# azizhazeinita

```
library(ggplot2)
library(MASS)
library(poLCA)
```

```
## Loading required package: scatterplot3d
```

```
BostonData <- Boston

### --- Step 1 --- ###
# 1.1. Split samples into two random samples of sizes 70% and 30%.
crim=BostonData[,1]
rm=BostonData[,6]
age=BostonData[,7]
dis=BostonData[,8]
lstat=BostonData[,13]
medv=BostonData[,14]
df <- data.frame(crim,rm,age,dis,lstat,medv)

X<- sample(c(rep(0, 0.7 * nrow(df)), rep(1, 0.3 * nrow(df))))
table(X)
```

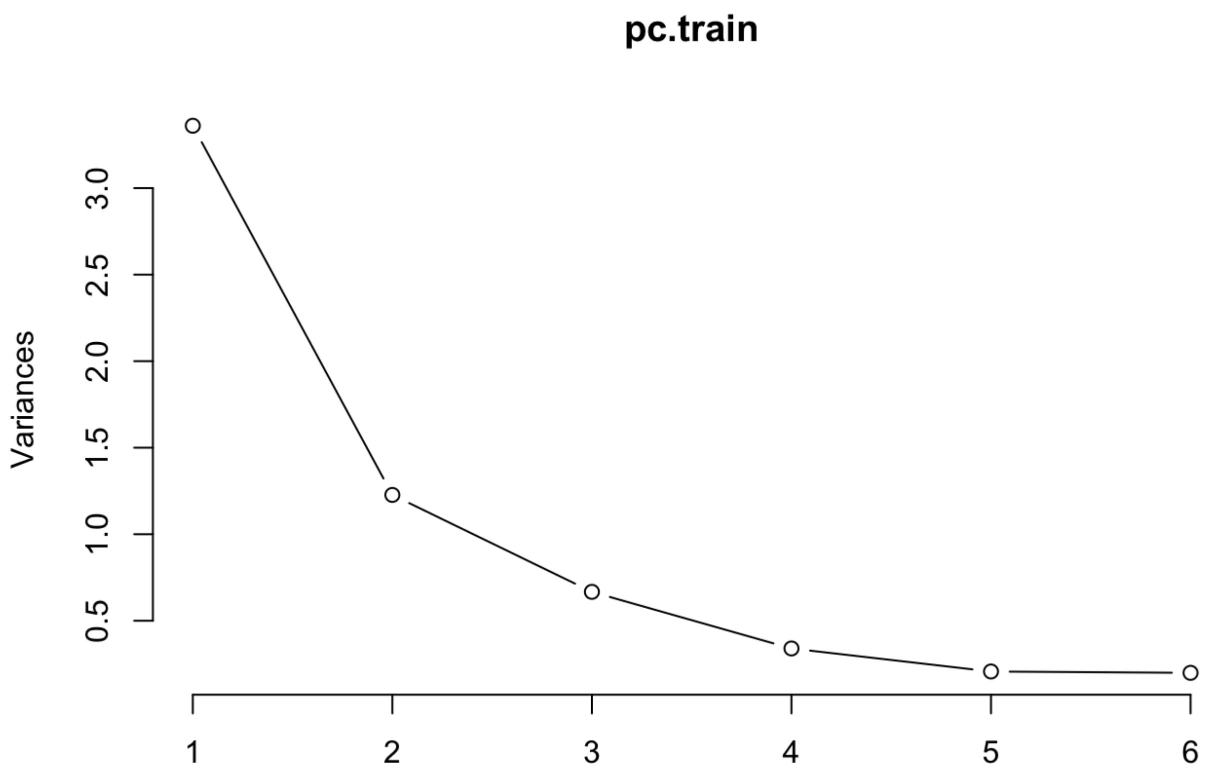
```
## X
##   0   1
## 354 151
```

```
train <- df[X == 0, ]
test <- df[X== 1, ]

### --- Step 2 --- ###
# 2.1. Standardize your data so that each variable has mean = 0 and variance = 1
# 2.2. Scale the test set using the mean and standard deviation of the training set.
X.train.mean = colMeans(train)
X.train.sd   = sapply(train, sd)
X.train.scale  = scale(train, center=X.train.mean, scale=X.train.sd)
X.test.scale   = scale(test, center=X.train.mean, scale=X.train.sd) #scaling test by tra
in parameters

# 3. Perform PCA on the train data. Use princomp (R) and PCA (Python) function.
pc.train <- prcomp(X.train.scale)

### --- Step 3 --- ###
# 3.1. Display cumulative sum of variance accounted for (cumulative proportion) by each
 additional PCA factor.
plot (pc.train, type='l')
```



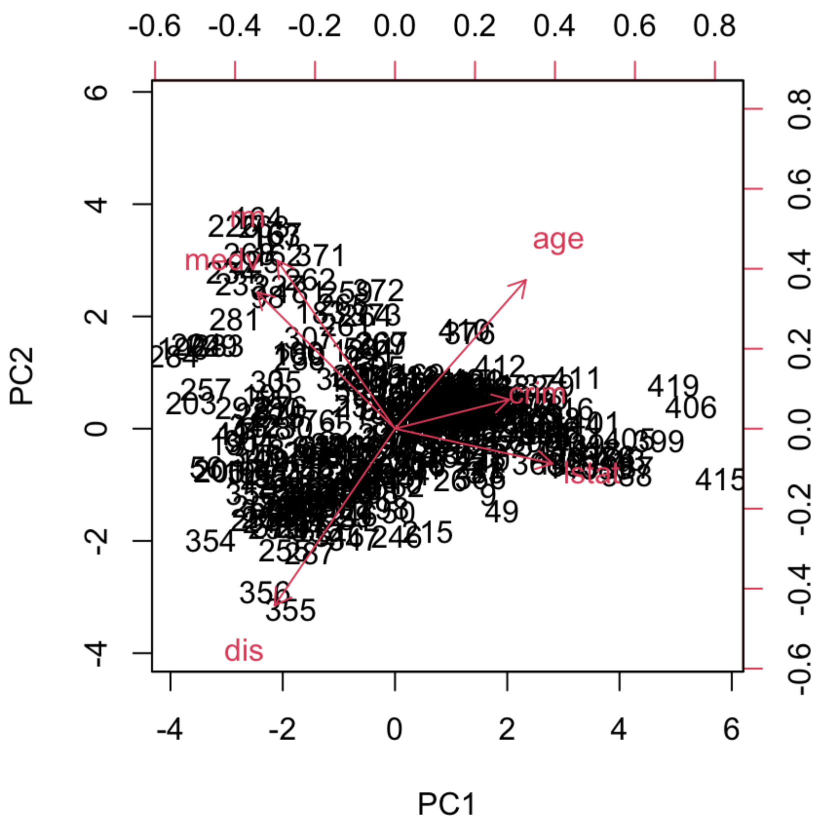

```
summary(pc.train)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.8331 1.1077 0.8167 0.58290 0.45475 0.44609
## Proportion of Variance 0.5601 0.2045 0.1112 0.05663 0.03447 0.03317
## Cumulative Proportion  0.5601 0.7646 0.8757 0.93237 0.96683 1.00000
```

```
# 3.2. Apply the "elbow rule" to decide how many components you'd like to include.
'Selected number of component: 2'
```

```
## [1] "Selected number of component: 2"
```

```
### --- Step 4 --- ###
# 4.1. Plot loading 1 against all of the other loadings (6 pairwise comparisons).
#biplot(pc.train[,c(1,2)])
biplot(pc.train$x, pc.train$rotation)
```




```
'Interpretation is documented in PDF document'
```

```
## [1] "Interpretation is documented in PDF document"
```

```
# 4.2.1. Show that Component loadings are orthogonal.
round(cor(t(pc.train$rotation) %*% pc.train$rotation))
```

```
##     PC1 PC2 PC3 PC4 PC5 PC6
## PC1   1   0   0   0   0   0
## PC2   0   1   0   0   0   0
## PC3   0   0   1   0   0   0
## PC4   0   0   0   1   0   0
## PC5   0   0   0   0   1   0
## PC6   0   0   0   0   0   1
```

```
# 4.2.2. Show that Component scores are orthogonal.
round(cor(t(pc.train$x) %*% pc.train$x))
```

```
##     PC1 PC2 PC3 PC4 PC5 PC6
## PC1   1   0   0   0   0   0
## PC2   0   1   0   0   0   0
## PC3   0   0   1   0   0   0
## PC4   0   0   0   1   0   0
## PC5   0   0   0   0   1   0
## PC6   0   0   0   0   0   1
```

```
# 4.2.3.1. predict the component scores in the Test using the predict() function in R an
d transform function in Python
pc.test.predict <- predict(pc.train, newdata = X.test.scale)

# 4.2.3.2. matrix multiply the predicted component scores from (1) above with transpose
 of component loadings you derived from training data set from Step 2 above.
matrix.multiply <- pc.test.predict[,1:2] %*% t(pc.train$rotation)[1:2,]

# 4.2.3.3. Compute the Variance Account For (R2) in the Test sample. That yields a measu
re of Test performance.

VAF_train <- round(cor(as.vector(X.train.scale), as.vector(pc.train$x[,1:2] %*% t(pc.tra
in$rotation)[1:2,])),2)^2
VAF_test <- round(cor(as.vector(X.test.scale), as.vector(matrix.multiply)),2)^2
VAF_train
```

```
## [1] 0.7569
```

```
VAF_test
```

```
## [1] 0.4761
```

```r
#Rotating Principal Components Solutions using varimax() in R
x=prcomp(df,scale=TRUE)
summary(x)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     1.8167 1.1044 0.8508 0.58416 0.45690 0.45419
## Proportion of Variance 0.5501 0.2033 0.1206 0.05687 0.03479 0.03438
## Cumulative Proportion  0.5501 0.7533 0.8740 0.93083 0.96562 1.00000
```

```r
cor(as.vector(scale(df)), as.vector(x$x[,1:2]  %*% t(x$rotation)[1:2,]))^2
```

```
## [1] 0.7533215
```

```r
y=varimax(x$rotation[,1:2])
y
```

```
## $loadings
## 
## Loadings:
##       PC1    PC2   
## crim   0.137  0.336
## rm    -0.636  0.119
## age           0.614
## dis    0.111 -0.655
## lstat  0.439  0.259
## medv  -0.609       
## 
##                  PC1   PC2
## SS loadings    1.000 1.000
## Proportion Var 0.167 0.167
## Cumulative Var 0.167 0.333
## 
## $rotmat
##            [,1]      [,2]
## [1,]  0.7160922 0.6980057
## [2,] -0.6980057 0.7160922
```

```r
cor(as.vector(scale(df)), as.vector(x$x[,1:2] %*% y$rotmat %*% t(y$rotmat) %*% t(x$rotat
ion)[1:2,]))^2
```

```
## [1] 0.7533215
```