# Problem 1: 8.10#

##a##

```r
stocks <- read.table(file = "http://www.public.iastate.edu/~maitra/stat501/datasets/stocks.dat",
    col.names = c("JPM", "Citibank", "Wells", "RoyalDutchShell", "ExxonMobil"))
s <- cov(stocks)
```

```r
stocks.pc <- prcomp(stocks)
```

```r
stocks.pc$rotation
```

##b##

```r
s.pc3 <- (stocks.pc$sdev)^2
```

```r
cumsum(s.pc3)/sum(s.pc3)
```

##c##

```r
a <- 0.05
m <- 3
n <- nrow(stocks)

CIB <- cbind((stocks.pc$sdev^2)[1:3]/(1-qnorm(a/(2*m))*sqrt(2/n)),
             (stocks.pc$sdev^2)[1:3]/(1+qnorm(a/(2*m))*sqrt(2/n)))
colnames(CIB) <- c("LCI", "UCI")
```

##d##

"almost 90% of the variance was accounted for in just the first 3 components
it is thus reasonable to describe this data with fewer than the 5 variables measured."

# Problem 2: 8.22#

##a##

```r
bulls <- read.table(file="http://www.public.iastate.edu/~maitra/stat501/datasets/bulls.dat")
bulls <- bulls[,3:9]
bulls.pc <- prcomp(bulls)
cumsum(bulls.pc$sdev^2/sum(bulls.pc$sdev^2))
```

"# 80% of variation is explained by 1 factor,
# 99% of variation is explained by 2 factors"

```r
PCs.proportion.variation.enuff <- function(lambda, q = 1, nobs) {
    den <- sum(lambda) # sum of all the eigenvalues
    num <- sum(lambda[1:q]) # sum of the first q eigenvalues
    if (num/den == 1) return(1)
    else {
        se <- sqrt(2 * sum(lambda[1:q]^2)/(nobs*den^2)) # asymptotic sd of the test statistic
        test.stat <- (num/den - 1)/se
        return(pnorm(test.stat))
    }
}
```
p1 <- dim(bulls)[2]
n <- dim(bulls)[1]
p <- p1 - 1
for (i in 1:(p1 - 1)) cat(i, PCs.proportion.variation.enuff(s, q = i, nobs = nrow(bulls)), "n")
# Maitra shows that is no change after two components

bulls.cov <- cov(bulls)
plot(eigen(bulls.cov)$values, type = "b")
# Scree plot confirms 1-2 components

# b Here are the components:
bulls.pc$rotation
# The first component appears to be mainly associated with V4 and V9,
# That is, PrctFFB and SaleWt. Since they are positively correlated in PC1
# but negatively correlated in PC2, we could think of PC1 as
# value of the cow due to PrctFFB and PC2 as value of the cow not due to PrctFFB

# c An index is usually considered a constant loading along a factor, which does not
# appear to be the case here. It would be possible to force one by taking
# SaleHt, SaleWt, and Frame but considering these variables do not form a
# component together, it may not explain very much

# d
bulls.pcr <- prcomp(bulls, scale = T)
par(mfrow = c(1, 2))
plot(bulls.pc$x[, 2], bulls.pc$x[, 1], xlab = "PC2", ylab = "PC1", main = "PCs using S", col = bulls[, 1] + 20)
plot(bulls.pcr$x[, 2], bulls.pcr$x[, 1], xlab = "PC2", ylab = "PC1", main = "PCs using R", col = bulls[, 1] + 20)
# From these plots we can see that, while there is a lot of overlap,
# It appears that there may be some separation between a few of the breeds.
# There are also a few outliers: one cow scored much higher on PC2 that the others,
# another cow was much lower on PC1 than might be expected.

# e
# First PC...
qqnorm(as.numeric(bulls.pc$rotation[, 1]), main = "Q-Q Plot of PC1")

# Following this result, it does not appear as though PC1
# follows a normal distribution

# Question 3 (a)
ids <- scan("http://www.public.iastate.edu/~maitra/stat501/datasets/zipdigit.dat")
ids <- as.factor(ids)

# fit lm on each coordinate
lmfun <- function(y = y, x = x) lmfit <- lm(y ~ x)

lmval <- apply(X = ziptrain, MARGIN=2, FUN = lmfun, x = ids)

# use the anova function on the list of lm output provided for each variable

tmp <- lapply(lmval, anova)
pvalues <- sapply(sapply(tmp, "[", 5), "[", 1)

# the inner sapply extracts the fifth element of each list item, the outer the
# first element.

qvalues <- pvalues * length(pvalues) / rank(pvalues)

sum(qvalues < 0.05)
# 256

# none appear to be significant at the 5% level even after controlling for false
# discovery rates
#
# Let us now get the 100 most significant pixel coordinates.
#
red.zip <- ziptrain[ , order(pvalues)][ ,1:100]

# i.
source(file = "http://www.public.iastate.edu/~maitra/stat501/Rcode/BoxMTest.R")

# Box's M-test statistic can not be computed on the entire dataset because of
# the inability to invert dispersion matrices. One option is to change the
# to utilize a generalized inverse.
#
# Or we can take a subset of the variables and check
#
# BoxMTTest(X = red.zip[ , 1:10], cl = ids)
# ------------------------------------------------
#  MBox Chi-sqr. df P
#  ------------------------------------------------
#  15203.1238 14866.7559         495       0.0000
#  ------------------------------------------------
# Covariance matrices are significantly different.
# $MBox
# 0
# 15203.12
# $ChiSq
# 0
# 14866.76

# $df
# 495

# $pValue
# 0

# For the first ten variables, the null hypothesis is rejected so there is
# significant evidence against the null hypothesis at the 5% level for these
# first ten coordinates.
# Since there is overwhelming evidence against the null in the first ten
# coordinates, this points to the strong possibility that the null hypothesis
# may not be supported in the presence of the alternative for the entire
# coordinates set
#

# ii.

# Because of the test statistic depending on the inverses of the individual
# variances and covariances, we have two options: either stabilize the
# dispersion matrix by eliminating some more variables or take generalized
# inverses. We will take the generalized inverse route: note that for us, there
# is no loss because we will be using a nonparametric bootstrap test, so
# we are not hung up about distributional assumptions that are lost in the
# process.
#
# The function ginv in the MASS library can be used to calculate the generalized
# inverse of a matrix.
#
library(MASS)

# get a list of the generalized inverses of the individual variance-covariance
# matrices

Sinvlist <- tapply(as.matrix(red.zip), rep(ids, ncol(red.zip)),
functional(x, nc) ginv(cov(matrix(x, nc = nc))),
ncol(red.zip))

sqrtmat <- function(mat) {
  ed <- eigen(mat, symmetric = T)
  return((ed$vectors)%*%diag(sqrt(ifelse(ed$values >= 0, ed$values, 0)))%*%t(ed$vectors))
}

Sqrtinvlist <- lapply(Sinvlist, sqrtmat)
getMLEs <- function(X, ids, Sinvs)
{
    #The "denominator" term in the estimate
t1 <- matrix(data=0, nrow=ncol(X), ncol=ncol(X))
    #The "numerator" term in the estimate
t2 <- numeric(length=ncol(X))
    #The sample mean of observations for
    # each group is the MLE of the
    # population mean for the group under
    # the alternative hypothesis of
    # different means for each group
    Xbar <- apply(X, MARGIN=2, FUN=function(x) tapply(x, INDEX=ids, FUN=mean))
    for (i in 0:9)    {
        Xi <- X[ids==i, ]
        n <- ns[i+1]
        Sinv <- Sinvs[[i+1]]
        mui <- Xbar[i+1,]
        t1 <- t1+n*Sinv
        t2 <- t2+n*Sinv%*%mui
    }
    #compute the MLE under H0 from the two
    #sums
    muhat <- solve(t1)%*%t2
    return(list(MLE0=muhat, MLE1=Xbar))
}

mles <- getMLEs(X = red.zip, ids = ids, Sinvs = Sinvlist)

# Find the likelihood ratio test statistic (-2log(lambda)).
LRTest <- function(X, Sinvs, id)
{
    #obtain the MLEs under H0 and H1
    mle <- getMLEs(X, id, Sinvs=Sinvs)
    MLE0 <- mle$MLE0
    MLE1 <- mle$MLE1
    LRT <- 0
    #compute the likelihood ratio test statistic
    ns <- table(ids)
    for (i in 0:9)    {
        Sinv <- Sinvs[[i+1]]
        n <- ns[i+1]
    }
Xbar <- MLE1[(i+1),]
LRT <- LRT+n*t(Xbar-MLE0)%*%Sinv%*%(Xbar-MLE0)
}
return(LRT)
}

# Calculate the residuals under H0 with data X. ids and generalized inverses
resids.data <- function(X, ids, Sinvs, SqrtSinv, mles) {
m <- nrow(X)
p <- ncol(X)
resids <- matrix(data=0, nrow=m, ncol=p)
for (i in 0:9)
{
  #MLEs
  Xi <- X[ids==i,]
  #subtract the MLE for mu from all rows
  resids1 <- sweep(Xi, MARGIN=2, STATS=mles$MLE0, FUN="-")
  Sinv <- SqrtSinv[(i+1)]
  #calculate the standardized residuals
  #resids2 <- apply(resids1, MARGIN=1, FUN=function(x, Sinv)
  #               as.matrix(x)%*%Sinv, Sinv = Sinv)
  resids2 <- as.matrix(resids1)%*%Sinv
  resids[ ids==i, ] <- resids2
}
return(resids)
}

resids <- resids.data(X = red.zip, ids = ids, Sinvs = Sinvlist,
                       SqrtSinv = Sqrtinvlist, mles = mles)

#Function to convert the residuals back into the original scale. X should be
#one of the bootstrapped datasets which has variables in rows and observations
#in columns. MLE0 is the MLE of mu under H0.
convert.resid <- function(X, ids, Ssqrts, mles) {
{
  #The bootstraped datasets have variables
  # in rows and observations in columns,
  # which is less intuitive
  X <- t(X)
m <- nrow(X)
p <- ncol(X)
bootvals <- matrix(data=0, nrow=m, ncol=p)
for (i in 0:9)
{
  #get the appropriate of rows from the
  #dataset
  Xi <- X[ids==i,]
  #get the square root matrix

# Question 3 (b)
#
#
# It is reasonable to use the variance-covariance matrix because the
# observations are measured on the same scale and arguably pixels with greater
# variability in intensity should contribute more. So we perform PCA on the
# matrix

ziptrain.pc <- prcomp(ziptrain, retx = T)

   #Proportion of variance explained
pc.sum <- cumsum(ziptrain.pc$sdev^2)/sum(ziptrain.pc$sdev^2)

source("http://www.public.iastate.edu/~maitra/stat501/Rcode/
Pcs.proportion.variation.enuff.R")
for (i in 1:256) cat("i = ", i, PCs.proportion.variation.enuff(lambda=ziptrain.pc$sdev^2, q = i, propn=0.8, nobs=2000), ","n")

# 26 PCs included.

#using radviz
source("http://www.public.iastate.edu/~maitra/stat501/Rcode/radviz2d.R")
source("http://www.public.iastate.edu/~maitra/stat501/Rcode/mmnorm.R")
source("http://www.public.iastate.edu/~maitra/stat501/Rcode/circledraw.R")
par(mfrow = c(2,1))

radviz2d(dataset = cbind(ziptrain, ids), name = "PCs of Zip")

radviz2d(dataset = cbind(ziptrain.pc$x[,1:26], ids), name = "PCs of Zip")

#
# because of scaling issues when using PCs, the original plot actually looks
# more distinct.
#
#
# test for homogeneity of dispersions
#
BoxMTest(X = ziptrain.pc$x[,1:26], ids)

   # MBox Chi-sqr. df P
   # 60012.3244 56636.7491    3159    0.0000

# Covariance matrices are significantly different.
# $MBox
# 0
# 60012.32

# $\text{ChiSq}$
# 0
# 56636.75

# $\text{df}$
# [1] 3159

# $\text{pValue}$
# 0
# 0

# Not unexpectedly, the dispersions are different

Spcinvlist <- tapply(as.matrix(ziptrain.pc$x[,1:26]),
rep(ids, ncol(ziptrain.pc$x[,1:26])),
function(x, nc) ginv(cov(matrix(x, nc = nc))),
ncol(ziptrain.pc$x[,1:26]))

Sqrtpcinvlist <- lapply(Spcinvlist, sqrtmat)

mles.pc <- getMLEs(X = ziptrain.pc$x[, 1:26], ids = ids, Sinvs = Spcinvlist)

resids.pc <- resids.data(X = ziptrain.pc$x[, 1:26], ids = ids,
Sinv = Spcinvlist, SqrtSinv = Sqrtpcinvlist,
mles = mles.pc)

boot.sam.pc <- bootstrap(X = ziptrain.pc$x[,1:26], nrep = 101, ids = ids,
Ssqrts = Sqrtpcinvlist, Sinvs = Spcinvlist,
resids = resids.pc, mles = mles.pc)

lrt.stat <- as.vector(LRTTest(X = ziptrain.pc$x[,1:26], Sinvs = Spcinvlist,
id = ids))

# [1] 70852.37
#
# H_0 not rejected at the 5% level,