ST-6034
Multivariate Methods for Data Analysis

Assignment 1

Aditya Vikram Dash
120224162

Question 1:

(a) Given a real symmetric matrix M, i.e.

$$M = M^T$$

If M were to have complex eigenvalues, then we can write,

$$Mx = Mx$$
$$M\bar{x} = \bar{M}x$$

Under complex conjugation,

$$\bar{x}^T Mx = \bar{x}^T \lambda x = \lambda ||x||^2 \quad ..(i)$$
$$x^T M\bar{x} = x^T \bar{\lambda} x = \bar{\lambda} ||x||^2 \quad ..(ii)$$

Since M is symmetric,

$$\bar{x}^T Mx = (Mx)^T \bar{x} = x^T A^T x = x^T A\bar{x}$$

Subtracting (ii) from (i), we get,

$$(\bar{\lambda} - \lambda)||x||^2 = 0$$

Only way this is possible for a non-zero z is if,

$$\lambda = \bar{\lambda}$$

Therefore, λ is real.

(b)

```
library(Matrix)
p=111

#Matrix of 'p' dimension with random normal entries
M = matrix(rnorm(p*p),ncol=p)
s.matrix=as.matrix(forceSymmetric(m))

#Calculating Eigen Values and Eigen Vectors of the Symmetric
Matrix
eg = eigen(s.matrix)
lambda = diag(eg$Values)
eg_vec = eg$vectors

#Inverse of matrix of eigen vectors
eg_vec_inverse = round(solve(eg_vec),4)

#Transpose of matrix of eigen vectors
eg_vec_transpose = round(t(eg_vec),4)
```

(c) **M** is symmetric. Therefore the minimum of $x^T M x$ with $\|x\|=1$ is the minimum eigenvalue of **M,** which is equal to −1 because **M** is a reflection.

Question 4:

(a)

```
library(av)
library(magick)
```

(b)

```
av_video_images("~/Desktop/waveclip.mp4", destdir =
"~/Desktop/WaveClip Images", format = "jpg", fps = NULL)

image_read("~/Desktop/WaveClip Images/")


files <- list.files("~/Desktop/WaveClip Images/", pattern =
"*.jpg", full.names=TRUE)
all_im <- lapply(files, load.image("~/Desktop/WaveClip
Images/")

########################################################

library(imagerExtra)
library(magick)
library(readbitmap)
library(here)
library(dplyr)
library(terra)
library(jpeg)

setwd("~/Desktop/WaveClip Images/")

image.files <- dir("~/Desktop/WaveClip Images/", pattern =
"*.jpg", full.names = TRUE)

output<-vector("list", length(image.files))

############################################################

#Get the list of all images
allFiles = list.files(path = "~/Desktop/WaveClip Images/",
pattern = ".jpg", full.names = T)

#Get the dataframe with info
allInfo = image_info(image_read(allFiles))

#Attach the file names
allInfo$fileName = list.files(path = "~/Desktop/WaveClip
Images/", pattern = ".jpg")
allInfo$fileName

#Save
write.csv(allInfo, "allImageInfo.csv")

View(allInfo)
```

Question 3:

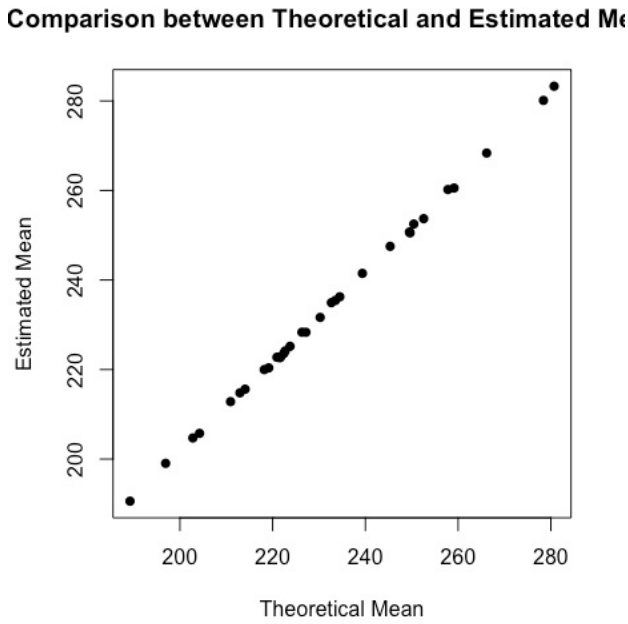
(a) $X_i \sim N_{60}(\mu, \Sigma)$ for i = 1,2,..,900

$\Longrightarrow E[X_i] = \mu$ and $cov(X_i, X_i') = \Sigma$

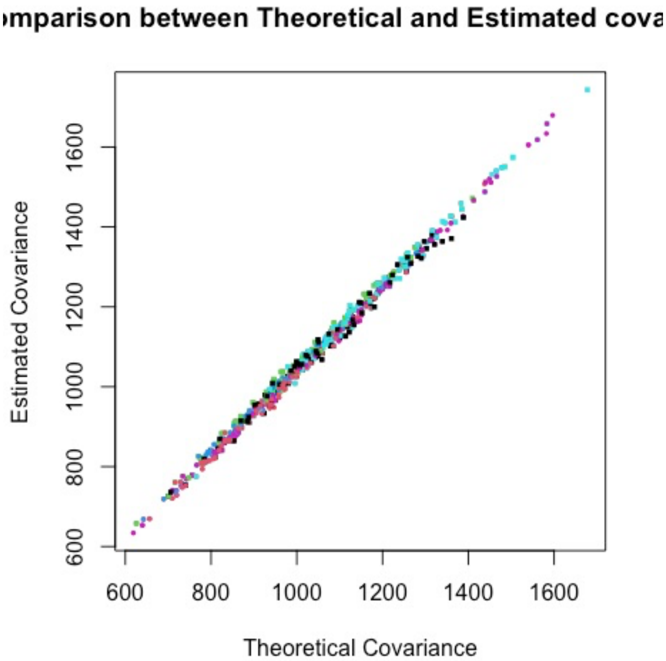
Let $Y_1 = AX_1$ where A is of dimension (32 x 60)
Then,

$E[Y_1] = E[AX_1] = A.E[X_1] = A\,\mu$ (dimension 32 x 1)

$cov[Y_1, Y_1'] = cov[AX_1, (AX_1)'] = cov[AX_1, X'_1A'] = A\Sigma A'$



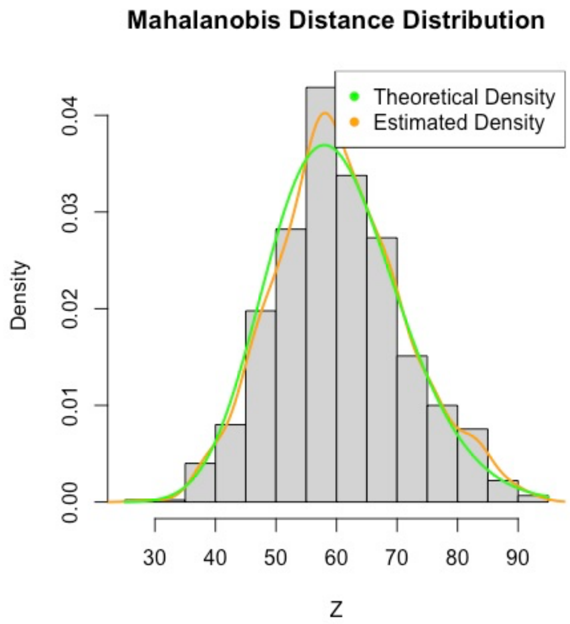
In the figure above "Comparison between Theoretical and Estimated Means", we can observe it is a perfect linear relationship. This is due to the usage of eigenvectors which maximize the variance.

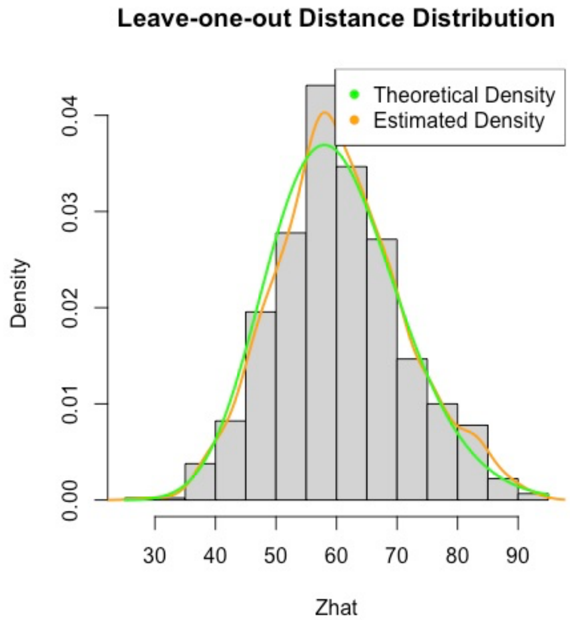


In the figure above "Comparison between Theoretical and Estimated Covariance" we can observe a positive correlation between the Estimated and Theoretical Covariance values.

```
library(MASS)
load("~/Desktop/HWK2.RData")

N <- 900
P <- 60

V_mu <- as.matrix(data$V_mu)
M_Sigma <- as.matrix(data$M_Sigma)
B <- data$B

X <- mvrnorm(n = N, V_mu = data$V_mu, M_Sigma = data$M_Sigma)
Xbar <- colMeans(X)
Shat <- cov(X)

Y <- NULL
for (i in 1:N) {
  Y <- cbind(Y, B %*% X[i,])
}
Ybar <- rowMeans(Y)
V_mu_Y <- B %*% V_mu
M_SigmaY <- B %*% M_Sigma %*% t(B)
Shat_Y <- B %*% Shat %*% t(B)

plot(x = V_mu_Y, y = Ybar, type = 'p', main = "Comparison between
Theoretical and Estimated Means", xlab = 'Theoretical Mean', ylab =
'Estimated Mean', cex = 1, pch = 16)
matplot(x = M_SigmaY, y = Shat_Y, type = 'p', main = "Comparison
between Theoretical and Estimated covariance", xlab = 'Theoretical
Covariance', ylab = 'Estimated Covariance', pch = c(15,16), cex = 0.5)
```

(b)






```
      Z <- numeric(N)
      Z_hat <- numeric(N)
      for (i in 1:N) {
        Z[i] <- t(X[i,] - V_mu) %*% solve(M_Sigma) %*% (X[i,] -
V_mu)
        Z_hat[i] <- t(X[i,-i] - V_mu[-i]) %*% solve(M_Sigma)[-i,-i]
%*% (X[i,-i] - V_mu[-i])
      }

      hist(Z, prob = TRUE, main = 'Mahalanobis Distance
Distribution')
      lines(density(Z), col = 'orange', lwd = 2)
      x <- rchisq(N, df = P)
      curve(dchisq(x, df = P), col = 'green', add = TRUE, lwd = 2)
```

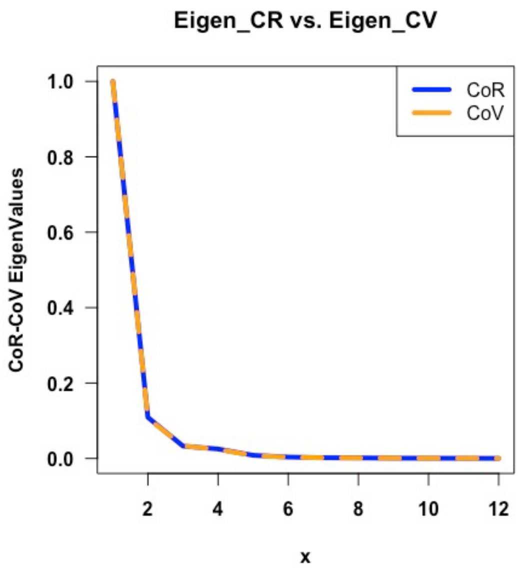
```
      legend(x = 'topright', legend = c('Theoretical Density',
'Estimated Density'), col = c('green', 'orange'), pch = 16)

      hist(Z_hat, prob = TRUE, main = 'Leave-one-out Distance
Distribution')
      lines(density(Z_hat), col = 'orange', lwd = 2)
      curve(dchisq(x, df = P), col = 'green', add = TRUE, lwd = 2)
      legend(x = 'topright', legend = c('Theoretical Density',
'Estimated Density'), col = c('green', 'orange'), pch = 16)
```

Question 2:

```
#Correlation
cr_function=cor(dat)
D = scale(dat,center=T,scale=T)
n = nrow(dat)
CoR = cr_form = (1/n)*(t(D)%*%D)*(n/(n-1))
diag(cr_form)
diag(cr_function)

#Covariance by Function
cv_function = var(dat)
X = as.matrix(dat); x_bar = as.matrix(apply(dat,MARGIN=2,mean));
n = nrow(dat)
# covariance by matrix-formula
CoV = cv_form=((1/n)*(t(X)%*%X) -
t(rbind(apply(X,2,mean)))%*%rbind(apply(X,2,mean)))*(n/(n-1))
diag(cv_function)
diag(cv_form)
```

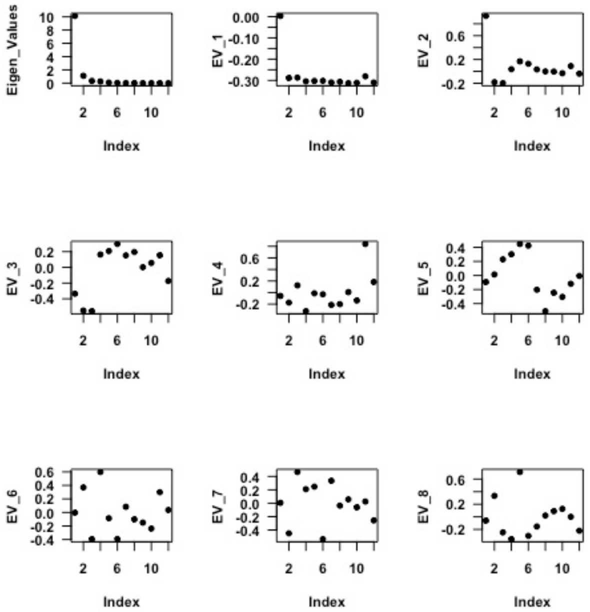
(a)

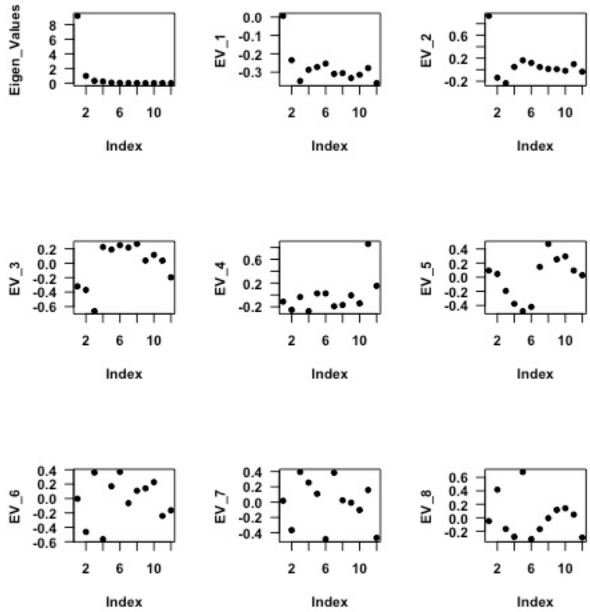
```
#Eigen Decompostion of matrices
e_cr = eigen(CoR); ecv = eigen(CoV)
e_val.cr = e_cr$values; e_vec.cr = e_cr$vectors
e_val.cv = ecv$values; evec.cv = ecv$vectors

e_val.cr.norm = (e_val.cr)/max(e_val.cr)
e_val.cv.norm = (e_val.cv)/max(e_val.cv)
?scale
x=c(1:12)
par(mfrow=c(1,1))
par(font.axis=2,font.lab=2,bg="white",font.main=2,las=1)


#Matplot
matplot(x,cbind(e_val.cr.norm,e_val.cv.norm),
        col=c("blue","orange"),type="l",lwd=4,
        main="Eigen_CR vs. Eigen_CV",
        ylab="CoR-CoV EigenValues")
legend("topright",legend=c("CoR","CoV"),
       col=c("blue","orange"),lty=1,lwd=4)
```



(b)

```
names=c("EV_1","EV_2","EV_3","EV_4","EV_5","EV_6","EV_7","EV
_8")
par(mfrow=c(3,3))
plot(e_val.cr,pch=20,cex=1.2,ylab="Eigen_Values")
for(j in 1:8){
  plot(e_vec.cr[,j],pch=20,cex=1.2,ylab=names[j])
}

par(mfrow=c(3,3))
plot(e_val.cv,pch=20,cex=1.2,ylab="Eigen_Values")
for(j in 1:8){
  plot(evec.cv[,j],pch=20,cex=1.2,ylab=names[j])
}
```





(c)

```
dat = USJudgeRatings
D = scale(dat,scale=T,center=T)
n = nrow(dat)
cor.mat = (1/n)*(t(D)%*%D)*(n/(n-1))
dim(cor.mat) # sq. matrix

#Spectral Decomposition of Correlation Matrix
e = eigen(cor.mat)
G = e$vectors
D = e$values
D = diag(D)
e = eigen(cor.mat)
R=(G%*%D%*%t(G))

#Design Matrix
des_mat = model.matrix(~G[,1:4])[,-1]
sds = apply(dat,MARGIN=2,sd)
means = apply(dat,MARGIN=2,mean)
y1 = numeric(12)
for(j in 1:12)
{
  y1[j] = (dat[,j]-means[j])/sds[j]
}
colnames(des_mat) = c("pred_1","pred_2","pred_3","pred_4")
l_mo_1 = lm(y1~.,data = data.frame(des_mat))
b1 = coef(l_mo_1)[-1]

des_mat2 = model.matrix(~t(G)[,1:4])[,-1]
colnames(des_mat2) = c("pred_1","pred_2","pred_3","pred_4")
y2=numeric(12)
for(j in 1:12)
{
  y2[j] = (dat[,j]-means[j])/sds[j]
}
l_mo_2 = lm(y2~., data=data.frame(des_mat2))
b2 = coef(l_mo_2)[-1]

par(mfrow=c(1,1))
x=seq(1:4)
plot(x=x,y=b1,col="orange",cex=2,pch=20,ylab="Coefficients",
main="Coefficients Comparison")
points(b2,col="green",pch=20,cex=2)
points(y2,col="blue",pch=20,cex=2)
legend("bottomright",legend=c("G-coefs","G'-coefs","G'-
yi's"),col=c("orange","green","blue")
       ,pch=20)
```