Compulsory exercise 2: Group 5

TMA4268 Statistical Learning V2022

Aleksander Johnsen Solberg and Gjermund Oscar Lyckander

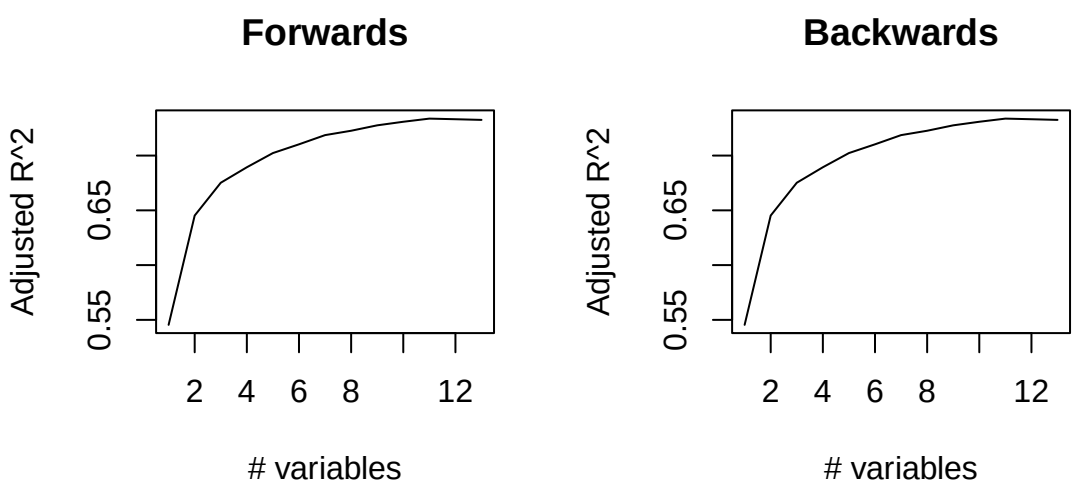
04 April, 2022

Problem 1

```
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, ])
```

a)

```
set.seed(1)
forward_stepwise = regsubsets(medv ~ ., data = boston.train, nvmax = 13, method = 'forward')
backward_stepwise = regsubsets(medv ~ ., data = boston.train, nvmax = 13, method = 'backward')

forward_stepwise_summary = summary(forward_stepwise)
backward_stepwise_summary = summary(backward_stepwise)
#forward_stepwise_summary
#backward_stepwise_summary

par(mfrow=c(1,2))
plot(forward_stepwise_summary$adjr2, xlab = '# variables', ylab = 'Adjusted R^2', type='l', main='Forwa
plot(forward_stepwise_summary$adjr2, xlab = '# variables', ylab = 'Adjusted R^2', type='l', main='Backwa
```



b)

```
forward_stepwise_summary$outmat
```

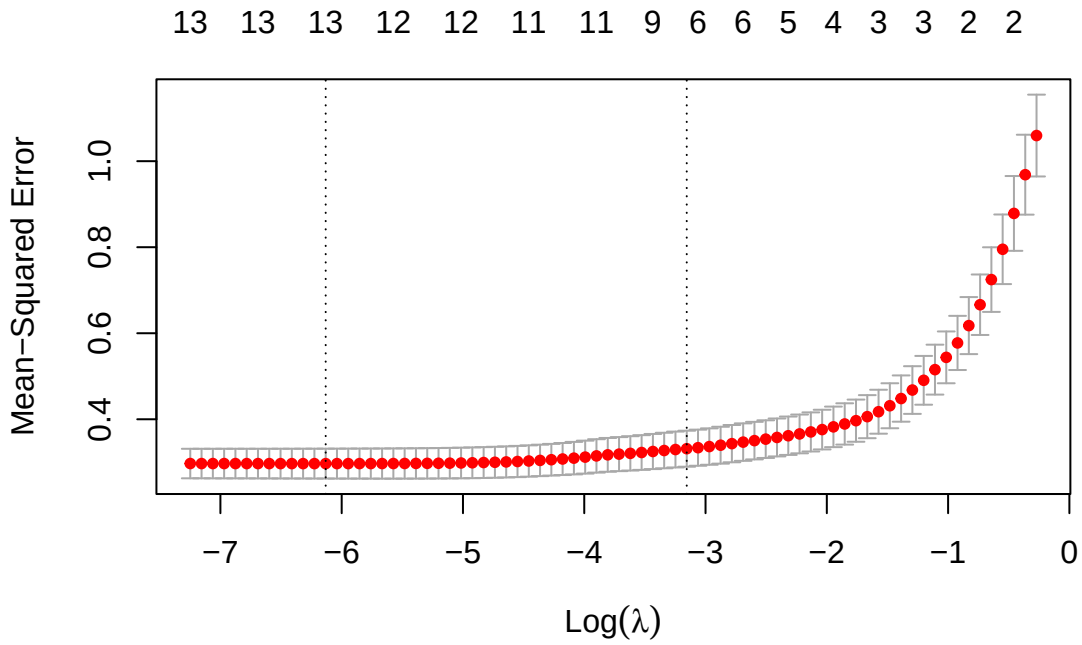
```
##           crim zn  indus chas nox rm  age dis rad tax ptratio black lstat
## 1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " "     " "   "*"
## 2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "     " "   "*"
## 3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "*"     " "   "*"
## 4  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     " "   "*"
## 5  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     "*"   "*"
## 6  ( 1 )  " "  " " " "   " "  "*" "*" " " "*" " " " " "*"     "*"   "*"
## 7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     "*"   "*"
## 8  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
## 9  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 10  ( 1 ) " "  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*"  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

We choose the predictors 'rm, 'dis', 'ptratio' and 'lstat'.

c)

i)

```
set.seed(1)
y = boston.train$medv
x = data.matrix(boston.train[, -14])

cv_lasso = cv.glmnet(x, y, alpha=1, nfolds=5)
plot(cv_lasso)
```



ii)

```
lasso_best_lambda = cv_lasso$lambda.min
lasso_best_lambda
```

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

iii)

```
coef(glmnet(x, y, alpha=1, lambda=lasso_best_lambda))
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  0.023622904
## crim        -0.081992849
## zn           0.094717791
## indus        0.002619428
## chas         0.087341100
## nox         -0.175365927
## rm           0.312648954
## age         -0.011212120
## dis         -0.317143728
## rad          0.270168177
## tax         -0.207314714
## ptratio     -0.204052488
## black        0.102877803
## lstat       -0.428298373
```

d)

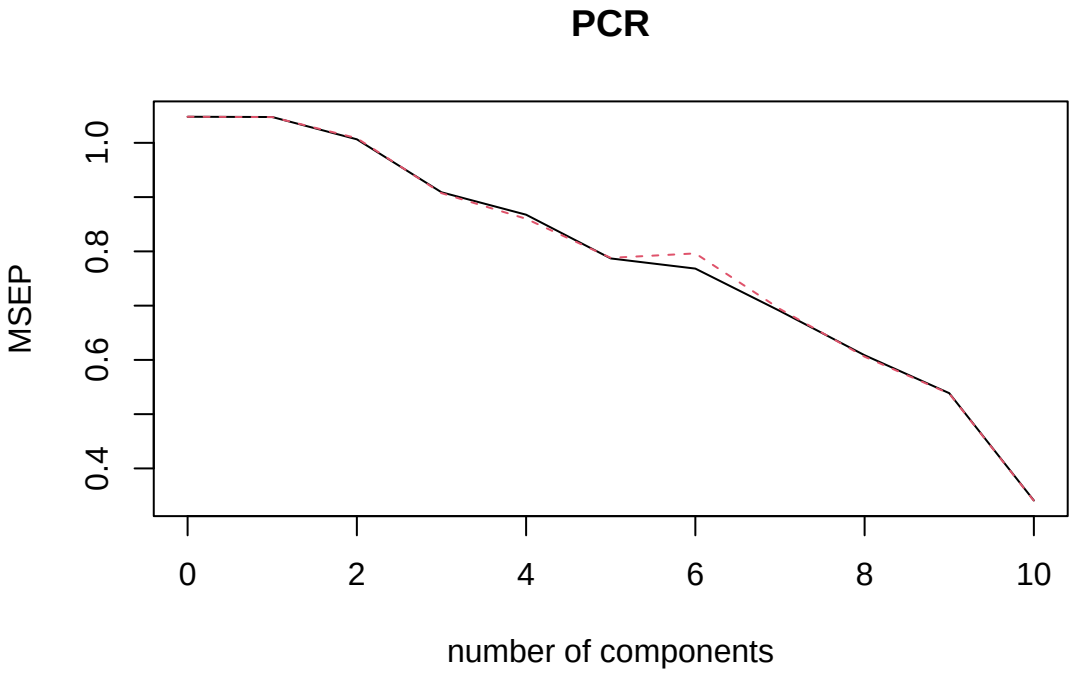
TRUE, FALSE, FALSE, TRUE

Problem 2

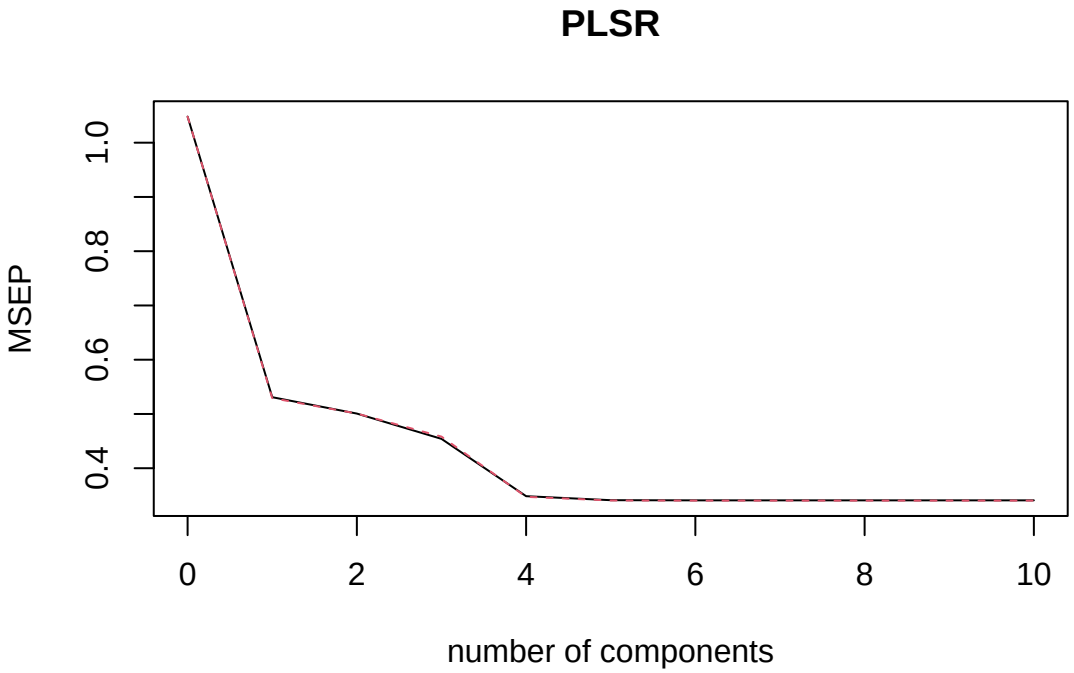
```
library(MASS)
set.seed(1)

# load a synthetic dataset
id <- "1CWZYfrLOrFdrIZ6Hv73e3xxt0SFgU4Ph" # google file ID
synthetic <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id))

# split into training and test sets
train.ind = sample(1:nrow(synthetic), 0.8 * nrow(synthetic))
synthetic.train = data.frame(synthetic[train.ind, ])
synthetic.test = data.frame(synthetic[-train.ind, ])

# show head(..)
# Y: response variable; X: predictor variable
#head(synthetic)
```

a)

```
pcr_fit <- pcr(Y~., data = synthetic.train, scale = TRUE, validation = "CV")
plsr_fit <- plsr(Y~., data = synthetic.train, scale = TRUE, validation = "CV")

#summary(plsr_fit)

validationplot(pcr_fit, val.type = "MSEP", main = "PCR")
```




```
validationplot(plsr_fit, val.type = "MSEP", main = "PLSR")
```



b)

We see that the mean squared error of prediction is is lower for PLSR compared to PCR for number of components < 10, which is to be expected since PLSR takes the response, Y, into account when fitting the model.

If we look at a regular linear regression model we see from the p-values of X4, X5, X6, X7, X8, X9 and X10 that they don't have a significant relationship with Y. This explains why PLSR reaches close to its minimum at 4 components, since X4 to X10 do little to effect to response.

For PCR the MSEP decreases for each additional component, since the predictors are not seen in relation to the response.

```
pcr_lm <- lm(Y~., data = synthetic.train, scale = TRUE, validation = "CV")
summary(pcr_lm)
```

```
##
## Call:
## lm(formula = Y ~ ., data = synthetic.train, scale = TRUE, validation = "CV")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.71228 -0.38071  0.03553  0.39454  1.67541
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.010162   0.020538   0.495    0.621
## X1           0.767846   0.020221  37.974   <2e-16 ***
## X2           1.405740   0.064383  21.834   <2e-16 ***
## X3          -1.295200   0.064779 -19.994   <2e-16 ***
## X4          -0.006841   0.036084  -0.190    0.850
## X5          -0.012883   0.034905  -0.369    0.712
## X6          -0.028925   0.036076  -0.802    0.423
## X7          -0.039057   0.035481  -1.101    0.271
## X8          -0.017084   0.034760  -0.491    0.623
## X9          -0.003340   0.036131  -0.092    0.926
## X10         -0.012034   0.036762  -0.327    0.743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5799 on 789 degrees of freedom
## Multiple R-squared:  0.6828, Adjusted R-squared:  0.6788
## F-statistic: 169.8 on 10 and 789 DF,  p-value: < 2.2e-16
```

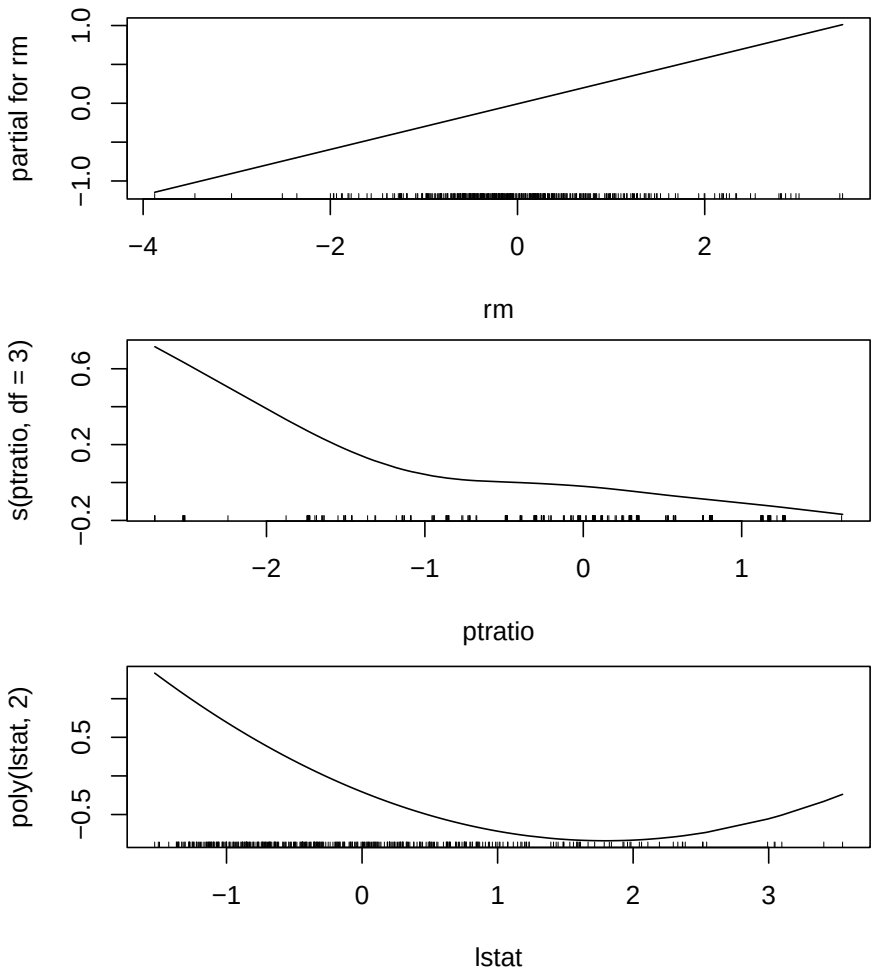
Problem 3

a)

TRUE, FALSE, FALSE, TRUE

b)

```
additive_model = gam(medv ~ rm + s(ptratio, df=3) + poly(lstat, 2), data=boston.train)
plot(additive_model)
```



Problem 4

a)

FALSE, TRUE, TRUE, TRUE

b)

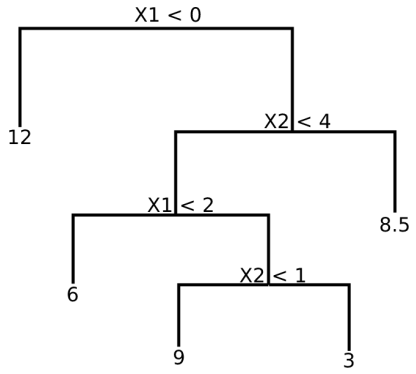


Figure 1: Tree

c)

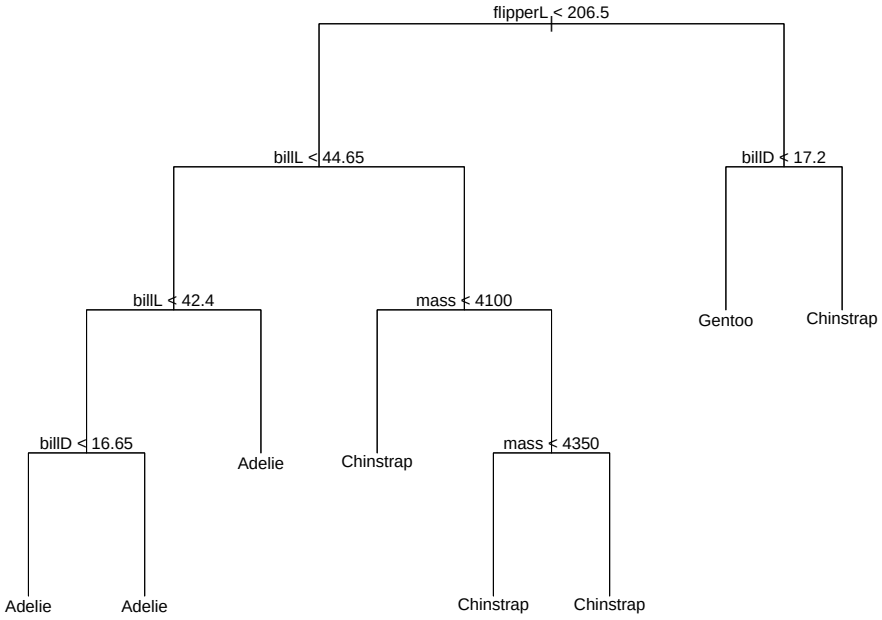
```
library(tidyverse)
library(palmerpenguins) # Contains the data set "penguins".
data(penguins)

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

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

# 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, ]
```

i)

```
penguin.tree = tree(formula=species ~ ., data=train, split='gini' )
summary(penguin.tree)
```

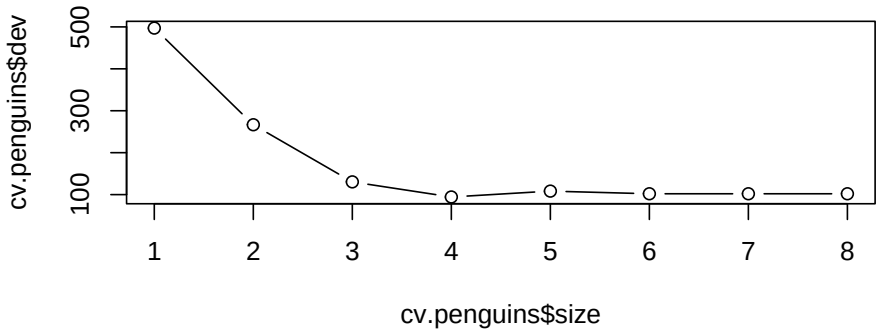
```
##
## Classification tree:
## tree(formula = species ~ ., data = train, split = "gini")
## Variables actually used in tree construction:
## [1] "flipperL" "billL"    "billD"    "mass"
## Number of terminal nodes:  8
## Residual mean deviance:  0.1869 = 42.06 / 225
## Misclassification error rate: 0.04292 = 10 / 233
```

```
plot(penguin.tree, type='uniform')
text(penguin.tree, pretty=0)
```



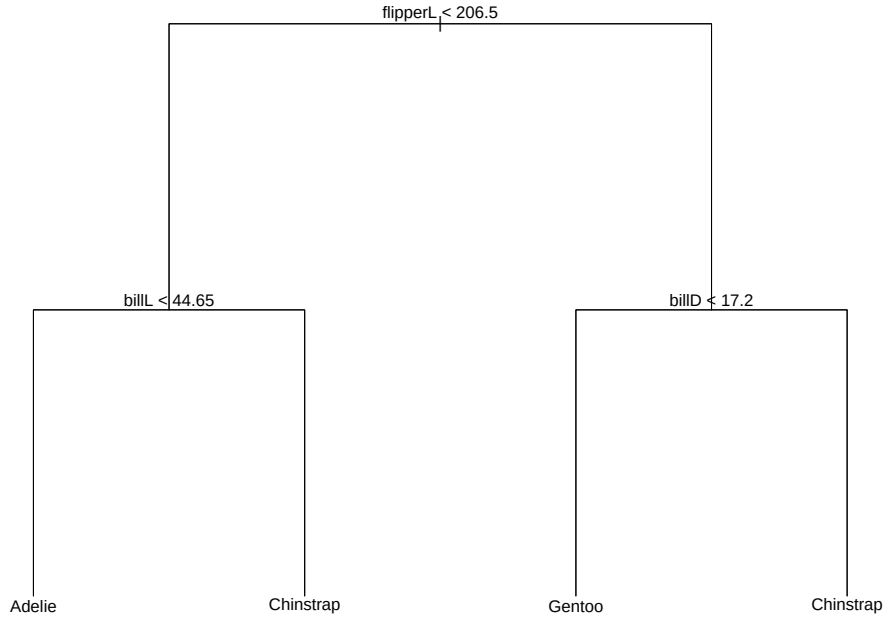
ii)

```
set.seed(123)
cv.penguins = cv.tree(penguin.tree, K=10)
#cv.penguins$dev
plot(cv.penguins$dev ~ cv.penguins$size, type='b')
```



iii)

```
prune.penguins = prune.tree(penguin.tree, best=4)
plot(prune.penguins, type='uniform')
text(prune.penguins, pretty=0)
```




```
tree.predict = predict(prune.penguins, test, type='class')
misclass = table(tree.predict, test$species)
misclass
```

```
##
## tree.predict Adelie Chinstrap Gentoo
##    Adelie        42         5      1
##    Chinstrap      0        15      0
##    Gentoo         0         0     37
```

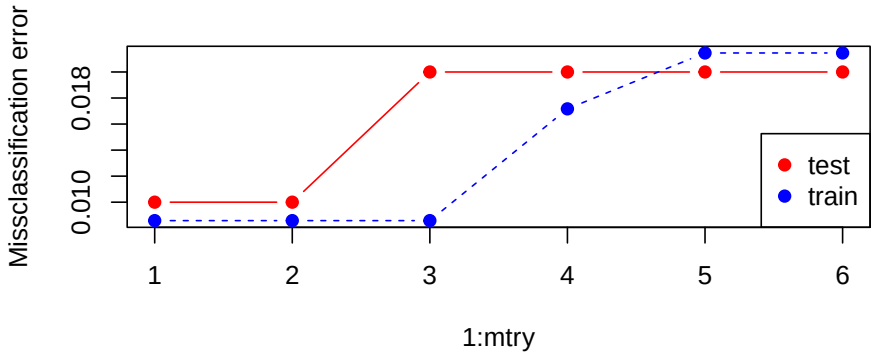
```
1-sum(diag(misclass))/sum(misclass)
```

```
## [1] 0.06
```

d)

Using random forest. Trying different choices for variable mtry, and plotting the misclassification errors.

```
set.seed(1001)

train.err = double(6)
test.err = double(6)

for(mtry in 1:6) {
  rf.penguins = randomForest(species ~ ., data=train, mtry=mtry, ntree=500)
  train.err[mtry] = rf.penguins$err.rate[500]

  rf.predict = predict(rf.penguins, newdata=test, type='class')
  misclass = table(rf.predict, test$species)
  misclass
  test.err[mtry] = 1-sum(diag(misclass))/sum(misclass)
}

matplot(1:mtry, cbind(test.err, train.err), pch=19, type='b', ylab='Missclassification error', col=c('r
legend('bottomright', legend=c('test', 'train'), pch=19, col=c('red','blue'))
```



We find that a good choice for mtry is 2, which also approximately corresponds to the square root of the number of covariates.

```
rf.penguins = randomForest(species ~ ., data=train, mtry=2, ntree=500)

rf.predict = predict(rf.penguins, newdata=test, type='class')
misclass = table(rf.predict, test$species)
misclass
```

```
##
## rf.predict  Adelie Chinstrap Gentoo
##   Adelie        42         2      0
##   Chinstrap      0        18      0
##   Gentoo         0         0     38
```

```
1-sum(diag(misclass))/sum(misclass)
```

```
## [1] 0.02
```

```
importance(rf.penguins)
```

```
##           MeanDecreaseGini
## island          17.3964364
## billL           52.2639236
## billD           25.0545916
## flipperL        37.1186075
## mass            14.5237520
## sex              0.9137075
```

We see that the two most influential variables are 'billL' and 'flipperL'.

Problem 5

a)

FALSE, FALSE, TRUE, TRUE

b)

i)

```
# Setting training and test data into data frame
svm.train = data.frame(x=train[,2:7], y = train[,1])
svm.test = data.frame(x=test[,2:7], y = test[,1])
```

```r
# Support vector classifier
svm.linear <- svm(species ~ ., data = svm.train, type = "C-classification", kernel = "linear", scale = F
#summary(svm.linear)

# Cross-validation of support vector classifier
CV.linear <- tune(svm, species~., data = svm.train, kernel = "linear", ranges=list(cost=c(0.001,0.01,0.1
#summary(CV.linear)

best.linear = CV.linear$best.model
#summary(best.linear)

svm.radial <- svm(species ~ ., data = svm.train, type = "C-classification", kernel = "radial", scale = F
#summary(svm.radial)

CV.radial <- tune(svm, species~., data = svm.train, scale = FALSE, kernel = "radial", ranges=list(cost=c
#summary(CV.radial)

best.radial = CV.radial$best.model
#summary(best.radial)
```

We see that the for the linear boundary the optimal cost parameter is 0.1, which gives an error of 0.004347826. For the radial boundary the best cost and gamma parameters are 1e06 and 1e-07 respectively, which gives an error of 0.008514493.

ii)

```r
# Confusion tables
pred.linear = predict(best.linear, svm.test)
table(predict=pred.linear, truth=svm.test[,7])
```

```
##            truth
## predict     Adelie Chinstrap Gentoo
##   Adelie        41         0      0
##   Chinstrap      1        20      0
##   Gentoo         0         0     38
```

```r
pred.radial = predict(best.radial, svm.test)
table(predict=pred.radial, truth=svm.test[,7])
```

```
##            truth
## predict     Adelie Chinstrap Gentoo
##   Adelie        42         1      0
##   Chinstrap      0        19      0
##   Gentoo         0         0     38
```

From the confusion tables above we see that the misclassification error of the support vector classifier is 0. The error rate of the support vector machine is 0.01.

iii)

The high cost in the support vector machine implies a low tuning parameter, C. This could mean that our model is underfitted. The low value of gamma in the support vector machine indicates that the model may be too constrained.

With these factors in mind we prefer the support vector classifier, which has both a lower training and test error.

Problem 6

a)

i)

We can see that the features 'Logged.GDP.per.capita', 'Social.support' and 'Healthy.life.expectancy' all point in the same direction, suggesting that these are corrolated. We also see that 'Freedom.to.make.life.choices' and 'Perceptions.of.corruption' point in opposite directions, meaning they are negatively corrolated, i.e. a country with high perceptions of corruption will have lower freedom to make life choices.

ii)

Afghanistan can be considered an outlier

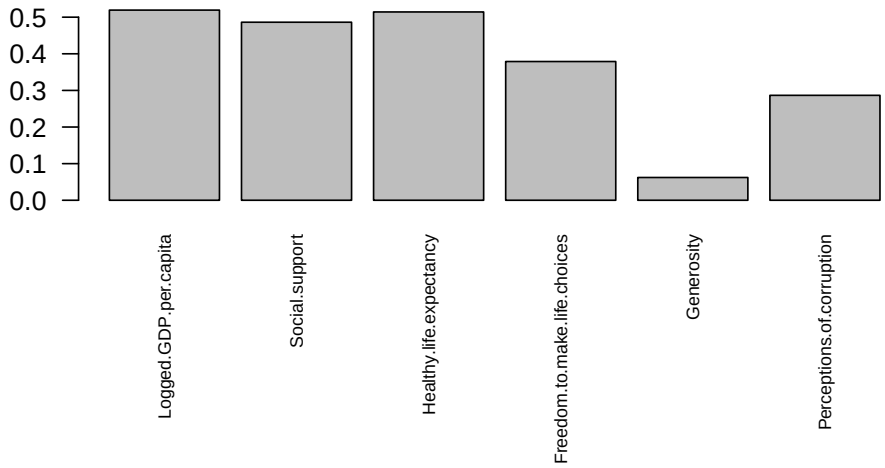
b)

```
# load a synthetic dataset
id <- "1NJ1SuUBebl5P8rMSIwm_n3S8a7K43yP4" # google file ID
happiness <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id),fileEncoding="UTF
#colnames(happiness)
cols = c('Country.name',
         'Ladder.score',  # happiness score
         'Logged.GDP.per.capita',
         'Social.support',
         'Healthy.life.expectancy',
         'Freedom.to.make.life.choices',
         'Generosity',  # how generous people are
         'Perceptions.of.corruption')
# We continue with a subset of 8 columns:
happiness = subset(happiness, select = cols)
rownames(happiness) <- happiness[, c(1)]
# And we creat an X and a Y matrix
happiness.X = happiness[, -c(1, 2)]
happiness.Y = happiness[, c(1, 2)]
happiness.XY = happiness[, -c(1)]
# scale
happiness.X = data.frame(scale(happiness.X))
#str(happiness)
```

```
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)
```

i)

```
par(mar=c(13,3,1,1))
barplot(abs(data.frame(pca_mat$rotation)$PC1), names.arg=cols[-c(1,2)], las=2, cex.names=0.7)
```



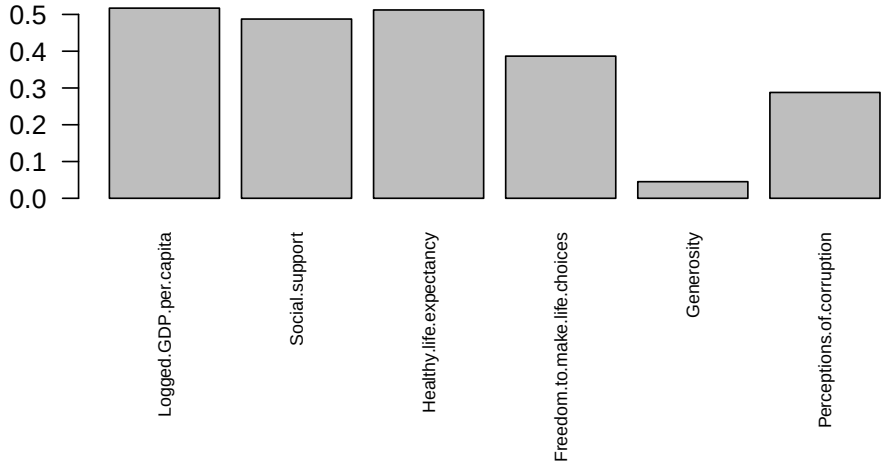
ii)

```
?plsr()
plsr_model = plsr(Ladder.score ~ ., data=happiness.XY, scale=TRUE)
summary(plsr_model)
```

```
## Data:    X dimension: 149 6
##  Y dimension: 149 1
## Fit method: kernelpls
## Number of components considered: 6
## TRAINING: % variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                51.87    68.64    84.48    88.10    94.12   100.00
## Ladder.score     75.10    75.50    75.55    75.58    75.58    75.58
```

iii)

```
par(mar=c(13,3,1,1))
barplot(abs(plsr_model$loadings[, c('Comp 1')]), las=2, cex.names=0.7)
```



iv)

The three most important predictors are 'Logged.GDP.per.capita', 'Social.support', and 'Healthy.life.expectancy'.

c)

FALSE, FALSE, TRUE, TRUE

d)

i)

```
set.seed(123)
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)

Interpretation 1: Countries within a cluster have happiness scores similar to eachother.

Interpretation 2: