Assignment 4

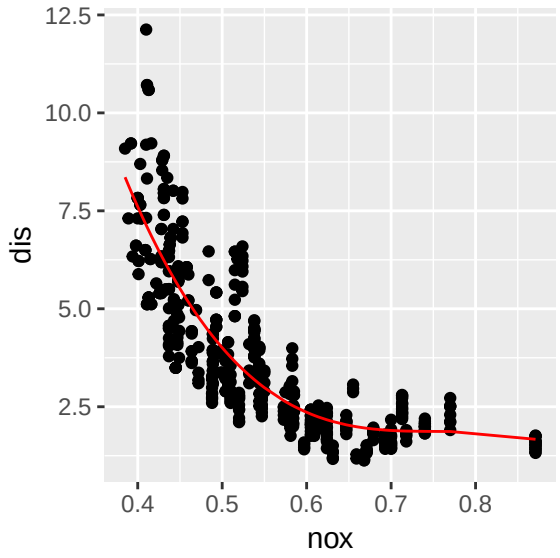
Name Student no.

Mark Qu 5.

Q1

a)

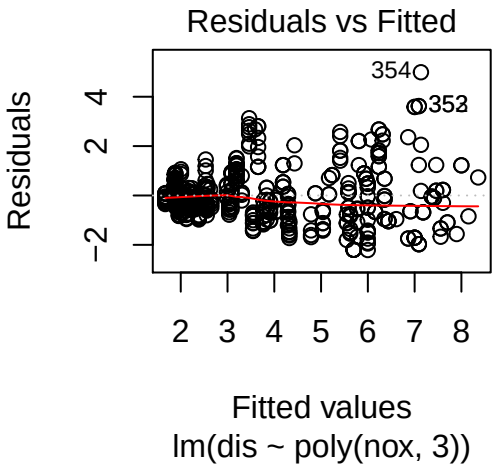
```
library(ggplot2)
library(MASS)
f1 <- lm(dis ~ poly(nox,3), data=Boston )
ggplot(data=Boston, aes(x=nox, y=dis))+ geom_point() +
  geom_line(aes(y=fitted(f1)), color="red")
```




```
mean(residuals(f1)^2)
```

```
## [1] 1.094805
```

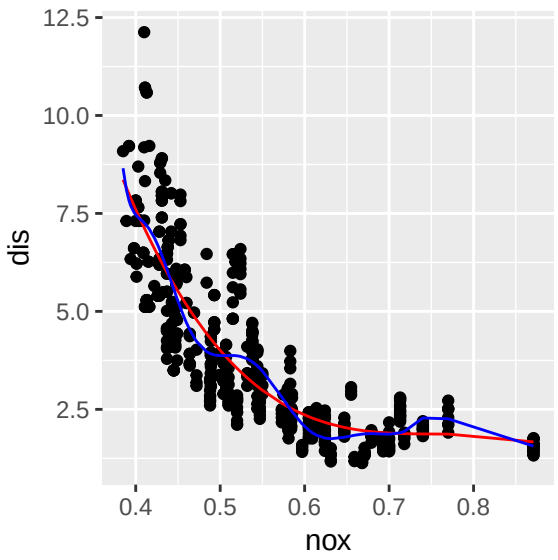
```
plot(f1,1)
```



A reasonable fit. Residual plots show increasing variance. Very mild curvature.

b)

```
f2 <- lm(dis ~ poly(nox,10), data=Boston )
ggplot(data=Boston, aes(x=nox, y=dis))+ geom_point() +
  geom_line(aes(y=fitted(f1)), color="red")+
  geom_line(aes(y=fitted(f2)), color="blue")
```




```
mean(residuals(f2)^2)
```

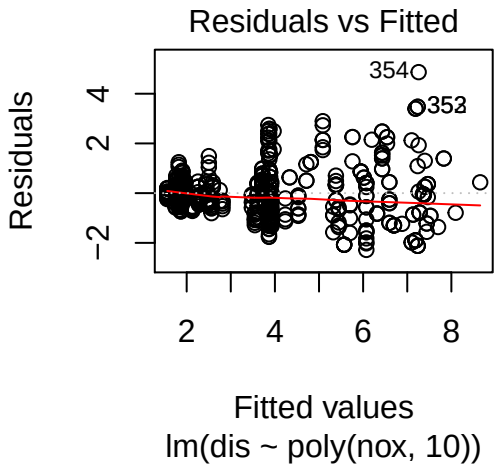
```
## [1] 1.011485
```

```
anova(f1,f2)
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ poly(nox, 3)
## Model 2: dis ~ poly(nox, 10)
##   Res.Df    RSS Df Sum of Sq     F    Pr(>F)
```

```
## 1    502 553.97
## 2    495 511.81  7     42.16 5.825 1.655e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(f2, 1)
```



Some of the terms in f2 are significant (not all = 0). From the graph it overfits the data, but picks up the increase in dis with increasing nox past 0.65.

c)

Split the data into 5 groups of approximately equal size. for each degree (j) between 1 and 10, for each hold out sample, fit the model on the rest and calculate the average test error on the hold out sample, this gives mse1... mse5. the cv error for degree j is the (weighted) average of the mse1.... mse5. Pick the degree with the smallest cv error.

d)

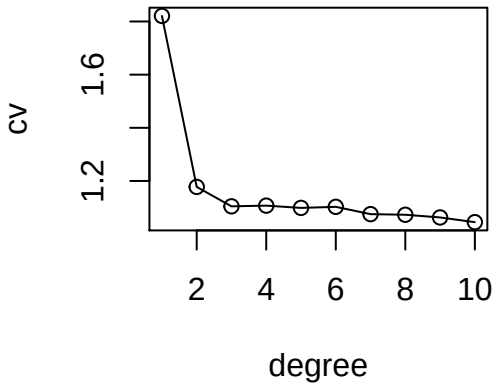
```
set.seed(123)


k <- 5
fold <- sample(k, nrow(Boston), replace=T)
fsize <- table(fold)

mse <- vector(length=k)
degree <- 1:10
cv <- vector(length=length(degree))

for (j in 1:length(degree)){
  for (i in 1:k){
    foldi <- Boston[fold==i,]
    foldOther <- Boston[fold!=i,]
    f <- lm(dis ~ poly(nox,degree[j]), data=foldOther )
    pred <- predict(f, foldi)
```

```
    mse[i] <-mean((pred - foldi$dis)^2) # MSEi
  }
  cv[j]<- weighted.mean(mse, fsize)
}
plot(degree, cv)
lines(degree,cv)
```




```
degree[which.min(cv)] # produces the lowest CV
```

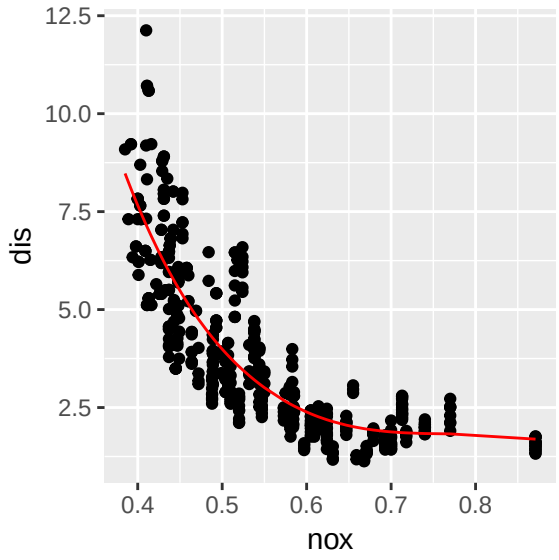
```
## [1] 10
```

e)

```
library(splines)
f3 <- lm(dis ~ bs(nox, df=4), data=Boston)
attr(bs(Boston$nox, df=4), "knots")
```

```
##   50%
## 0.538
```

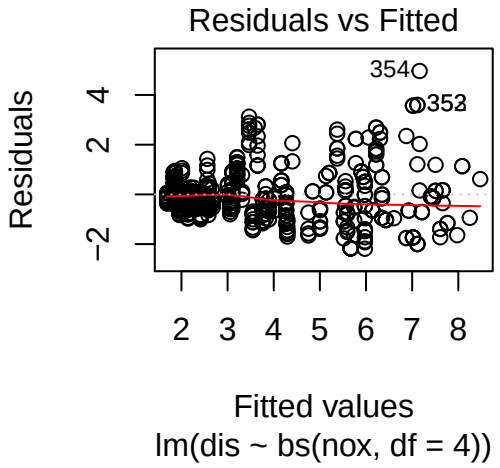
```
ggplot(data=Boston, aes(x=nox, y=dis))+ geom_point() +
  geom_line(aes(y=fitted(f3)), color="red")
```



```r
mean(residuals(f3)^2)
```

```
## [1] 1.094149
```

```r
plot(f3, 1)
```



f)

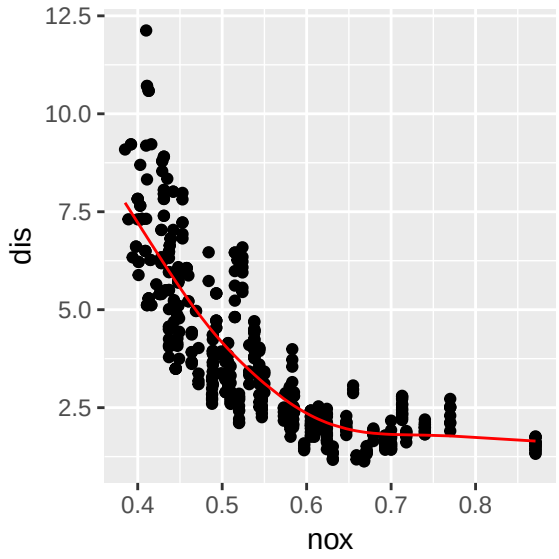
```r
f4 <- smooth.spline(Boston$nox,Boston$dis, cv= T)
```

```
## Warning in smooth.spline(Boston$nox, Boston$dis, cv = T): cross-validation
## with non-unique 'x' values seems doubtful
```

```r
f4
```

```
## Call:
## smooth.spline(x = Boston$nox, y = Boston$dis, cv = T)
##
## Smoothing Parameter  spar= 1.050661  lambda= 0.07031691 (16 iterations)
## Equivalent Degrees of Freedom (Df): 4.128915
## Penalized Criterion (RSS): 407.277
## PRESS(l.o.o. CV): 1.119355
```

```r
ggplot(data=Boston, aes(x=nox, y=dis))+ geom_point() +
  geom_line(aes(y=fitted(f4)), color="red")
```




```
mean(residuals(f4)^2)
```

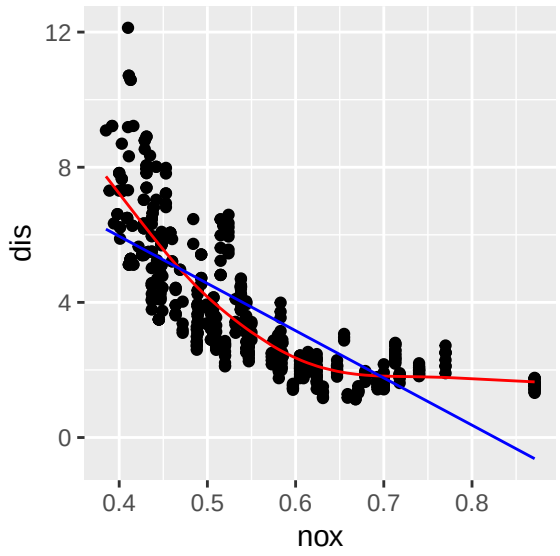
```
## [1] 1.104526
```

g)

```
f5 <- smooth.spline(Boston$nox,Boston$dis, spar=2)
f5
```

```
## Call:
## smooth.spline(x = Boston$nox, y = Boston$dis, spar = 2)
##
## Smoothing Parameter  spar= 2  lambda= 508171.7
## Equivalent Degrees of Freedom (Df): 1.999947
## Penalized Criterion (RSS): 762.6087
## GCV: 1.82113
```

```
ggplot(data=Boston, aes(x=nox, y=dis))+ geom_point() +
  geom_line(aes(y=fitted(f4)), color="red")+
  geom_line(aes(y=fitted(f5)), color="blue")
```



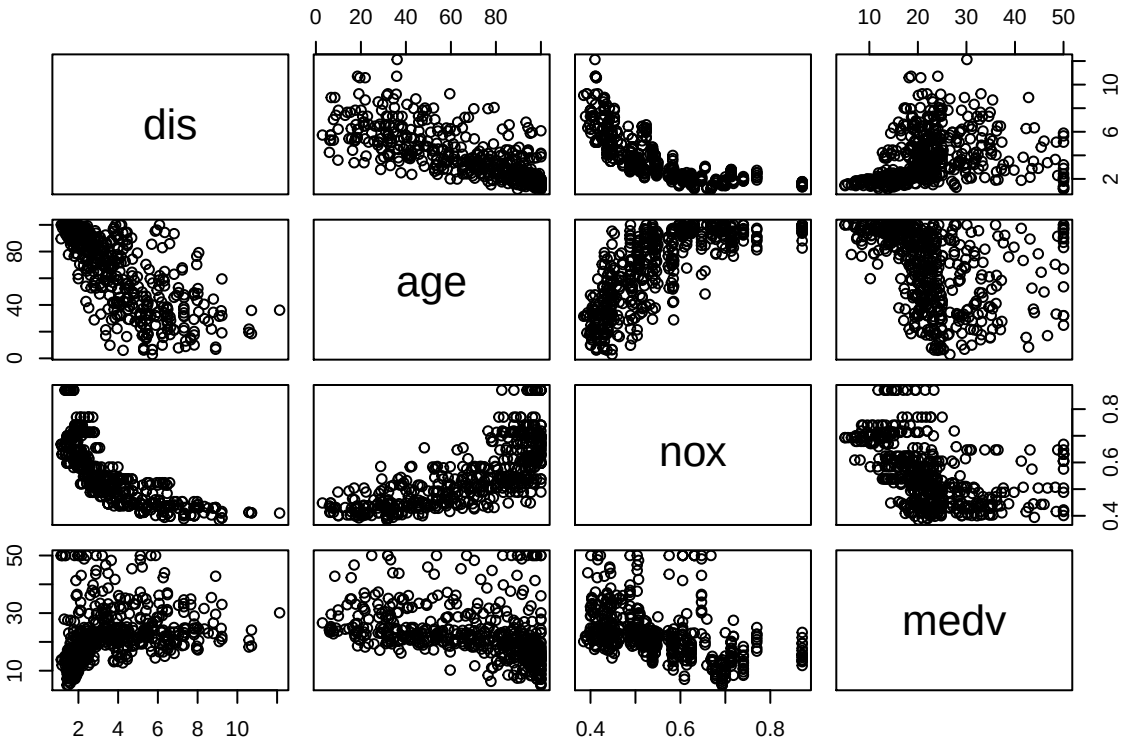
This fit is too smooth.

Q2

a)

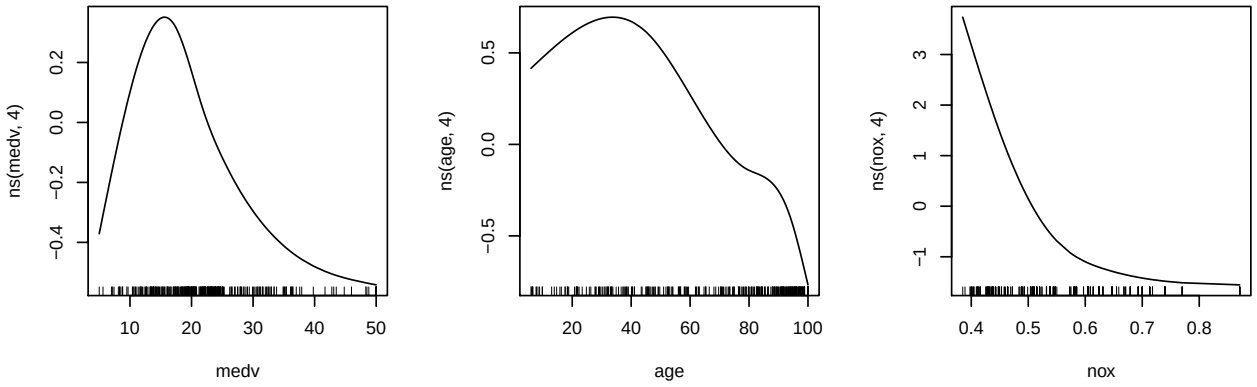
```
set.seed(123)
s <- sample(nrow(Boston), round(.6*nrow(Boston)))
Boston1 <- Boston[s,]
Boston2 <- Boston[-s,]
gfit1 <-  lm(dis ~ ns(medv,4)+ ns(age,4)+ ns(nox,4), data=Boston1)

pairs(Boston[, c(8, 7, 5, 14)])
```



b)

```
suppressMessages(library(gam))
par(mfrow=c(1,3))
plot.Gam(gfit1)
```



Linear model does not seem appropriate.

c)

```
attr(ns(Boston1$age,4), "knots")
```

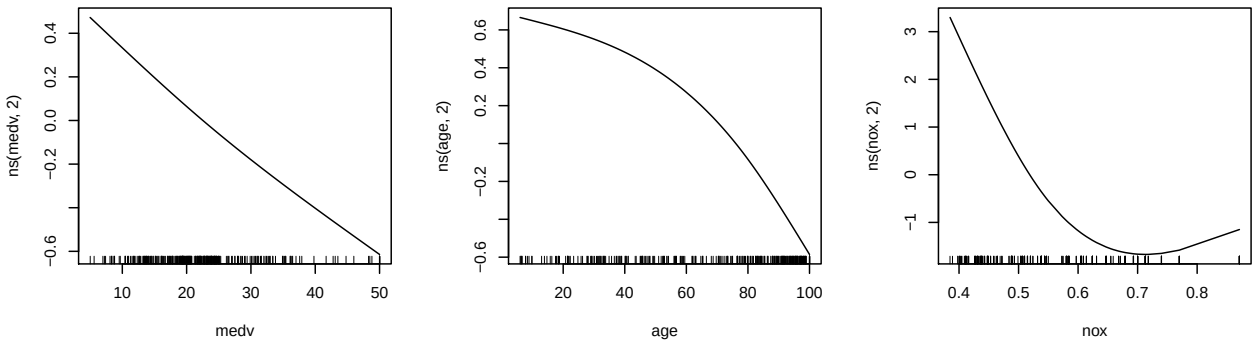
```
##     25%     50%     75%
## 45.325 78.800 93.900
```

```
attr(ns(Boston1$age,2), "knots")
```

```
##  50%
## 78.8
```

```
gfit2 <-  lm(dis ~ ns(medv,2)+ ns(age,2)+ ns(nox,2), data=Boston1)

par(mfrow=c(1,3))
plot.Gam(gfit2)
```




```
anova(gfit1, gfit2)
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ ns(medv, 4) + ns(age, 4) + ns(nox, 4)
## Model 2: dis ~ ns(medv, 2) + ns(age, 2) + ns(nox, 2)
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    291 239.58
## 2    297 257.08 -6   -17.503 3.5433 0.002112 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Reject H0, the model with fewer df is not appropriate

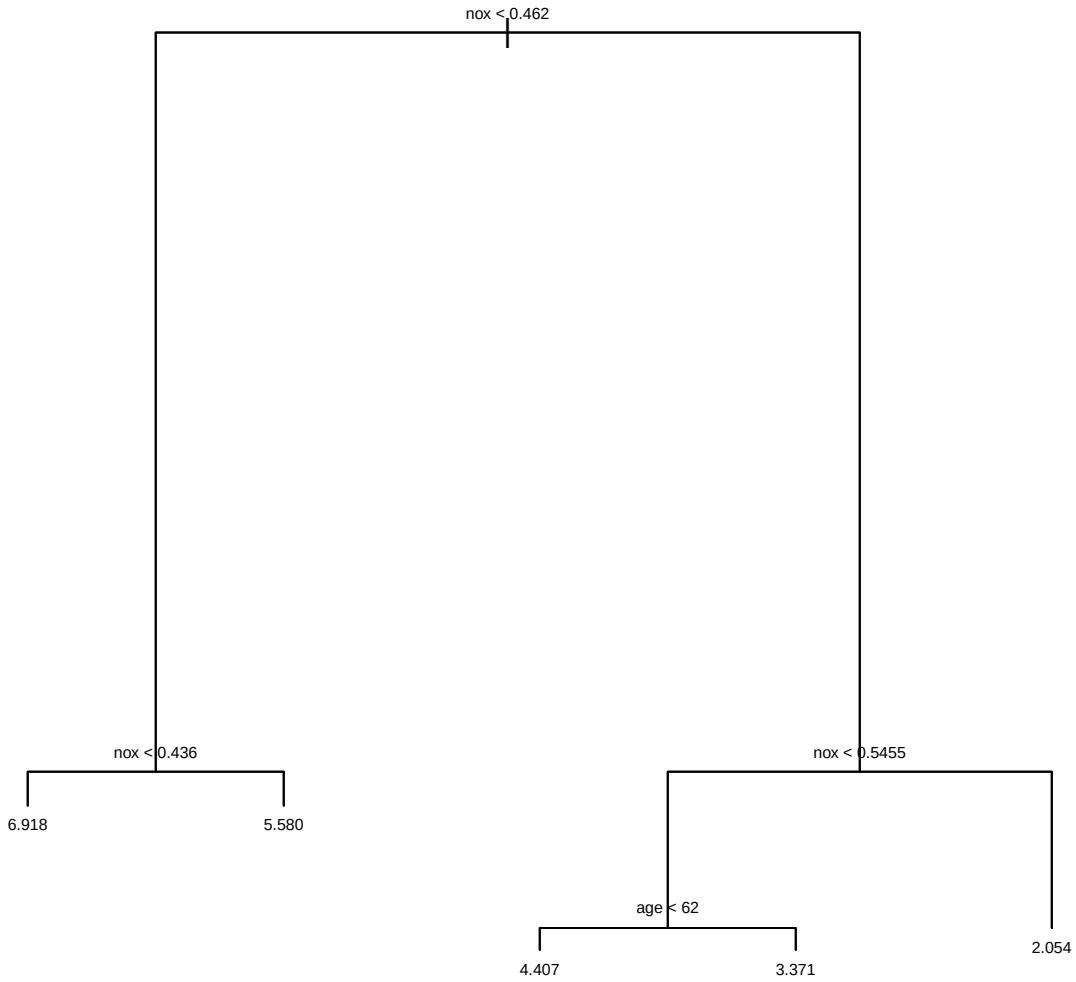
Q4

```
library(tree)
tree <- tree(dis ~ medv+age+nox, data=Boston1)
summary(tree)
```

```
##
## Regression tree:
## tree(formula = dis ~ medv + age + nox, data = Boston1)
## Variables actually used in tree construction:
## [1] "nox" "age"
## Number of terminal nodes:  5
## Residual mean deviance:  0.757 = 226.4 / 299
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.88700 -0.51640 -0.04421  0.00000  0.43300  3.79200
```

The fitted tree has 5 leaf nodes.

```r
plot(tree)
text(tree, cex=.5, pretty=0)
```




```r
mean(residuals(tree)^2)  # training MSE
```

```
## [1] 0.7445936
```

```r
pred <- predict(tree, Boston2)
mean((Boston2$dis - pred)^2) # test MSE
```

```
## [1] 1.319002
```

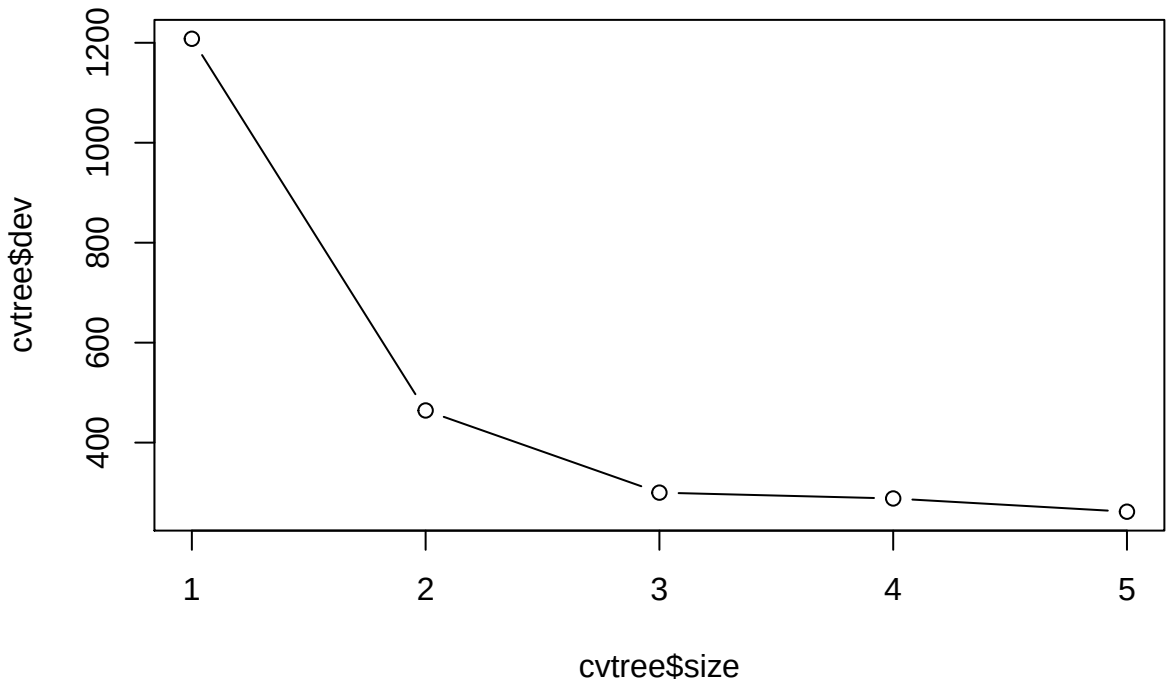
b)

```r
cvtree <- cv.tree(tree)
cvtree
```

```
## $size
## [1] 5 4 3 2 1
##
## $dev
## [1]  261.7371  288.1630  299.8596  464.2841 1208.0095
```

```
##
## $k
## [1]      -Inf  22.45566  34.89386 160.30947 758.08806
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cvtree$size,cvtree$dev,type="b")
```




```
w <- which.min(cvtree$dev)
cvtree$size[w]
```

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

```
# no pruning required
```

c)

```
pred <- predict(tree, Boston2)
mean((Boston2$dis - pred)^2) # test MSE for tree
```

```
## [1] 1.319002
```

```
pred <- predict(gfit2, Boston2)
mean((Boston2$dis - pred)^2) # test MSE for gam
```

```
## [1] 1.38424
```

The GAM has a slightly lower test MSE, but this is seed dependant

Q5

a)

```
set.seed(1)
x <- rnorm(100)
y <- 1 + .2*x+3*x^2+.6*x^3 + rnorm(100)
d <- data.frame(x=x,y=y)
summary(lm(y~poly(x, 10, raw = TRUE), data = d))
```

```
##
## Call:
## lm(formula = y ~ poly(x, 10, raw = TRUE), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9774 -0.5895 -0.1238  0.4923  2.1505
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.17283    0.19971   5.873 7.28e-08 ***
## poly(x, 10, raw = TRUE)1     0.71409    0.59009   1.210    0.229
## poly(x, 10, raw = TRUE)2     1.86854    1.29174   1.447    0.152
## poly(x, 10, raw = TRUE)3    -0.33114    1.68567  -0.196    0.845
## poly(x, 10, raw = TRUE)4     1.90383    2.14977   0.886    0.378
## poly(x, 10, raw = TRUE)5     0.55110    1.35654   0.406    0.686
## poly(x, 10, raw = TRUE)6    -1.26499    1.31956  -0.959    0.340
## poly(x, 10, raw = TRUE)7    -0.15569    0.39731  -0.392    0.696
## poly(x, 10, raw = TRUE)8     0.31987    0.32511   0.984    0.328
## poly(x, 10, raw = TRUE)9     0.01628    0.03817   0.426    0.671
## poly(x, 10, raw = TRUE)10 -0.02690    0.02749  -0.979    0.330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9719 on 89 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.9455
## F-statistic: 172.7 on 10 and 89 DF,  p-value: < 2.2e-16
```

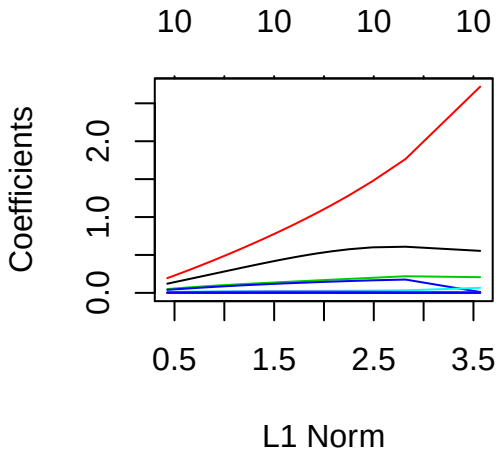
b)

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 2.0-16
```

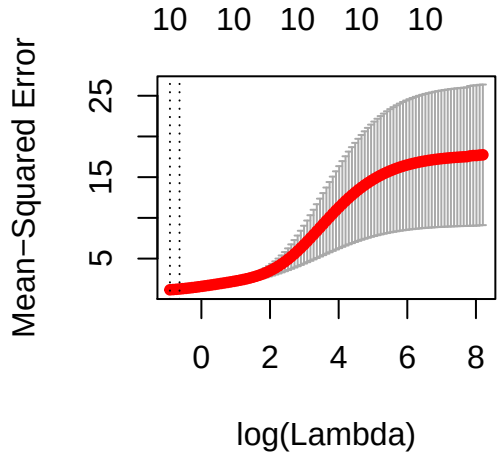
```
X <- model.matrix(y~poly(x, 10, raw = TRUE), data = d)
grid <- seq(0.001, 50, length = 100)
ridge.fit <- glmnet(X,y,alpha=0, lambda = grid) # for ridge
plot(ridge.fit)
```




```
cv.out <- cv.glmnet(X,y,alpha=0)
cv.out$lambda.min
```

```
## [1] 0.4000507
```

```
plot(cv.out)
```



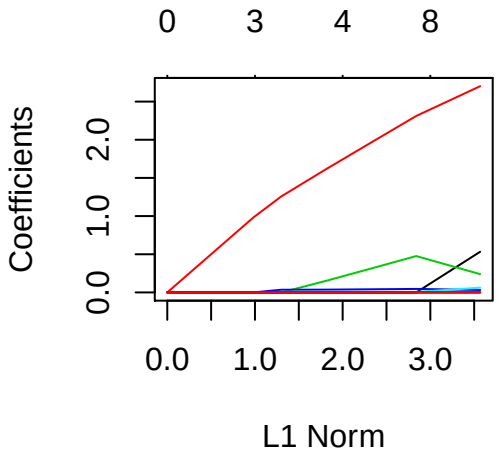

```
ridge.fit <- glmnet(X,y,alpha=0, lambda = cv.out$lambda.min)

coef(ridge.fit)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                 1.493577e+00
## (Intercept)                 .
## poly(x, 10, raw = TRUE)1    6.119103e-01
## poly(x, 10, raw = TRUE)2    1.859173e+00
## poly(x, 10, raw = TRUE)3    2.221955e-01
## poly(x, 10, raw = TRUE)4    1.736758e-01
## poly(x, 10, raw = TRUE)5    3.293830e-02
## poly(x, 10, raw = TRUE)6    1.120493e-02
## poly(x, 10, raw = TRUE)7    3.820191e-03
## poly(x, 10, raw = TRUE)8   -7.804433e-05
## poly(x, 10, raw = TRUE)9    2.623561e-04
## poly(x, 10, raw = TRUE)10 -2.576105e-04
```

c)

```
lasso.fit <- glmnet(X,y,alpha=1, lambda = grid)
plot(lasso.fit)
```




```
cv.out <- cv.glmnet(X,y,alpha=1)
cv.out$lambda.min
```

```
## [1] 0.04599611
```

```
lasso.fit <- glmnet(X,y,alpha=1, lambda = cv.out$lambda.min)

coef(lasso.fit)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                      s0
## (Intercept)                1.157917497
## (Intercept)                .
## poly(x, 10, raw = TRUE)1   0.434919485
## poly(x, 10, raw = TRUE)2   2.653795464
## poly(x, 10, raw = TRUE)3   0.321163376
## poly(x, 10, raw = TRUE)4   0.039994184
## poly(x, 10, raw = TRUE)5   0.037512827
## poly(x, 10, raw = TRUE)6   .
## poly(x, 10, raw = TRUE)7   0.002180384
## poly(x, 10, raw = TRUE)8   .
## poly(x, 10, raw = TRUE)9   .
## poly(x, 10, raw = TRUE)10  .
```

```
# lasso.fit <- glmnet(X,y,alpha=1, lambda = 2)
#
# coef(lasso.fit)
```

d)

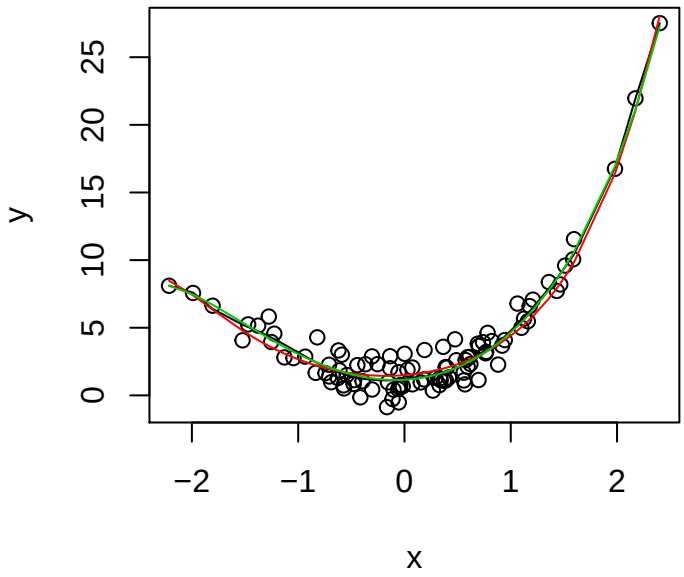
```
plot(x,y)
pred1<- predict(lm(y~poly(x, 10, raw = TRUE), data = d), newdata = data.frame(x=sort(x)))
lines(sort(x),pred1)
```

```
Xord <- X[order(X[,2]),]

pred2<- predict(ridge.fit, newx = Xord)
lines(sort(x),pred2, col = 2)



pred3<- predict(lasso.fit, newx = Xord)
lines(sort(x),pred3, col = 3)
```



Q6

```
ttrain <- read.csv("data/ttrain.csv", header=T, row.names=1)
ttest <- read.csv("data/ttest.csv", header=T, row.names=1)
head(ttrain)
```

```
##      Class    Sex   Age Survived
## 633    3rd   Male Adult       No
## 1735  Crew   Male Adult      Yes
## 900   Crew   Male Adult       No
## 1941   1st Female Adult      Yes
## 2067   2nd Female Adult      Yes
## 101    1st   Male Adult       No
```

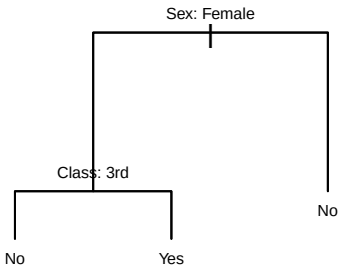
a)

```
library(tree)
tree <- tree(Survived ~ ., data=ttrain)
summary(tree)
```

```
##
```

```
## Classification tree:
## tree(formula = Survived ~ ., data = ttrain)
## Variables actually used in tree construction:
## [1] "Sex"   "Class"
## Number of terminal nodes:  3
## Residual mean deviance:  0.9852 = 1732 / 1758
## Misclassification error rate: 0.2107 = 371 / 1761
```

```
plot(tree)
text(tree, cex=.5, pretty=0)
```



Fitted model: males have no chance of survival. For females, those in 3rd class have no chance of survival. The rest are predicted as survived. Age is not in the model.

```
prob <- predict(tree, ttrain)[,2]
pred <- factor(ifelse(prob < .5, "No", "Yes"))
tab <- table(pred, ttrain$Survived)
tab
```

```
##
## pred    No  Yes
##   No  1178  356
##   Yes   15  212
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.6267606
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.01257334
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.9339207
```

```
mean(pred != ttrain$Survived)
```

```
## [1] 0.2106758
```

For the training set:

62.6760563% of survivors are mis classified.

Of those who died 1.2573345% are mis classified.

93.3920705% of predicted survivors actually survived.

Overall error rate is 0.2106758.

```
prob <- predict(tree, ttest)[,2]
pred <- factor(ifelse(prob < .5, "No", "Yes"))
```

```
tab <- table(pred, ttest$Survived)
tab
```

```
##
## pred   No Yes
##   No  292 101
##   Yes   5  42
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.7062937
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.01683502
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.893617
```

```
mean(pred != ttest$Survived)
```

```
## [1] 0.2409091
```

For the test set:

70.6293706% of survivors are mis classified.

Of those who died 1.6835017% are mis classified.

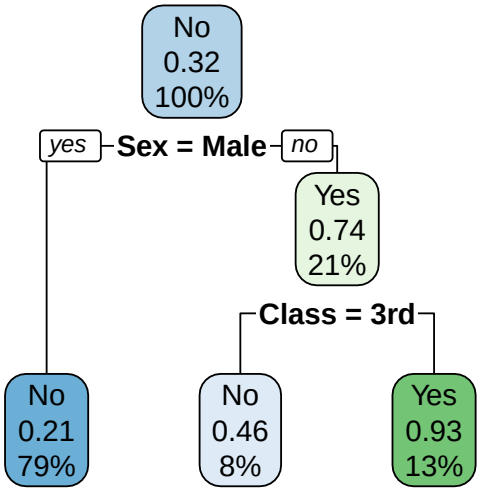
89.3617021% of predicted survivors actually survived.

Overall error rate is 0.2409091.

Using Rpart:

```
library(rpart)
rp <- rpart(Survived ~ ., data=ttrain)

library(rpart.plot)
rpart.plot(rp)
```

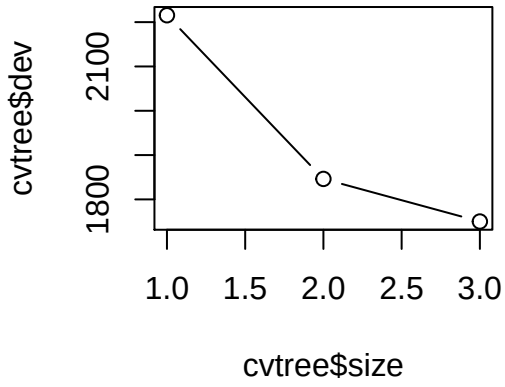


b)

```
cvtree <- cv.tree(tree)
cvtree
```

```
## $size
## [1] 3 2 1
##
## $dev
## [1] 1750.138 1846.204 2215.514
##
## $k
## [1]     -Inf 111.2986 371.2492
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cvtree$size,cvtree$dev,type="b")
```




```
w <- which.min(cvtree$dev)
cvtree$size[w]
```

```
## [1] 3
```

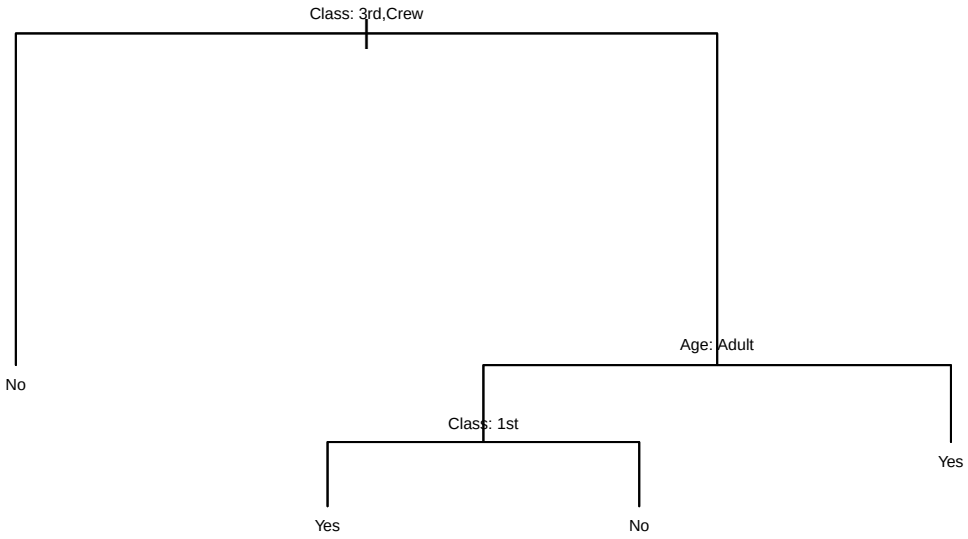
```
# no pruning required
```

c)

```
tree <- tree(Survived ~ Age + Class, data=ttrain)
summary(tree)
```

```
##
## Classification tree:
## tree(formula = Survived ~ Age + Class, data = ttrain)
## Number of terminal nodes:  4
## Residual mean deviance:  1.148 = 2017 / 1757
## Misclassification error rate: 0.2731 = 481 / 1761
```

```
plot(tree)
text(tree, cex=.5, pretty=0)
```




```
prob <- predict(tree, ttrain)[,2]
pred <- factor(ifelse(prob < .5, "No", "Yes"))
unique(cbind(ttrain[, 1:3], pred))
```

```
##      Class    Sex   Age pred
## 633    3rd   Male Adult   No
## 1735  Crew   Male Adult   No
## 1941   1st Female Adult  Yes
## 2067   2nd Female Adult   No
## 101    1st   Male Adult  Yes
## 1484   3rd Female Adult   No
## 226    2nd   Male Adult   No
## 2198  Crew Female Adult   No
## 1511   3rd   Male Child   No
## 1542   3rd Female Child   No
## 1499   2nd   Male Child  Yes
## 1525   2nd Female Child  Yes
## 1520   1st Female Child  Yes
## 1492   1st   Male Child  Yes
```

Fitted model: 3rd class and crew have no chance of survival. For those in 1st and 2nd class children are predicted to have survived. Adults in 1st class are predicted to have survived, but those in the 2nd did not.

```
prob <- predict(tree, ttest)[,2]
pred <- factor(ifelse(prob < .5, "No", "Yes"))
tab <- table(pred, ttest$Survived)
tab
```

```
##
## pred   No Yes
##   No  271  99
##   Yes  26  44
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.6923077
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.08754209
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.6285714
```

```
mean(pred != ttest$Survived)
```

```
## [1] 0.2840909
```

For the test set:

69.2307692% of survivors are mis classified.

Of those who died 8.7542088% are mis classified.

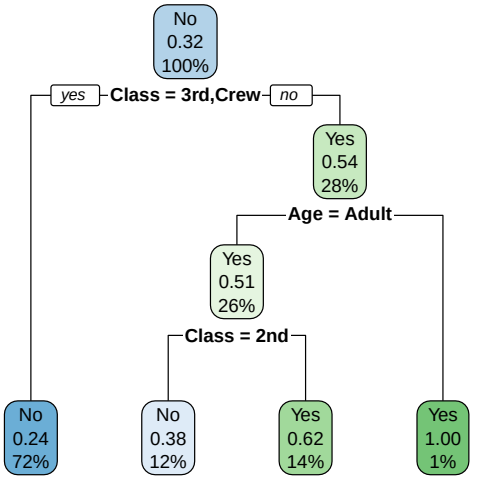
62.8571429% of predicted survivors actually survived.

Overall error rate is 0.2840909.

Using Rpart:

```
rp <- rpart(Survived ~ Age + Class, data=ttrain)

rpart.plot(rp)
```



d)

```
suppressMessages(library(randomForest))
bag <- randomForest(Survived ~ ., data=ttrain)
varImpPlot(bag)
```


```
pred <- predict(bag, newdata=ttest)
tab <- table(pred, ttest$Survived)
tab
```

```
##
## pred   No Yes
##   No  292 100
##   Yes   5  43
```

```
tab[1,2]/sum(tab[,2])
```

```
## [1] 0.6993007
```

```
tab[2,1]/sum(tab[,1])
```

```
## [1] 0.01683502
```

```
tab[2,2]/sum(tab[2,])
```

```
## [1] 0.8958333
```

```
mean(pred != ttest$Survived)
```

```
## [1] 0.2386364
```

For the test set:

69.9300699% of survivors are mis classified.

Of those who died 1.6835017% are mis classified.

89.5833333% of predicted survivors actually survived.

Overall error rate is 0.2386364.

Order of variable importance is sex, class and age.

Q7

Any sensible answer for the tuning part is ok.

```
heart <- read.csv("data/heart.csv", row.names=1)
head(heart)
```

```
##   Age Sex    ChestPain RestBP Chol Fbs RestECG MaxHR ExAng Oldpeak Slope
## 1  63   1      typical    145  233   1       2   150     0     2.3     3
## 2  67   1 asymptomatic    160  286   0       2   108     1     1.5     2
## 3  67   1 asymptomatic    120  229   0       2   129     1     2.6     2
## 4  37   1   nonanginal    130  250   0       0   187     0     3.5     3
## 5  41   0   nontypical    130  204   0       2   172     0     1.4     1
## 6  56   1   nontypical    120  236   0       0   178     0     0.8     1
##   Ca       Thal AHD
## 1  0      fixed  No
## 2  3     normal Yes
## 3  2 reversable Yes
## 4  0     normal  No
## 5  0     normal  No
## 6  0     normal  No
```

```
heart <- na.omit(heart)
set.seed(2)
s <- sample(nrow(heart), 200)
heartTrain <- heart[s,]
heartTest <- heart[-s,]
```

```
library(e1071)
fit.svm <- svm(AHD~., data = heartTrain, kernel = "radial")
summary(fit.svm)
```

```
##
## Call:
## svm(formula = AHD ~ ., data = heartTrain, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05882353
##
## Number of Support Vectors:  106
##
##  ( 52 54 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
pred <- predict(fit.svm, newdata = heartTest)

table(pred,  heartTest$AHD)
```

```
##
## pred  No Yes
##   No  37  10
##   Yes 13  37
```

```r
mean(pred != heartTest$AHD)
```

```
## [1] 0.2371134
```

```r
tune.out <- tune(svm,AHD~., data = heartTrain, kernel = "radial", ranges = list(cost = 10^seq(-1, 6, by
tune.out
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost gamma
##  10000 1e-06
##
## - best performance: 0.16
```

```r
fit.svm <- svm(AHD~., data = heartTrain, kernel = "radial", cost = 10000, gamma = 0.00001)
summary(fit.svm)
```

```
##
## Call:
## svm(formula = AHD ~ ., data = heartTrain, kernel = "radial",
##     cost = 10000, gamma = 1e-05)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10000
##       gamma:  1e-05
##
## Number of Support Vectors:  77
##
##  ( 37 40 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```r
pred <- predict(fit.svm, newdata = heartTest)

table(pred,  heartTest$AHD)
```

```
##
## pred  No Yes
##   No  42  12
##   Yes  8  35
```

```r
mean(pred != heartTest$AHD)
```

```
## [1] 0.2061856
```

```r
bag <- randomForest(AHD ~ ., data=heartTrain)
pred <- predict(bag, heartTest)
mean(pred != heartTest$AHD, na.rm=T) # test error
```

```
## [1] 0.2268041
```