# STOR 565 Spring 2019 Homework 4

**Due on 02/28/2019 in Class**

*Coleman Breen*

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

- team (check)
  - wins (check)
  - points (check)
  - points3 (check)
  - free_throws (check)
  - off_rebounds (check)
  - def_rebounds (check)
  - assists (check)
  - steals (check)
  - personal_fouls (check)

(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.

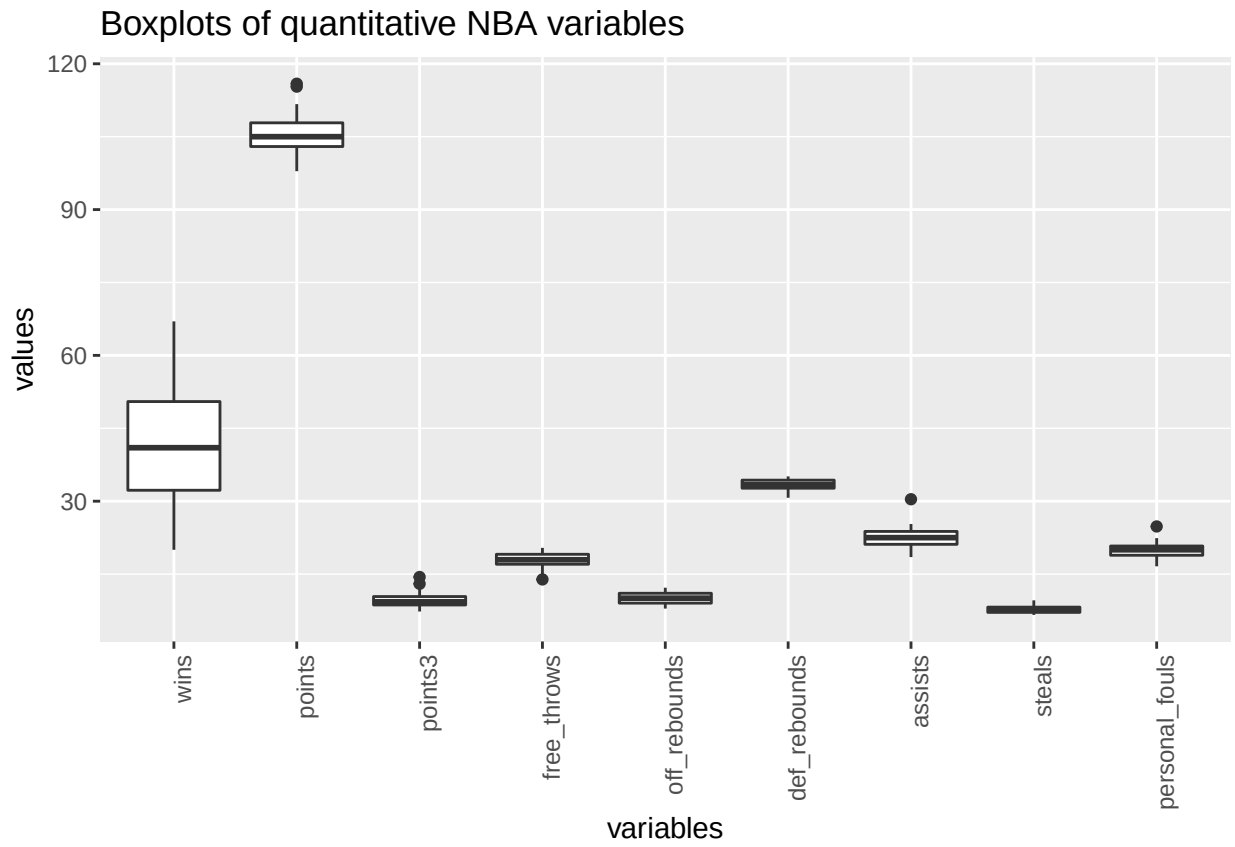
```
library(tidyverse, quietly = TRUE)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ tidy
## 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 purrr::when()       masks foreach::when()
```

```
nba.data <- read.csv("nba-teams-2017.csv")
nba.data2 <- nba.data %>%
  select(team, wins, points, points, points3,
         free_throws, off_rebounds, def_rebounds,
         assists, steals, personal_fouls)

ggplot(stack(nba.data2), aes(x = ind, y = values)) +
  geom_boxplot() +
  ggtitle("Boxplots of quantitative NBA variables") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("variables")
```




The scales are radically different. For example, the median points is about 105 while the median steals is less than 10. I will standardize the quantitative variables.

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

```
#--> Compute principle components; standardize
pr.out <- prcomp(nba.data2[,-1], scale=TRUE)

#--> Look at the loadings
head(pr.out$rotation[,1:4], 5)
```

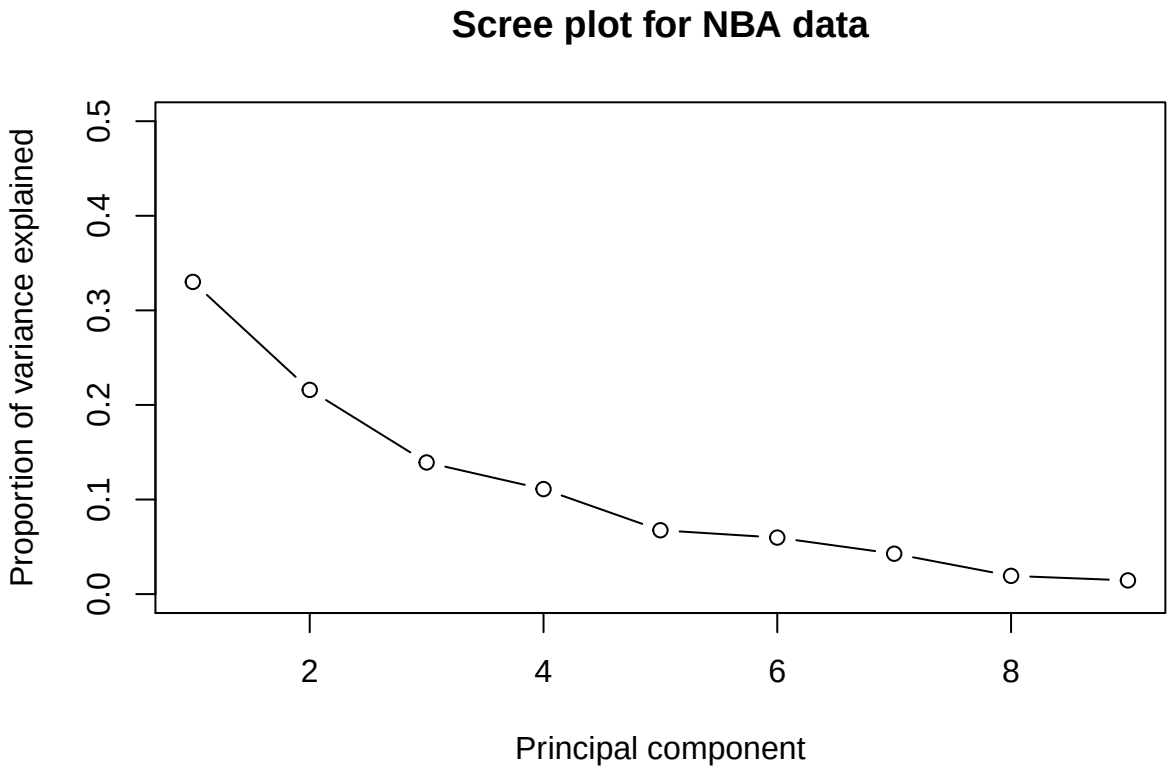
```
##                       PC1         PC2         PC3         PC4
## wins          -0.42366308  0.07555818 -0.11681772  0.21657745
## points        -0.50246947 -0.21708761  0.19677723 -0.07946391
## points3       -0.41664778  0.17268215 -0.08316881 -0.51204012
## free_throws   -0.24452950 -0.41519726  0.30946852 -0.33272209
## off_rebounds   0.08111297 -0.39160091  0.47926467  0.46463787
```

I've only shown the first four (of nine) loadings above. Additionally, I've only shown the first few elements of each (as requested).

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

```
#--> Scree
pr.var <- pr.out$sdev^2 # convert to variance
pve = pr.var/ sum(pr.var) # what's the total variance

plot(pve, main="Scree plot for NBA data",xlab = "Principal component",
```

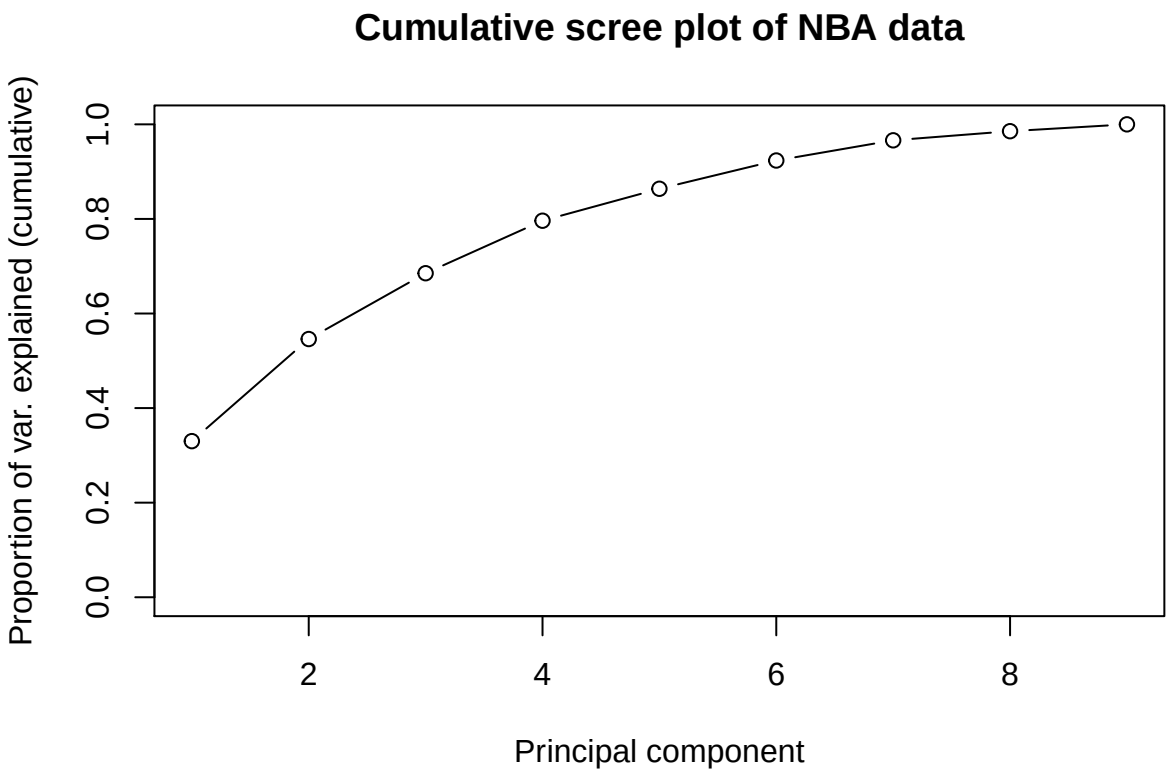
```
     ylab = "Proportion of variance explained",
     ylim =c(0,.5), type ='b')
```




(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), main="Cumulative scree plot of NBA data",
     xlab = "Principal component",
     ylab = "Proportion of var. explained (cumulative)",
     ylim =c(0,1), type='b')
```




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

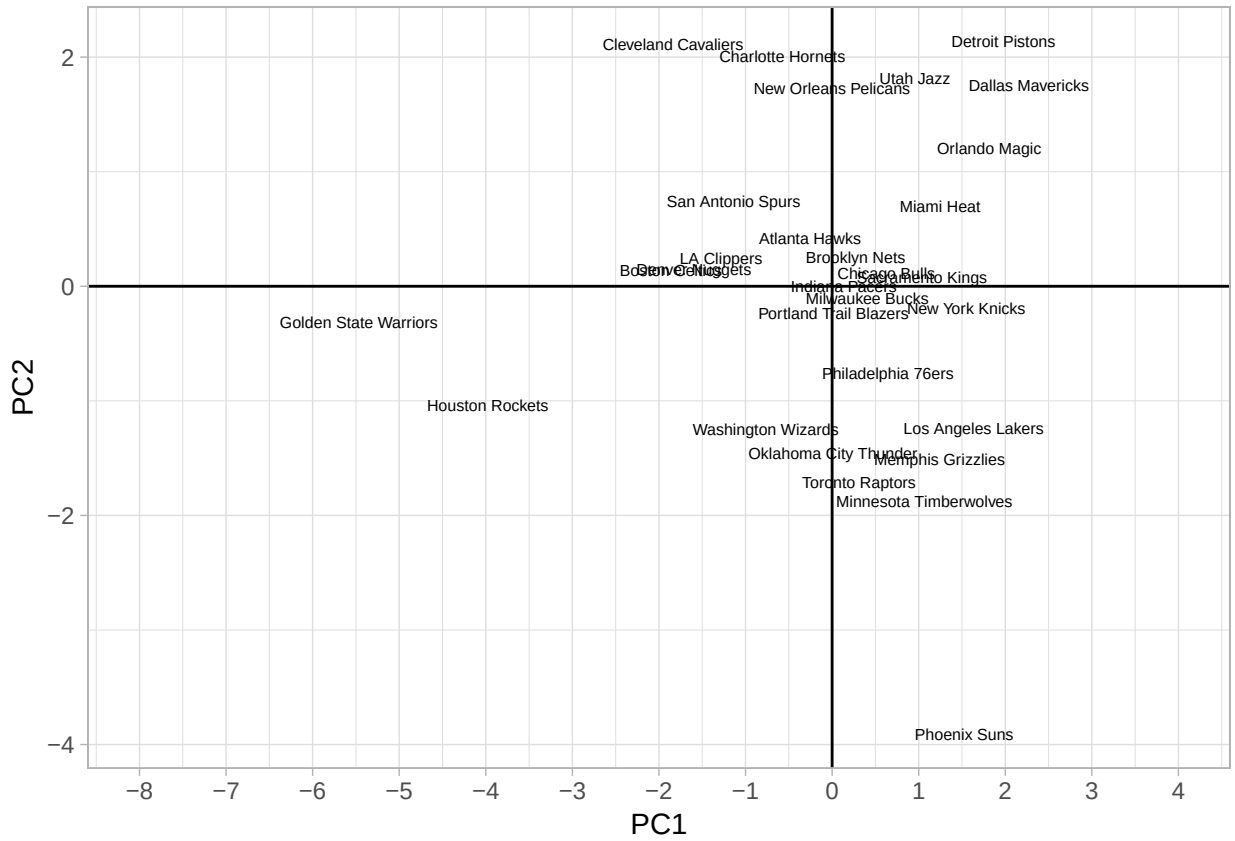
```
cdf <- cumsum(pve)
sum(cdf < .90) + 1 # all the components that don't get us there plus the one that does
```

```
## [1] 6
```

We need to include 6 PCs to account for at least 90% of the variance.

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

```
#--> Get a dataframe (c/o Dr. Bhamidi's lecture)
pca_scores <- pr.out$x
low_dim_rep <- pca_scores%>%
  data.frame() %>%
  mutate(team = nba.data$team) %>%
  select(team, everything())

#--> Plot (c/o Dr. Bhamidi's lecture)
ggplot(low_dim_rep,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_light()
```




Recall that PC1 contains mostly positive relationships between wins and other variables: points, points3, free_throws, def_rebounds, assists, and steals. Teams that are farther along in the negative PC1 direction (because wins and the other aformentioned variables are negative) do all of those things well–they rebound, assist, and score. This makes sense because Golden State and Houston are the standouts in this category. Both of these teams have been known in recent years to be offensive juggernauts.

A team that is far in the positive PC2 direction (corresponding to more wins) is expected to be strong at defensive rebounding and three point shooting. It makes sense that the Pelicans are in this category because they have/had Anthony Davis–known for his ability to grab boards. The loading for PC2 is more complicated because there are also inverse relationships between wins and points, free_throws, and off_rebounds.

All things considered, the first two principal components do a reaonable job "seperating" teams based on their reputation and play style. However, we need to incorporate more principle components to develop an even finer method to accurately characterize our teams and reduce dimensionality without losing important information.

2. (*20 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.

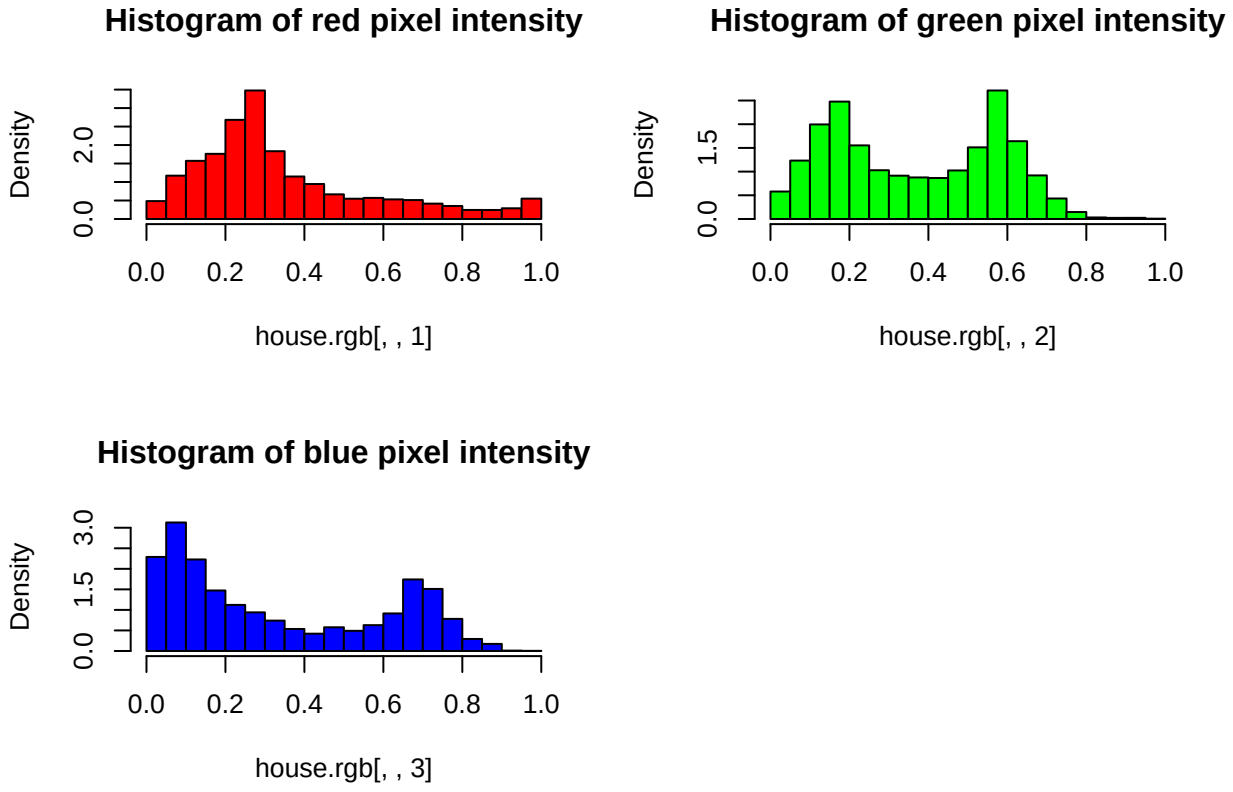
Answer the following questions:

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

```r
library(png)

#--> Read it in
house.rgb <- readPNG("Redfin_house.png")
X <- house.rgb[, , 1]

#--> Dimension
dim(X)
```

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

```r
#--> Plot
par(mfrow = c(2,2))
hist(house.rgb[, , 1], main="Histogram of red pixel intensity",
           freq=FALSE, col="red")
hist(house.rgb[, , 2], main="Histogram of green pixel intensity",
           freq=FALSE, col="green")
hist(house.rgb[, , 3], main="Histogram of blue pixel intensity",
           freq=FALSE, col="blue")
```

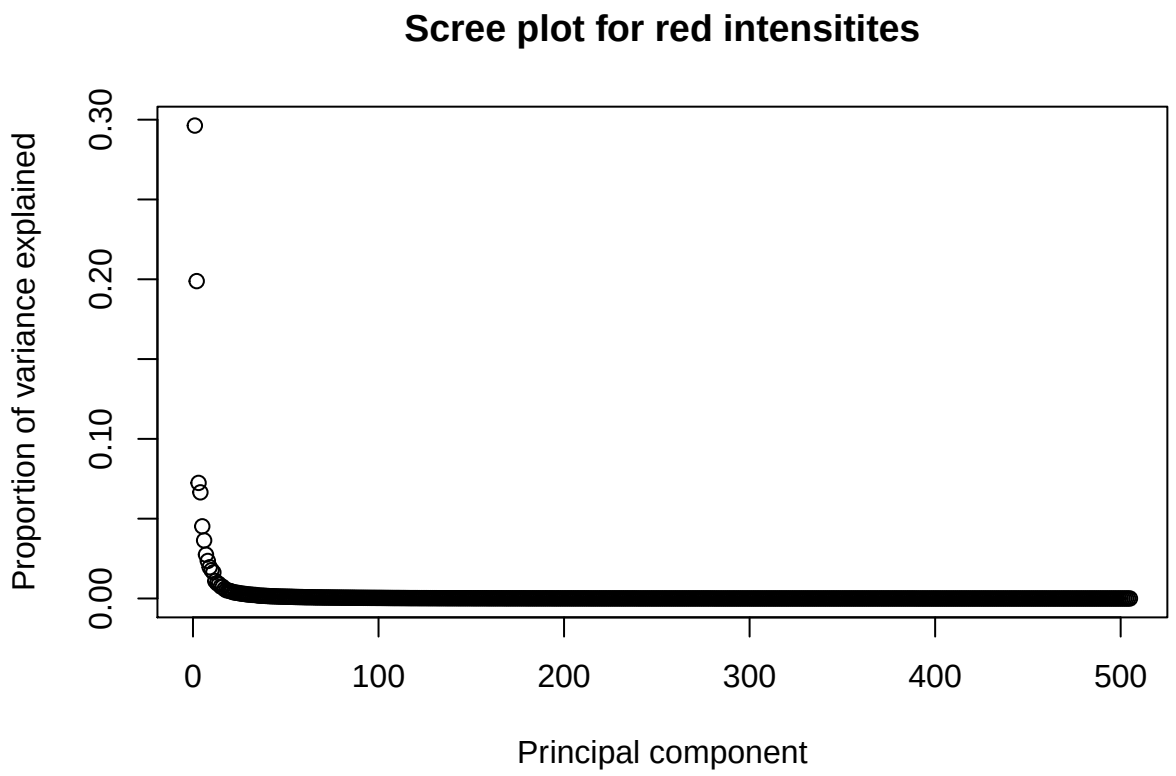



$X$, the matrix of red intensities, is 505 by 798 pixels.

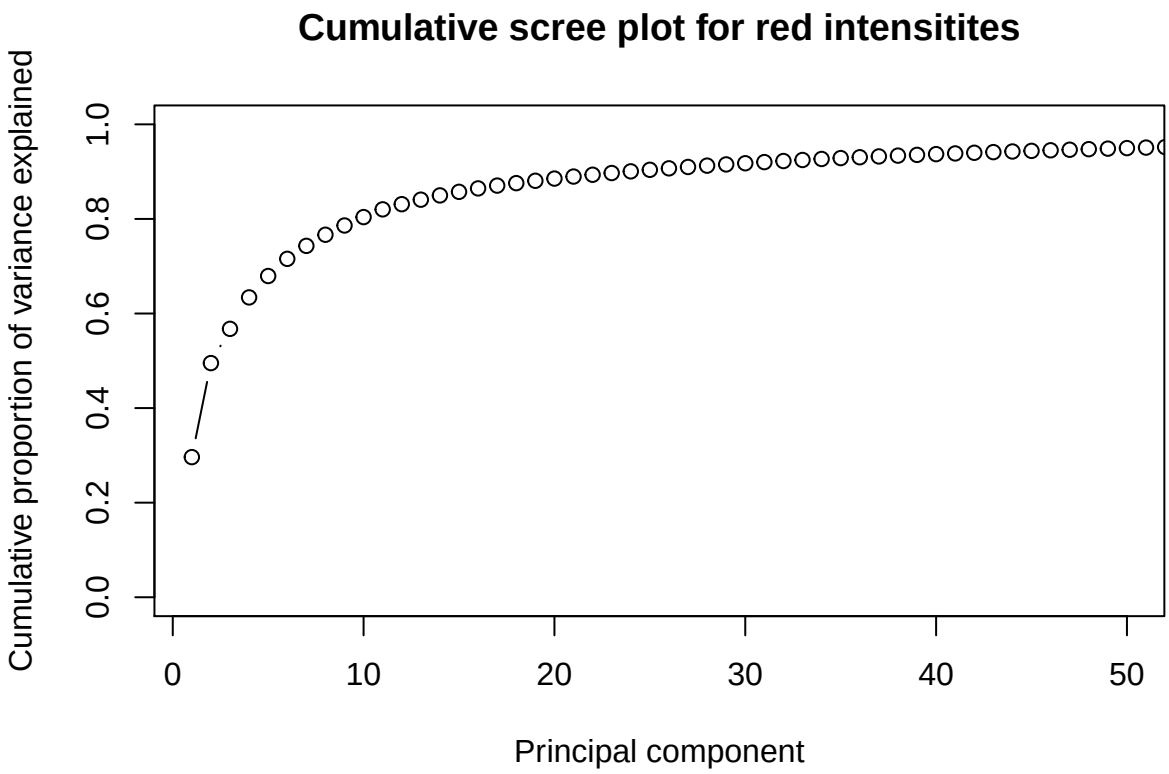
(b) Plot the scree plots for this data, which illustrate the percentage variation explained against the number of principal components and the cumulative percentage variation explained against the number of principal components. How many PCs are needed to explain 90% of the total variation of $X$?

```r
#--> Compute pc
pr.out <- prcomp(X, scale=TRUE) # scale=TRUE used in class
```

```r
pr.var <- pr.out$sdev^2 # compute variance
pve <- pr.var/ sum(pr.var) # proportion of variance

#--> Plot
plot(pve, main="Scree plot for red intensitites",
     xlab = "Principal component",
     ylab = "Proportion of variance explained")
```




```r
plot(cumsum(pve), main="Cumulative scree plot for red intensitites",
     xlab = "Principal component",
     ylab = "Cumulative proportion of variance explained",
     ylim =c(0, 1), type = "b", xlim =c(1, 50))
```




```
#--> How many do we need for 90%
cdf <- cumsum(pve)
sum(cdf < .90) + 1
```

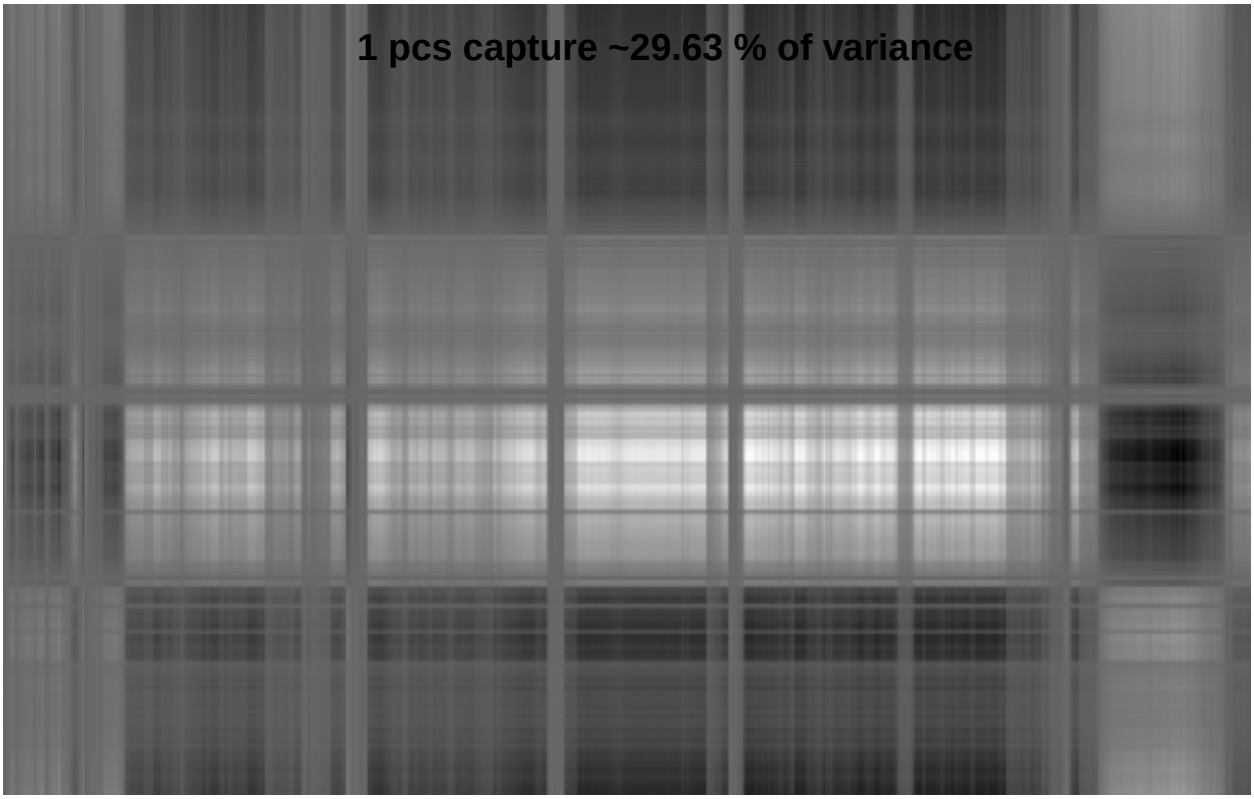
```
## [1] 24
```

See above for scree and cumulative scree plots. We need 24 principal components to account for at least 90% of the variance.

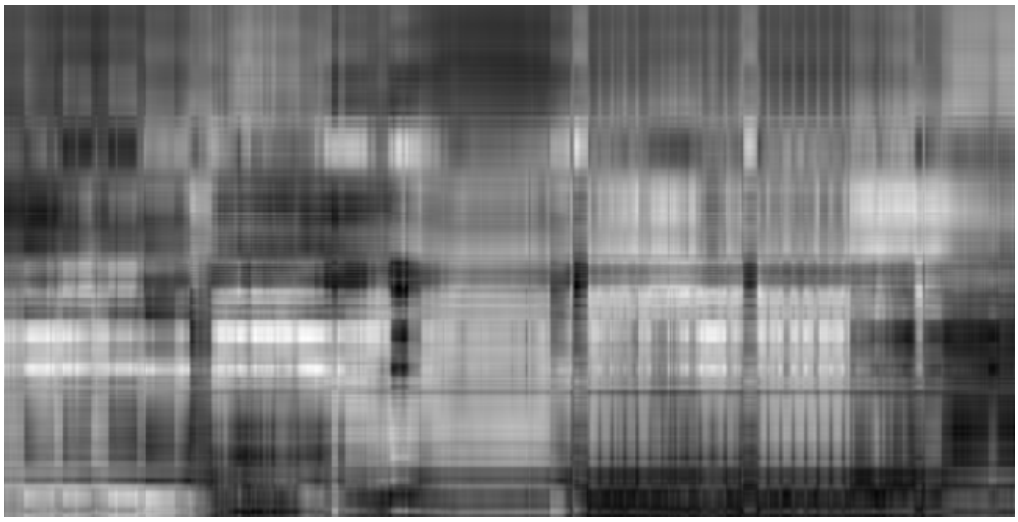
(c) For $d = 1, 5, 10, 15, 20, 30, 50, 100, 200$ project the image onto the first $d$ principal components and plot the resulting compressed image for each $d$. For each of the nine plots, include the cumulative percentage variation explained by the projection in the title of your plots.

```
d <- c(1,5,10,15,20,30,50,100,200) # PCs we want
pc.Image <- list() # initialize
Image <- scale(X) # scale to red

for (i in 1:length(d)){
  #--> Grab the pcs we want
  u.proj <- pr.out$rotation
  u.proj[, -1*(1:d[i])] <- 0 # make all except (1:d[i]) zero

  #--> Projection
  projection <- (Image%*%u.proj)%*%t(u.proj) # t is a function apparently

  #--> Draw the image, get values in [0,1]
  scaled <- (projection - min(as.numeric(projection)))
  scaled <- scaled / max(as.numeric(scaled))
```

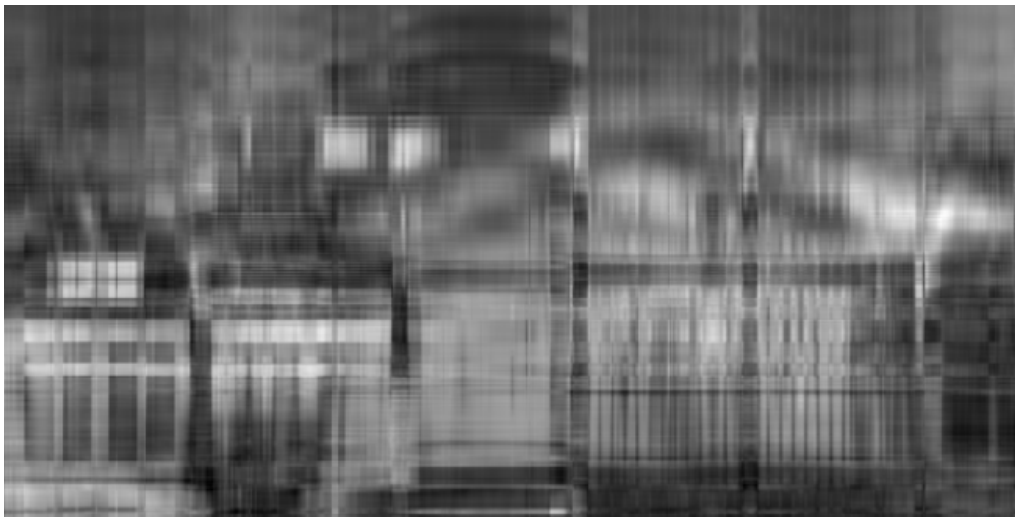
```
  pc.Image[[i]] <- as.raster(scaled)
}

#--> Loop thru and show the images
library(grid)

for (i in 1:length(pc.Image)){
  plot.new()
  (grid.raster(pc.Image[[i]]))
  title(paste0(toString(d[i]), " pcs capture ~",
    toString(round(cdf[d[i]], digits=5)*100), " % of variance"),
    font.main=2)
}
```



**5 pcs capture ~67.924 % of variance**



**10 pcs capture ~80.389 % of variance**



**15 pcs capture ~85.728 % of variance**

## 20 pcs capture ~88.53 % of variance

**30 pcs capture ~91.777 % of variance**

**50 pcs capture ~94.977 % of variance**

## 100 pcs capture ~98.046 % of variance

## 200 pcs capture ~99.595 % of variance

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.

```
data(College)
boolean.v <- rep(c(TRUE,FALSE), ceiling(nrow(College)/2))
boolean.v <- boolean.v[-1] # get it to the right size, drop the first T

set.seed(919)
shuffled <- sample(boolean.v)

train <- College[shuffled, ]
test <- College[!shuffled, ]
```

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

```
#--> Fit linear model
college.lm <- lm(Apps~., data=train)
lm.pred <- predict.lm(college.lm, newdata=select(test, -Apps))

#--> RMSE
lm.rmse <- sum((lm.pred - select(test, Apps))^2) / length(lm.pred)
lm.rmse
```

```
## [1] 1637287
```

The test error for the standard linear model is 1,637,287.

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

```
#--> Prepare data (put in matrix form)
x.train <- data.matrix(train, rownames.force = FALSE)
y.train <- train$Apps

x.test <- data.matrix(test, rownames.force = FALSE)
y.test <- test$Apps

#--> Ridge cross validation
cv.out <- cv.glmnet(x=x.train, y=y.train, alpha=0, nfolds=5)
college.ridge <- glmnet(x=x.train, y=y.train, alpha=0, lambda=cv.out$lambda.min)

#--> Fit actual model
lambda <- college.ridge$lambda
ridge.pred <- predict(college.ridge, s=lambda, newx=x.test)

#--> Error
ridge.rmse <- sum((ridge.pred - y.test)^2) / length(ridge.pred)
ridge.rmse
```

```
## [1] 551185.1
```

The test error for the ridge regression model is 551,185.1.

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

```
#--> Lasso cross validation
cv.out <- cv.glmnet(x=x.train, y=y.train, alpha=1, nfolds=5) # 1 for lasso
college.lasso <- glmnet(x=x.train, y=y.train, alpha=0, lambda=cv.out$lambda.min)

#--> Fit actual model
lambda <- college.lasso$lambda # to set s
lasso.pred <- predict(college.lasso, s=lambda, newx=x.test)

#--> Error
lasso.rmse <- sum((lasso.pred - y.test)^2) / length(lasso.pred)
lasso.rmse
```

```
## [1] 134362.1
```

```
#--> Non-zero
sum(!coef(college.lasso) == 0)
```

```
## [1] 19
```

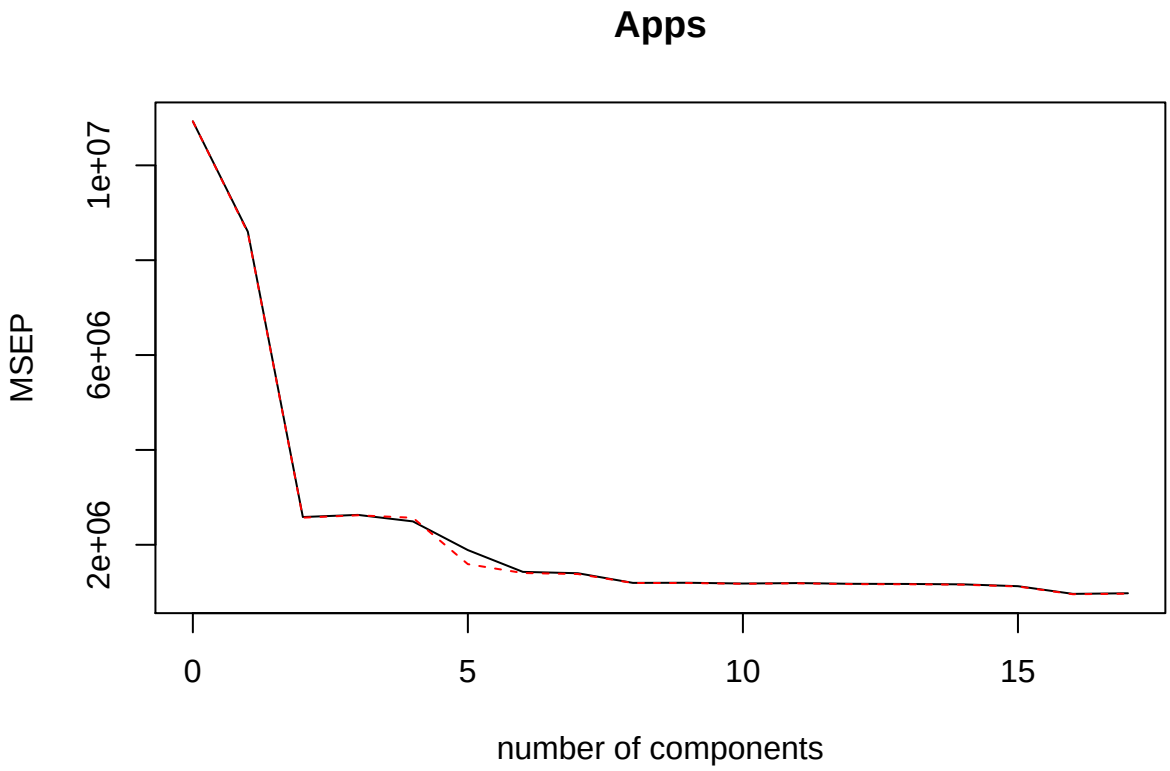
The test error for the LASSO model is 134,362.1. There are 19 non-zero coefficient estimates.

(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.

```
#--> Fit PCR
college.pcr <-pcr(Apps~., data = train, scale = TRUE, validation="CV")

#--> Summary of fit
summary(college.pcr)
```

```
## Data:    X dimension: 388 17
##  Y dimension: 388 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            3306     2933     1608     1622     1579     1374     1195
## adjCV         3306     2929     1604     1619     1603     1262     1185
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1183     1094     1095      1088      1092      1085      1083
## adjCV     1176     1093     1092      1085      1090      1082      1080
##        14 comps  15 comps  16 comps  17 comps
## CV         1080      1061     982.1     988.8
## adjCV      1076      1058     978.3     984.3
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       31.62    56.45    63.35    69.11    74.46    79.60    83.72
## Apps    23.51    77.52    77.54    79.65    88.15    88.43    88.90
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X       87.37    90.29     92.84     95.08     97.00     98.02     98.91
## Apps    89.69    90.00     90.08     90.09     90.33     90.42     90.46
##       15 comps  16 comps  17 comps
## X        99.41     99.79    100.00
## Apps     90.84     92.31     92.43
```

```
#--> How many components should we include?
validationplot(college.pcr, val.type = "MSEP")
```




```
num_comp <- college.pcr$ncomp

#--> Make predictions
pcr.pred <- predict(college.pcr, test, ncomp = num_comp)

#--> Compute MSPE
pcr.rmse <- mean((pcr.pred - y.test)^2)

#--> Report
num_comp
```

```
## [1] 17
```

```
pcr.rmse
```

```
## [1] 1637287
```

Five-fold cross validation supports using $M = 17$. Because this is all of the variables, PCR is the same as linear regression. The test error for Principle Component Regression is 1,637,287 (the same as OLS linear regression).

(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?

```
#--> Make a data frame
comp <- data.frame(lm.rmse, ridge.rmse, lasso.rmse, pcr.rmse)
comp
```

```
##   lm.rmse ridge.rmse lasso.rmse pcr.rmse
```

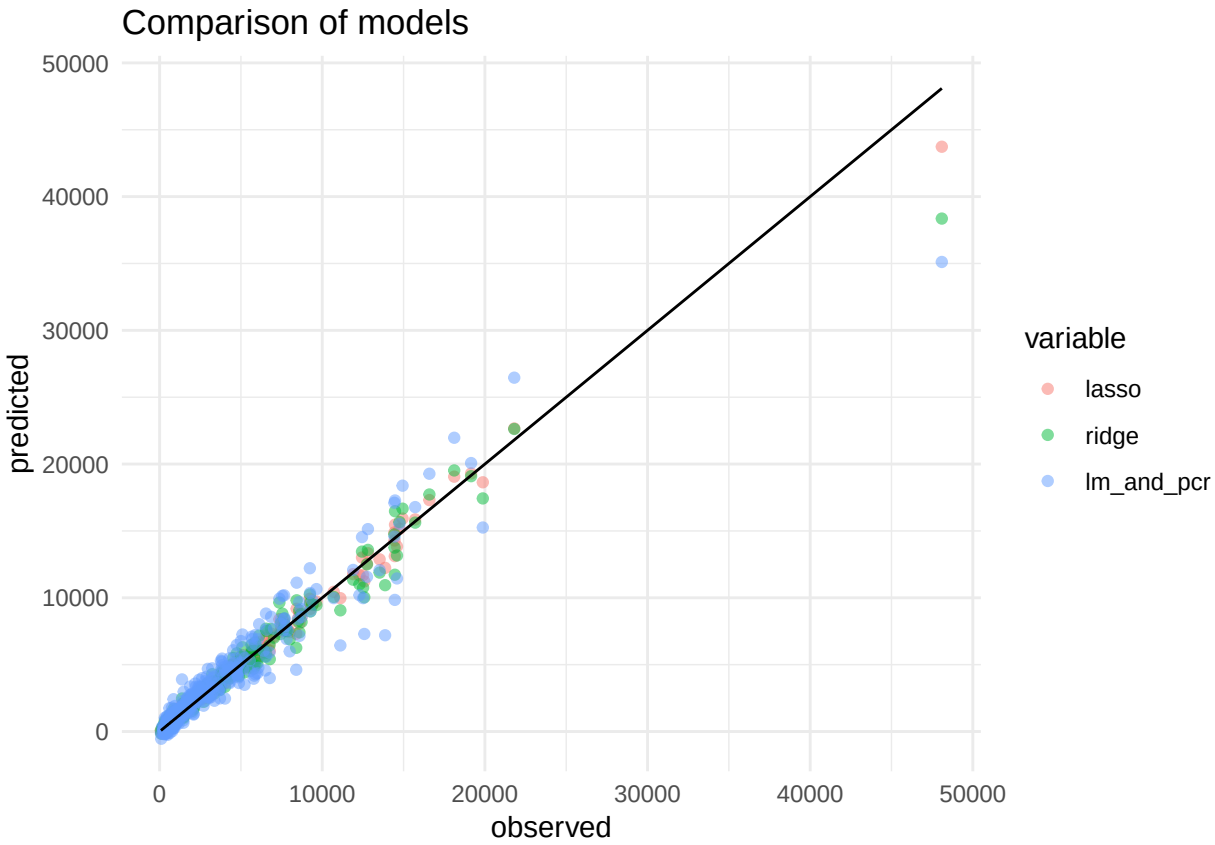
```
## 1 1637287   551185.1   134362.1  1637287
```

```
#--> Put predictions in one data frame
lasso.df <- data.frame(real.apps = y.test, lasso.predicted.apps = lasso.pred,
                       ridge.predicted.apps = ridge.pred, lm.pred)
names(lasso.df) <- c("observed","lasso", "ridge", "lm_and_pcr")

#--> Coerce data
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
lasso.df2  <- melt(lasso.df,id.var=1)

#--> Plot
ggplot(lasso.df2, aes(x=observed, y=value, color=variable)) +
  geom_point(alpha=.5) +
  geom_line(aes(x=observed,y=observed),color="black") +
  ggtitle("Comparison of models") +
  theme_minimal() +
  ylab("predicted")
```




There is a noticeable range in the prediction error committed by each of our four models. In order, LASSO performed the best (RMSE~130,000), followed by ridge (RMSE~550,000). The OLS standard linear model

(RMSE~1,600,000) and principle component regression (RMSE~1,600,000) are tied for last because they are functionally the same since the pcr includes all 17 predictors.

Based on the above plot, if we stick with LASSO, we can predict the number of applications reasonably well. Additionally, its root mean squared prediction error is an entire order of magnitude smaller than the other methods and is not an unreasonable model, so in a formal setting I would reccomend using the LASSO model for prediction.