Compulsory Exercise 2: Group 37

TMA4268 Statistical Learning V2019

Anders Bendiksen and Helge Bergo

09 April, 2020

Problem 1

a) Ridge Regression

Using λ as a tuning parameter and β as the ridge regression coefficients, the goal is to minimize

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

where λ is greater than zero, and the residual sum of squares is

$$RSS = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij})^2$$

Setting $\hat{\beta}_0 = 0$, in other words setting the mean to zero, the first equation can be rewritten as

$$(y - X\hat{\beta}_{Ridge})^T (y - X\hat{\beta}_{Ridge}) + \lambda \hat{\beta}_{Ridge}^T \hat{\beta}_{Ridge}$$

When this is differentiated with respect to $\hat{\beta}_{Ridge}$ and equal to zero, we get

$$-2X^T(y - X\hat{\beta}_{Ridge}) + 2\lambda\hat{\beta}_{Ridge} = 0$$
$$X^T X \hat{\beta}_{Ridge} + \lambda \hat{\beta}_{Ridge} = X^T y$$
$$\hat{\beta}_{Ridge} = (X^T X + \lambda I)^{-1} X^T y$$

b)

Since the expected value of $y = X\beta + \epsilon$ is $X\beta$, the expectation value of $\hat{\beta}_{Ridge}$ is

$$\mathrm{E}[\hat{\beta}_{Ridge}] = (X^T X + \lambda I)^{-1} X^T E[y]$$

$$= (X^T X + \lambda I)^{-1} X^T X \beta$$

The variance-covariance matrix is then (since $\mathrm{Var}[y] = \sigma^2$)

$$\mathrm{Var}[\hat{\beta}_{Ridge}] = \mathrm{Var}[(X^TX+\lambda I)^{-1}X^Ty]$$
$$= (X^TX+\lambda I)^{-1}X^T\mathrm{Var}[y]((X^TX+\lambda I)^{-1}X^T)^T$$
$$\sigma^2(X^TX+\lambda I)^{-1}X^TX(X^TX+\lambda I)^{-1})^T$$

c) Multiple choice

(i) TRUE
(ii) FALSE
(iii) FALSE
(iv) TRUE

d) Forward Selection

```r
library(ISLR)
set.seed(1)
train.ind = sample(1:nrow(College), 0.5*nrow(College))
college.train = College[train.ind,]
college.test = College[-train.ind,]
```

After dividing the data into a training and test set, the `regsubsets` function was used to create a forward selection model on the data, from the `leaps`-library.

```r
library(leaps)
regfit.fwd = regsubsets(Outstate~.,data=college.train,method="forward", nvmax = 18)
reg.summary = summary(regfit.fwd)
```

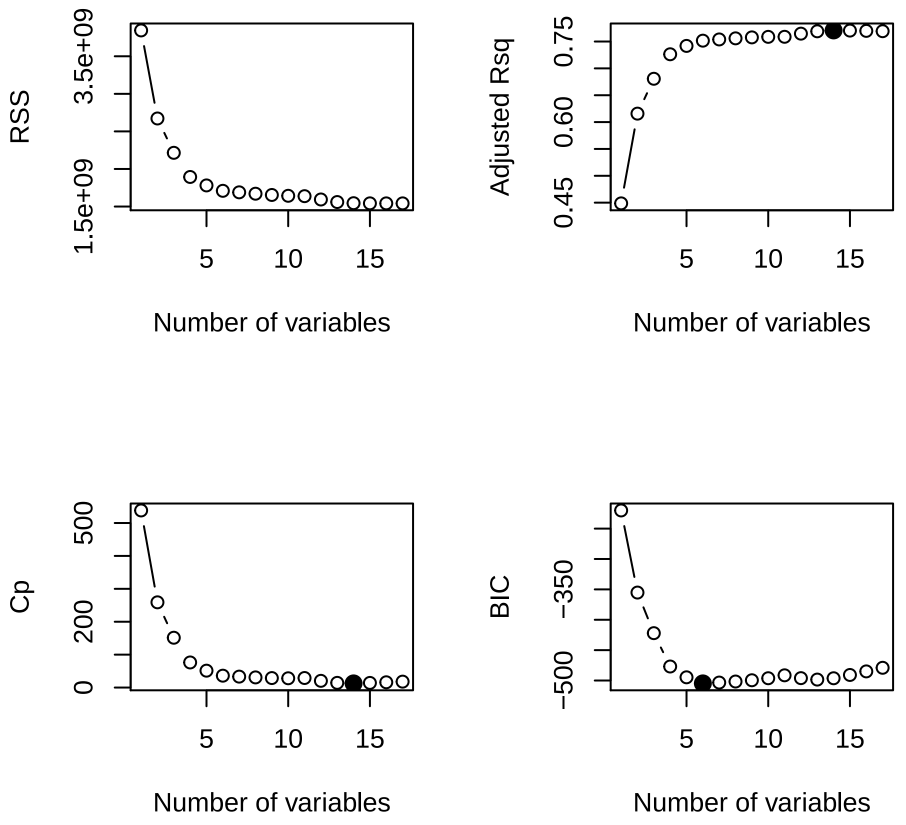
To decide on which model is best, the number of variables used in the selection was plotted against `RSS`, `Cp`, `BIC` and `adjusted R`^{2}.

```r
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of variables",ylab="RSS",type="b")

plot(reg.summary$adjr2,xlab="Number of variables",ylab="Adjusted Rsq",type="b")
max.adjr2 = which.max(reg.summary$adjr2)
points(max.adjr2,reg.summary$adjr2[max.adjr2], col="black",cex=2,pch=20)

plot(reg.summary$cp,xlab="Number of variables",ylab="Cp",type="b")
min.cp = which.min(reg.summary$cp)
points(min.cp,reg.summary$cp[min.cp], col="black",cex=2,pch=20)

plot(reg.summary$bic,xlab="Number of variables",ylab="BIC",type="b")
min.bic = which.min(reg.summary$bic)
points(min.bic,reg.summary$bic[min.bic], col="black",cex=2,pch=20)
```



The maximum `adjusted R`^{2} is the one with 14 variables, with a value of 0.7706887, shown as a filled dot in the upper right plot. This is also the same number of variables as for the lowest `Cp`. However, all the plots are pretty flat after around 6 or 7 variables used, and it seems like using only 6 variables still gives a good `adjusted R`^{2} value of 0.7516133, without the increased complexity of adding 7 more variables. The model is then:

```
coef(regfit.fwd,6)
```

```
##   (Intercept)    PrivateYes    Room.Board      Terminal   perc.alumni
## -4726.8810613  2717.7019276     1.1032433    36.9990286    59.0863753
##        Expend     Grad.Rate
##     0.1930814    33.8303314
```

For the MSE, the following code calculates the MSE for all the variables.

```
val.errors = rep(NA,17)
x.test = model.matrix(Outstate~.,data=college.test) # notice the -index!
for (i in 1:17) {
    coefi = coef(regfit.fwd,id=i)
    pred = x.test[,names(coefi)]%*%coefi
    val.errors[i] = mean((college.test$Outstate-pred)^2)
}
```

The MSE of the model with 6 variables is then:

```
val.errors[6]
```

```
## [1] 3844857
```

e)

Using the Lasso method from the `glmnet`-library, a new model was selected.

To select the tuning parameter λ, cross-validation was performed, and the λ giving the lowest MSE was selected.

```r
cv.out = cv.glmnet(x.train,y.train, alpha = 1)
best.lambda = cv.out$lambda.min
best.lambda
```

```
## [1] 10.7207
```

This was used on the test set, to get the MSE for the lasso.

```r
lasso.pred = predict(lasso.model,s=best.lambda ,newx=x.test)
MSE.lasso = mean((lasso.pred-y.test)^2)
MSE.lasso
```

```
## [1] 3688061
```

Finally, the coefficients of the model are shown here:

```r
lasso.coef = predict(cv.out,type="coefficients",s=best.lambda)[1:18,]
lasso.coef
```

```
##   (Intercept)    PrivateYes          Apps        Accept        Enroll
## -1.172140e+03  2.230467e+03 -2.825215e-01  6.615811e-01 -3.778631e-01
##     Top10perc     Top25perc   F.Undergrad   P.Undergrad    Room.Board
##  4.589180e+01 -1.485674e+01 -5.800132e-02 -5.713770e-02  1.088115e+00
##         Books      Personal           PhD      Terminal     S.F.Ratio
## -9.185125e-01 -3.005419e-01  4.013410e+00  2.996744e+01 -6.936391e+01
##   perc.alumni        Expend     Grad.Rate
##  4.686967e+01  1.480013e-01  2.431539e+01
```

Problem 2

a) Multiple choice

(i) FALSE
(ii) FALSE
(iii) TRUE
(iv) TRUE

b)

The basis representations for the cubic spline with three knots are as follows.

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } -\infty \leq x \geq q_1 \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i + (x - q_1)^3 & \text{if } \quad q_1 \leq x \geq q_2 \\ \beta_{03} + \beta_{13}x_i + \beta_{23}x_i^2 + \beta_{33}x_i^3 + \epsilon_i + (x - q_2)^3 & \text{if } \quad q_2 \leq x \geq q_3 \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i + (x - q_3)^3 & \text{if } \quad q_3 \leq x \geq \infty \end{cases}$$

X can be expressed as a vector of components x_i

$$X = [x_1, x_2, x_3, ..., x_n]$$

And Y can similarly be expressed as:

$$Y = [y_1, y_2, y_3, ..., y_n]$$

With i being a number 1 to n, and n being the number of components in the X and Y vectors. The basis functions are the functions or operations that are applied to X in order to fit Y.

$$y_i = \begin{cases} \beta_{01} + \beta_{11}B_1(x_i) + \beta_{21}B_2(x_i) + \beta_{31}B_3(x_i)_i + \epsilon_i & \text{if } -\infty \leq x \geq q_1 \\ \beta_{02} + \beta_{12}B_1(x_i) + \beta_{22}B_2(x_i) + \beta_{32}B_3(x_i) + \epsilon_i + B_4(x, \zeta) & \text{if } \quad q_1 \leq x \geq q_2 \\ \beta_{03} + \beta_{13}B_1/x_i) + \beta_{23}B_2(x_i) + \beta_{33}B_3(x_i) + \epsilon_i + B_4(x, \zeta) & \text{if } \quad q_2 \leq x \geq q_3 \\ \beta_{02} + \beta_{12}B_1(x_i) + \beta_{22}B_2(x_i) + \beta_{32}B_3(x_i) + \epsilon_i + B_4(x, \zeta) & \text{if } \quad q_3 \leq x \geq \infty \end{cases}$$

The basis functions; B_1, B_2, B_3, B_4 can then be expressed as:

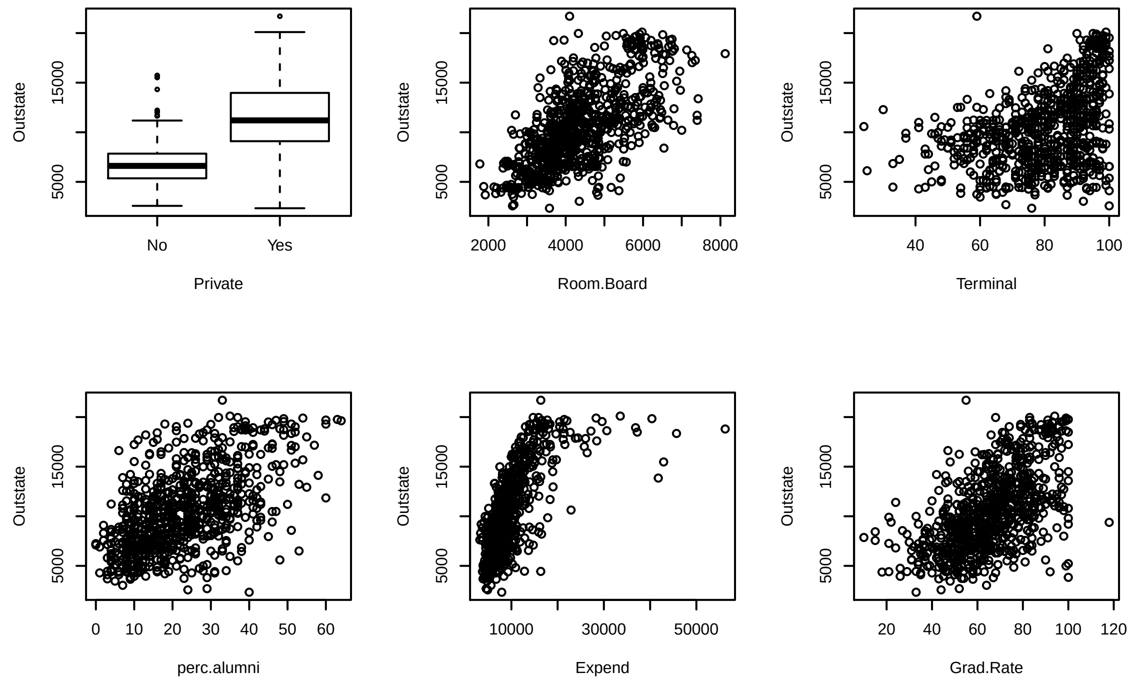
$$\begin{aligned} B_1(x) &= x \\ B_2(x) &= x^2 \\ B_3(x) &= x^3 \\ B_4(x) &= (x - \zeta)^3 \end{aligned}$$

c)

The variables

```
## [1] "Private"     "Room.Board"  "Terminal"    "perc.alumni" "Expend"
## [6] "Grad.Rate"
```

were plotted against `Outstate` to look at the relationship between the variables.



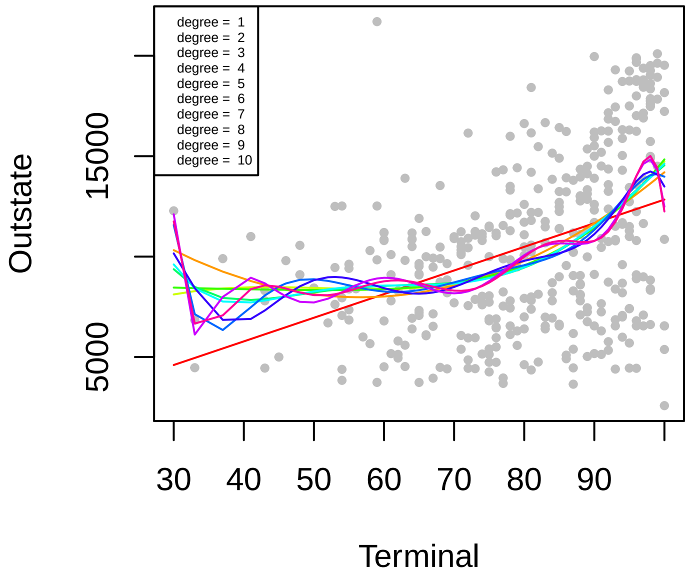
From these plots, it seems like `Room.board`, `perc.alumni` and `Grad.Rate` all have quite linear relationships with `Outstate`, while both `Terminal` and `Expend` seem to follow a non-linear relationship.

d)

(i)

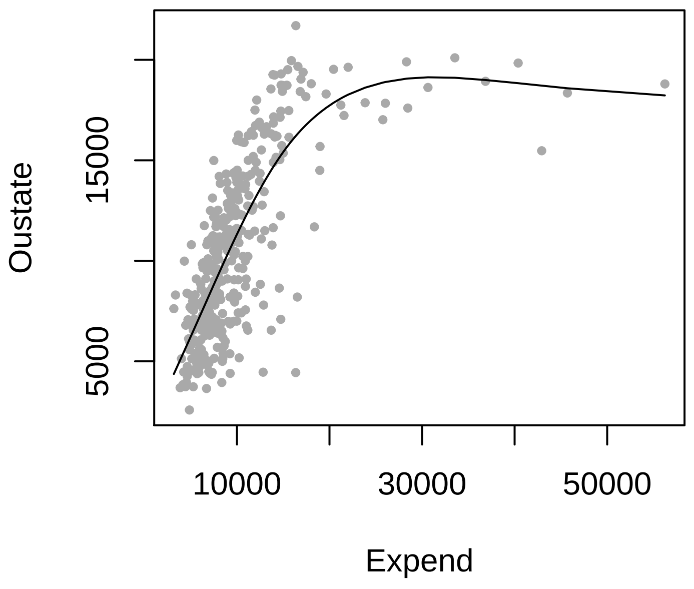
The data from `college.train` was fitted with polynomial regression for the degrees $d = 1, \ldots, 10$. The code and plot is shown below.

```r
cols = rainbow(10)
deg = 1:10
polyfunc = function(d) {
  model = lm(Outstate ~ poly(Terminal,d), data=college.train)
  lines(cbind(college.train$Terminal,model$fit)[order(college.train$Terminal),],
        col=cols[d])
  pred = predict(model, college.train)
  mean((pred - college.train$Outstate)^2)
}
plot(college.train$Terminal, college.train$Outstate, col = "gray", pch=19,
     cex = 0.5, xlab = "Terminal", ylab = "Outstate")
MSE.poly = sapply(deg, polyfunc)
legend("topleft",legend = paste("degree = ",deg), col = cols, cex = 0.4)
```



(ii)

```r
library(splines)
expend.range = range(college.train$Expend)
expend.grid = seq(from=expend.range[1], to=expend.range[2])
plot(college.train$Expend, college.train$Outstate, col = "darkgrey", pch=19,
     cex = 0.5, xlab = "Expend", ylab = "Oustate")
fit.smoothspline = smooth.spline(college.train$Expend, college.train$Outstate,cv=TRUE)
lines(fit.smoothspline)
```



The degrees of freedom was chosen using cross-validation, and the result was 4.661.

(iii)

```
MSE.smoothspline.train = mean((predict(fit.smoothspline, college.train$Expend)$y -
                                college.train$Outstate)^2)
MSE.smoothspline.train
```

```
## [1] 6871281
```

```
MSE.poly
```

```
##  [1] 15075161 14330586 14249448 14247330 14231485 14230392 14153207 14097911
##  [9] 13841526 13822205
```

The MSE for the polynomial regression is much higher than the MSE for the smoothing splines, but this makes a lot of sense when looking at the initial plots from 2.c). For the `Expend` variable, it seems like the data have a clearer trend than for the `Terminal` variable, and therefore the MSE is much lower.

Problem 3

a) Multiple choice

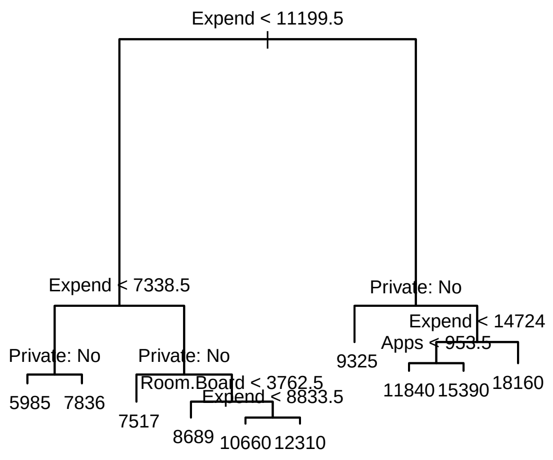
(i) TRUE
(ii) TRUE
(iii) TRUE
(iv) FALSE

b)

In order to find the best method for the data at hand, we first considered regression trees. This was quite obvious, as the tuition is not a “classifiable” value. Next up is bagging and random forest. Random forest is probably the best choice of these, as the trees are decorrelated. We will however find the MSE of all these three methods, and compare them to each other.

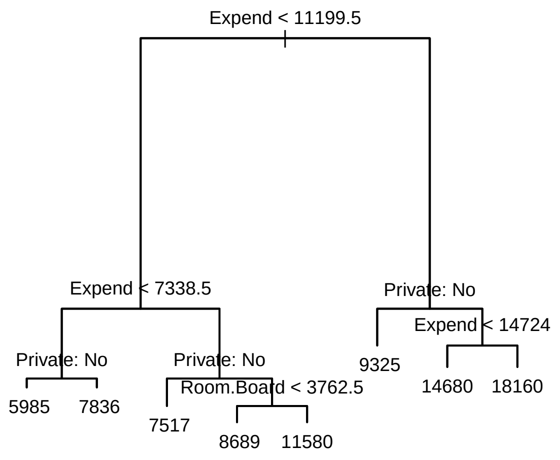
First, a regression tree was created.

```r
tree <- tree(Outstate ~.,data=college.train, method="anova")
plot(tree, cex = 0.6)
text(tree, pretty = 0, cex = 0.6)
```



Using cross-validation, the tree with the lowest deviance was found to be the tree with a size of 8. The tree was then pruned, and the MSE was calculated for the pruned tree.

```r
tree.cv <- cv.tree(tree)
tree.mindev = tree.cv$size[which.min(tree.cv$dev)]
tree.prune = prune.tree(tree, best = tree.mindev)
plot(tree.prune, cex = 0.6)
text(tree.prune, pretty = 0, cex = 0.6)
```



```r
tree.predict = predict(tree.prune, newdata = college.test)
MSE.tree = mean((tree.predict - college.test$Outstate)^2)
```

Then, doing the same using bagging, the MSE was calculated again.

```r
tree.bag <- bagging(Outstate~.,data=college.train, nbagg=25)
tree.bag.predict <- predict(tree.bag, college.test)
MSE.bag = mean((tree.bag.predict - college.test$Outstate)^2)
```

Finally, a random forest was created with the dataset.

```
tree.randomForest = randomForest(Outstate~., data = college.train, mtry =
                        ncol(college.train)/3, ntree = 500,importance = TRUE)
randomForest.predict = predict(tree.randomForest, newdata = college.test)
MSE.randomForest = mean((randomForest.predict - college.test$Outstate)^2)
```

The resulting MSEs are shown for the three methods in the following table.

Method	**MSE**
Regression tree	4.5×10^6
Bagging	3.3×10^6
Random forest	2.6×10^6

As can be seen, the MSE is lowest for the random forest model, and it is significantly lower than both the pruned regression tree, and the bagging. This is most probably because of the decorrelated trees. Bagging was better than regression trees

c)

The results from Problem 1-3 are shown in the table below, in chronological order.

Method	**MSE**
Forward selection	3.8×10^6
Lasso	3.7×10^6
Polynomial regression	1.4×10^7
Smoothing spline	6.9×10^6
Regression tree	4.5×10^6
Bagging	3.3×10^6
Random forest	2.6×10^6

In terms of prediction error, the best model is by far the random forest, compared to all the others. However, the random forest model is not the most interpretable model, and if that would be a goal, the pruned regression tree, lasso, or forward selection would be better models. They have a slightly higher MSE, but this might be worth it for the simplicity. All in all, the forward selection model might be the best model when interpretability is desired.

Problem 4

a) Multiple choice

(i) TRUE
(ii) TRUE
(iii) TRUE
(iv) TRUE

b)

First, we convert the variables to factors, and fit a support vector classifier using the `e1071` package and the `svm`function, and cross validation to find the best cost parameter.

```
d.train$diabetes <- as.factor(d.train$diabetes)
d.test$diabetes <- as.factor(d.test$diabetes)
library(e1071)

svm.linear = tune(svm,diabetes~.,data=d.train,kernel="linear",
                  ranges = list(cost=c(0.001,0.01,0.1,1,5,10,100)))
svm.linear.pred  = predict(svm.linear$best.model,d.test)
svm.linear.table = table(predict=svm.linear.pred, truth = d.test$diabetes)
svm.linear.error = sum(svm.linear.table[2:3]) / sum(svm.linear.table)
```

The confusion table and the misclassification error rate is:

```
svm.linear.table
```

```
##        truth
## predict   0   1
##       0 135  34
##       1  20  43
```

```
svm.linear.error
```

```
## [1] 0.2327586
```

Then, a support vector machine was fitted, again with cross validation, but this time to find the optimal combination of cost and γ.

```
svm.radial = tune(svm,diabetes~.,data=d.train,kernel="radial",
                       ranges = list(cost=c(0.001,0.01,0.1,1,5,10,100),
                                     gamma=c(0,0001, 0.001,0.01,0.1,1,5,10,100)))
svm.radial.pred  = predict(svm.radial$best.model,d.test)
svm.radial.table = table(predict=svm.radial.pred, truth = d.test$diabetes)
svm.radial.error = sum(svm.radial.table[2:3]) / sum(svm.radial.table)
```

The confusion table and the misclassification error rate is:

```
svm.radial.table
```

```
##        truth
## predict   0   1
##       0 140  38
##       1  15  39
```

```
svm.radial.error
```

```
## [1] 0.2284483
```

Comparing these two, the misclassification error rate is actually identical for the given test set, but there are some differences in the confusion matrices. There are more negative predictions in the radial model, both true and false negatives, 3 more on each. This is interesting, and shows the difference between the two types of boundaries and the impact a difference in cost and γ gives. For the given data, the preferred model would probably be the linear one, as this is both simpler, and gives a higher number of true positives. In the case of diabetes, misclassification in the form of false positives is better than false negatives, in our opinion.

c)

Comparing the SVMs to a linear discriminant analysis, the following code gives a fit using LDA.

```
lda.fit = lda(diabetes~., data = d.train)
lda.pred = predict(lda.fit,d.test)
```

```r
lda.table = table(predict=lda.pred$class, truth = d.test$diabetes)
lda.error = sum(lda.table[2:3]) / sum(lda.table)
lda.table
```

```
##        truth
## predict   0   1
##       0 137  34
##       1  18  43
```

```r
lda.error
```

```
## [1] 0.2241379
```

As can be seen, the misclassification rate is very similar, with only one less false negative compared to the support vector classifier. The main difference between the two methods is that the SVM uses only some observations as vectors to create the separating hyperplane, while the LDA uses all observations. This makes SVM less dependant on observations far from the hyperplane, while LDA is more affected by outliers in the data.

d) Multiple choice

(i) FALSE
(ii) FALSE
(iii) TRUE
(iv) TRUE

e) Link to logistic regression and hinge loss.

Using $f(\boldsymbol{x}_i)$ as corresponding to the linear predictor in the logistic regression approach,

$$f(\boldsymbol{x})_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}}}{1+e^{\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}}}$$

the logistic regression model is on the form

$$p_i = \frac{e^{f(\boldsymbol{x}_i)}}{1+e^{f(\boldsymbol{x}_i)}}.$$

In logistic regression, the observations contribute by a weight $p_i(1-p_i)$, so the regression model can be rewritten to

$$f(\boldsymbol{x}_i) = \log(\frac{p_i}{1-p_i})$$

This means that the loss function

$$\log(1+exp(-y_i f(\boldsymbol{x}_i))$$

is the deviance for the $y=-1,1$ encoding in a logistic regression model.

Problem 5

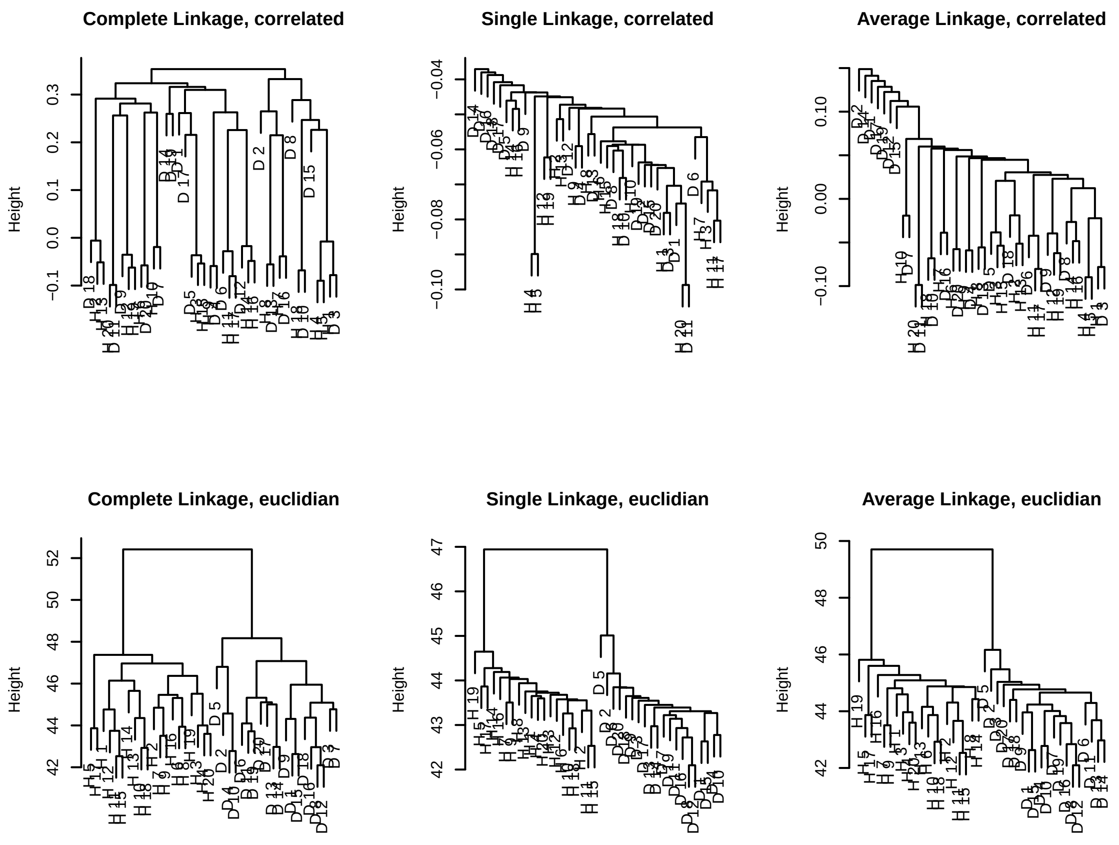
a)

Performing hierarchical clustering with both correlation and euclidian distance metrics.

```
hc.complete = hclust(dist(GeneData, method = "euclidean", diag = FALSE, upper = FALSE,
                          p = 2), method ="complete")
hc.single = hclust(dist(GeneData, method = "euclidean", diag = FALSE, upper = FALSE,
                        p = 2), method ="single")
hc.average = hclust(dist(GeneData, method = "euclidean"), method = "average")

dd = as.dist(cor(t(GeneData)))

corr.comp <- hclust(dd, method ="complete")
corr.av <- hclust(dd, method ="average")
corr.single <- hclust(dd, method ="single")
```



b)

As can be seen, the correlation based dendrogrammes are all over the place, with no clear way of classifying the samples into two classes. The euclidian distance based dendrograms however, are a different story. All of them classify the two classes perfectly, with no means of determining which one is the best. Usually, average or complete linking is better than single linking.

c)

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

ϕ is the weight, while X is the variable. P is the number of elements in the principal components, while n is the number of samples, or more specifically, the number of principal components.

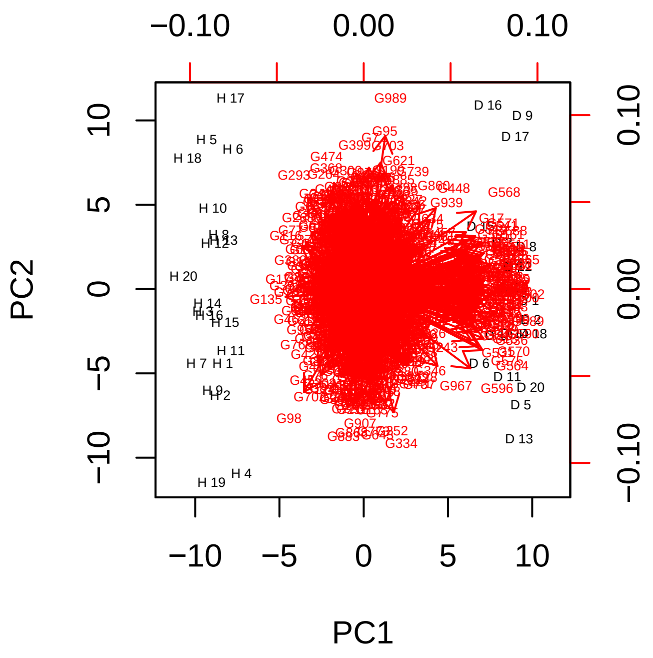
The principal component score of the first principal component can be expressed as:

$$z_1 = \sum_{i=1}^{n} \sum_{j=1}^{p} \phi_{j1} X_{ij}$$

d)

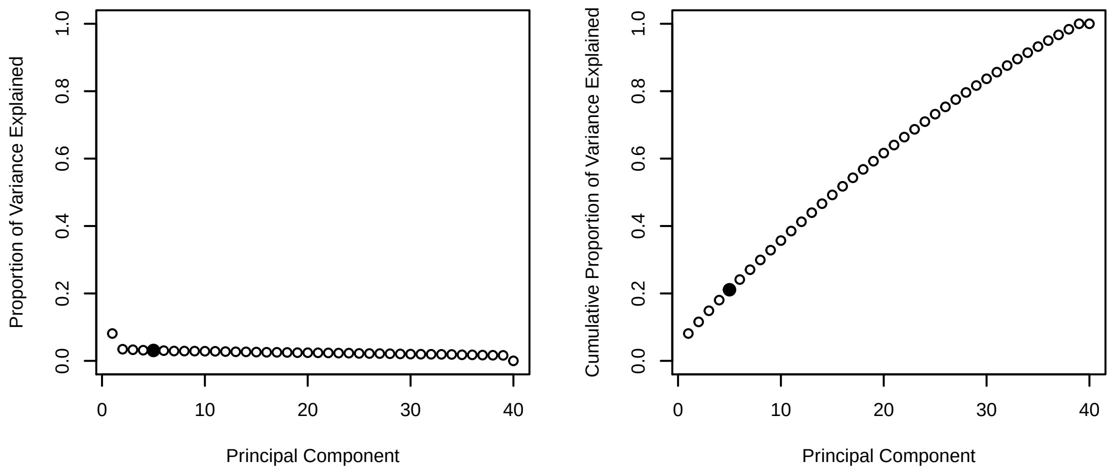
(i)

```r
pca = prcomp(GeneData, scale=TRUE)
biplot(pca, scale = 0, cex = 0.4)
```



```r
par(mfrow=c(1,2), cex = 0.6)
pca.var = pca$sdev^2
pve = pca.var/sum(pca.var)

plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained",
     ylim=c(0,1),type='b')
points(5,pve[5], col="black",cex=2,pch=20)
plot(cumsum(pve), xlab="Principal Component",
     ylab="Cumulative Proportion of Variance Explained",
     ylim=c(0,1),type='b')
points(5,cumsum(pve)[5], col="black",cex=2,pch=20)
```



```r
print(cumsum(pca.var[1:5]))
```

```
## [1]  81.08273 115.57514 148.56256 180.12518 210.96586
```

(ii)

We can see that the 5 first principal components stand for 21% of the total variance.

e)

In order to find the genes that vary the most across one can perform PCA on the untransposed dataset GeneData. The Principal Components that have the most variance would be the genes that vary the most.

```r
tr.genedata <- t(GeneData)
pca.gene = prcomp(tr.genedata, scale=TRUE)
gene <- pca.gene$rotation[,1]
gene_abs <- abs(gene)
gene_sorted <- sort(gene_abs, decreasing = TRUE)
print(names(gene_sorted[1:5]))
```

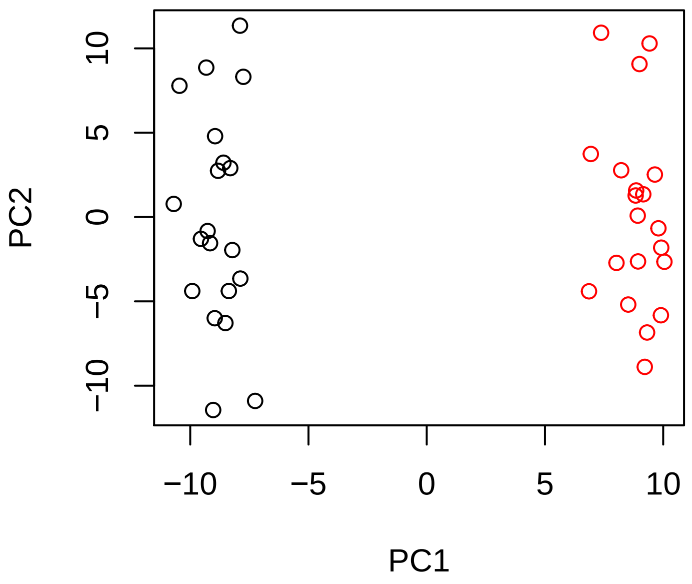
```
## [1] "D 8"  "D 2"  "D 19" "D 14" "D 1"
```

Thus, the 5 genes that contribute most variance to PC1 are; D8, D2, D19, D14 and D1.

f)

K-means was used to separate the tissue samples into two groups.

```r
km <- kmeans(GeneData, 2, nstart=20)
plot(pca$x[,1:2], col=km$cluster, pca=c(GeneData[1:20], GeneData[21:40]))
```



The error rate of the K-means is zero. This can be seen by the perfect separation of the two groups.