# Compulsory exercise 2: Group 13

TMA4268 Statistical Learning V2020

Vemund Tjessem, Erik Andre Klepp Vik

29 mars, 2020

```r
# 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 $\beta_{Ridge}$ 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

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

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. This is done assuming 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.

## b)

The expectation value of $y = X\beta + \epsilon$ is $\mathrm{E}[y] = X\beta$, as $\mathrm{E}[\epsilon] = 0$. The expectation value of $\beta_{Ridge}$ 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 $\mathrm{Var}[y] = \mathrm{Var}[X\beta] + \mathrm{Var}[\epsilon] = \sigma^2$.

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

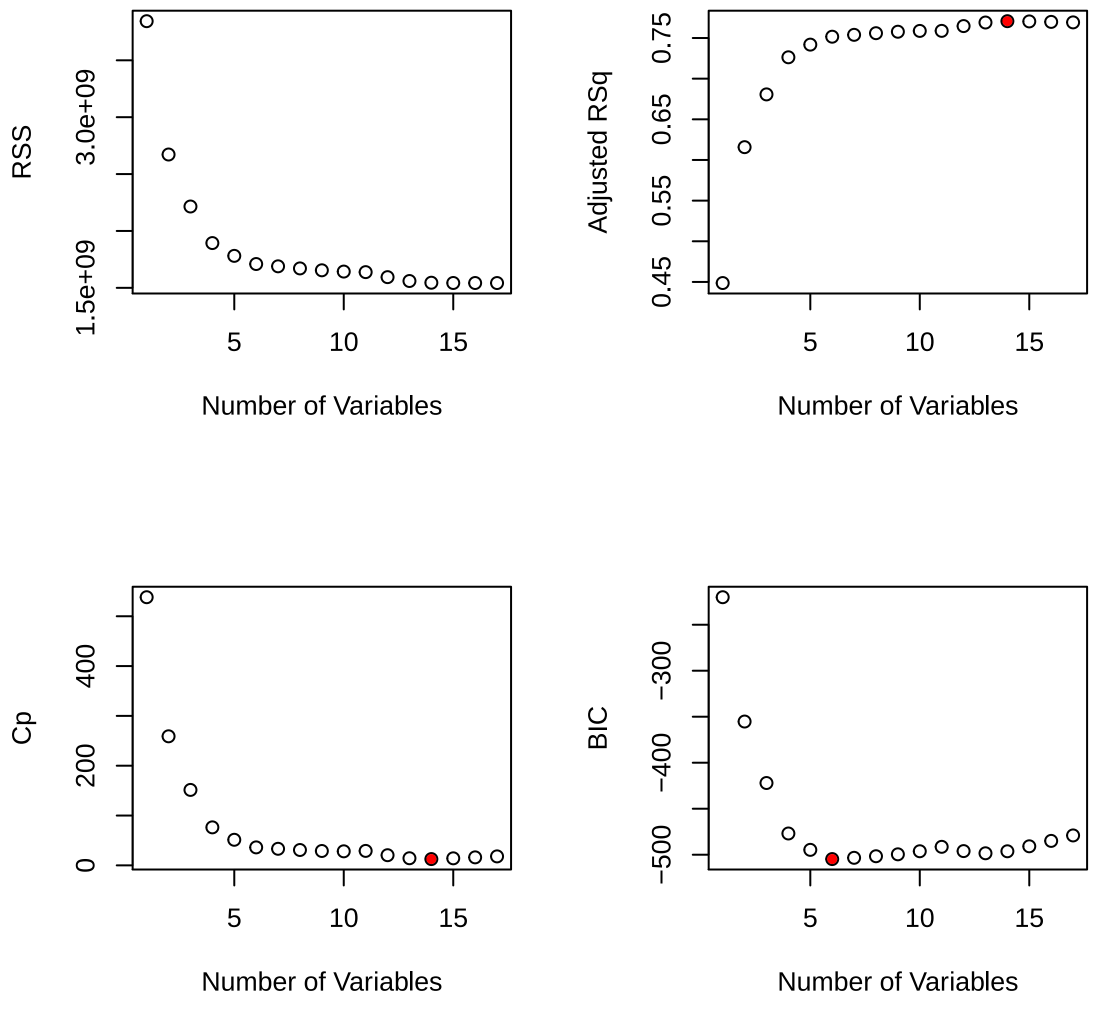
## c)

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

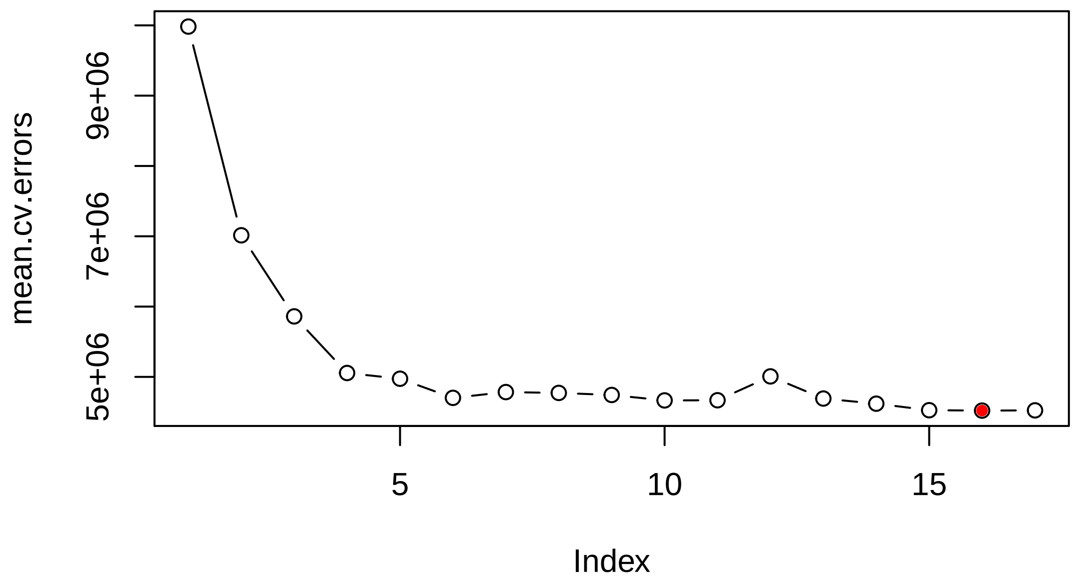
## d)

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

```r
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)
```



```r
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)
# mean.cv.errors 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)
```



```r
# Calculating the MSE for model with 6 predictors
x.test = model.matrix(Outstate ~ ., data = college.test)
coef6 = coef(fwd.fit, id = 6)
co.names = names(coef6)[-1]
co.names[1] = "Private"
```

```
pred = x.test[, names(coef6)] %*% coef6
MSE.forward = mean((college.test$Outstate - pred)^2)
```

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 model with 6 predictors has a MSE of $3.8448572 \times 10^6$. The 6 predictors are Private, Room.Board, Terminal, perc.alumni, Expend, Grad.Rate.

### 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 $\lambda$. 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 $\lambda$ 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 $3.9033653 \times 10^6$.

## Problem 2

### a)

(i) False
(ii) False
(iii) True
(iv) True (fra video om smoothing splines)

## b)

The basis functions are

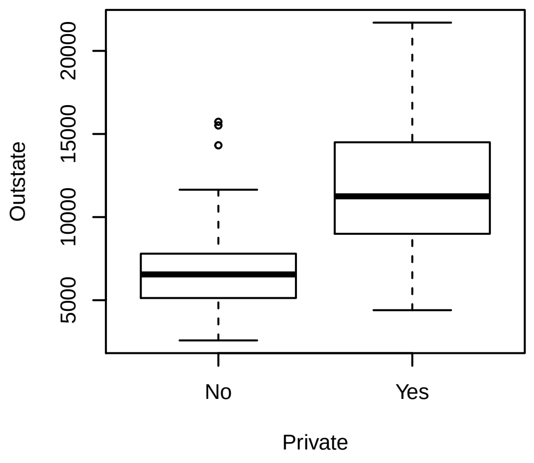
$$\begin{aligned}
b_1(x) &= x^1 && \text{(7a)}\\
b_2(x) &= x^2 && \text{(7b)}\\
b_3(x) &= x^3 && \text{(7c)}\\
b_4(x) &= (x - q_1)^3_+ && \text{(7d)}\\
b_5(x) &= (x - q_2)^3_+ && \text{(7e)}\\
b_6(x) &= (x - q_3)^3_+ && \text{(7f)}
\end{aligned}$$

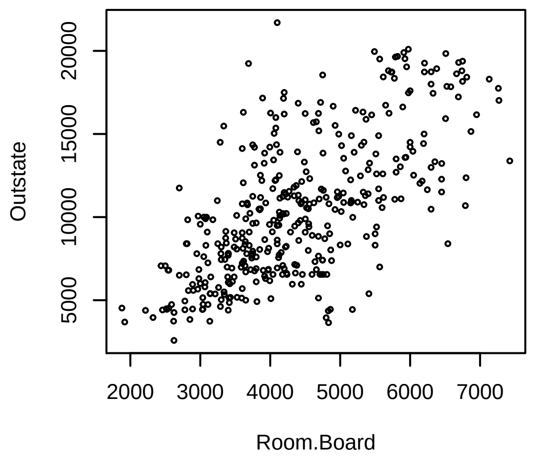
## c)

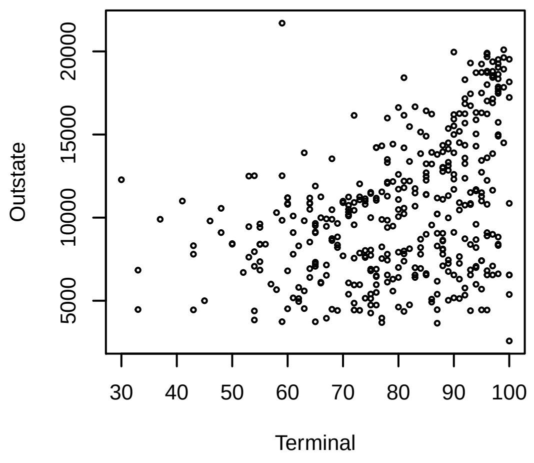
```r
for.reg = regsubsets(Outstate ~ ., data = college.train, method = "forward")
coef.for = coef(for.reg, id = 6)
co.names = names(coef.for)[-1]
co.names[1] = "Private"
```

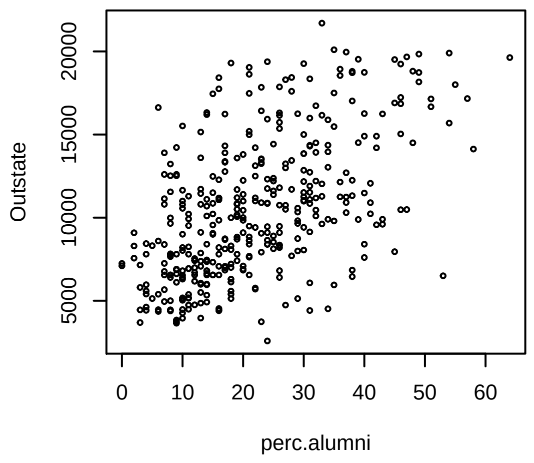
Will investigate the relationship between `Outstate` and the following 6 predictors: Private, Room.Board, Terminal, perc.alumni, Expend, Grad.Rate.

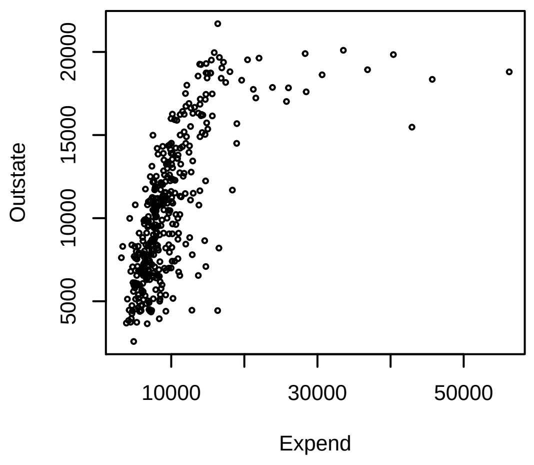
```r
par(mfrow = c(3, 2))
plot(Outstate ~ Private, data = college.train)
plot(Outstate ~ Room.Board, data = college.train, cex = 0.5)
plot(Outstate ~ Terminal, data = college.train, cex = 0.5)
plot(Outstate ~ perc.alumni, data = college.train, cex = 0.5)
plot(Outstate ~ Expend, data = college.train,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

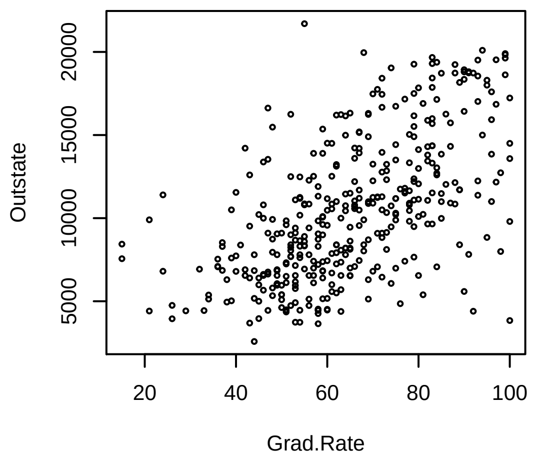

Outstate
5000
10000
15000
20000
No
Yes
Private


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

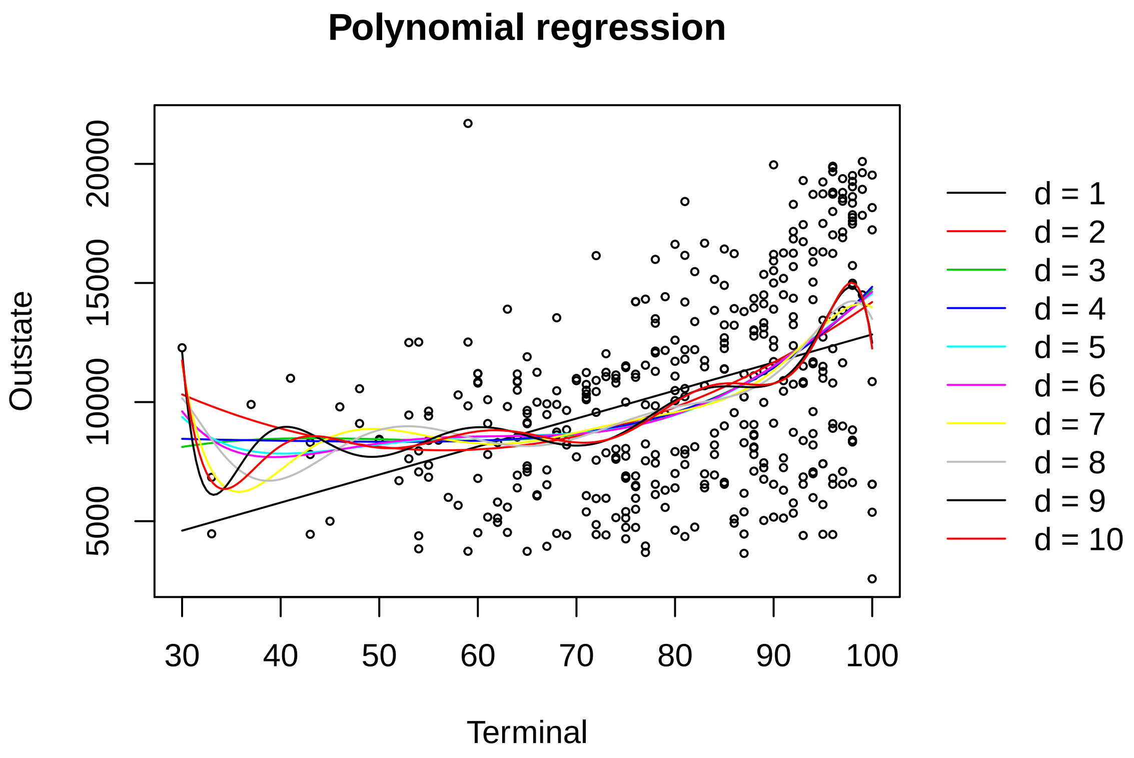

Outstate
5000
10000
15000
20000
20
40
60
80
100
Grad.Rate

The relationship between `Outstate` and `Room.Board` seems to be approximately linear, same for `perc.alumni`. 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, however the relation between `Outstate` and `Grad.Rate` does.

### d)

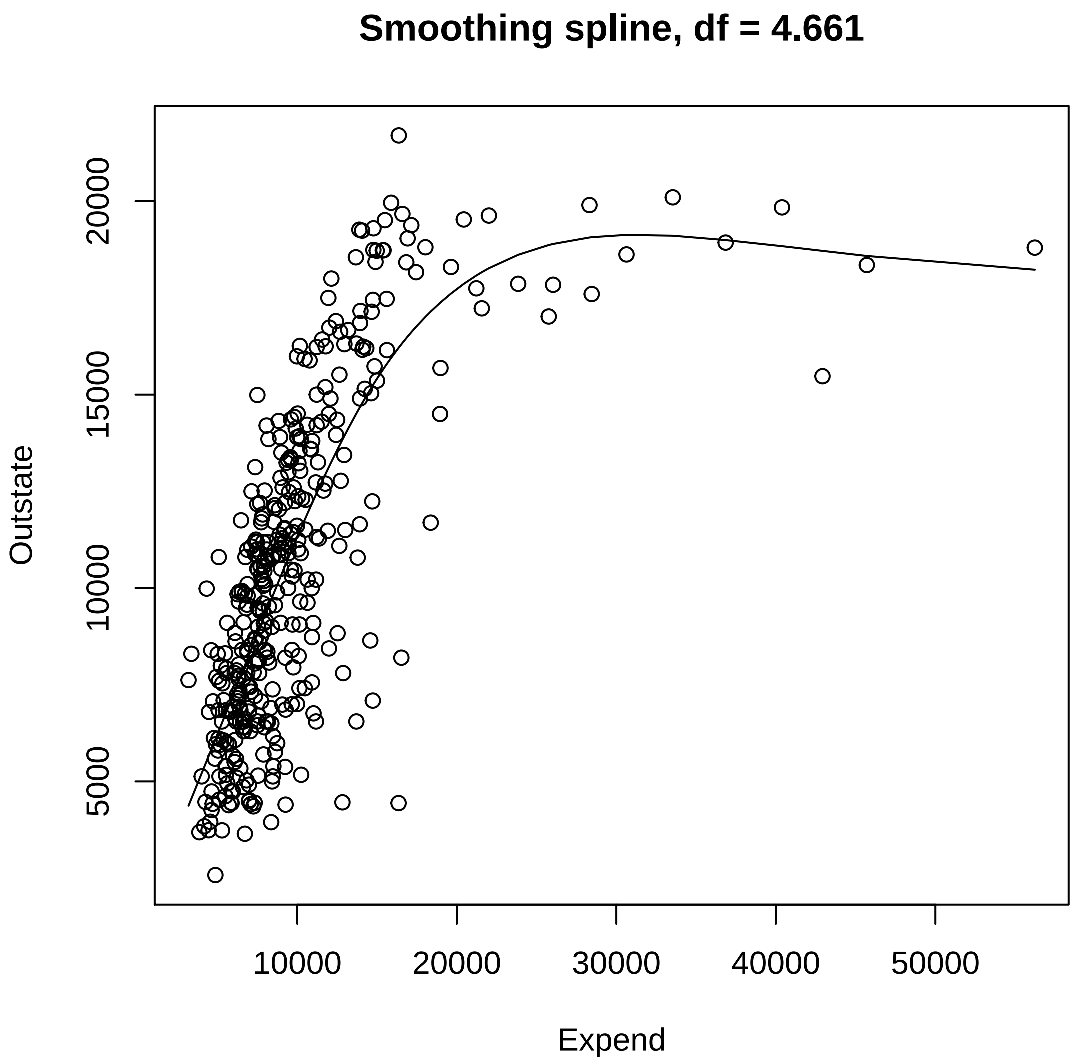
(i) Fit polynomial regression models for `Outstate` as a function of `Terminal` with polynomial degrees $d = 1, \dots, 10$.

```r
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`.

```r
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)
```

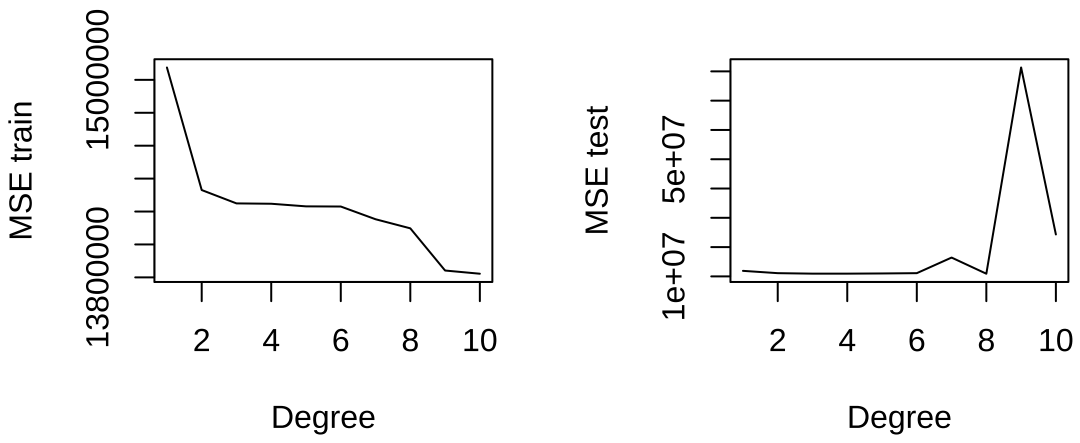


```r
# points(college.test$Expend, college.test$Outstate, pch=2, col=2)
MSE.smthspl.train = mean((predict(smthspl.fit, x)$y - y)^2)
MSE.smthspl.test = mean((predict(smthspl.fit, college.test$Expend)$y - college.test$Outstate)^2)
```

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

```r
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")
```



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

The smallest training error of the polynomials was $1.3822205 \times 10^7$, which corresponds to the polynomial of degree 10. As can be seen from the plot to the ledt 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. This is known as overfitting.

The training MSE for the smoothing spline is $6.8712814 \times 10^6$. It was expected that the smoothing spline would have a lower training MSE as it uses `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

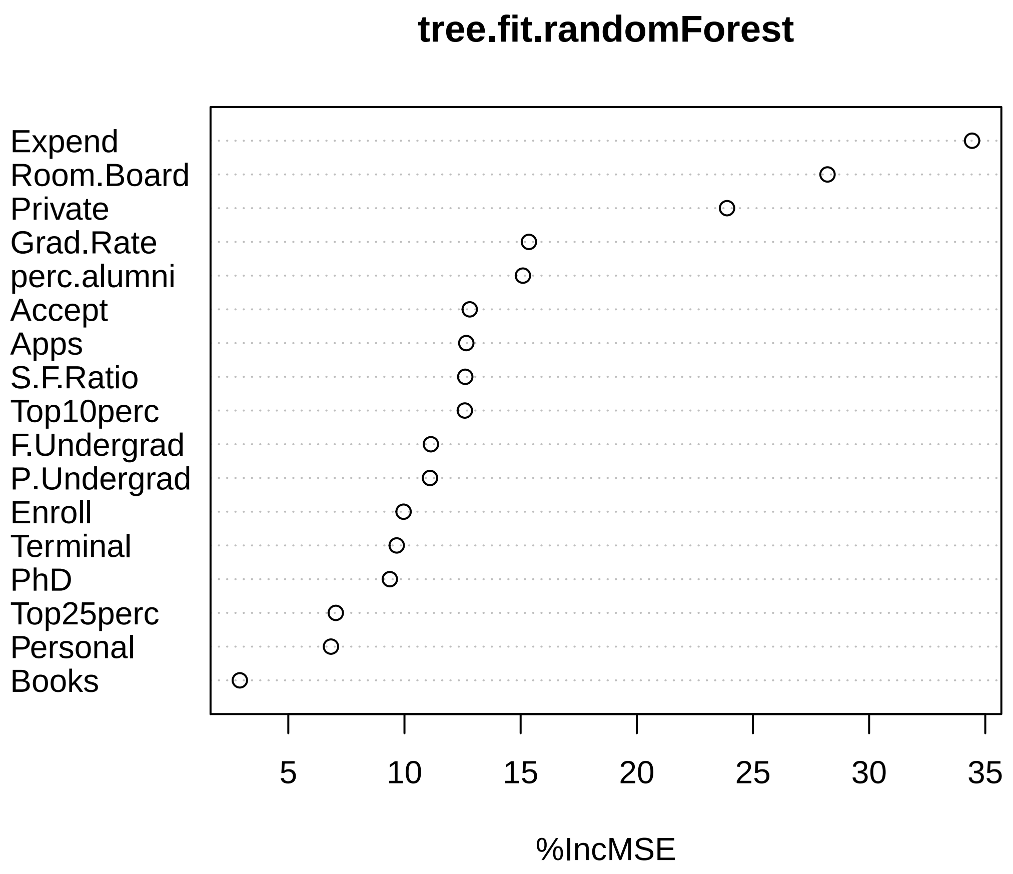
### b)

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.

```r
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)
```

```
##                %IncMSE IncNodePurity
## Private     23.884645     572396197
## Apps        12.660213     119525827
## Accept      12.809286     113271754
## Enroll       9.960338     105290813
## Top10perc   12.597062     367734616
## Top25perc    7.042636     161501521
## F.Undergrad 11.136443     122819717
## P.Undergrad 11.097160     164097235
## Room.Board  28.209705    1012135468
## Books        2.912269      71226082
## Personal     6.834351     119306116
## PhD          9.369483     225362542
## Terminal     9.665515     196862503
## S.F.Ratio   12.615659     390623584
## perc.alumni 15.097476     337449297
## Expend      34.432054    2218648378
## Grad.Rate   15.356554     545391655
```

```r
varImpPlot(tree.fit.randomForest, type = 1)
```



## c)

Compare MSEs of the different methods.

```
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 prediction error is the model from forward selection. If the goal was to develop an interpretable model the best may be a pruned tree, assuming it it not to bushy.

## Problem 4

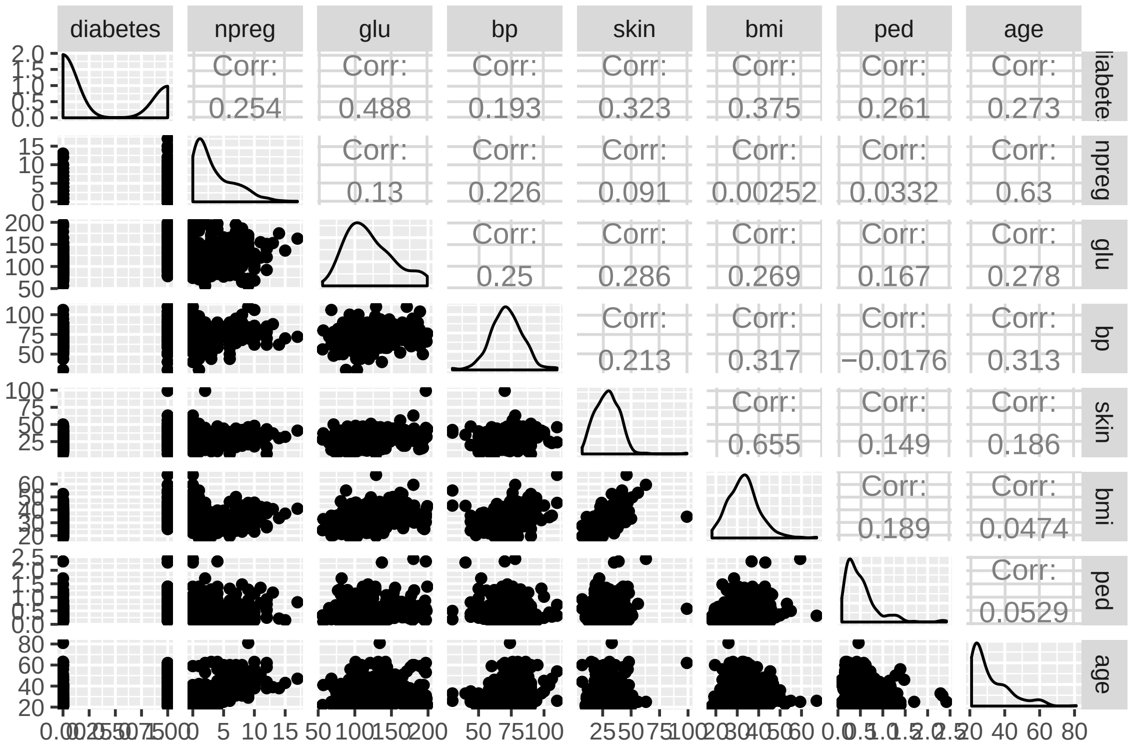
```
id <- "1Fv6xwKLSZHldRAC1MrcK2mzdOYnbgvOE"  # google file ID
d.diabetes <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
d.train = d.diabetes$ctrain
d.test = d.diabetes$ctest
d.train$diabetes <- as.factor(d.train$diabetes)
d.test$diabetes <- as.factor(d.test$diabetes)
```

### a)

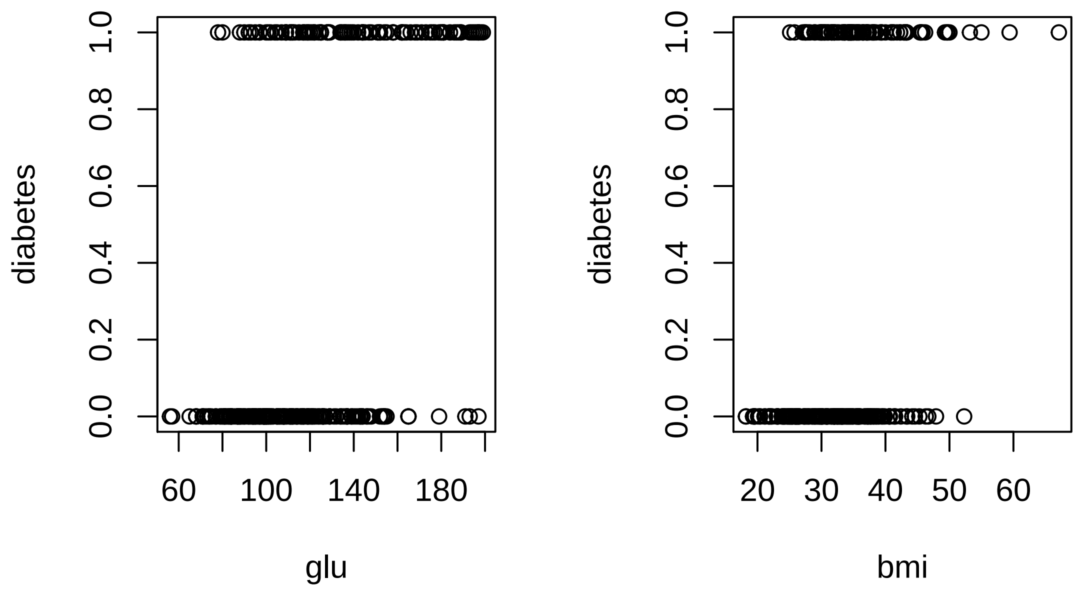
```
summary(d.train)
```

```
##     diabetes          npreg             glu               bp        
##  Min.   :0.0000   Min.   : 0.000   Min.   : 56.00   Min.   : 30.00  
##  1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.: 96.75   1st Qu.: 64.00  
##  Median :0.0000   Median : 2.000   Median :114.00   Median : 71.00  
##  Mean   :0.3333   Mean   : 3.467   Mean   :120.13   Mean   : 71.56  
##  3rd Qu.:1.0000   3rd Qu.: 5.250   3rd Qu.:140.25   3rd Qu.: 80.00  
##  Max.   :1.0000   Max.   :17.000   Max.   :199.00   Max.   :110.00  
##       skin            bmi             ped              age       
##  Min.   : 7.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00  
##  1st Qu.:22.00   1st Qu.:27.98   1st Qu.:0.2567   1st Qu.:23.00  
##  Median :29.00   Median :32.80   Median :0.4150   Median :27.00  
##  Mean   :29.14   Mean   :33.03   Mean   :0.5004   Mean   :31.55  
##  3rd Qu.:36.00   3rd Qu.:37.12   3rd Qu.:0.6210   3rd Qu.:37.25  
##  Max.   :99.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

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



```r
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)

```
# svmfit_linear = svm(diabetes~., data = d.train, kernel = 'linear', cost = 1,
# scale=FALSE)
dim(d.train)
```

```
## [1] 300   8
```

```
# summary(svmfit_linear)
```

Support vector calssifier with a linear boundary.

```
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)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.2033333
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.3333333 0.14572086
## 2 1e-02 0.2233333 0.10546780
## 3 1e-01 0.2033333 0.06749486
## 4 1e+00 0.2200000 0.07235031
## 5 5e+00 0.2200000 0.06703601
## 6 1e+01 0.2200000 0.06703601
## 7 5e+01 0.2200000 0.06703601
```

```
best_model = CV_linear$best.model
summary(best_model)
```

```
##
## Call:
## best.tune(method = svm, train.x = diabetes ~ ., data = d.train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 50)), kernel = "linear")
##
```

```
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  150
##
##  ( 75 75 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
# best_model = svm(diabetes~., data = d.train, kernel = 'linear', cost = 0.1,
# scale=FALSE)
y_pred = predict(best_model, d.test)
table(predict = y_pred, truth = d.test[, 1])
```

```
##        truth
## predict   0   1
##       0 137  35
##       1  18  42
```

Support vector classifier with radial boundary.

```
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)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1  0.01
##
## - best performance: 0.2066667
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-03 1e-02 0.3333333 0.05211573
## 2  1e-02 1e-02 0.3333333 0.05211573
## 3  1e-01 1e-02 0.3333333 0.05211573
## 4  1e+00 1e-02 0.2066667 0.04388537
## 5  5e+00 1e-02 0.2166667 0.04513355
## 6  1e+01 1e-02 0.2100000 0.05454639
## 7  5e+01 1e-02 0.2266667 0.07166451
## 8  1e-03 1e-01 0.3333333 0.05211573
## 9  1e-02 1e-01 0.3333333 0.05211573
## 10 1e-01 1e-01 0.2700000 0.08951171
## 11 1e+00 1e-01 0.2200000 0.05921294
## 12 5e+00 1e-01 0.2700000 0.07609286
## 13 1e+01 1e-01 0.2866667 0.07062333
## 14 5e+01 1e-01 0.2933333 0.07336700
## 15 1e-03 1e+00 0.3333333 0.05211573
## 16 1e-02 1e+00 0.3333333 0.05211573
## 17 1e-01 1e+00 0.3333333 0.05211573
## 18 1e+00 1e+00 0.3066667 0.06813204
## 19 5e+00 1e+00 0.3266667 0.06245986
## 20 1e+01 1e+00 0.3300000 0.05762801
## 21 5e+01 1e+00 0.3300000 0.05762801
## 22 1e-03 1e+01 0.3333333 0.05211573
## 23 1e-02 1e+01 0.3333333 0.05211573
## 24 1e-01 1e+01 0.3333333 0.05211573
## 25 1e+00 1e+01 0.3333333 0.05211573
## 26 5e+00 1e+01 0.3333333 0.05211573
## 27 1e+01 1e+01 0.3333333 0.05211573
## 28 5e+01 1e+01 0.3333333 0.05211573
## 29 1e-03 1e+02 0.3333333 0.05211573
## 30 1e-02 1e+02 0.3333333 0.05211573
## 31 1e-01 1e+02 0.3333333 0.05211573
## 32 1e+00 1e+02 0.3333333 0.05211573
## 33 5e+00 1e+02 0.3333333 0.05211573
## 34 1e+01 1e+02 0.3333333 0.05211573
## 35 5e+01 1e+02 0.3333333 0.05211573
```

```
best_model = CV_radial$best.model
summary(best_model)
```

```
##
## Call:
## best.tune(method = svm, train.x = diabetes ~ ., data = d.train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 50), gamma = c(0.01, 0.1, 1, 10, 100)),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  176
##
##  ( 88 88 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
# best_model = svm(diabetes~., data = d.train, kernel = 'linear', cost = 0.1,
# scale=FALSE)
y_pred = predict(best_model, d.test)
table(predict = y_pred, truth = d.test[, 1])
```

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

## Problem 5

### a)