STAT415homework7

1. Perform Principal Component Analysis on the predictors. Make a screeplot of the eigenvalues. How many eigevalues does one need to explain 90% of the variance in the data? Report loadings of the first two PCs.Interpret them if you can.

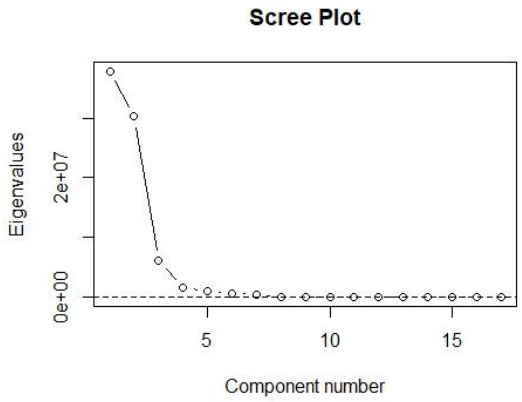
```
library(ISLR)

X = model.matrix(Apps~., data = College)[,-1]
XPCA = prcomp(x = X, center = T, scale = F)
summary(XPCA)

## Importance of components%s:
##                               PC1       PC2       PC3       PC4
PC5
## Standard deviation     6151.7492 5506.3030 2465.8686 1.259e+03 990.3
943
## Proportion of Variance    0.4861    0.3894    0.0781 2.035e-02   0.0
126
## Cumulative Proportion     0.4861    0.8755    0.9536 9.739e-01   0.9
865
##                               PC6       PC7       PC8       PC9     P
C10
## Standard deviation     783.31965 606.58942 200.45559 159.16467 21.27
256
## Proportion of Variance   0.00788   0.00473   0.00052   0.00033  0.00
001
## Cumulative Proportion    0.99442   0.99915   0.99966   0.99999  0.99
999
##                          PC11  PC12  PC13  PC14  PC15  PC16   PC17
## Standard deviation     14.84 12.56 9.004 6.037 5.326 2.912 0.2712
## Proportion of Variance  0.00  0.00 0.000 0.000 0.000 0.000 0.0000
## Cumulative Proportion   1.00  1.00 1.000 1.000 1.000 1.000 1.0000

# get the eigenvalues
R = cov(X)
e = eigen(R) #solving for the eigenvalues and eigenvectors from the cor
relation matrix
L = e$values #placing the eigenvalues in L
#This is the proportion of variance accounted for by each PC
Lupdate = L

plot(L,main="Scree Plot",ylab="Eigenvalues",xlab="Component number",typ
e='b')
abline(h=1, lty=2)
```




```
proportion = rep(0, length(Lupdate))
sum = 0
for (i in 1:length(Lupdate)){
  sum = sum + Lupdate[i]/sum(Lupdate)
  proportion[i] = sum
}
proportion
```

```
##  [1] 0.4860689 0.8754912 0.9535893 0.9739405 0.9865390 0.9944199 0.9
991459
##  [8] 0.9996620 0.9999873 0.9999932 0.9999960 0.9999980 0.9999991 0.9
999995
## [15] 0.9999999 1.0000000 1.0000000
```

```
min(which(proportion>0.9))
```

```
## [1] 3
```

```
# find loadings
XPCA$rotation[,1:2]
```

```
##                        PC1           PC2
## PrivateYes   3.626551e-05 -4.208858e-05
## Accept      -3.590779e-02  4.039561e-01
## Enroll      -2.965408e-02  1.603959e-01
## Top10perc    1.814640e-03  9.651757e-04
## Top25perc    1.625550e-03  1.169184e-03
## F.Undergrad -2.015731e-01  8.464071e-01
## P.Undergrad -6.636837e-02  1.501683e-01
## Outstate     5.717100e-01 -1.562345e-02
## Room.Board   1.085628e-01  1.645878e-02
## Books        1.863408e-03  4.261902e-03
## Personal    -2.680066e-02  3.391813e-02
## PhD          1.002484e-03  1.251639e-03
## Terminal     9.462382e-04  1.084315e-03
## S.F.Ratio   -4.213703e-04  8.271495e-05
## perc.alumni  1.087178e-03 -2.656222e-04
```

```
## Expend       7.832137e-01  2.654212e-01
## Grad.Rate    1.390595e-03  1.063867e-04
```

Comment: The screeplot has been shown above. According to the proportion result, 3 eigenvalues are needed to explain 90% of the variance in the data. The first two PCs' loadings have been shown above. The first component accounts for the most variance (48.6%). To show the percentage of variance accounted for by each variable– divide the eigenvalues by the number of variables since each scaled variable has a variance of 1. Also note, a property of the principal component scores is that they are not correlated with each other– they are completely orthogonal.

2. Fit a PCR model on the training set, with the number of principal components K chosen by cross-validation. Report the training and test error obtained, along with the value of K selected.

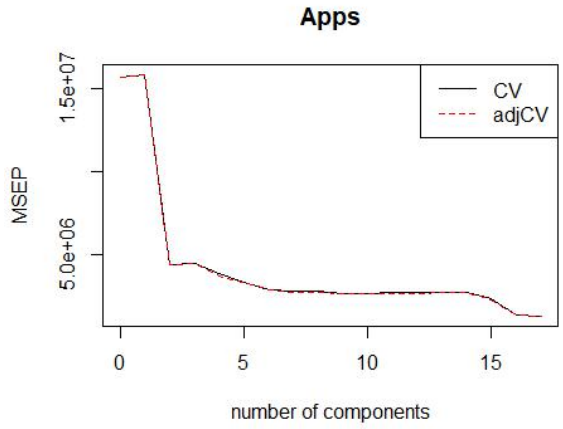
```
library(pls)

set.seed(23456)
test_id = sample(1:nrow(College), size = trunc(0.3 * nrow(College)))
origin = c(1:nrow(College))
train_id = origin[-c(test_id)]
test = College[test_id,]
train = College[-test_id,]
CollegePCR = pcr(Apps~., data = College, subset = train_id, scale = TRU
E, validation = "CV")
summary(CollegePCR)

## Data:    X dimension: 544 17
##  Y dimension: 544 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 c
omps
## CV            3954     3976     2108     2118     1963     1830
1713
## adjCV         3954     3977     2105     2117     1939     1837
1708
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 c
omps
## CV        1681     1678     1641      1645      1650      1649
1660
## adjCV     1665     1668     1636      1640      1645      1644
1655
##        14 comps  15 comps  16 comps  17 comps
## CV         1663      1533      1181      1160
## adjCV      1658      1507      1171      1151
##
## TRAINING: % variance explained
```

```
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X      32.163    57.52    64.51    70.14    75.56    80.67    84.09
## Apps    0.652    72.61    72.61    77.30    79.97    82.85    83.95
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 c
omps
## X       87.43    90.43     92.89     95.00     96.71     97.76     9
8.70
## Apps    84.01    84.34     84.49     84.51     84.52     84.52     8
4.53
##       15 comps  16 comps  17 comps
## X        99.36     99.86    100.00
## Apps     91.37     93.46     93.62
```

```
# choose principal components k
validationplot(CollegePCR, val.type = "MSEP", legendpos = "topright")
```




```
# Training error
CollegePCR.train = predict(CollegePCR, train[,names(train)!="Apps"], nc
omp=17)
PCRTrainMSE = mean((CollegePCR.train - College[-test_id,"Apps"])^2)
PCRTrainMSE
```

```
## [1] 993164.6
```

```
# test error
CollegePCR.test = predict(CollegePCR, test[,names(test)!="Apps"], ncomp
=17)
PCRTestMSE = mean((CollegePCR.test - College[test_id,"Apps"])^2)
PCRTestMSE
```

```
## [1] 1300431
```

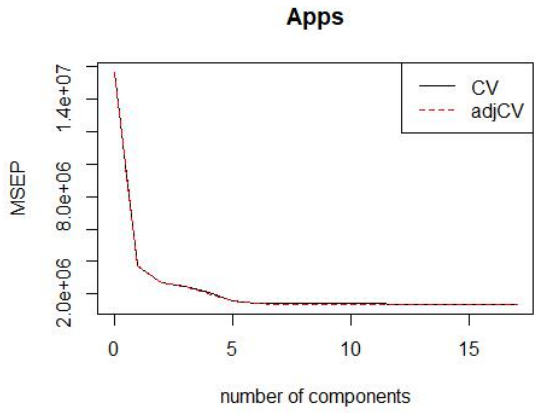
Comment: According to the plot, we can find that when k=17, the MSEP is the smallest. Thus we choose 17 principal components through CV. Then the corresponding training error is 993164.6 and test error is 1300431.

3. Fit a PLS model on the training set, with the number of principal components K chosen by cross-validation. Report the training and test error obtained, along with the value of K selected.

```
CollegePLS = plsr(Apps~., data = College, subset = train_id, scale = TR
UE, validation = "CV")
summary(CollegePLS)

## Data:    X dimension: 544 17
##  Y dimension: 544 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 c
omps
## CV            3954     1935     1647     1569     1444     1268
1199
## adjCV         3954     1930     1637     1565     1416     1244
1186
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 c
omps
## CV        1190     1186     1184      1183      1181      1179
1180
## adjCV     1178     1174     1173      1172      1170      1168
1169
##        14 comps  15 comps  16 comps  17 comps
## CV         1179      1179      1179      1179
## adjCV      1168      1168      1168      1168
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       25.31    35.82    62.56    64.78    67.63    72.00    74.77
## Apps    77.36    85.46    87.12    91.63    93.33    93.49    93.53
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 c
omps
## X       78.98    82.34     86.65     89.38     90.80     92.58     9
4.37
## Apps    93.56    93.59     93.60     93.61     93.62     93.62     9
3.62
##       15 comps  16 comps  17 comps
## X        96.71     98.97    100.00
## Apps     93.62     93.62     93.62

# choose principal components k
validationplot(CollegePLS, val.type = "MSEP", legendpos = "topright")
```




```
# Training error
CollegePLS.train = predict(CollegePLS, train[,names(train)!="Apps"], nc
omp=6)
PLSTrainMSE = mean((CollegePLS.train - College[-test_id,"Apps"])^2)
PLSTrainMSE

## [1] 1014074

# test error
CollegePLS.test = predict(CollegePLS, test[,names(test)!="Apps"], ncomp
=6)
PLSTestMSE = mean((CollegePLS.test - College[test_id,"Apps"])^2)
PLSTestMSE

## [1] 1374134
```

Comment: According to the plot, we can find that when k=6, the MSEP is quite small and keeps almost constant later. Thus we choose 12 principal components through CV. Then the corresponding training error is 1014074 and test error is 1374134.

4. Comment on the results obtained, including also the methods from homework 6. Which approach would you recommend for this dataset and why?

Comment: For the *PCR method*, the corresponding training error is 993164.6 and test error is 1300431. For the *PLS method*, the corresponding training error is 1014074 and test error is 1374134. For the *Linear Regression method*, the corresponding training error is 993164 and test error is 1300431. For the *Forward selection method*, the training and test error is 1043037 and 1334782. For the *backward selection method*, the training and test error is 1021693 and 1355206. For the *AIC*, the training and test error are 1001215 and 1282321. For the *BIC*, the training and test error are 1033273 and 1380054. For the *ridge regression*, the training error is 1385774 and the test error is 1223317. For the *lasso regression*,the training MSE is 993997 and test MSE is 1293278. According to the result shown above, the Ridge Regression method has the smallest test error(1223317). Thus Ridge Regression method is the most suitable for this dataset.