Compulsory exercise 2: Group 13

TMA4268 Statistical Learning V2020

Vemund Tjessem, Erik Andre Klepp Vik

05 april, 2020

```
# install.packages('knitr') #probably already installed
# install.packages('rmarkdown') #probably already installed
# install.packages('ggplot2') #plotting with ggplot install.packages('ggfortify')
# install.packages('MASS') install.packages('dplyr')
library(knitr)
library(rmarkdown)
library(ggplot2)
library(ggfortify)
library(GGally)
library(MASS)
library(dplyr)
library(ISLR)
library(leaps)
library(glmnet)
library(tree)
library(randomForest)
library(e1071)
```

Problem 1

a)

The ridge regression coefficients β_{Ridge} are the ones that minimize

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

with $\lambda > 0$ being a tuning parameter. The residual sum of squares is defined as

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

Will assume that X has been centered such that the mean is zero, i.e $\beta_0 \approx 0$. It is also smart to standardize the predictors before using ridge regression, as ridge regression is not scale invariant. Equation 1 can be rewritten in terms of matrices and vectors as

$$(y - X\hat{\beta}_{Ridge})^\top (y - X\hat{\beta}_{Ridge}) + \lambda \hat{\beta}_{Ridge}^\top \hat{\beta}_{Ridge} \tag{3}$$

Differentiating this with respect to $\hat{\beta}_{Ridge}$ and setting equal to 0 gives

$$-2X^\top (y - X\hat{\beta}_{Ridge}) + 2\lambda \hat{\beta}_{Ridge} = 0 \tag{4a}$$
$$X^\top X\hat{\beta}_{Ridge} + \lambda \hat{\beta}_{Ridge} = X^\top y \tag{4b}$$
$$\hat{\beta}_{Ridge} = (X^\top X + \lambda I)^{-1} X^\top y \tag{4c}$$

Where I is the identity matrix.

b)

The expectation value of $y = X\beta + \epsilon$ is $\mathrm{E}[y] = X\beta$, as $\mathrm{E}[\epsilon] = 0$. The expectation value of β_{Ridge} is then

$$\mathrm{E}[\hat{\beta}_{Ridge}] = (X^\top X + \lambda I)^{-1} X^\top \mathrm{E}[y] \tag{5a}$$
$$= (X^\top X + \lambda I)^{-1} X^\top X\beta \tag{5b}$$

This is a biased estimator as long as $\lambda \neq 0$.

The variance covariance matrix of y is $\mathrm{Var}[y] = \mathrm{Var}[X\beta] + \mathrm{Var}[\epsilon] = \sigma^2$.

$$\mathrm{Var}[\hat{\beta}_{Ridge}] = \mathrm{Var}[(X^\top X + \lambda I)^{-1} X^\top y] \tag{6a}$$
$$= (X^\top X + \lambda I)^{-1} X^\top \mathrm{Var}[y][(X^\top X + \lambda I)^{-1} X^\top]^\top \tag{6b}$$
$$= \sigma^2 (X^\top X + \lambda I)^{-1} X^\top X[(X^\top X + \lambda I)^{-1}]^\top \tag{6c}$$

c)

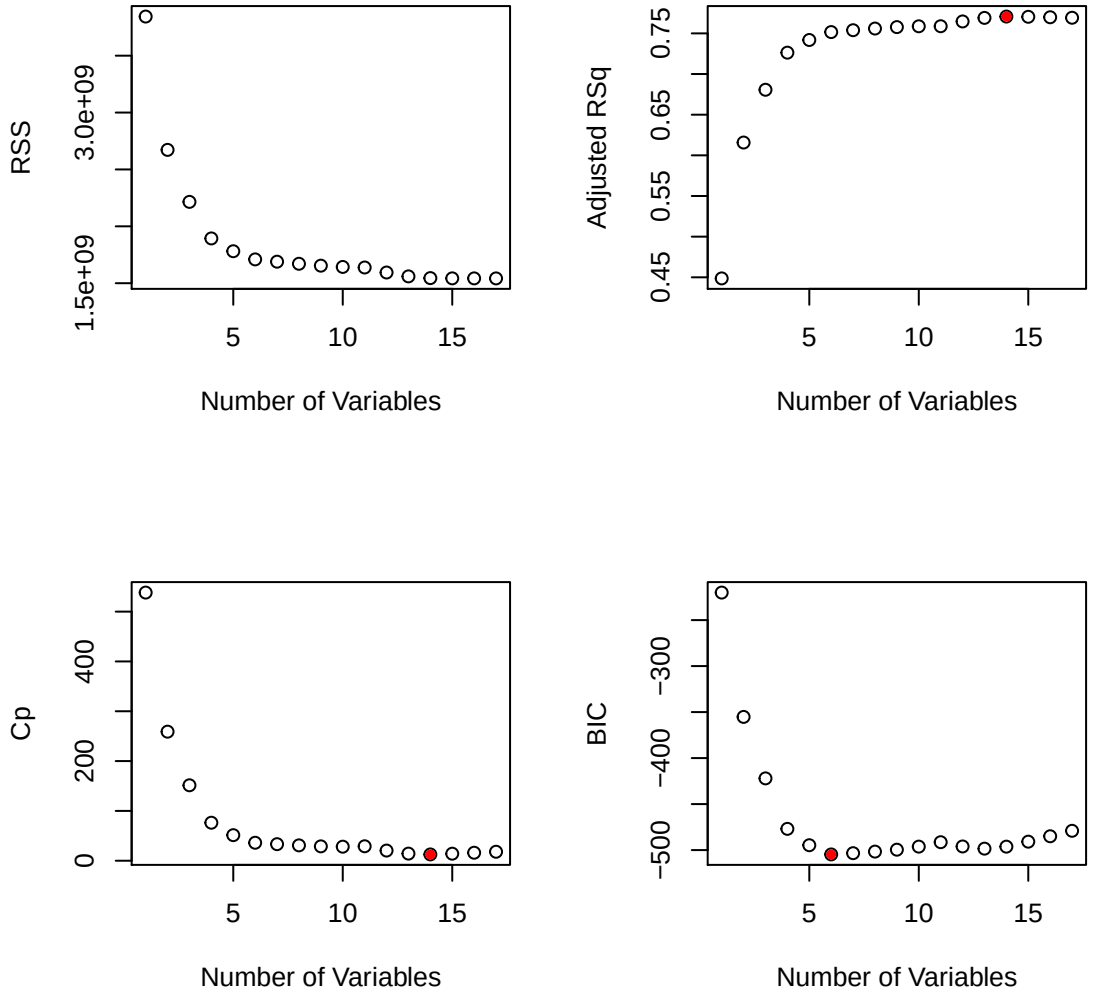
(i) True
(ii) False
(iii) False
(iv) True

d)

Forward selection will be performed with `Outstate` as response using the `regsubsets` function.

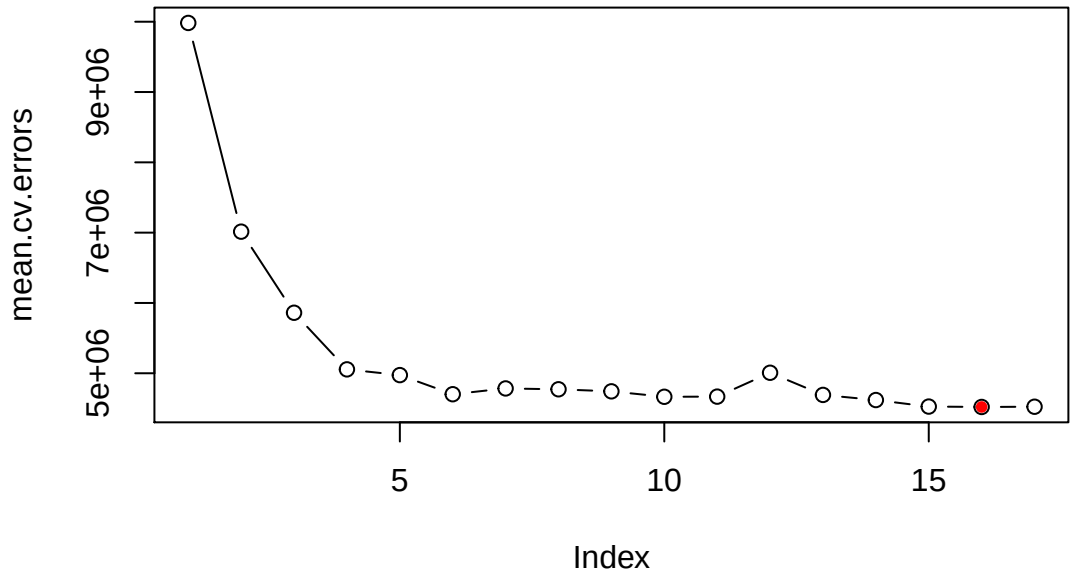
```
set.seed(1)
train.ind = sample(1:nrow(College), 0.5 * nrow(College))
college.train = College[train.ind, ]
college.test = College[-train.ind, ]
n_predictors = dim(College)[2] - 1
fwd.fit = regsubsets(Outstate ~ ., college.train, nvmax = n_predictors, method = "forward")
fwd.fit.summary = summary(fwd.fit)
par(mfrow = c(2, 2))
```

```
plot(fwd.fit.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = )
plot(fwd.fit.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq")
fwd_best_adjr2 = which.max(fwd.fit.summary$adjr2)
points(fwd_best_adjr2, fwd.fit.summary$adjr2[fwd_best_adjr2], col = "red", cex = 1,
    pch = 20)
plot(fwd.fit.summary$cp, xlab = "Number of Variables", ylab = "Cp")
fwd_best_cp = which.min(fwd.fit.summary$cp)
points(fwd_best_cp, fwd.fit.summary$cp[fwd_best_cp], col = "r ed", cex = 1, pch = 20)
fwd_best_bic = which.min(fwd.fit.summary$bic)
plot(fwd.fit.summary$bic, xlab = "Number of Variables", ylab = "BIC")
points(fwd_best_bic, fwd.fit.summary$bic[fwd_best_bic], col = "red", cex = 1, pch = 20)
```




```
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
```

```
    xvars = names(coefi)
    mat[, xvars] %*% coefi
}
k = 10
set.seed(1)
folds = sample(1:k, nrow(college.train), replace = TRUE)
cv.errors = matrix(NA, k, n_predictors, dimnames = list(NULL, paste(1:n_predictors)))
# Perform CV
for (j in 1:k) {
    best_subset_method = regsubsets(Outstate ~ ., data = college.train[folds != j,
        ], nvmax = n_predictors, method = "forward")
    for (i in 1:n_predictors) {
        pred = predict(best_subset_method, college.train[folds == j, ], id = i)
        cv.errors[j, i] = mean((college.train$Outstate[folds == j] - pred)^2)
    }
}
# Compute mean cv errors for each model size
mean.cv.errors = apply(cv.errors, 2, mean)
# Plot the mean cv errors
par(mfrow = c(1, 1))
plot(mean.cv.errors, type = "b")
min_cverror = which.min(mean.cv.errors)
points(min_cverror, mean.cv.errors[min_cverror], col = "red", cex = 1, pch = 20)
```




```
# Calculating the MSE for model with 6 predictors
x.test = model.matrix(Outstate ~ ., data = college.test)
coef6 = coef(fwd.fit, id = 6)
pred = x.test[, names(coef6)] %*% coef6
```

```
MSE.forward = mean((college.test$Outstate - pred)^2)
co.names = names(coef6)[-1]  # Extract the names of the predictors used, minus the intercept
co.names[1] = "Private"  # change name from PrivateYes to Private
```

The obvious choice might be the model with 14 predictors, as this had both the highest adjusted R^2 and the smallest C_p. However, since the improvement is very small for the larger models it may be unnecesary to have such a large model. See that the model with 6 predictors has the smallest BIC. BIC is defined in a way that normally favors a smaller model. Cross validation also shows that 6 would be a good choice. It is not the one with the lowest mean error, but it is quite good compared to the rest and better than both 5 and 7. The 6 predictors are Private, Room.Board, Terminal, perc.alumni, Expend, Grad.Rate, which gives a model on the form

$$Y = \beta_0 + \beta_1 X_{\text{Private}} + \beta_2 X_{\text{Room.Board}} + \beta_3 X_{\text{Terminal}} + \beta_4 X_{\text{perc.alumni}} + \beta_5 X_{\text{Expend}} + \beta_6 X_{\text{Grad.Rate}} + \epsilon \quad (7)$$

vet ikke hvilken av disse som bør brukes

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_{\text{Private}} + \hat{\beta}_2 x_{\text{Room.Board}} + \hat{\beta}_3 x_{\text{Terminal}} + \hat{\beta}_4 x_{\text{perc.alumni}} + \hat{\beta}_5 x_{\text{Expend}} + \hat{\beta}_6 x_{\text{Grad.Rate}} \quad (8)$$

The model with 6 predictors has a MSE of 3.8448572×10^6.

e)

Model selection using the Lasso method. Since the package `glmnet` does not use the model formula language we need to set up `x` and `y`.

```
x.train = model.matrix(Outstate ~ ., data = college.train)[, -1]  # -1 is to remove intercept
y.train = college.train$Outstate
x.test = model.matrix(Outstate ~ ., data = college.test)[, -1]
y.test = college.test$Outstate
lasso.fit = glmnet(x.train, y.train, alpha = 1)  # alpha = 1 gives the Lasso method
set.seed(1)
lasso.fit.cv = cv.glmnet(x.train, y.train, alpha = 1)
lasso.lambda = lasso.fit.cv$lambda.1se
lasso.pred = predict(lasso.fit, s = lasso.lambda, newx = x.test)
MSE.lasso = mean(as.numeric((lasso.pred - y.test)^2))
lasso.coeffs = coef(lasso.fit, s = lasso.lambda)
nonzero.names = rownames(lasso.coeffs)[lasso.coeffs[, 1] != 0]
nonzero.names[2] = "Private"
```

Used the function `cv.glmnet` to perform 10 fold cross validation and choose a value for λ. Instead of choosing the model with the lowest MSE in the cross validation, which used all the predictors, we chose the value `lamdba.1se` which is the largest value of λ which gives an error within 1 standard error of the minimum. The value was $\lambda = 367.77$. The reason for this is that it is a much smaller model, which only uses 8 predictors. The predictors were Private, Top10perc, Room.Board, Personal, Terminal, S.F.Ratio, perc.alumni, Expend, Grad.Rate. The MSE on the test set was 3.9033653×10^6.

Problem 2

a)

(i) False

(ii) False
(iii) True
(iv) True

b)

The basis functions are

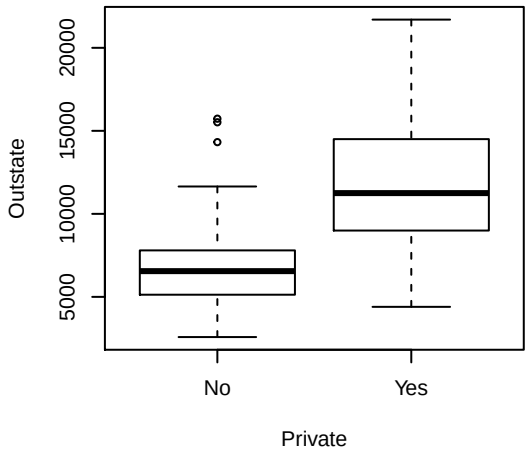
$$b_1(x) = x^1 \tag{9a}$$
$$b_2(x) = x^2 \tag{9b}$$
$$b_3(x) = x^3 \tag{9c}$$
$$b_4(x) = (x - q_1)_+^3 \tag{9d}$$
$$b_5(x) = (x - q_2)_+^3 \tag{9e}$$
$$b_6(x) = (x - q_3)_+^3 \tag{9f}$$

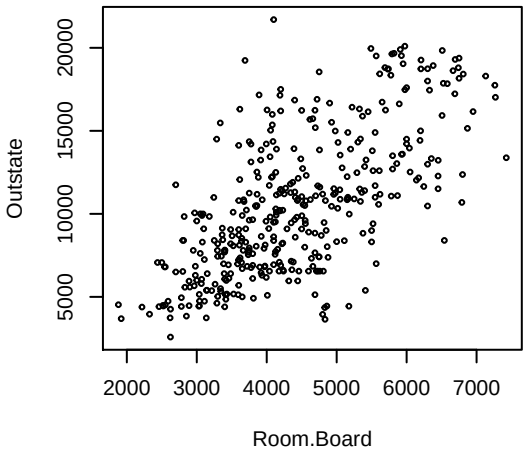
c)

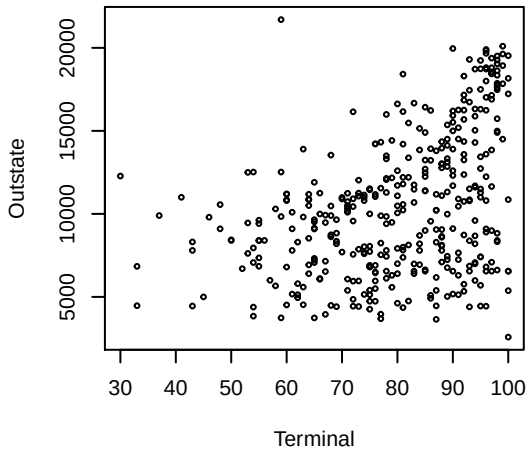
```
for.reg = regsubsets(Outstate ~ ., data = college.train, method = "forward")
coef.for = coef(for.reg, id = 6)
co.names = names(coef.for)[-1]
co.names[1] = "Private"
```

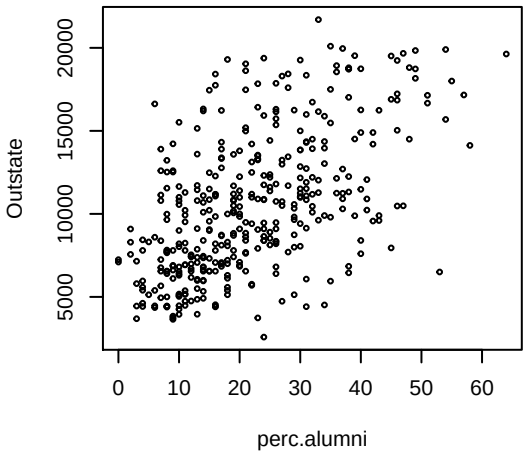
Will investigate the relationship between `Outstate` and the following 6 predictors: Private, Room.Board, Terminal, perc.alumni, Expend, Grad.Rate.

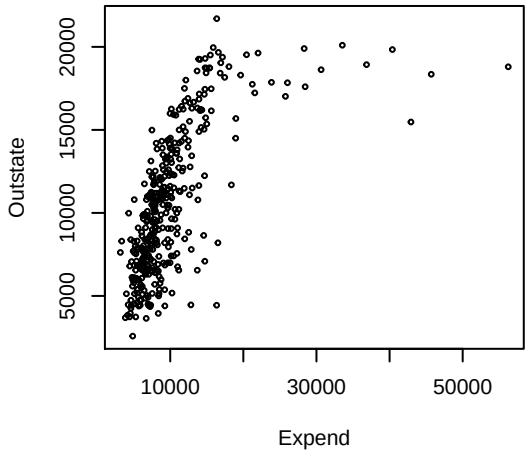
```
par(mfrow = c(3, 2))
plot(Outstate ~ Private, data = college.train)
plot(Outstate ~ Room.Board, data = college.train, cex = 0.5)
plot(Outstate ~ Terminal, data = college.train, cex = 0.5)
plot(Outstate ~ perc.alumni, data = college.train, cex = 0.5)
plot(Outstate ~ Expend, data = college.train, cex = 0.5)
plot(Outstate ~ Grad.Rate, data = college.train, cex = 0.5)
```

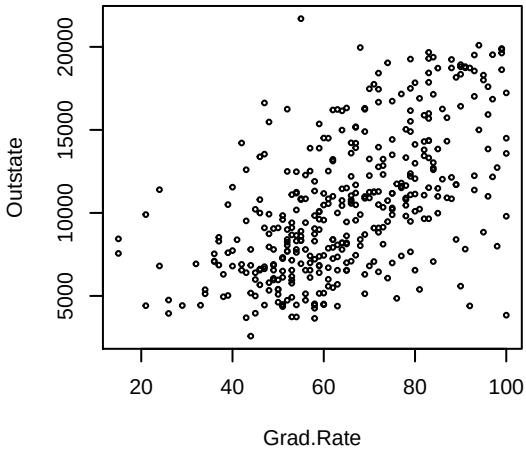

Outstate
5000
10000
15000
20000
No
Yes
Private


Outstate
5000
10000
15000
20000
2000
3000
4000
5000
6000
7000
Room.Board


Outstate
5000
10000
15000
20000
30
40
50
60
70
80
90
100
Terminal


Outstate
5000
10000
15000
20000
0
10
20
30
40
50
60
perc.alumni


Outstate
5000
10000
15000
20000
10000
30000
50000
Expend

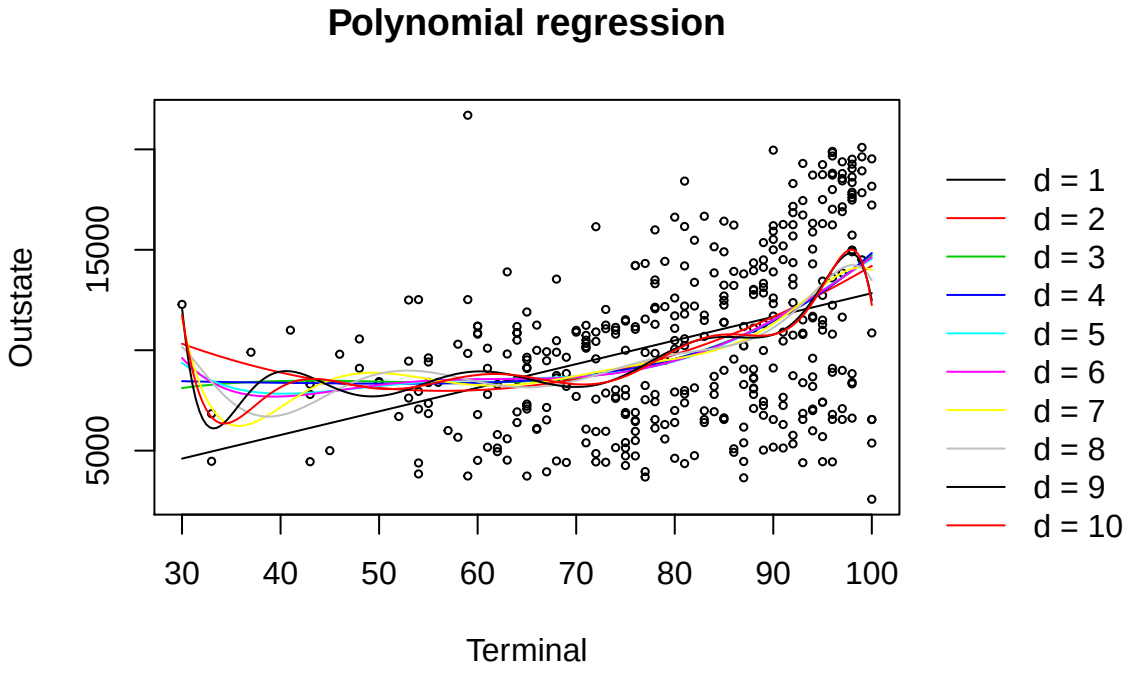

Outstate
5000
10000
15000
20000
20
40
60
80
100
Grad.Rate

The relationship between `Outstate` and `Room.Board` seems to be approximately linear, same for `perc.alumni` although it is very spread out. For `Terminal` on the other hand the slope seems to increase with increasing value for `Terminal`, it could maybe benefit from a non-linear transformation. The relation between `Outstate` and `Expend` does not seem linear and would most likely benefit from a non-linear transformation, `Grad.Rate` on the other hand would probably not benefit much.

d)

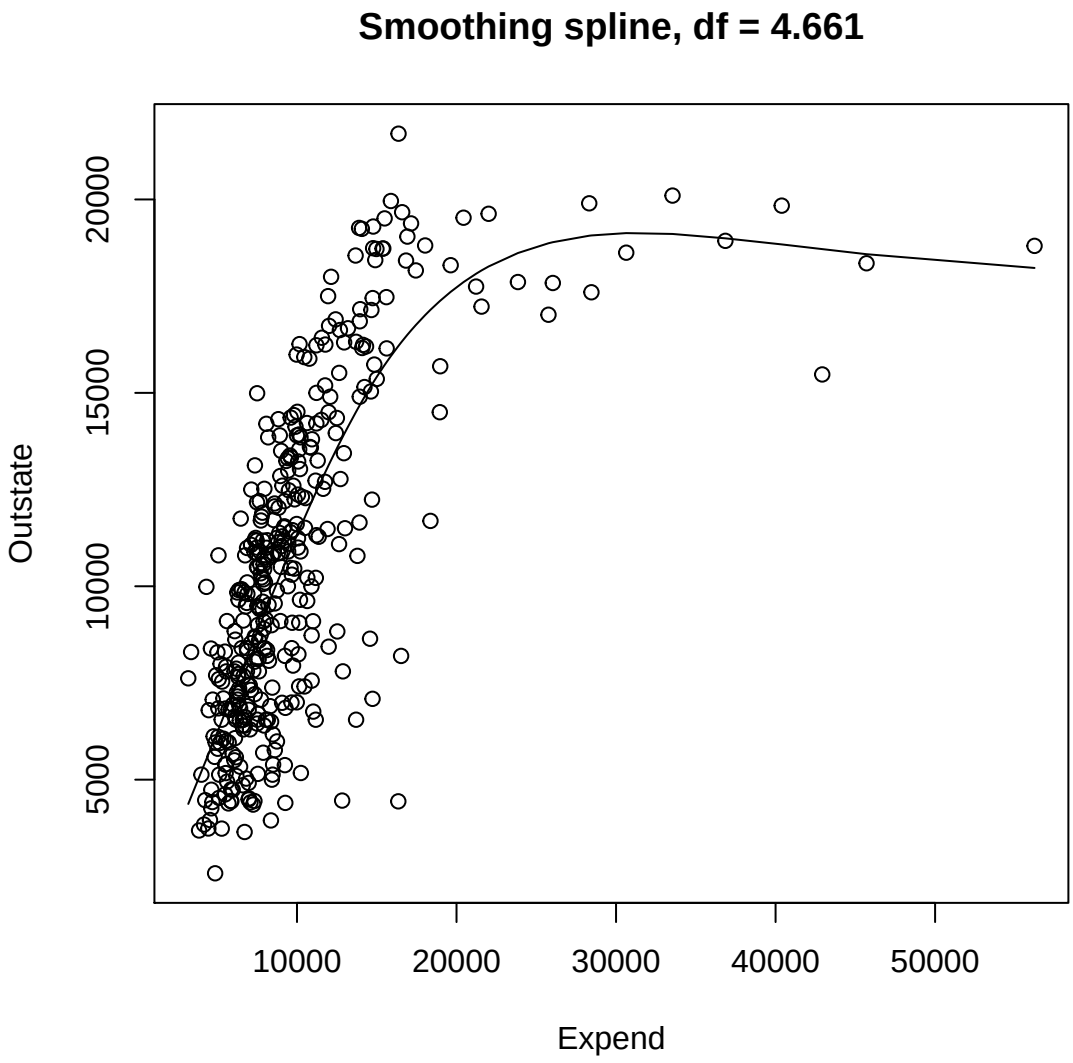
(i) Fit polynomial regression models for `Outstate` as a function of `Terminal` with polynomial degrees $d = 1, \ldots, 10$.

```
par(mar = c(5.1, 4.1, 4.1, 6.5), xpd = TRUE)
degs = 10
MSE.poly.train = rep(NA, degs)
MSE.poly.test = rep(NA, degs)
plot(Outstate ~ Terminal, data = college.train, main = "Polynomial regression", cex = 0.5)
d <- seq(min(college.train$Terminal), max(college.train$Terminal), length.out = 200)
for (degree in 1:degs) {
    fm <- lm(Outstate ~ poly(Terminal, degree), data = college.train)
    assign(paste("college.train", degree, sep = "."), fm)
    lines(d, predict(fm, data.frame(Terminal = d)), col = degree)
    # Calculate training MSE
    MSE.poly.train[degree] = mean((predict(fm, college.train) - college.train$Outstate)^2)
    MSE.poly.test[degree] = mean((predict(fm, college.test) - college.test$Outstate)^2)
}
legend("topright", inset = c(-0.32, 0.1), legend = paste("d =", 1:degs), col = c(1:degs),
    lty = 1, bty = "n")
```



(ii) Choose a suitable smoothing spline model to predict `Outstate` as a function of `Expend`.

```
x = college.train$Expend
y = college.train$Outstate
smthspl.fit = smooth.spline(x, y, cv = T)
plot(y ~ x, main = paste("Smoothing spline, df =", round(smthspl.fit$df, 3)), xlab = "Expend",
    ylab = "Outstate")
lines(smthspl.fit)
```

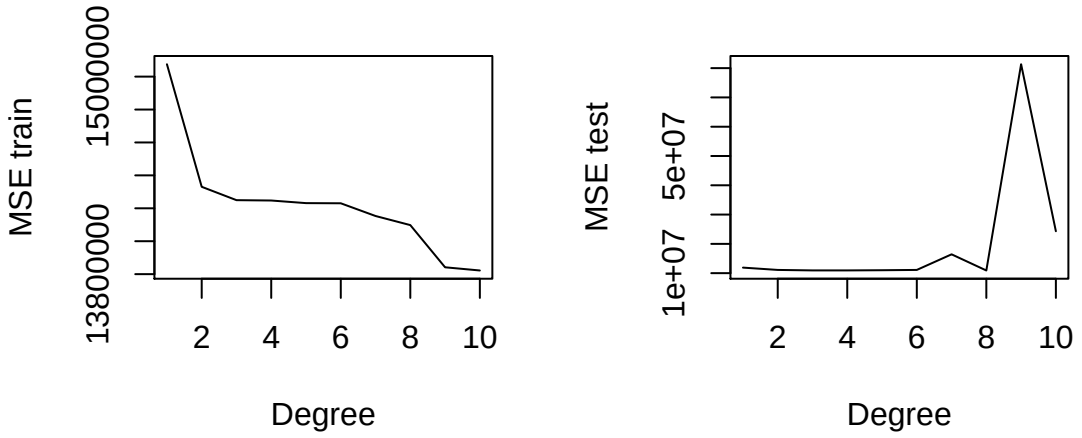



```
# points(college.test$Expend, college.test$Outstate, pch=2, col=2)
MSE.smthspl.train = mean((predict(smthspl.fit, x)$y - y)^2)
MSE.smthspl.test = mean((predict(smthspl.fit, college.test$Expend)$y - college.test$Outstate)^2)
```

By putting `cv=T` cross validation is used to determine the degrees of freedom. They are determined to be 4.660711, higher values of `df` gives a more overfitted line.

(iii) Training MSE

```
par(mfrow = c(1, 2))
plot(1:degs, MSE.poly.train, type = "l", xlab = "Degree", ylab = "MSE train")
plot(1:degs, MSE.poly.test, type = "l", xlab = "Degree", ylab = "MSE test")
```




```
best.train = which.min(MSE.poly.train)
```

The smallest training error of the polynomials was 1.3822205×10^7, which corresponds to the polynomial of degree 10. As can be seen from the plot to the left the training error decreases with increasing degree of the polynomial. This is expected since an increase in the order of a polynomial makes it more flexible and allows it to fit the training data better. However, even though the training error decreases it does not mean the model is better, as can be seen in the right plot the test error increases for 7 and 9. This is known as overfitting.

The training MSE for the smoothing spline is 6.8712814×10^6. It was expected that the smoothing spline would have a lower training MSE as it uses the predictor `Expend`, which is less spread than `Terminal` which is what the polynomials are fit on.

Problem 3

a)

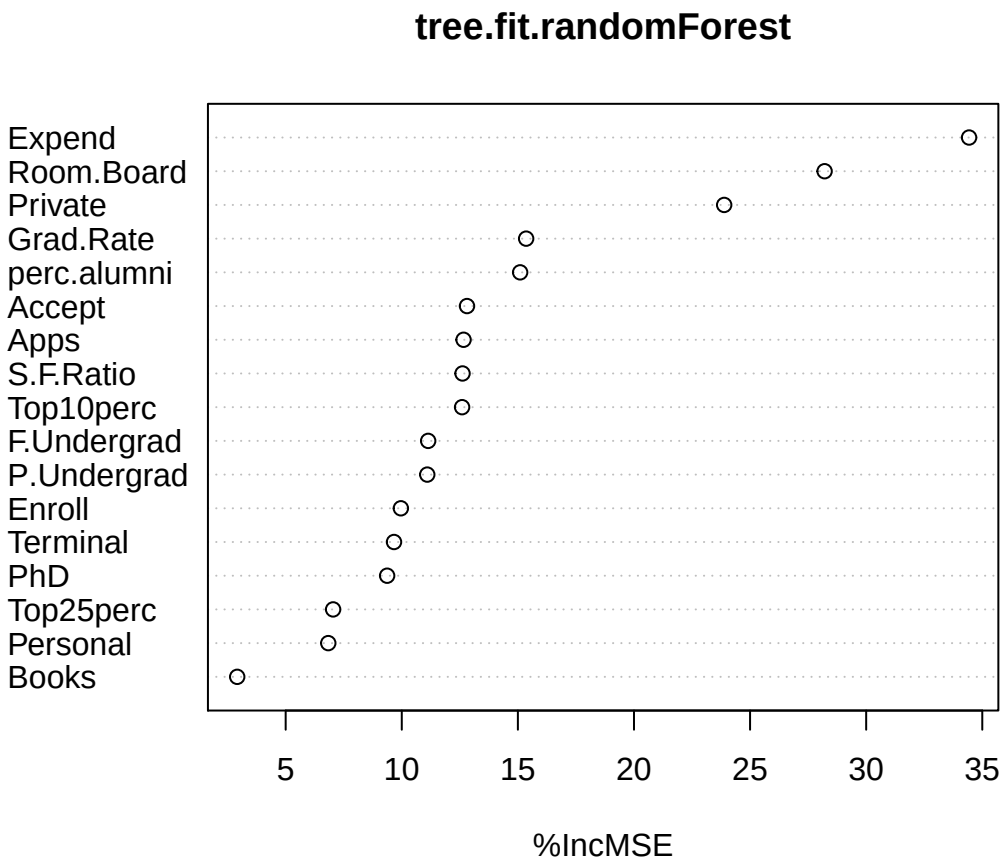
(i) False
(ii) True
(iii) True
(iv) False

b)

Will use random forest as there are a few strong predictors which would make the trees correlated if normal bagging was used. A random forest can help decrease the variance by injecting more randomness. This is

a regression tree so $m = p/3$ will be used. A disadvantage of using a random forest is that it does not give one tree, which makes it difficult to visualize and interperet compared to for example a pruned tree.

```
set.seed(1)
tree.fit.randomForest = randomForest(Outstate ~ ., data = college.train, mtry = ncol(college.train)/3,
    ntree = 500, importance = TRUE)
yhat.randomForest = predict(tree.fit.randomForest, newdata = college.test)
MSE.randomForest = mean((yhat.randomForest - college.test$Outstate)^2)
# importance(tree.fit.randomForest)
varImpPlot(tree.fit.randomForest, type = 1)
```



c)

Compare square root of the MSEs of the different methods. We hae taken the square root of the test errors since it makes it much easier to compare since the numers are smaller, and the order is not changed.

```
sqrt(MSE.forward)
```

```
## [1] 1960.831
```

```
sqrt(MSE.lasso)
```

```
## [1] 1975.694
```

```
best.poly = which.min(MSE.poly.test)
sqrt(MSE.poly.test[best.poly])
```

```
## [1] 3303.655
```

```
sqrt(MSE.smthspl.test)
```

```
## [1] 2437.966
```

```
sqrt(MSE.randomForest)
```

```
## [1] 1602.157
```

The best method in terms of test error is the random forest model. If the goal was to develop an interpretable model the best may be a pruned tree, assuming it is not too bushy as it easy to visualize when it is just a single tree. From the models we have fit the most interpretable might be the forward selected model, as this is a linear model which also gives quite good performance.

Problem 4

```
id <- "1Fv6xwKLSZHldRAC1MrcK2mzdOYnbgv0E"  # google file ID
d.diabetes <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
d.train = d.diabetes$ctrain
d.test = d.diabetes$ctest
d.train$diabetes <- as.factor(d.train$diabetes)
d.test$diabetes <- as.factor(d.test$diabetes)
```

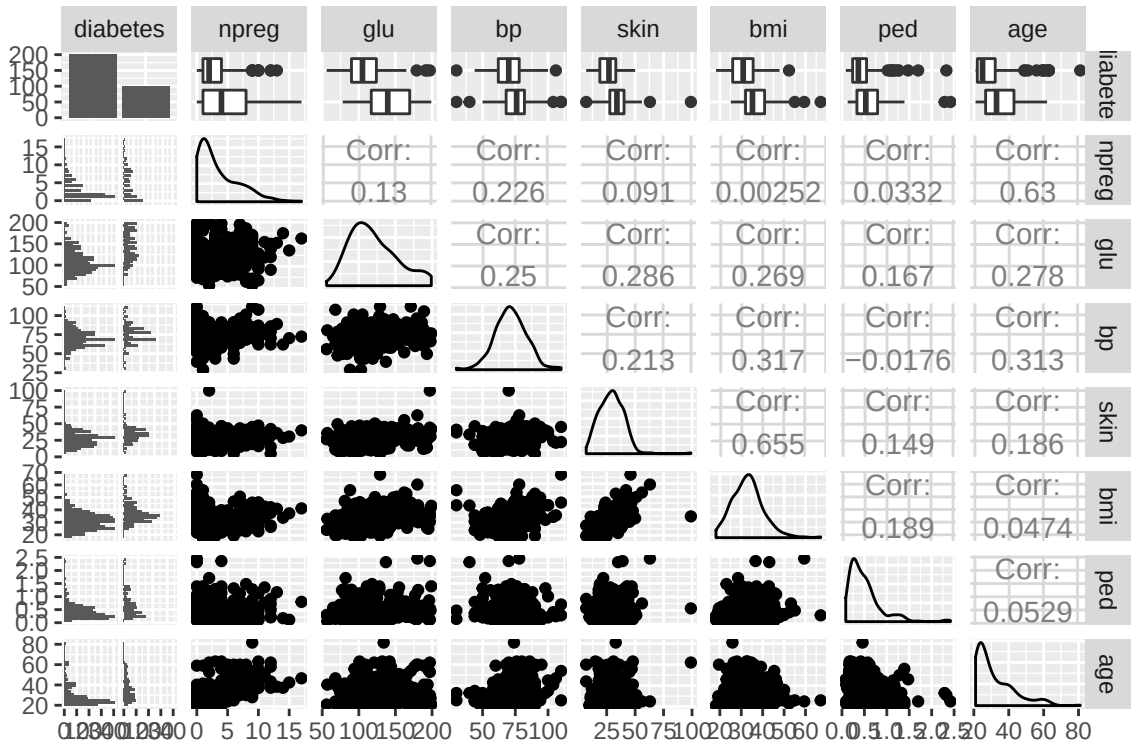
a)

```
summary(d.train)
```

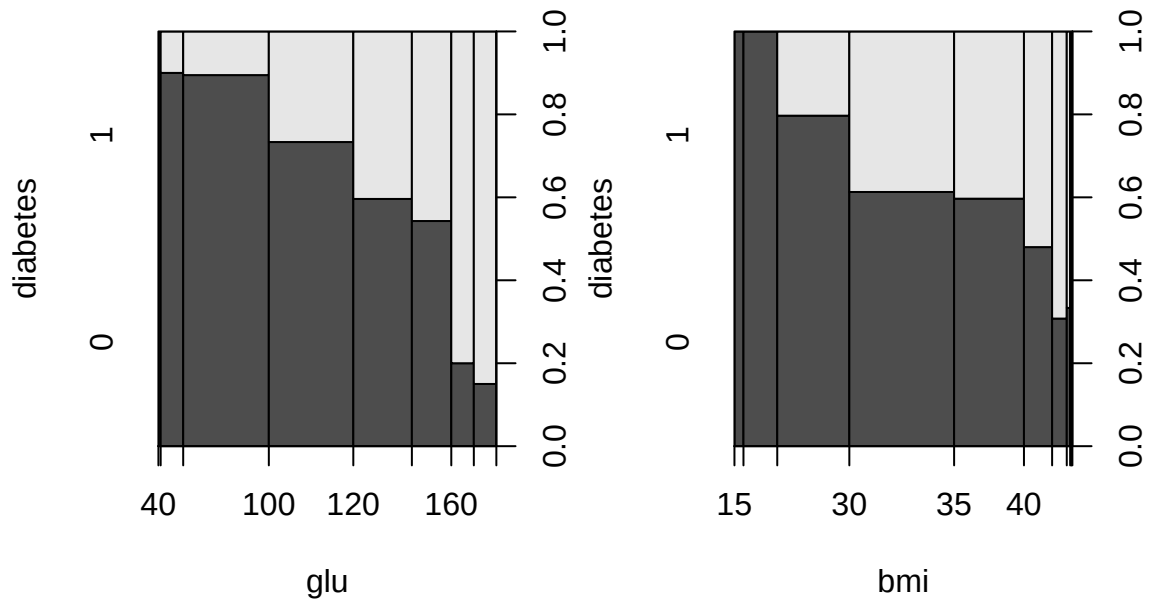
```
##  diabetes     npreg             glu               bp              skin
##  0:200    Min.   : 0.000   Min.   : 56.00   Min.   : 30.00   Min.   : 7.00
##  1:100    1st Qu.: 1.000   1st Qu.: 96.75   1st Qu.: 64.00   1st Qu.:22.00
##           Median : 2.000   Median :114.00   Median : 71.00   Median :29.00
##           Mean   : 3.467   Mean   :120.13   Mean   : 71.56   Mean   :29.14
##           3rd Qu.: 5.250   3rd Qu.:140.25   3rd Qu.: 80.00   3rd Qu.:36.00
##           Max.   :17.000   Max.   :199.00   Max.   :110.00   Max.   :99.00
##       bmi             ped              age
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00
##  1st Qu.:27.98   1st Qu.:0.2567   1st Qu.:23.00
```

```
##  Median :32.80   Median :0.4150   Median :27.00
##  Mean   :33.03   Mean   :0.5004   Mean   :31.55
##  3rd Qu.:37.12   3rd Qu.:0.6210   3rd Qu.:37.25
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

```
ggpairs(d.train)
```




```
par(mfrow = c(1, 2))
plot(diabetes ~ glu, data = d.train)
plot(diabetes ~ bmi, data = d.train)
```



(i) True
(ii) True
(iii) True
(iv) True? Ser på ggpairs plotet at sannsynlighetsfordeligen er forskjøvet mot 0

b)

Support vector classifier with a linear boundary.

```
set.seed(10)
CV_linear = tune(svm, diabetes ~ ., data = d.train, kernel = "linear", ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 50)))
# summary(CV_linear)
best_model_lin = CV_linear$best.model
# summary(best_model_lin)
y_pred_lin = predict(best_model_lin, d.test)
confMat_lin = table(predict = y_pred_lin, truth = d.test$diabetes)
confMat_lin
```

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

```
misrate_lin = 1 - sum(diag(confMat_lin))/sum(confMat_lin)
misrate_lin
```

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

Support vector machine with radial boundary.

```
set.seed(10)
CV_radial = tune(svm, diabetes ~ ., data = d.train, kernel = "radial", ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 50), gamma = c(0.01, 0.1, 1, 10, 100)))
# summary(CV_radial)
best_model_rad = CV_radial$best.model
# summary(best_model_rad)
y_pred_rad = predict(best_model_rad, d.test)
confMat_rad = table(predict = y_pred_rad, truth = d.test$diabetes)
confMat_rad
```

```
##        truth
## predict   0   1
##       0 139  37
##       1  16  40
```

```
misrate_rad = 1 - sum(diag(confMat_rad))/sum(confMat_rad)
misrate_rad
```

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

MANGLER DISKUSJON HER

c)

MANGLER DISKUSJON HER.

```
mylogit <- glm(diabetes ~ ., data = d.train, family = "binomial")
mylogit.pred <- predict(mylogit, d.test[, -1], type = "response")
# Confusion matrix
confusionMatrix(d.test$diabetes, mylogit.pred)

# Misclassification error
misClassError(d.test$diabetes, mylogit.pred)
```

d)

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

e)

```
# https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-iii-5dff33fa015d
```

Lets assume $f(\boldsymbol{x}_i)$ correponds to the linear predictor in the logistic regression approach, that is, from module 4:

$$f(\boldsymbol{x}_i) = \frac{e^{\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}}}{1+e^{\beta_0+\beta_1 x_{i1}+\ldots+\beta_p x_{ip}}} \tag{10}$$

This means that the logistic regression model will take the form

$$p_i = \frac{e^{f(\boldsymbol{x}_i)}}{1+e^{f(\boldsymbol{x}_i)}} \tag{11}$$

Since all observations in logisitic regression contribute weighted by $p_i(1-p_i)$, we rearrange the regression model from model 4 to get this form, and we get

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

That is, the loss function

$$\log(1+\exp(-y_i f(\boldsymbol{x}_i))) \tag{12}$$

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

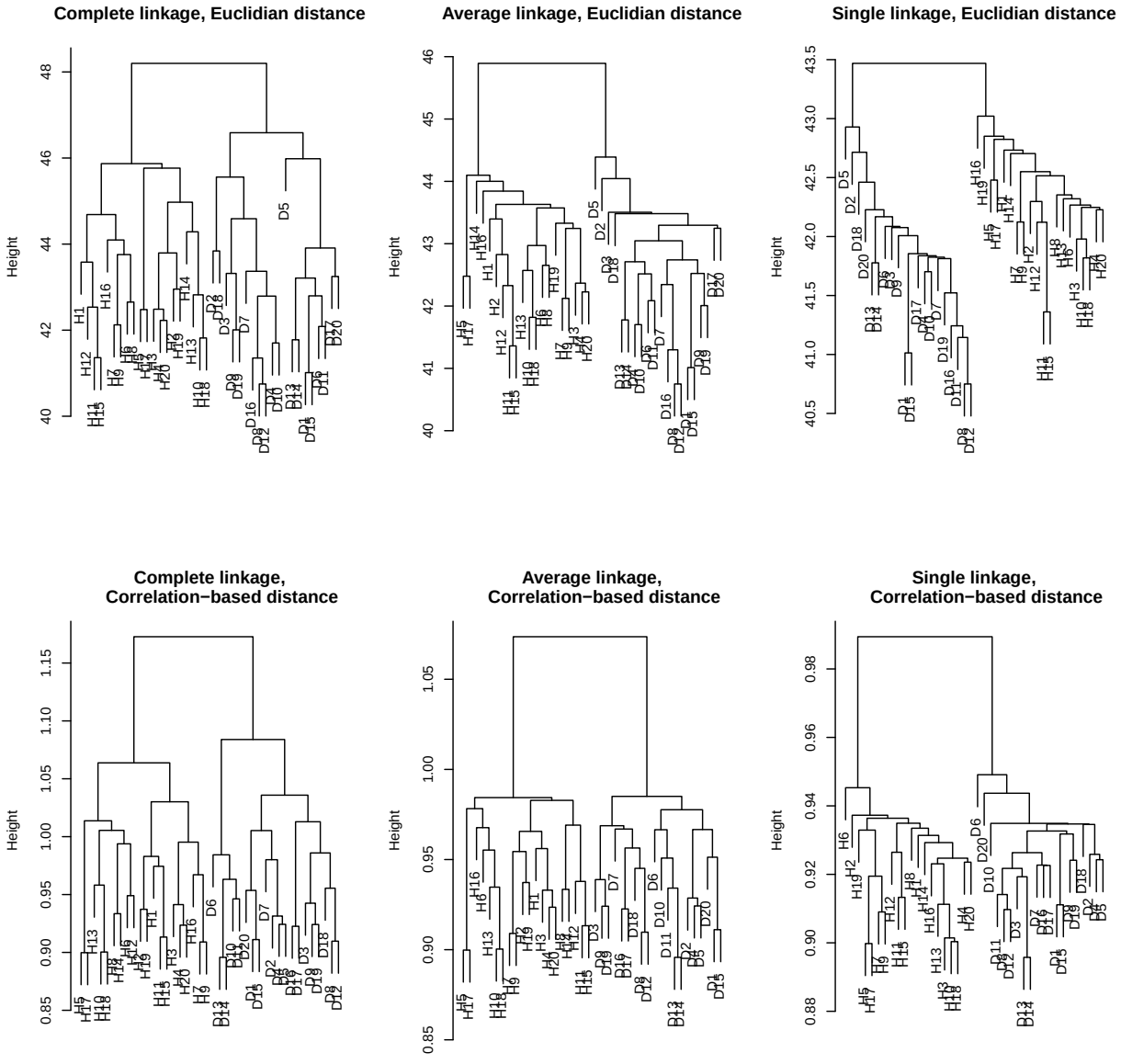
Problem 5

```
id <- "1VfVCQvWt121UN39NXZ4aR9Dmsbj-p9OU"  # google file ID
GeneDatas <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = F)
colnames(GeneDatas)[1:20] = paste(rep("H", 20), c(1:20), sep = "")
colnames(GeneDatas)[21:40] = paste(rep("D", 20), c(1:20), sep = "")
# print(colnames)
row.names(GeneDatas) = paste(rep("G", 1000), c(1:1000), sep = "")
GeneData = t(GeneDatas)
```

a)

```
par(mfrow = c(2, 3))
plot(hclust(dist(scale(GeneData)), method = "complete"), main = "Complete linkage, Euclidian distance",
    xlab = "", sub = "")
plot(hclust(dist(scale(GeneData)), method = "average"), main = "Average linkage, Euclidian distance",
    xlab = "", sub = "")
plot(hclust(dist(scale(GeneData)), method = "single"), main = "Single linkage, Euclidian distance",
    xlab = "", sub = "")
plot(hclust(as.dist(1 - cor(t(scale(GeneData)))), method = "complete"), main = "Complete linkage,
     Correlation-based distance",
```

```
    xlab = "", sub = "")
plot(hclust(as.dist(1 - cor(t(scale(GeneData)))), method = "average"), main = "Average linkage,
     Correlation-based distance",
    xlab = "", sub = "")
plot(hclust(as.dist(1 - cor(t(scale(GeneData)))), method = "single"), main = "Single linkage,
     Correlation-based distance",
    xlab = "", sub = "")
```



b)

As per now, all methods will classify all tissues correctly, as we know that the first ten variables should be value 1, and the last ten should be value 2. Therefore, all metods could be used in this example. However, complete and average methods tend to perform better than single linkage methods.

c)

With Principal Component Analysis, the first principal component loading vector solves the following optimization problem,

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

That is, the PCA analysis finds a low dimension that captures most of the variability of the data. In the above equation, $\phi_{11}, \ldots, \phi_{p1}$ are the loadings of the first principal component, making up the principal component loading vector, $\phi_{11} = (\phi_{11} \ldots \phi_{p1})^T$, consisting of p elements. Because the system is normalized, the squared sums of the loading vector components must become one, that is

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

As the loading vector defines a direction in feature space along which the data vary the most, we can project n number of data points $x_1, \ldots, x_n$ onto this direction, where the projected values will be the principal component scores $z_{11}, \ldots, z_{n1}$ themselves.

The first principal component consists of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features:

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{p1} X_p \tag{13}$$

Thus, the second principal component is the linear combination of $X_1, \ldots, X_p$ that has maximal variance among all linear combinations that are uncorrelated with Z_1. The second PC scores $z12, z22, \ldots, zn2$ take the form $z_{i2} = \phi_{12} x_{i1} + \phi_{22} x_{i2} + \cdots + \phi_{p2} x_{ip}$, where ϕ_2 is the second principal component loading vector. The third principal component is chosen by taking finding the linear combination with maximal variance among all linear combinatinans uncorrelated to Z_2 and thus also Z_2, and so on for the next PCs. It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be orthogonal (perpendicular) to the direction ϕ_1 and so on. The principal component directions $\phi_1, \phi_2 \phi_3, \ldots$ are therefore the ordered sequence of right singular vectors of the matrix X, and the variances of the components are $\frac{1}{n}$ times the squares of the singular values. There are at most $\min(n-1, p)$ principal components.

d)

i)

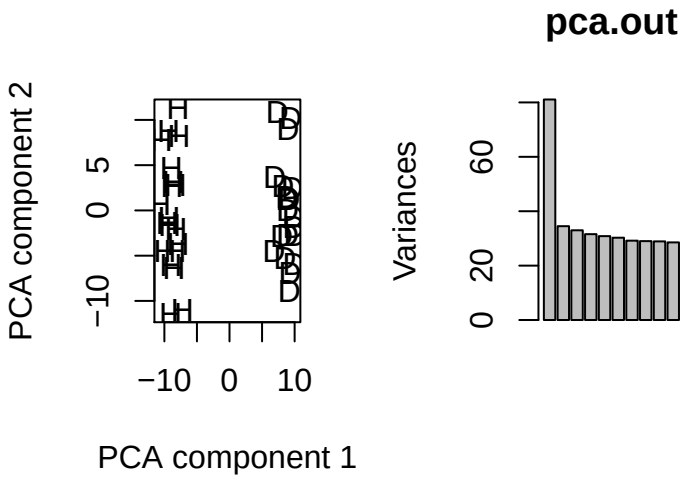
```
pca.out = prcomp(GeneData, scale = TRUE)
names(pca.out)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

```r
par(mfrow = c(1, 2))
plot(pca.out$x[, 1:2], xlab = "PCA component 1", ylab = "PCA component 2", pch = rownames(GeneData))
# pch = c(GeneData[1:20], GeneData[21:40]))
summary(pca.out)
```

```
## Importance of components:
##                             PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.00460 5.87302 5.74347 5.61806 5.55344 5.50107 5.40069
## Proportion of Variance 0.08108 0.03449 0.03299 0.03156 0.03084 0.03026 0.02917
## Cumulative Proportion  0.08108 0.11558 0.14856 0.18013 0.21097 0.24123 0.27040
##                             PC8    PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     5.38575 5.3762 5.34146 5.31878 5.25016 5.18737 5.1667
## Proportion of Variance 0.02901 0.0289 0.02853 0.02829 0.02756 0.02691 0.0267
## Cumulative Proportion  0.29940 0.3283 0.35684 0.38513 0.41269 0.43960 0.4663
##                            PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     5.10384 5.04667 5.03288 4.98926 4.92635 4.90996 4.88803
## Proportion of Variance 0.02605 0.02547 0.02533 0.02489 0.02427 0.02411 0.02389
## Cumulative Proportion  0.49234 0.51781 0.54314 0.56803 0.59230 0.61641 0.64030
##                            PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     4.85159 4.79974 4.78202 4.70171 4.66105 4.64595 4.59194
## Proportion of Variance 0.02354 0.02304 0.02287 0.02211 0.02173 0.02158 0.02109
## Cumulative Proportion  0.66384 0.68688 0.70975 0.73185 0.75358 0.77516 0.79625
##                            PC29    PC30   PC31    PC32    PC33   PC34    PC35
## Standard deviation     4.53246 4.47381 4.4389 4.41670 4.39404 4.3591 4.23504
## Proportion of Variance 0.02054 0.02001 0.0197 0.01951 0.01931 0.0190 0.01794
## Cumulative Proportion  0.81679 0.83681 0.8565 0.87602 0.89533 0.9143 0.93226
##                           PC36    PC37   PC38    PC39     PC40
## Standard deviation     4.2184 4.12936 4.0738 4.03658 4.64e-15
## Proportion of Variance 0.0178 0.01705 0.0166 0.01629 0.00e+00
## Cumulative Proportion  0.9501 0.96711 0.9837 1.00000 1.00e+00
```

```r
plot(pca.out)
```



```
# Didnt use color, but H for healthy and D for diseased.
```

ii)

```
pca.var = pca.out$sdev^2
pve = pca.var/sum(pca.var)  #To get the variance
pve
```

```
##  [1] 8.108273e-02 3.449241e-02 3.298741e-02 3.156262e-02 3.084068e-02
##  [6] 3.026180e-02 2.916747e-02 2.900628e-02 2.890346e-02 2.853115e-02
## [11] 2.828944e-02 2.756422e-02 2.690880e-02 2.669509e-02 2.604914e-02
## [16] 2.546892e-02 2.532988e-02 2.489274e-02 2.426893e-02 2.410770e-02
## [21] 2.389282e-02 2.353792e-02 2.303748e-02 2.286775e-02 2.210608e-02
## [26] 2.172536e-02 2.158489e-02 2.108590e-02 2.054315e-02 2.001497e-02
## [31] 1.970378e-02 1.950727e-02 1.930759e-02 1.900201e-02 1.793553e-02
## [36] 1.779528e-02 1.705163e-02 1.659573e-02 1.629395e-02 2.152759e-32
```

```
pve.perc = pve * 100
```

By summing the variation reduction of the first five components we get that the first five PC´s reduce the variance by cumsum(pve.perc)[5] %.

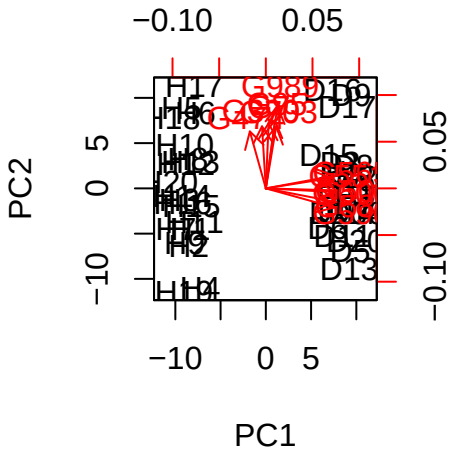
e)

Use your results from PCA to find which genes that vary the most accross the two groups.

```
pca.loading = pca.out$rotation[, 1:2]
informative_loadings = rbind(head(pca.loading[order(pca.loading[, 1], decreasing = TRUE),
    ]), head(pca.loading[order(pca.loading[, 2], decreasing = TRUE), ]))
informative_loadings
```

```
##               PC1          PC2
## G502  0.094850438 -0.002729035
## G589  0.094497659 -0.018373912
## G565  0.091838234  0.016964824
## G590  0.091731695 -0.025556766
## G600  0.091673220 -0.004948422
## G551  0.087683596  0.013883967
## G989  0.015472152  0.110029421
## G95   0.012194885  0.090990473
## G7    0.003764118  0.087251726
## G399 -0.005157145  0.083070538
## G703  0.013810197  0.082616817
## G474 -0.021381601  0.076298114
```

```
biplot(x = pca.out$x[, 1:2], y = informative_loadings, scale = 0)
```

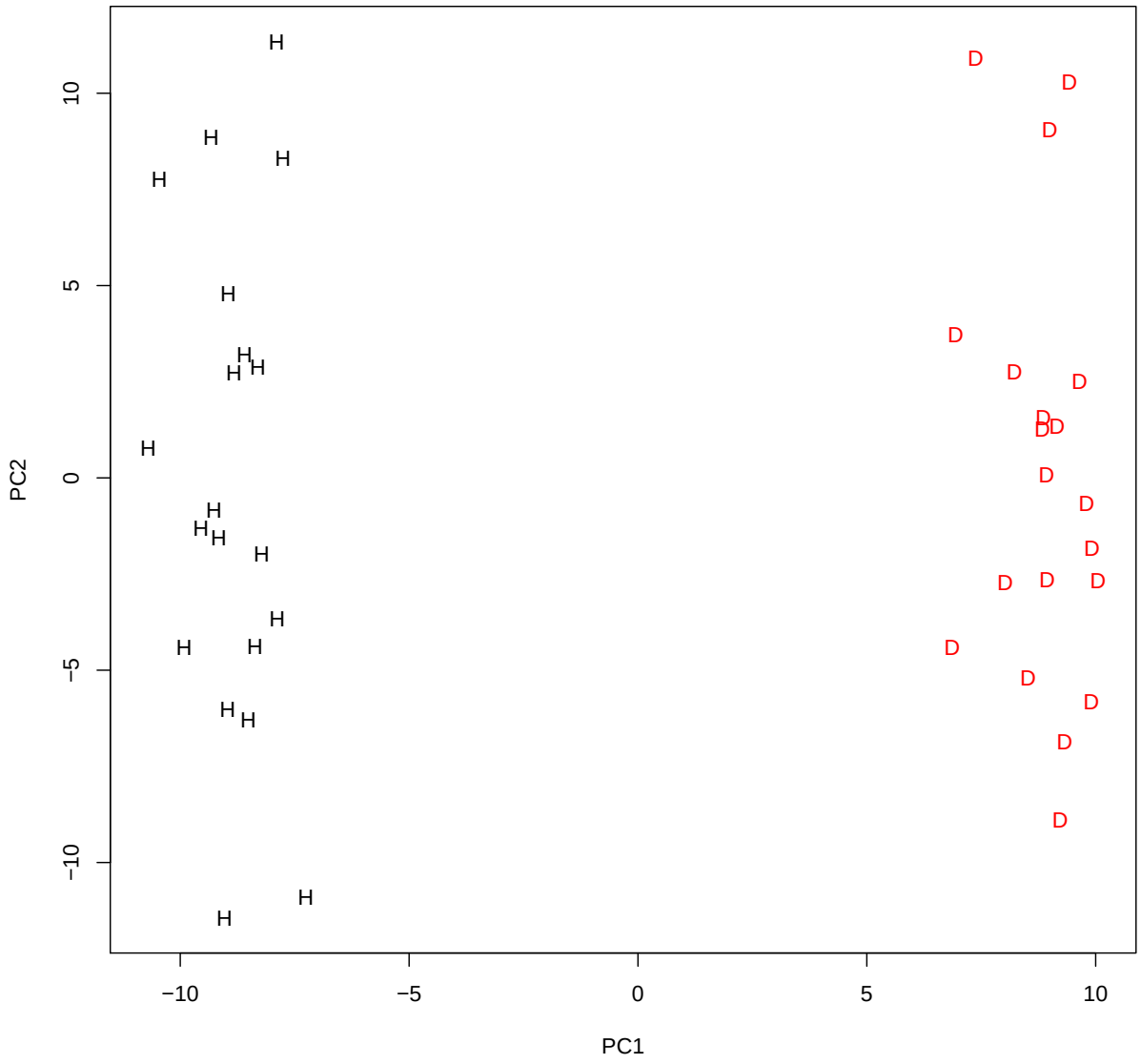


Wee see that the 10 genes that vary the most across the two groups in decreasing direction are G502, G589, G565, G590, G600, G551, G989, G95 and G7.

f)

Use K-means to seperate the tissue samples into two groups. Plot the values in a two-dimensional space with PCA. What is the error rate of K-means?

```
km.out = kmeans(GeneData, 2, nstart = 20)  #Separate GeneData into 2 groups. Could also do nstart=1
# PCA with true labels, combined with cluster
plot(pca.out$x[, 1:2], col = (km.out$cluster), pca.out = c(GeneData[1:20], GeneData[21:40]),
    pch = rownames(GeneData))
```




```
# Error rate, in this case zero.
(km.out$cluster)
```

```
##  H1  H2  H3  H4  H5  H6  H7  H8  H9 H10 H11 H12 H13 H14 H15 H16 H17 H18 H19 H20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  D1  D2  D3  D4  D5  D6  D7  D8  D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
```

The error rate of k-means in this case is zero.