Compulsory exercise 2 TMA4268: Group 6

TMA4268 Statistical Learning V2022

August Arnstad, Markus Stokkenes and Ulrik Unneberg

04 april, 2022

We would like to start by saying that in order to avoid getting a massive size of this report, some output has been commented out. We did not see it necessary to include what has been left out. We apologize for having a paper more than 14 pages long, but most of it is plots and output.

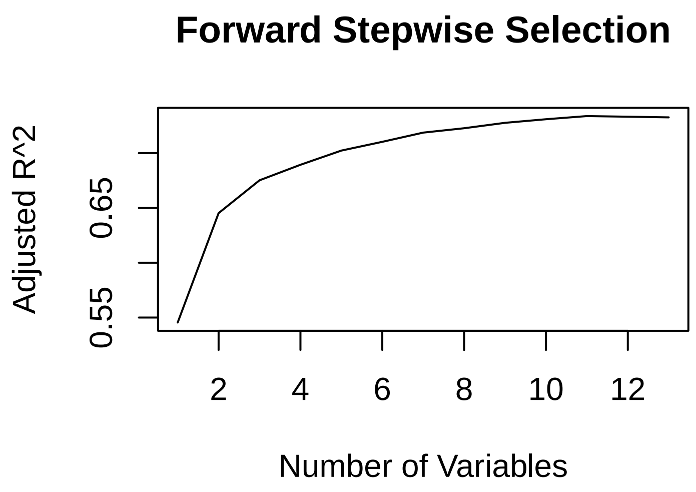
Task 1)

a)

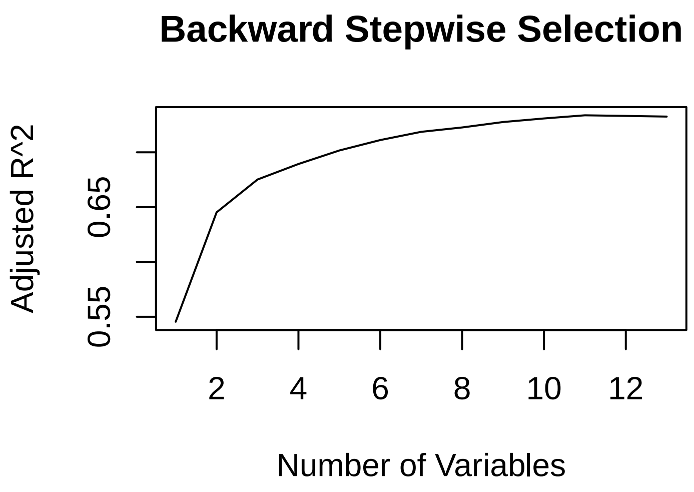
```
#str(Boston)

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)

#PROBLEM 1a)
#Perform Forward Stepwise Selection and Backward Stepwise Selection on boston.train method,
# and plot a graph of adjusted R2 on the y-axis and a number of predictors on the x-axis.

# split into training and test sets
train.ind = sample(1:nrow(boston), 0.8 * nrow(boston)) #We do a 80-20 split for training and test respe
boston.train = data.frame(boston[train.ind, ])
summary(boston.train )
```

```
##       crim                 zn                 indus               chas
##  Min.   :-0.41937   Min.   :-0.487240   Min.   :-1.55630   Min.   :-0.27233
##  1st Qu.:-0.41055   1st Qu.:-0.487240   1st Qu.:-0.86902   1st Qu.:-0.27233
##  Median :-0.39028   Median :-0.487240   Median :-0.21089   Median :-0.27233
##  Mean   :-0.01658   Mean   : 0.000275   Mean   : 0.01823   Mean   : 0.02003
##  3rd Qu.: 0.01197   3rd Qu.: 0.102320   3rd Qu.: 1.01499   3rd Qu.:-0.27233
##  Max.   : 8.12884   Max.   : 3.800473   Max.   : 2.42017   Max.   : 3.66477
##       nox                 rm                 age                 dis
##  Min.   :-1.46443   Min.   :-3.87641   Min.   :-2.333128   Min.   :-1.26582
##  1st Qu.:-0.92076   1st Qu.:-0.56344   1st Qu.:-0.886356   1st Qu.:-0.80613
##  Median :-0.14407   Median :-0.09555   Median : 0.363251   Median :-0.29410
##  Mean   :-0.01425   Mean   : 0.02572   Mean   :-0.001632   Mean   :-0.00502
##  3rd Qu.: 0.59809   3rd Qu.: 0.49439   3rd Qu.: 0.906790   3rd Qu.: 0.63942
##  Max.   : 2.72965   Max.   : 3.47325   Max.   : 1.116390   Max.   : 3.95660
##       rad                tax              ptratio              black
##  Min.   :-0.98187   Min.   :-1.31269   Min.   :-2.7047   Min.   :-3.87923
##  1st Qu.:-0.63733   1st Qu.:-0.75495   1st Qu.:-0.4876   1st Qu.: 0.19238
##  Median :-0.52248   Median :-0.43455   Median : 0.2977   Median : 0.37862
##  Mean   : 0.00854   Mean   : 0.02023   Mean   : 0.0234   Mean   :-0.03297
##  3rd Qu.: 1.65960   3rd Qu.: 1.52941   3rd Qu.: 0.8058   3rd Qu.: 0.42720
##  Max.   : 1.65960   Max.   : 1.79642   Max.   : 1.6372   Max.   : 0.44062
##      lstat               medv
##  Min.   :-1.52961   Min.   :-1.90634
##  1st Qu.:-0.81403   1st Qu.:-0.62605
##  Median :-0.18107   Median :-0.11230
##  Mean   :-0.01568   Mean   : 0.03562
##  3rd Qu.: 0.56986   3rd Qu.: 0.32534
##  Max.   : 3.54526   Max.   : 2.98650
```

```
boston.test = data.frame(boston[-train.ind, ])

# Forward
regfit_fwd = regsubsets(medv~., data = boston.train, nvmax=14, method = "forward")
regfit_fwd_summary = summary(regfit_fwd)
#regfit_fwd_summary
plot(regfit_fwd_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R^2", type = "l", main="F
```



```
# Backward
regfit_bwd = regsubsets(medv~., data = boston.train,  nvmax=14, method = "backward")
regfit_bwd_summary = summary(regfit_bwd)
plot(regfit_bwd_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted R^2", type = "l", main="B
```



b)

```
regfit_fwd_summary
```

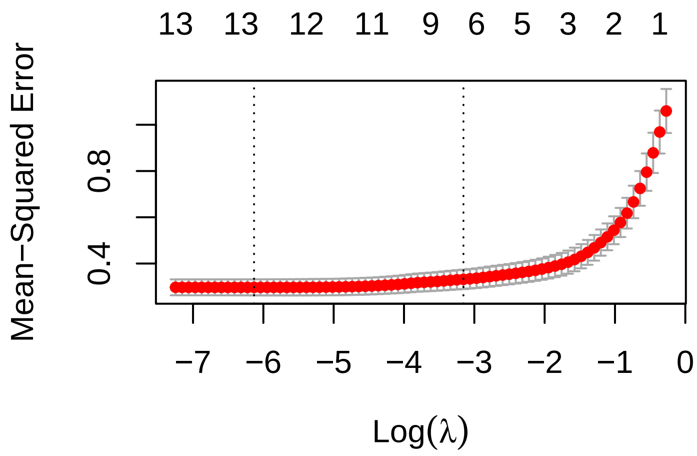
```
## Subset selection object
## Call: regsubsets.formula(medv ~ ., data = boston.train, nvmax = 14, 
##     method = "forward")
## 13 Variables  (and intercept)
##         Forced in Forced out
## crim        FALSE      FALSE
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        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 ) "*"  "*" "*"  "*"  "*" "*" "*" "*" "*" "*" "*"    "*"   "*"
```

From the fourth line of the summary: The four best predictors, according to the Forward Stepwise Selection, is the predictors *rm*, *dis*, *ptratio*, and *lstat*.

c)

i) We will now do a 5-fold cross-validation on the boston.train data set, with the Lasss, and plot the MSE as a function of $\log \lambda$

```r
set.seed(1)
x = model.matrix(medv~., data=boston.train)
y = boston.train$medv

set.seed(1)
cv.lasso = cv.glmnet(x, y , nfolds=5, alpha=1)
plot(cv.lasso)
```



ii) We λ with minimum MSE is marked with the dotted line to the left:

```r
cv.lasso$lambda.min
```

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

iii) We will now find the coefficients at the best λ, i.e. at one standard deviation away from the lowest, in the direction of fewer predictors, i.e. to the right. This is a model consisting of 7 predictors, and we find the coefficients with

```
coef(cv.lasso)
```

```
## 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)

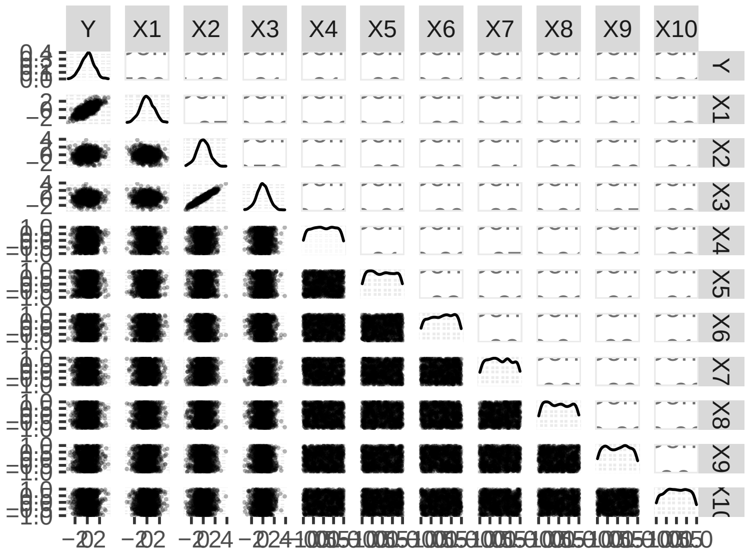
TRUE, FALSE, FALSE, TRUE

Task 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)
```

```
##             Y          X1          X2         X3         X4         X5
## 1 -1.43753239 -0.75905055 -0.69720326 -0.3016852 -0.7434697  0.8807558
## 2 -1.70972989 -0.28635632  0.04809182  0.5791725 -0.7446170  0.9935311
## 3  1.33931240  0.09574117 -0.89605758 -0.9636347  0.5554647 -0.5341800
## 4  0.20354906 -0.28702695  1.72952687  1.4289705 -0.1596993 -0.7161976
## 5 -0.09261896  0.02345825  0.51201583  0.1544345  0.4318039 -0.8674060
## 6  1.69952325  1.19231791 -0.98179754 -0.9567773 -0.6933918  0.4656891
##          X6         X7         X8          X9        X10
## 1 -0.8705750 -0.7448252 -0.4639697  0.62502272 -0.8149674
## 2  0.3532248 -0.5860332 -0.7964403  0.84868110 -0.1065119
## 3  0.4707434 -0.6588069 -0.7327518 -0.29429307  0.6588927
## 4 -0.7774007  0.2502145  0.5987052 -0.04428773  0.6247479
## 5 -0.9066908  0.8946086 -0.9700185  0.09082626  0.6102134
## 6 -0.7381794  0.8650175  0.4108119  0.75677429 -0.2281439
```

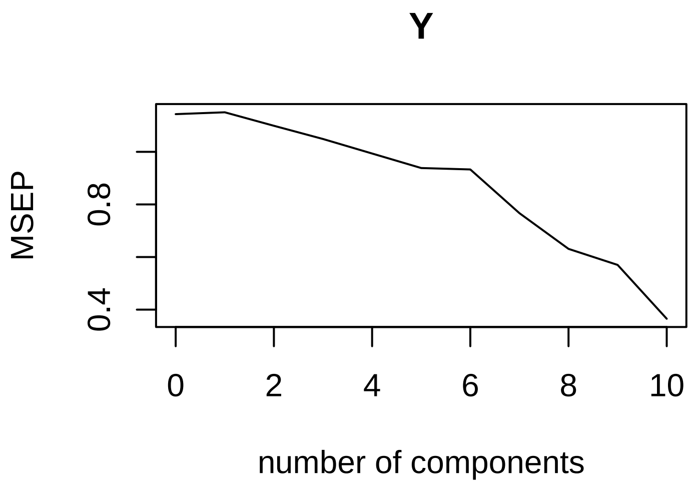
```
ggpairs(synthetic, lower = list(continuous = wrap("points", alpha = 0.3,    size=0.1),
              combo = wrap("dot", alpha = 0.4,            size=0.2) ),)
```



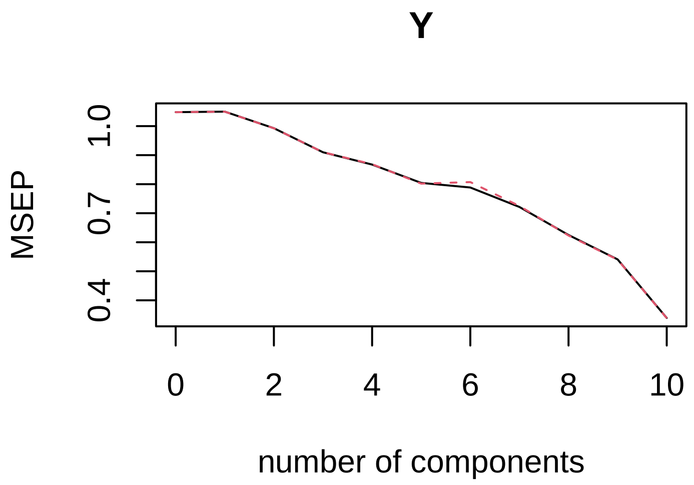
We first take a look at the data. At first glance, we first observe that there is a prominant correlation between X_2 and X_3, and between Y and X_1.

a)

```
#Principle Component Regression
library(pls)
set.seed(1)
pcr.fit = pcr(Y ~., data=synthetic.train, scale = TRUE, validation = "CV")
#summary(pcr.fit)
validationplot(pcr.fit, val.type = "MSEP", newdata =synthetic.test)
```

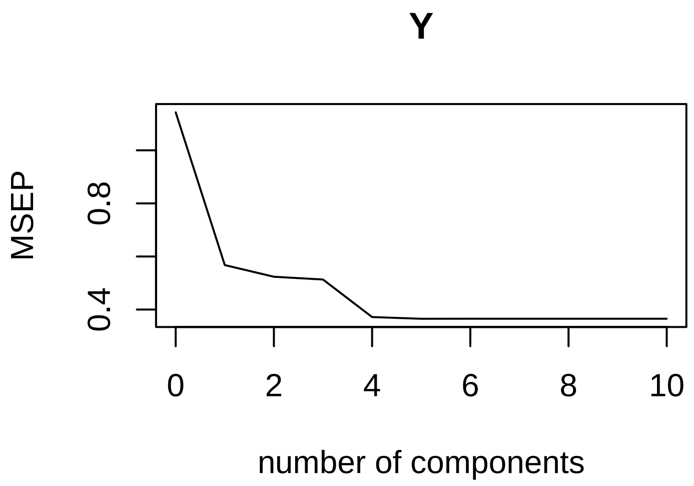


```
#Testing both cross-validation and with test data. Similar results.
validationplot(pcr.fit, val.type = "MSEP")
```

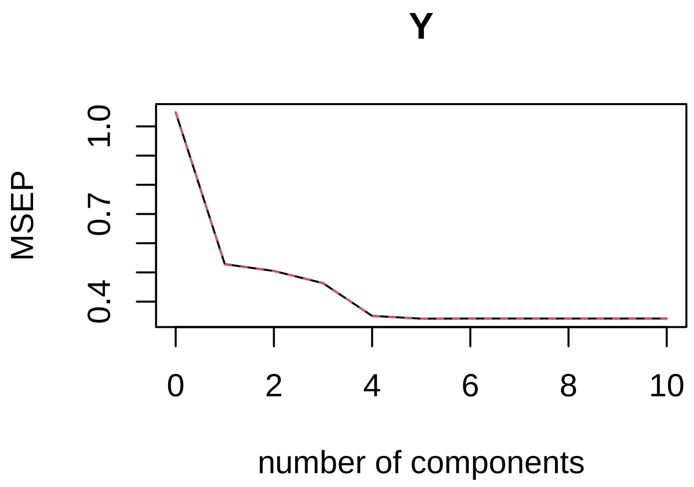


The MSEP for the PCR is clearly minimized with 10 components, i.e. a full model. This suggests that a PCR model won't benefit over an Ordinary Least Squares model.

```
#Partial Least Squares
pls.fit = plsr(Y ~., data=synthetic.train, scale = TRUE, validation = "CV")
#summary(pls.fit)
validationplot(pls.fit, val.type = "MSEP", newdata = synthetic.test)
```



```
#Testing both cross-validation and with test data. Similar results.
validationplot(pls.fit, val.type = "MSEP")
```



We observe that for the Partial Least Squares model, the MSEP flat out at the minimum MSEP from 4 components or more. This is a substantial improvement in the number of predictors; 4 instead of 10.

b)

In general, the PCR is an unsupervised method, meaning that the correlation between predictors and the response won't influence the choice of principle components. The PCR selects a linear combination of predictors along the direction with the most variance. The PLS, on the other hand, is a supervised method, prioritizing the predictors which correlates the most to Y. The components in the PLS is a linear combination of the predictors, where each of them is weighted according to how much Y varies with the particular predictor.

As we see from the ggpairs plot, the strong correlation between Y and X_1 suggests that a simple model describes the trends quite well. The PCR doesn't pick up this relation, as the model is unsupervised; the principle component might be in any direction where the variance of the predictors is large. We observe that the variables $ X_2 $ and X_3 have a large correlation, and thus common covariance along the same direction. These variables will dominate in the first principle component, completely ignoring the relationship between Y and X_1. As there seems to be little correlation between Y and $X_2$, X_3. This explains why the MSEP doesn't seem to change going from zero to one component in the PCR.

The first component in the PLS is dominated by X_1, and we thus see that when we include this component, the MSEP drop drastically. It further drops to the minimum MSEP when the number of components reach four.

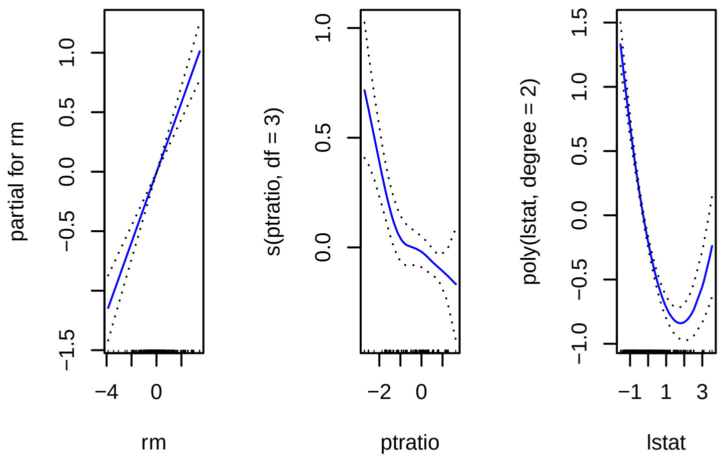
Task 3)

a)

TRUE, FALSE, FALSE, TRUE

b)

```r
set.seed(1)

data(Boston)

boston<-scale(Boston, center=T, scale=T)

train.id=sample(1:nrow(boston), 0.8*nrow(boston))
boston.train = data.frame(boston[train.id, ])
#str(boston.train)
genaddmod<-gam(medv~rm + s(ptratio, df=3) + poly(lstat, degree=2), data=boston.train)

par(mfrow=c(1,3))
plot(genaddmod, se=T, col="blue")
```

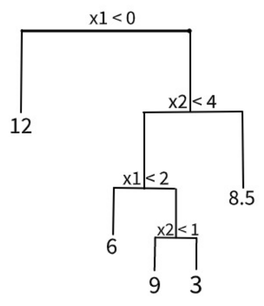


Task 4

a)

FALSE, TRUE, TRUE, TRUE

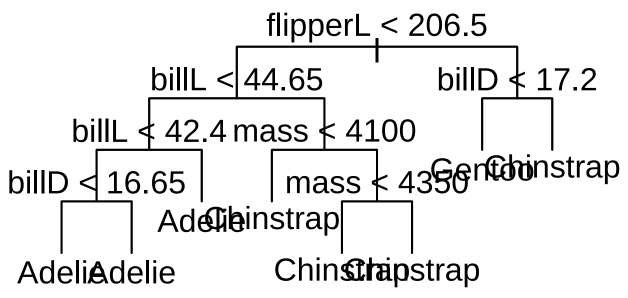
b)



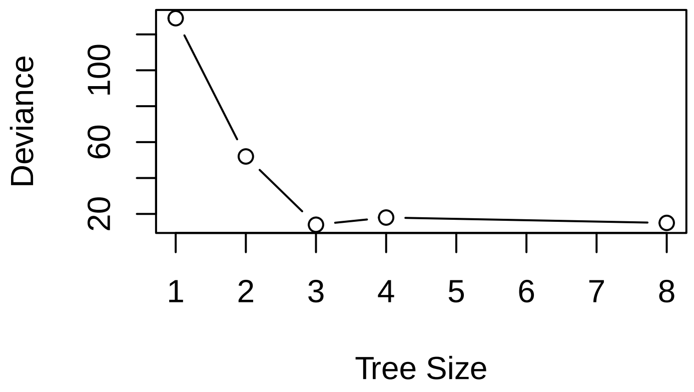
c)

```r
library(tidyverse)
library(palmerpenguins)  # Contains the data set 'penguins'.
library(tree)
library(randomForest)
library(e1071)
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, ]
```

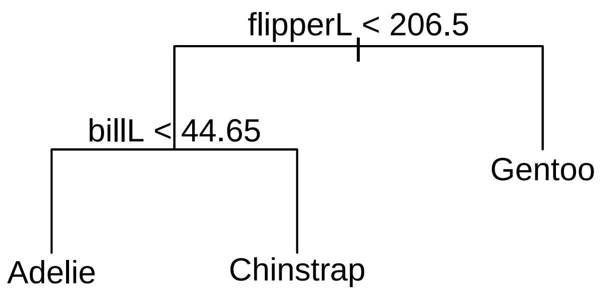
```
# create tree with default parameters using gini index
penguin.tree <- tree(species ~ ., train, split = "gini")

# plot the full tree
plot(penguin.tree, type = "uniform")
text(penguin.tree, pretty = 0)
```



```
set.seed(123)

# cost-complexity pruning using 10-fold CV
cv.penguin <- cv.tree(penguin.tree, FUN = prune.misclass, K = 10)

# plot deviance as a function of tree size
plot(cv.penguin$dev ~ cv.penguin$size, type = "b", xlab = "Tree Size", ylab = "Deviance")
```



```
# prune the tree according to the optimal tree size
prune.penguin <- prune.tree(penguin.tree, best = 3)

# plot the pruned tree
plot(prune.penguin, type = "uniform")
text(prune.penguin, pretty = 0)
```



```
# get predictions from training data using the pruned tree
pred.cv <- predict(prune.penguin, newdata = test, type = "class")

# create confusion table for pruned tree
misclass.cv <- table(pred.cv, test$species)

# misclassification rate
error.cv <- 1 - sum(diag(misclass.cv))/sum(misclass.cv)
```

```r
cat("Misclassification error rate after cost-complexity pruning:", error.cv)
```

```
## Misclassification error rate after cost-complexity pruning: 0.06
```

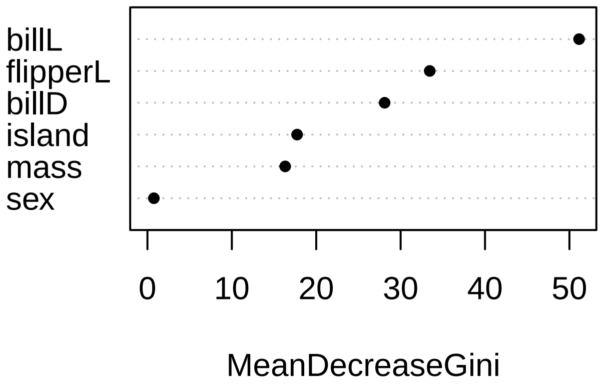
d)

```r
# create tree using random forest method
rf.penguin = randomForest(species ~ ., data = train, mtry = round(sqrt(ncol(Penguins_reduced) - 1)), nt

# get predictions
pred.rf = predict(rf.penguin, newdata = test, type = "class")

# confusion table
misclass.rf <- table(pred.rf, test$species)

# misclassification rate
error.rf = 1 - sum(diag(misclass.rf))/sum(misclass.rf)

cat("Misclassification error rate for random forest method:", error.rf)
```

```
## Misclassification error rate for random forest method: 0.02
```

Since we are doing classification we choose the tuning parameter m (number of splits) to be $\approx \sqrt{p/3}$, where p is the number of predictors, as is recommended by the creators of the random forest method.

```r
# variable importance plot; type = 2 because we are doing classification and are interested in the gini
varImpPlot(rf.penguin, pch = 20, main = "", type = 2)
```



As is evident from the variable importance plot above (as well as the previously plotted trees from 4c), the most significant variables for prediction of penguin species are bill length and flipper length.

Task 5

a)

FALSE, TRUE, TRUE, TRUE

b)

```r
# find optimal cost for support vector classifier
cv.svc = tune(svm, species ~ ., data = train, kernel = "linear", ranges = list(cost = 10^seq(-1, 2, 0.25

# create the optimal classifier
svc.best <- cv.svc$best.model

# get preditions
svc.pred <- predict(svc.best, test)

# confusion table
svc.misclass <- table(svc.pred, test$species)

# ------------------------------------------- #

# find optimal cost and gamma parameter for support vector machine with radial kernel
cv.svm = tune(svm, species ~ ., data = train, kernel = "radial", ranges = list(cost = 10^seq(-1, 2, 0.25

# create optimal SVM
svm.best <- cv.svm$best.model

# get predictions
svm.pred <- predict(svm.best, test)

# confusion table
svm.misclass <- table(svm.pred, test$species)
```

```r
# optimal parameter for linear case and corresponding error
summary(cv.svc)[1]
```

```
## $best.parameters
##       cost
## 6 1.778279
```

```r
summary(cv.svc)[2]
```

```
## $best.performance
## [1] 0
```

```r
# optimal parameters for radial case and corresponding error
summary(cv.svm)[1]
```

```
## $best.parameters
##     cost gamma
## 117  100  0.01
```

```
summary(cv.svm)[2]
```

```
## $best.performance
## [1] 0
```

We can observe from the above results that the opimal cost for the linear classifier is 1.778279, while the optimal cost and gamma parameter for the radial SVM are 100 and 0.01, respectiely. In both cases, training error is zero.

```
# confusion table for linear classifier
svc.misclass
```

```
##
## svc.pred    Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

```
# confusion table for SVM with radial kernel
svm.misclass
```

```
##
## svm.pred    Adelie Chinstrap Gentoo
##   Adelie        42         0      0
##   Chinstrap      0        20      0
##   Gentoo         0         0     38
```

In both cases, all test penguins are classified correctly, so the misclassification error rates are zero. This makes it difficult to decide on a preferred classifier. Although the radial kernel might be more widely used in practice, we would perhaps prefer the simpler model, because the cross validation and training is less computationally expensive, while still yielding the same results as the radial SVM.

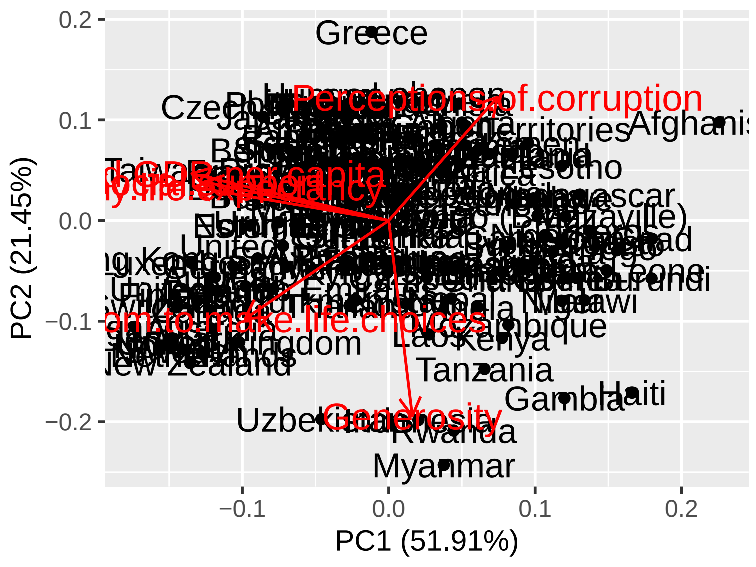
Task 6)

```
set.seed(1)
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',
```

```r
         '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))

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

summary(pca_mat)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6
## Standard deviation     1.7648 1.1344 0.8384 0.7200 0.50030 0.35657
## Proportion of Variance 0.5191 0.2145 0.1172 0.0864 0.04172 0.02119
## Cumulative Proportion  0.5191 0.7335 0.8507 0.9371 0.97881 1.00000
```

```r
# 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)

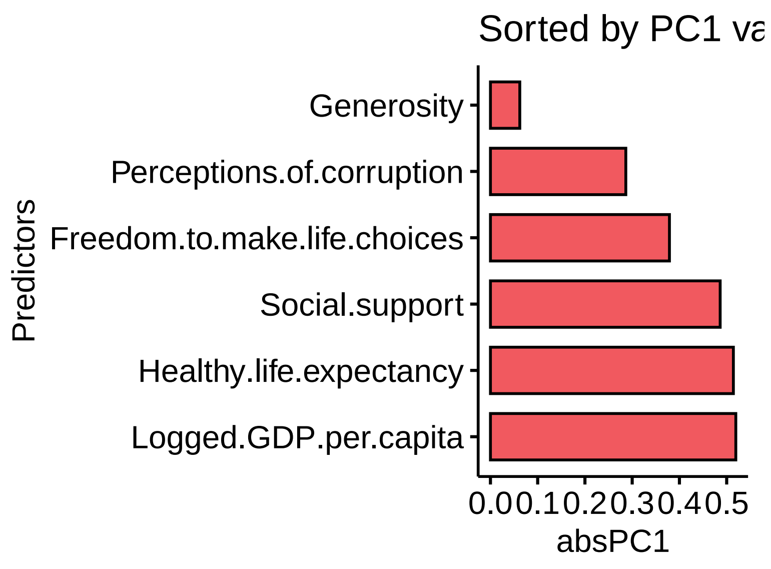
i) We are to comment on two characteristics, and the first one we notice is that the predictors "perception of corruption" and "freedom to make life choices" are opposites. This can be interpreted to mean that in countries with a high score of “perception of corruption” have little “freedom to make choices” and vise versa. This can be seen as they are negatively correlated in both PC1 and PC2. The second characteristic we choose to mention, is that of “generosity” and “healthy life expectancy”. We notice that “generosity” almost only depends on PC2, while “healthy life expectancy” almost only depends on PC1. That would mean that these are nearly uncorrelated. One might then say that “healthy life expectancy” does not affect a countries “generosity” so two countries could be equally generous, even if one has a bad and one has a good “healthy life expectancy” score.

ii) Afghanistan

b)

i)

```
set.seed(1)
df<-data.frame(pca_mat$rotation)$PC1

absPC <- data.frame(absPC1 = abs(df),
  Predictors = c(colnames(happiness.XY)[-1]))

ggbarplot(absPC, x = "Predictors", y = "absPC1",
          main="Sorted by PC1 value",
          fill="#f1595f",
          sort.val = "desc",
          sort.by.groups = FALSE,
          x.text.angle = 0,
          orientation="horizontal",
          lab.size = 0.1
          )
```

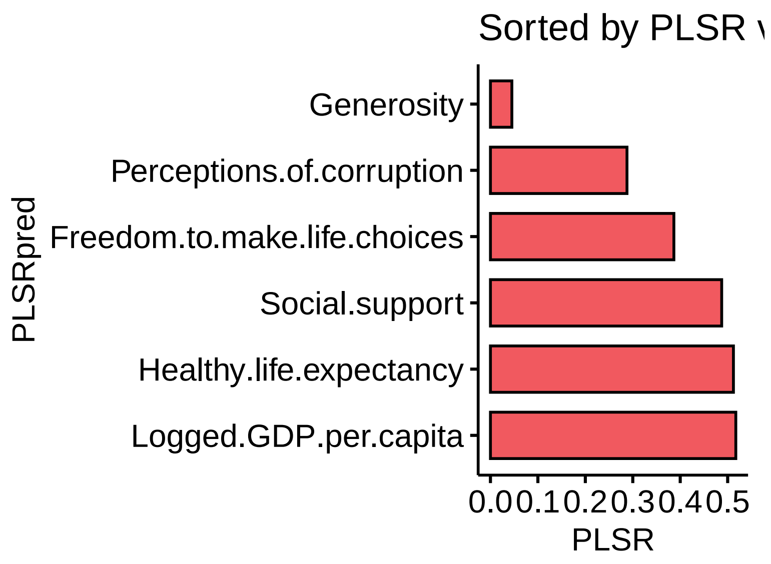


ii)

```r
set.seed(1)
plsr_model <- plsr(Ladder.score~., data=happiness.XY, scale=T)
```

iii)

```r
set.seed(1)
absPLSR <- data.frame(PLSR =
  abs(plsr_model$loadings[,c('Comp 1')]),
  PLSRpred = c(colnames(happiness.XY)[-1]))

ggbarplot(absPLSR, x = "PLSRpred", y = "PLSR",
          main="Sorted by PLSR value",
          fill="#f1595f",
          sort.val = "desc",
          sort.by.groups = FALSE,
          x.text.angle = 0,
          orientation="horizontal",
          lab.size = 0.1
          )
```



As expected, the absolute values of the PCA and the results from the PLSR are very similar.

iv) The three most important factors based on the PLSR are the predictors corresponding to the largest values in the barplot, i.e. Logged GDP per capita, healthy life expectancy and social support in that order. This is also seen from the PCA barplot, which is very similar to the PSLR barplot.

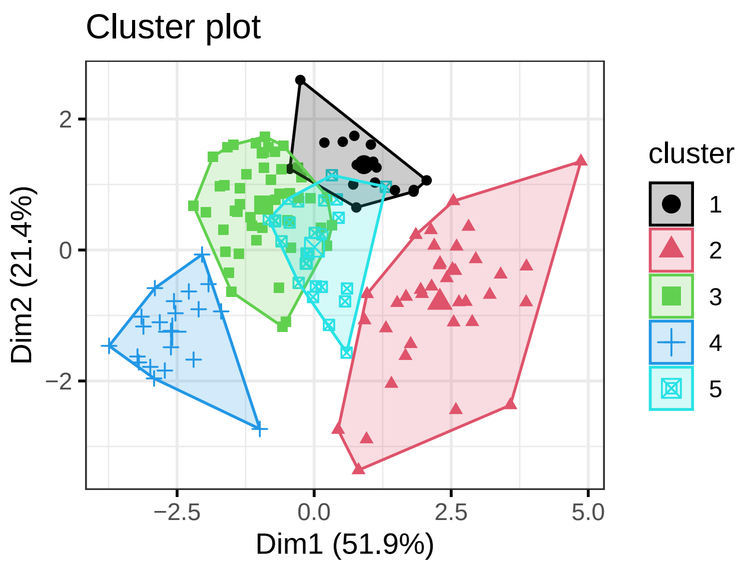
c)

FALSE, FALSE, FALSE, TRUE

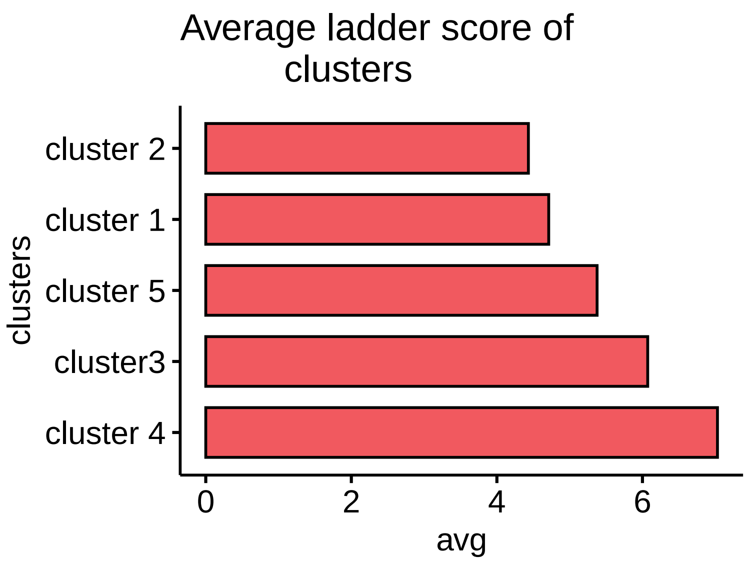
d)

i)

```
set.seed(1)
K = 5  # We choose this
km.out = kmeans(happiness.X, K)

fviz_cluster(km.out, data=happiness.X,
             palette = palette(rainbow(5)),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```



```
cluster_happiness<-function(){
  avgtest=rep(NA, K)
  for (i in 1:K){
    test=km.out$cluster==i
    index=which(test==TRUE)
    avgtest[i]<-mean(happiness.XY[index, ]$Ladder.score)
  }
  return (avgtest)
}

avg_happiness_cluster=cluster_happiness()

avg_df<-data.frame(avg=avg_happiness_cluster,
    clusters=c("cluster 1", "cluster 2",
    "cluster3", "cluster 4", "cluster 5"))

ggbarplot(avg_df, x = "clusters", y = "avg",
          main="Average ladder score of
          clusters",
          fill="#f1595f",
          sort.val = "desc",
          sort.by.groups = FALSE,
          x.text.angle = 0,
          orientation="horizontal"
          )
```



ii) We make a plot that is easier to interpret, and we see from the barplot that cluster 4(Finland, Switzerland, Australia) has the highest average ladder/happiness score and cluster 2(Burundi, India, Myanmar) has the lowest. It is also worth to mention that happiness seems to go from left to right in declining order.