HW4

William Florez

12 de junio de 2017

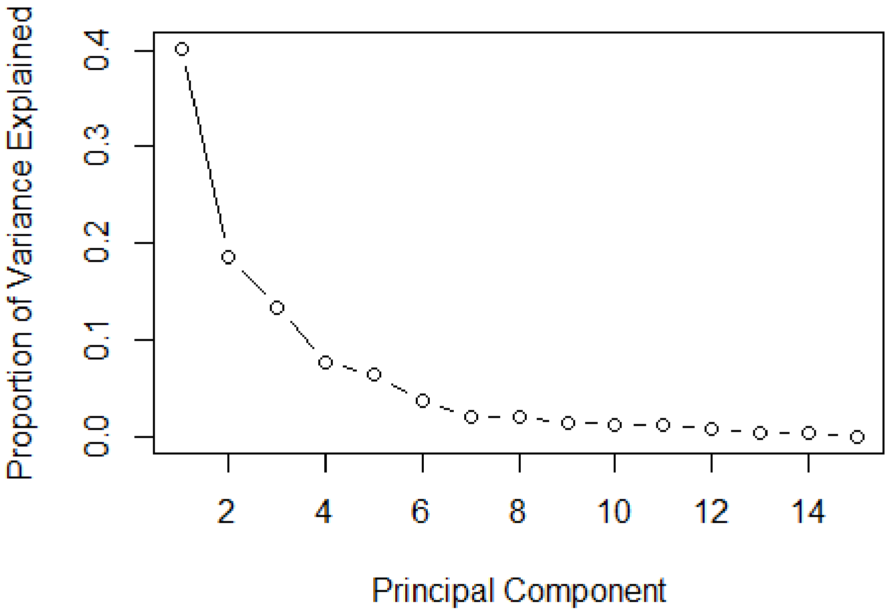
HW4

Question 1

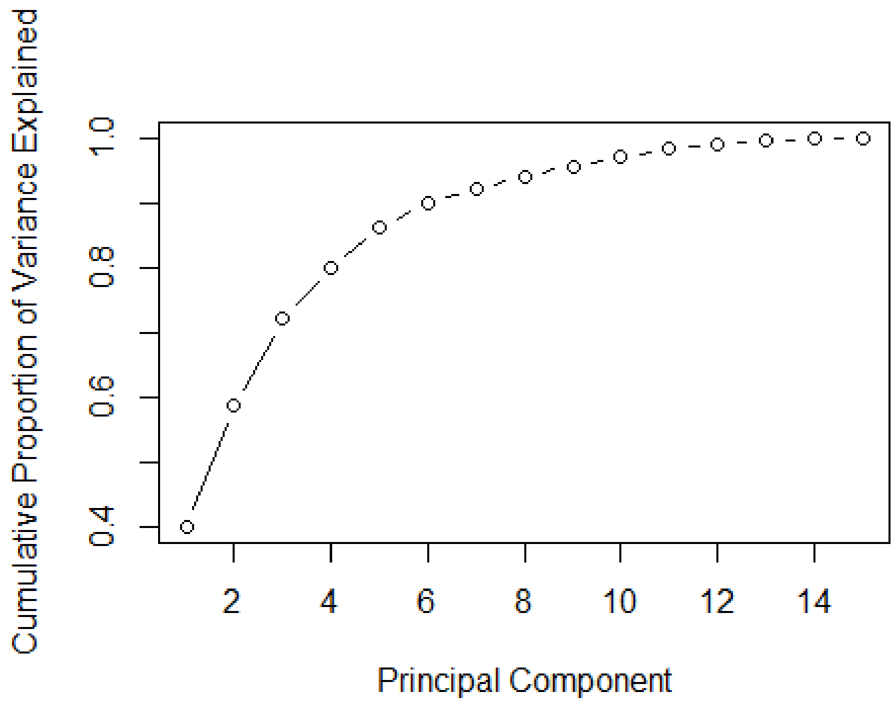
```
library(ggplot2)
library(MASS)
library(caTools)
library(factoextra) # chart fviz_pca_var
#library(leaps) # best combinations of predition
#library(relaimpo) #relative importance of PCAs

setwd("C:/Users/ce02144/Documents/HW4")

#Reading data
uscrime <- read.table("uscrime.txt", header = TRUE)
str(uscrime)

## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553
0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081
0.1 ...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...

#PCA analysis
uscrime.pca = prcomp(uscrime[,-16], center = TRUE, scale. = TRUE)
summary(uscrime.pca)

## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
```

```
## Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377
## Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688
## Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.56729 0.55444 0.48493 0.44708 0.41915 0.35804
## Proportion of Variance 0.02145 0.02049 0.01568 0.01333 0.01171 0.00855
## Cumulative Proportion  0.92142 0.94191 0.95759 0.97091 0.98263 0.99117
##                           PC13   PC14    PC15
## Standard deviation     0.26333 0.2418 0.06793
## Proportion of Variance 0.00462 0.0039 0.00031
## Cumulative Proportion  0.99579 0.9997 1.00000

str(uscrime.pca)

## List of 5
##  $ sdev    : num [1:15] 2.453 1.674 1.416 1.078 0.979 ...
##  $ rotation: num [1:15, 1:15] -0.304 -0.331 0.34 0.309 0.311 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:15] "M" "So" "Ed" "Po1" ...
##   .. ..$ : chr [1:15] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:15] 13.86 0.34 10.56 8.5 8.02 ...
##   ..- attr(*, "names")= chr [1:15] "M" "So" "Ed" "Po1" ...
##  $ scale   : Named num [1:15] 1.257 0.479 1.119 2.972 2.796 ...
##   ..- attr(*, "names")= chr [1:15] "M" "So" "Ed" "Po1" ...
##  $ x       : num [1:47, 1:15] -4.2 1.17 -4.17 3.83 1.84 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:15] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"

#compute standard deviation of each principal component
std_dev <- uscrime.pca$sdev
#compute variance
pca_var <- std_dev^2
#check variance of first 4 components
pca_var[1:4]

## [1] 6.018953 2.801847 2.004944 1.162208

#proportion of variance explained by first 4 components
prop_varex <- pca_var/sum(pca_var)
prop_varex[1:4]

## [1] 0.40126351 0.18678980 0.13366296 0.07748052

#total proportion of variance explained by first 4 components
sum(prop_varex[1:4])

## [1] 0.7991968

#scree plot
plot(prop_varex, xlab = "Principal Component",
```

```r
                 ylab = "Proportion of Variance Explained",
                 type = "b")
```




```r
#cumulative scree plot the fist 4 PCA explain near to 80% of variance.
There a space that could be improve for better fit
plot(cumsum(prop_varex), xlab = "Principal Component",
              ylab = "Cumulative Proportion of Variance Explained", type
= "b")
```




```
#transform test into PCA
#select the first 4 components
crime.scale = scale( uscrime$Crime)
factor.predictors = uscrime.pca$x[, 1:4]
#I want to check what happend with all PCA
factor.predictors.total = uscrime.pca$x[, 1:15]

#running lm with PCA
model.pca = lm(formula = crime.scale ~ factor.predictors)
model.pca.total = lm(formula = crime.scale ~ factor.predictors.total)
#running lm hw3
model.hw3 = lm(Crime ~ ., data = uscrime)

#Checkin how well are the first 4 PCA vs HW3
summary(model.pca)

##
## Call:
## lm(formula = crime.scale ~ factor.predictors)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44212 -0.54532 -0.07519  0.51002  2.09521
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -7.699e-17  1.269e-01   0.000  1.00000
```

```
## factor.predictorsPC1  1.686e-01  5.228e-02   3.225  0.00244 **
## factor.predictorsPC2 -1.812e-01  7.662e-02  -2.365  0.02273 *
## factor.predictorsPC3  6.514e-02  9.058e-02   0.719  0.47602
## factor.predictorsPC4  1.796e-01  1.190e-01   1.509  0.13872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8699 on 42 degrees of freedom
## Multiple R-squared:  0.3091, Adjusted R-squared:  0.2433
## F-statistic: 4.698 on 4 and 42 DF,  p-value: 0.003178
```

```r
summary(model.hw3)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = uscrime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

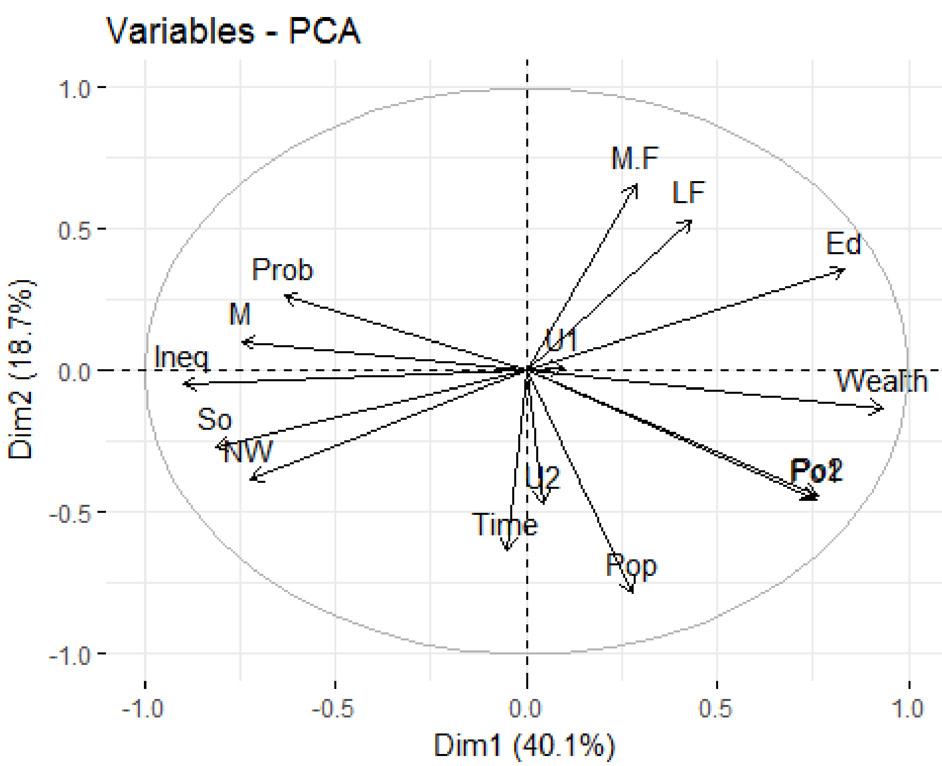
```r
#R2 is pretty small, so maybe first 4-PCA are not sufficient for
reduction

#Just checking what happend if I use all PCA, and the results is the same
R2 of HW3. It looks to have the same R2 that HW3
```

```
summary(model.pca.total)

##
## Call:
## lm(formula = crime.scale ~ factor.predictors.total)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02321 -0.25361 -0.01731  0.29214  1.32554
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -1.365e-16  7.885e-02   0.000 1.000000
## factor.predictors.totalPC1   1.686e-01  3.249e-02   5.191 1.24e-05 ***
## factor.predictors.totalPC2  -1.812e-01  4.761e-02  -3.806 0.000625 ***
## factor.predictors.totalPC3   6.514e-02  5.629e-02   1.157 0.255987
## factor.predictors.totalPC4   1.796e-01  7.393e-02   2.429 0.021143 *
## factor.predictors.totalPC5  -5.922e-01  8.142e-02  -7.274 3.49e-08 ***
## factor.predictors.totalPC6  -1.557e-01  1.072e-01  -1.453 0.156305
## factor.predictors.totalPC7   3.032e-01  1.405e-01   2.158 0.038794 *
## factor.predictors.totalPC8   7.425e-02  1.437e-01   0.517 0.609159
## factor.predictors.totalPC9  -9.612e-02  1.644e-01  -0.585 0.562890
## factor.predictors.totalPC10  1.456e-01  1.783e-01   0.817 0.420261
## factor.predictors.totalPC11  7.910e-02  1.901e-01   0.416 0.680272
## factor.predictors.totalPC12  7.488e-01  2.226e-01   3.364 0.002059 **
## factor.predictors.totalPC13  2.115e-01  3.027e-01   0.699 0.489962
## factor.predictors.totalPC14  5.667e-01  3.296e-01   1.719 0.095517 .
## factor.predictors.totalPC15 -1.609e+00  1.173e+00  -1.371 0.180174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5405 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07

#I continue with first 4 PCA

#converting PCA with rotation to inicial data
model.pca.coef <- as.vector(coef(model.pca)[-1])
cf = uscrime.pca$rotation[,1:4] %*% t(t(model.pca$coefficients[2:5])) /
uscrime.pca$scale
a = data.frame(cf =-setNames(as.vector(cf), rownames(cf)))

#As it is very complicated to interpreted PCs I look deeper to find some
insights

# Correlation between variables and principal components
var_cor_func = function(var.loadings, comp.sdev){
var.loadings * comp.sdev
```

```
}
loadings = uscrime.pca$rotation
sdev = uscrime.pca$sdev
var.coord = var.cor = t(apply(loadings, 1, var_cor_func, sdev))
var.coord[, 1:4]
```

```
##                PC1         PC2           PC3         PC4
## M      -0.74511332  0.10512512  0.2441399137 -0.02194425
## So     -0.81176941 -0.26509475  0.0220087146  0.31530140
## Ed      0.83321219  0.35923219  0.0959166378  0.08596834
## Po1     0.75718917 -0.45164011  0.0717124787  0.35926327
## Po2     0.76297599 -0.44184024  0.0751381137  0.37939869
## LF      0.43222620  0.53468560  0.3844760237 -0.15444821
## M.F     0.28552693  0.66008180 -0.2876696074  0.01129835
## Pop     0.27742141 -0.78209078  0.1090588365 -0.03461118
## NW     -0.72027195 -0.38166152  0.1115998697  0.25793570
## U1      0.09936425  0.01351549 -0.9331592167 -0.19705915
## U2      0.04446040 -0.46820433 -0.8191340804 -0.07427074
## Wealth  0.93154715 -0.12920386  0.0142513123  0.12701406
## Ineq   -0.89743196 -0.04606896 -0.0004169385 -0.08696271
## Prob   -0.63514085  0.26500250 -0.1666198235  0.53151884
## Time   -0.05060947 -0.63631965  0.3165612965 -0.58278708
```

```
fviz_pca_var(uscrime.pca)
```

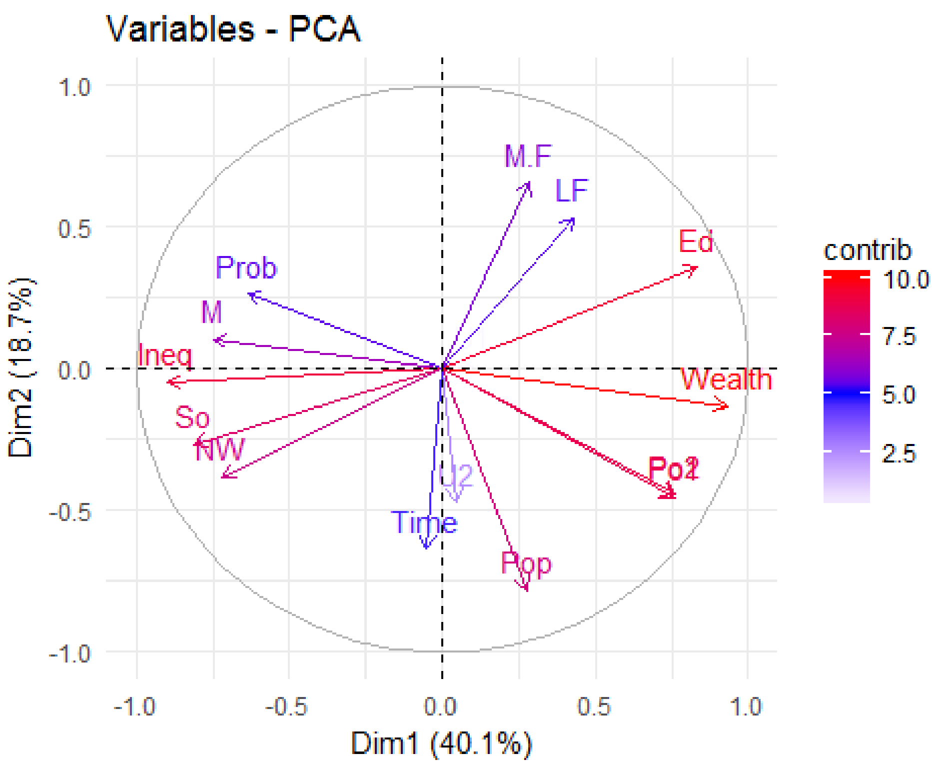



```
#Dim1 explain 40% and Dim2 18.7%. The most positive correlated variable
in Dim1 is Wealth and in Dim2 is M.F. The most negative correlated
```

```
variables in Dim1 is Ineq and Dim2 is Pop

#Contributions of the variables to the PC
var.cos2 = var.coord^2

comp.cos2 = apply(var.cos2, 2, sum)
contrib = function(var.cos2, comp.cos2)
{var.cos2 * 100 / comp.cos2}

var.contrib = t(apply(var.cos2, 1, contrib, comp.cos2))
var.contrib[, 1:4]
```

```
##                PC1          PC2          PC3         PC4
## M       9.22409415  0.394428809 2.972865e+00  0.04143410
## So     10.94824313  2.508175088 2.415945e-02  8.55397587
## Ed     11.53427509  4.605810608 4.588657e-01  0.63590652
## Po1     9.52550180  7.280154447 2.564999e-01 11.10559526
## Po2     9.67165553  6.967646540 2.815907e-01 12.38533809
## LF      3.10385379 10.203579444 7.372864e+00  2.05249438
## M.F     1.35448195 15.550741441 4.127486e+00  0.01098365
## Pop     1.27867164 21.830813305 5.932249e-01  0.10307395
## NW      8.61930160  5.198910276 6.211909e-01  5.72452070
## U1      0.16403609  0.006519577 4.343194e+01  3.34125357
## U2      0.03284171  7.823956481 3.346630e+01  0.47462623
## Wealth 14.41746002  0.595808306 1.012995e-02  1.38809688
## Ineq   13.38080178  0.075748228 8.670452e-06  0.65070236
## Prob    6.70222751  2.506429665 1.384685e+00 24.30824140
## Time    0.04255422 14.451277784 4.998196e+00 29.22375704
```

```
fviz_pca_var(uscrime.pca, col.var = "contrib") +
scale_color_gradient2(low = "white", mid = "blue",
high = "red", midpoint = 5) + theme_minimal()
```

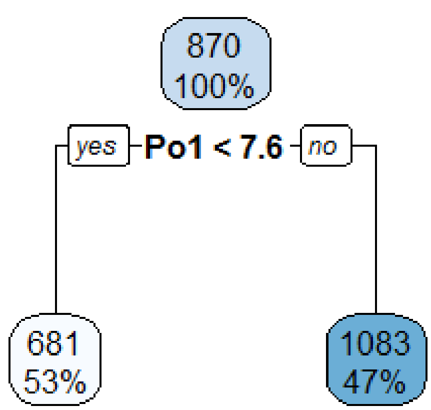



```
#The variables that has most contribution are in Dim1 Wealth and ineq.
And in Dim2 M.F and Pop
```

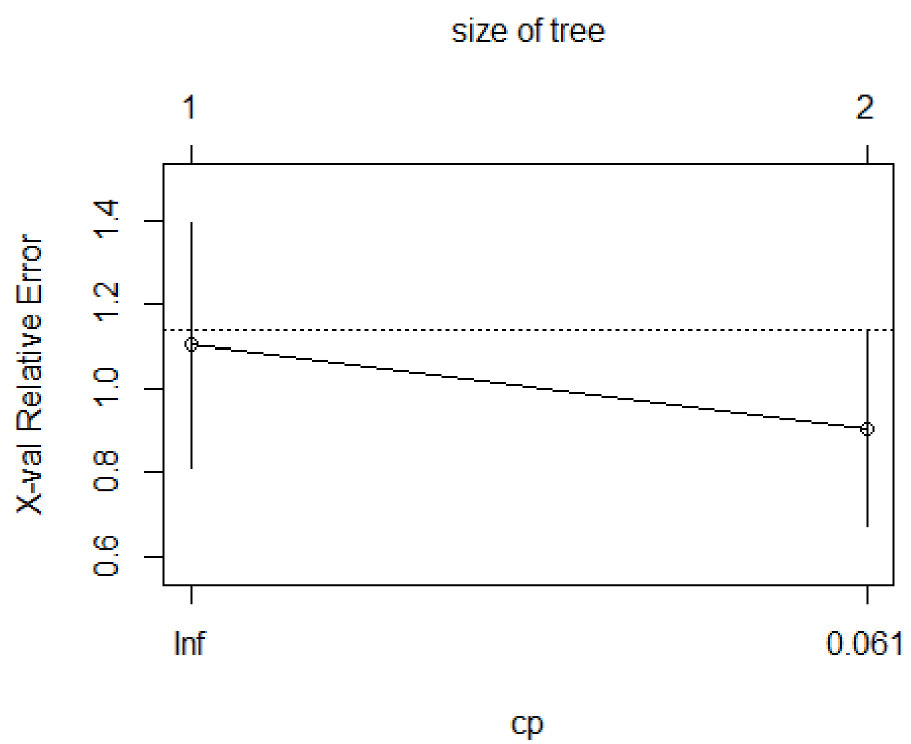
Question 2

```
library(ggplot2)
library(MASS)
library(caTools)
library(factoextra) # chart fviz_pca_var
library(rpart)
library(rpart.plot)
library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

library(caret)

## Loading required package: lattice

setwd("C:/Users/ce02144/Documents/HW4")

#Reading data
uscrime <- read.table("uscrime.txt", header = TRUE)

#splitting data
spl = sample.split(uscrime, 0.7)
train = subset(uscrime, spl == T)
test = subset(uscrime, spl == F)


#Runnung tree and random forest. Anova for tree regression
cart.model = rpart(Crime ~ . , data = train, method = "anova")

#Plotting tree
rpart.plot(cart.model)
```

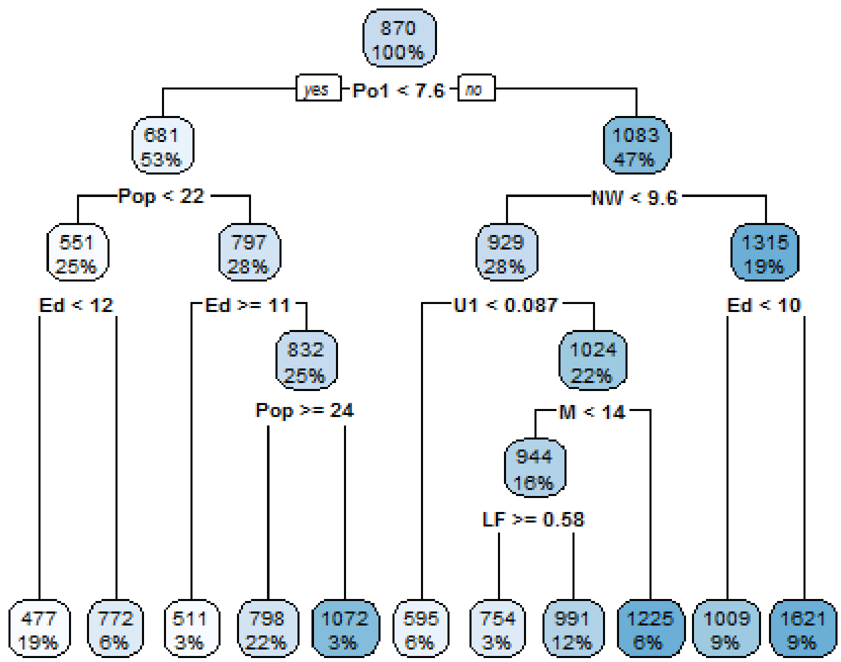


```r
#Pretty small tree

# display the results
printcp(cart.model)

##
## Regression tree:
## rpart(formula = Crime ~ ., data = train, method = "anova")
##
## Variables actually used in tree construction:
## [1] Po1
##
## Root node error: 3517722/32 = 109929
##
## n= 32
##
##        CP nsplit rel error  xerror    xstd
## 1 0.36623      0   1.00000 1.10394 0.29122
## 2 0.01000      1   0.63377 0.90324 0.23322

# visualize cross-validation results
plotcp(cart.model)
```



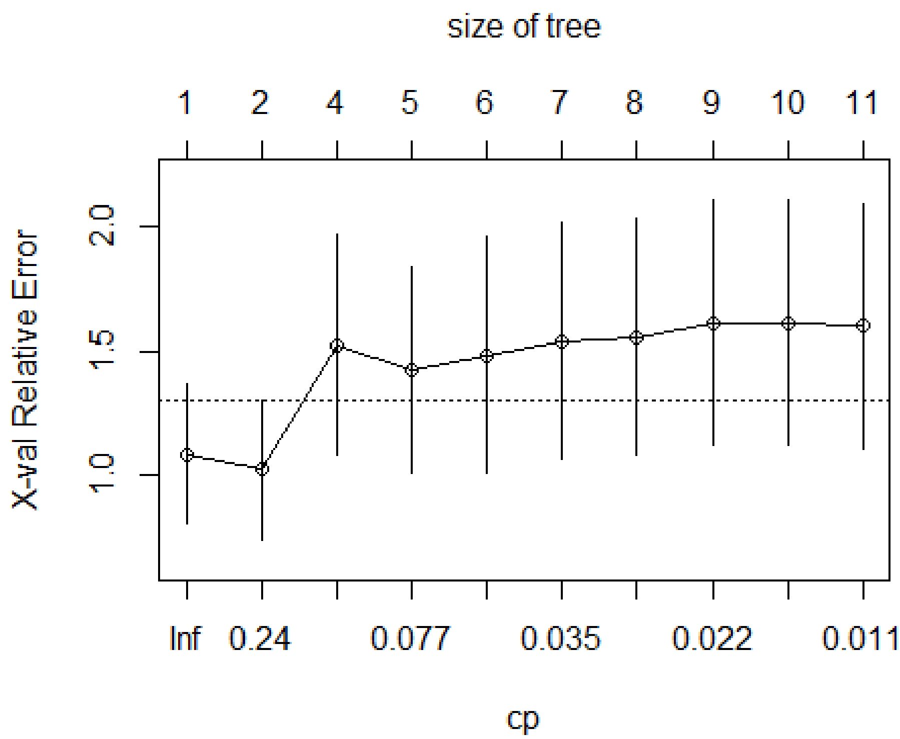

```r
#I want to know what happend if I force the model for a broader tree

cart.model.broad = rpart(Crime ~ . , data = train, method = "anova",
minsplit = 2)
```

```r
# create additional plots
rpart.plot(cart.model.broad)
```

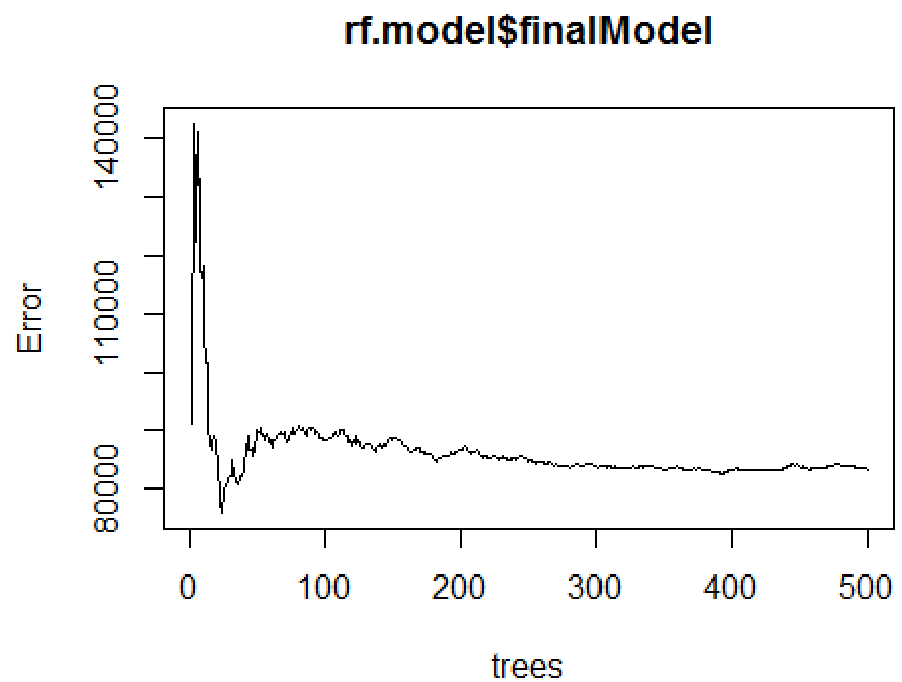



```r
# display the results
printcp(cart.model.broad)

##
## Regression tree:
## rpart(formula = Crime ~ ., data = train, method = "anova", minsplit =
2)
##
## Variables actually used in tree construction:
## [1] Ed  LF  M   NW  Po1 Pop U1
##
## Root node error: 3517722/32 = 109929
##
## n= 32
##
##          CP nsplit rel error xerror    xstd
## 1  0.366234      0  1.000000 1.0824 0.28412
## 2  0.156401      1  0.633766 1.0187 0.28009
## 3  0.081384      3  0.320963 1.5214 0.44484
## 4  0.072729      4  0.239579 1.4235 0.41882
## 5  0.037025      5  0.166850 1.4810 0.47747
## 6  0.032158      6  0.129826 1.5381 0.47870
## 7  0.026118      7  0.097668 1.5556 0.47721
## 8  0.018636      8  0.071549 1.6123 0.49564
```

```
## 9  0.012774      9  0.052914 1.6109 0.49427
## 10 0.010000     10  0.040140 1.5995 0.49466

# visualize cross-validation results
plotcp(cart.model.broad)
```




```
#the last tree could show more information but the cost if a higher
noise. So depend of the analysis of cost-benefic it could be an approah
for decision

#running Random Forest
rf.model = train(Crime ~ . , data = train, method = "rf", metric =
"RMSE", preProc = c("center", "scale"))

#plotting #tree vs RMSE
plot(rf.model$finalModel)
```




```
#making prediction
pred.cart = predict(cart.model, newdata = test)
pred.rf = predict(rf.model, newdata = test)

#Measuring quality of model
ModelMetrics::mse(test$Crime, pred.cart)

## [1] 147420.3

ModelMetrics::mse(test$Crime, pred.rf)

## [1] 130044.9

#As rf could use different trees has a better fit
```

Question 3

A logistic regression for could be *buy, sell or maintain a equity ETF*. And ETF is an exchange-traded fund.

The possible outcome and threshold could be:

1. Sell. Threshold < 40%
2. Maintain. Threshold >= 40 & <= 60%
3. Buy. Threshold > 60%

As a predictors I could use:
CPI actual - CPI Forecast

Central Bank Rate actual - Central Bank Forecast
GDP actual - GDP Forecast
Consumer Sentiment
Private Payrolls
Dividend Yield - Corporate Bond Yield
US 2year - US 10year yield spread
Price to Earning ratio
Price to Book Ratio
Leading indicator

Question 4

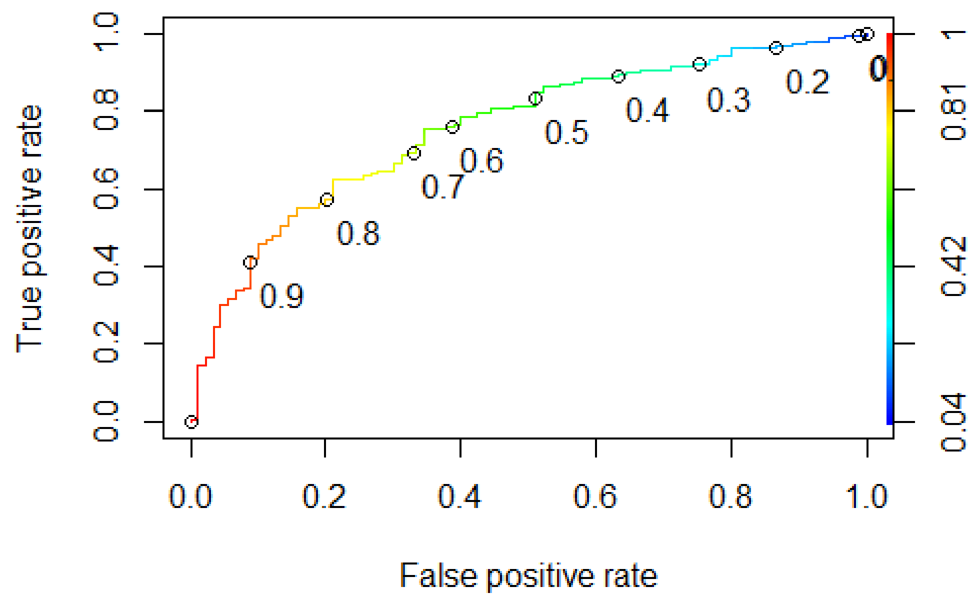
```
library(randomForest)
library(caTools)
library(rpart)
library(rpart.plot)
library(randomForest)
library(SnowballC)
library(ROCR)

setwd("C:/Users/ce02144/Documents/HW4")

#Reading data
germancredit <- read.table("germancredit.txt", header = TRUE)
str(germancredit)

## 'data.frame':    999 obs. of  21 variables:
##  $ A11  : Factor w/ 4 levels "A11","A12","A13",..: 2 4 1 1 4 4 2 4 2 2
...
##  $ X6   : int  48 12 42 24 36 24 36 12 30 12 ...
##  $ A34  : Factor w/ 5 levels "A30","A31","A32",..: 3 5 3 4 3 3 3 3 5 3
...
##  $ A43  : Factor w/ 10 levels "A40","A41","A410",..: 5 8 4 1 8 4 2 5 1
1 ...
##  $ X1169: int  5951 2096 7882 4870 9055 2835 6948 3059 5234 1295 ...
##  $ A65  : Factor w/ 5 levels "A61","A62","A63",..: 1 1 1 1 5 3 1 4 1 1
...
##  $ A75  : Factor w/ 5 levels "A71","A72","A73",..: 3 4 4 3 3 5 3 4 1 2
...
##  $ X4   : int  2 2 2 3 2 3 2 2 4 3 ...
##  $ A93  : Factor w/ 4 levels "A91","A92","A93",..: 2 3 3 3 3 3 3 1 4 2
...
##  $ A101 : Factor w/ 3 levels "A101","A102",..: 1 1 3 1 1 1 1 1 1 1 ...
##  $ X4.1 : int  2 3 4 4 4 4 2 4 2 1 ...
##  $ A121 : Factor w/ 4 levels "A121","A122",..: 1 1 2 4 4 2 3 1 3 3 ...
##  $ X67  : int  22 49 45 53 35 53 35 61 28 25 ...
##  $ A143 : Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ A152 : Factor w/ 3 levels "A151","A152",..: 2 2 3 3 3 2 1 2 2 1 ...
##  $ X2   : int  1 1 1 2 1 1 1 1 2 1 ...
##  $ A173 : Factor w/ 4 levels "A171","A172",..: 3 2 3 3 2 3 4 2 4 3 ...
```

```
##  $ X1   : int  1 2 2 2 2 1 1 1 1 1 ...
##  $ A192 : Factor w/ 2 levels "A191","A192": 1 1 1 1 2 1 2 1 1 1 ...
##  $ A201 : Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
##  $ X1.1 : int  2 1 1 2 1 1 1 1 2 2 ...

germancredit$X1.1[germancredit$X1.1 == 2] = 0
germancredit$X1.1 <- factor(germancredit$X1.1, labels = c('Bad', 'Good'))

table(germancredit$X1.1)

##
##  Bad Good
##  300  699

good = 699/(300+699)
bad = 1- good
good

## [1] 0.6996997

bad

## [1] 0.3003003

spl = sample.split(germancredit$X1.1, 0.7)
train = subset(germancredit, spl == T)
test = subset(germancredit, spl == F)

# Part 1

#Logit regression
logisticModel = glm(X1.1 ~  .,
                    family = binomial(link='logit'),
                    data = train)

summary(logisticModel)

##
## Call:
## glm(formula = X1.1 ~ ., family = binomial(link = "logit"), data =
train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5748  -0.6095   0.3258   0.6902   2.3121
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.524e+00  1.383e+00  -1.825 0.067986 .
## A11A12       4.811e-01  2.759e-01   1.744 0.081166 .
## A11A13       1.035e+00  4.476e-01   2.313 0.020723 *
```

```
## A11A14       1.872e+00  2.872e-01   6.518 7.15e-11 ***
## X6          -3.863e-02  1.161e-02  -3.328 0.000875 ***
## A34A31       7.725e-01  7.113e-01   1.086 0.277466
## A34A32       1.565e+00  5.897e-01   2.653 0.007971 **
## A34A33       1.454e+00  6.326e-01   2.299 0.021483 *
## A34A34       2.258e+00  5.979e-01   3.776 0.000160 ***
## A43A41       2.057e+00  4.773e-01   4.310 1.63e-05 ***
## A43A410      1.733e+00  9.475e-01   1.829 0.067440 .
## A43A42       7.891e-01  3.201e-01   2.466 0.013682 *
## A43A43       9.405e-01  3.037e-01   3.097 0.001953 **
## A43A44       5.091e-01  8.354e-01   0.609 0.542266
## A43A45       2.148e-01  6.640e-01   0.324 0.746265
## A43A46      -1.821e-01  4.932e-01  -0.369 0.711990
## A43A48       2.175e+00  1.282e+00   1.696 0.089930 .
## A43A49       1.904e+00  4.637e-01   4.107 4.00e-05 ***
## X1169       -8.461e-05  5.677e-05  -1.490 0.136108
## A65A62       4.562e-01  3.725e-01   1.225 0.220732
## A65A63       2.700e-01  4.734e-01   0.570 0.568466
## A65A64       1.572e+00  6.677e-01   2.355 0.018532 *
## A65A65       7.193e-01  3.214e-01   2.238 0.025232 *
## A75A72       4.141e-01  5.530e-01   0.749 0.454039
## A75A73       2.404e-01  5.244e-01   0.458 0.646643
## A75A74       8.422e-01  5.665e-01   1.487 0.137121
## A75A75       2.176e-01  5.334e-01   0.408 0.683287
## X4          -3.250e-01  1.101e-01  -2.952 0.003160 **
## A93A92       3.541e-01  4.896e-01   0.723 0.469536
## A93A93       1.285e+00  4.840e-01   2.654 0.007947 **
## A93A94       6.669e-01  5.693e-01   1.171 0.241414
## A101A102    -3.033e-01  5.113e-01  -0.593 0.553024
## A101A103     9.307e-01  4.972e-01   1.872 0.061244 .
## X4.1         1.478e-01  1.058e-01   1.397 0.162509
## A121A122    -2.754e-01  3.150e-01  -0.874 0.381968
## A121A123     7.373e-02  2.943e-01   0.251 0.802173
## A121A124    -3.062e-01  5.594e-01  -0.547 0.584145
## X67          3.019e-02  1.172e-02   2.576 0.010002 *
## A143A142     2.036e-01  4.993e-01   0.408 0.683419
## A143A143     8.137e-01  2.993e-01   2.719 0.006557 **
## A152A152     6.010e-01  2.854e-01   2.106 0.035232 *
## A152A153     3.297e-01  6.108e-01   0.540 0.589368
## X2          -3.282e-01  2.327e-01  -1.411 0.158386
## A173A172    -8.944e-01  8.785e-01  -1.018 0.308603
## A173A173    -9.339e-01  8.406e-01  -1.111 0.266583
## A173A174    -8.586e-01  8.516e-01  -1.008 0.313329
## X1          -3.359e-01  3.194e-01  -1.052 0.292945
## A192A192     1.590e-01  2.492e-01   0.638 0.523416
## A201A202     1.371e+00  8.232e-01   1.665 0.095936 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 854.5  on 698  degrees of freedom
## Residual deviance: 597.1  on 650  degrees of freedom
## AIC: 695.1
##
## Number of Fisher Scoring iterations: 5

#showing coefficients of the model
coefficients(logisticModel)

##   (Intercept)        A11A12        A11A13        A11A14            X6
## -2.523635e+00  4.811496e-01  1.035369e+00  1.871541e+00 -3.863230e-02
##        A34A31        A34A32        A34A33        A34A34        A43A41
##  7.725032e-01  1.564527e+00  1.454474e+00  2.257535e+00  2.056978e+00
##       A43A410        A43A42        A43A43        A43A44        A43A45
##  1.732672e+00  7.891266e-01  9.405067e-01  5.090826e-01  2.148367e-01
##        A43A46        A43A48        A43A49         X1169        A65A62
## -1.820805e-01  2.174796e+00  1.904341e+00 -8.460603e-05  4.561813e-01
##        A65A63        A65A64        A65A65        A75A72        A75A73
##  2.699583e-01  1.572390e+00  7.193080e-01  4.140542e-01  2.404081e-01
##        A75A74        A75A75            X4        A93A92        A93A93
##  8.421507e-01  2.176286e-01 -3.249592e-01  3.540917e-01  1.284757e+00
##        A93A94      A101A102      A101A103          X4.1      A121A122
##  6.669328e-01 -3.033430e-01  9.306850e-01  1.477989e-01 -2.753919e-01
##      A121A123      A121A124           X67      A143A142      A143A143
##  7.372638e-02 -3.061800e-01  3.019213e-02  2.036272e-01  8.136998e-01
##      A152A152      A152A153            X2      A173A172      A173A173
##  6.009510e-01  3.296825e-01 -3.282497e-01 -8.944320e-01 -9.338649e-01
##      A173A174            X1      A192A192      A201A202
## -8.586497e-01 -3.359073e-01  1.590420e-01  1.370554e+00

#Part 2

predLog = predict(logisticModel, newdata = test,type = "response" )
#  Taking into account that incorrectly identifying a bad customer as
good, is 5 times worse than incorrectly classifying a good customer as
bad, the threshold is 1 to 5 or 80%

table(test$X1.1, predLog >= 0.8)

##
##         FALSE TRUE
##   Bad      72   18
##   Good     90  120

acc = table(test$X1.1, predLog >= 0.8)
# acurracy (True Positve + True Negative) / Total
(acc[1,1]+ acc[2,2])/(nrow(test))

## [1] 0.64
```

```r
#Checking for AUC and ROC
predRocLog = prediction(predLog, test$X1.1)
as.numeric(performance(predRocLog, "auc")@y.values)

## [1] 0.7559259

# Building ROC Prediction function
ROCRpred = prediction(predLog , test$X1.1)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
plot(ROCRperf, colorize = TRUE, print.cutoffs.at = seq(0,1,by = 0.1),
text.adj = c(-0.2,1.7))
```




```r
#The 80% threshold give us a rate near to 20% of false posive
#And near to 60% of True posive.
```