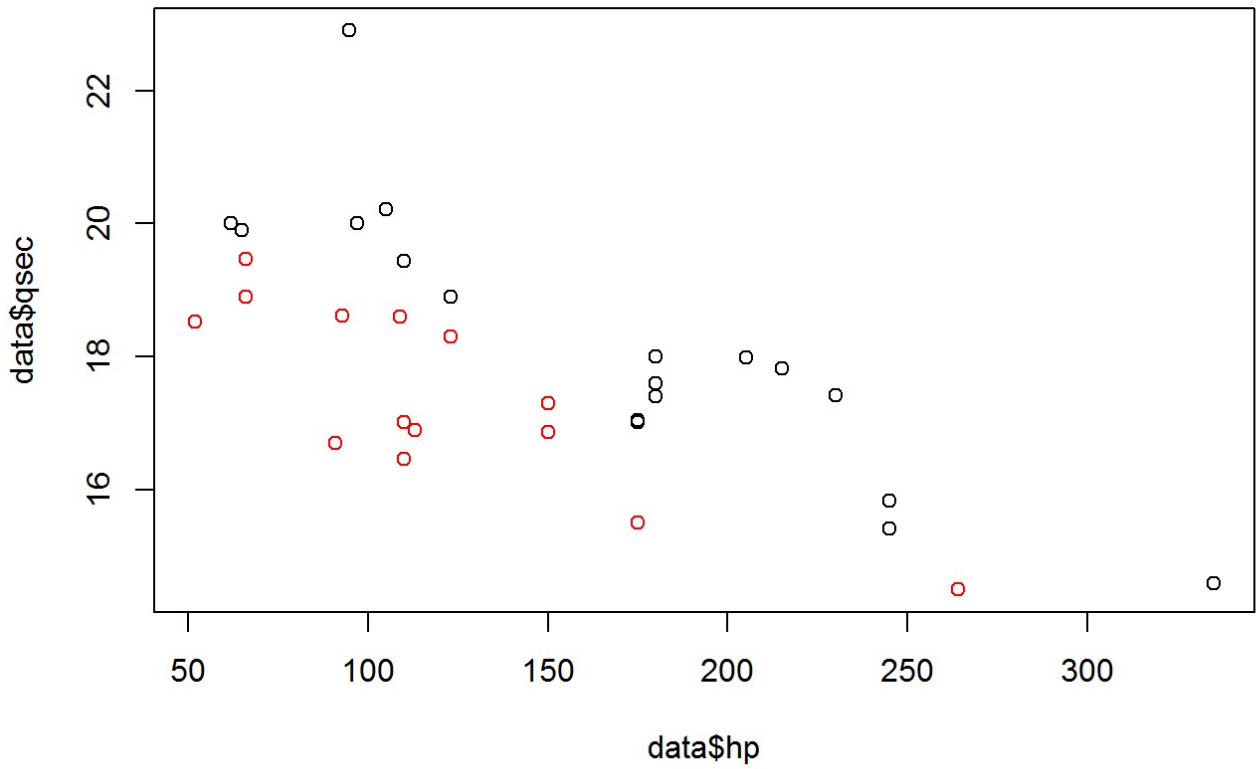
```
data = mtcars
plot(data$hp, data$qsec, col=data$am+1)
```



The assumption of LDA is that the covaranance of all the calses should be equal (in practice simmilar) to each other. Looking at the plot above we can see how this asusmption seems to be fullfiled. The data will not be classified perfeclty, even if class priors are choosen perfeclt because no matter where you put the class sepperation, som elements will be missclassified.

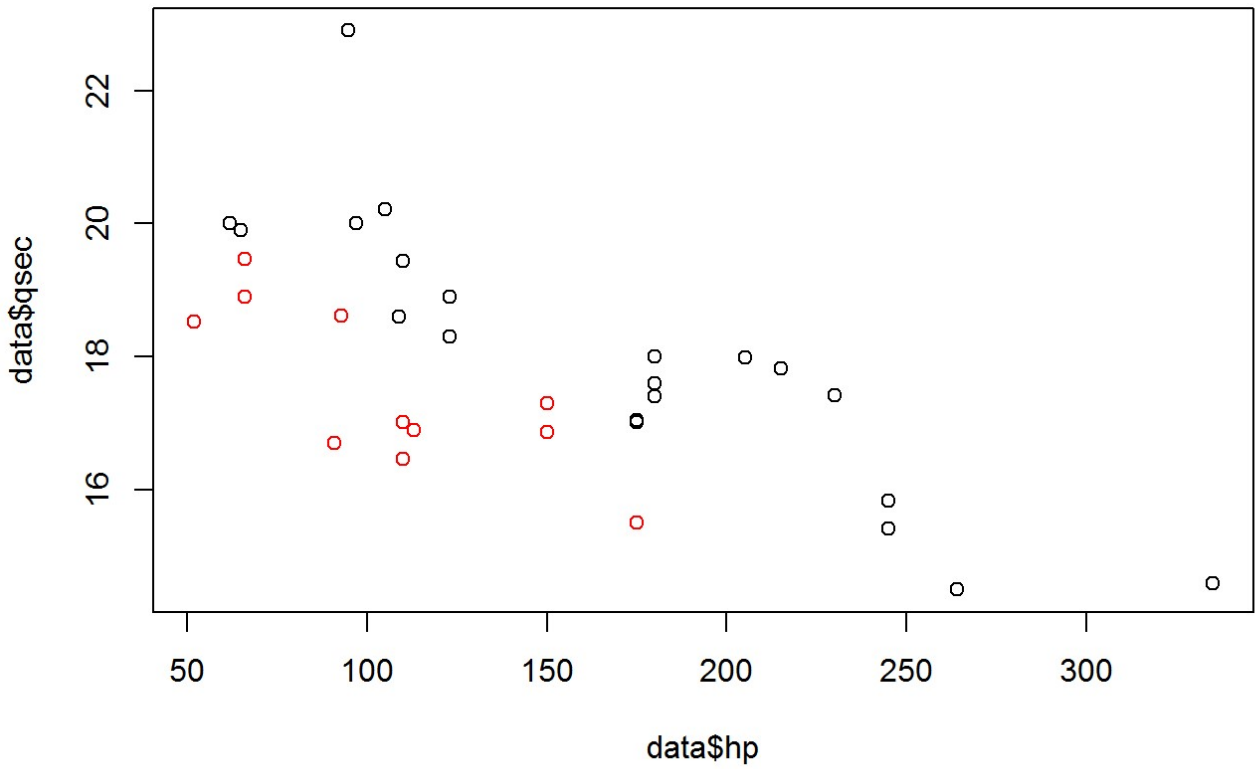
LDA with equal priors (0.5,0.5)

```
fit_equal = lda(am ~ qsec + hp, data = data, prior = c(0.5, 0.5))
plot(data$hp, data$qsec, col=predict(fit_equal)$class)
```



LDA with proportial priiors

```
fit_prop = lda(am ~ qsec + hp, data = data)
plot(data$hp, data$qsec, col=predict(fit_prop)$class)
```



looking at the missclassfication rate of both proportial priors and equal priors

```
##
##       0  1
##    0 16  2
##    1  3 11
```

```
## [1] 0.15625
```

```
##
##       0  1
##    0 17  4
##    1  2  9
```

```
## [1] 0.1875
```

we can observe how equal priors had a lower missclassfication rate compared to proportional priors.

Looking at the models we can se that

```
## Call:
## lda(am ~ qsec + hp, data = data, prior = c(0.5, 0.5))
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##       qsec       hp
## 0 18.18316 160.2632
## 1 17.36000 126.8462
##
## Coefficients of linear discriminants:
##              LD1
## qsec -0.91338199
## hp   -0.02403582
```

```
## Call:
## lda(am ~ qsec + hp, data = data)
##
## Prior probabilities of groups:
##       0       1
## 0.59375 0.40625
##
## Group means:
##       qsec       hp
## 0 18.18316 160.2632
## 1 17.36000 126.8462
##
## Coefficients of linear discriminants:
##              LD1
## qsec -0.91338199
## hp   -0.02403582
```

the slope hasn't changed anything between the two models but the intercept has.

IMplement kernel density estmiation with Epanechnikov kernel that uses matricies X, XTest and a sclara λ to estimate thedensity from X and predict it as XTest.

```
epanechnikov = function(x) {
  x_nor = norm(x,"F")
  if(x_nor^2 >= 0){
    return(1 - x_nor^2)
  }
  return(0)
}

kernel = function(X,XTest,lambda){
 n = nrow(X)
 return(apply(XTest,1, function(x){
   value
   for(i in 1:n){
     value = 1/(n*lambda) * sum(epanechnikov(X[i,1] - x))
   }
   return(value)
 }))
}
```

Assignemnt 3

```
data = read.csv2("wine.csv", sep = ",")
data$class[which(data$class == 2)] = -1
for(i in 1:ncol(data)){data[,i] = as.numeric(data[,i])}

set.seed(12345)
samples = sample(1:nrow(data),floor(nrow(data)*0.7))
training = data[samples,]
test = data[-samples,]
```

```
## Warning: package 'neuralnet' was built under R version 3.3.2
```

```
set.seed(12345)
f <- as.formula(paste("class ~", paste(training[!training %in% "class"], collapse =
 " + ")))
nn <- neuralnet(f, training, hidden = 0, act.fct = "tanh")
print(colMeans(nn$generalized.weights[[1]]))
```

```
##  [1] 231970.62144271849      0.22880223380     -0.08355913868
##  [4]      0.36360044651     -0.10056894706     -0.70109274930
##  [7]      0.51774032174     -0.54126161117     -0.61752469646
## [10]      0.14282213230     -0.12506973603      0.03550855728
## [13]     -0.21326869228     -0.02152119292
```

We can observe how the feature “proline” has a significant higher weight compared to all other features, this would indicate that it’s the most important while the feaure “alcohol” is the least important.

```
pred_train = sign(compute(nn, training)$net.result)
pred_test =  sign(compute(nn, test)$net.result)
miss_rate_train = sum(pred_train != training$class) / nrow(training)
miss_rate_test = sum(pred_test != test$class) / nrow(test)
print(miss_rate_test)
```

```
## [1] 0
```

```
print(miss_rate_train)
```

```
## [1] 0
```

```
set.seed(12345)
nn <- neuralnet(f, training, hidden = 1, act.fct = "tanh")
plot(nn)
print(colMeans(nn$generalized.weights[[1]]))
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
pred_train = sign(compute(nn, training)$net.result)
pred_test =  sign(compute(nn, test)$net.result)
miss_rate_train = sum(pred_train != training$class) / nrow(training)
miss_rate_test = sum(pred_test != test$class) / nrow(test)
print(miss_rate_test)
```

```
## [1] 0.6666666667
```

```
print(miss_rate_train)
```

```
## [1] 0.4945054945
```