Compulsory exercise 2: 28

Lars A. M. Olsen, Ida M. Sandum, Ellen Skrimstad

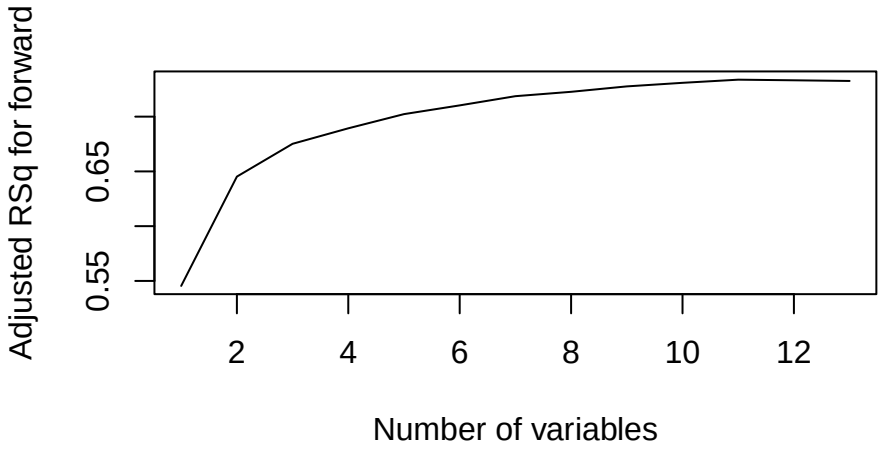
2022-04-03

Problem 1

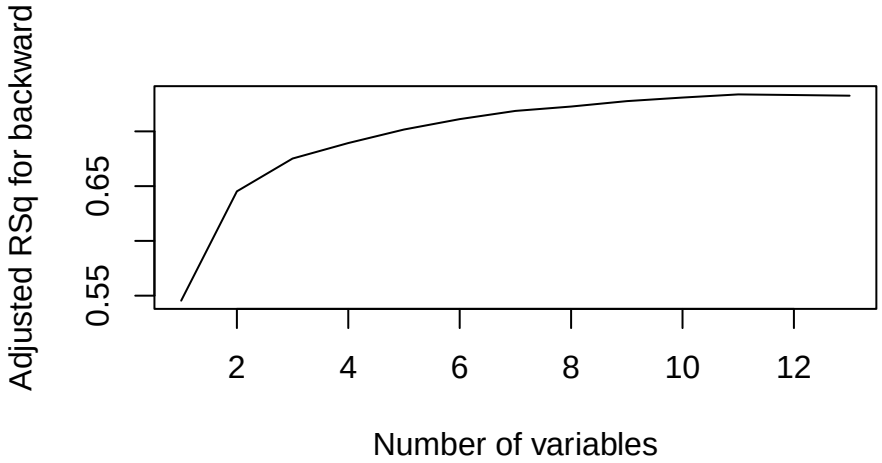
a)

Performing forward and backward stepwise selection:

```
library(MASS)
library(leaps)
# Hiding to save space as this was given in the project description
#str(Boston)

set.seed(1)
boston <- scale(Boston, center =T, scale =T)
train.ind = sample(1:nrow(boston), 0.8*nrow(boston))
boston.train = data.frame(boston[train.ind, ])
boston.test = data.frame(boston[-train.ind, ])
number_variables = 14

# Forward method
forward_method = regsubsets(medv~., data=boston.train,
                            nvmax =number_variables, method = "forward")
forward_method_summary = summary(forward_method)

#backward method
backward_method2 = regsubsets(medv~., data=boston.train, nvmax =number_variables, method = "backward")
backward_method2_summary = summary(backward_method2)

plot(forward_method_summary$adjr2, xlab="Number of variables",
     ylab = "Adjusted RSq for forward", type="l")
```




```
plot(backward_method2_summary$adjr2, xlab="Number of variables",
     ylab = "Adjusted RSq for backward", type="l")
```



b)

We can find the four predictors forward stepwise selection considers the best by looking at the summary of the method. The four best are in decreasing order lstat, rn, ptratio and dis.

```
forward_method_summary
```

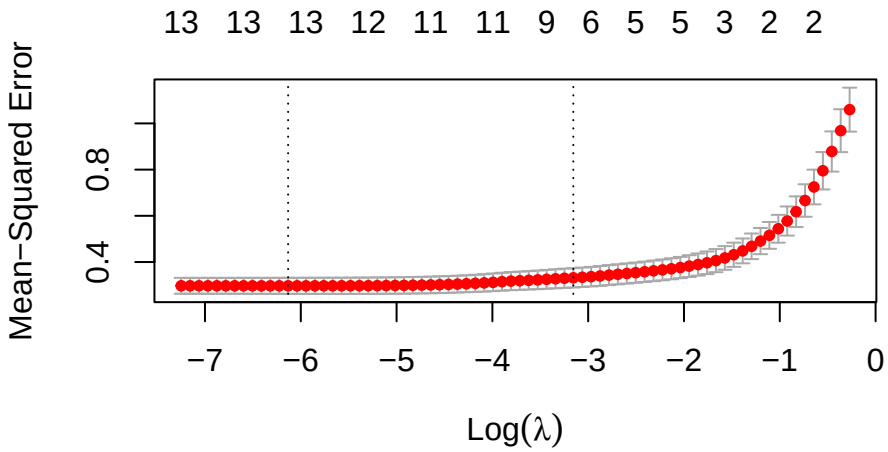
```
## Subset selection object
## Call: regsubsets.formula(medv ~ ., data = boston.train, nvmax = number_variables,
##     method = "forward")
```

```
## 13 Variables  (and intercept)
##         Forced in Forced out
## crim        FALSE      FALSE
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## 1 subsets of each size up to 13
## Selection Algorithm: forward
##           crim zn  indus chas nox rm  age dis rad tax ptratio black lstat
## 1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " "     " "   "*"
## 2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "     " "   "*"
## 3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "*"     " "   "*"
## 4  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     " "   "*"
## 5  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     "*"   "*"
## 6  ( 1 )  " "  " " " "   " "  "*" "*" " " "*" " " " " "*"     "*"   "*"
## 7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     "*"   "*"
## 8  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
## 9  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 10  ( 1 ) " "  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*"  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

c)

Now to run cross-validation with Lasso.

```
library(glmnet)
set.seed(1)
x_train   <- model.matrix(medv~.,data= boston.train)
y_train   <- boston.train$medv

set.seed(1)
cv.out <- cv.glmnet(x_train, y_train, alpha=1, nfolds=5)
plot(cv.out)
```



ii) The best lambda is as printed below around 0.003.

```
best_lambda_lasso <- cv.out$lambda.min
best_lambda_lasso
```

```
## [1] 0.002172032
```

iii)

```
coef(cv.out)
```

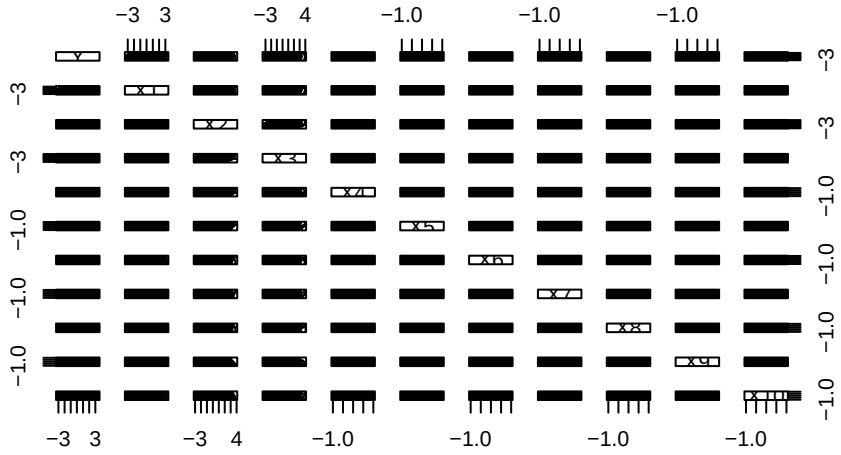
```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  0.024709949
## (Intercept)  .
## crim         .
## zn           .
## indus        .
## chas         0.070509379
## nox         -0.004688111
## rm           0.350606336
## age          .
## dis         -0.060636734
## rad          .
## tax          .
## ptratio     -0.165115064
## black        0.081161951
## lstat       -0.423934302
```

d)

TRUE, FALSE, FALSE, TRUE

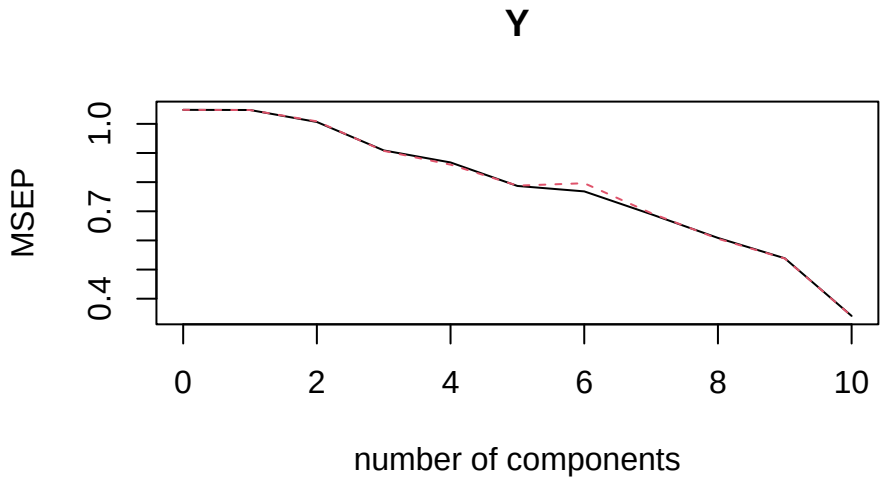
Problem 2

To understand the data set we start by plotting a pairs-plot.

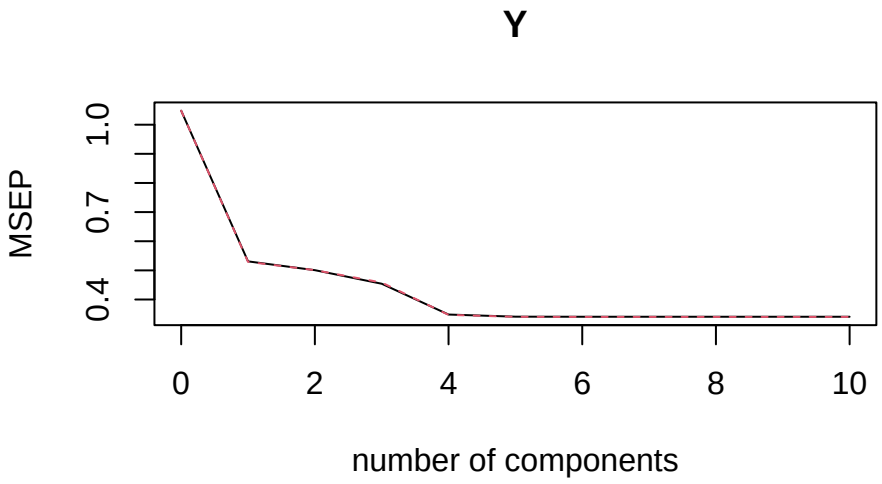


a)

```
pcr_model <- pcr(Y~., data = synthetic.train, scale = TRUE, validation="CV")
validationplot(pcr_model, val.type ="MSEP")
```




```
library(pls)
plsr_model <- plsr(Y~., data=synthetic.train,scale=TRUE, validation="CV")
validationplot(plsr_model,val.type="MSEP")
```



b)

From the plots we can see that for PLSR the mean square error of prediction drops low for far fewer components.For PLSR the error end up at 0.4 after only adding 4 components, while for PCR this happens only after adding all ten components.

PCR is unsupervised, that is it does not use the values for Y to choose directions. Because of the it in does not necessarily choose the direction which explains the correlation to the response first, but instead the direction which explains the most variance in the covariates. This is also the main difference between PCR and PLS, that PLS includes the response Y when choosing directions. Thus we can assume that in general PLS model decreases MSEP for lower numbers of components in the case that some of the covariates are strongly related to the response.

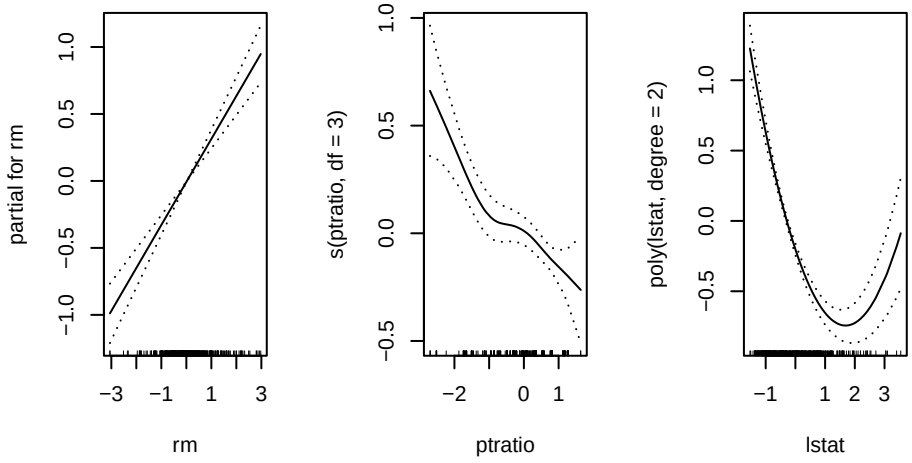
Specifically for this data set we can see from the pairs plot above that there is correlation between Y and X_1 and between X_2 and X_3. This explains the difference between the two plots, namely how PLS has a large drop after adding the first component, because it chooses this one as X_1, while PCR most likely chooses a direction to explain most of both X_2 and X_3 first and as such does not have a similar drop. In fact PCR does not explain any of the correlation to the response in the first component.

Problem 3

a)

TRUE, FALSE, FALSE, TRUE # b)

```
library(gam)
additive_model <- gam(medv ~ rm + s(ptratio,df=3) + poly(lstat,degree = 2), data=boston.train)
par(mfrow=c(1,3))
plot(additive_model, se=T)
```

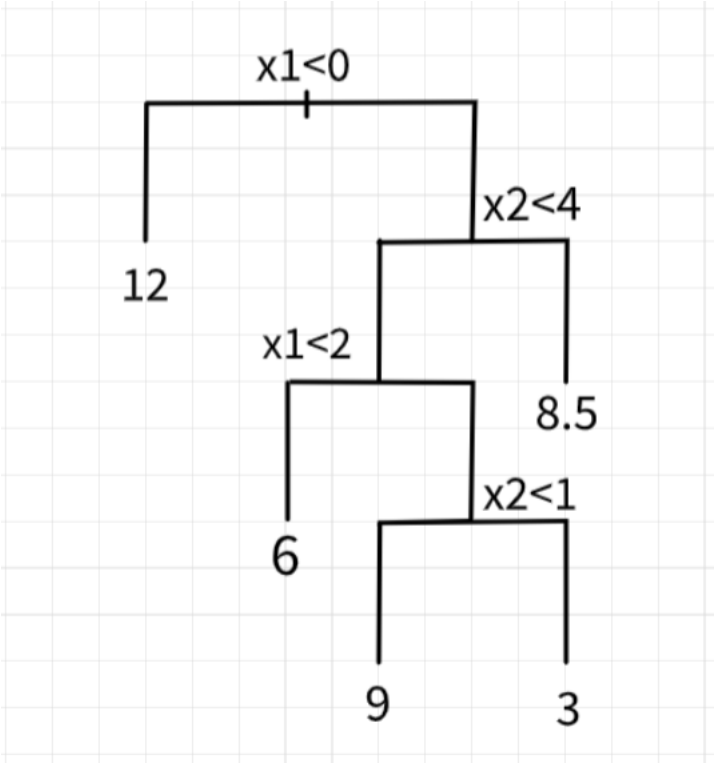


Problem 4

a)

FALSE, TRUE, TRUE, TRUE

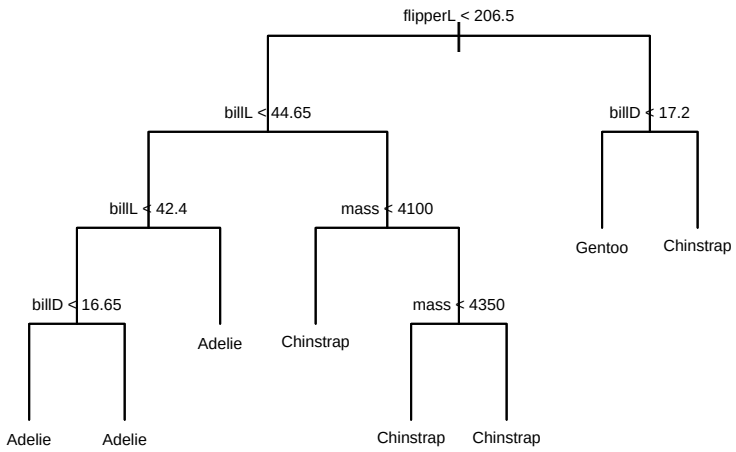
b)



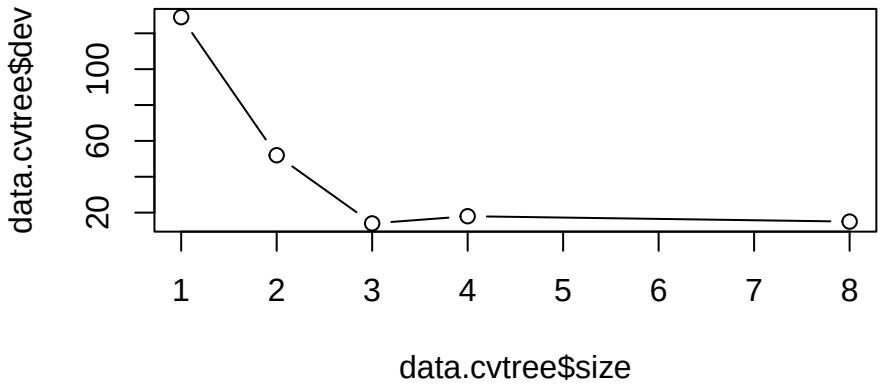
See figure 1.

c)

```
library(tidyverse)
library(tree)
library(palmerpenguins)  # Contains the data set 'penguins'.
data(penguins)

names(penguins) <- c("species", "island", "billL", "billD", "flipperL", "mass", "sex",
                     "year")

Penguins_reduced <- penguins %>% dplyr::mutate(mass = as.numeric(mass), flipperL = as.numeric(flipperL)
                                               year = as.numeric(year)) %>% drop_na()

# We do not want 'year' in the data (this will not help for future predictions)
Penguins_reduced <- Penguins_reduced[, -c(8)]

set.seed(4268)
# 70% of the sample size for training set
training_set_size <- floor(0.7 * nrow(Penguins_reduced))
train_ind <- sample(seq_len(nrow(Penguins_reduced)), size = training_set_size)
train <- Penguins_reduced[train_ind, ]
test <- Penguins_reduced[-train_ind, ]

# 4c, i)

# Simple classification tree using Gini index
data.tree = tree(formula=species ~ ., data = train, split="gini")
plot(data.tree, type="uniform")
text(data.tree,pretty=1, cex= 0.5)
title("Classification tree")
```

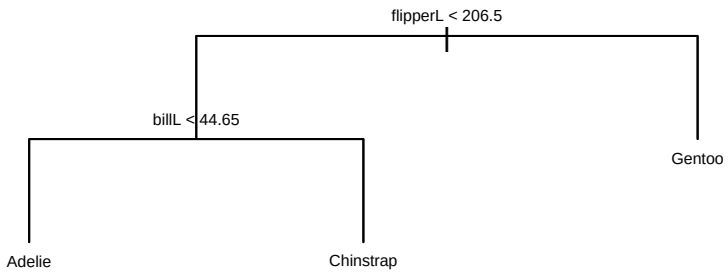



```
# ii)
# Cost-complexity pruning 10-fold CV
set.seed(123)
data.cvtree = cv.tree(data.tree, FUN=prune.misclass, K=10)
plot(data.cvtree$size, data.cvtree$dev, type = "b")
```



We see from the plot that the deviation is lowest for 3 terminal nodes.

```r
# iii)
# New tree using 3 terminal nodes
prune.data <- prune.misclass(data.tree, best = 3)
plot(prune.data, type="uniform")
text(prune.data, pretty = 1, cex = 0.5)
```



From the two plots of the trees we see that flipper length and bill length and bill depth are used in the top splits.

```r
# Predicting species
pred.prune = predict(prune.data, newdata = test, type = "class")
response.test = Penguins_reduced$species[-train_ind]

# Misclassification table
misclass.prune = table(pred.prune, response.test)
misclass.prune
```

```
##            response.test
## pred.prune  Adelie Chinstrap Gentoo
##   Adelie        42         3      0
##   Chinstrap      0        14      0
##   Gentoo         0         3     38
```

```r
error_rate = 1 - sum(diag(misclass.prune))/sum(misclass.prune)
```

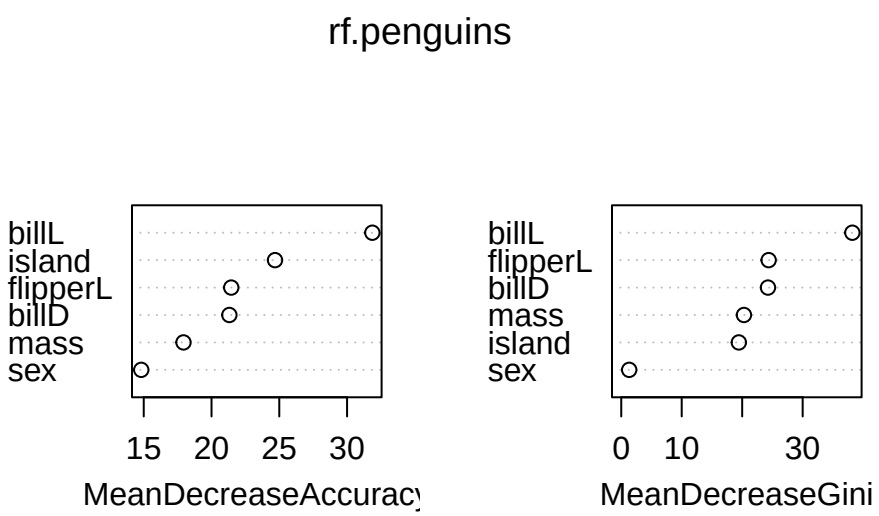
We see that the misclassification error rate is 0.06.

d)

Now we are going to use a more advanced method, and we have chosen to use random forests. We let the number of trees be 500, as a large number of trees will not lead to overfitting in the case of random forests.

We also choose $m = \sqrt{p}$ as the number of predictors to be considered at each split. We want this number to be rather low, in case there is a very strong predictor in the data set. We do not want such a predictor to overshadow the rest of them. Considering only a subset of the predictors is a way to overcome this problem.

```
# d)
library(randomForest)

# Using the random forest method
rf.penguins = randomForest(species ~ ., data = train, mtry = round(sqrt(ncol(Penguins_reduced)-1)) -
    1, ntree = 500, importance = TRUE)

varImpPlot(rf.penguins)
```



In this plot we see that bill length, flipper length and bill depth are important variables, which was expected from the previous trees. We also see that island has a high mean decrease accuracy. This was not expected from the trees, as island has not been a split. Combining previous information and this new information, we therefore say bill length and flipper length are the two most important covariates.

```
# Predicting species using random forest
pred.rf = predict(rf.penguins, newdata = test)

misclass.rf = table(pred.rf, response.test)
misclass.rf
```

```
##            response.test
## pred.rf     Adelie Chinstrap Gentoo
##   Adelie        42         1      0
##   Chinstrap      0        19      0
##   Gentoo         0         0     38
```

```
error_rf = 1 - sum(diag(misclass.rf))/sum(misclass.rf)
```

The misclassification error rate for the test data is now 0.01, which is an improvement from before.

Problem 5

a)

FALSE, TRUE, TRUE, TRUE

b)

```
# 5b)
library(e1071)

set.seed(123)

# 10 fold cross-validation to find cost parameter for linear boundary
CV_linear = tune(svm, species ~ ., data = train, kernel = "linear", ranges = list(cost = c(0.001,
    0.01, 0.1, 1, 5, 10, 50)))
best_model_linear = CV_linear$best.model
cost_linear = CV_linear$best.parameters
error_linear = CV_linear$best.performance

# 10 fold cross-validation to find cost and gamma parameter for radial boundary
CV_kernel = tune(svm, species ~ ., data = train, kernel = "radial", ranges = list(cost = c(0.01,
    0.1, 1, 5, 10, 100, 1000), gamma = c(0.01, 0.1, 1, 10, 100)))
best_model_radial = CV_kernel$best.model
cost_kernel = CV_kernel$best.parameters$cost
gamma_kernel = CV_kernel$best.parameters$gamma
error_radial = CV_kernel$best.performance
```

We see that the best cost when classifying using linear boundary is 5, and this gives an error rate of 0. Using radial boundary we have a cost of 5, a gamma of 0.1, and an error rate of 0.

```
# Prediction of species using linear boundary
pred_linear = predict(best_model_linear, test)
misclass_linear = table(pred_linear, response.test)
error_linear_pred = 1 - sum(diag(misclass_linear))/sum(misclass_linear)

# Prediction of species using radial boundary
pred_radial = predict(best_model_radial, test)
misclass_radial = table(pred_radial, response.test)
error_radial_pred = 1 - sum(diag(misclass_radial))/sum(misclass_radial)

misclass_linear
```

```
##             response.test
## pred_linear Adelie Chinstrap Gentoo
##   Adelie        41         0      0
##   Chinstrap      1        20      0
##   Gentoo         0         0     38
```

```
misclass_radial
```

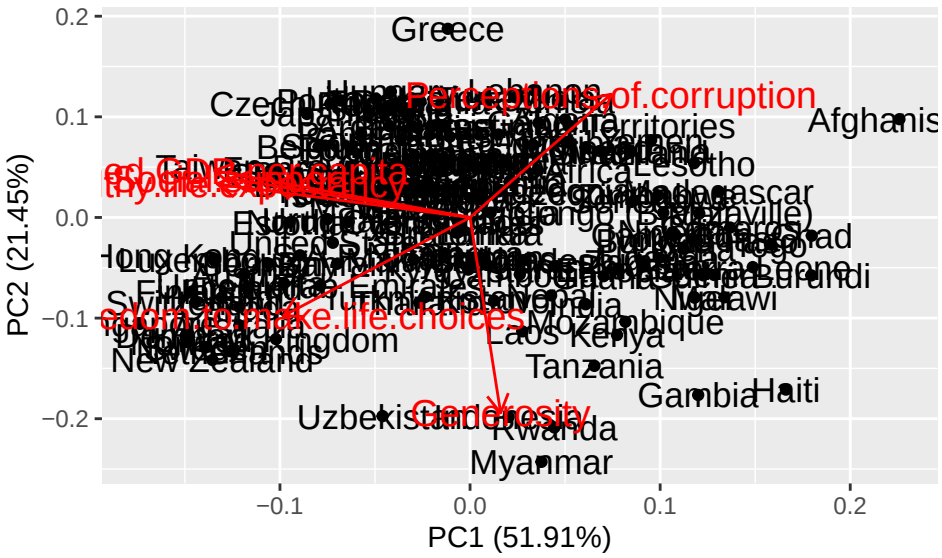
```
##            response.test
## pred_radial Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

From the misclassification tables we see that one of the penguins is misclassified using linear boundary, and none are misclassified using radial boundary. The misclassification error rates are 0.01 and 0, respectively.

Using this information it seems like a radial classifier boundary is better, and we therefore prefer that one in the case of the penguins data set. This obviously depends on the data set, and is not a general decision.

Problem 6

```
library(ggfortify)
pca_mat = prcomp(happiness.X, center = T, scale = T)

# Score and loadings plot:
autoplot(pca_mat, data = happiness.X, colour = "Black", loadings = TRUE, loadings.colour = "red",
    loadings.label = TRUE, loadings.label.size = 5, label = T, label.size = 4.5)
```



a)

i) Two characteristics observed in relations between the variables are:

- We see that PC2 has a significant effect on "Generosity", wile PC1 hardly has any effect. "Logged GDP per capita", "Social support" and "Healthy life expectancy", on the other hand, has a low effect from PC2, but high effect from PC1. This indicates that there is a low correlation between the variables generosity and the three others.

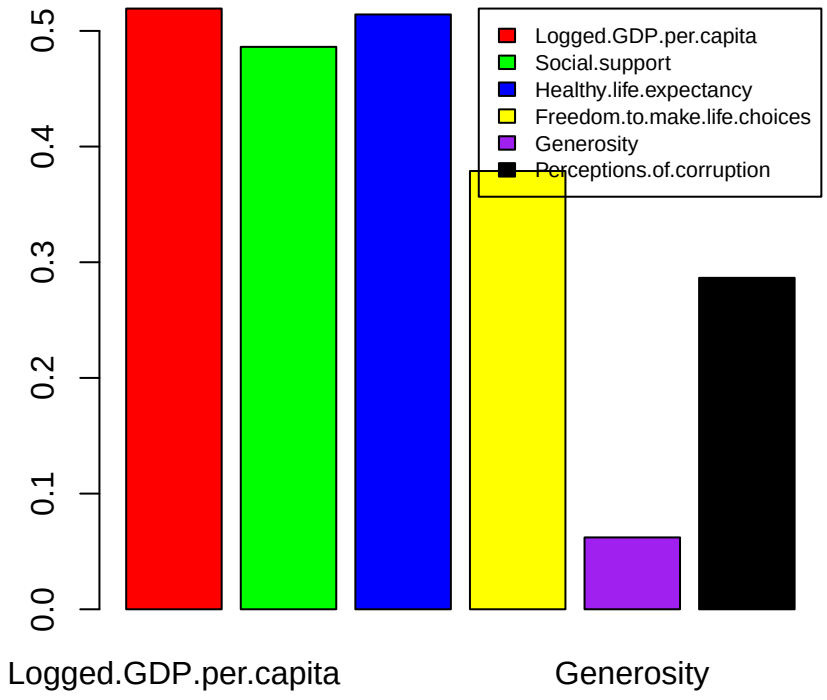
- We also observe that "Freedom to make life choices" and "perception of corruption" are on opposite sides of the plot. This indicates that these are different. So, if a country has one, it does not have much of the other, since they almost are completely different in both PC1 and PC2.

ii) Afghanistan can be consider to an outlier among the countries in the selection.

b)

i)

```
first_princ <- data.frame(pca_mat$rotation)$PC1

?barplot()

barplot(abs(first_princ), names.arg = rownames(pca_mat$rotation),legend = colnames(happiness.XY)[-1], c
```

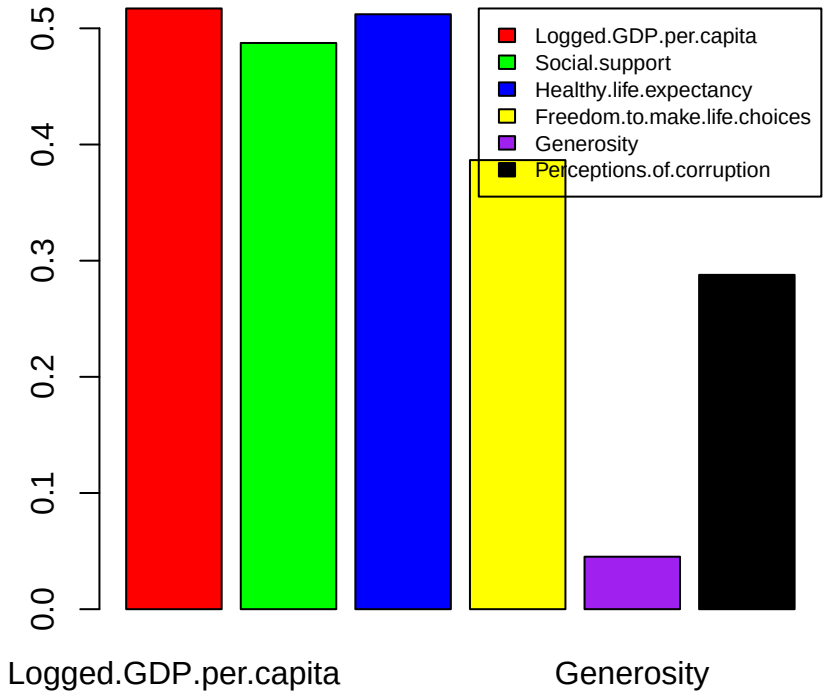


ii)

```
library(pls)
plsr_model <- plsr(Ladder.score ~ ., data=happiness.XY, scale=T)
```

iii)

```
plsr_plot <- plsr_model$loadings[,c('Comp 1')]
barplot(abs(plsr_plot),legend = colnames(happiness.XY)[-1], col = c("red","green","blue","yellow","purp
```



We see that these plots are relatively similar. But if we look at the numerical values for each, we see that the biggest difference is that Generosity is slightly smaller for the PLSR, than it is for PCA. For the other variables the numerical values are almost the same for PLSR and PCA.

iv) The three most important predictors to predict the happiness score based on the PLSR bar graph from iii) are

- logged GDP per capita
- Social support
- Healthy life expectancy These are the bars that are largest.

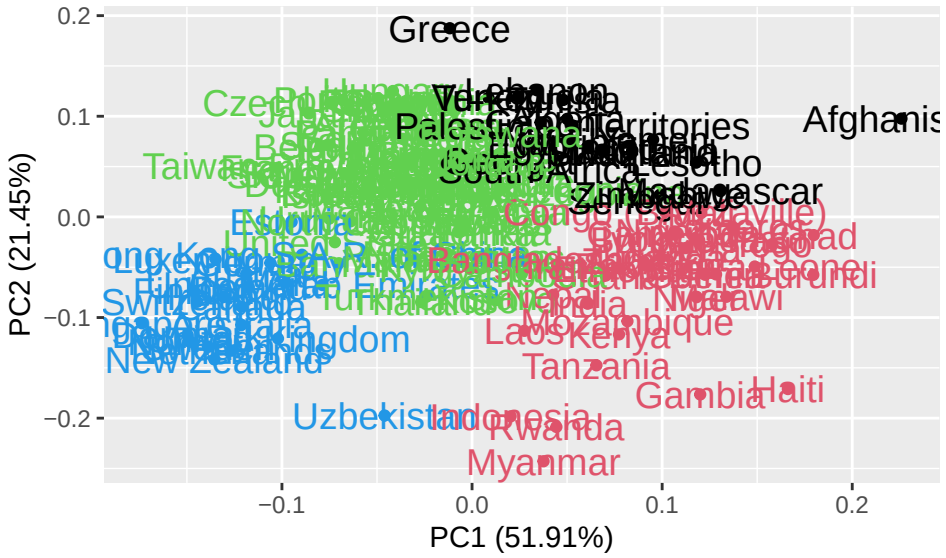
c)

FALSE, FALSE, FALSE, TRUE

d)

i)

```
library(stats)
set.seed(1)
K = 4  # your choice
km.out = kmeans(happiness.X, K)

autoplot(pca_mat, data = happiness.X, colour = km.out$cluster, label = T, label.size = 5,
    loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 3)
```



ii)

Printing the countries in each cluster, we find that: - cluster 1: countries in black - cluster 2: countries in red - cluster 3: countries in green - cluster 4: countries in blue

Then we print the average happiness score for each cluster:

```
set.seed(1)
average_lst <- list(1:K)
average_lst
```

```
## [[1]]
## [1] 1 2 3 4
```

```r
for (i in 1:K){
  cluster_nr <- km.out$cluster == i
  country = which(cluster_nr==TRUE)
  average = mean(happiness.XY[country, ]$Ladder.score)
  average_lst[i] = average
}
average_lst
```

```
## [[1]]
## [1] 4.463091
##
## [[2]]
## [1] 4.575147
##
## [[3]]
## [1] 5.875
##
## [[4]]
## [1] 7.030952
```

From this plot we see that the blue cluster, 4, containing countries such as Norway, Iceland, Denmark and New Zealand, has the highest average on the happiness score. While the black cluster, 1, containing Afghanistan and South-America, has the lowest average.

If we look at the plot, we see a pattern with happiness and distribution of countries along the x-axis. The happiest cluster is the blue cluster, where the countries in general are far left in the plot. The second happiest cluster, the green cluster, has a distribution of countries in the middle left of the plot. The two clusters left, black and red, has approximately the same happiness score, which is low, and these countries are in centered around the middle right of the plot.

We conclude that for a given country, happiness increases the further to the left the country is located in the plot.