Compulsory exercise 2: Group 18

TMA4268 Statistical Learning V2022

Thomas Rødland, Erlend Lokna

30 mars, 2022

```r
set.seed(1)
boston <- scale(Boston, center=T, scale=T)

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

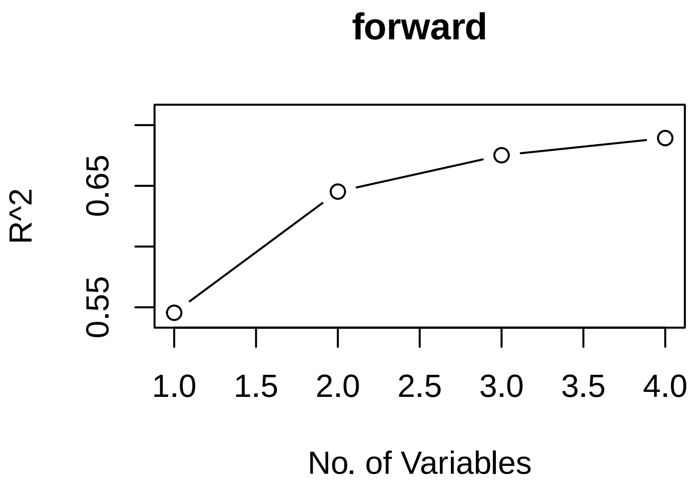
Problem 1

a)

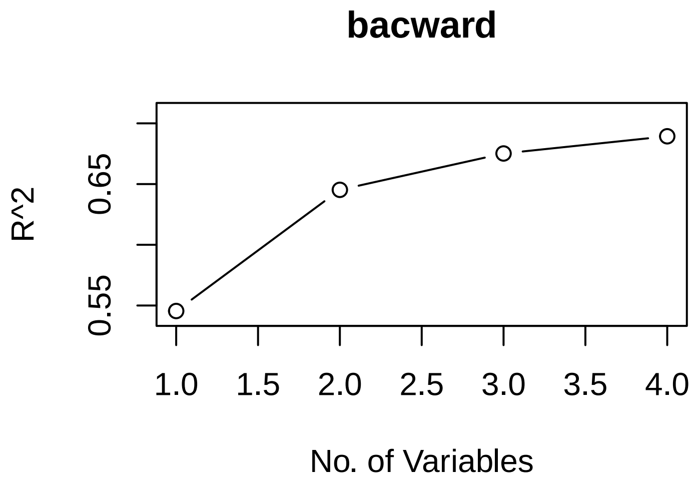
```r
#forward subset selection:
regfit.fwd = regsubsets(medv~., data = boston.train, nvmax=4, method = "forward")
regfit.bcwd = regsubsets(medv~., data = boston.train, nvmax=4, method = "backward")
```

plotting the R-squared vs number of predictors for forward and backwards:

```r
#plotting the R-squared vs number of predictors
plot(summary(regfit.fwd)$adjr2, xlab='No. of Variables',ylab='R^2',
     main="forward",type='b', xlim=c(1,4), ylim=c(0.54, 0.71))
```



```
plot(summary(regfit.bcwd)$adjr2, xlab='No. of Variables',ylab='R^2', ylim=c(0.54, 0.71),
     main="bacward",type='b', xlim=c(1,4))
```



b)

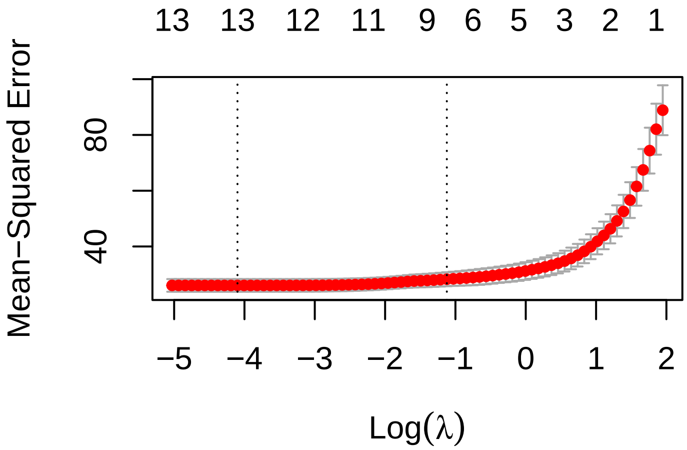
```
#the best four predictors:
coef(regfit.fwd, 4)
```

```
## (Intercept)          rm         dis     ptratio       lstat
##  0.02268276  0.34728504 -0.14653616 -0.21570329 -0.53034047
```

c)

```
#K-fold cross-validation (k=5)
#setup:
set.seed(123)
df.X.std <- scale(dplyr::select(Boston, -medv))
X.train <- as.matrix(df.X.std)[train.ind,] #converting to matrix.
X.test <- as.matrix(df.X.std)[-train.ind,]
Y.train <- Boston[train.ind, "medv"]
Y.train = as.matrix(Y.train)
Y.test <- Boston[-train.ind, "medv"]
Y.test = as.matrix(Y.test)

#calculation using cv.glmnet():
kf5_cv = cv.glmnet(x=X.train, y=Y.train, alpha = 1, nfolds = 5)
```

```
#plot:
plot(kf5_cv)
```



Lambda corresponding to minimal Mean-Squared error:

```
kf5_cv$lambda.min
```

```
## [1] 0.01658476
```

coefficients:

```
coef(kf5_cv, s=kf5_cv$lambda.min)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 22.74991687
## crim        -0.75916622
## zn           0.87882705
## indus        0.06224523
## chas         0.80268368
## nox         -1.62765265
## rm           2.87560832
## age         -0.10735056
## dis         -2.92307108
## rad          2.52855436
## tax         -1.95602522
## ptratio     -1.88338836
## black        0.95034028
## lstat       -3.94210652
```

d)

1. True
2. False
3. False
4. True

Problem 2

```
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)
```

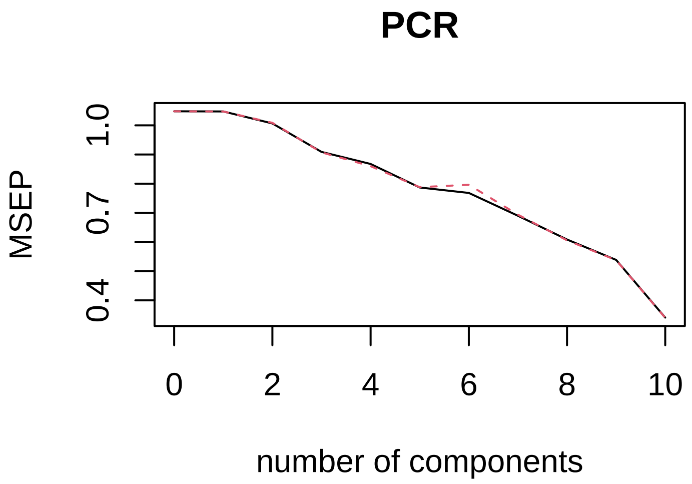
```
##             Y          X1          X2         X3         X4         X5
## 1 -1.43753239 -0.75905055 -0.69720326 -0.3016852 -0.7434697  0.8807558
## 2 -1.70972989 -0.28635632  0.04809182  0.5791725 -0.7446170  0.9935311
## 3  1.33931240  0.09574117 -0.89605758 -0.9636347  0.5554647 -0.5341800
## 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
```

a)

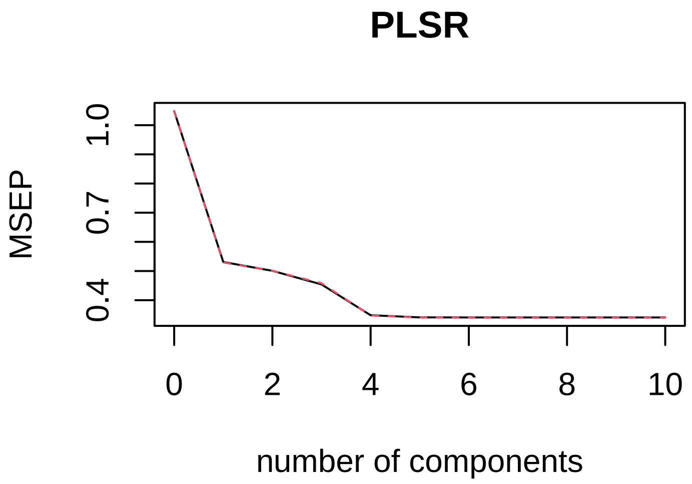
Fitting PCR and PLSR on the synthetic.train data set:

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

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



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

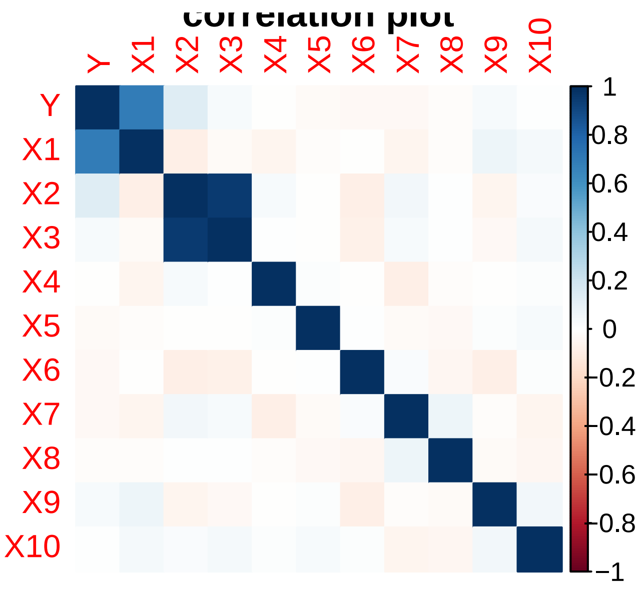


b)

We can clearly see that the PSLR method shrinks the MSEP quicker for fewer components when comparing to the PCR method. PCR is a unsupervised method while PLSR is supervised. Meaning that for a data set with correlated values, the PLSR will perform better.

Plotting the correlation:

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



We see that there is a correlation between X2 and X3, as shown by the blue squares in the figure above.

Problem 3

a)

1. True
2.
3. False (the extra term makes it less wiggly not more smooth?)
4. True (increase in k -> higher bias, lower variance ?)

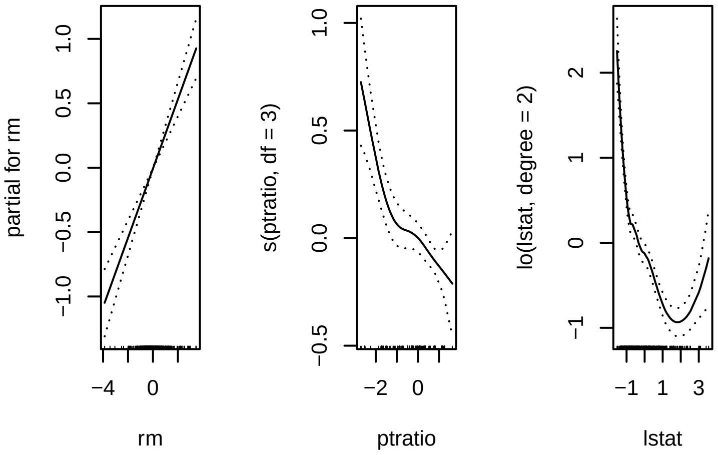
b)

using the gam-function to create an adaptive model:

```
adpt_mod <- gam(medv ~rm  + s(ptratio, df=3) + lo(lstat, degree=2, span=1), data=boston.train)
```

plotting results:

```
par(mfrow=c(1,3)) #to partition the Plotting Window
plot(adpt_mod, se=TRUE)
```



Problem 4

a)

1. False (?)
2. True
3. True
4. False

b)

See figure below:

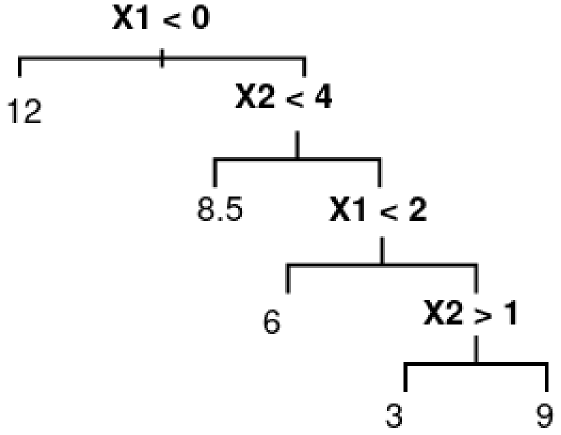


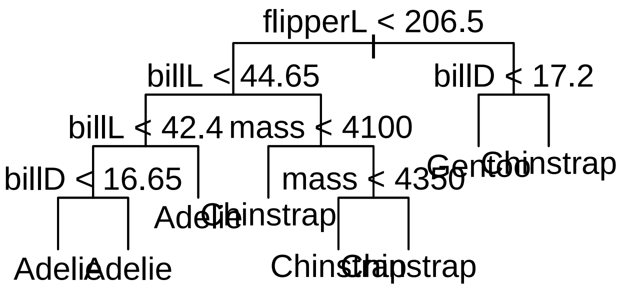
Figure 1: Sketch

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)
    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, ]
```

```
penguins.tree <- tree(species ~ ., data=train, split = "gini")
plot(penguins.tree, type="uniform")
text(penguins.tree, pretty = 0)
```

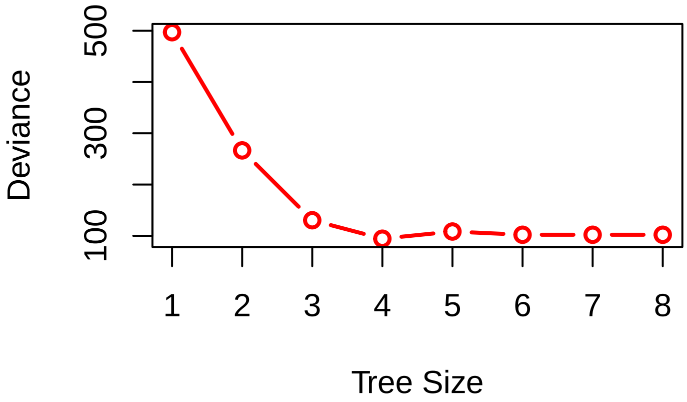
i)



```
set.seed(123)
#10 fold
cv.penguins <- cv.tree(penguins.tree, K = 10)
```

```
plot(cv.penguins$dev ~ cv.penguins$size, type = "b", lwd = 2, col = "red",
    xlab = "Tree Size", ylab = "Deviance")
```

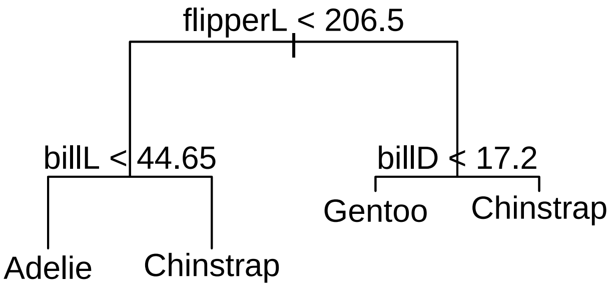
ii)



We see that the optimal tree contains 4 leaves. Creating the optimal tree with 4 leaves:

```
prune.penguins <- prune.tree(penguins.tree, best = 4)
plot(prune.penguins, style="uniform", type = "proportional")
text(prune.penguins, pretty=0)
```

iii)



d)

classification tree based on a more advanced method?

Problem 5

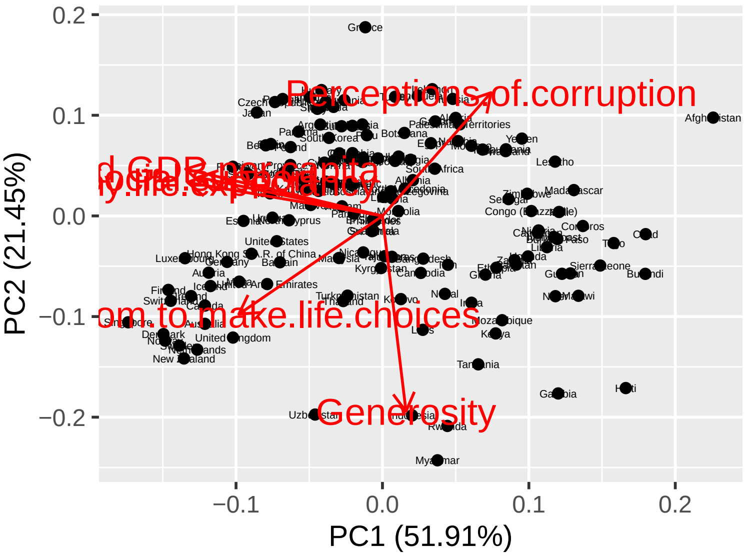
a)

1. True?
2.
3.
4.

Problem 6

```
# 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

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))

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 = 1.5)
```



a)

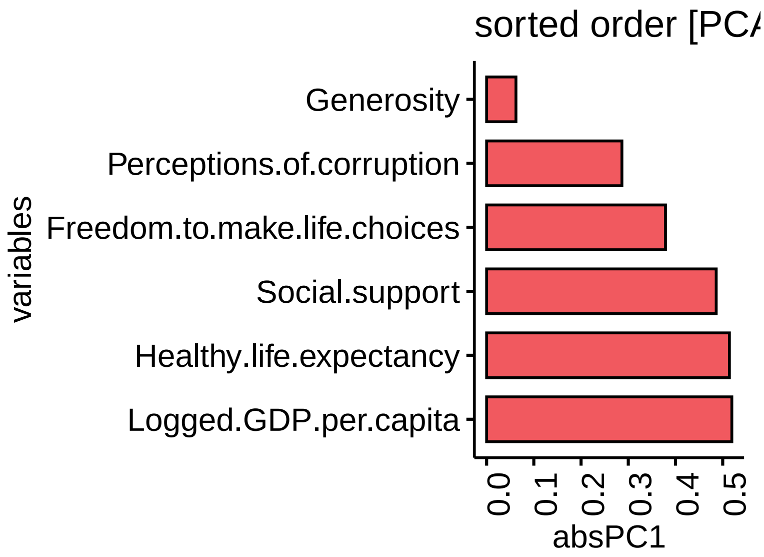
i) First of all we observe that the "logged GDP", "social support", "healthy life expectancy" and "ladder score" practically have the same direction in the plot above. This implies a correlation, and we can assume that each one effects the other. We also observe that "freedom to make life choices" and "perception of corruption" is close to opposite in direction. This may be interpreted as countries having more freedom experience less corruption. Also there is no relation between corruption and generosity as the angle between the two directions are sufficient. The same can be said about the generosity and "logged GDP", "social support", "healthy life expectancy" and "ladder score".

ii)

b)

i)

```
library(ggpubr)

plotData <- data.frame(absPC1 = abs(data.frame(pca_mat$rotation)$PC1),
                       variables = c(colnames(happiness.X)))

ggbarplot(plotData, x = "variables", y = "absPC1",
          main="sorted order [PCA]",
          fill="#f1595f",
          sort.val = "desc",
          sort.by.groups = FALSE,
          x.text.angle = 90,
          orientation="horizontal"
          )
```

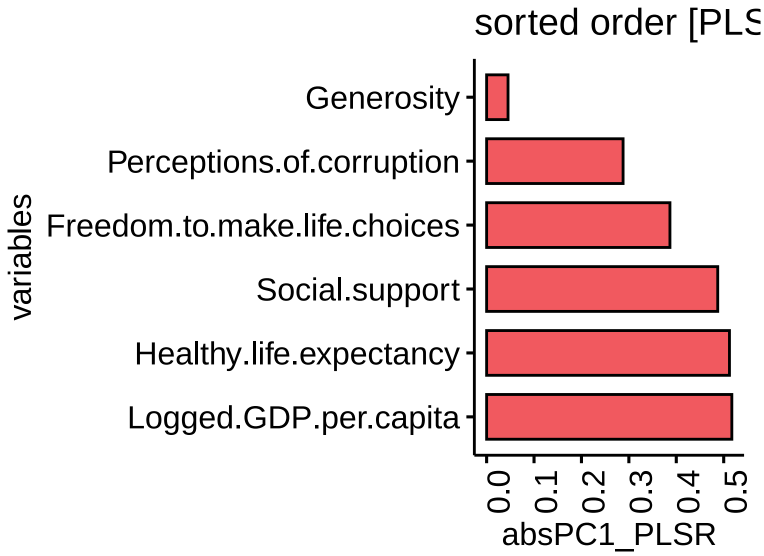


ii)

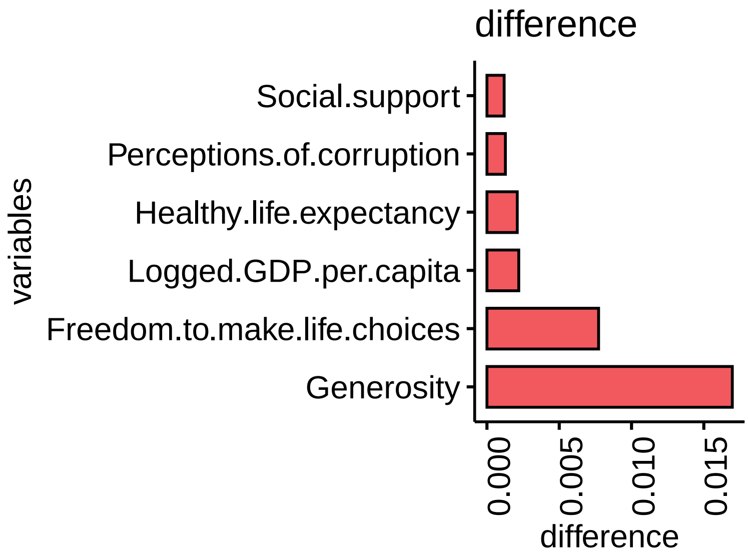
```
plsr_mat <- plsr(Ladder.score~., data = happiness.XY, scale=T)
```

iii)

```
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"
          )
```



```
plotData$difference <- abs(plotData$absPC1 - plotData$absPC1_PLSR)

ggbarplot(plotData, x = "variables", y = "difference",
          main="difference",
          fill="#f1595f",
          sort.val = "desc",
          sort.by.groups = FALSE,
          x.text.angle = 90,
          orientation="horizontal"
          )
```



iv)

The three most important predictors are:

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

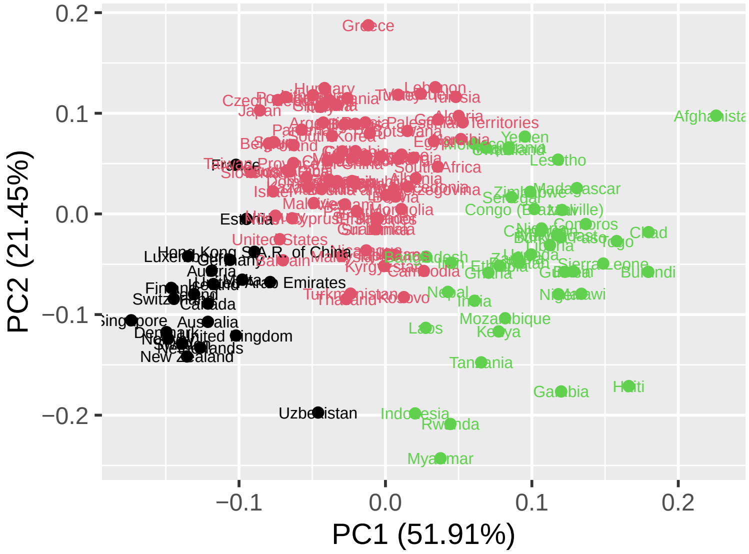
c)

1. False
2. False (https://datascience.stackexchange.com/questions/63697/will-k-means-clustering-converge-to-the-same-results-given-the-same-data-set)
3.
4. True

d)

i)

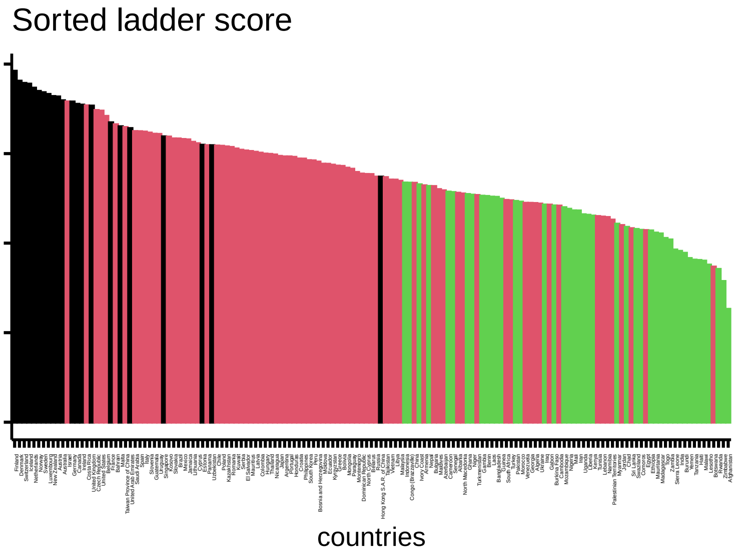
```
K = 3  # your choice
km.out = kmeans(happiness.X, K)

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.

ii) To answer this question we can sort the ladder scores and visualise the cluster results:

```
happiness.XY$countries <- rownames(happiness.XY) #adding country names in separate coloumn. Makes the p

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 ladder score. 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.