TMA4268-Project2

Magnus Grytten, Jokhongir Khayrullaev & Rashad Naghiyev

```
# Libraries used throughout the exercise
library("ggplot2")
library("tidyverse")
library("palmerpenguins")
library("GGally")
library("MASS")
library("caret")
library("leaps")
library("glmnet")
library("pls")
library("gam")
library("e1071")
library("tree")
library("randomForest")
library("ggfortify")
```

Problem 1

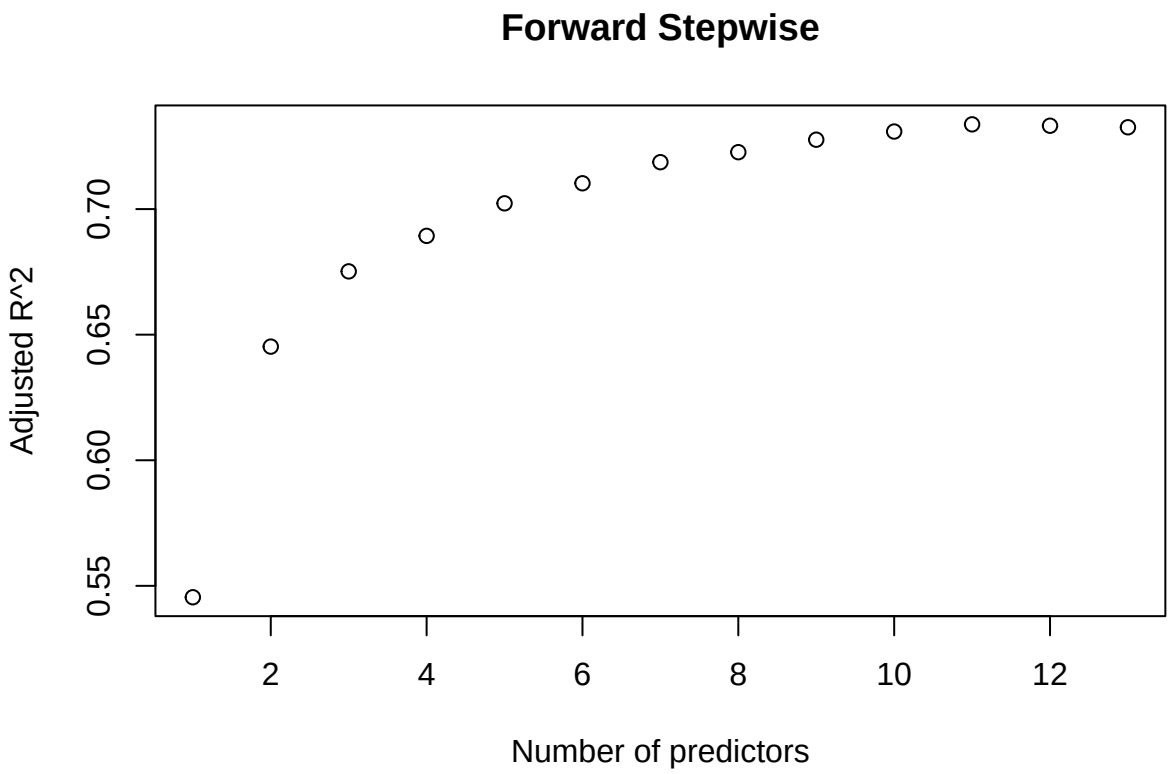
```
set.seed(1)
# pre-processing by scaling NB! Strictly speaking, pre-processing should be done
# on a training set only and it should be done on a test set with statistics of
# the pre-processing from the training set. But, we're preprocessing the entire
# dataset here for convenience.
boston <- scale(Boston, center = T, scale = T)
# split into training and test sets
train.ind = sample(1:nrow(boston), 0.8 * nrow(boston))
boston.train = data.frame(boston[train.ind, ])
```

a)

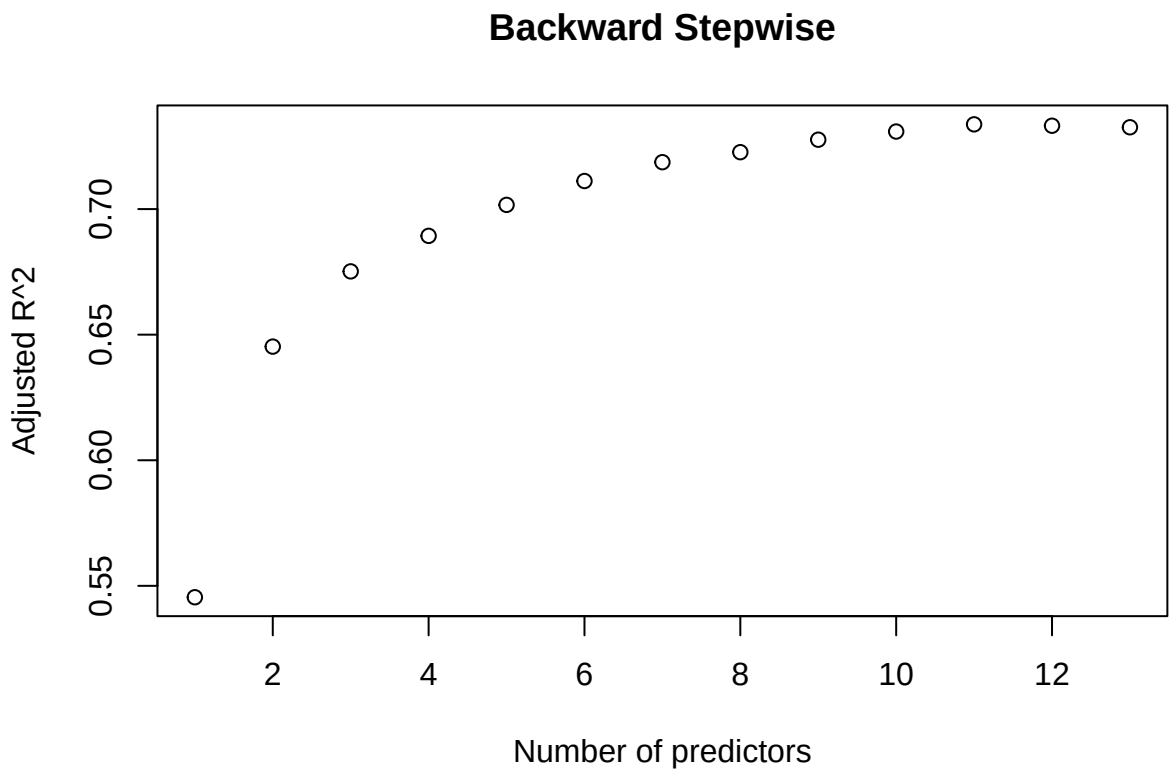
```
res_forward <- regsubsets(medv~. ,data = boston.train, nvmax = 13, method = "forward")
res_forward_sum <- summary(res_forward)

res_backward <- regsubsets(medv~. ,data = boston.train, nvmax = 13, method = "backward")
res_backward_sum <- summary(res_backward)

plot(res_forward_sum$adjr2, xlab = "Number of predictors",
     ylab = "Adjusted R^2", main = "Forward Stepwise")
```




```
plot(res_backward_sum$adjr2, xlab = "Number of predictors",
     ylab = "Adjusted R^2", main = "Backward Stepwise")
```



b)

```
coefs <- res_forward_sum$which[4,-1]
print(names(coefs[coefs == TRUE]))
```

```
## [1] "rm"      "dis"     "ptratio" "lstat"
```

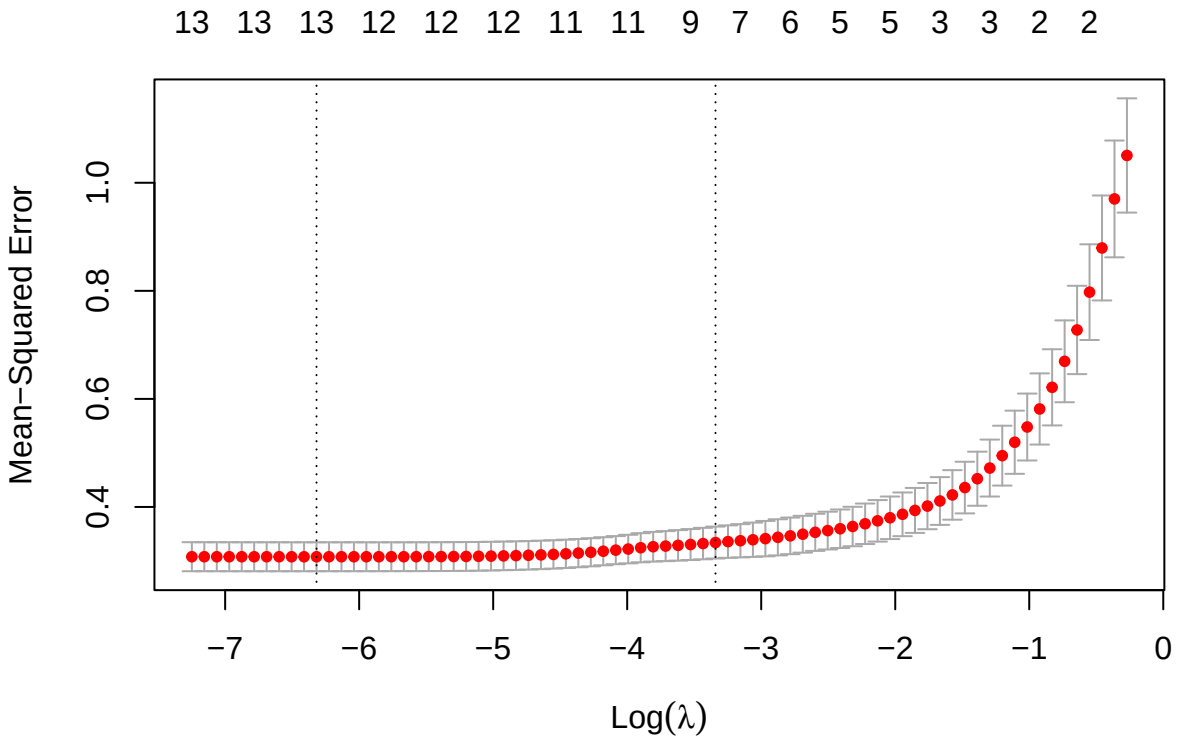
From our previous results we find that the four best predictors from forward stepwise are, "rm", "dis", "ptratio" and "lstat". These are not however necessarily the best overall four predictors we would find from best subset selection.

c)

(i)

```
x <- model.matrix(medv~.,-1,data = boston.train)
y <- boston.train$medv

cv_lasso <- cv.glmnet(x,y,nfolds = 5)
plot(cv_lasso)
```



(ii)

```
best_lambda <-cv_lasso$lambda.min
print(best_lambda)
```

```
## [1] 0.001803259
```

The lambda with the lowest MSE is 0.001803259. The log of this is -6.31816, which is what is what we see in the previous plot.

(iii)

```
fit_lasso <- glmnet(x,y)
coef(fit_lasso,s = best_lambda)
```

```
## 15 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.023606403
## (Intercept)  .
## crim        -0.082544050
## zn           0.095554758
## indus        0.006767917
## chas         0.087275698
```

```
## nox         -0.176974474
## rm           0.312664541
## age         -0.011672213
## dis         -0.317825160
## rad          0.274929406
## tax         -0.212678382
## ptratio     -0.204780585
## black        0.103330382
## lstat       -0.428624759
```

Above you can find a print out of the fitted coefficients for lambda = 0.001803259.

d)

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

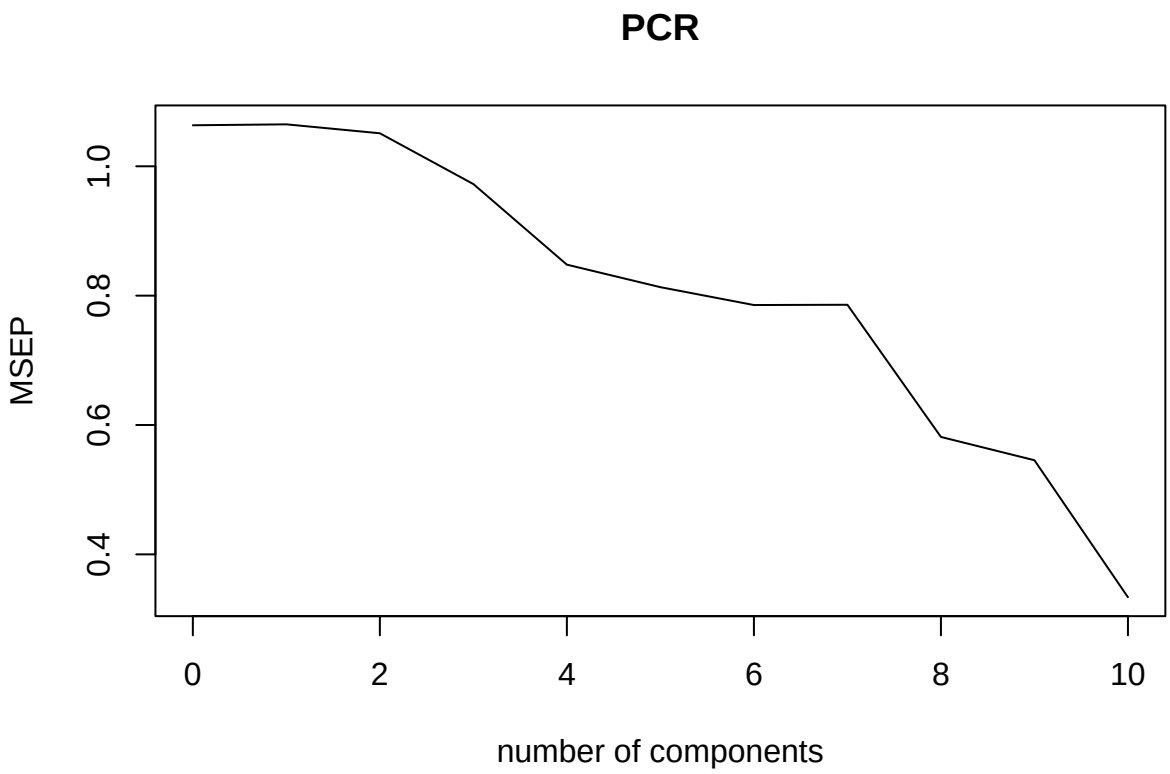
Problem 2

```
# 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, ])
```

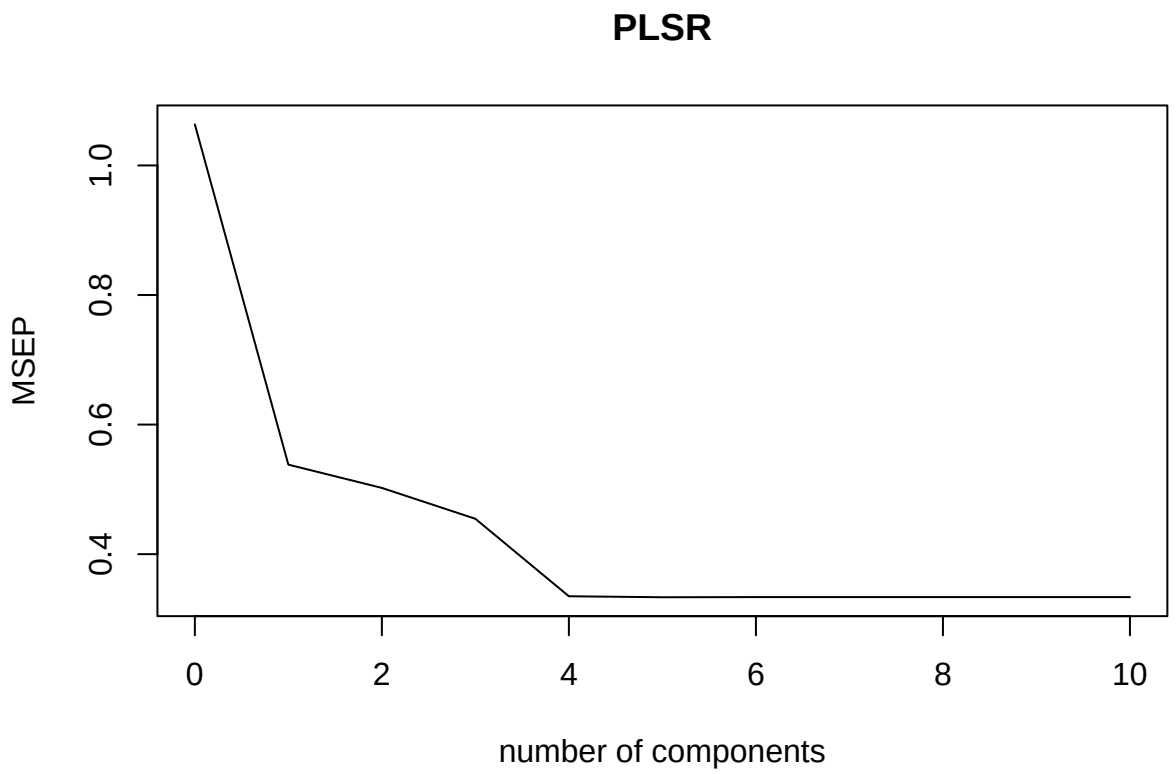
a)

```
pcr_fit = pcr(Y~., data = synthetic.train, scale = TRUE, validation = "none")
pls_fit = plsr(Y~., data = synthetic.train, scale = TRUE, validation = "none")

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



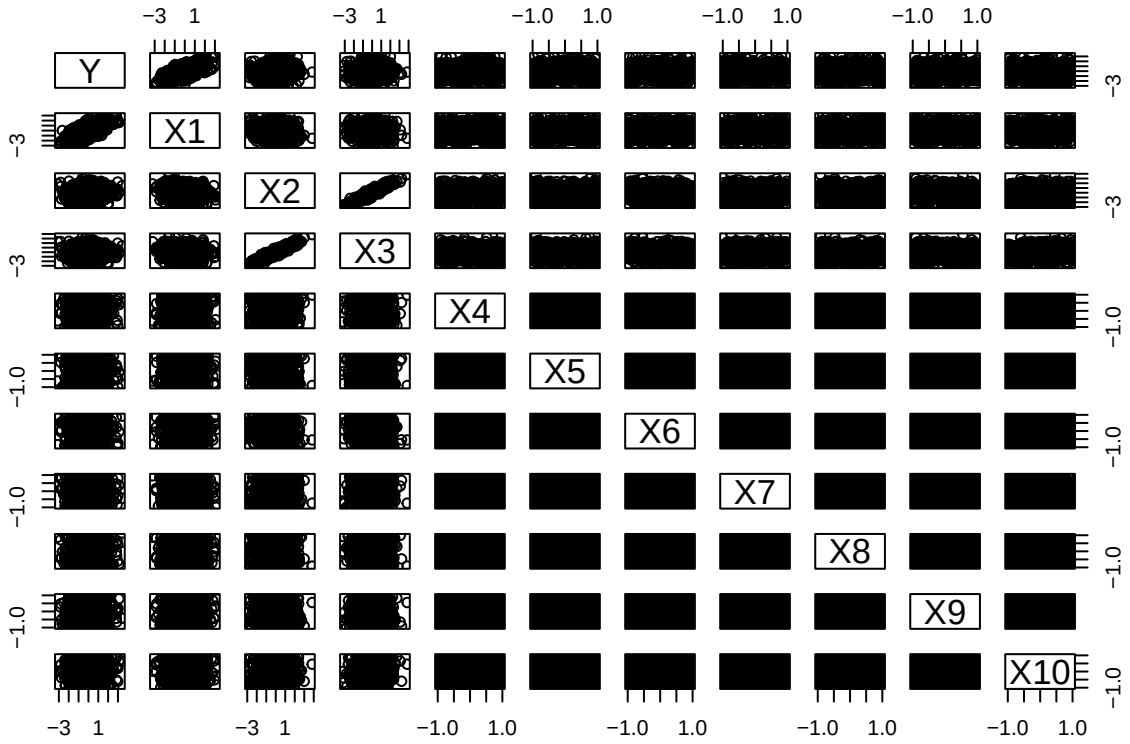

```
validationplot(pls_fit, val.type = "MSEP", newdata = synthetic.test, main = "PLSR")
```



b)

Seeing as PCR and PLSR tend to behave similarly and have similar performance it is somewhat surprising that PLSR outperforms PCR in this case. However by looking at the dataset,

```
plot(synthetic)
```



we find that X4 to X10 seems to all be uniformly distributed between -1 and 1 and uncorrelated to the other predictors. We also see that Y and X1, and X3, X4 are correlated, while the rest of the predictors are weakly- or uncorrelated. Since there are so many of the predictors that are just noise it now makes more sense that PLSR outperforms PCR since it is a supervised method, that attempts to find directions that explain the response. As opposed to PCR where in this case many of the principal components end up being weakly correlated to the repose due to many of the predictors. being noise.

Problem 3

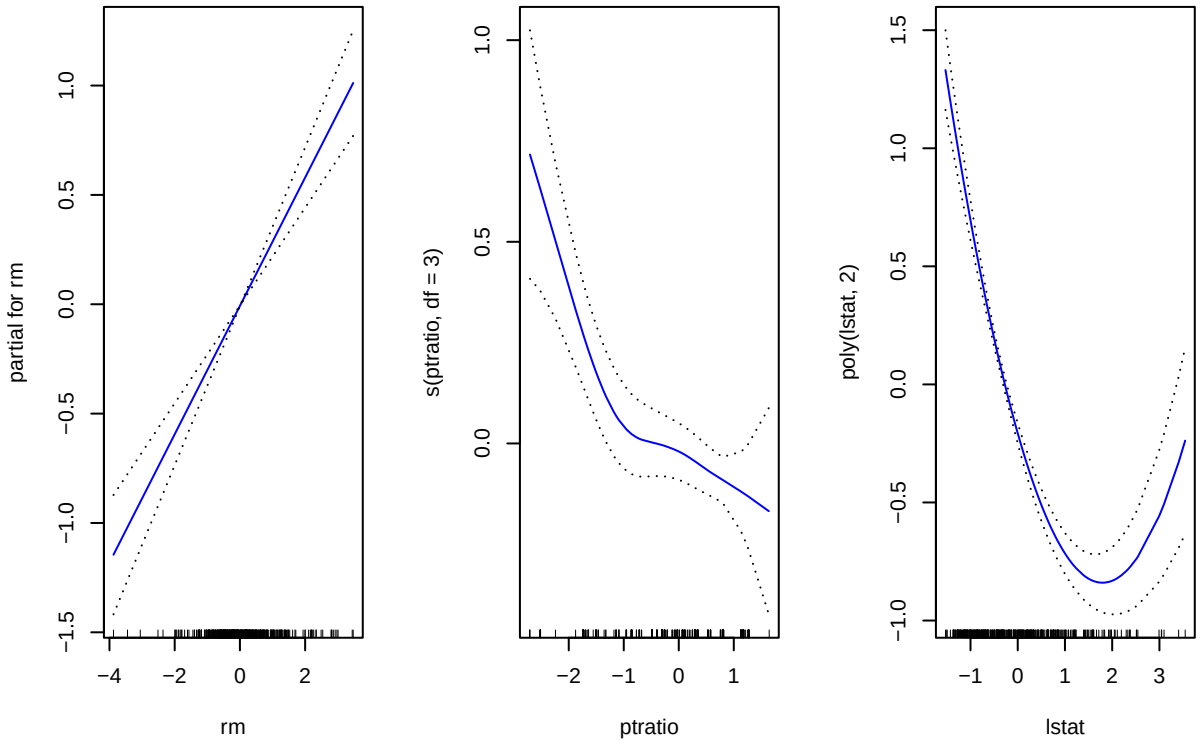
a)

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

b)

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


par(mfrow = c(1,3))
plot(fit, se=T, col='blue')
```



Problem 4

a)

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

b)

```
#knitr::include_graphics("tree.png")
```

c)

```
par(mfrow=c(1,1))
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()
```

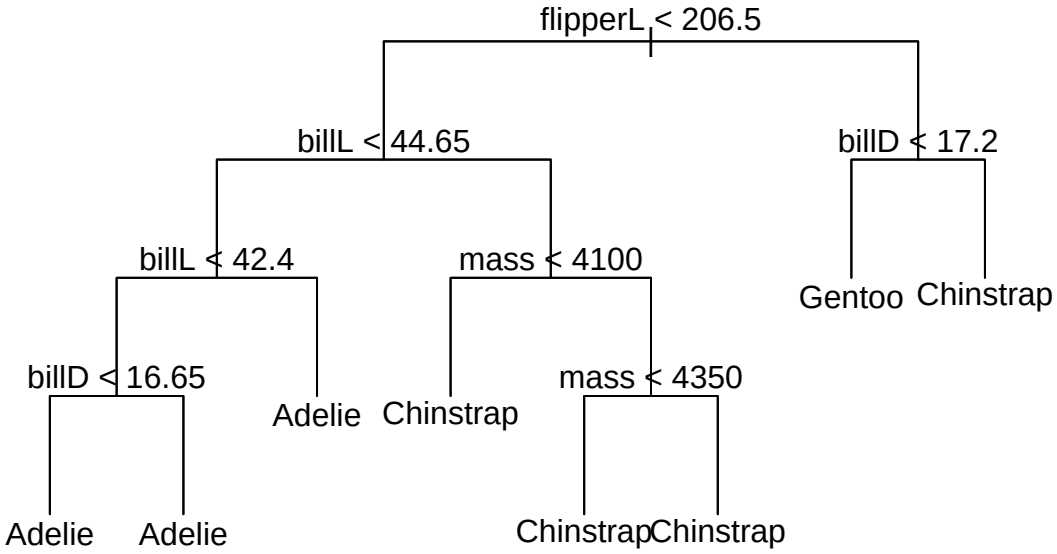
```r
# We do not want 'year' in the data (this will not help for future predictions)
Penguins_reduced <- Penguins_reduced[, -c(8)]
set.seed(4268)
# 70% of the sample size for training set
training_set_size <- floor(0.7 * nrow(Penguins_reduced))
train_ind <- sample(seq_len(nrow(Penguins_reduced)), size = training_set_size)
train <- Penguins_reduced[train_ind, ]
test <- Penguins_reduced[-train_ind, ]
```

i)

```r
species_treeG = tree(species ~ ., train, split='gini')
summary(species_treeG)
```

```
##
## 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
```

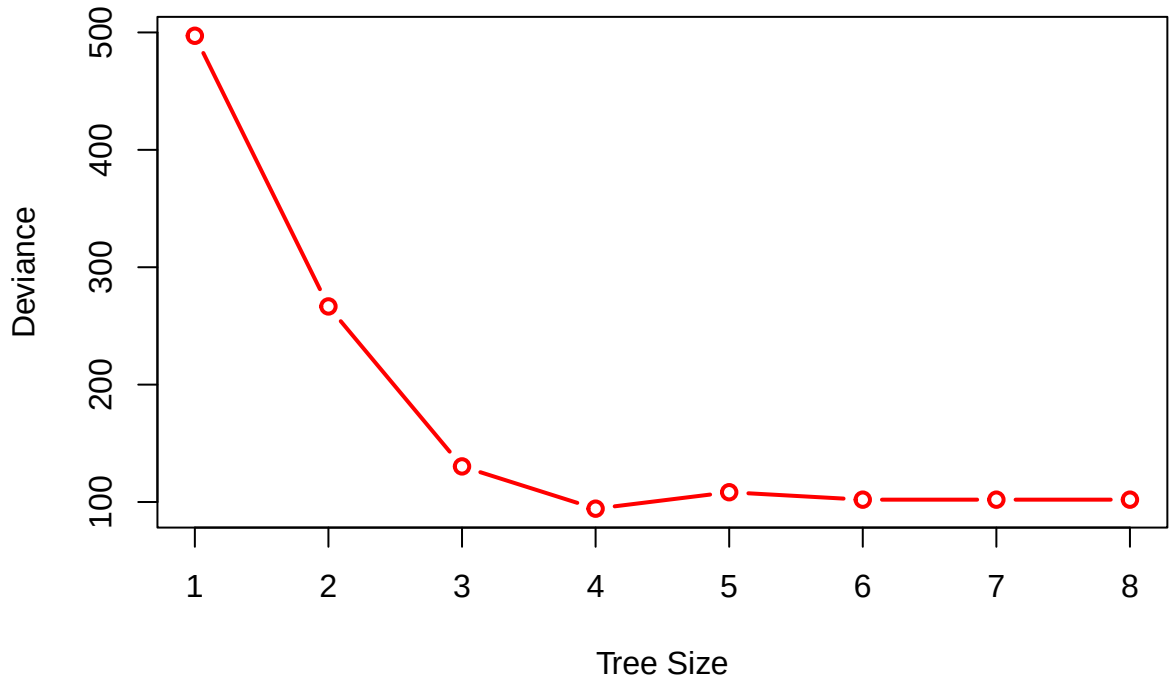
```r
plot(species_treeG, type="uniform")
text(species_treeG, pretty=0)
```



ii)

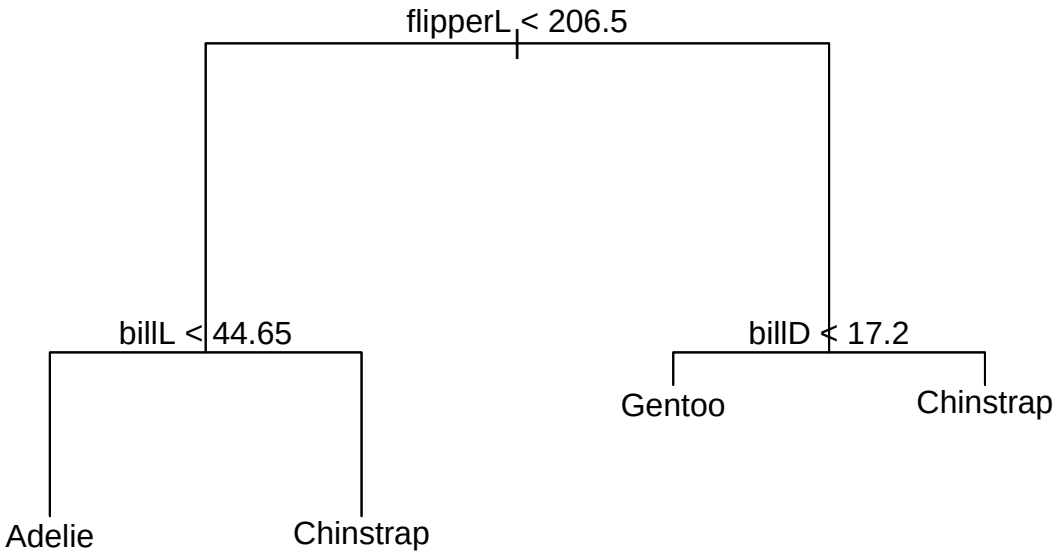
```
set.seed(123)

cv.species <- cv.tree(species_treeG, K = 10)
plot(cv.species$dev ~ cv.species$size, type = "b", lwd = 2, col = "red",
     xlab = "Tree Size", ylab = "Deviance")
```



From CV plot, we can see that the best tree size is 4. So, we use this size to prune our tree.

```
prune.species <- prune.tree(species_treeG, best = 4)
plot(prune.species)
text(prune.species, pretty = 0)
```



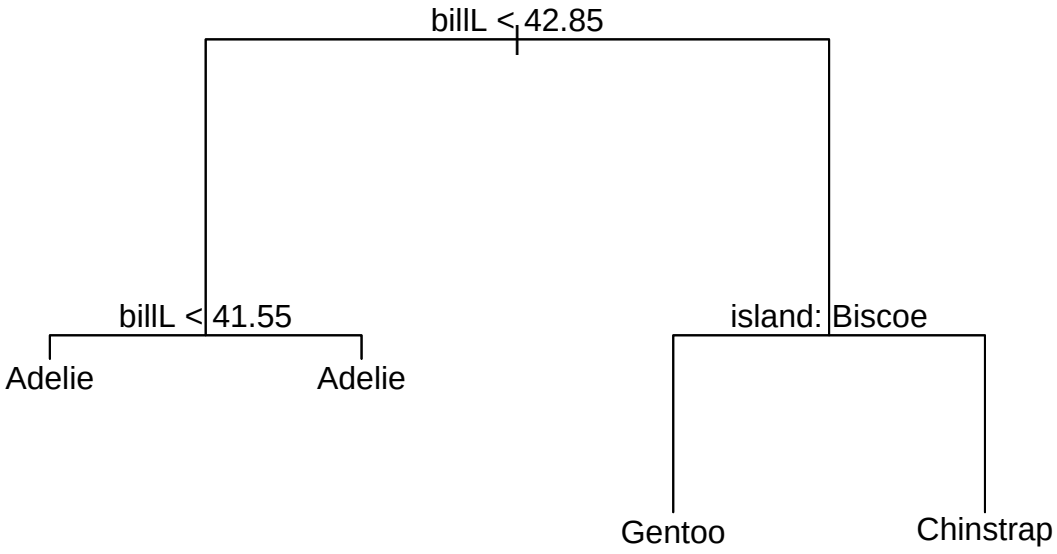
This is pruned tree using training dataset.

iii) Now we can use our test dataset.

```
species_treeG_test = prune.tree(tree(species ~ ., test,  split='gini'), best=4)
summary(species_treeG_test)
```

```
##
## Classification tree:
## tree(formula = species ~ ., data = test, split = "gini")
## Variables actually used in tree construction:
## [1] "billL"  "island"
## Number of terminal nodes:  4
## Residual mean deviance:  0.09899 = 9.503 / 96
## Misclassification error rate: 0.02 = 2 / 100
```

```
plot(species_treeG_test)
text(species_treeG_test, pretty = 0)
```



And the misclassification:

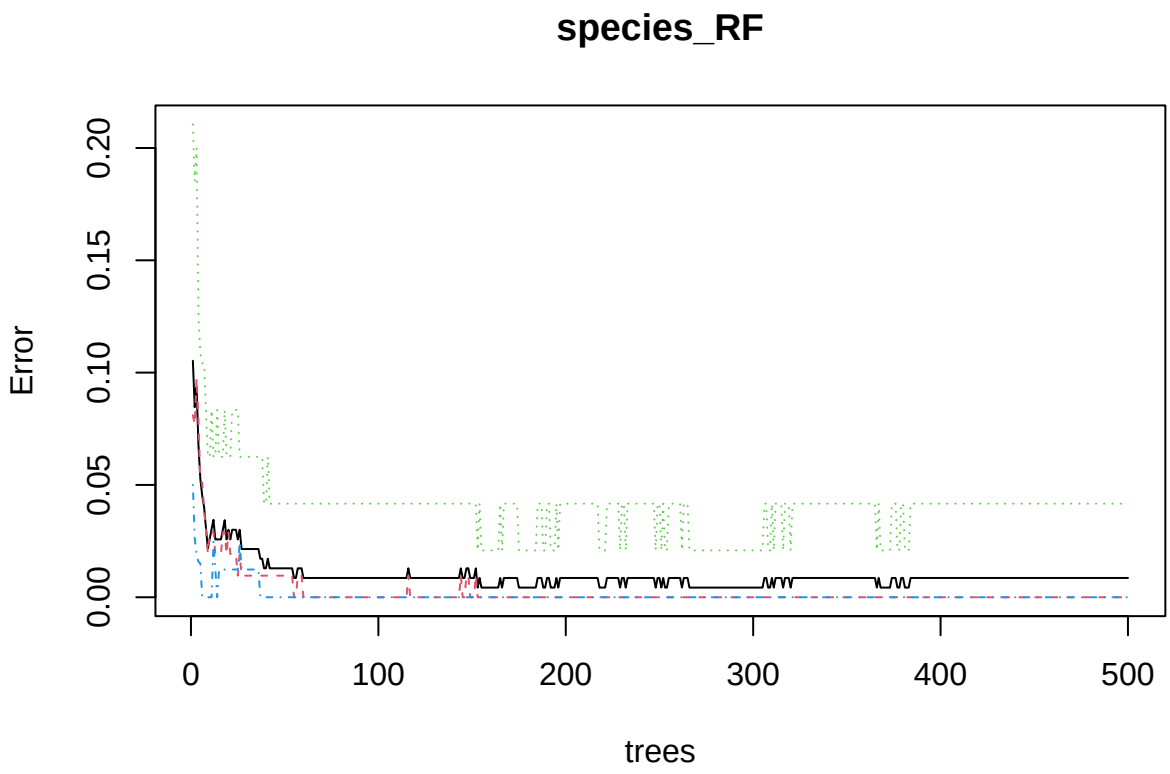
```
print(paste( "Misclassification error -", 0.02))
```

```
## [1] "Misclassification error - 0.02"
```

d)

Now, we try the same idea with more complex method - random forests.

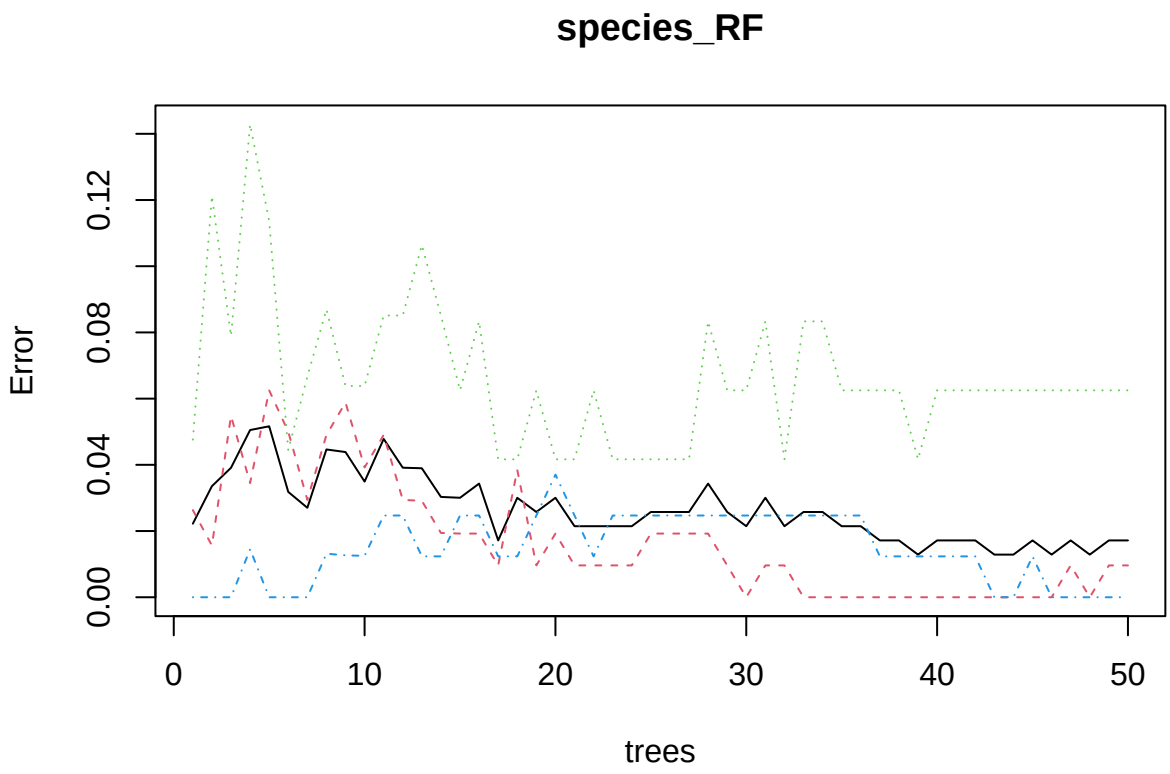
```
species_RF = randomForest(species ~ ., data = train,
                          mtry = 2, importance = TRUE)
plot(species_RF)
```



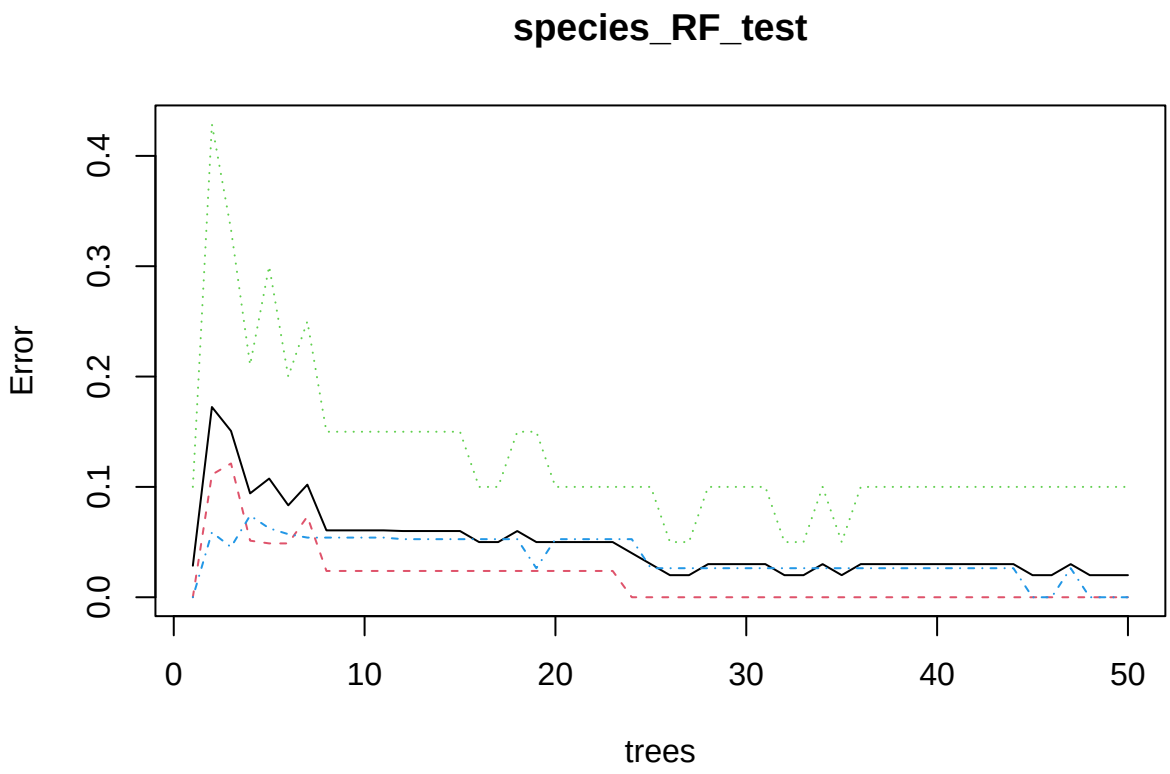
mtry - split number is found by taking square root of number of predictors ($\sqrt{6} \approx 2$)

From the graph, we can say that the best tree size is approximately 50.

```
species_RF = randomForest(species ~ ., data = train,
                          mtry = 2, ntree =50, importance = TRUE)
plot(species_RF)
```




```
species_RF_test = randomForest(species ~ ., data = test,
                           mtry = 2, ntree =50, importance = TRUE)
plot(species_RF_test)
```




```
yhat.rf = predict(species_RF_test, newdata = test)
misclass.rf = table(yhat.rf, test$species)
print(misclass.rf)
```

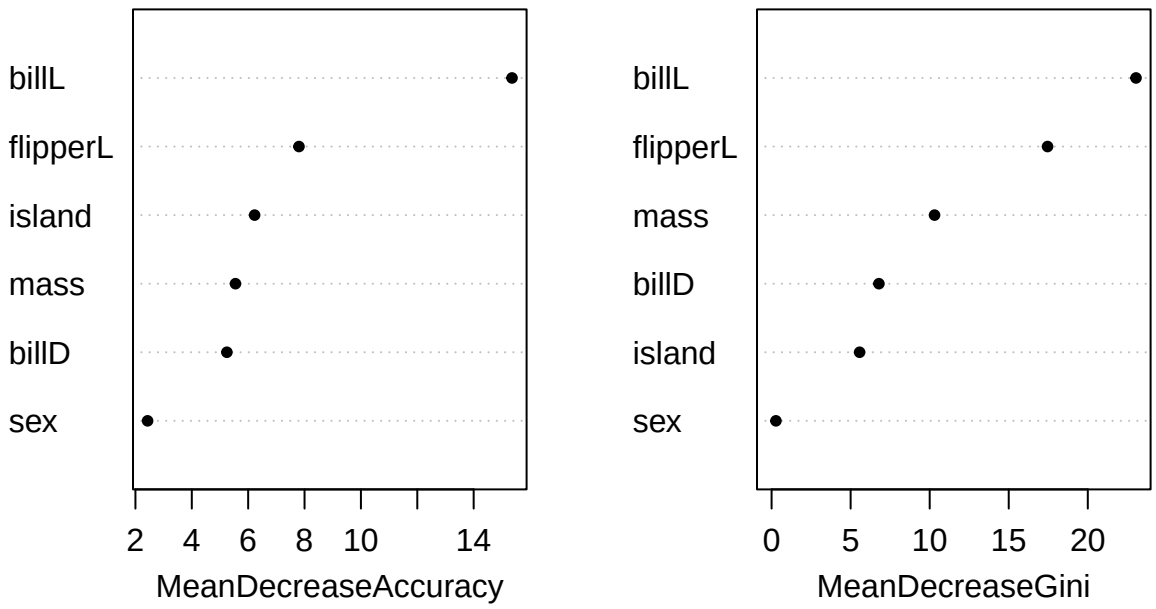
```
##
## yhat.rf     Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

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

```
## [1] 0
```

Importance

```
varImpPlot(species_RF_test, pch = 20, main = "")
```



From the importance graph, it is clearly seen that variables such as bill and flipperL are most influential.

Problem 5

a)

a.i. False a.ii. True a.iii. False a.vi. True

b)

b.i.

We begin solving the problem by loading the "penguins" data and deviding it into test and train datasets.

```
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, ]
```

Once the data is loaded, we continue by fitting linear svm model into our data for predicting penguin species from their island, billL, billD, flipperL, mass, sex first.

```
train_model_lin<-svm(species~.,data=train,kernel="linear")
```

The linear svm model's error rate depends on chosen cost value, we perform cross-validation on training data for defining the best cost value resulting in lowest error rate.

```
tune_tml<-tune(svm, species~., data=train, kernel="linear",
               ranges = list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100,200)))
```

We can see the best model's cost parameter value using the command below

```
best_tml_model_par<-tune_tml$best.parameters
print(best_tml_model_par)
```

```
##   cost
## 4    1
```

and the error rate of the best model can be seen running the line below.

```
#summary(tune_tml)
```

As one can see the best linear svm model is with cost=1 and it has error=0.

Once we are done with selecting the best linear svm model we start selecting radial svm model. The first step is building radial svm model

```
train_model_rad<-svm(species~.,data=train,kernel="radial")
```

Secondly we perform cross validation of the model in order to choose optimal
gamma and cost parameters in similar principle as for linear svm model.

```
tune_tmr<-tune(svm, species~., data=train, kernel="radial",
               ranges = list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100,200),
                             gamma=c(0.5,1,2,3,4,5,6)))
```

We can see that the best model parameters are

```
best_tmr_model_par<-tune_tmr$best.parameters
print(best_tmr_model_par)
```

```
##   cost gamma
## 4    1   0.5
```

The error rate of best radial svm model can be seen via running line below

```
#summary(tune_tmr)
```

As one can see the best radial model is with cost=5, gamma=0.5 and it has er- ror=0.004166667.

We save the best linear svm model and best radial svm model for further usage here

```
best_tml_model<-tune_tml$best.model
best_tmr_model<-tune_tmr$best.model
```

b.ii.

We will construct confusion tables and calculate missclassification error rate for the best linear and radial svm models defined above using the "penguin"s test dataset.

We will start with the linear svm model. We will predict penguins species using the test data, after that we will compare our predictions with truth given in test dataset.

```
#Predictions using  the linear  svm model and constructing its confusion table
linpred <- predict(best_tml_model,test)
linpred_table<-table(predict=linpred, truth=test$species)
print(linpred_table)
```

```
##            truth
## predict     Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

Once the confusion table is calculated, we calculate misclassification error for this model.

```
linpred_missclass_error=1-sum(diag(linpred_table))/sum(linpred_table)
paste0("Linear vsm model's misclassification error is: ", linpred_missclass_error*100, "%")
```

```
## [1] "Linear vsm model's misclassification error is: 0%"
```

We can see that linear svm model has o % misclassification error which is pretty good.

We perform the same procedure for radial svm model.

```
radpred <- predict(best_tmr_model,test)
radpred_table<-table(predict=radpred, truth=test$species)
radpred_missclass_error=1-sum(diag(radpred_table))/sum(radpred_table)
print(radpred_table)
```

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

```
paste0("Radial vsm model's misclassification error is: ", radpred_missclass_error*100, "%")
```

```
## [1] "Radial vsm model's misclassification error is: 1%"
```

Radial svm model has 1 % misclassification error, that is not bad although worse comparing to linear svm model with 0% misclassification error.

b.iii.

In this case I would prefer linear vsm model other than radial vsm model. Becau- se, it has lower misclassification error and there is no big differance in the models running time.

Problem 6

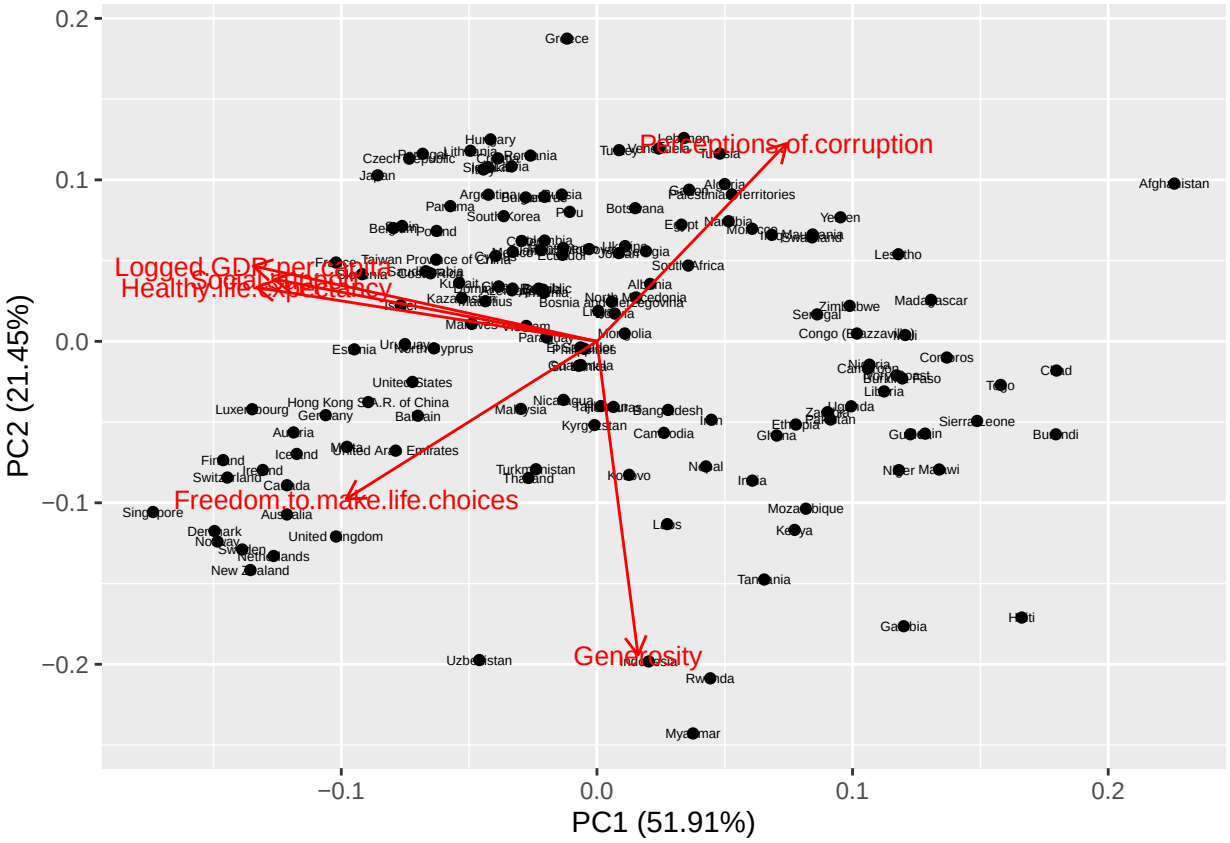
a)

We start by loading the world-happiness-report-2021 dataset in order to asnwer the problem a) questions.

```
id <- "1NJ1SuUBebl5P8rMSIwm_n3S8a7K43yP4" # google file ID
happiness <- read.csv(
  sprintf("https://docs.google.com/uc?id=%s&export=download", id),
  fileEncoding="UTF-8-BOM")
colnames=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=str(happiness)
```

```
## 'data.frame':    149 obs. of  8 variables:
##  $ Country.name                : chr  "Finland" "Denmark" "Switzerland" "Iceland" ...
##  $ Ladder.score                : num  7.84 7.62 7.57 7.55 7.46 ...
##  $ Logged.GDP.per.capita       : num  10.8 10.9 11.1 10.9 10.9 ...
##  $ Social.support              : num  0.954 0.954 0.942 0.983 0.942 0.954 0.934 0.908 0.948 0.934 ..
##  $ Healthy.life.expectancy     : num  72 72.7 74.4 73 72.4 73.3 72.7 72.6 73.4 73.3 ...
##  $ Freedom.to.make.life.choices: num  0.949 0.946 0.919 0.955 0.913 0.96 0.945 0.907 0.929 0.908 ...
##  $ Generosity                  : num  -0.098 0.03 0.025 0.16 0.175 0.093 0.086 -0.034 0.134 0.042 ..
##  $ Perceptions.of.corruption   : num  0.186 0.179 0.292 0.673 0.338 0.27 0.237 0.386 0.242 0.481 ...
```

```
library(ggfortify)
library(ggplot2)
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 = 3.5, label = T, label.size = 1.8)
```



a.i.

We can see from the given plot there are 6 loading directions Generosity, Freedom to make life choices, Perception of corruption, Healthy life expec- tancy, Social support, Logged GDP per capita. We can see that Freedom to make life choices is controlled by decrease in both PC1, PC2 variables almost evenly, since the vector showing characteristics direction starts when PC1, PC2
are at zero and it is pointing at the angle 45 degrees.

On the other hand the Generosity is mostly controlled by PC2. Its vector has the same origin, however the vector is almost vertical that means PC1 has al- most no effect on it.

a.ii.

We can consider Afganistan as outlier.

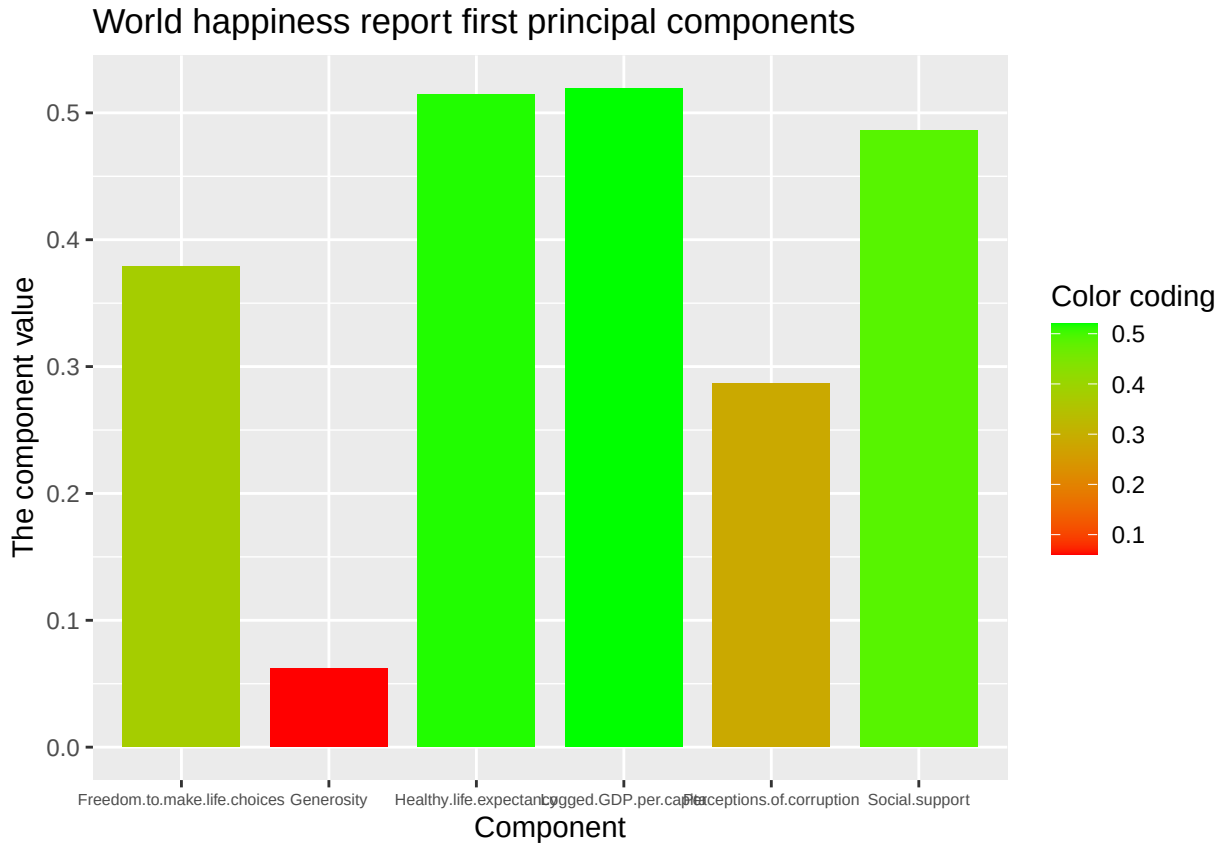
b)

b.i.

The graphical representation of absolute values of first principal components (PC1) is generated by the code below.

```
PC1_values = unname(abs(subset(pca_mat$rotation, select = PC1)))
PC1_rownames = rownames((subset(pca_mat$rotation, select = PC1)))
PC1 = data.frame(PC1_rownames,PC1_values)

plot<-ggplot(PC1,aes(x=PC1_rownames, y=PC1_values, fill=PC1_values))+
  geom_bar(stat='identity', position = "dodge", width = 0.8, show.legend=NA)
plot<-plot+theme(axis.text.x=element_text(size=5.5))
plot<-plot+ggtitle("World happiness report first principal components")
plot+xlab("Component")+ylab("The component value")+
  labs(fill="Color coding")+scale_fill_gradient(low="red", high="green")
```



b.ii.

We fit PLSR on happiness.XY with response of Ladder.score (=happiness score) while using remaining variables as predictors.

```
set.seed(1)
library(pls)
To_predict = happiness.XY$Ladder.score
cols2 = c('Logged.GDP.per.capita',
          'Social.support',
          'Healthy.life.expectancy',
          'Freedom.to.make.life.choices',
          'Generosity', # how generous people are
          'Perceptions.of.corruption')
```

```
For_predicting = subset(happiness.XY, select = cols2)
plsr_model <-plsr(To_predict~.,data=For_predicting,scale=T)
```

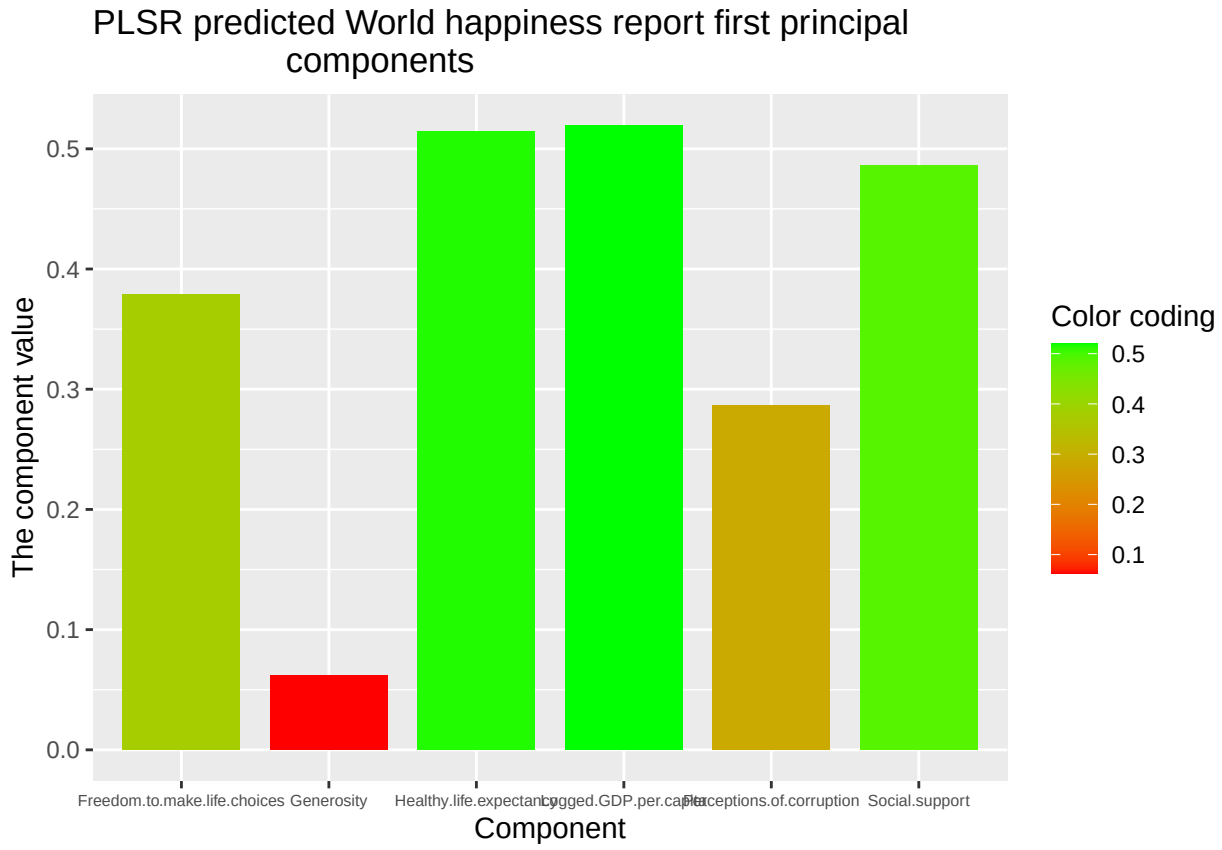
b.iii.

Here we plot predicted by PLSR above absolute values of first principal compo- nents (happiness).

```
plsr_results=plsr_model$loadings[,c("Comp 1")]

plsr_results_values = unname(abs(plsr_results))
plsr_results_rownames = names(plsr_results)
plsr_results_to_plot = data.frame(plsr_results_rownames,plsr_results_values)

plot2<-ggplot(plsr_results_to_plot,aes(x=plsr_results_rownames,
                                       y=plsr_results_values,
                                       fill=plsr_results_values))+
  geom_bar(stat='identity', position = "dodge", width = 0.8, show.legend=NA)
plot2<-plot+theme(axis.text.x=element_text(size=5.3))
plot2<-plot+ggtitle("PLSR predicted World happiness report first principal
                    components")
plot2+xlab("Component")+ylab("The component value")+
  labs(fill="Color coding")+scale_fill_gradient(low="red", high="green")
```



b.iv.

As one can see from the graph above, Healthy life expectancy, Logged GDP per capita, Social support are the most important variables for predicting the happiness.

c)

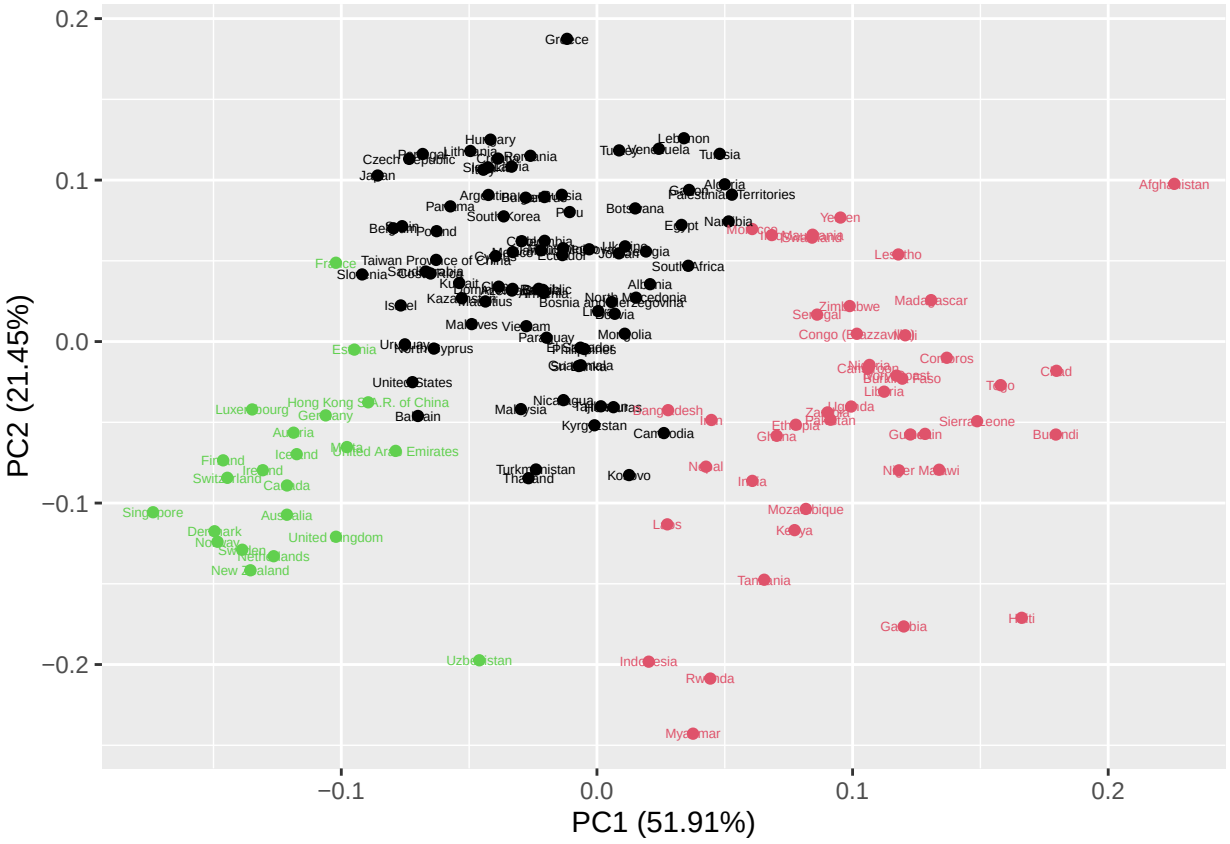
c.i. False c.ii. False c.iii. True c.iv. True

d)

d.i.

The given condition for clustering United States in one cluster while Norway, Sweden, Finland, Denmark in another cluster by K-means clusterization was satis- fied by setting K=3.4.

```
K = 3.4 # your choice
km.out = kmeans(happiness.X, K)
autoplot(pca_mat, data = happiness.X, colour = km.out$cluster, label = T, label.size = 1.8,
         loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 0.6)
```



d.ii. As one can see from the clusters statistics below, the happiest countries are in the cluster number 1 (black in the plot above), the cluster number two (red in the plot) is on second place and the cluster number 3 (green in the plot above) is third. In average, the countries in the first cluster are more than 50 % happier than the counties in the third cluster.

```
Clusters = km.out$cluster
Cluster_with_happiness = merge(happiness.Y, Clusters, by="row.names", all=TRUE)
Cluster_with_happiness<-Cluster_with_happiness[with(Cluster_with_happiness,
                                                    order(y)), ]
library(tidyverse)
```

```
Cluster_with_happiness_1<-Cluster_with_happiness%>%filter(y==1)
Cluster_with_happiness_1<-Cluster_with_happiness_1[with(Cluster_with_happiness_1,
                                                        order(Ladder.score)),]
Cluster_with_happiness_2<-Cluster_with_happiness%>%filter(y==2)
Cluster_with_happiness_2<-Cluster_with_happiness_2[with(Cluster_with_happiness_2,
                                                        order(Ladder.score)),]
Cluster_with_happiness_3<-Cluster_with_happiness%>%filter(y==3)
Cluster_with_happiness_3<-Cluster_with_happiness_3[with(Cluster_with_happiness_3,
                                                        order(Ladder.score)),]
print1 <- function(x,pcol,clusnum){
  print(paste0("There are ", length(x$Ladder.score)))
  print(paste0("countries in cluster number ", clusnum, " (in ", pcol, " in the plot above)"))
  print("The cluster has following statistics:")
  summary(x$Ladder.score)
}
print1(Cluster_with_happiness_1, "black", 1)
```

```
## [1] "There are 83"
## [1] "countries in cluster number 1 (in black in the plot above)"
## [1] "The cluster has following statistics:"

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.467   5.232   5.840   5.711   6.176   7.157
```

```
print1(Cluster_with_happiness_2, "red", 2)
```

```
## [1] "There are 44"
## [1] "countries in cluster number 2 (in red in the plot above)"
## [1] "The cluster has following statistics:"

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.523   4.017   4.630   4.455   5.026   5.345
```

```
print1(Cluster_with_happiness_3, "green", 3)
```

```
## [1] "There are 22"
## [1] "countries in cluster number 3 (in green in the plot above)"
## [1] "The cluster has following statistics:"

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.477   6.624   7.169   7.015   7.385   7.842
```