STAT 428 – Homework #3
Luis Steven Lin

Problem 1

Para a

```
# Gram-Schmidt function

gram.Schmidt = function (A){
      m = nrow(A)
      n = ncol(A)
      Q = matrix(0,m,n)
      R = matrix(0,n,n)

      # loop through columns
      for (j in 1:n){
            Q[,j] = A[,j]

            # loop through rows until j-1 row th
            for (i in (1:j)[-j]){
                  # Fill i,j element of R
                  R[i,j] = t(Q[,i])%*%A[,j]
                  # Update the calculation of j element of Q
                  Q[,j] = Q[,j] - R[i,j]*Q[,i]

            }
            # Fill in the j,j element of R
            R[j,j] = sqrt( sum(Q[,j]^2) )
            # Fill in the j element of Q
            Q[,j] = Q[,j]/R[j,j]
      }
return(list("Q"=Q,"R"=R))
}

# Input Data

M = matrix(c(rep(1,5),
            -1.0,-0.5,0.0,0.5,1.0,
              1.0,0.25,0.0,0.25,1.0), nrow=5, ncol=3, byrow=F)

b = c(1.0,0.5,0.0,0.5,2.0)
```

```
# Find the QR decomposition

> gram.Schmidt(M)
$Q
          [,1]          [,2]       [,3]
[1,] 0.4472136 -6.324555e-01  0.5345225
[2,] 0.4472136 -3.162278e-01 -0.2672612
[3,] 0.4472136  1.570092e-17 -0.5345225
[4,] 0.4472136  3.162278e-01 -0.2672612
[5,] 0.4472136  6.324555e-01  0.5345225

$R
         [,1]          [,2]      [,3]
[1,] 2.236068 -5.551115e-17 1.1180340
[2,] 0.000000  1.581139e+00 0.0000000
[3,] 0.000000  0.000000e+00 0.9354143
```

Para b

```
# Part b
# Find x of the linear system Mx = b


# Back substituion function

solve.Backsubs = function (A,b){
      # Ax = b -> QRx = b -> Rx = Q´b

      # Find the QR decomposition
      GS=gram.Schmidt(A)
      Q = GS$Q
      R = GS$R
      m = ncol(A)
      Qb = t(Q)%*%b
      x = rep(0,m)

      # Loop starting from last row of R
      # and work from the bottom up
      for(i in m:1){
            x[i] = (Qb[i]- R[i,]%*%x)/R[i,i]
      }

return(x)

}
```

```
> # Use back substituion algorithm
>
> solve.Backsubs(M,b)
[1] 0.08571429 0.40000000 1.42857143
>
> # Double check using solve function and
> # least square formula:
>
> GS=gram.Schmidt(M)
> solve(GS$R, t(GS$Q)%*%b)
           [,1]
[1,] 0.08571429
[2,] 0.40000000
[3,] 1.42857143
>
> solve(t(M)%*%M)%*%t(M)%*%b
           [,1]
[1,] 0.08571429
[2,] 0.40000000
[3,] 1.42857143
>
```

Problem 2

Cholesky decomposition of the matrix A

$$A = \begin{pmatrix} 2 & -2 \\ -2 & 5 \end{pmatrix}$$

$$A = L \cdot L'$$

Where L is a lower triangular matrix:

$$L = \begin{pmatrix} a & 0 \\ b & c \end{pmatrix}, L' = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \rightarrow L \cdot L' = \begin{pmatrix} a^2 & ab \\ ab & b^2 + c^2 \end{pmatrix}$$

Therefore:

$$a^2 = 2 \qquad \rightarrow a = \sqrt{2}$$

$$ab = -2 \qquad \rightarrow b = -2/\sqrt{2} = -\sqrt{2}$$

$$b^2 + c^2 = 5 \quad \rightarrow c = \sqrt{3}$$

The Cholesky decomposition $A = L \cdot L'$ is then given by

$$L = \begin{pmatrix} \sqrt{2} & 0 \\ -\sqrt{2} & \sqrt{3} \end{pmatrix}$$

Problem 3

Para a & b

```
getwd()
eNose=read.csv('DB_220_IDLH_2mins.txt', header=F, row.names=1, sep='\t')

# Examine data
dim(eNose)
rownames(eNose)
colnames(eNose)
eNose[1:5,1:5]

## Part a and b
# output the first 3 principal components
# use the 147 by 108 matrix as input matrix
# output the "cumulative" screeplot to show
# the importance of first 25 PC directions

find.pca = function(data){
      # Use built-in R function for PCA
      my.pca = prcomp(data)

      # Get the cumulative propotions of PC
      cumProp = summary(my.pca)$imp["Cumulative Proportion",]

      # Output the cumulative scree plot of first, 25 PC's
      plot(1:25,cumProp[1:25],type="o", pch=16,
      xlab="Principal Components",
      ylab="Proportion of variance explained",
      main="Cumulative Screeplot of the first 25 PCs")

      # Output the
      return(my.pca$rotation[,1:3])

}
```

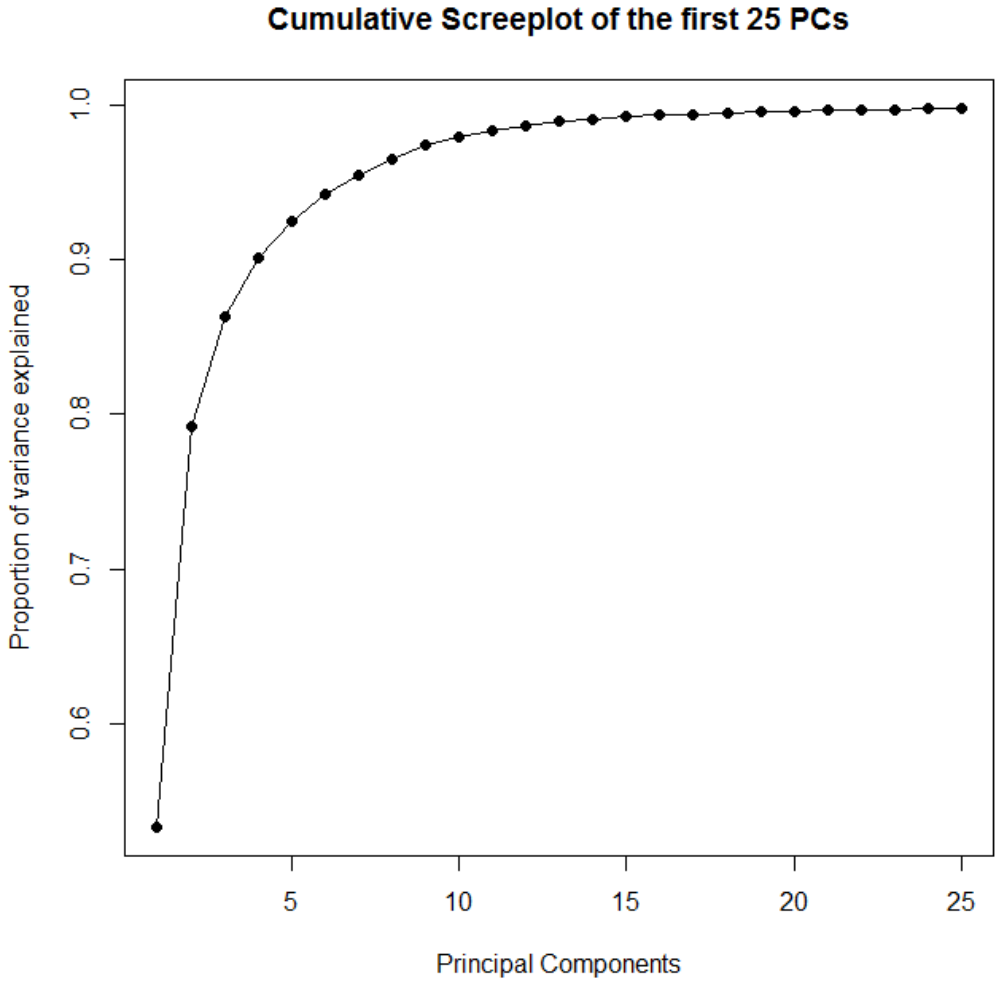
```
> find.pca(eNose)
```

Partial Output

```
                PC1           PC2           PC3
V2   -0.0279994983  0.0782122126 -0.0230522707
V3    0.0102516708 -0.0002028779 -0.0019802636
V4    0.0020603778 -0.0064515716 -0.0013467122
V5    0.0032480041  0.0263254098 -0.0107141887
.....

.....

V105 -0.1808470931 -0.1776892952 -0.2548669668
V106  0.1206777153 -0.0022734036  0.2176693657
V107 -0.1864515444 -0.2489955194 -0.3765902815
V108 -0.1294091669 -0.1285144265 -0.1606695380
V109  0.1093729430  0.0005143414  0.2491517842
```



Part c

```
## Part c
# conduct the linear discriminant analysis on the original
# data and obtain the mis-classification error

library(MASS)

# 147 chemical analytes can be separated into
# 21 groups with each group has 7 analytes

# get group labels
y = rownames(eNose)

# replace "-" with "_" to have the same pattern
y = gsub("-","_",y,fixed=TRUE)

# Keep the group name only (string before "_")
y = sapply(strsplit(y,"_",fixed = TRUE),"[[",1)

# String as levels
y = as.factor(y)

# Run Linear Discriminant Analysis
my.lda=lda(eNose,y)
summary(my.lda)
my.lda$prior
my.lda$counts
my.lda$means[c(1:5),1:5]
my.lda$scaling[c(1:5),1:5]
my.lda$lev
my.lda$svd
my.lda$N
my.lda$call

# Use LDA to predict group
my.pred=predict(my.lda)

# Tabulate actual vs prediction
table(y,my.pred$class)

# Compute misclassification error
sum(my.pred$class!=y)/length(y)
```

Partial Output

```
> my.lda$means[c(1:5),1:5]
                 V2          V3          V4          V5          V6
Ammonia  -8.1503100  1.39917257 -3.58633543 -2.35900443  3.32215429
Arsine   -0.7861711  0.24919171  0.14995143 -0.52430729 -0.25051571
Cl2      -6.0397491  1.61035343 -2.38343686 -6.48695814  3.88737157
Control  -0.1147003 -0.05921329  0.04472057  0.06831800  0.03271386
Diborane -0.9648044 -0.27619057  0.26873714  0.06459486 -0.07660357
> my.lda$scaling[c(1:5),1:5]
          LD1        LD2         LD3         LD4        LD5
V2 -0.1240998  0.2743885 -0.47610861  0.01910402 0.37790468
V3 -0.4192133  0.3301030 -0.04334153  0.55530896 0.69251116
V4  0.2576231 -0.2763742  0.03961398  0.51873261 0.07765936
V5 -0.3918420  1.1385129 -1.20184903 -0.24277506 0.56680232
V6 -0.8852818  1.0946434 -0.52282862 -0.07181317 0.10813625
> my.lda$lev
 [1] "Ammonia"   "Arsine"    "Cl2"       "Control"   "Diborane"  "DM"
 [7] "F2"        "H2S"       "HCHO"      "HCl"       "HCN"       "HF"
[13] "HNO3"      "Hydrazine" "MA"        "MH"        "NO2"
"Phosgene"
[19] "Phosphine" "SO2"       "TA"
> my.lda$svd
 [1] 254.546221 197.400161 148.497375 125.074096 105.453530  95.415719
 [7]  83.589384  68.896212  57.893601  52.496854  42.661832  36.997572
[13]  32.552515  24.071367  22.657381  19.541351  17.128815  12.675959
[19]   8.158525   5.644875

> # Compute misclassification error
> sum(my.pred$class!=y)/length(y)
[1] 0
>
```

- Using a linear discriminant analysis on the original data the reported miss classification errors is zero.

Para d

```
## Part d
# Reduce the dimensionality of the original data using the
# first 20 PC directions and conduct the LDA on the reduced data set

# Get rotated data (scores) of the first 20 PC s
my.pca = prcomp(eNose)
scores = my.pca$x[,1:20]

# Note that my.pca$x, predict(my.pca, eNose) and predict(my.pca)
# are all equivalent.

# Run Linear Discriminant Analysis
my.lda=lda(scores,y)
summary(my.lda)
my.lda$prior
my.lda$counts
my.lda$means[c(1:5),1:5]
my.lda$scaling[c(1:5),1:5]
my.lda$lev
my.lda$svd
my.lda$N
my.lda$call

# Use LDA to predict group
my.pred=predict(my.lda)

# Tabulate actual vs prediction
table(y,my.pred$class)

# Compute misclassification error
sum(my.pred$class!=y)/length(y)
```

Partial Output

```
> my.lda$means[c(1:5),1:5]
                PC1       PC2        PC3       PC4        PC5
Ammonia   300.18734 -63.15085 -9.7089557  15.23220 24.7068951
Arsine    -42.52398 133.96503 17.8139494 -25.46784 10.8001737
Cl2      -112.27476   8.51804 41.4727185  17.75683 -0.3646411
Control   -28.38580 141.63437 47.9555567 -16.94342 -0.7793765
Diborane  -63.91860 115.08815 -0.8782314 -28.38990 21.2347759
> my.lda$scaling[c(1:5),1:5]
             LD1          LD2          LD3           LD4           LD5
PC1 -0.138852063 -0.009813287 -0.006792948  0.0001221271  0.0004016013
PC2  0.024753668 -0.109951234 -0.010783627 -0.0046164711 -0.0101024566
PC3  0.018678161  0.018807402 -0.122819754 -0.0999304763  0.0293449253
PC4 -0.042138238 -0.030394380  0.137510240 -0.1333171092  0.0360192218
PC5 -0.003232265  0.100062798 -0.071270522 -0.0649478760 -0.1430698185
> my.lda$lev
 [1] "Ammonia"   "Arsine"    "Cl2"       "Control"   "Diborane"  "DM"
 [7] "F2"        "H2S"       "HCHO"      "HCl"       "HCN"       "HF"
[13] "HNO3"      "Hydrazine" "MA"        "MH"        "NO2"
"Phosgene"
[19] "Phosphine" "SO2"       "TA"
> my.lda$svd
 [1] 70.35746838 41.10503506 34.18839127 28.87907477 28.09686536
23.99569310
 [7] 21.42804618 18.95413101 16.05979846 14.41137395 12.87688312
8.64260178
[13]  8.06625431  7.57806610  6.58052405  4.99184515  1.36907486
0.51681724
[19]  0.15826426  0.02283107


> # Compute misclassification error
> sum(my.pred$class!=y)/length(y)
[1] 0
>
```

- After reducing the dimensionality of the original data using the first 20 PC directions and conducting the LDA on the reduced data set, the reported miss classification errors is zero.

Para e

```
## Part e
# Write an R code to separate the data into training set and testing set.
# Use the training set to construct the model and testing set to test
# your model. Output the prediction error of the testing set

# Randomly split data set into training and test sets (80/20)
# optional: can create data frame data.frame(cbind(eNose,y))

# number of testing points
nTest = as.integer(0.2*nrow(eNose))

# indices of testing points
set.seed(1)
indicesTest = sample(1:nrow(eNose), size= nTest, replace=FALSE)

# training and testing data sets
eNoseTest = eNose[indicesTest,]
yTest = y[indicesTest]
eNoseTrain = eNose[-indicesTest,]
yTrain = y[-indicesTest]

# Construct model with training set
# Run Linear Discriminant Analysis
my.lda=lda(eNoseTrain,yTrain)
summary(my.lda)
my.lda$prior
my.lda$counts
my.lda$means[c(1:5),1:5]
my.lda$scaling[c(1:5),1:5]
my.lda$lev
my.lda$svd
my.lda$N
my.lda$call

# Use LDA to predict group of testing data
my.pred=predict(my.lda,eNoseTest)

# Tabulate actual vs prediction
table(yTest,my.pred$class)

# Compute misclassification error
sum(my.pred$class!=yTest)/length(yTest)
```

Partial Output

```
> my.lda$means[c(1:5),1:5]
                 V2          V3         V4         V5           V6
Ammonia  -8.1503100  1.39917257 -3.5863354 -2.3590044  3.322154286
Arsine   -0.8234802  0.41391233  0.3923347 -0.7310155 -0.368113333
Cl2      -6.0864715  1.47391467 -2.3903380 -6.4483108  3.810629500
Control  -0.1439615 -0.01014517  0.0101450  0.1429902  0.007249333
Diborane -1.1049286 -0.33159480 -0.1257974  0.2823152  0.013912800
> my.lda$scaling[c(1:5),1:5]
          LD1       LD2        LD3        LD4         LD5
V2 -0.8491139  1.312082  0.2040981 -0.4368560  0.15454262
V3  3.1324904 -3.344985 -1.3035757 -1.2036771 -0.05316388
V4 -0.4309198 -1.527091  0.6082557 -1.0302906  0.93733098
V5  1.2640726  0.317345 -0.7902730 -0.5844898 -0.01615678
V6  0.2560299  1.322829 -0.3383375 -1.2683225  0.70588448
> my.lda$lev
 [1] "Ammonia"   "Arsine"    "Cl2"       "Control"   "Diborane"  "DM"
 [7] "F2"        "H2S"       "HCHO"      "HCl"       "HCN"       "HF"
[13] "HNO3"      "Hydrazine" "MA"        "MH"        "NO2"
"Phosgene"
[19] "Phosphine" "SO2"       "TA"
> my.lda$svd
 [1] 368.485576 238.467013 222.817872 153.922633 152.982819  92.565534
 [7]  84.534279  69.794932  62.299641  49.369361  45.020421  37.621872
[13]  35.994295  32.079697  25.224388  23.299146  18.272095  11.190861
[19]   6.843185   5.838483


> # Compute misclassification error
> sum(my.pred$class!=yTest)/length(yTest)
[1] 0
```

- After separating the data into training set and testing set, and constructing the model and test your model using training set and testing set respectively, the prediction error of the testing set is 0.
- Note: the prediction error can vary depending on the random seed and method of splitting the data set.

Para f (Bonus)

```
## Part f (BONUS)
# Write an R code to compare if the first two PCA directions
# and first two LDA directions generate the same space span.
# (Hint: find a reasonable measurement to evaluate the
# similarity of two directions)

# get principal components and linear discriminant
# from the original data
my.pca = prcomp(eNose)
PC1 = my.pca$rotation[,'PC1']
PC2 = my.pca$rotation[,'PC2']
my.lda=lda(eNose,y)
LD1 = my.lda$scaling[,'LD1']
LD2 = my.lda$scaling[,'LD2']

similarity = function(a,b){
      dot = a%*%b
      mag.a = sqrt(sum(a^2))
      mag.b = sqrt(sum(b^2))
      s = dot/(mag.a*mag.b)
      return(s)
}
similarity(PC1,LD1)
similarity(PC1,LD2)
similarity(PC2,LD1)
similarity(PC2,LD2)
```

```
> similarity(PC1,LD1)
             [,1]
[1,] -0.04125647
> similarity(PC1,LD2)
             [,1]
[1,] -0.00466558
> similarity(PC2,LD1)
             [,1]
[1,] -0.00314187
> similarity(PC2,LD2)
             [,1]
[1,] -0.03886166
```

- Cosine similarity is a measure of similarity between two vectors where:
  - $\cos\theta = \frac{u \cdot v}{\|u\|\|v\|}$
  - output = 1: same direction
  - output = -1: opposite direction
  - output = 0: independence

- In this case we can see that the cosine similarity measures are close to zero, indicating that the directions between the first two PCA directions and first two LDA are different.
- Thus, we can conclude that the first two PCA directions and first two LDA directions do not generate the same space span