Compulsory exercise 2: Group 18

TMA4268 Statistical Learning V2022

Erlend Nonås Lokna, Thomas Fardal Rødland

04 April, 2022

Problem 1a)

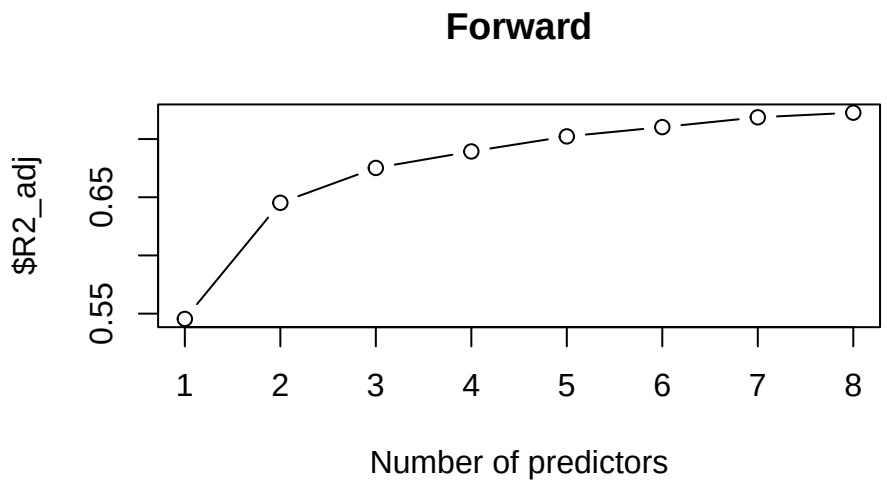
```
set.seed(1)
boston.forward <- regsubsets(medv ~ ., data = boston.train, method = "forward")
boston.backward <- regsubsets(medv ~ ., data = boston.train, method = "backward")

summary.forward <- summary(boston.forward)
summary.backward <- summary(boston.backward)

summary.forward$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 ) " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
```

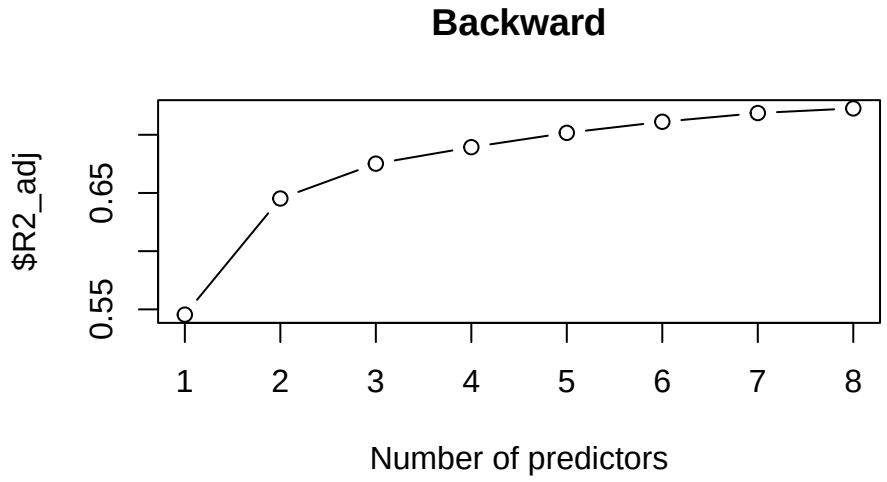
```
summary.backward$outmat
```

```
##          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 ) " "  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
```

```
plot(summary.forward$adjr2, xlab = "Number of predictors", ylab = "$R2_adj", main = "Forward",
    type = "b")
```




```
plot(summary.backward$adjr2, xlab = "Number of predictors", ylab = "$R2_adj", main = "Backward",
    type = "b")
```

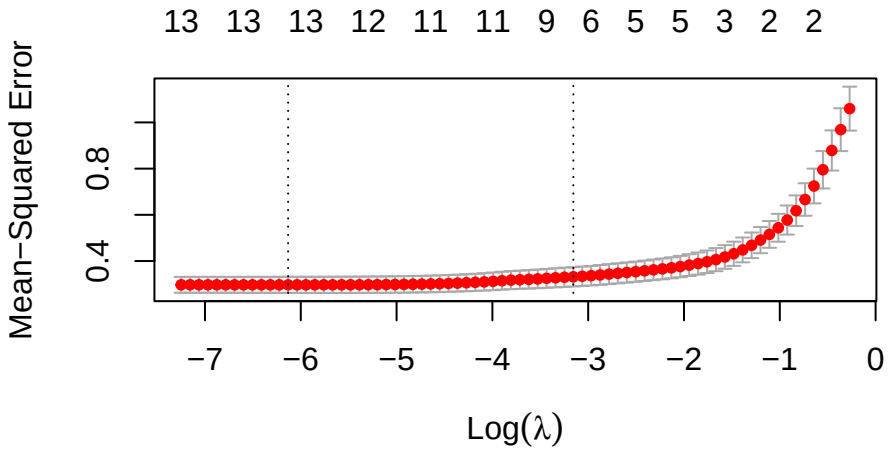


1b)

From the output matrix of the forward stepwise selection we see that rm, ptratio, dis and lstat are the four best predictors.

1c)

```
# K-fold-cross-validation prepare data
set.seed(1)
x.lasso <- as.matrix(boston.train[, 1:13])
y.lasso <- boston.train[, 14]
boston.lasso = glmnet(x = x.lasso, y = y.lasso, family = "gaussian", alpha = 1)
# plot(boston.lasso, xvar = 'lambda', label = TRUE)

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




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

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  0.024709949
## 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
```

```
paste("Best lambda: ", lasso.cv$lambda.min)
```

```
## [1] "Best lambda:  0.00217203153907965"
```

The fitted coefficients at the best lambda can be seen from the outputmatrix

1d)

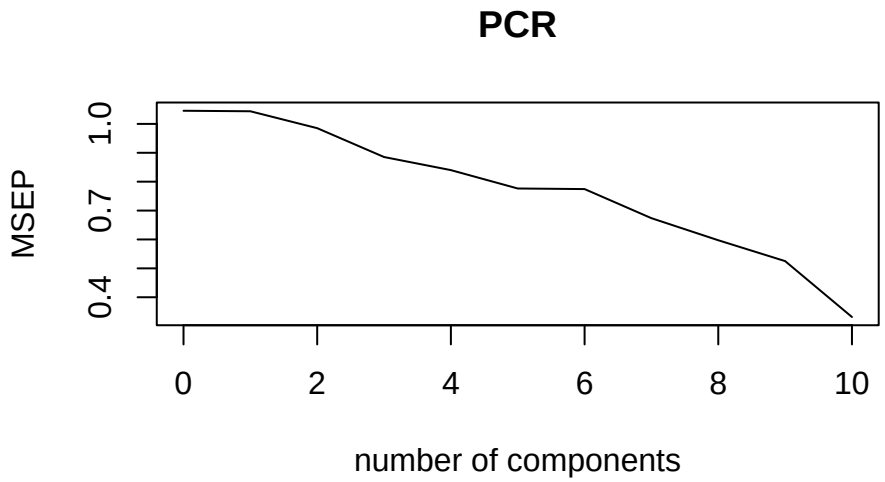
- True
- False
- False
- True

Problem 2

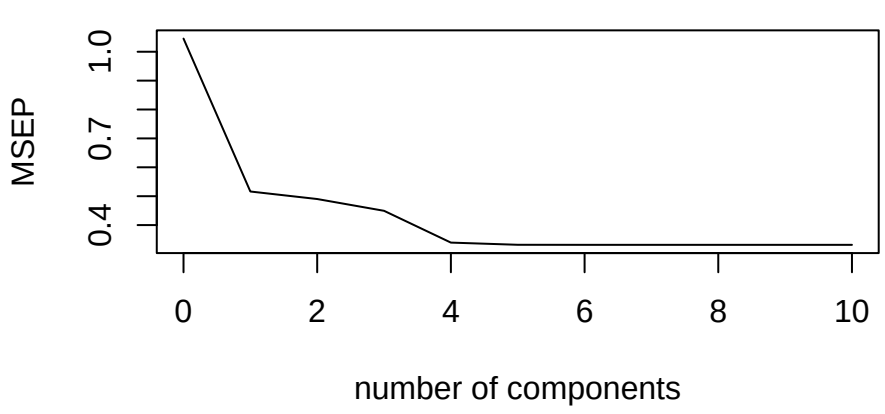
a)

```
set.seed(1)

# load a synthetic dataset
id <- "1CWZYfrLOrFdrIZ6Hv73e3xxtOSFgU4Ph"  # 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)
synth.pcr <- pcr(Y ~ ., data = synthetic.train, scale = TRUE)
synth.plsr <- plsr(Y ~ ., data = synthetic.train, scale = TRUE)

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

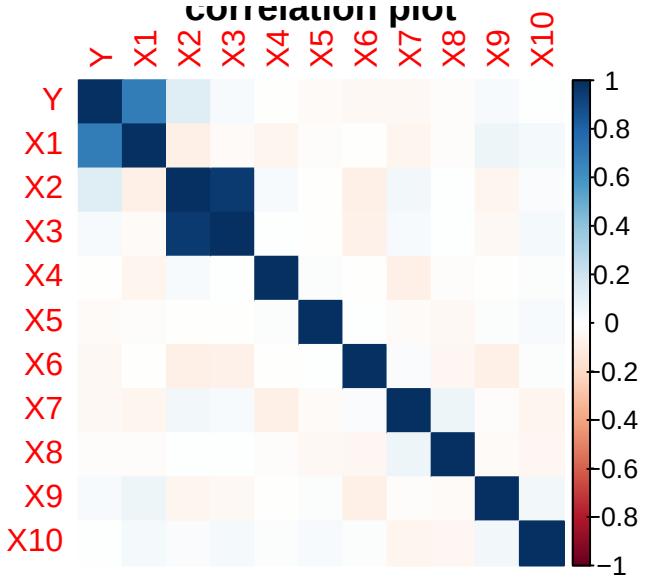



```
validationplot(synth.plsr, val.type = "MSEP", main = "PLSR")
```



2b)

```
library(corrplot)
M = cor(synthetic.train)
corrplot(M, method = "color", main = "correlation plot")
```



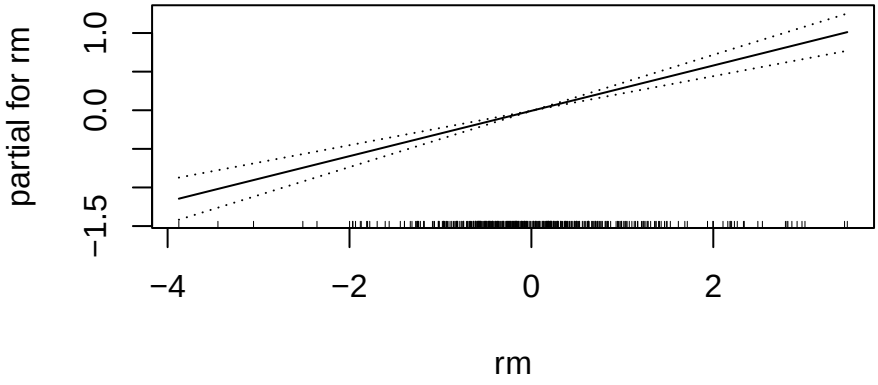
From the outputmatrix of correlations between variables we see that some variables are highly correlated, while others have almost no correlation what so ever. Based on the performance of PCR and PLSR, visualized by the plots, one can explain the difference on this dataset by two ideas. First of all PCR and PLSR are are regression methods that fall under two different categories, unsupervised and supervised learning respectively. This means that while the PLSR has a target or a response Y, PCR does not. Because of this one might get a worse performance from PCR when variables are highly correlated. In this data set specifically we see that X_2 and X_3 have a really high correlation, as do the response Y and X_1. Since we have a high correlation in these predictors we will struggle with the issue that the directions with the lower variance will be dropped, and so we will not get as good predictive power as with PLSR where we do have a response.

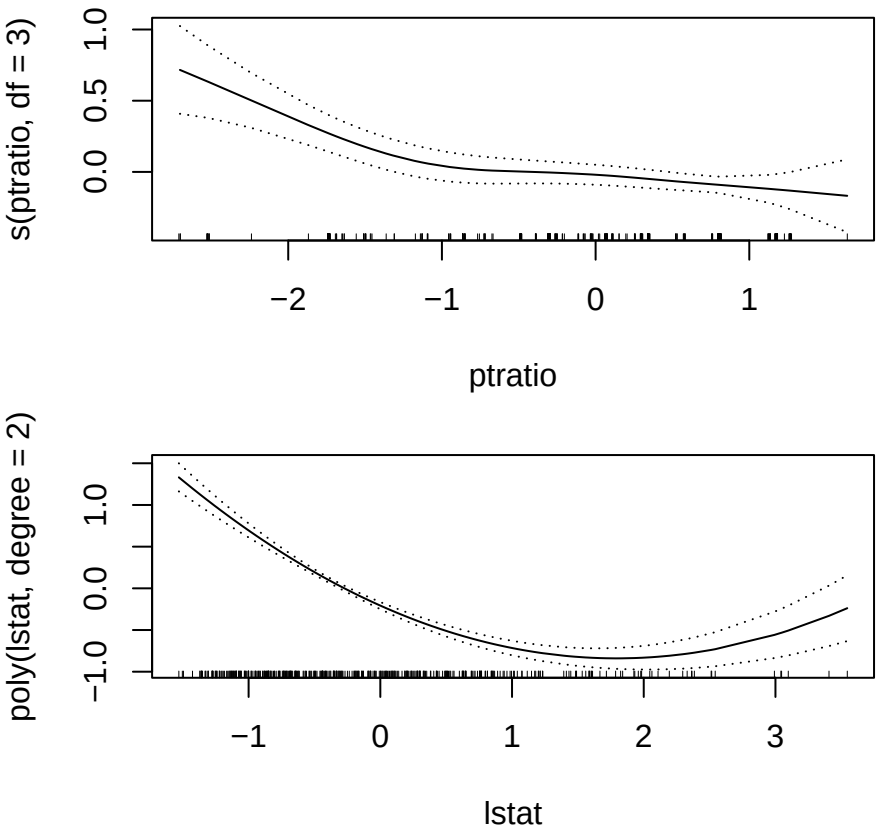
Problem 3

3a)

- True
- False
- False
- True

```
# rm.lm <- lm(rm~.)
boston.gam <- gam(medv ~ rm + s(ptratio, df = 3) + poly(lstat, degree = 2), data = boston.train)
plot(boston.gam, se = TRUE)
```





Problem 4

4a)

- False
- True
- True
- True

b)

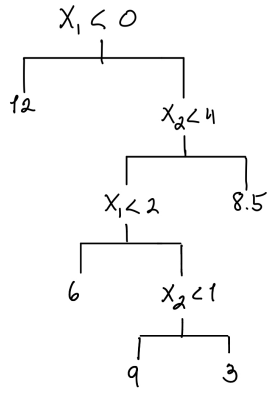


Figure 1: A nice image.

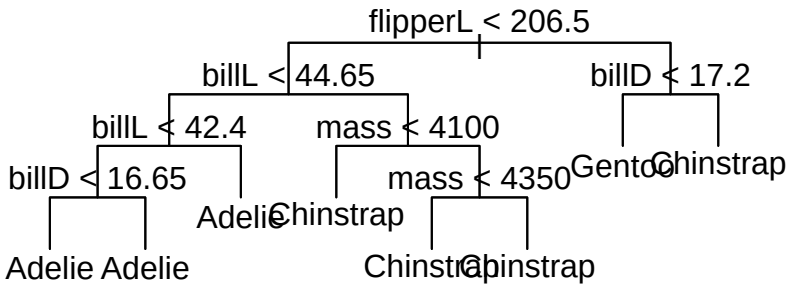
c)

```
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, ]
```

```
penguin.tree <- tree(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
```

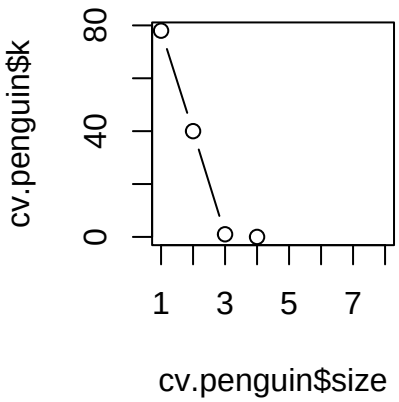
```
plot(penguin.tree, type = "uniform")
text(penguin.tree, pretty = 1)
```




```
set.seed(123)
cv.penguin <- cv.tree(penguin.tree, FUN = prune.misclass)
```

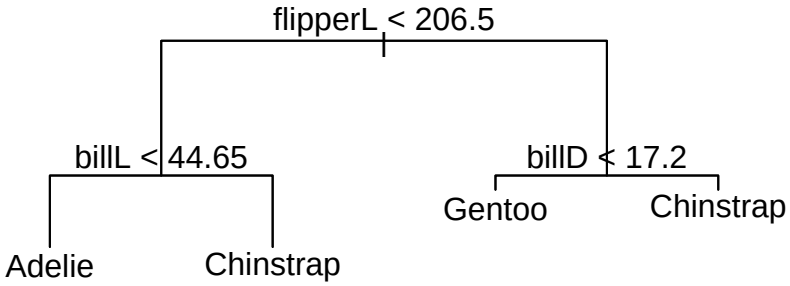
We see that the tree with only 4 terminal nodes has the lowest k

```
par(mfrow = c(1, 2))
plot(cv.penguin$size, cv.penguin$k, type = "b")
```



Now we prune the tree

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



With the pruned optimal tree we take a look at how it performs on the test set.

```
set.seed(123)
tree.pred.penguin <- predict(prune.penguin, newdata = test, type = "class")


conf <- confusionMatrix(tree.pred.penguin, test$species)
conf$table
```

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        42         5      1
##   Chinstrap      0        15      0
##   Gentoo         0         0     37
```

```
conf$overall[1]
```

```
## Accuracy
##     0.94
```

d)

As a more advanced method we choose the random forest

```
rf <- randomForest(species ~ ., data = train, proximity = TRUE)
```

```
set.seed(1)
rf.pred <- predict(rf, newdata = test, type = "class")
```

```
conf_rf <- confusionMatrix(rf.pred, test$species)
conf_rf$table
```

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        42         2      0
##   Chinstrap      0        18      0
##   Gentoo         0         0     38
```

```
conf_rf$overall[1]
```

```
## Accuracy
##     0.98
```

```
rf$importance
```

```
##          MeanDecreaseGini
## island         18.3743845
## billL          51.1530887
## billD          24.7025773
## flipperL       36.1109040
## mass           16.2076984
## sex             0.8576727
```

We get an accuracy of 98% and so we have a misclassification error of 2%. The tuning parameters are set to default 500 trees and the number of variables randomly sampled are the square root of the number of predictors. These can of course be optimized further, but we found that they did a good job. One can see from the importance feature that bill length and flipper length are the two most important predictors

Problem 5

5a)

- False
- True
- True
- True

5b)

```
penguins.svc <- svm(species ~ ., data = train, kernel = "linear", scale = TRUE)
penguins.svm <- svm(species ~ ., data = train, kernel = "radial", scale = TRUE)
```

```
set.seed(123)
penguin.svc.tuned <- tune(svm, species ~ ., data = train, kernel = "linear", range = list(cost = c(0.01
    0.1, 0.5, 1, 2, 5, 10, 100)))
penguin.svm.tuned <- tune(svm, species ~ ., data = train, kernel = "radial", range = list(cost = c(0.1,
    0.5, 1, 2, 5), gamma = c(0.5, 1, 2, 3, 4)))
penguin.svc.tuned$best.parameters
```

```
##   cost
## 6    5
```

```
penguin.svm.tuned$best.parameters
```

```
##   cost gamma
```

```
## 5    5   0.5
```

We see from the output that a cost of 5 and a γ of 0.5 are the best parameters

```
new.svc <- svm(species ~ ., data = train, kernel = "linear", scale = TRUE, cost = 5)
new.svm <- svm(species ~ ., data = train, kernel = "radial", scale = TRUE, cost = 5,
    gamma = 0.5)
new.svm.pred <- predict(new.svm, newdata = test, type = "class")
new.svc.pred <- predict(new.svc, newdata = test, type = "class")

conf_svc <- confusionMatrix(new.svc.pred, test$species)
conf_svc$table
```

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        41         0      0
##   Chinstrap      1        20      0
##   Gentoo         0         0     38
```

```
conf$overall[1]
```

```
## Accuracy
##     0.94
```

```
conf_svm <- confusionMatrix(new.svm.pred, test$species)
conf_svm$table
```

```
##            Reference
## Prediction  Adelie Chinstrap Gentoo
##   Adelie        41         2      0
##   Chinstrap      1        18      0
##   Gentoo         0         0     38
```

```
conf_svm$overall[1]
```

```
## Accuracy
##     0.97
```

We see that the SVC has an accuracy of 94% and the SVM an accuracy of 97%. We would choose svc over svm when the performance of the two models are so close because of its simplicity.

Problem 6

6a)

```
# 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-8-BOM")

# 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)
```

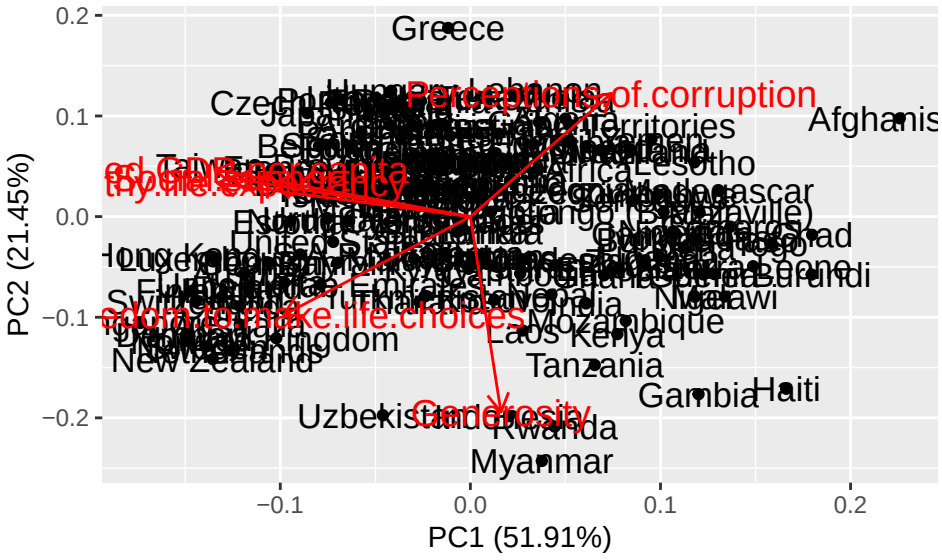
i)

We can see that GDP, Social Support and Health Life Expectancy are correlated. On the other hand we have negative correlation between Perceptions of Corruption and Freedom of Life Choices.

ii)

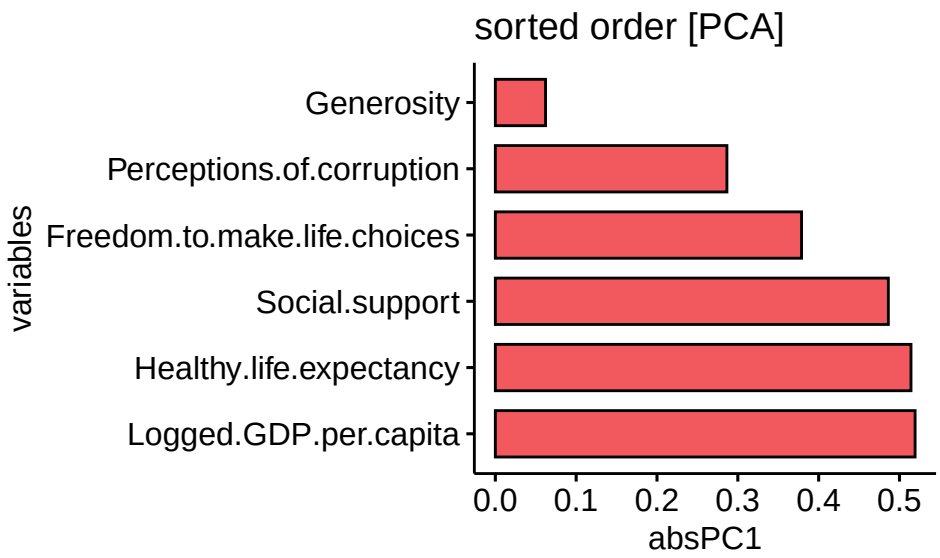
Afghanistan is the outlier.

```
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)
```



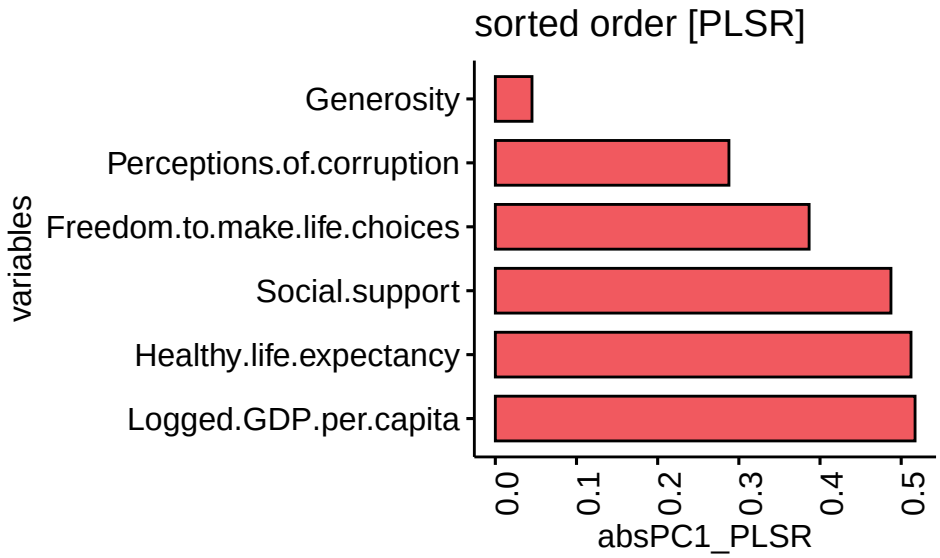

```
library("ggpubr")
plotData <- data.frame(absPC1 = abs(data.frame(pca_mat$rotation)$PC1), variables = c(colnames(happiness
ggbarplot(plotData, x = "variables", y = "absPC1", main = "sorted order [PCA]", fill = "#f1595f",
```

```
    sort.val = "desc", sort.by.groups = FALSE, x.text.angle = 0, orientation = "horizontal",
    cex.names = 0.3)
```



Here the plot is rotated 90 degrees for easier reading.

```
plsr_mat <- plsr(Ladder.score ~ ., data = happiness.XY, scale = T)
plotData$absPC1_PLSR <- abs(plsr_mat$loadings[, c("Comp 1")])
ggbarplot(plotData, x = "variables", y = "absPC1_PLSR", main = "sorted order [PLSR]",
    fill = "#f1595f", sort.val = "desc", sort.by.groups = FALSE, x.text.angle = 90,
    orientation = "horizontal")
```



The three most important predictors are:

1. Logged GDP per capita
2. Healthy life expectancy
3. Social support

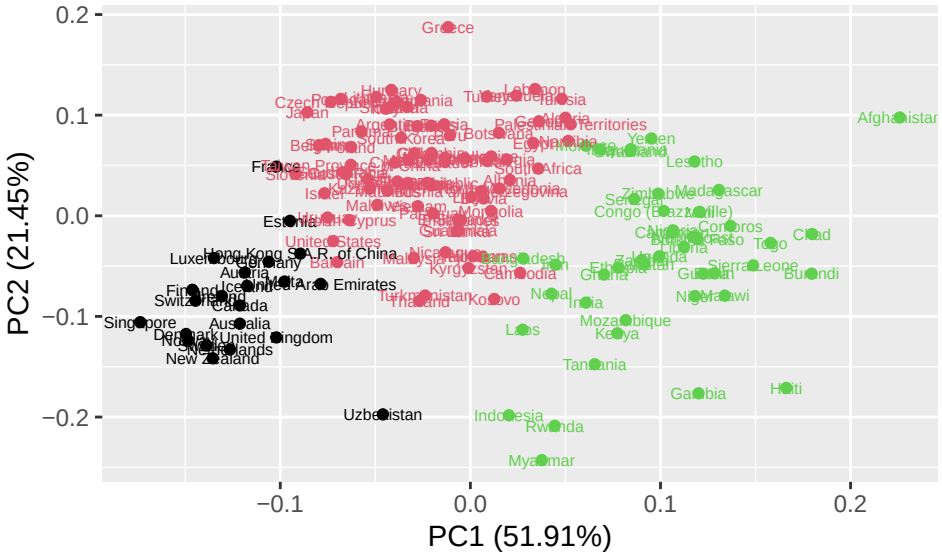
c)

- False

- False
- False
- True

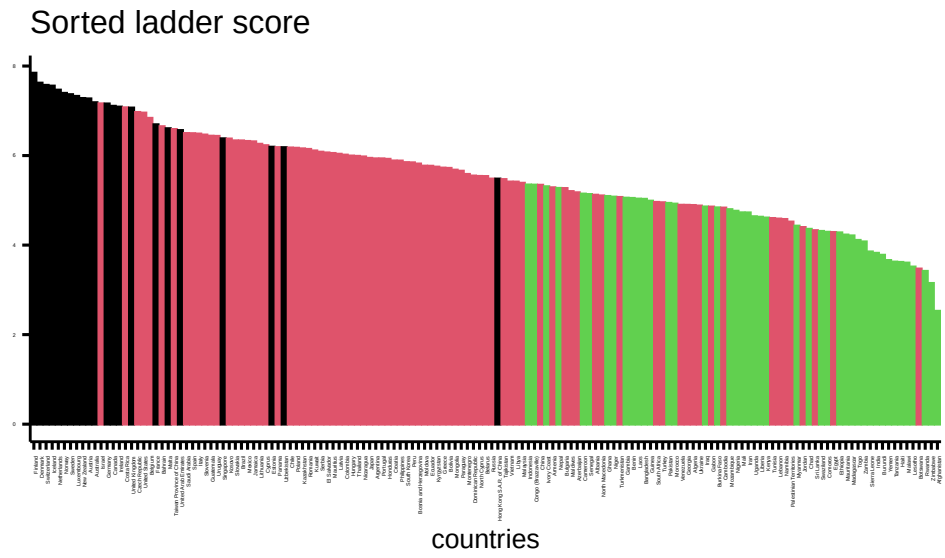
d)

```
set.seed(123)
K = 3  # your choice
km.out = kmeans(happiness.X, K)
autoplot(pca_mat, data = happiness.X, colour = km.out$cluster, label = T, label.size = 2,
    loadings = F, loadings.colour = "blue", loadings.label = F, loadings.label.size = 3)
```



Using k = 3, we satisfy the condition. USA is in a different cluster than the Scandinavian countries.

```
set.seed(123)
happiness.XY$countries <- rownames(happiness.XY)  #adding country names in separate coloumn. Makes the
ggbarplot(happiness.XY, x = "countries", y = "Ladder.score", main = "Sorted ladder score",
    fill = km.out$cluster, color = km.out$cluster, sort.val = "desc", sort.by.groups = FALSE,
    x.text.angle = 90, x.text.size = 1, ylab = FALSE) + theme(text = element_text(size = 2)) +
    theme(title = element_text(size = 10))
```



The K-means algorithm clusters the countries that share similarities within all predictors for the happiness. As you see in the plot above the countries at the top of the ladder is clustered together. With K=3 the algorithm finds three classes of countries. As one can see from the groupings developed countries with high welfare is to a degree clustered together and they have a high happiness score. On the other hand the clustering shows us that countries affected by social instability, conflicts and poverty experience less happiness. Of course there are outliers in the model, and we are not claiming that welfare and money are the only factors that influence happiness, but it does contribute.