STAT 5401 - Homework 4

Hashim Gulzar

March 28th, 2018

Quesiton 1

(a)

Let us use the code as provided:

```
sat<-read.table('sat.dat.txt', row.names = 1)
names(sat)<-c("EduExp", "PTR", "Salary", "SAT.prct", "SAT.verbal", "SAT.math", "SAT.total")
X<-as.matrix(sat[,c(1:3, 5:6, 4)])
head(X)
```

```
##            EduExp  PTR Salary SAT.verbal SAT.math SAT.prct
## Alabama     4.405 17.2 31.144        491      538        8
## Alaska      8.963 17.6 47.951        445      489       47
## Arizona     4.778 19.3 32.175        448      496       27
## Arkansas    4.459 17.1 28.934        482      523        6
## California  4.992 24.0 41.078        417      485       45
## Colorado    5.443 18.4 34.571        462      518       29
```

To calculate the correlation matrix, we can simply use the `cor` function in *R*

```
cor(X)
```

```
##                EduExp          PTR       Salary  SAT.verbal    SAT.math
## EduExp      1.0000000 -0.371025386  0.869801513 -0.41004987 -0.34941409
## PTR        -0.3710254  1.000000000 -0.001146081  0.06376664  0.09542173
## Salary      0.8698015 -0.001146081  1.000000000 -0.47696364 -0.40131282
## SAT.verbal -0.4100499  0.063766636 -0.476963635  1.00000000  0.97025604
## SAT.math   -0.3494141  0.095421730 -0.401312817  0.97025604  1.00000000
## SAT.prct    0.5926274 -0.213053607  0.616779867 -0.89326296 -0.86938393
##              SAT.prct
## EduExp      0.5926274
## PTR        -0.2130536
## Salary      0.6167799
## SAT.verbal -0.8932630
## SAT.math   -0.8693839
## SAT.prct    1.0000000
```

There is a strong correlation between `EduExp` and `Salary`, i.e the more you 'invest' in a pupil, is strongly correlated with that pupil earning a higher salary in the future, which makes sense intuitively.

`PTR` (pupil to teacher ratio) is negatively correlated with `SAT.prct`, which again makes sense. As a teacher has less time to give attention to certain pupils one-on-one, they are likely to do worse on the SAT.

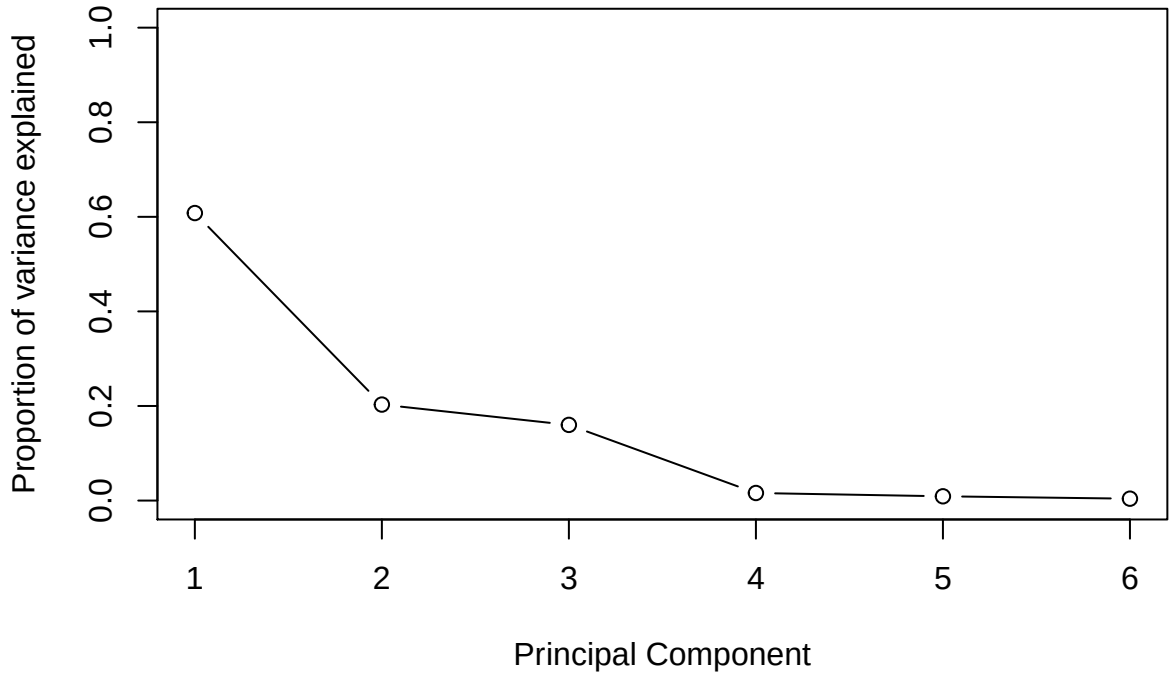
(b)

To perform a principal component analysys on the correlation matrix we can simply make use the of `prcomp` function in *R*

```
sat.eigen<-eigen(cor(X))
sat.pca<-as.matrix(X) %*% sat.eigen$vectors
```

We can now make a scree plot as follows:

```
# Calculate the variance of the principal components from the SAT dataset
sat.var<-round(sat.eigen$values/sum(sat.eigen$values), digits=3)

# Plotting it out
plot(sat.var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```



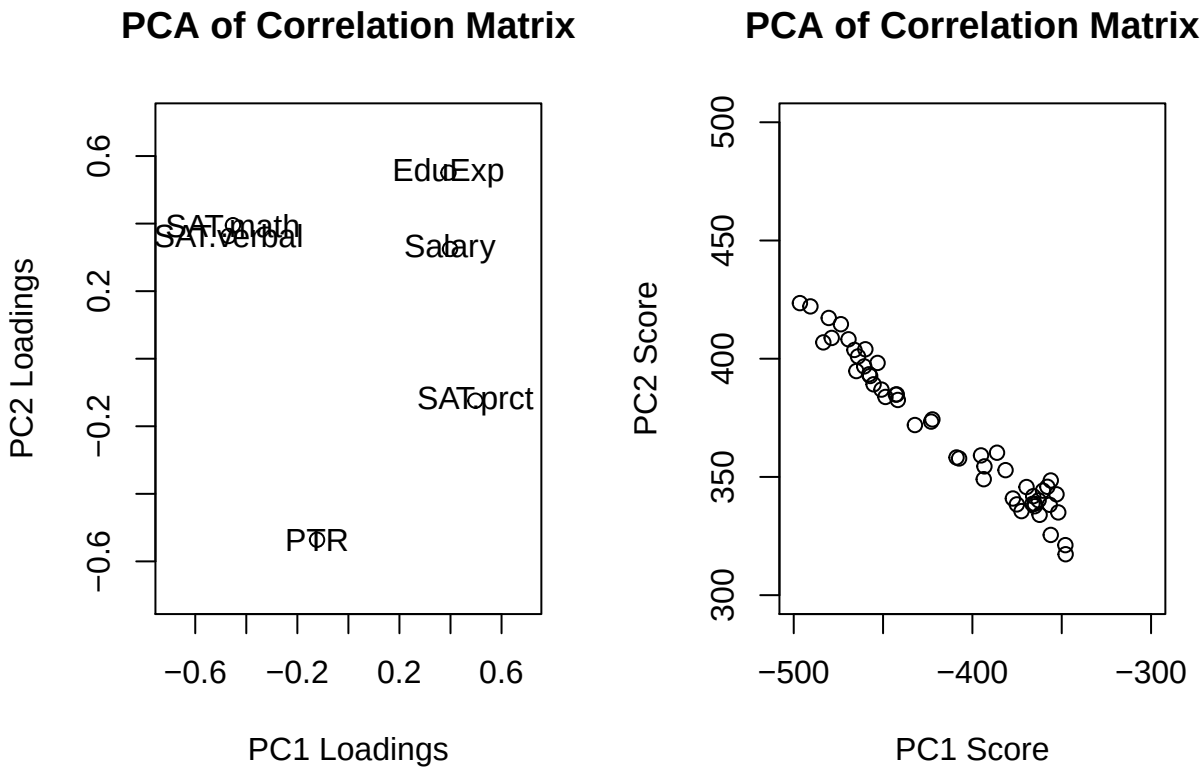
The first two PC's explain close to 80% of the variance.

(c)

We can plot both the PC loadings and PC scores using the `plot` function in *R* as follows:

```
par(mfrow=c(1,2))
plot(sat.eigen$vectors[,1:2],
     xlab = "PC1 Loadings",
     ylab = "PC2 Loadings",
     main = "PCA of Correlation Matrix",
     xlim = c(-0.7, 0.7),
     ylim = c(-0.7, 0.7))
text(sat.eigen$vectors[,1:2], labels = colnames(X))
```

```r
plot(sat.pca[,1:2],
     xlab = "PC1 Score",
     ylab = "PC2 Score",
     main = "PCA of Correlation Matrix",
     xlim = c(-500, -300),
     ylim = c(300, 500))
```



Based on the loadings of the first Principal component it would seem that `SAT.math`, `SAT.verbal`, `Salary`, and `EduExp` are strongly correlated with PC1. The first principal component increases with `EduExp`, `Salary` and `SAT.prct`. This suggests, they vary together.

(d)

```r
W<-matrix(c(sat_dat[,8], sat.pca[,1]), nrow=50, ncol=2)
cor(W)
```

```
##             [,1]       [,2]
## [1,]  1.0000000 -0.9861733
## [2,] -0.9861733  1.0000000
```

The correlation between `SAT.total` and the first Principal Component is -0.98 and therefore strongly negatively correlated.

Quesiton 2

(a)

As stated all variables are normally distributed, therefore we can calculate the conditional distribution as follows:

$$\mathbf{X}|\mathbf{y} \sim N(\boldsymbol{\mu}_X + \boldsymbol{\Sigma}_{Xy}\boldsymbol{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\boldsymbol{Y}}), \boldsymbol{\Sigma}_X - \boldsymbol{\Sigma}_{Xy}\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_{Xy}^T)$$

Let $\mathbf{X}$ = EduExp, PTR, Salary, SAT.verbal, SAT.math and $\mathbf{y}$ = SAT.prct.

```
s_covar<-cov(X)
X_covar<-s_covar[1:5,1:5]
y_covar<-s_covar[6,6] # Variance of SAT.prct
Xy_covar<-s_covar[1:5,6] # Covariance of SAT.prct with X Variables
```

Now we can calculate the conditional variance.

```
cond_covar<-X_covar - Xy_covar %*% solve(y_covar) %*% t(Xy_covar)
```

Therefore will the correlation be:

```
# standard deviation
sd_covar<-diag(1/diag(cond_covar)^(1/2))
cond_cor<-(sd_covar) %*% cond_covar %*% (sd_covar)
```

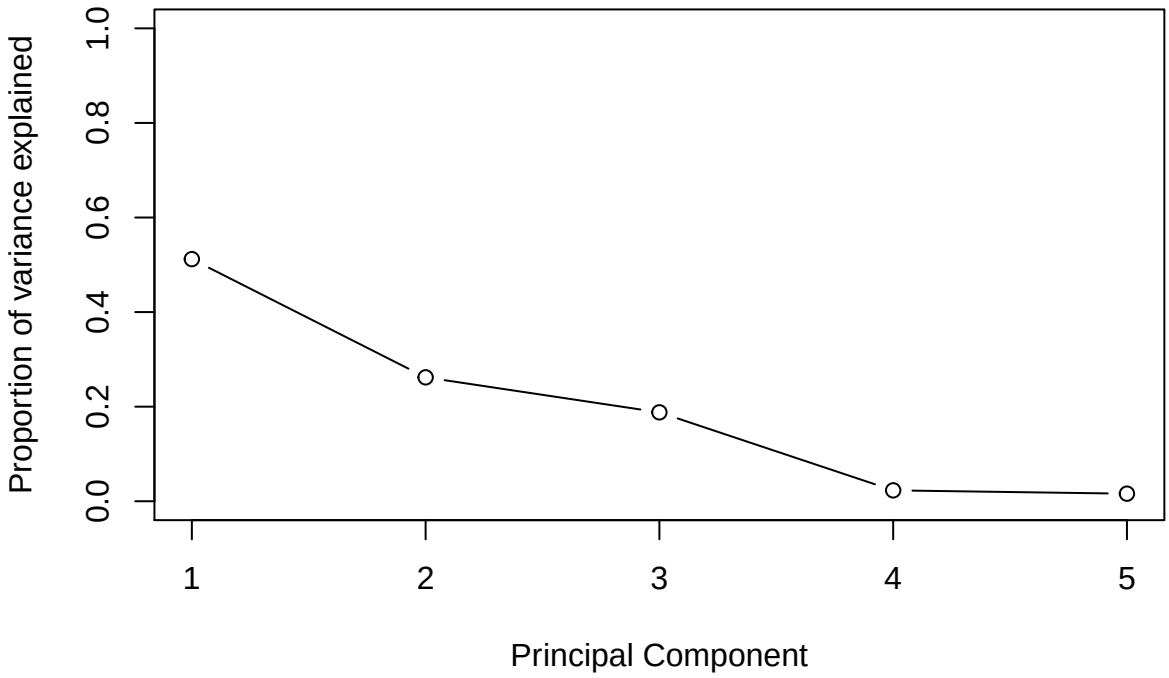
Therefore, the correlation will be:

```
cond_cor
```

```
##            [,1]       [,2]      [,3]       [,4]       [,5]
## [1,]  1.0000000 -0.3110154 0.7953712  0.3295378  0.4165828
## [2,] -0.3110154  1.0000000 0.1693763 -0.2881202 -0.1860090
## [3,]  0.7953712  0.1693763 1.0000000  0.2090832  0.3468431
## [4,]  0.3295378 -0.2881202 0.2090832  1.0000000  0.8718591
## [5,]  0.4165828 -0.1860090 0.3468431  0.8718591  1.0000000
```

(b)

```
sat.con.eigen<-eigen(cond_cor)
sat.con_var<-round(sat.con.eigen$values/sum(sat.con.eigen$values), digits=3)
plot(sat.con_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```



(c)

We can retrieve the PC loadings from the variable we just performed PCA on as follows:

```
# Using eigendecomposition, just retrieving the vectors. (5 x 5 matrix)
B.hat<-sat.con.eigen$vectors
```

```
# Using the scale function to scale the first 5 columns of our data matrix
Z.hat<-scale(X[,1:5])
```

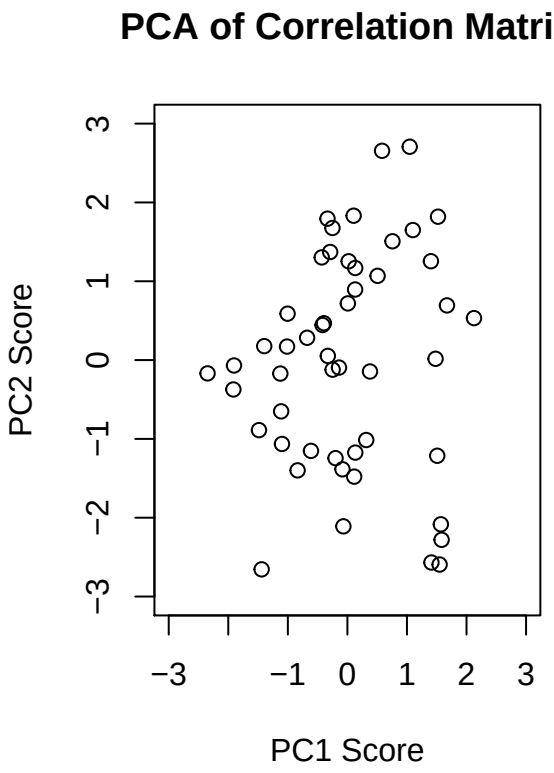
Now, the pseudo PC scores will be:

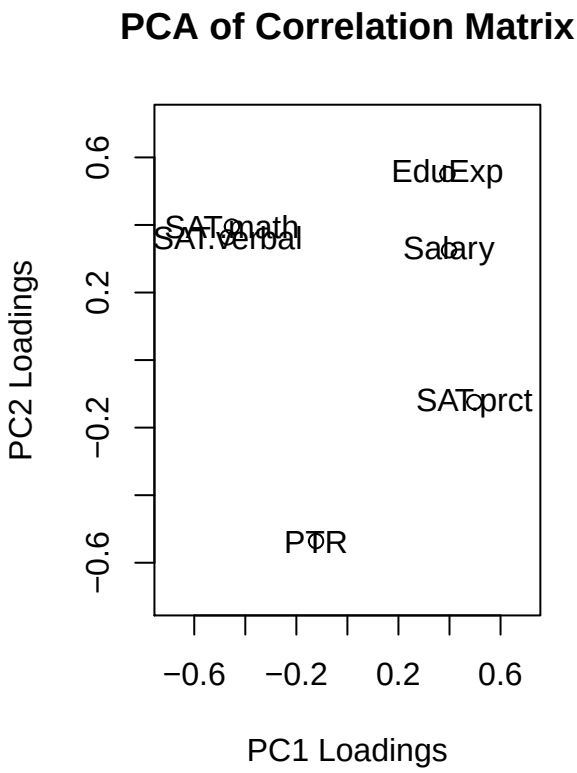
```
Y.hat<-Z.hat %*% B.hat
```

We can plot both the PC loadings and PC scores using the `plot` function in *R* as follows:

```
par(mfrow=c(1,2))
plot(Y.hat[,1:2],
     xlab = "PC1 Score",
     ylab = "PC2 Score",
     main = "PCA of Correlation Matrix",
     xlim = c(-3, 3),
     ylim = c(-3, 3))
plot(sat.eigen$vectors[,1:2],
     xlab = "PC1 Loadings",
     ylab = "PC2 Loadings",
     main = "PCA of Correlation Matrix",
```

```
     xlim = c(-0.7, 0.7),
     ylim = c(-0.7, 0.7))
text(sat.eigen$vectors[,1:2], labels = colnames(X))
```





(d)

The first PC accounts for more variation in `SAT.total` using the pseudo PC scores. This result differs to what we calculated previously as, using pesudo pc scores we are able to explain more of the variation for one of our variables.

Quesiton 3

Here are the two images that I will be working with.

(a)

First let me import both the pictures:

```
pic_A<-readJPEG('pic_A.jpg')
pic_B<-readJPEG('pic_B.jpg')
```

Separating the red, green, and blue components into vectors

```
r <- pic_A[,,1]
g <- pic_A[,,2]
b <- pic_A[,,3]
```

We will be performing the PCA on the covariance matrix

```
A.r_eigen<-eigen(cov(r))
A.g_eigen<-eigen(cov(g))
A.b_eigen<-eigen(cov(b))
```

Calculating it simply using eigendecomposition

```
A.r_PCA<-as.matrix(r) %*% A.r_eigen$vectors
A.g_PCA<-as.matrix(g) %*% A.g_eigen$vectors
A.b_PCA<-as.matrix(b) %*% A.b_eigen$vectors
```

We can also collect it in a list:

```
A_PCA <- list(A.r_PCA, A.g_PCA, A.b_PCA)
```

(b)

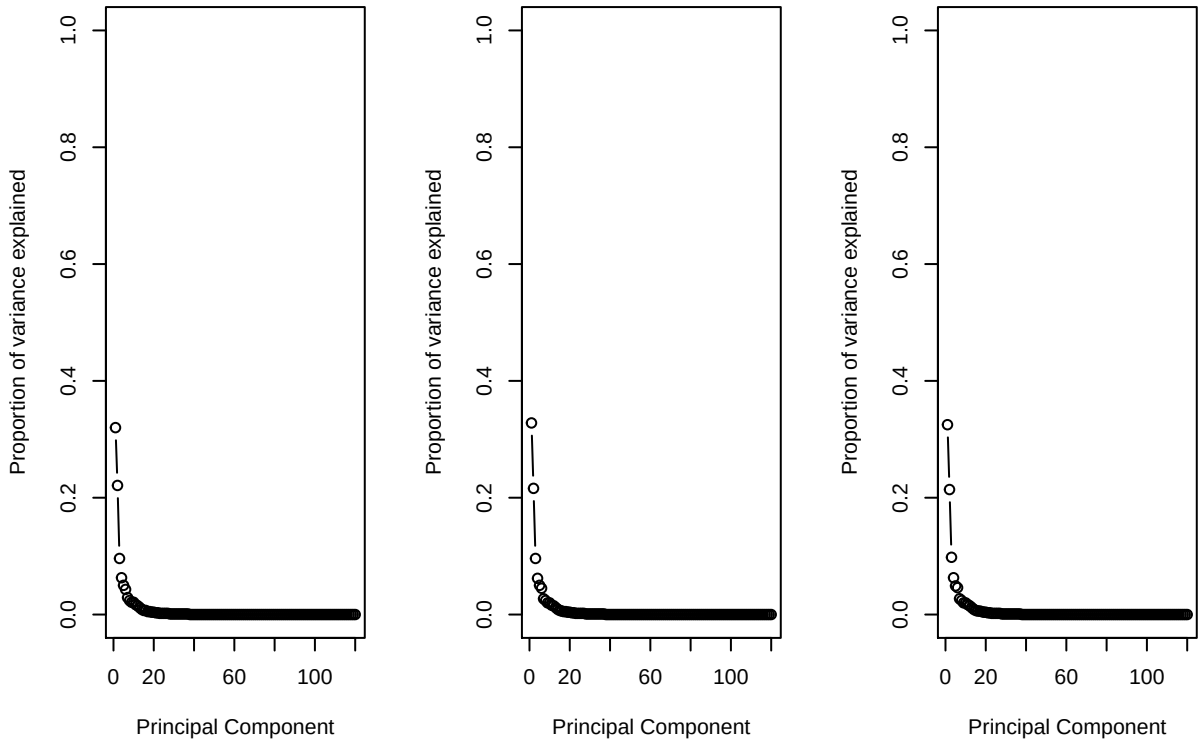
To calculate the scree plots we need to know the proportion of variance explained by each Principal Component for each of our color schemes:

```
A.r_var<-round(A.r_eigen$values/sum(A.r_eigen$values), digits=3)
A.g_var<-round(A.g_eigen$values/sum(A.g_eigen$values), digits=3)
A.b_var<-round(A.b_eigen$values/sum(A.b_eigen$values), digits=3)
```

Now, we can plot them simply

```
par(mfrow=c(1,3))
plot(A.r_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```

```
plot(A.g_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
plot(A.b_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```



Looks like around 45 PC's should be enough to reconstruct the image.

(c)

```
for(i in c(45,60,80,110, 120)){
  r.A.recon<- r %*% A.r_eigen$vectors[,1:i] %*% t(A.r_eigen$vectors[,1:i])
  g.A.recon<- g %*% A.g_eigen$vectors[,1:i] %*% t(A.g_eigen$vectors[,1:i])
  b.A.recon<- b %*% A.b_eigen$vectors[,1:i] %*% t(A.b_eigen$vectors[,1:i])


  pic_A.recon <-array(c(r.A.recon, g.A.recon, b.A.recon), dim = c(120,120,3))
writeJPEG(pic_A.recon,paste("pic_A.compressed",round(i,0),"_components.jpg",sep=""))
}
```

Here are the reconstructed images, ordered in ascending order of PC's

(d)

First, I will perform the eigendecomposition on the covariance of the three color schemes for Picture B

```
B.r_eigen<-eigen(cov(pic_B[,,1]))
B.g_eigen<-eigen(cov(pic_B[,,2]))
B.b_eigen<-eigen(cov(pic_B[,,3]))
```

I am going to use the first 50 principal component loadings to reconstruct my second image. We shall refer to these as `Vsmall`

```
V_small.r<-A.r_eigen$vectors[,1:100]
V_small.g<-A.g_eigen$vectors[,1:100]
V_small.b<-A.b_eigen$vectors[,1:100]
```

```
# Reconstructed
B.r<-pic_B[,,1] %*%  V_small.r %*% t(V_small.r)
B.g<-pic_B[,,2] %*%  V_small.g %*% t(V_small.g)
B.b<-pic_B[,,3] %*%  V_small.b %*% t(V_small.b)
```

```
# Make it a matrix
B.recon<-array(c(B.r, B.g, B.b), dim=c(120,120,3))
```

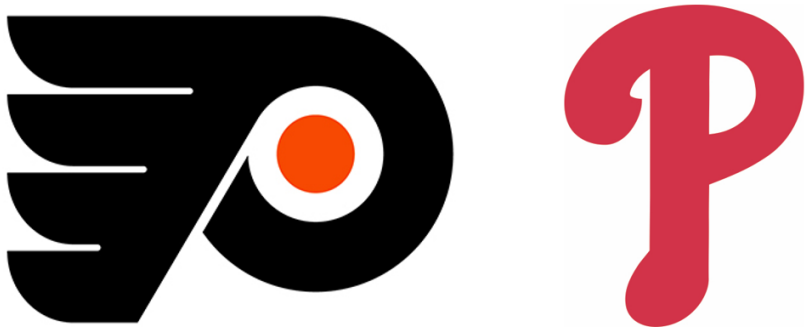
Now, let us reconstruct this image

```
writeJPEG(B.recon,paste("pic_B.",round(100,0),"_components.jpg",sep=""))
```

Here is picture B, reconstructed in order of PC's

(e)

Here are my original images:



```
pic_C<-readJPEG('flyers.jpg')
pic_D<-readJPEG('phillies.jpg')
```

Separating the red, green, and blue components into vectors

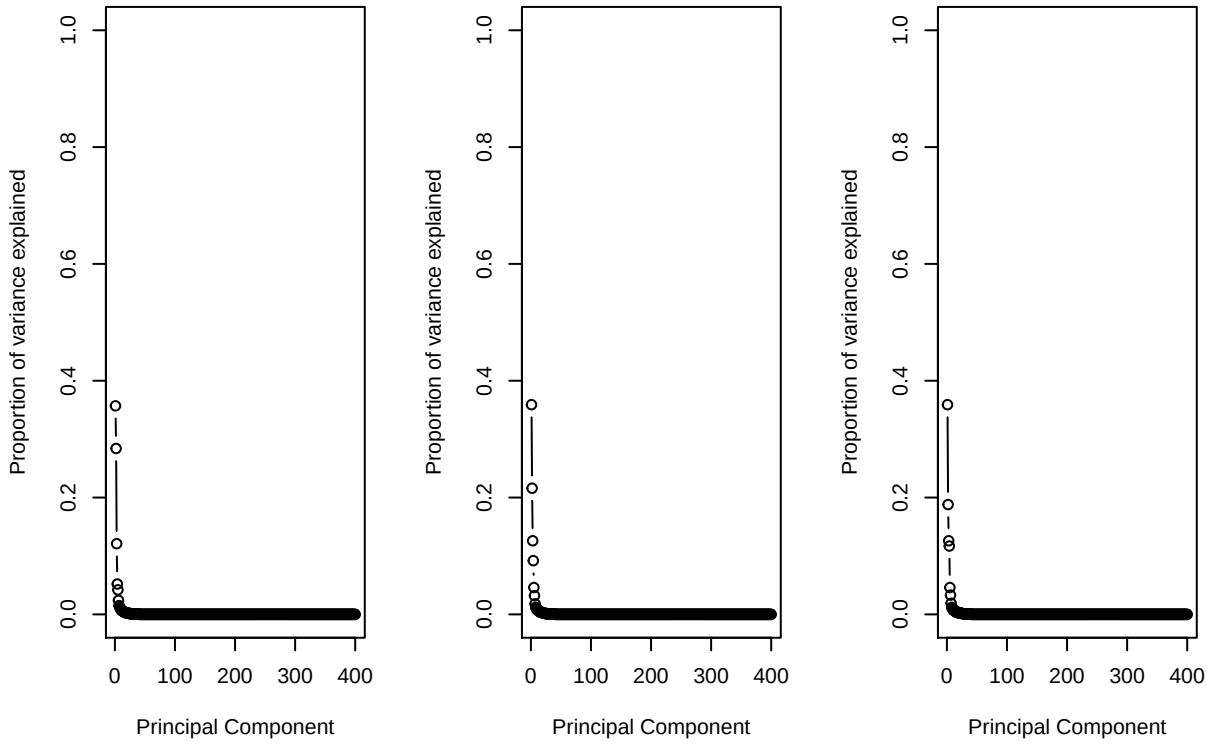
```
r_C <- pic_C[,,1]
g_C <- pic_C[,,2]
b_C <- pic_C[,,3]

# Perform PCA on each color scheme
C.r_eigen<-eigen(cov(r_C))
C.g_eigen<-eigen(cov(g_C))
C.b_eigen<-eigen(cov(b_C))

C.r_PCA<-as.matrix(r_C) %*% C.r_eigen$vectors
C.g_PCA<-as.matrix(g_C) %*% C.g_eigen$vectors
C.b_PCA<-as.matrix(b_C) %*% C.b_eigen$vectors


C_PCA<-list(C.r_PCA, C.g_PCA, C.b_PCA)
```

Calculating the proportion of variance explained:

```
C.r_var<-round(C.r_eigen$values/sum(C.r_eigen$values), digits=3)
C.g_var<-round(C.g_eigen$values/sum(C.g_eigen$values), digits=3)
C.b_var<-round(C.b_eigen$values/sum(C.b_eigen$values), digits=3)
```

The scree plots:

```
par(mfrow=c(1,3))
plot(C.r_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
plot(C.g_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
plot(C.b_var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```




```
for(i in c(20,40,80,150,200, 300)){
  r.C.recon<- r_C %*% C.r_eigen$vectors[,1:i] %*% t(C.r_eigen$vectors[,1:i])
  g.C.recon<- g_C %*% C.g_eigen$vectors[,1:i] %*% t(C.g_eigen$vectors[,1:i])
  b.C.recon<- b_C %*% C.b_eigen$vectors[,1:i] %*% t(C.b_eigen$vectors[,1:i])


  pic_C.recon <-array(c(r.C.recon, g.C.recon, b.C.recon), dim = c(278,400,3))
  writeJPEG(pic_C.recon,paste("pic_C.compressed",round(i,0),"_components.jpg",sep=""))
}
```

Here are the reconstructed images:

First, let me peform the eigendecomposition on picture D:

```
D.r_eigen<-eigen(cov(pic_D[,,1]))
D.g_eigen<-eigen(cov(pic_D[,,2]))
D.b_eigen<-eigen(cov(pic_D[,,3]))
```

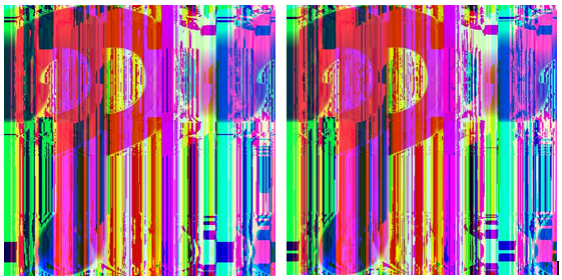
```
# First 80 eigenvectors for Picture C
W_small.r<-C.r_eigen$vectors[,1:100]
W_small.g<-C.g_eigen$vectors[,1:100]
W_small.b<-C.b_eigen$vectors[,1:100]
```

```
# Reconstructed
D.r<-pic_D[,,1][,1:400] %*%  W_small.r %*% t(W_small.r)
D.g<-pic_D[,,2][,1:400] %*%  W_small.g %*% t(W_small.g)
D.b<-pic_D[,,3][,1:400] %*%  W_small.b %*% t(W_small.b)
```

```
# Make it a matrix
D.recon<-array(c(D.r, D.g, D.b), dim=c(500,500,3))
```

Now, let us reconstruct this image

```
writeJPEG(D.recon,paste("pic_D.",round(100,0),"_components.jpg",sep=""))
```



I have found that surprisingly some of the image is still recognizable.

Quesiton 4

(a)

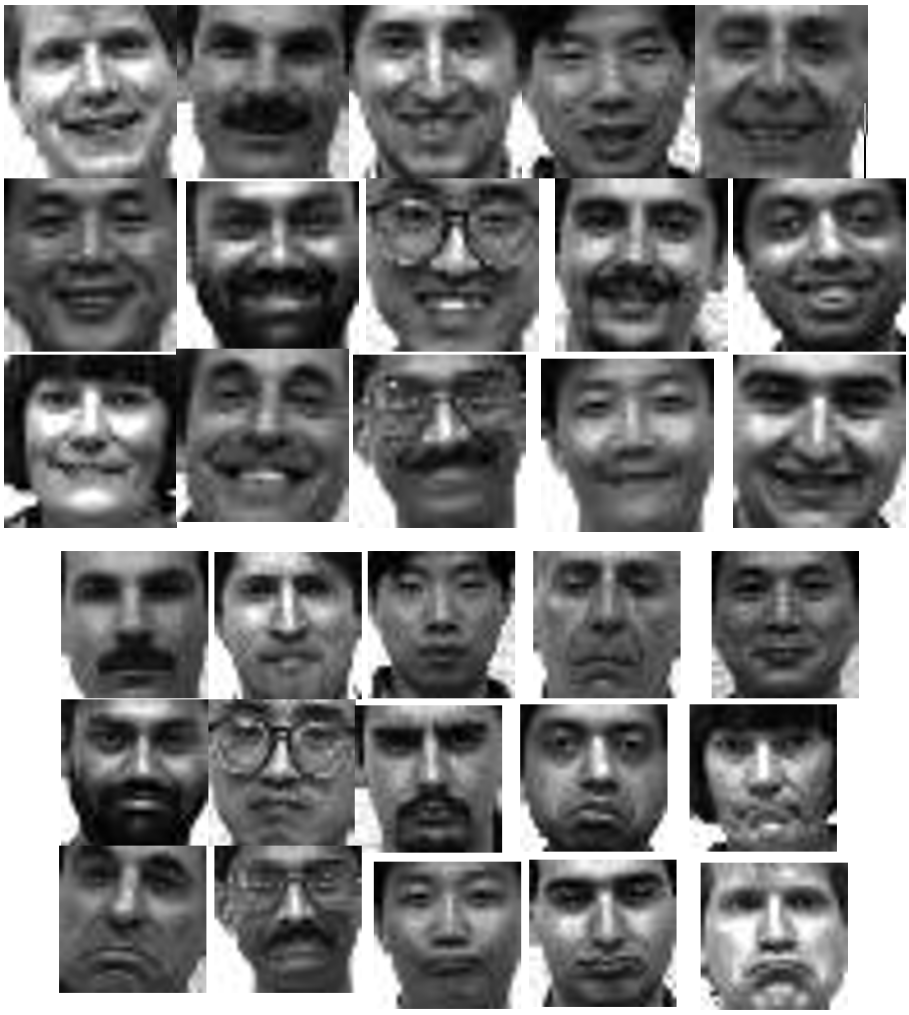
Let us first subset our data

```
facedata<-unlist(data)
facedata<-as.numeric(facedata)
facedata<-matrix(facedata, nrow=165, ncol=1026)
```

```
sad<-subset(data, data$`facial expression` == 'sad')
sad<-unlist(sad)
sad<-as.numeric(sad)
sad<-matrix(sad, nrow= 15, ncol = 1026)


# Now for happy
happy<-subset(data, data$`facial expression` == 'happy')
happy<-unlist(happy)
happy<-as.numeric(happy)
happy<-matrix(happy, nrow = 15, ncol=1026)
```

```
# Constructing the happy and sad images
for(i in c(1:15)){
  sad_img<-matrix(sad[i, 3:1026], nrow=32,ncol=32)
  writeJPEG(sad_img, paste("person" ,round(i,0),"sad.jpg"))
  happy_img<-matrix(happy[i,3:1026], nrow=32, ncol=32)
  writeJPEG(happy_img, paste("person", round(i,0), "happy.jpg"))
}
```

Here are the happy and sad faces.



(b)

Let us do the following:

```
# For sad
mean_sad<-colMeans(sad[,3:1026])
mean_sad<-matrix(mean_sad, nrow=32, ncol=32)


# For happy
mean_happy<-colMeans(happy[,3:1026])
mean_happy<-matrix(mean_happy, nrow=32, ncol=32)


# For all faces
mean_face<-colMeans(facedata[,3:1026])
mean_face<-matrix(mean_face, nrow=32, ncol=32)
```

Now, we can print out the mean faces

```
writeJPEG(mean_sad, paste("avg_sad.jpg"))
writeJPEG(mean_happy, paste("avg_happy.jpg"))
writeJPEG(mean_face, paste("avg_face.jpg"))
```

Yes, I can indeed tell the difference between the average faces.

(c)

We can perform the eigendecomposition on the training data set, but first we have to unlist it and convert it to numeric form

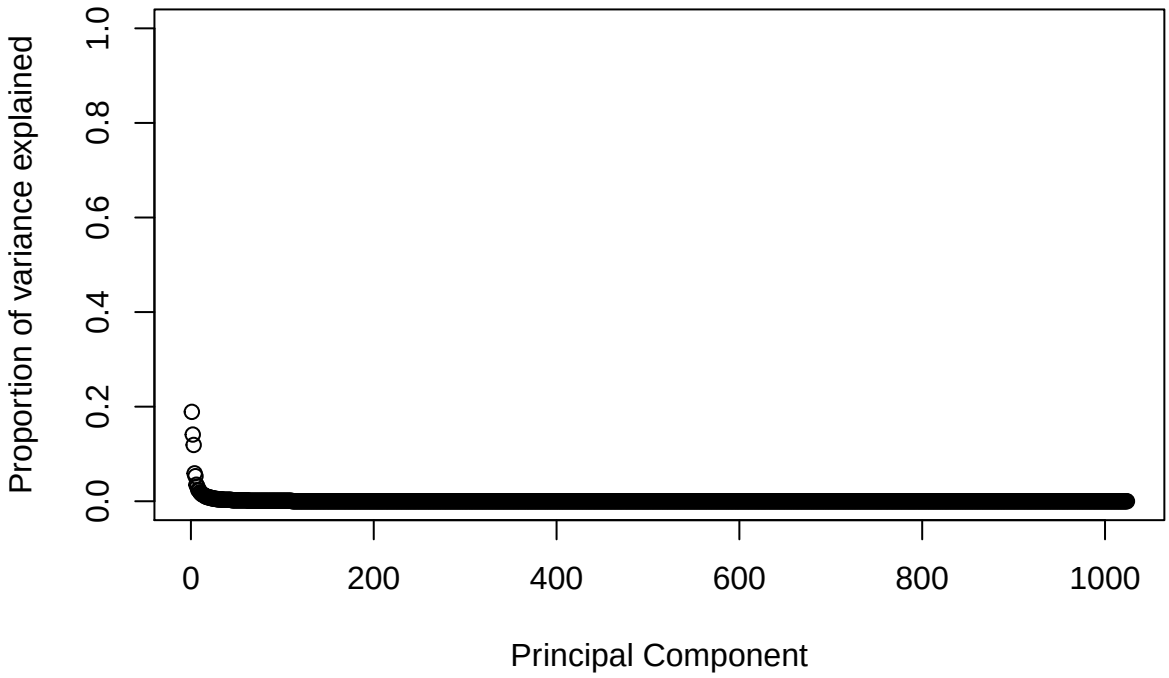
```
facetrain<-unlist(traindata)
facetrain<-as.numeric(facetrain)
facetrain<-matrix(facetrain, nrow=150, ncol=1026)
```

Now, we can perform the eigendecomposition

```
# First two columns are just subject ID and expression
facetrain_eigen<-eigen(cov(facetrain[,3:1026]))

# Also, we can perform PCA
facetrain_PCA<-facetrain[,3:1026] %*% facetrain_eigen$vectors
```

Now, we can perform a screeplot:

```
facetrain.var<-round(facetrain_eigen$values/sum(facetrain_eigen$values), digits=3)

# Plotting it out
plot(facetrain.var, xlab = "Principal Component", ylab = "Proportion of variance explained",
     ylim=c(0,1), type='b')
```



It seems like around 5 PC's will explain 50% variation in our data. Let us see how much of the variation the top 25 eigenvectors explain.

```
sum(facetrain.var[1:25])
```

```
## [1] 0.842
```

Looks the top 25 eigenvectors explain 84.2 % of the variation.

(d)

```
for(i in c(1:25)){
  min_u<-min(facetrain_eigen$vectors[,i])
  max_u<-max(facetrain_eigen$vectors[,i])
  eigen_face<-matrix((facetrain_eigen$vectors[,i] - min_u)/ (max_u - min_u),
                     nrow = 32,
                     ncol = 32)
  writeJPEG(eigen_face, paste("eigenface", round(i,0), ".jpg"))
}
```

Here are all the eigenfaces:

(e)

```
# 1 x 1024
A<-facetrain[1,3:1026] - colMeans(traindata[,3:1026])
load_train<- facetrain_eigen$vectors[,1:200]
weights<-as.numeric(t(load_train) %*% A)
```

```
for(i in c(10,20,40,80,100,200)){
  running_total<-matrix(0, nrow = 1024, ncol=1)
  uw_k <-facetrain_eigen$vectors[,1:i] %*% weights[1:i]
  running_total <- running_total + uw_k
  recon_img<-matrix((colMeans(traindata[,3:1026]) + running_total) ,
                    nrow=32,
                    ncol=32)
  writeJPEG(recon_img, paste('recon_img', round(i,0), 'componenets.jpg'))
}
```

Here are the reconstructed images, ordered in terms of magnitude.

With 40 components, I can see a reasonable image.

(f)

Let us load in the testing data set

```
facetest<-unlist(testdata)
facetest<-as.numeric(facetest)
facetest<-matrix(facetest, nrow=15, ncol=1026)
```

```
# Load the first image
test_1<-facetest[1, 3:1026]

# Subtracting the mean of the training image
test_1<-test_1 - colMeans(traindata[,3:1026])
```

```
recon_test<-matrix(test_1, nrow=32,ncol=32) %*% matrix(facetrain_eigen$vectors[,1:25], nrow=32, ncol=32)

writeJPEG(recon_test, paste("reconstruct_test_img.jpg"))
```

Bonus Problem

(a)

Using matrix notation, the above model can be written as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$Var[\boldsymbol{\epsilon}] = \sigma^2\mathbf{X^T X}$

(b)

Let us generate the data

```
n = 200
x = rnorm(n)

# Only 1 variable, therefore no variance-covariance matrix.
epsilon = mvrnorm(n, mu = mean(x), Sigma =  var(x))
beta0 = 1
beta1 = 1
y = beta0 + beta1*x + epsilon
```

(c)

(d)

(e)