Machine Learning - Exercise 3

Robin Perälä 175910

18.4.2022

Task 1 (short questions)

Answer the following questions with short answers. Motivate your answers. Answer with no more than 10 sentences.

a) *What is the main difference between supervised learning models and unsupervised learning models?*

Supervised learning models try to predict the effect of independent variables on the dependent variable y.

Unsupervised learning models do not have a dependent variable. They can instead be used to understand the data, e.g. by clustering.

b) *What are the main differences (e.g., clusters, starting point for algorithm, algorithm itself, etc.) between k-means clustering and hierarchical clustering?*

K-means clustering partitions the data into a specified amount of clusters k, while the number of partitions is not a parameter of the hierarchical clustering algorithm. Instead the hierarchical clustering creates a dendrogram which can afterwards be cut into clusters. The leaves with the shortest branches in the dendrogram represent observations that are similar to each other.

K-means clustering starts by randomly assigning data points into k clusters. After that the algorithm reassignes the data points iteratively into the cluster with the nearest centroid, until finally finding the local minimum of the within sum of squares.

There is no random assignment for hierarchical clustering. Instead it starts by treating all data points as their own cluster. Then the algorithm one by one checks the distance (e.g. Euclidean) between all clusters and merges the closest two clusters into one. When there's only one cluster left, the algorithm is complete.

Both try to minimize a measure for the distance within clusters, e.g. Euclidean.

c) *Give an example of where an unsupervised model is used in real-world setting such as a company (come up with an own example).*

Cluster analysis can be used for finding people with similar interests. On a simple level: people who like to watch baseball on tv would be put into one group. On a more advanced level: people who have similar personalities and are emotionally in a similar situation would be put into the same group. This could be extended for finding dating partners. (Although a possible issue could be seen in only meeting similar people. Meeting different people could be seen as a good thing)

Some financial applications would be to cluster stocks with similar characteristics or clustering the economy into segments

Task 2 (PCA)

Load data and preparation

```
rm(list=ls())   #clear environment
set.seed(707)     #random seed for reproducibility

library(haven)  #For importing Stata data
library(tibble) #For transforming columns to rownames
library(glmnet) #For Lasso and Ridge regression
library(pls)    #For principal component regression

#Load Stata data with read_dta function from haven
USCompanies_data_winsorized = read_dta("USCompanies_data_winsorized.dta")
USDataNoNA = na.omit(USCompanies_data_winsorized)
```

a) *Your first task is to assign the variable conm as the row identifier (this can be done using e.g., "column_to_rownames" option in "tibble" and "tidyverse" packages). If you do this step correctly, you should see the companynames as row identifiers, and their names should show in the PCA plot.*

```
USData = column_to_rownames(USDataNoNA, var = "conm")
```

b) *Estimate a Principal Component Analysis (PCA). Report the principal component loadings for the first three principal components.*

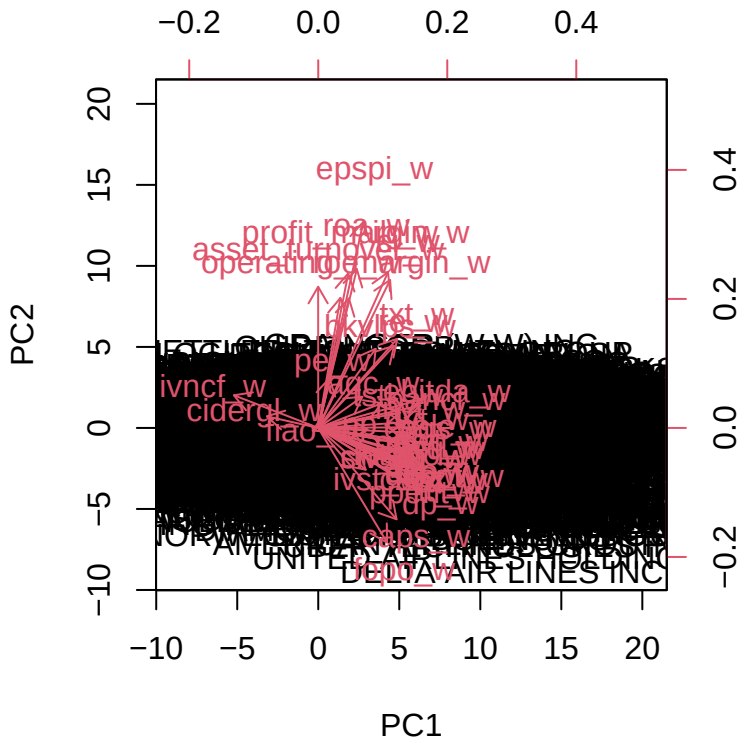
```
myPCA = prcomp(USData, scale=TRUE)
PCA123 = myPCA$rotation[,c("PC1", "PC2", "PC3")]
PCA123
```

```
##                          PC1          PC2           PC3
## pe_w               0.0219053930  0.096371165 -0.0113914291
## aco_w              0.1776612005 -0.074228021  0.0877447857
## ap_w               0.1719006527 -0.051919539 -0.0242413464
## aqc_w              0.0855188921  0.060476459  0.0457296191
## at_w               0.2048657950 -0.085942439 -0.0052405004
## bkvlps_w           0.1111887708  0.153368733 -0.0288619552
## caps_w             0.1517004115 -0.176998257  0.1094768160
## capx_w             0.1789824182 -0.088686383  0.0531349404
## ceq_w              0.1921768335 -0.028127299 -0.0536498852
## ch_w               0.1749259936 -0.066191931  0.0281891406
## chech_w            0.1321019068 -0.047864361 -0.0004230128
## ci_w               0.1392220782  0.286712728 -0.2990872322
## cidergl_w         -0.0994198160  0.022449699 -0.0788968769
## cogs_w             0.1854856866 -0.009457599  0.0331211840
## dlc_w              0.1665910435 -0.074068654  0.0272017497
## dp_w               0.1892199917 -0.119991818  0.0995393876
## dvt_w              0.1686655624  0.010812345 -0.0836297670
## ebitda_w           0.1993147086  0.055591935 -0.0697150564
## epspi_w            0.0873125428  0.399092905 -0.2082262110
## fiao_w            -0.0107227055 -0.005257862 -0.1380125128
## fopo_w             0.1334079351 -0.222146344  0.2234302168
```

```
## gdwl_w              0.1707580016 -0.025926553  0.0501196530
## ib_w                0.1351812137  0.301705620 -0.2925063687
## icapt_w             0.2034869158 -0.079032558  0.0088316702
## intano_w            0.1539221029 -0.043325791  0.1023650985
## invt_w              0.1580719947  0.029264332  0.0370626773
## ivncf_w            -0.1633353760  0.063250657  0.0041930886
## ivst_w              0.0919001304 -0.080960271 -0.0795657645
## lco_w               0.1837586794 -0.072741355  0.0796601099
## lt_w                0.1990884680 -0.097991537  0.0137156035
## np_w                0.0956312681 -0.003356195 -0.0557914022
## oancf_w             0.1980089884  0.043335793 -0.0600022765
## ppent_w             0.1724903568 -0.104748192  0.0760749797
## re_w                0.1467670753  0.159193200 -0.1860158151
## rect_w              0.1762904057 -0.033354446 -0.0485021658
## sale_w              0.1993068839  0.001383857  0.0278366727
## siv_w               0.0956890308 -0.048015493 -0.1208591105
## teq_w               0.1927555741 -0.036123631 -0.0492601012
## tstk_w              0.1234700453  0.048056030 -0.0238159543
## txt_w               0.1530371693  0.172670944 -0.1919250518
## roa_w               0.0747814333  0.307681924  0.3216677329
## roe_w               0.0593391283  0.250963229  0.1396934837
## profit_margin_w     0.0580653859  0.299364789  0.4280341877
## asset_turnover_w   -0.0002174737  0.273567318  0.2571566797
## operating_margin_w  0.0428551354  0.251900942  0.3855766734
```

c) *Plot a principal component plot, where the x-axis show PC1 loadings and the y-axis shows PC2 loadings. The command for this is "biplot()". (Note that the plot will most likely look a bit messy as there as so many companies in the database)*

```
biplot(myPCA, scale=0)
```



d) *Calculate the proportion of variance explained (PVE) for the first three principal components. Report these PVEs, as well as the cumulative PVE.*

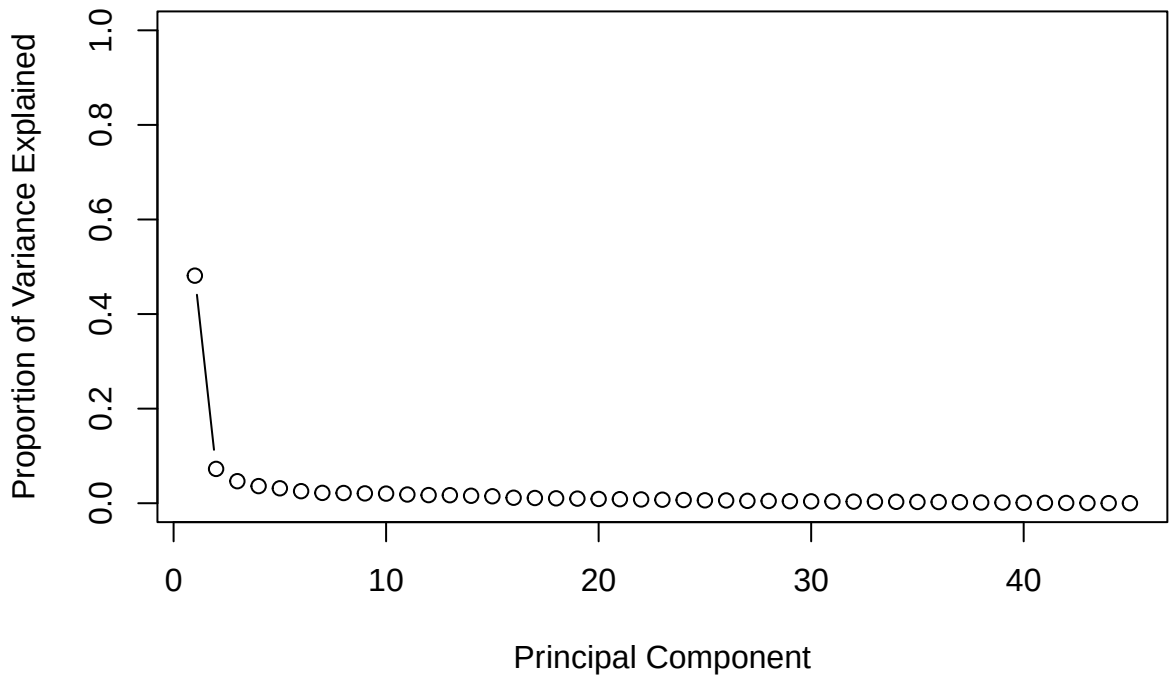
```
#Variance
myPCA$var=(myPCA$sdev)^2

#PVE for all PCs
PVE = myPCA$var/sum(myPCA$var)

#PVE for the first three PCs and cumulative PVE
PVE123 = PVE[1:3]
cumsum123 = cumsum(myPCA$var[1:3])/sum(myPCA$var)

PVEtable = rbind(round(PVE123, 4),
                 round(cumsum123, 4))

colnames(PVEtable) = colnames(PCA123)
rownames(PVEtable) = c("PVE", "Cumulative PVE")
print(PVEtable)
```

```
##                   PC1    PC2    PC3
## PVE            0.4812 0.0726 0.0466
## Cumulative PVE 0.4812 0.5538 0.6003
```

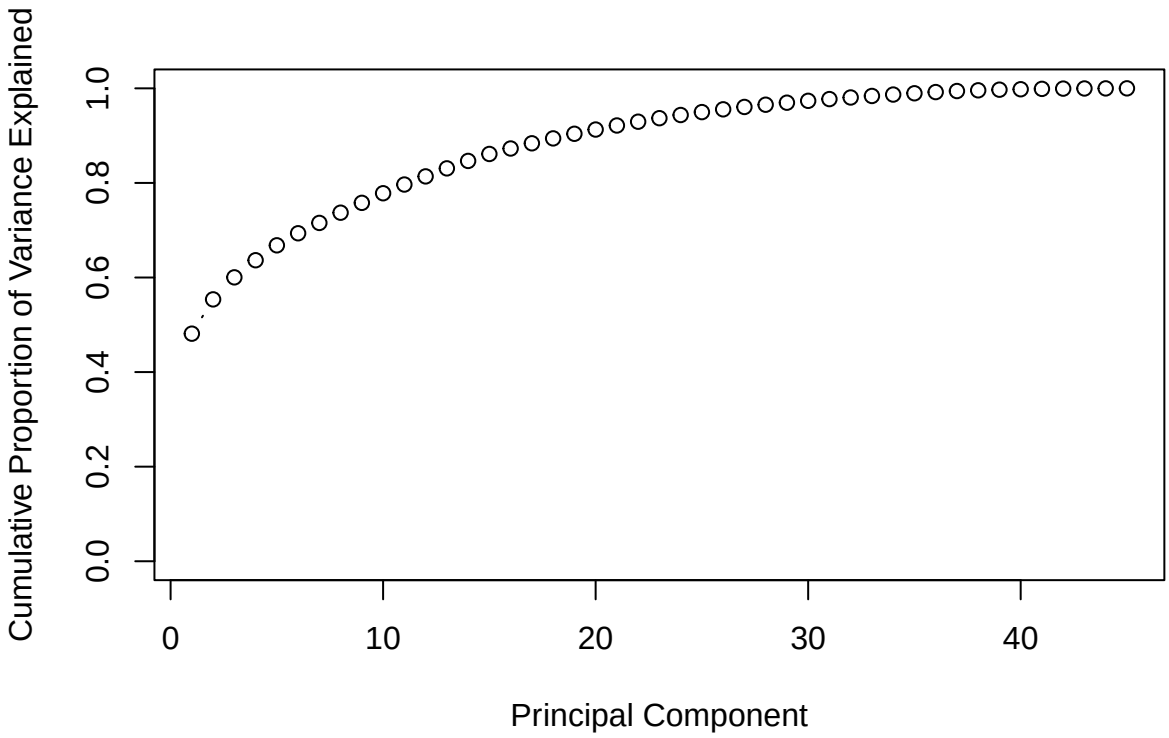
e) *Show plots for the PVE and the cumulative PVE (so-called "scree plots") respectively. The x-axis*

should show the number of principal components and the y-axis should show the percentage of variance explained (PVE).

```
plot(PVE, xlab="Principal Component", ylab="Proportion of Variance Explained",
     ylim=c(0,1), type="b")
```




```
plot(cumsum(PVE), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained",
     ylim=c(0,1), type="b")
```



f) *Searching for the "elbow" in a scree plot is an ad-hoc approach of choosing the number of important principal components. Based on your plots in e), how many principal components appear to be important (i.e., where is the "elbow" in the scree plot)?*

Searching for the "elbow" means eye-balling an optimal amount of principal components. This is done by looking at their proportion of variance explained and seeing where the marginal benefit of adding another component diminishes.

Looks like after the first component the marginal benefit is small. To be on the safer side we will use two variables

Task 3 (Cluster analysis)

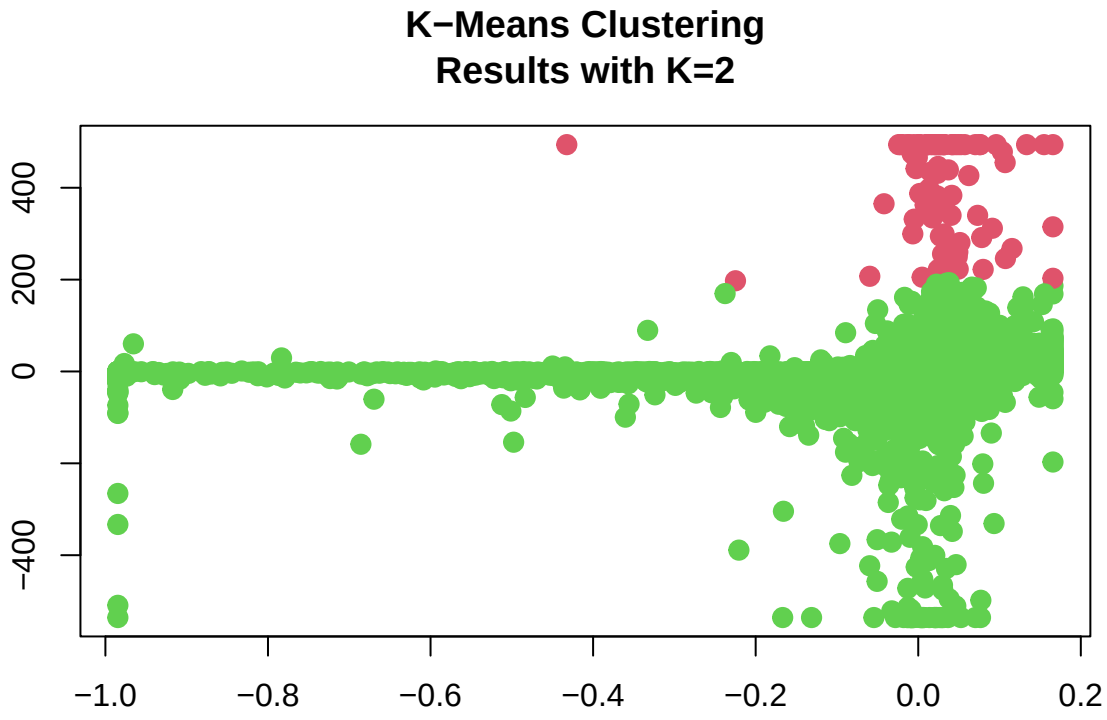
Cluster analysis. In this step, use only information for two variables, namely, roa_w and pe_w.

a) *Use k-means clustering to separate firms into two clusters (K=2). Use 1 random assignment. Show a plot of the results.*

```
x=USData[,c("roa_w", "pe_w")]
kmOut2C1R=kmeans(x, centers=2, nstart=1)
head(kmOut2C1R$cluster)
```

```
##                    AAR CORP AMERICAN AIRLINES GROUP INC
##                           2                           2
##     CECO ENVIRONMENTAL CORP  PINNACLE WEST CAPITAL CORP
##                           2                           2
##           PROG HOLDINGS INC         ABBOTT LABORATORIES
##                           2                           2
```

```
plot(x, col=(kmOut2C1R$cluster+1), main="K-Means Clustering
Results with K=2", xlab="", ylab="", pch=20, cex=2)
```

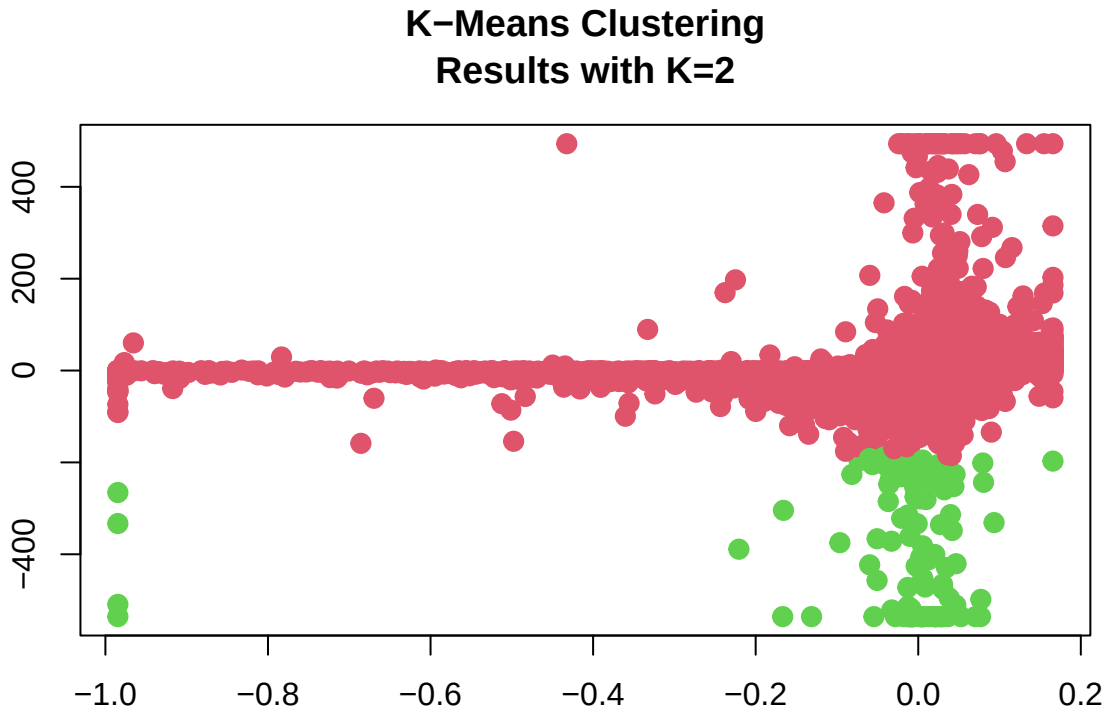


b) *Use k-means clustering to separate firms into two clusters (K=2). Use 10 random assignments. Show a plot of the results.*

```
kmOut2C10R=kmeans(x, centers=2, nstart=10)
head(kmOut2C10R$cluster)
```

```
##                    AAR CORP AMERICAN AIRLINES GROUP INC
##                           1                           1
##     CECO ENVIRONMENTAL CORP  PINNACLE WEST CAPITAL CORP
##                           1                           1
##           PROG HOLDINGS INC         ABBOTT LABORATORIES
##                           1                           1
```

```
plot(x, col=(kmOut2C10R$cluster+1), main="K-Means Clustering
Results with K=2", xlab="", ylab="", pch=20, cex=2)
```



c) *Compare the within-cluster sum of squares for the clustering results in a) versus b). What do these numbers explain and why do they differ (note that they do not have to differ).*

```
kmOut2C1R$withinss
```

```
## [1]   958869.8 23748046.3
```

```
kmOut2C10R$withinss
```

```
## [1] 19778717  1945397
```

The within-cluster sum of squares measures the squared Euclidean distance between all data points in the cluster. All these distances are then summed to obtain the within-cluster sum of squares

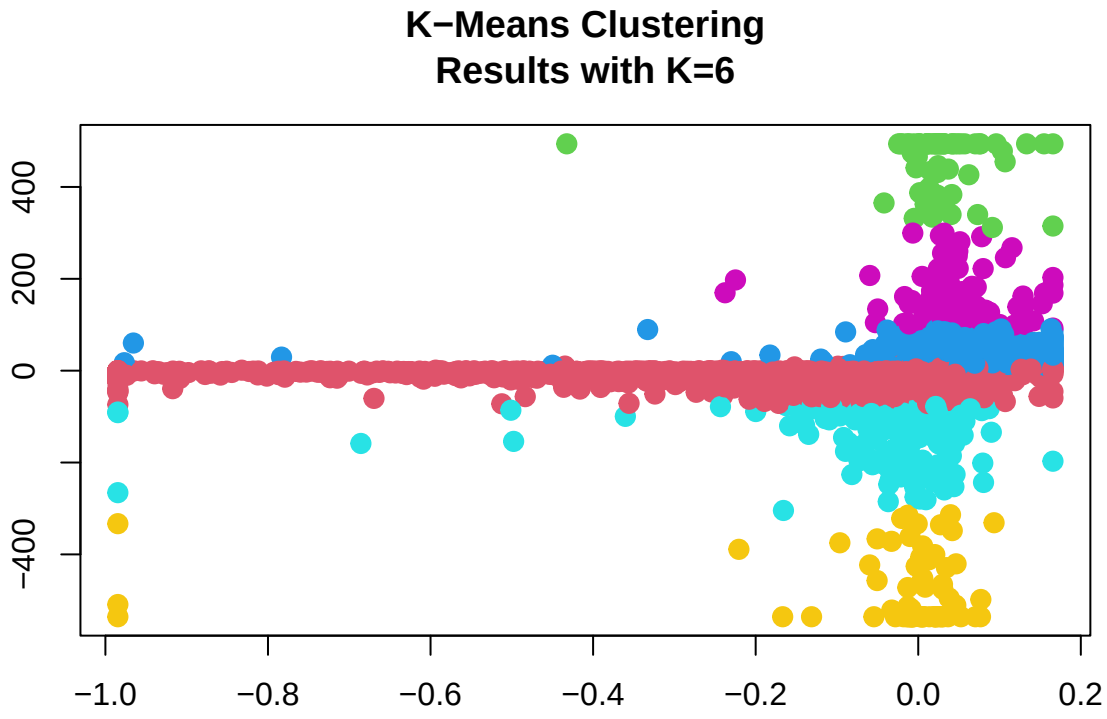
The results are exactly the same. They do not differ using our random seed. The results depend on which seed we are using. When using 10 random assignments and choosing the best one, the result is usually more stable and less dependent on randomness. The model using only one assignment will be more dependent on randomness.

d) *Use k-means clustering to separate firms into six clusters (K =6). Use 10 random assignments. Show a plot of the results.*

```
kmOut6C10R=kmeans(x, centers=6, nstart=10)
head(kmOut6C10R$cluster)
```

```
##                    AAR CORP AMERICAN AIRLINES GROUP INC
##                           3                           1
##     CECO ENVIRONMENTAL CORP  PINNACLE WEST CAPITAL CORP
##                           3                           3
##           PROG HOLDINGS INC         ABBOTT LABORATORIES
##                           1                           3
```

```
plot(x, col=(kmOut6C10R$cluster+1), main="K-Means Clustering
Results with K=6", xlab="", ylab="", pch=20, cex=2)
```



Task 4 (Lasso, Ridge and PCR)

In this exercise, we will compare the prediction performance of Lasso regression, Ridge regression, and Principal Component Regression (PCR) in predicting roa_w. Since the sample is rather large, we will hold out a test data set throughout this question. This will be used to compare the performance of the three models.

To do this, use the validation set approach to split the original sample (used in question 2) into two parts. Assign half (or some portion) of the observations to a training set, and assign the remaining observations to the test set. Keep the test set the same throughout this question.

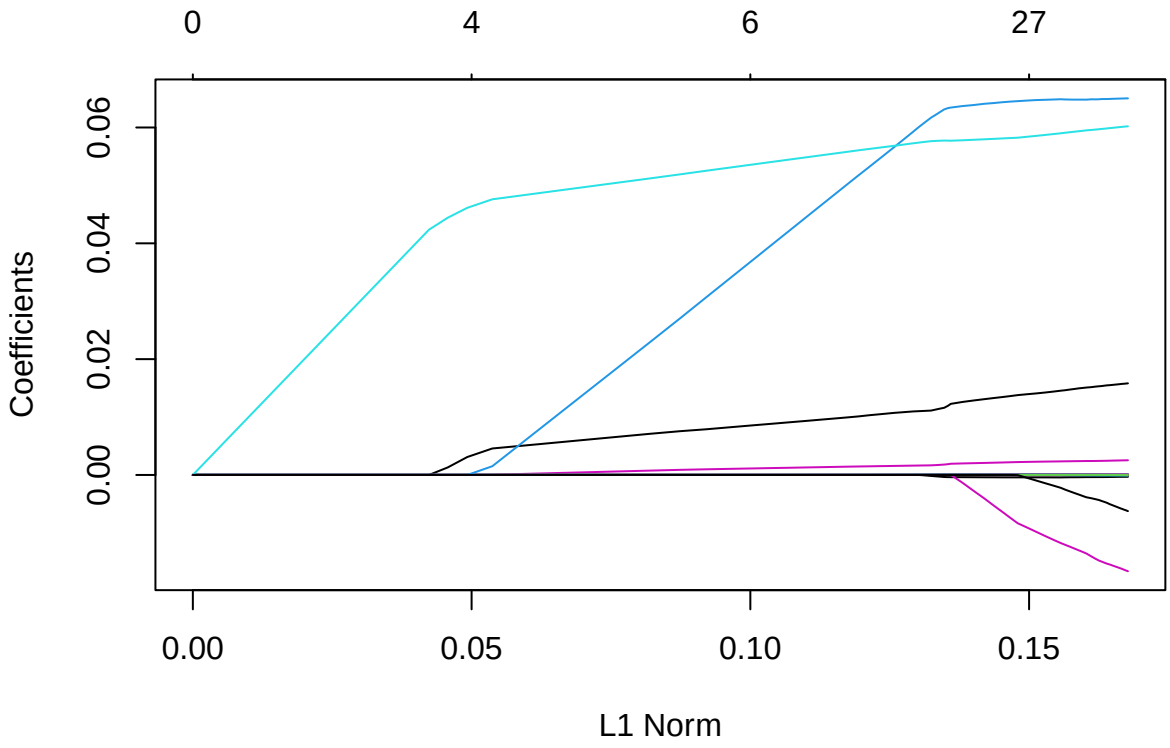
```
y = USData$roa_w
x = model.matrix(roa_w~., USData)[,-1]

#Validation set (50-50 split)
train=sample(c(TRUE, FALSE), nrow(USData), rep=TRUE)
test = !train
```

a) *Train the model on the training data using Lasso regression. Find the best lambda (tuning parameter).*

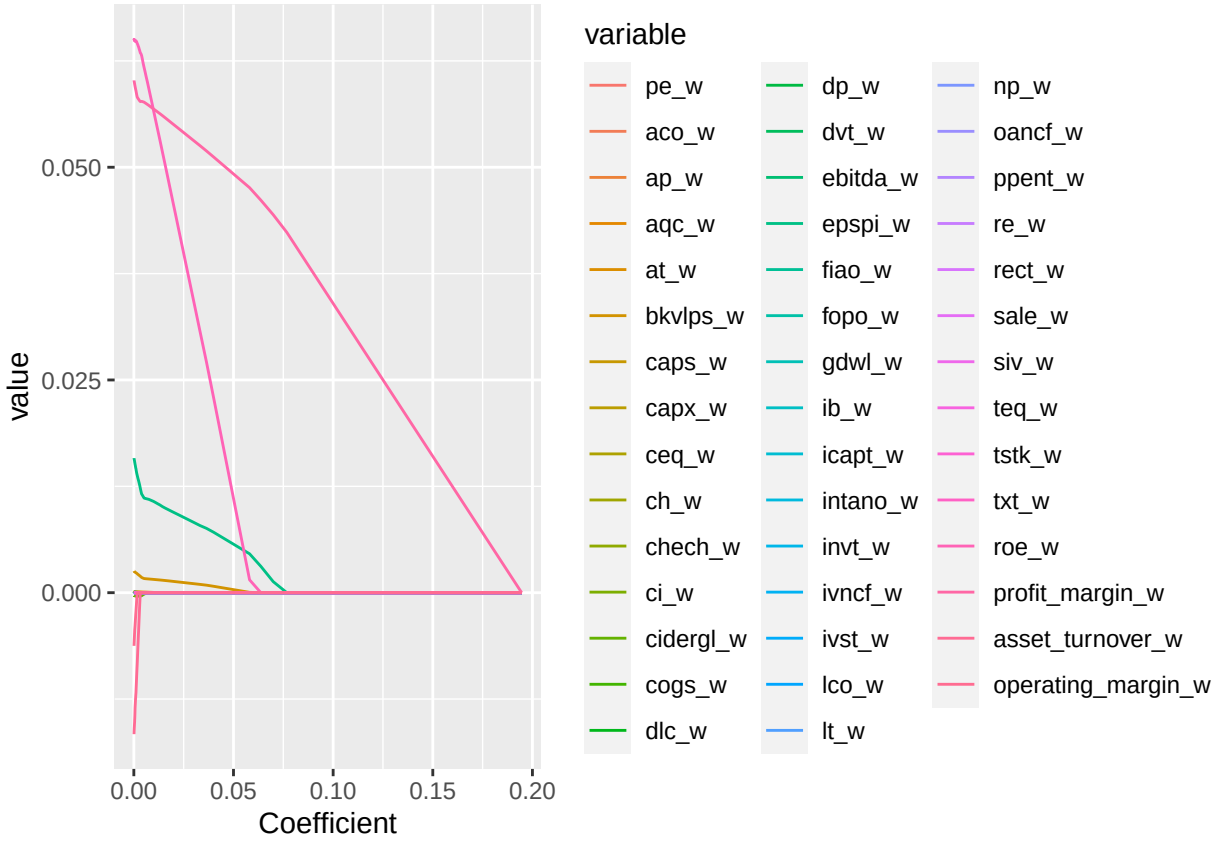
```
#Use function glmnet (from library glmnet) to perform Lasso regression. Alpha=1 is Lasso
lasso = glmnet(x[train,], y[train],
               alpha=1, standardize=TRUE)

#Find best lambda using 10-fold Cross-Validation
lassoCV=cv.glmnet(x[train,], y[train], alpha=1, standardize=TRUE, nfolds=10)

#Best tuning parameter (lambda)
signif(lassoCV$lambda.min, 4)
```

```
## [1] 0.0008041
```

```
#Plot the coefficients against L1 norm
plot(lasso)
```




```
#Plot the coefficients against lambda
#ggplot for nicer plotting
library(ggplot2); library(data.table)
coefNLambda = as.data.frame(t(as.matrix(coef(lasso)[-1,])))
coefNLambda$Lambda = lasso$lambda

coefNLambda = melt.data.table(data=data.table(coefNLambda),
                              id.vars='Lambda',
                              variable.name='variable')

ggplot(data=coefNLambda, aes(x=Lambda, y=value, col=variable)) + geom_line() + xlab("Coefficient")
```

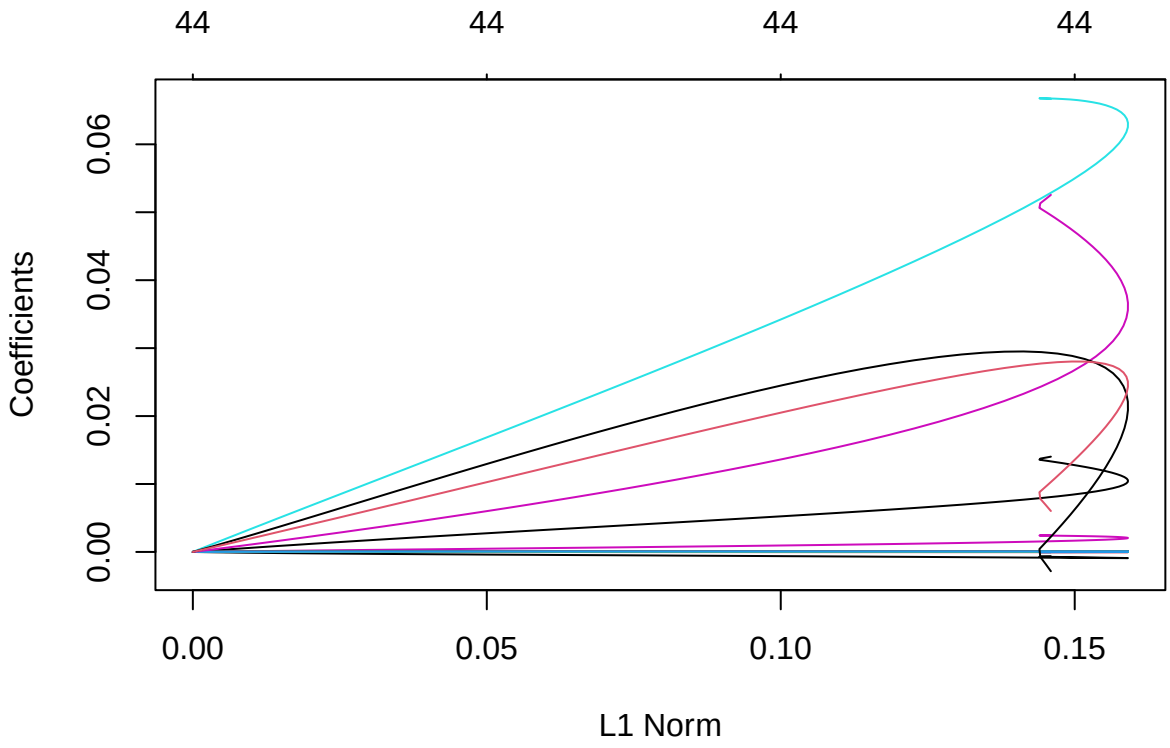


b) *Train the model on the training data using Rdige regression. Find the best lambda (tuning parameter).*

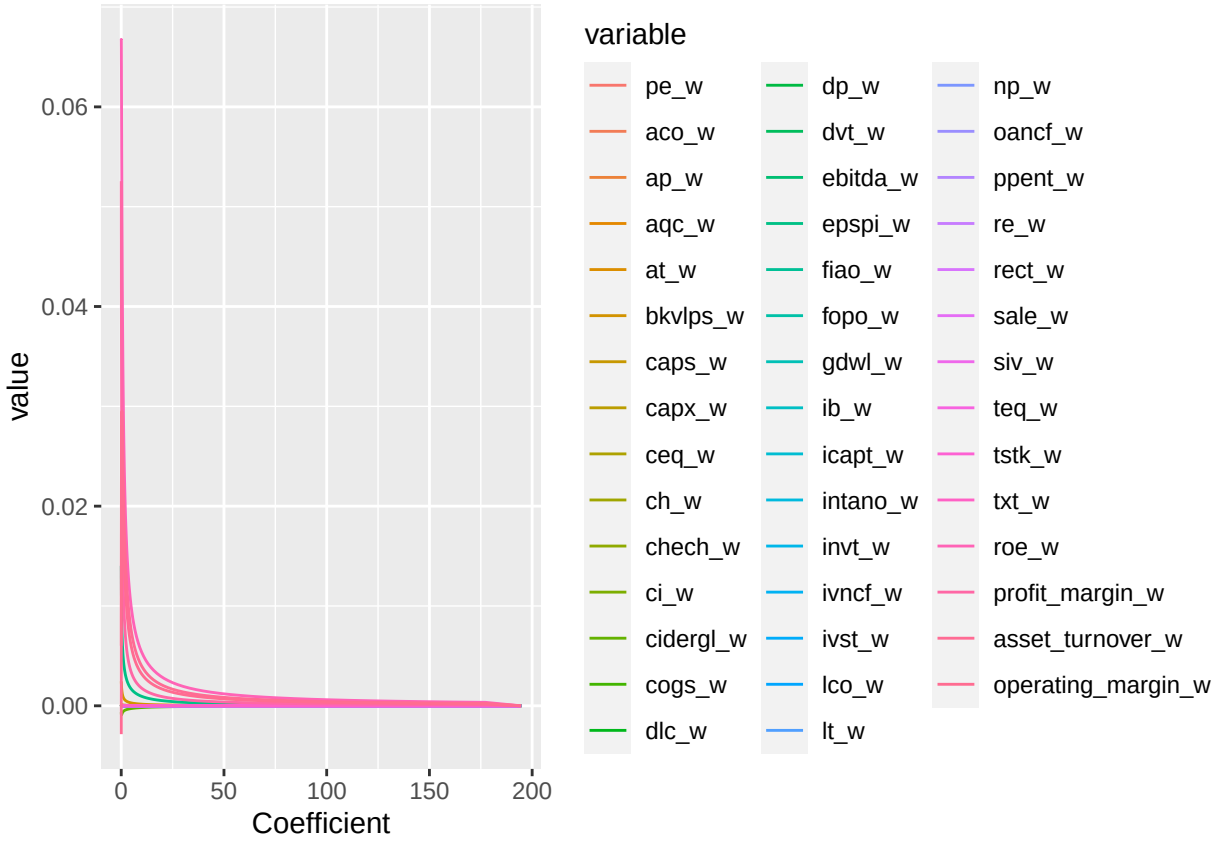
```
#Use function glmnet (from library glmnet) to perform Ridge regression. Alpha=0 is Ridge
ridge = glmnet(x[train,], y[train],
               alpha=0, standardize=TRUE)

#Find best lambda using 10-fold Cross-Validation
ridgeCV=cv.glmnet(x[train,], y[train], alpha=0, standardize=TRUE, nfolds=10)

#Best tuning parameter (lambda)
signif(ridgeCV$lambda.min, 4)
```

```
## [1] 0.01946
```

```
#Plot the coefficients against L1 norm
plot(ridge)
```

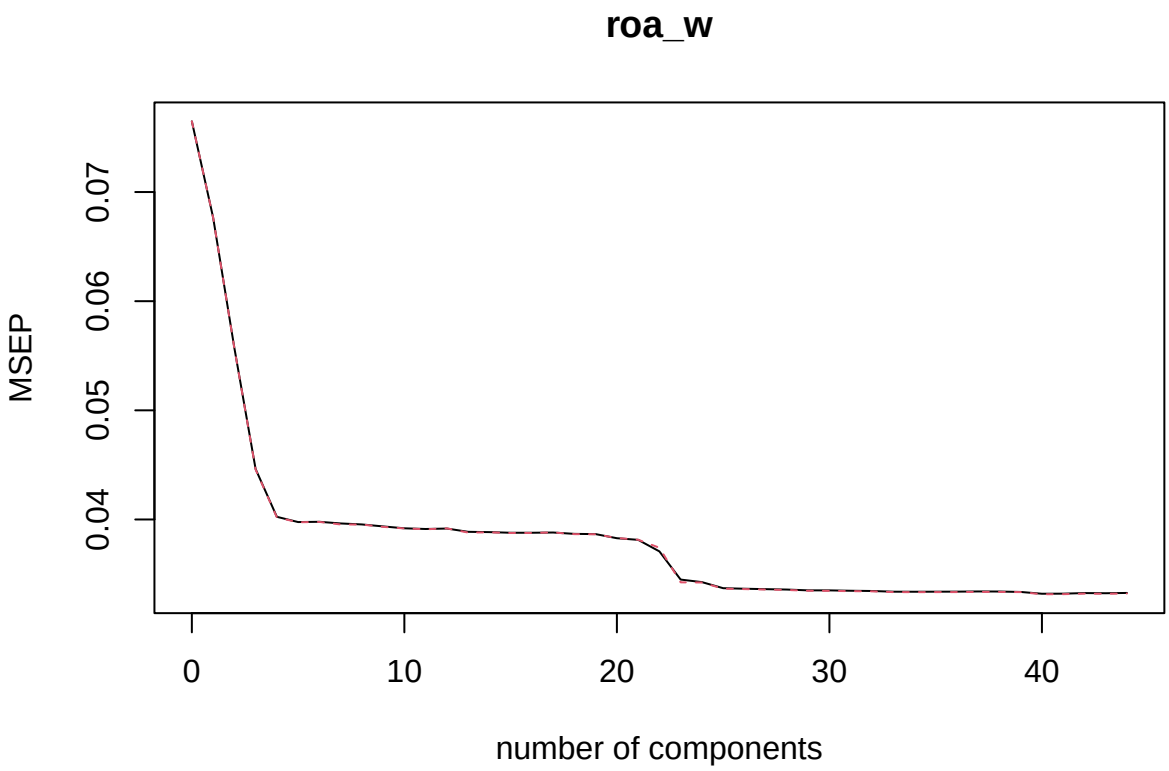



```
#Plot the coefficients against lambda
#ggplot for nicer plotting
library(ggplot2); library(data.table)
coefNLambda = as.data.frame(t(as.matrix(coef(ridge)[-1,])))
coefNLambda$Lambda = ridge$lambda

coefNLambda = melt.data.table(data=data.table(coefNLambda),
                              id.vars='Lambda',
                              variable.name='variable')

ggplot(data=coefNLambda, aes(x=Lambda, y=value, col=variable)) + geom_line() + xlab("Coefficient")
```

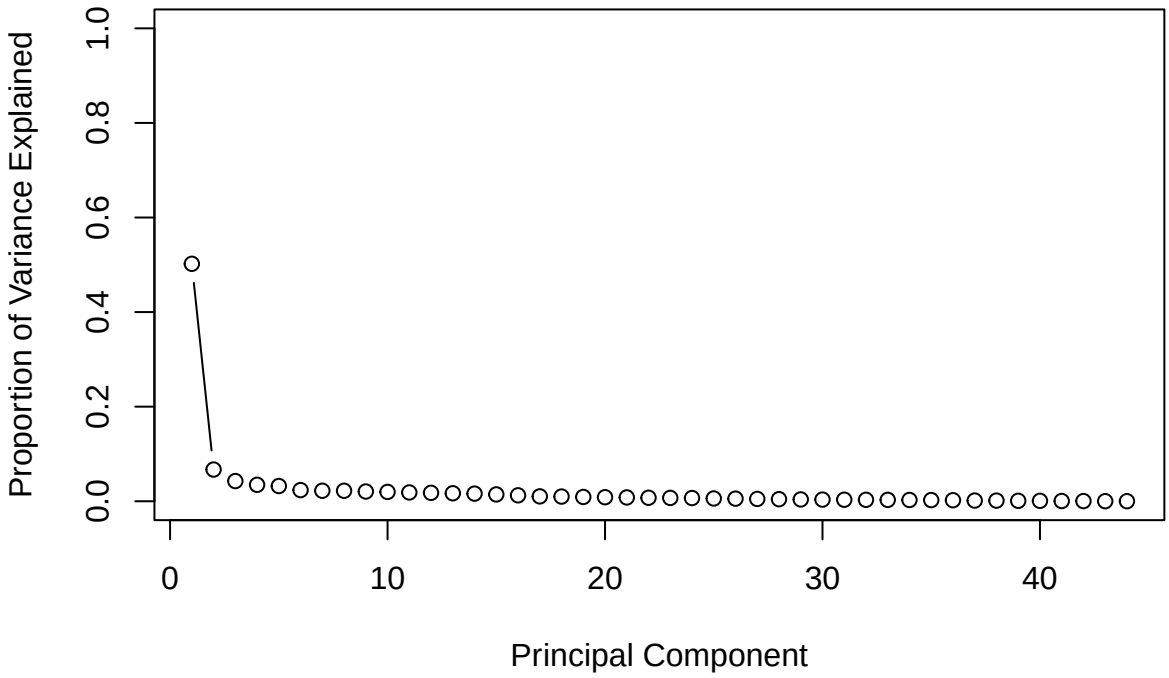


c) *Train the model on the training data using Principal Component Regression (PCR). Present a validation plot. Use the scree plot to decide on an optimal number of principal components by identifiying where there is a "elbow" in the scree plot. Report this number.*

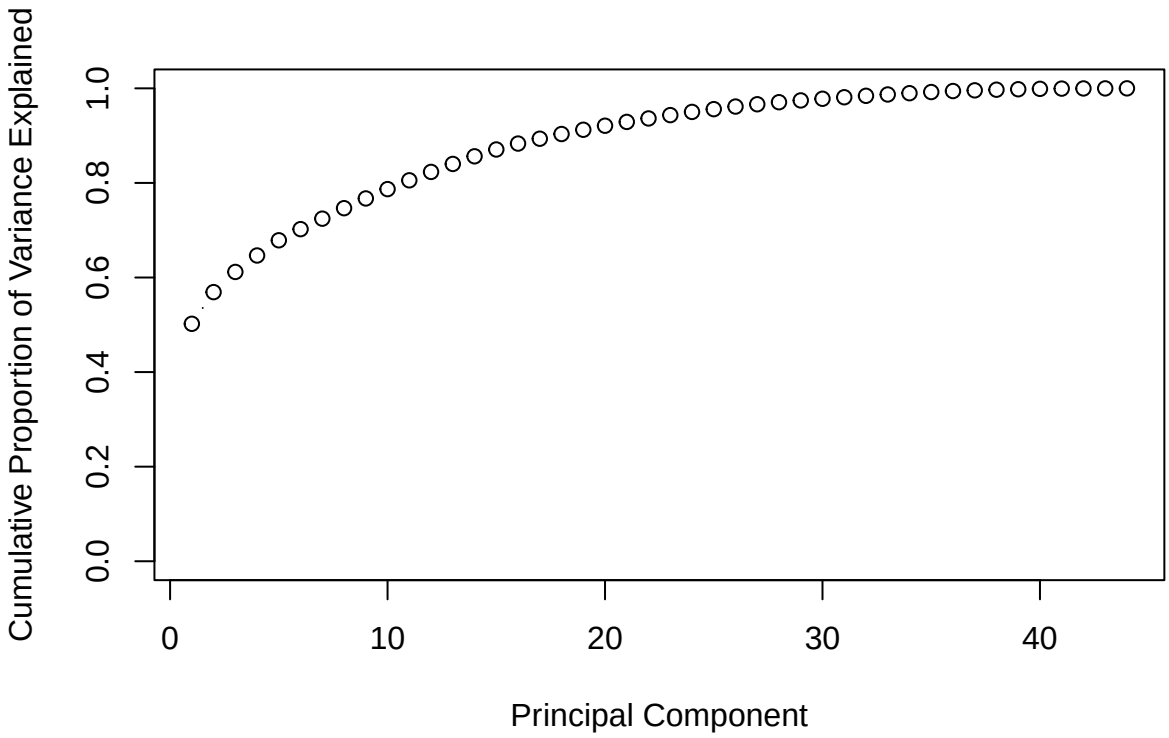
```
#PCR Regression using function pcr (part of pls library)
PCRfit=pcr(roa_w~., data=USData, scale=TRUE, subset=train, validation ="CV")

#Validation plot
validationplot(PCRfit, val.type="MSEP")
```




```
#Scree plotting
PVE=PCRfit$Xvar/PCRfit$Xtotvar
plot(PVE, xlab="Principal Component", ylab="Proportion of Variance Explained",
     ylim=c(0,1), type="b")
```




```
plot(cumsum(PVE), xlab="Principal Component", ylab="Cumulative Proportion of Variance Explained",
     ylim=c(0,1), type="b")
```



There is an elbow after the first component To be safer, we use two components in the following task

d) *Compare the three models in a)-c) by calculating the test MSE using the test data that was held out in using the validation set approach. Report the test MSEs. Which model performs best?*

```
#Using best lambda
lassoPred=predict(lasso, s=lassoCV$lambda.min, newx=x[test,])
lassoMSE=mean((lassoPred-y[test])^2)

#Using best lambda
ridgePred=predict(ridge, s=ridgeCV$lambda.min, newx=x[test,])
ridgeMSE=mean((ridgePred-y[test])^2)

#Using 2 PCAs
PCRpred=predict(PCRfit, x[test,], ncomp=2)
PCRMSE = mean((PCRpred -y[test])^2)

MSEtable = cbind(round(lassoMSE, 4),
                 round(ridgeMSE, 4),
                 round(PCRMSE, 4))
colnames(MSEtable) = c("Lasso", "Ridge", "PCR")
rownames(MSEtable) = c("MSE")
print(MSEtable)
```

```
##      Lasso  Ridge    PCR
## MSE 0.0291 0.0293 0.0494
```

Lasso performs best, with ridge a close second. PCR is performing poorly because we use only two components. Choosing the number of PCR components based on a Cross-Validation would improve prediction performance.