# STATS 415 Homework 7 Solutions

*March 15, 2018*

This homework continues Homework 6, with the same task of predicting the number of applications received using the other variables in the `College` data set. Use the exact same split into training and test data as you used in Homework 6.

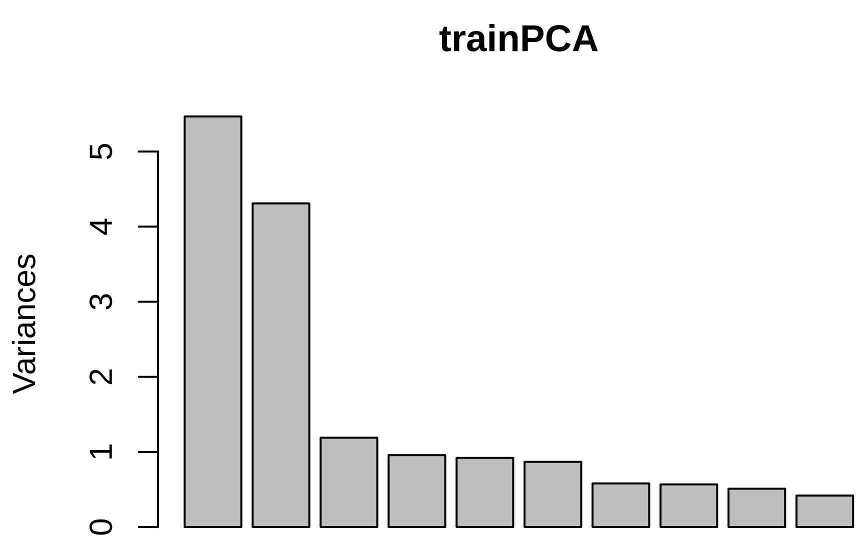
We start by reconstructing the data from homework 6.

```
set.seed(23456)
test_id <- sample(1:nrow(College), size = trunc(0.3 * nrow(College)))
test <- College[test_id, ]
train <- College[-test_id, ]
```

1. [5 points] Perform Principal Component Analysis on the predictors. Make a scree plot of the eigenvalues. How many eigenvalues does one need to explain 90% of the variance in the data? Report loadings of the first two PCs. Interpret them if you can.

Note that we are scaling the data: this is because the predictors all have different units and we don't want those units to drive choice of PCs.

```
X <- model.matrix(Apps ~ ., data = train)[, -1]
trainPCA <- prcomp(X, center = T, scale = T)
plot(trainPCA)
```



```r
summary(trainPCA)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.3383 2.0760 1.09045 0.97864 0.95962 0.93183
## Proportion of Variance 0.3216 0.2535 0.06995 0.05634 0.05417 0.05108
## Cumulative Proportion  0.3216 0.5752 0.64510 0.70143 0.75560 0.80668
##                            PC7    PC8     PC9    PC10   PC11    PC12
## Standard deviation     0.76222 0.7535 0.71438 0.64701 0.5989 0.53851
## Proportion of Variance 0.03418 0.0334 0.03002 0.02462 0.0211 0.01706
## Cumulative Proportion  0.84086 0.8743 0.90428 0.92890 0.9500 0.96706
##                           PC13    PC14    PC15    PC16    PC17
## Standard deviation     0.42278 0.40126 0.33510 0.28973 0.15501
## Proportion of Variance 0.01051 0.00947 0.00661 0.00494 0.00141
## Cumulative Proportion  0.97757 0.98704 0.99365 0.99859 1.00000
```

From the summary, we see that 9 PCs are required to explain 90% of the variance in the data. The loadings of the first two PCs are below:

```r
trainPCA$rotation[, 1:2]
```

```
##                     PC1         PC2
## PrivateYes  -0.21719312  0.30832675
## Accept       0.01354467 -0.42308644
## Enroll       0.05759724 -0.44186744
## Top10perc   -0.33794373 -0.14304740
## Top25perc   -0.30659474 -0.17748745
## F.Undergrad  0.08276223 -0.44503906
## P.Undergrad  0.12708142 -0.28870327
## Outstate    -0.37518094  0.03955743
## Room.Board  -0.27910545 -0.03274659
## Books       -0.03461765 -0.06568874
## Personal     0.15319038 -0.16596749
## PhD         -0.22981506 -0.26756998
## Terminal    -0.24451063 -0.25548777
## S.F.Ratio    0.27216743 -0.11560800
## perc.alumni -0.29665727  0.07953914
## Expend      -0.33455104 -0.07861896
## Grad.Rate   -0.29696625 -0.00861864
```

Principal components are a linear combination of the predictors, weighted by "loadings". The loadings for the first two PCs are reported above. Loadings within a PC can be interpreted as a whole: together they indicate what variables the PC deems are important (relative magnitude), and how they are correlated within that PC (relative sign). For example, the first PC seems to capture a similar amount of variability in all the predictors, except for `Books`, `F.Undergrad`, `Accept`, and `Enroll`, which might suggest that the data do not exhibit much variation in those predictors. The first two PCs also capture, for example, a negative relationship between `S.F.Ratio` and `perc.alumni`, suggesting that colleges that have a higher percentage of their alumni donate also tend to have smaller class sizes (a lower student-to-faculty ratio).

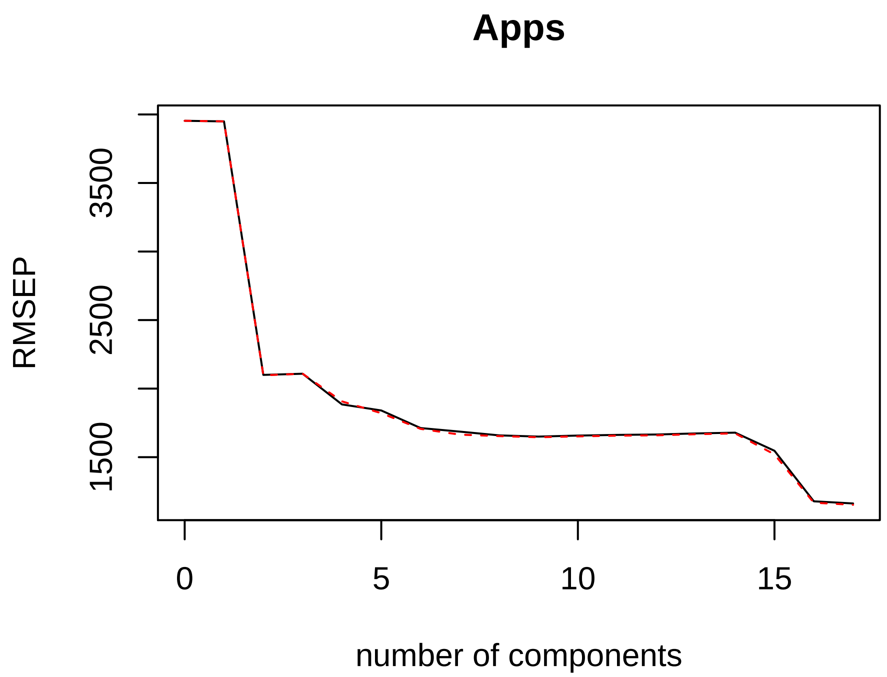
2. [5 points] Fit a PCR model on the training set, with the number of principal components $K$ chosen by cross-validation. Report the training and test error obtained, along with the value of $K$ selected.

```r
set.seed(23456)
trainPCR <- pcr(Apps ~ ., data = train, scale = T, validation = "CV")
```

```r
summary(trainPCR)
```

```
## 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 comps
## CV            3954     3949     2100     2109     1885     1841     1712
## adjCV         3954     3950     2097     2109     1907     1821     1706
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1686     1659     1650      1657      1662      1665      1673
## adjCV     1664     1654     1645      1652      1657      1659      1668
##        14 comps  15 comps  16 comps  17 comps
## CV         1679      1547      1178      1162
## adjCV      1673      1520      1168      1153
##
## 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 comps
## X       87.43    90.43     92.89     95.00     96.71     97.76     98.70
## Apps    84.01    84.34     84.49     84.51     84.52     84.52     84.53
##       15 comps  16 comps  17 comps
## X        99.36     99.86    100.00
## Apps     91.37     93.46     93.62
```

```r
validationplot(trainPCR)
```




The above cross-validation results suggest that we should choose $K = 17$ principal components to minimize CV error.

```
trainMSE.PCR <- mean((predict(trainPCR, newdata = train, ncomp = 17) - train$Apps)^2)
trainMSE.PCR
```

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

```
testMSE.PCR  <- mean((predict(trainPCR, newdata = test,  ncomp = 17) - test$Apps)^2)
testMSE.PCR
```

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

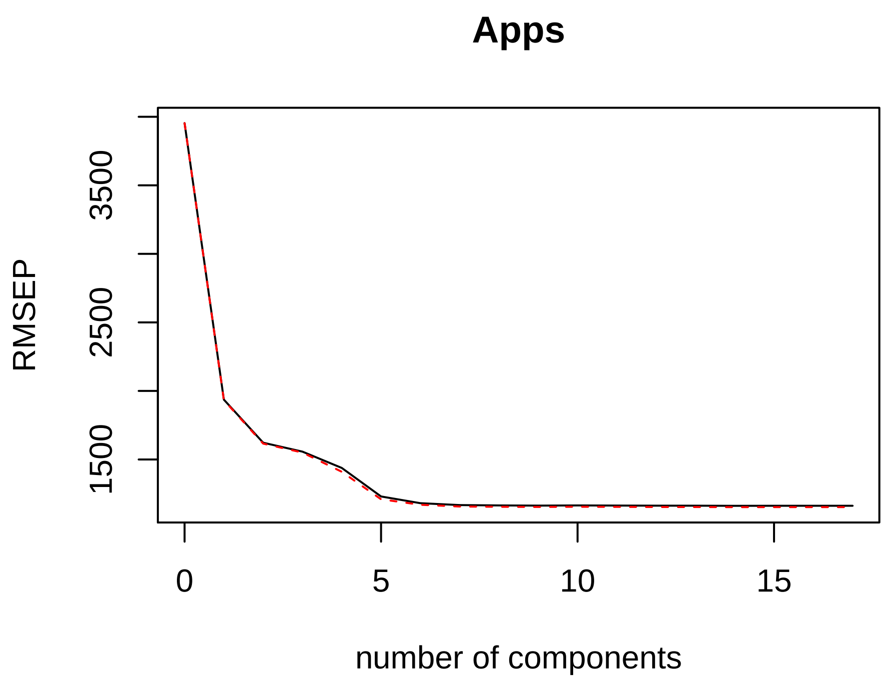
3. [5 points] Fit a PLS model on the training set, with the number of principal components $K$ chosen by cross-validation. Report the training and test error obtained, along with the value of $K$ selected

```
set.seed(23456)
trainPLS <- plsr(Apps ~ ., data = train, scale = T, validation = "CV")
summary(trainPLS)
```

```
## 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 comps
## CV            3954     1937     1622     1557     1438     1230     1181
## adjCV         3954     1933     1616     1551     1410     1210     1170
```

```
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1167     1165     1163      1165      1164      1163      1163
## adjCV     1157     1155     1154      1155      1154      1153      1153
##        14 comps  15 comps  16 comps  17 comps
## CV         1162      1162      1162      1162
## adjCV      1152      1153      1153      1153
##
## 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 comps
## X       78.98    82.34     86.65     89.38     90.80     92.58     94.37
## Apps    93.56    93.59     93.60     93.61     93.62     93.62     93.62
##       15 comps  16 comps  17 comps
## X        96.71     98.97    100.00
## Apps     93.62     93.62     93.62
```

```
validationplot(trainPLS)
```




The cross-validation results above suggest we should choose $K = 14$ components to minimize CV error (and choose the simplest model that achieves that minimum).

```
trainMSE.PLS <- mean((predict(trainPLS, newdata = train, ncomp = 14) - train$Apps)^2)
trainMSE.PLS
```

```
## [1] 993169.5
```

```
testMSE.PLS  <- mean((predict(trainPLS, newdata = test,  ncomp = 14) - test$Apps)^2)
testMSE.PLS
```

```
## [1] 1300759
```

4. [5 points] Comment on the results obtained, including also the methods from homework 6. Which approach would you recommend for this dataset and why?

Recall that MSE is in units of the response variable, squared. Here, this makes for extremely high MSEs: this complicates the process of finding meaningful performance differences between methods. Furthermore, MSEs do not account for model complexity. We transform them to adjusted $R^2$ values, instead:

```r
adjR2 <- function(MSE, Y, numpred) {
  # MSE is the MSE to be converted to an adjusted R^2
  # Y is the *full response vector* used to compute the MSE
  # numpred is the number of predictors in the model
  n <- length(Y)
  TSS <- sum((Y - mean(Y))^2)
  RSS <- n * MSE
  1 - (RSS/(n - numpred - 1)) / (TSS / (n - 1))
}
```

We tabulate the results below:

| Method | Num. Predictors | Training Error | Test Error | Train R^2 | Test R^2 |
|---|---|---|---|---|---|
| OLS | 17 | 993164.6 | 1300431 | 0.934 | 0.896 |
| Forward | 7 | 1043037.0 | 1334782 | 0.932 | 0.898 |
| Backward | 7 | 1021693.0 | 1355206 | 0.934 | 0.897 |
| AIC | 10 | 1001215.0 | 1282321 | 0.935 | 0.901 |
| BIC | 6 | 1033273.0 | 1380054 | 0.933 | 0.895 |
| Ridge | 17 | 1385774.0 | 1223317 | 0.908 | 0.902 |
| Lasso | 17 | 993997.0 | 1293278 | 0.934 | 0.897 |
| PCR | 17 | 993164.6 | 1300431 | 0.934 | 0.896 |
| PLS | 14 | 993169.5 | 1300759 | 0.935 | 0.898 |

We see that the methods perform similarly in terms of their performance on test data. In particular, ridge regression achieves the highest adjusted $R^2$ on the test data, followed closely by the AIC-selected model, PLS, and forward selection.

Because the AIC-selected model achieves a high adjusted $R^2$ and uses fewer predictors than the other top performers, we might prefer it. However, the BIC-selected model uses the fewest predictors and still achieves a comparable $R^2$ value, making it the most interpretable model of those compared.