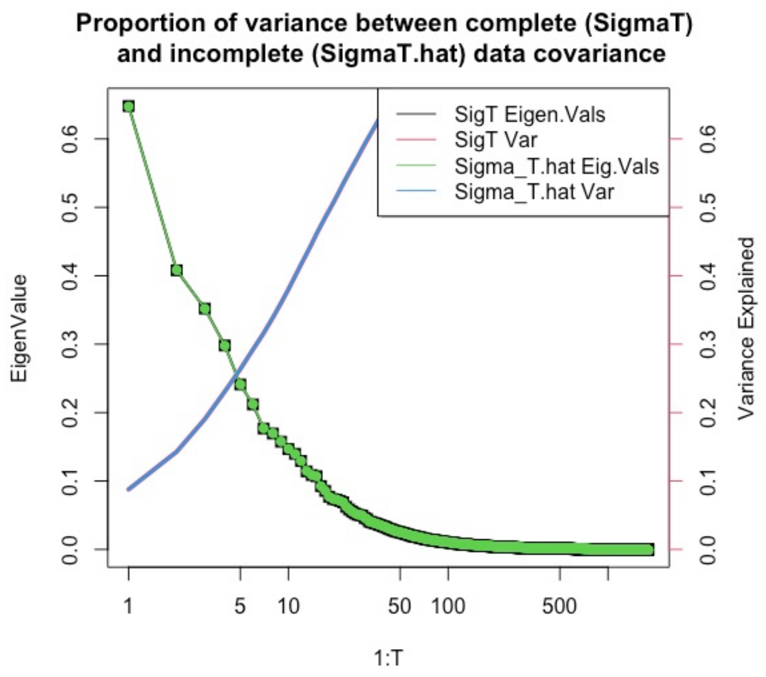
ST6034 - Multivariate Methods for Data Analysis
Assignment 2

Aditya Vikram Dash
MSc Data Science and Analytics
120224162

Question 1

```
rm(list = ls())
if(!is.null(dev.list())) dev.off()

uccstudentno <- 120224162
set.seed(uccstudentno + 2000)

load("~/Desktop/UCC 2021/Courses/ST6034 MVA/Waves.Rdata")

r <- data$r
g <- data$g
b <- data$b

nf <- 1790

##Generating Z-data Matrix
sigma_C <- cov(cbind(r, g, b))
a <- eigen(sigma_C)
b <- dim(r)[3]/2
nt <- dim(r)[3]
mn <- apply(cbind(r, g, b), 2, mean)
V <- a$vectors

for(b in 1:3) {
  sgnb <- sign( V[order( abs(V[,b]) )[3], b] )
  V[,b] <- sgnb*V[,b]
}

x_1 <- cbind(a$values/sum(a$values),V)
rownames(x_1) <- as.character(1:3)
colnames(x_1) <- c("Prop. Var-Explained", "Loading-Vector-1", "Loading-
Vector-2", "Loading-Vector-3")
tM = nf/2

pc <- r*V[1,1] + g*V[2,1] + b*V[3,1]
pc <- matrix(pc, ncol = nf)

##Direct Evaluation of the Linear Model
tx1 <- c(1:nf)
u <- (tx1 - mean(tx1)) / (sqrt(nf) * sd(tx1))
X <- cbind(rep(1, nf) / sqrt(nf), u)
Z <- pc - (pc %*% X ) %*% t(X)  # this is  y - X (X'X)^{-1} X'y

##Setting Dimensions of Z-data Matrix (N x t)
N <- nrow(Z)
t <- ncol(Z)

##Simulating Bernoulli Sequence
p <- 0.7
bin_sq <- rbinom(n = N*t, size = 1, prob = p)
U <- matrix(data = bin_sq, nrow = N, ncol = t)   # put values into matrix
with the same dimensions as Z

##Elements having U = 0 in Z are replaced with NA
ZNA_mrx <- Z
ZNA_mrx[U == 0] <- NA

##Number and Proportion of cases that are at least 10% and 25% complete
count_NA <- rowSums(apply(is.na(ZNA_mrx), 2, as.numeric))
```

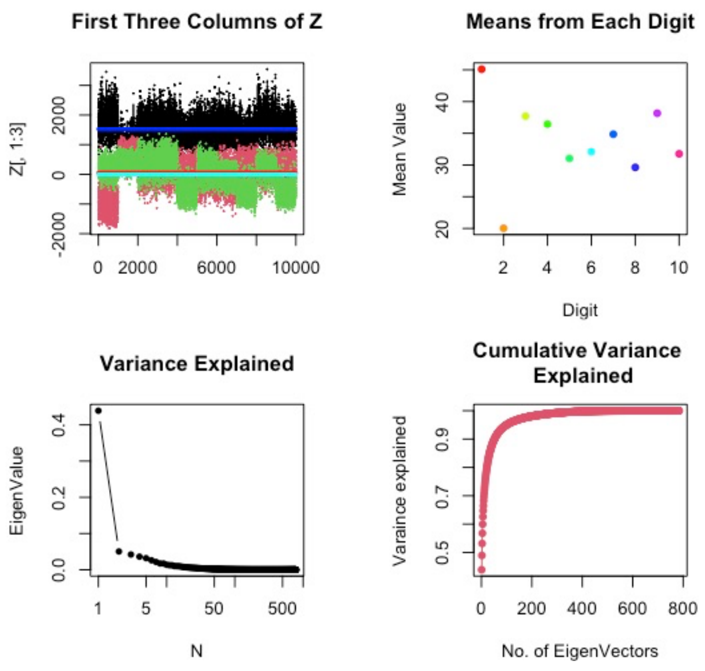
```
prop_NA <- count_NA / ncol(ZNA_mrx)  # proportion of NA's present in Z ==>
proportion not complete

num.complete.10 <- length(which(prop_NA < 1 - 0.1))
num.complete.25 <- length(which(prop_NA < 1 - 0.25))

prop.complete.10 <- num.complete.10 / nrow(ZNA_mrx)
prop.complete.25 <- num.complete.25 / nrow(ZNA_mrx)

##Pairwise Complete Observations(Covariance matrix for ZNA_mrx)
Sigma_T.hat <- cov(ZNA_mrx, use = 'pairwise.complete.obs')

##Plot - Proportion of variance between complete (SigmaT)
##and incomplete (SigmaT.hat) data covariance

eigen.Sigma_T <- eigen(cov(Z))
eig.Sigma_T.hat <- eigen(Sigma_T.hat)

length(eigen.Sigma_T$values)

par(mfrow = c(1,1), mar = rep(4,4))

##Complete
plot(1:length(eigen.Sigma_T$values), eigen.Sigma_T$values, pch = 15, col =
1, cex = 1.25,
     xlab = "1:T", ylab = "EigenValue", log = "x", main = "Proportion of
variance between complete (SigmaT) \n and incomplete (SigmaT.hat) data
covariance")
lines(1:length(eigen.Sigma_T$values), eigen.Sigma_T$values, col = 1, lwd =
2)
lines(1:length(eigen.Sigma_T$values),
cumsum(eigen.Sigma_T$values)/sum(eigen.Sigma_T$values), col = 2, pch = 16,
lwd = 3.5)
axis(4, col = 2)
mtext('Variance Explained', side = 4, line = 2.5)

##Incomplete
points(1:length(eig.Sigma_T.hat$values), eig.Sigma_T.hat$values, pch = 16,
col = 3, cex = 1)
lines(1:length(eig.Sigma_T.hat$values), eig.Sigma_T.hat$values, col = 3,
lwd = 1)
lines(1:length(eig.Sigma_T.hat$values),
cumsum(eig.Sigma_T.hat$values)/sum(eig.Sigma_T.hat$values), col = 4, pch =
16, lwd = 2)

legend('topright', legend = c('SigT Eigen.Vals', 'SigT Var', 'Sigma_T.hat
Eig.Vals', 'Sigma_T.hat Var'),
       col = c(1,2,3,4), lty = 1)
```

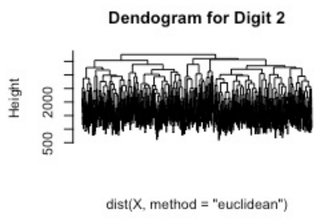

Proportion of variance between complete (SigmaT)
and incomplete (SigmaT.hat) data covariance
SigT Eigen.Vals
SigT Var
Sigma_T.hat Eig.Vals
Sigma_T.hat Var
EigenValue
Variance Explained
1:T
0.0
0.1
0.2
0.3
0.4
0.5
0.6
1
5
10
50
100
500

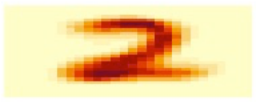
Question 2

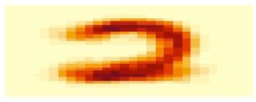
```
rm(list = ls())
if(!is.null(dev.list())) dev.off()

load("~/Desktop/UCC 2021/Courses/ST6034 MVA/Digits.Rdata")

###(a)
dim(digits)

a <- dim(digits)[1]
b <- dim(digits)[2]
c <- dim(digits)[3]
d <- dim(digits)[4]

means <- matrix(NA, nrow = 10)
for (i in 1:10) {
  means[i] <- mean(digits[, , , i])
}

##X <- matrix(data matrix)
X <- t(matrix(c(digits), ncol = 10000))
TSS <- t(X) %*% X     # Total Sums of Squares Matrix

##Spectral Decomposition of TSS
N_TSS <- nrow(TSS)
Eig_TSS <- eigen(TSS)
V_TSS <- Eig_TSS$vectors
D_TSS <- diag(Eig_TSS$values)

TSS.spec_decomp <- V_TSS %*% D_TSS %*% t(V_TSS)

##Transforming X to create Z
Z <- X %*% V_TSS

##(i)
##Plot - First 3 columns of Z
##Highlighting the 10 points corresponding to the means of the data from
each digit
par(mfrow = c(2,2), mar = rep(4,4))
matplot(Z[,1:3], type = 'bbp', pch = 1, cex = 0.1, col = c(1, 2, 3), main =
'First Three Columns of Z')
matlines(1:nrow(Z), rep(mean(Z[,1]), nrow(Z)), col = 'blue', lwd = 3)
matlines(1:nrow(Z), rep(mean(Z[,2]), nrow(Z)), col = 'red', lwd = 3)
matlines(1:nrow(Z), rep(mean(Z[,3]), nrow(Z)), col = 'cyan', lwd = 3)
plot(means, pch = 16, col = rainbow(10), main = 'Means from Each Digit',
xlab = 'Digit', ylab = 'Mean Value')

##(ii)
##Plot - Variance explained by each eigenvector as well as the cumulative
variance
x_1 <- Eig_TSS$values/sum(Eig_TSS$values)
matplot(x_1, type = 'bbp', pch = 16, cex = 0.85, xlab = "N", ylab =
"EigenValue", log = "x", main = "Variance Explained")
plot(1:N_TSS, cumsum(Eig_TSS$values) / sum(Eig_TSS$values), col = 2, pch =
16, main = 'Cumulative Variance \n Explained', ylab = 'Varaince explained',
xlab = 'No. of EigenVectors')
lines(1:N_TSS, cumsum(Eig_TSS$values) / sum(Eig_TSS$values), col = 2, pch =
16)

###(b)
```

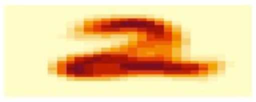
```
for (j in 1:10) {
  ##Clustering the Digits
  X <- t(matrix(c(digits[, , , j]), ncol = 1000))
  h_cl <- hclust(dist(X, method = 'euclidean'))
  m <- cutree(h_cl, k = 5)

  ##All cluster means
  for (k in 1:5) {
    Xbar <- apply(X[m == k, ], 2, mean)
  }

  ##Plot - Digit 2
  if (j == 2 + 1) {
    par(mfrow = c(3,2), mar = rep(4,4))
    plot(h_cl, cex = 0.01, main = 'Dendogram for Digit 2')
    for (k in 1:5) {
      Xbar <- apply(X[m == k, ], 2, mean)
      image(matrix(Xbar, ncol = 28), axes = F)
    }
  }

  ##Plot - Digit 9
  if (j == 9 + 1) {
    par(mfrow = c(3,2), mar = rep(4,4))
    plot(h_cl, cex = 0.01, main = 'Dendogram for Digit 9')
    for (k in 1:5) {
      Xbar <- apply(X[m == k, ], 2, mean)
      image(matrix(Xbar, ncol = 28), axes = F)
    }
  }
}
```

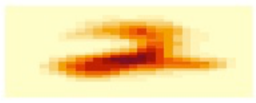


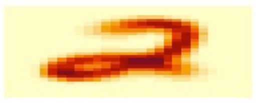

Dendogram for Digit 2
Height
500
2000
dist(X, method = "euclidean")

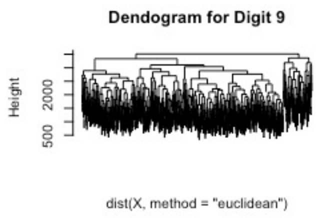


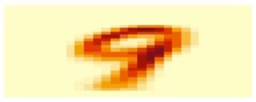


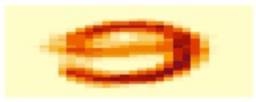


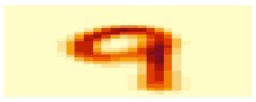


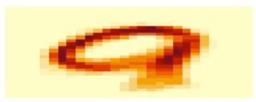


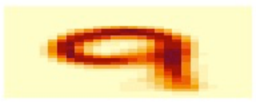

Dendogram for Digit 9
Height
500
2000
dist(X, method = "euclidean")







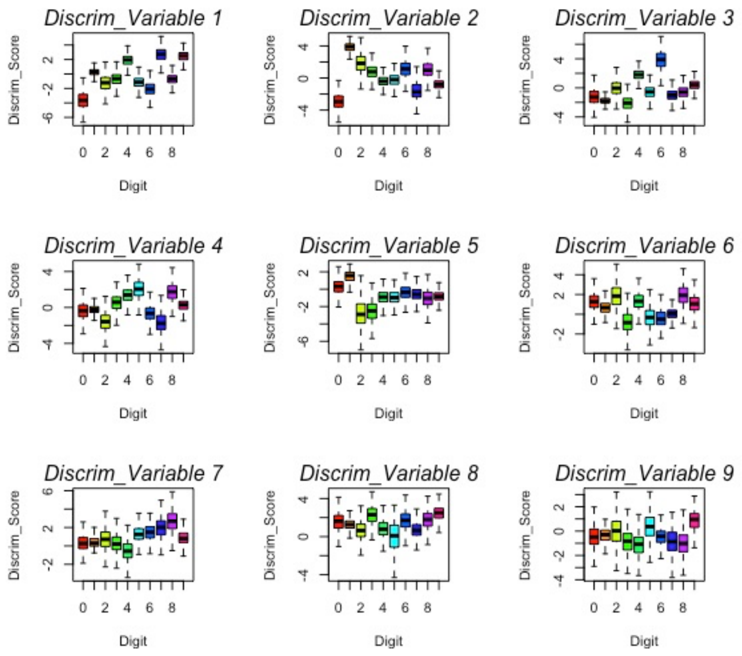




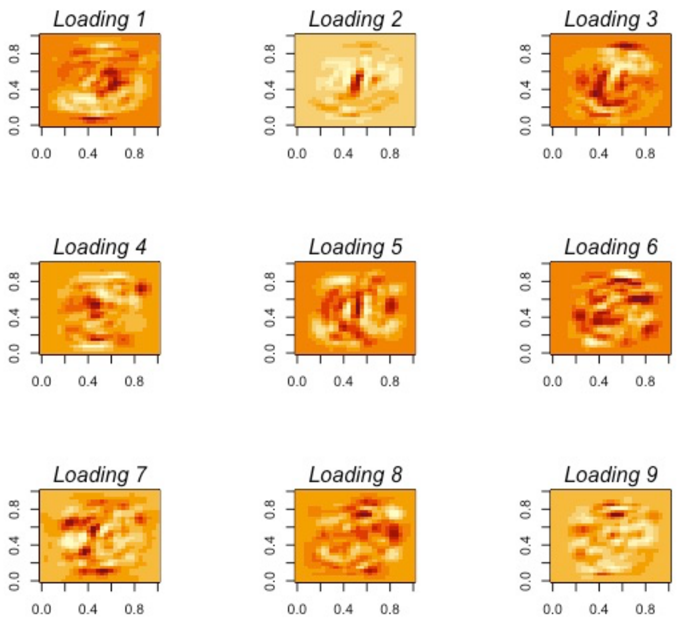
Question 3

```
###(i)
library(MASS)

K <- 90
X <- t(matrix(c(digits), ncol = 10000))     # make sure this matches X in
Q2!!!!

PC <- X %*% V_TSS[, 1:K]
Y <- c(matrix(rep(c(0:9), 1000), ncol = 10, byrow = TRUE))

PC_lda <- lda(Y ~ PC)
D <- PC %*% PC_lda$scaling

par(mfrow = c(3,3), mar = rep(4,4))
for (i in 1:9) {
  boxplot(D[,i] ~ Y, outline = FALSE, pch = ".", col = rainbow(10), xlab =
"Digit", ylab = "Discrim_Score")
  mtext(paste('Discrim_Variable', i, sep = ' '), font = 3, side = 3, line =
0.25, outer = FALSE)
}
mtext("Distribution of Discriminant Scores by Digit", font = 2, side = 3,
line = -1.5, outer = TRUE)

###(ii)
loadings <- PC_lda$scaling
par(mfrow = c(3,3), mar = rep(4,4))
for (i in 1:9) {
  img <- V_TSS[, 1:K] %*% loadings[,i]
  image(matrix(img, ncol = 28))
  mtext(paste('Loading', i, sep = ' '), font = 3, side = 3, line = 0.25,
outer = FALSE)
}
mtext("Loading Vectors", font = 2, side = 3, line = -1.5, outer = TRUE)

###(iv)
##Average within digit covariance
means <- rowMeans(PC_lda$means)
dim(PC)
```

Distribution of Discriminant Scores by Digit



Loading Vectors



Question 4

```
library(MASS)

PC_lda <- lda(Y ~ PC)
PC_qda <- qda(Y ~ PC)

D <- PC %*% PC_lda$scaling
D_lda <- lda(Y ~ D)
D_qda <- qda(Y ~ D)

##Classification and Estimation of Misclassification Rates
PCl_yhat <- as.numeric(predict(PC_lda)$class)
table(PCl_yhat, Y)/1000
error_PCl <- mean((Y - PCl_yhat)^2)

PCq_yhat <- as.numeric(predict(PC_qda)$class)
table(PCq_yhat, Y)/1000
error_PCq <- mean((Y - PCq_yhat)^2)

Dl_yhat <- as.numeric(predict(D_lda)$class)
table(Dl_yhat, Y)/1000
error_Dl <- mean((Y - Dl_yhat)^2)

Dq_yhat <- as.numeric(predict(D_qda)$class)
table(Dq_yhat, Y)/1000
error_Dq <- mean((Y - Dq_yhat)^2)

##Classification and estimation of Misclassification Rates with Leave-one-
Out CV

PC_lda.CV <- lda(Y ~ PC, CV = TRUE)
PC_qda.CV <- qda(Y ~ PC, CV = TRUE)

D <- PC %*% PC_lda$scaling
D_lda.CV <- lda(Y ~ D, CV = TRUE)
D_qda.CV <- qda(Y ~ D, CV = TRUE)

PCl_CV_yhat <- as.numeric(PC_lda.CV$class)
table(PCl_CV_yhat, Y)/1000
error_PCl.CV <- mean((Y - PCl_CV_yhat)^2)

PCq_CV_yhat <- as.numeric(PC_qda.CV$class)
table(PCq_CV_yhat, Y)/1000
error_PCq.CV <- mean((Y - PCq_CV_yhat)^2)

Dl_CV_yhat <- as.numeric(D_lda.CV$class)
table(Dl_CV_yhat, Y)/1000
error_Dl.CV <- mean((Y - Dl_CV_yhat)^2)

Dq_CV_yhat <- as.numeric(D_qda.CV$class)
table(Dq_CV_yhat, Y)/1000
error_Dq.CV <- mean((Y - Dq_CV_yhat)^2)
```