STOR 565 Fall 2019 Homework 4

Ted Henson

Remark. This homework aims to help you further understand the model selection techniques in linear model. Credits for **Theoretical Part** and **Computational Part** are in total 100 pt. For **Computational Part** , please complete your answer in the **RMarkdown** file and summit your printed PDF homework created by it.

Computational Part

1.(*10 pt*) Consider the Nba data posted on the Sakai class site. Create a new data frame that contains the following columns:

- team
  - wins
  - points
  - points3
  - free_throws
  - off_rebounds
  - def_rebounds
  - assists
  - steals
  - personal_fouls

(a) Create box plots of the quantitative features (i.e. all but) teams to see if you should scale the data when performing PCA. Describe your findings in words.

```
library(readr, quietly = T)
library(tidyverse, quietly = T)
```

```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v ggplot2 3.2.1     v forcats 0.4.0
```

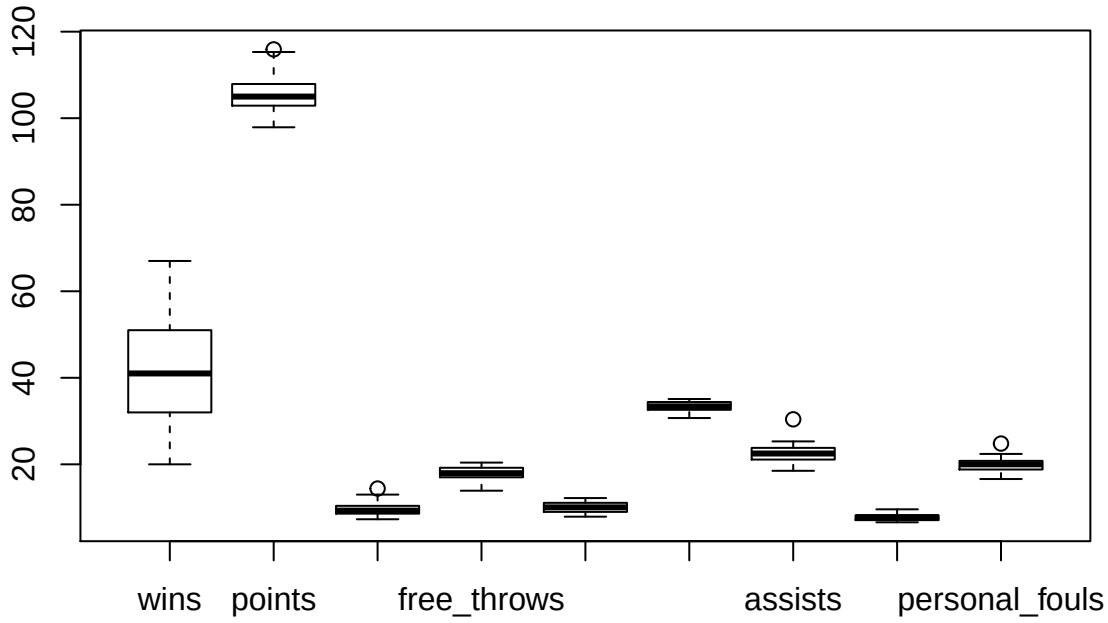
```
## -- Conflicts ------------------------------------------------------- tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x tidyr::expand()     masks Matrix::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x tidyr::pack()       masks Matrix::pack()
## x tidyr::unpack()     masks Matrix::unpack()
## x purrr::when()       masks foreach::when()
```

```
nba_teams_2017 <- read_csv("~/Machine Learning/nba-teams-2017.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   team = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
nba = nba_teams_2017 %>% dplyr::select(team,
                                       wins,
                                       points,
                                       points3,
                                       free_throws,
                                       off_rebounds,
                                       def_rebounds,
                                       assists,
                                       steals,
                                       personal_fouls)

boxplot(nba[,2:ncol(nba)])
```



We should scale the data as the features have different units of measurement. Wins and points have the largest variations.

(b) Obtain PC loadings of the first four princple components (PCs). **Display only the first few elements of each!**

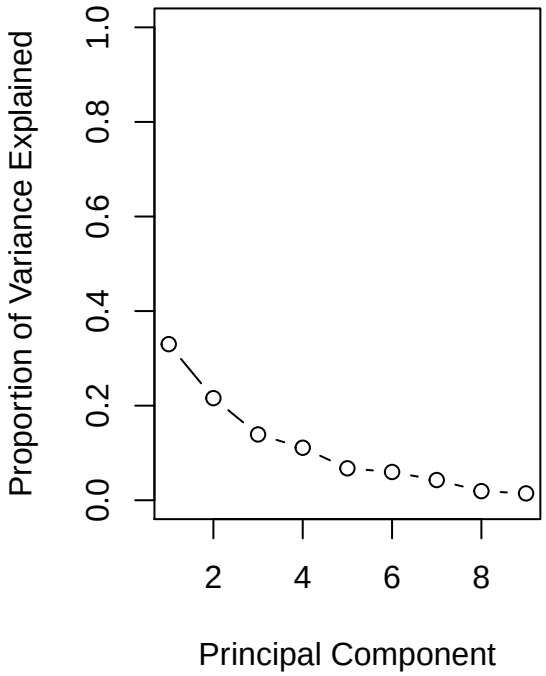
```
pc.out = prcomp(nba %>% select(-team), scale = TRUE)

pc.out$rotation[,1:4]
```

```
##                       PC1         PC2         PC3         PC4
## wins           -0.42366308  0.07555818 -0.11681772  0.21657745
## points         -0.50246947 -0.21708761  0.19677723 -0.07946391
## points3        -0.41664778  0.17268215 -0.08316881 -0.51204012
```

```
## free_throws    -0.24452950 -0.41519726  0.30946852 -0.33272209
## off_rebounds    0.08111297 -0.39160091  0.47926467  0.46463787
## def_rebounds   -0.26053718  0.26461371  0.57662166  0.15459073
## assists        -0.45236958  0.05322237 -0.26482231  0.27220000
## steals         -0.20525546 -0.41895791 -0.45168498  0.37698249
## personal_fouls  0.11583180 -0.58585494 -0.09277492 -0.34335891
```

(c) Plot a scree plot describing the amount explained by the various PCs.

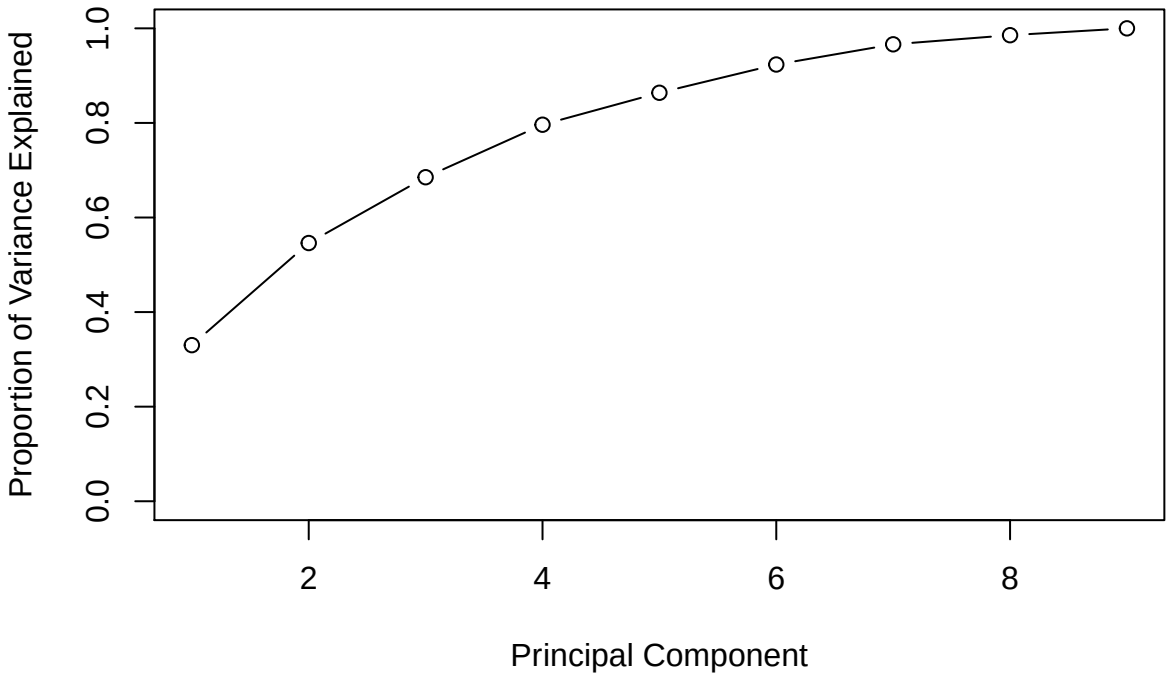
```
pr.var <- pc.out$sdev^2
pve = pr.var / sum(pr.var)
par(mfrow = c(1,2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0,1), type =
```



(d) Make another plot showing the cumulative percent of the variance explained. Precisely: for each $1 \leqslant k \leqslant 10$ you are plotting

$$\frac{\sum_{j=1}^{k} d_j^2}{\sum_{j=1}^{10} d_j^2}$$

```
plot(cumsum(pve), xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0,
```



(e) If you were to retain all PCs which explain at least 90% of the variance, how many PCs would you retain?

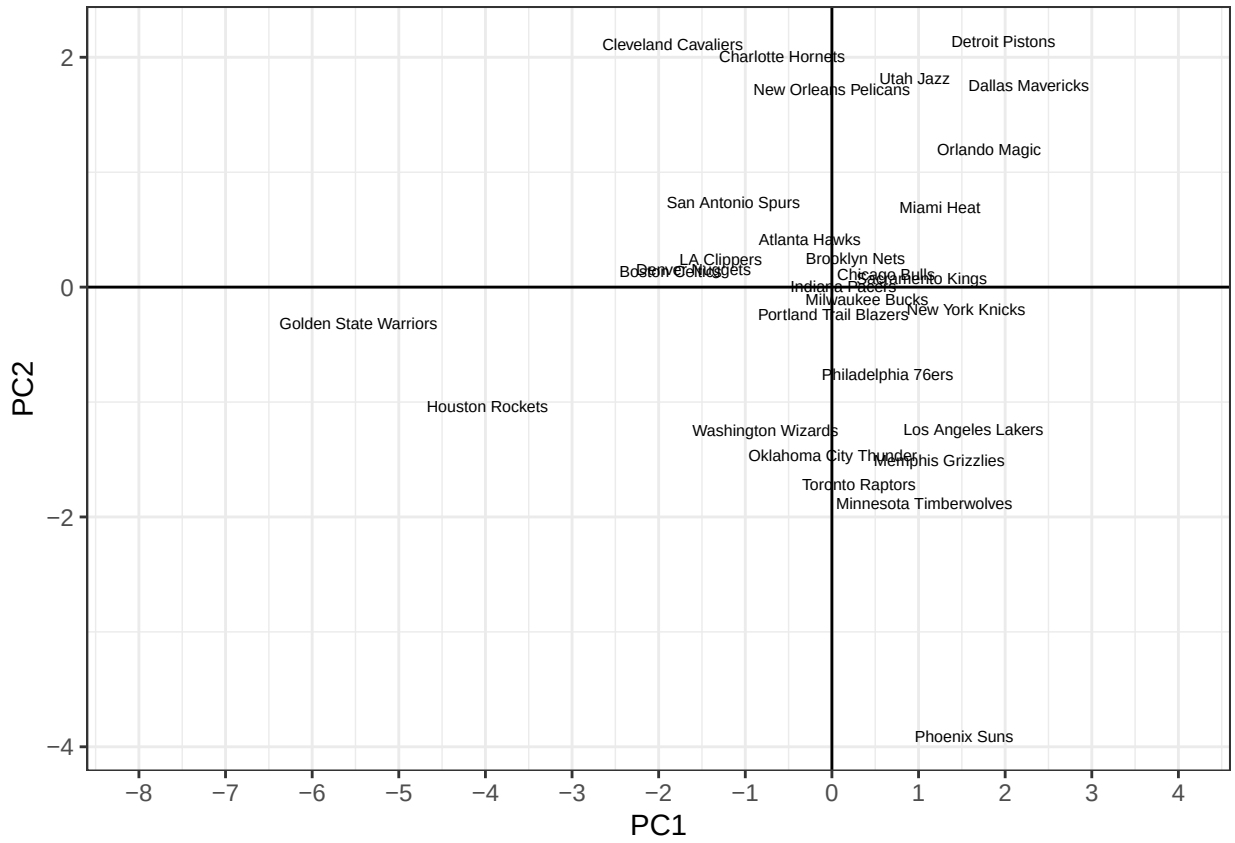
```
cumsum(pve)[6]
```

```
## [1] 0.92353
```

You would retain 6 principal components.

(f) Plot PC1 vs PC2 with the team names and describe your findings.

```
pca_scores <- pc.out$x
pca_scores_teams <- pca_scores %>%
data.frame() %>%
mutate(team = nba$team) %>%
select(team, everything())


ggplot(pca_scores_teams, aes(x = PC1, y = PC2)) +
geom_vline(xintercept = 0) +
geom_hline(yintercept = 0) +
geom_text(aes(label = team), size = 2) +
scale_x_continuous(breaks = -10:10) +
coord_cartesian(xlim = c(-8, 4)) +
theme_bw()
```



The best teams appear to have a low PC1 and low PC2 score, which makes sense as PC1 places a heavy negative weight on wins and points. The worst teams have a high PC1 scores as a result, particularly those who struggled on offense (Mavericks, Grizzlies, and Magic all were bottom half of the league in scoring and wins). A low PC2 score corresponds to a team that commits a lot of fouls and steals (the suns had the lowest PC2 and the most fouls). Our first two PCs combined together give a reasonable job of seperating out the teams.

2. (*15 pt*). **Important:** Please see the "Principal_components.rmd" file under the R demonstration folder in Lecture 5 for Sakai to see the code for manipulating figures. Using code like that demonstrated in class, download the .png file containing an image of a house posted to Redfin in the Data folder on Sakai. Note, you may have to download this first and then open it from your own computer. Set X to be the pixel intensity associated with the **red color** in the image using code like that performed in class. See the *Value* section of `?readPNG` to remind yourself of the organization of the raster array output of that function.

Answer the following questions:

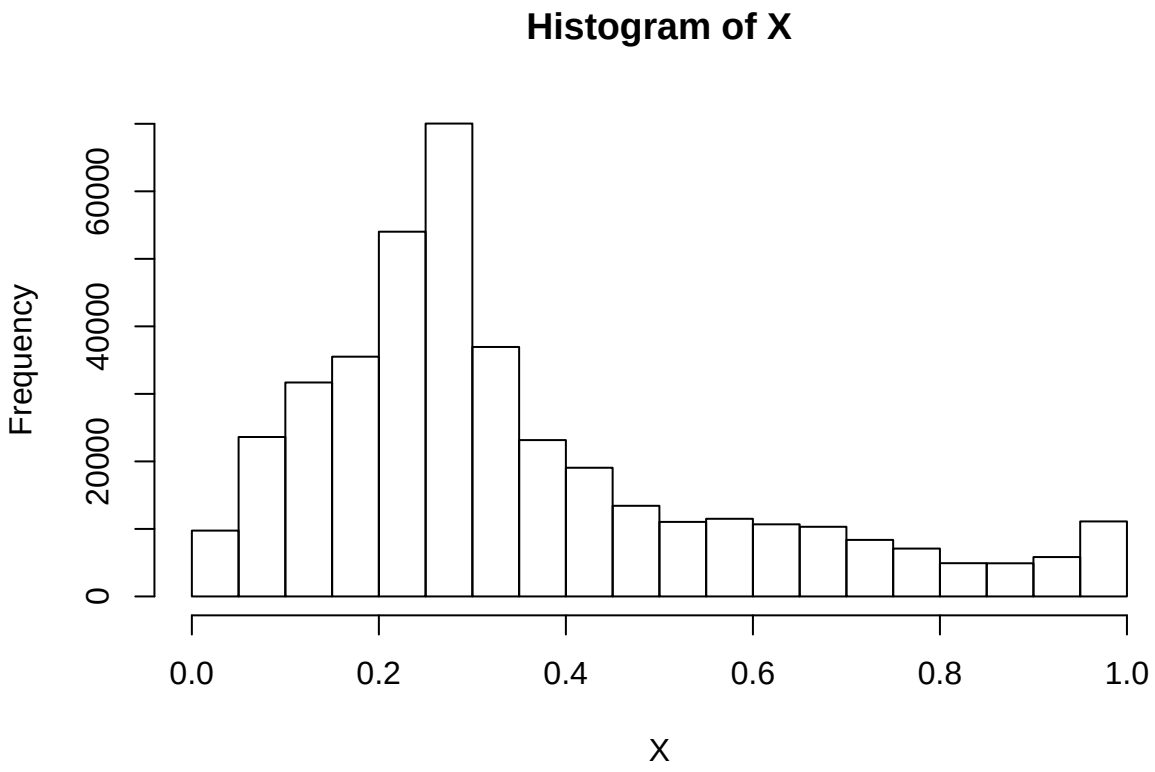
(a) What are the dimensions of X? Plot a histogram of the pixel intensities within the image.

```
library(png, quietly = T)
library(grid, quietly = T)

directory <- "Redfin_house.png"

house <- readPNG(directory)
house_plot <- as.raster(house)

X <- house[, , 1]
```

```
dim(X)
```

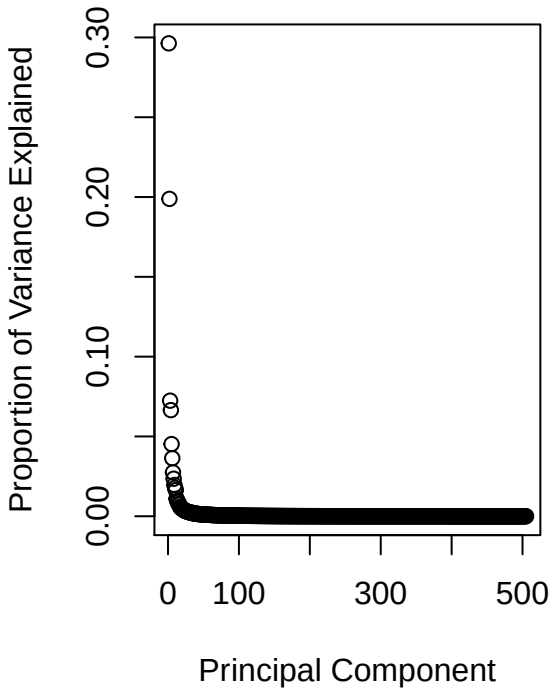
```
## [1] 505 798
```

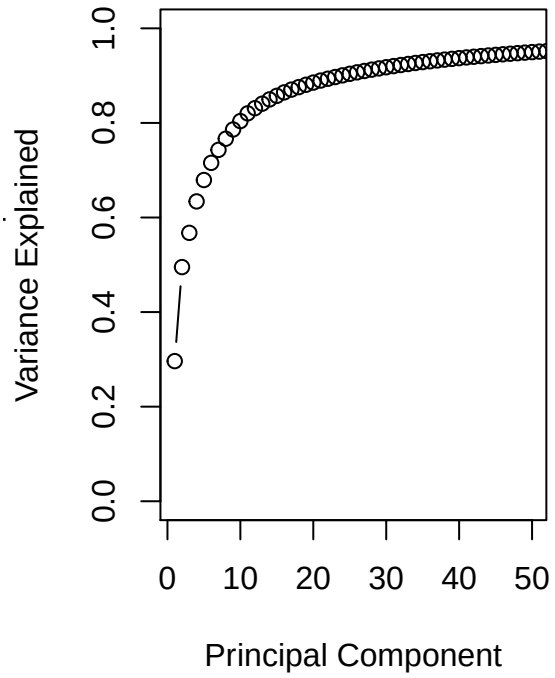
```
hist(X)
```



(b) Plot the scree plots for this data, which illustrate the percentage variation explained against the number of principal components and the cumulative percentage variation explained against the number of principal components. How many PCs are needed to explain 90% of the total variation of X?

```
pr.out <- prcomp(X, scale = TRUE)

pr.var <- pr.out$sdev^2
pve <- pr.var / sum(pr.var)
par(mfrow = c(1,2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained")
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of
Variance Explained", ylim = c(0, 1), type = "b", xlim = c(1, 50))
```






```
which(cumsum(pve) > .90)[1]
```

```
## [1] 24
```

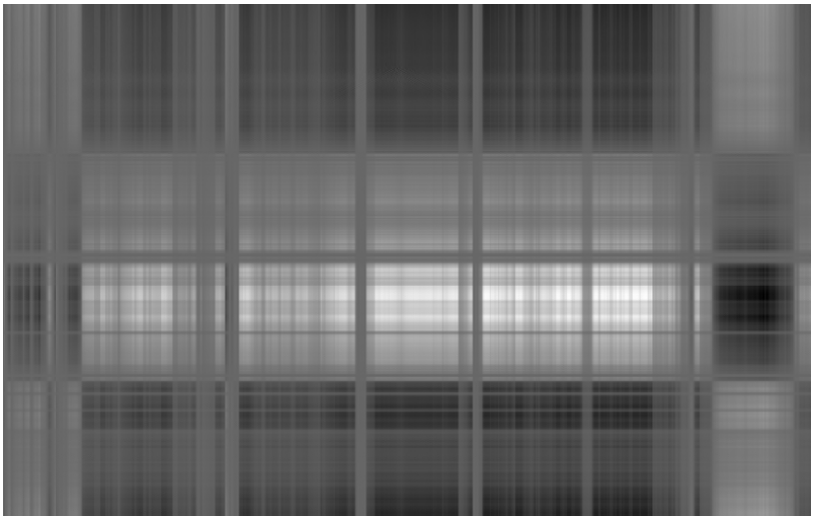
```
print('24 PCs are needed to explain 90% of the total variation')
```

```
## [1] "24 PCs are needed to explain 90% of the total variation"
```

(c) For $d = 1, 5, 10, 15, 20, 30, 50, 100, 200$ project the image onto the first d principal components and plot the resulting compressed image for each d. For each of the nine plots, include the cumulative percentage variation explained by the projection in the title of your plots.

```
W <- pr.out$rotation #the loading matrix
pc.image <- list()
num.pcs <- c(1, 5, 10, 15, 20, 30, 50, 100, 200)
#scale the original image
Image <- scale(X)
for(j in 1:length(num.pcs)){
u.proj <- W
#we will only use the first num.pcs PC loadings so set the remaining to 0
u.proj[, (num.pcs[j] + 1) : 396] <- 0
#Make the projection
projection <- (Image%*%u.proj)%*%t(u.proj)
#to draw an image, values need to be between 0 and 1
scaled <- (projection - min(as.numeric(projection)))
scaled <- scaled / max(as.numeric(scaled))
pc.image[[j]] <- as.raster(scaled)
```

```
pct.var = paste(cumsum(pve)[j], 'of total variation explained')
plot(pc.image[[j]])
title(main = pct.var,
      col = 'black',
      font = 3)
}
```



0.495140532287466 of total variation explained

0.567553432267641 of total variation explained

0.634043468120682 of total variation explained

0.679243724229033 of total variation explained

0.715608659603433 of total variation explained

0.743068687007813 of total variation explained

0.766576633908117 of total variation explained

0.786146959945272 of total variation explained

3. (Prediction, Textbook 6.9, *15 pt*) In this exercise, we will predict the number of applications received using the other variables in the `College` data set from `ISLR` package.

(a) Randomly split the data set into two sets, a training and a test set, as evenly as possible. There is an odd number of observations, so make the test set larger.

```
library(caret, quietly = T)
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

```
smp_size <- floor(0.5 * nrow(College))

## set the seed to make your partition reproducible
set.seed(123)

# College$Private  = ifelse(College$Private == 'Yes', 1, 0)
# College$Private = as.factor(College$Private)
row.names(College) = NULL
```

```
College = College %>% dplyr::select(Apps, everything())

College = as.data.frame(College)

  train_ind <- sample(seq_len(nrow(College)), size = smp_size)

train <- College[train_ind, ]
test <- College[-train_ind, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
lm.mod = lm(Apps ~ ., data = train)

preds = predict(lm.mod, test)
rmse = sqrt(mean((test$App - preds)^2))
rmse
```

```
## [1] 1172.175
```

(c) Fit a ridge regression model on the training set, with λ chosen by 5-fold cross-validation. Report the test error obtained.

```
train$Private = ifelse(train$Private == 'Yes', 1, 0)
test$Private = ifelse(test$Private == 'Yes', 1, 0)

ridge = cv.glmnet(x = as.matrix(train[,2:ncol(train)]),
                  y = train$Apps,
                  alpha = 0, nfolds = 5)
preds = predict(ridge, as.matrix(test[,2:ncol(test)]))
rmse = sqrt(mean((test$App - preds)^2))
print(paste('Test Error:', rmse))
```

```
## [1] "Test Error: 1709.16118657082"
```

(d) Fit a LASSO model on the training set, with λ chosen by 5-fold cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
train$Private = ifelse(train$Private == 'Yes', 1, 0)
test$Private = ifelse(test$Private == 'Yes', 1, 0)

lasso = cv.glmnet(x = as.matrix(train[,2:ncol(train)]),
                  y = train$Apps,
                  alpha = 1, nfolds = 5)
preds = predict(lasso, as.matrix(test[,2:ncol(test)]))
rmse = sqrt(mean((test$App - preds)^2))
print(paste('Test Error:', rmse))
```

```
## [1] "Test Error: 1333.36843910504"
```

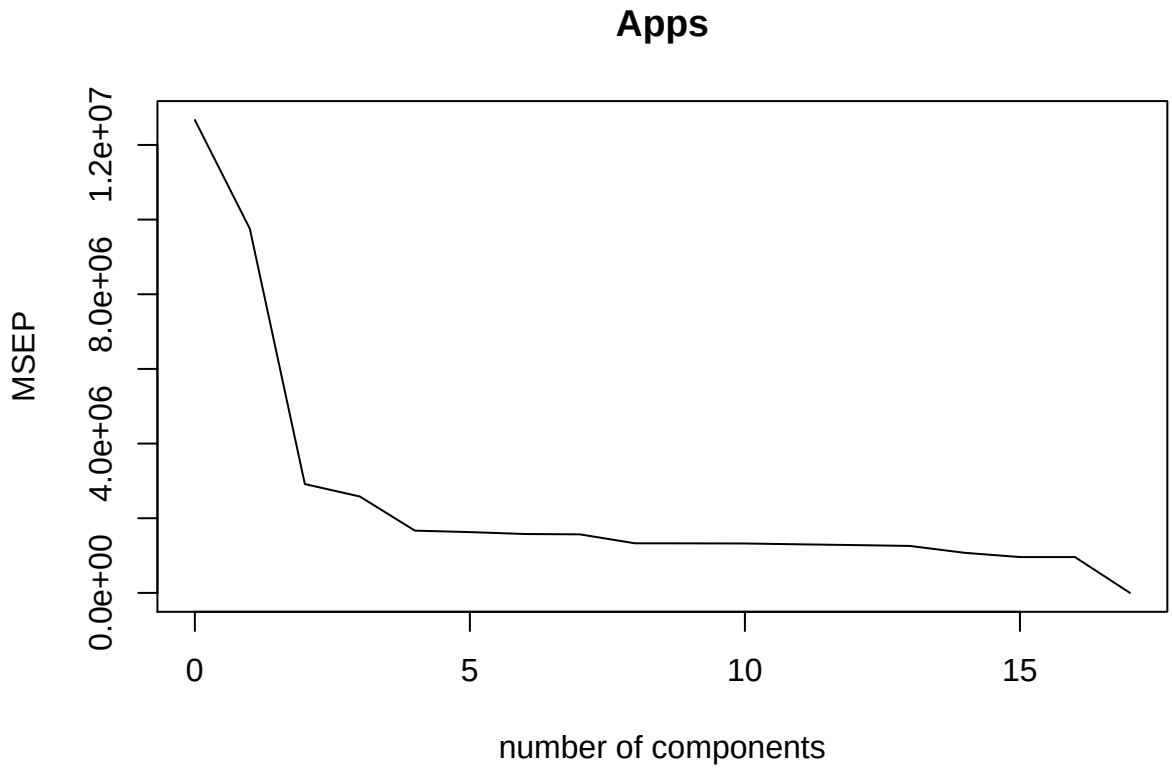
```
lasso.coeff = nrow(summary(coef(lasso, s= 'lambda.min')))
print(paste('Number of Non Zero Coefficients:', lasso.coeff))
```

```
## [1] "Number of Non Zero Coefficients: 11"
```

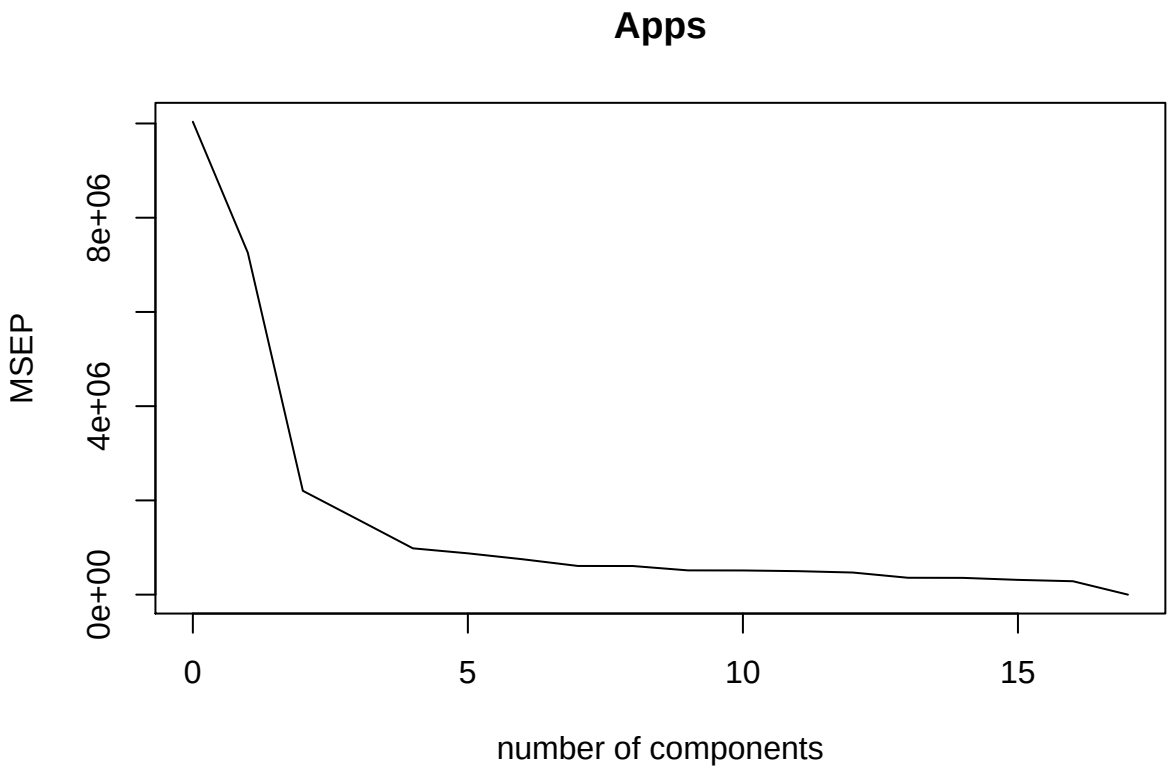
(e) Fit a PCR model on the training set, with M chosen by 5-fold cross-validation. Report the test error obtained, along with the value of M selected by cross-validation.

```
folds <- createFolds(train$Apps, k = 5)
train1 = train[folds$Fold1,]
```

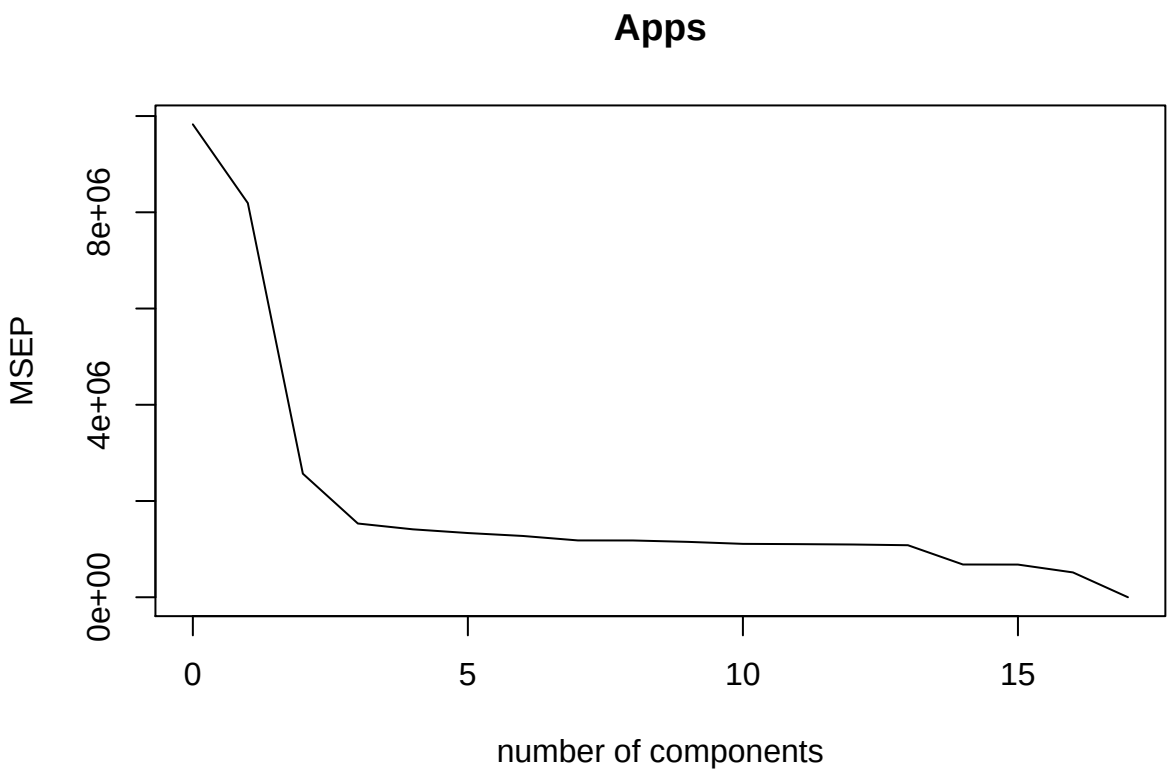
```
train2 = train[folds$Fold2,]
train3 = train[folds$Fold3,]
train4 = train[folds$Fold4,]
train5 = train[folds$Fold5,]

train1[,3:ncol(train1)] = scale(train1[,3:ncol(train1)])
train2[,3:ncol(train2)] = scale(train2[,3:ncol(train2)])
train3[,3:ncol(train3)] = scale(train3[,3:ncol(train3)])
train4[,3:ncol(train4)] = scale(train4[,3:ncol(train4)])
train5[,3:ncol(train5)] = scale(train5[,3:ncol(train5)])
#
#
#
# pc.out.1 = prcomp(train1[,2:ncol(train1)])
# pc.out.2 = prcomp(train2[,2:ncol(train2)])
# pc.out.3 = prcomp(train3[,2:ncol(train3)])
# pc.out.4 = prcomp(train4[,2:ncol(train4)])
# pc.out.5 = prcomp(train5[,2:ncol(train5)])

# train[,3:ncol(train)] = scale(train[,3:ncol(train)])
# train = as.data.frame(train)


pc.out.1 = pcr(Apps ~ . , data = train1)
validationplot(pc.out.1, val.type = "MSEP")
```



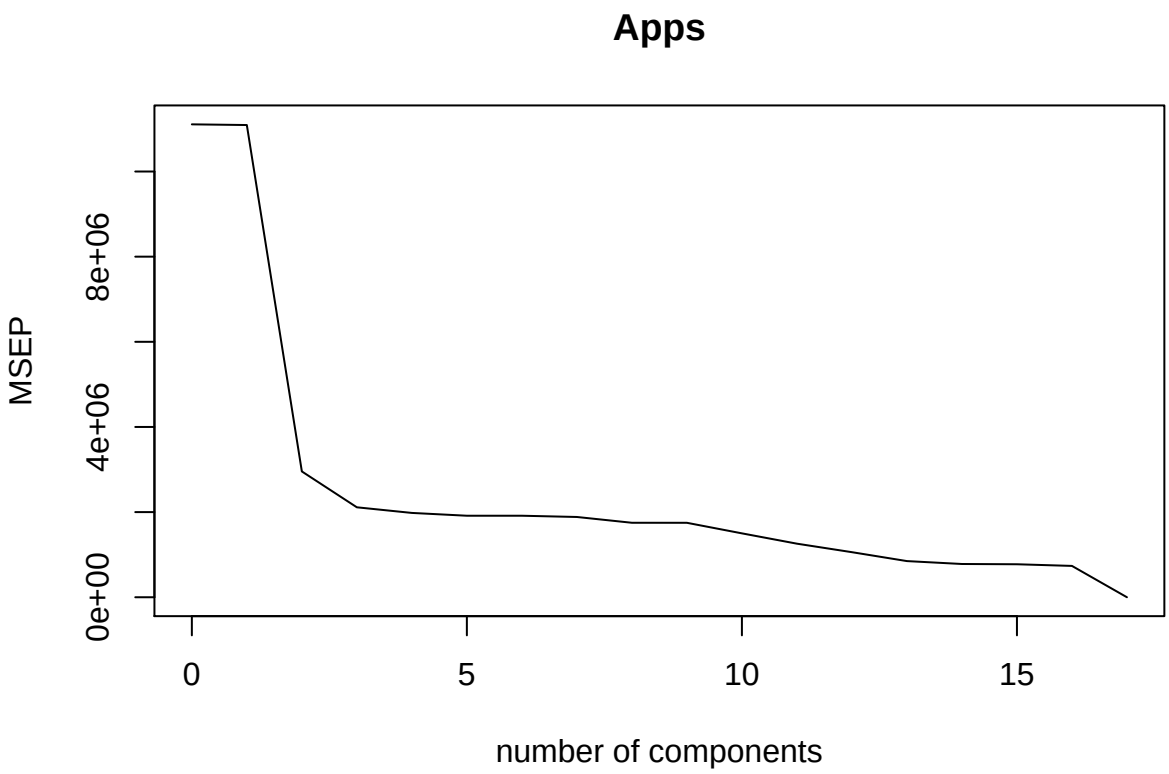
```
pc.out.2 = pcr(Apps ~ . , data = train2)
validationplot(pc.out.2, val.type = "MSEP")
```



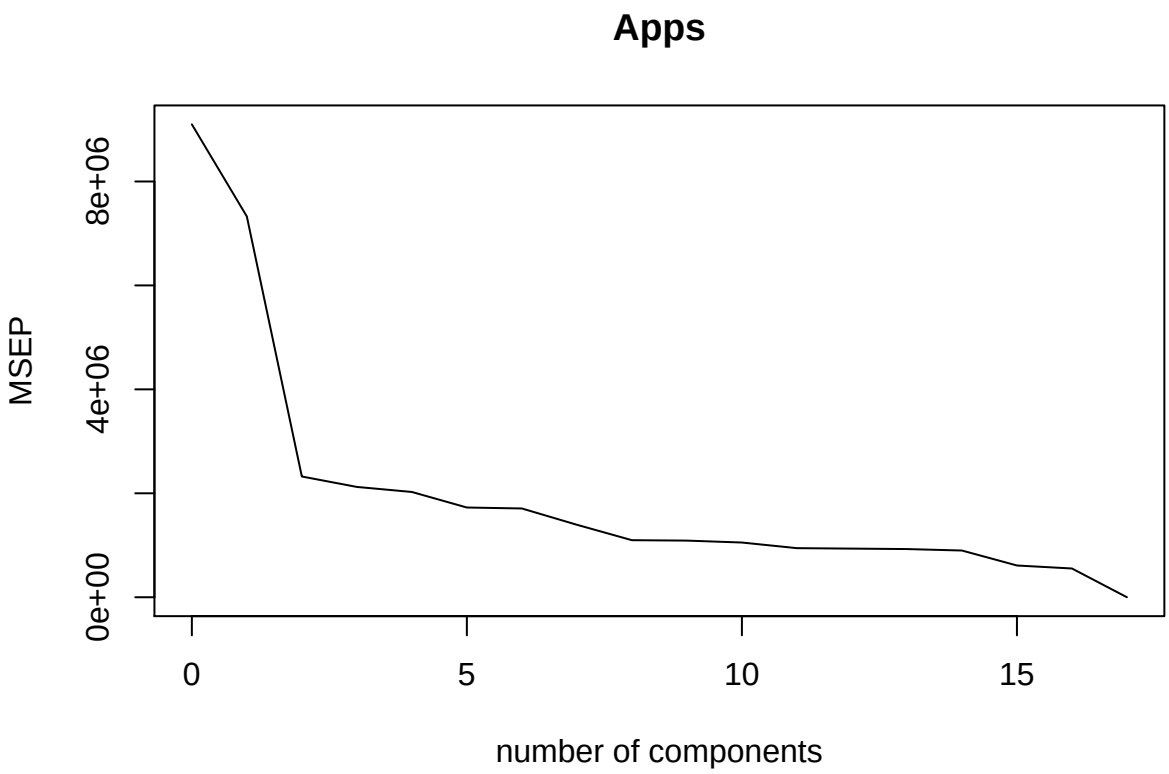

```
pc.out.3 = pcr(Apps ~ . , data = train3)
validationplot(pc.out.3, val.type = "MSEP")
```




```
pc.out.4 = pcr(Apps ~ . , data = train4)
validationplot(pc.out.4, val.type = "MSEP")
```




```
pc.out.5 = pcr(Apps ~ . , data = train5)
validationplot(pc.out.5, val.type = "MSEP")
```




```
train[,3:ncol(train)] = scale(train[,3:ncol(train)])
```

We chose 3 PCs as that is where the elbow is on all of our cross validated pcrs.

```
pcr.final.model  = pcr(Apps ~ ., data = train, ncomp = 3)

test[,3:ncol(test)] = scale(test[,3:ncol(test)])

preds = predict(pcr.final.model, test)
rmse = sqrt(mean((test$App - preds)^2))
print(paste('Test Error:', rmse))
```

```
## [1] "Test Error: 3163.78626896325"
```

(f) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these four approaches?

The best model was the ordinary least squares followed by the lasso, ridge, and the pcr. Ordinary least squares might have done the best in prediction since a lot of our features were contributing different, but relevant information. Our shrinkage methods and pcr may have shrunk the effects of important features too much.