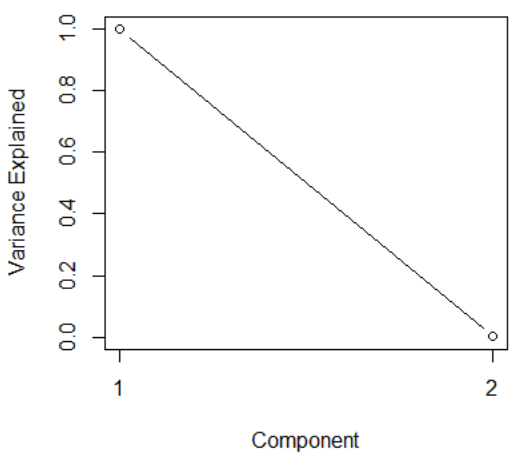
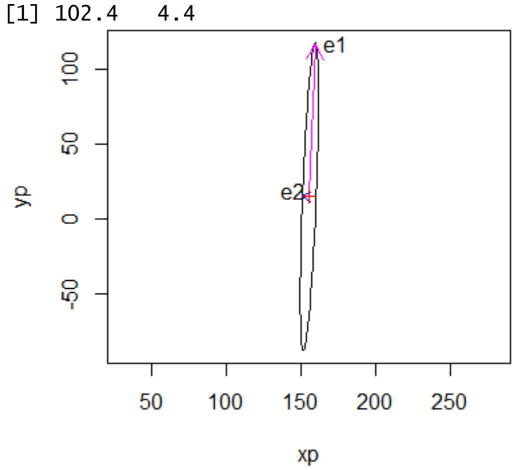
## MVA HW#5 OUTPUTS

### *Question 8.6*

```
> xbar = c(155.6, 14.7)
> S = matrix(c(7476.45, 303.62, 303.62, 26.1
9),2,2)
> pca.compute(S)
----------------
The PCA Procedure using the Covariance Matri
x
----------------
eigen() decomposition
$values
[1] 7488.8   13.8

$vectors
        [,1]    [,2]
[1,] -0.9992  0.0407
[2,] -0.0407 -0.9992

----------------
Variance of the Components
Var(Y1) = 7464
Var(Y2) = 38.5
----------------
Variance Explained by the Components
    Proportion Cumulative
Y1:    0.99816      0.998
Y2:    0.00184      1.000
----------------
```



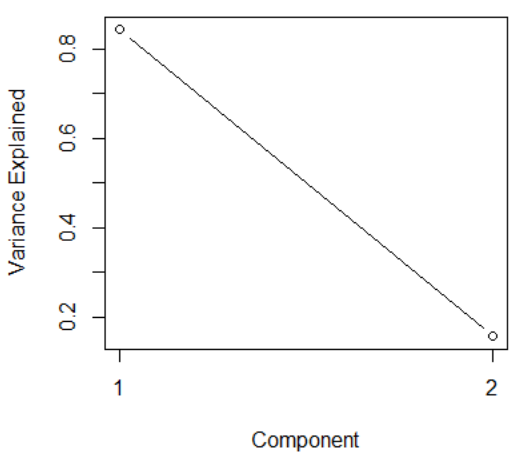

```
> draw.ellipse.for.mean(xbar,S,1.4)
eigen() decomposition
$values
[1] 7488.8   13.8
$vectors
        [,1]    [,2]
[1,] -0.9992  0.0407
[2,] -0.0407 -0.9992

Centers:
[1] 155.6  14.7
Axis:
[1] -102.31   -4.16
[1]   0.179 -4.398
Angle: -1.61
Length:
[1] 102.4   4.4
```




```
> pca.correlation(S)

y = 1     x = 1     cor = -1
y = 1     x = 2     cor = -0.687
----------------
y = 2     x = 1     cor = 0.00175
y = 2     x = 2     cor = -0.726
----------------
```

### *Question 8.7*

```
> (R = cov_to_cor(S))
      [,1]  [,2]
[1,] 1.000 0.686
[2,] 0.686 1.000
> pca.compute(R)
----------------
The PCA Procedure using the Covariance Matri
x
----------------
eigen() decomposition
$values
[1] 1.686 0.314

$vectors
      [,1]   [,2]
[1,] 0.707 -0.707
[2,] 0.707  0.707

----------------
Variance of the Components
Var(Y1) = 1
Var(Y2) = 1
----------------
Variance Explained by the Components
    Proportion Cumulative
Y1:      0.843      0.843
Y2:      0.157      1.000
----------------
```

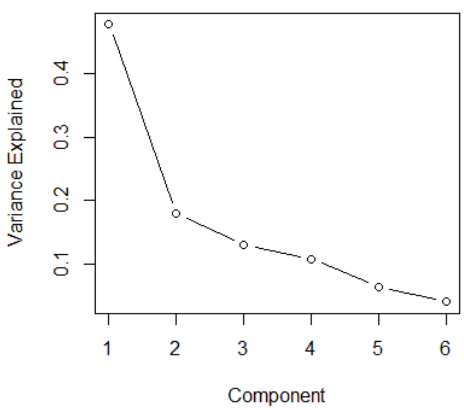


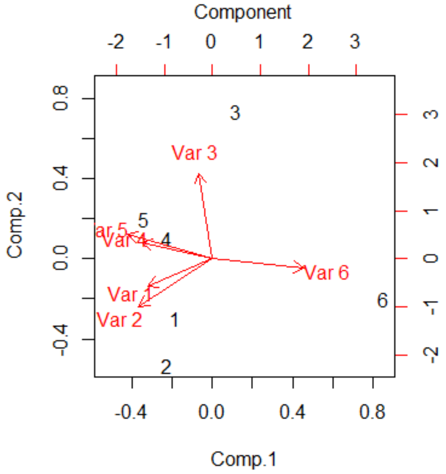

```
>
pca.correlation(R)

y = 1     x = 1     cor = 0.918
y = 1     x = 2     cor = 0.918
----------------
y = 2     x = 1     cor = -0.396
y = 2     x = 2     cor = 0.396
----------------
```

### *Question 8.13*

```
> X = matrix(c(0.889,1.389,1.555,2.222,1.945,1,2.813,1.437,0.999,2.312,2.312,2,
+              1.454,1.091,2.364,2.455,2.909,3,0.294,0.941,1.059,2.000,1.000,1,
+              2.727,2.545,2.819,2.727,4.091,0,3.937,1.250,1.937,2.937,3.749,1,
+              2.786,1.714,2.357,2.071,2.000,2,5.231,2.692,1.077,1.846,2.539,1,
+              1.150,1.100,0.950,2.000,1.000,1,6.500,2.562,1.749,2.562,2.499,1,
+              0.800,1.000,2.200,2.267,2.466,2,4.600,2.000,3.000,2.500,3.400,1,
+              3.500,1.286,2.714,1.286,1.252,3,3.444,2.556,2.388,2.389,3.000,1,
+              4.071,1.000,1.000,2.357,1.572,1,3.692,1.000,2.538,2.154,2.615,1,
+              5.167,3.000,1.000,2.667,3.666,0,0.500,1.000,1.000,2.000,1.000,0,
+              2.385,1.923,2.539,2.154,2.461,1,2.100,1.300,1.300,1.800,2.600,1,
+              5.000,3.250,3.125,2.375,3.375,0,4.571,1.214,3.286,2.571,3.572,1,
+              2.733,1.133,2.600,1.933,1.667,1,4.235,2.294,2.706,2.176,1.883,1,
+              0.000,1.000,1.941,2.000,2.000,0,0.750,1.125,3.000,1.875,2.000,3,
+              3.077,1.462,2.384,2.000,1.846,2,1.600,1.200,2.950,2.000,2.750,1,
+              6.273,3.636,1.182,2.545,3.364,0,2.625,1.000,2.438,1.937,2.062,2,
+              1.250,1.000,2.000,2.000,3.000,1,2.437,2.062,1.687,1.875,1.375,1,
+              4.454,1.727,2.637,2.636,3.546,1,0.133,1.000,1.000,2.000,1.000,0,
+              0.222,1.222,1.445,2.000,1.000,1,2.467,2.667,2.200,1.933,1.800,3,
+              4.000,1.000,4.000,2.167,2.500,0,5.385,3.154,2.384,2.846,2.539,1,
+              0.773,1.000,2.273,1.909,2.091,0,3.786,2.000,1.571,1.786,1.285,3,
+              1.923,1.615,1.693,2.000,1.846,1,1.000,1.333,1.834,2.000,1.917,1,
+              5.800,2.600,3.000,2.800,4.200,1,6.062,1.000,1.562,2.375,1.750,0,
+              3.706,1.235,1.530,2.118,2.294,1,2.444,2.333,1.223,2.444,1.776,3,
+              6.111,2.222,2.889,2.889,3.555,2,2.533,1.067,1.600,2.000,1.333,1,
+              2.167,1.000,2.167,2.000,2.500,1,2.375,1.062,2.375,2.000,2.125,3,
+              1.875,1.312,2.188,2.125,2.062,2,1.750,1.333,1.167,1.750,1.000,1,
+              7.333,1.333,1.459,1.958,1.542,3,5.250,1.375,2.812,2.125,2.563,3,
+              5.182,2.000,2.727,2.818,4.000,2,1.875,2.000,2.250,2.813,2.437,2,
+              5.400,2.000,1.200,1.800,1.400,2,1.154,1.000,1.923,1.846,2.462,1,
+              6.375,2.250,2.500,2.125,3.000,1,9.454,2.727,3.818,2.455,3.272,3,
+              1.000,1.000,1.917,1.833,2.167,1,1.444,1.111,2.000,2.111,2.000,1,
+              1.800,1.100,3.100,2.200,2.600,1,2.818,2.000,1.955,2.045,2.546,2,
+              10.461,2.154,2.769,2.000,2.923,0,4.143,1.929,2.642,2.429,3.142,3,
+              1.227,1.182,1.091,2.227,3.182,1,5.667,3.000,1.667,2.667,5.000,1,
+              4.111,2.556,2.222,2.778,3.778,1,4.444,1.667,2.222,2.000,2.444,0,
+              3.714,3.857,2.643,2.286,3.285,0,7.400,3.700,3.100,2.500,4.200,1,
+              3.182,2.455,1.636,2.273,3.000,1,5.200,2.600,0.800,1.800,2.000,0,
+              2.333,1.667,0.666,1.667,2.166,0,3.333,1.917,2.083,1.917,3.000,1,
+              5.250,2.750,2.500,2.000,4.000,0,7.714,4.000,3.071,2.929,4.428,3,
+              3.846,2.615,3.000,2.692,3.693,2,2.444,1.111,1.000,2.111,1.667,2,
+              5.333,1.917,3.000,2.250,1.917,1,1.556,1.778,3.444,2.667,3.333,1,
+              3.182,1.545,1.910,2.273,3.000,1,6.222,2.444,3.689,2.444,3.445,1,
+              7.231,1.000,3.154,2.308,4.384,2,3.857,1.071,3.000,2.071,2.286,1,
+              3.778,1.944,1.612,1.611,1.945,1,6.000,1.400,2.067,2.267,2.866,2,
+              2.333,3.583,2.334,2.333,2.667,2,7.571,2.143,3.143,2.571,3.929,1,
+              3.667,2.000,2.111,2.778,4.000,3,3.600,2.933,2.067,2.200,2.867,0,
+              3.364,1.273,1.810,2.000,2.273,0,4.100,1.900,2.800,2.000,2.600,2,
+              0.125,1.062,1.437,1.875,1.563,0,6.231,2.769,1.462,2.385,4.000,2,
+              3.000,1.455,2.090,2.273,3.272,2,0.889,1.000,1.000,2.000,1.000,2), ncol =
6, byrow = T)
> (S = cov(X))
      [,1]    [,2]  [,3]   [,4]     [,5]     [,6]
[1,] 4.655  0.9313 0.590 0.2769  1.07489  0.15815
[2,] 0.931  0.6128 0.111 0.1185  0.38889 -0.02485
[3,] 0.590  0.1109 0.571 0.0870  0.34799  0.11013
[4,] 0.277  0.1185 0.087 0.1104  0.21741  0.02181
[5,] 1.075  0.3889 0.348 0.2174  0.86217 -0.00882
[6,] 0.158 -0.0249 0.110 0.0218 -0.00882  0.86146
> (R = cor(X))
      [,1]    [,2]  [,3]   [,4]    [,5]    [,6]
[1,] 1.000  0.5514 0.362 0.3863  0.5366  0.0790
[2,] 0.551  1.0000 0.187 0.4554  0.5350 -0.0342
[3,] 0.362  0.1875 1.000 0.3464  0.4958  0.1570
[4,] 0.386  0.4554 0.346 1.0000  0.7046  0.0707
[5,] 0.537  0.5350 0.496 0.7046  1.0000 -0.0102
[6,] 0.079 -0.0342 0.157 0.0707 -0.0102  1.0000
```

```
> pca.compute(R)
----------------
The PCA Procedure using the Covariance Matrix
----------------
eigen() decomposition
$values
[1] 2.864 1.076 0.778 0.650 0.388 0.243

$vectors
        [,1]    [,2]   [,3]    [,4]    [,5]    [,6]
[1,] -0.4449 -0.0267 -0.339  0.5511  0.6009 -0.1465
[2,] -0.4293 -0.2917 -0.499  0.0614 -0.6873 -0.0764
[3,] -0.3588  0.3801  0.628  0.4211 -0.3318 -0.2116
[4,] -0.4629 -0.0210  0.125 -0.6656  0.2074 -0.5327
[5,] -0.5213 -0.0737  0.203 -0.2005  0.1032  0.7941
[6,] -0.0559  0.8740 -0.430 -0.1787 -0.0531  0.1163

----------------
Variance of the Components
Var(Y1) = 1
Var(Y2) = 1
Var(Y3) = 1
Var(Y4) = 1
Var(Y5) = 1
Var(Y6) = 1
----------------
Variance Explained by the Components
    Proportion Cumulative
Y1:     0.4774      0.477
Y2:     0.1794      0.657
Y3:     0.1296      0.786
Y4:     0.1084      0.895
Y5:     0.0647      0.959
Y6:     0.0405      1.000
----------------
```



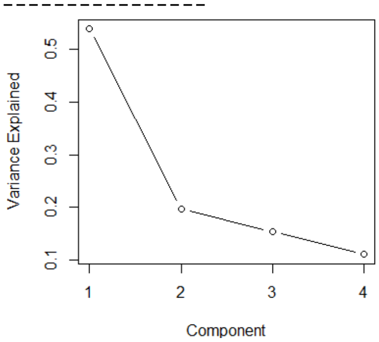



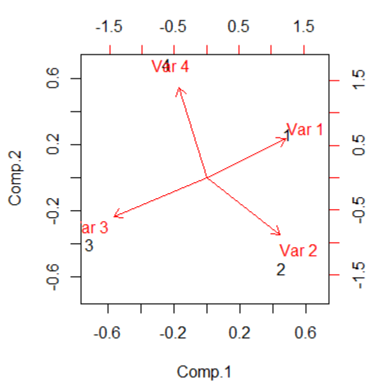

```
> pca.correlation(R)

y = 1     x = 1     cor = -0.753
y = 1     x = 2     cor = -0.727
y = 1     x = 3     cor = -0.607
y = 1     x = 4     cor = -0.783
y = 1     x = 5     cor = -0.882
y = 1     x = 6     cor = -0.0946
----------------
y = 2     x = 1     cor = -0.0277
y = 2     x = 2     cor = -0.303
y = 2     x = 3     cor = 0.394
y = 2     x = 4     cor = -0.0217
y = 2     x = 5     cor = -0.0765
y = 2     x = 6     cor = 0.907
----------------
y = 3     x = 1     cor = -0.299
y = 3     x = 2     cor = -0.44
y = 3     x = 3     cor = 0.554
y = 3     x = 4     cor = 0.11
y = 3     x = 5     cor = 0.179
y = 3     x = 6     cor = -0.379
----------------
y = 4     x = 1     cor = 0.444
y = 4     x = 2     cor = 0.0495
y = 4     x = 3     cor = 0.34
y = 4     x = 4     cor = -0.537
y = 4     x = 5     cor = -0.162
y = 4     x = 6     cor = -0.144
----------------
y = 5     x = 1     cor = 0.374
y = 5     x = 2     cor = -0.428
y = 5     x = 3     cor = -0.207
y = 5     x = 4     cor = 0.129
y = 5     x = 5     cor = 0.0643
y = 5     x = 6     cor = -0.0331
----------------
y = 6     x = 1     cor = -0.0723
y = 6     x = 2     cor = -0.0377
y = 6     x = 3     cor = -0.104
y = 6     x = 4     cor = -0.263
y = 6     x = 5     cor = 0.392
y = 6     x = 6     cor = 0.0573
----------------
```

## *Question 8.16 & 9.18*

```
> R =  matrix(c(1,0.4919,0.2635,0.4653,-0.2277,0.0652,
+              0.4919,1,0.3127,0.3506,-0.1917,0.2045,
+              0.2635,0.3127,1,0.4108,0.0647,0.2493,
+              0.4653,0.3506,0.4108,1,-0.2249,0.2293,
+              -0.2277,-0.1917,0.0647,-0.2249,1,-0.2144,
+              0.0652,0.2045,0.2493,0.2293,-0.2144,1
+              ), 6,6)
```

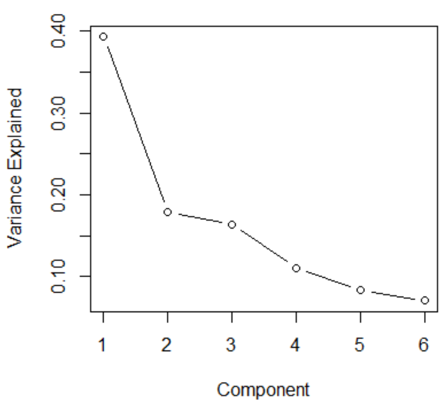
```
> pca.compute(R[-c(5:6),-c(5:6)])
----------------
The PCA Procedure using the Covariance Matrix
----------------
eigen() decomposition
$values
[1] 2.154 0.788 0.616 0.443

$vectors
       [,1]   [,2]   [,3]   [,4]
[1,] -0.527  0.457  0.249  0.672
[2,] -0.503  0.412 -0.614 -0.447
[3,] -0.443 -0.758 -0.368  0.306
[4,] -0.523 -0.215  0.652 -0.505

----------------
Variance of the Components
Var(Y1) = 1
Var(Y2) = 1
Var(Y3) = 1
Var(Y4) = 1
----------------
Variance Explained by the Components
    Proportion Cumulative
Y1:      0.538      0.538
Y2:      0.197      0.735
Y3:      0.154      0.889
Y4:      0.111      1.000
----------------
```

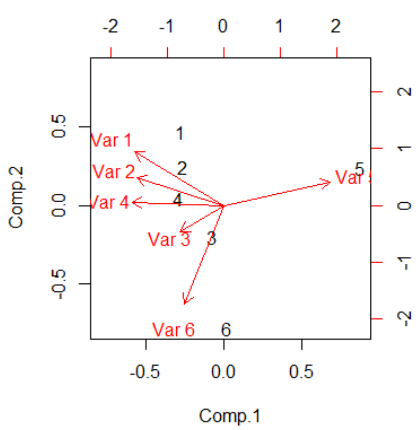



```
> pca.correlation(R[-c(5:6),-c(5:6)])

y = 1     x = 1     cor = -0.773
y = 1     x = 2     cor = -0.739
y = 1     x = 3     cor = -0.65
y = 1     x = 4     cor = -0.767
----------------
y = 2     x = 1     cor = 0.406
y = 2     x = 2     cor = 0.365
y = 2     x = 3     cor = -0.673
y = 2     x = 4     cor = -0.19
----------------
y = 3     x = 1     cor = 0.195
y = 3     x = 2     cor = -0.482
y = 3     x = 3     cor = -0.289
y = 3     x = 4     cor = 0.512
----------------
y = 4     x = 1     cor = 0.447
y = 4     x = 2     cor = -0.297
y = 4     x = 3     cor = 0.203
y = 4     x = 4     cor = -0.336
----------------
```

```
> pca.compute(R)
----------------
The PCA Procedure using the Covariance Matrix
----------------
eigen() decomposition
$values
[1] 2.355 1.072 0.984 0.664 0.500 0.424

$vectors
       [,1]    [,2]    [,3]    [,4]   [,5]    [,6]
[1,] -0.475  0.0221  0.4799  0.0457 -0.358  0.6426
[2,] -0.472 -0.0192  0.2090  0.7029  0.177 -0.4557
[3,] -0.393 -0.5607 -0.2644 -0.1755  0.597  0.2713
[4,] -0.496 -0.0773  0.0322 -0.6043 -0.324 -0.5260
[5,]  0.256 -0.8050  0.0130  0.2182 -0.482 -0.0768
[6,] -0.291  0.1754 -0.8092  0.2454 -0.382  0.1525

----------------
Variance of the Components
Var(Y1) = 1
Var(Y2) = 1
Var(Y3) = 1
Var(Y4) = 1
Var(Y5) = 1
Var(Y6) = 1
----------------
Variance Explained by the Components
    Proportion Cumulative
Y1:      0.3925      0.392
Y2:      0.1786      0.571
Y3:      0.1640      0.735
Y4:      0.1107      0.846
Y5:      0.0834      0.929
Y6:      0.0707      1.000
----------------
```




```
> pca.correlation(R)
y = 1     x = 1     cor = -0.729
y = 1     x = 2     cor = -0.724
y = 1     x = 3     cor = -0.603
y = 1     x = 4     cor = -0.762
y = 1     x = 5     cor = 0.393
y = 1     x = 6     cor = -0.447
----------------
y = 2     x = 1     cor = 0.0229
y = 2     x = 2     cor = -0.0199
y = 2     x = 3     cor = -0.581
y = 2     x = 4     cor = -0.08
y = 2     x = 5     cor = -0.833
y = 2     x = 6     cor = 0.182
----------------
y = 3     x = 1     cor = 0.476
y = 3     x = 2     cor = 0.207
y = 3     x = 3     cor = -0.262
y = 3     x = 4     cor = 0.032
y = 3     x = 5     cor = 0.0129
y = 3     x = 6     cor = -0.803
----------------
y = 4     x = 1     cor = 0.0372
y = 4     x = 2     cor = 0.573
y = 4     x = 3     cor = -0.143
y = 4     x = 4     cor = -0.493
y = 4     x = 5     cor = 0.178
y = 4     x = 6     cor = 0.2
----------------
y = 5     x = 1     cor = -0.253
y = 5     x = 2     cor = 0.125
y = 5     x = 3     cor = 0.422
y = 5     x = 4     cor = -0.229
y = 5     x = 5     cor = -0.341
y = 5     x = 6     cor = -0.27
----------------
y = 6     x = 1     cor = 0.418
y = 6     x = 2     cor = -0.297
y = 6     x = 3     cor = 0.177
y = 6     x = 4     cor = -0.343
y = 6     x = 5     cor = -0.05
y = 6     x = 6     cor = 0.0993
----------------
```

```
> factor_analysis.pca(R[-c(5:6),-c(5:6)],1)
eigen() decomposition
$values
[1] 2.154 0.788 0.616 0.443

$vectors
       [,1]   [,2]   [,3]   [,4]
[1,] -0.527  0.457  0.249  0.672
[2,] -0.503  0.412 -0.614 -0.447
[3,] -0.443 -0.758 -0.368  0.306
[4,] -0.523 -0.215  0.652 -0.505


------------
$L
       [,1]
[1,] -0.773
[2,] -0.739
[3,] -0.650
[4,] -0.767

------------
$h^2
      [,1]
[1,] 0.597
[2,] 0.546
[3,] 0.422
[4,] 0.589

------------
$e
      [,1]  [,2]  [,3]  [,4]
[1,] 0.403 0.000 0.000 0.000
[2,] 0.000 0.454 0.000 0.000
[3,] 0.000 0.000 0.578 0.000
[4,] 0.000 0.000 0.000 0.411

------------
$Residual Matrix
        [,1]    [,2]    [,3]    [,4]
[1,]  0.0000 -0.0789 -0.2386 -0.1277
[2,] -0.0789  0.0000 -0.1673 -0.2162
[3,] -0.2386 -0.1673  0.0000 -0.0879
[4,] -0.1277 -0.2162 -0.0879  0.0000

------------
$Variance Explained:
 0.538
```

```
> factor_analysis.pca(R[-c(5:6),-c(5:6)],2)
eigen() decomposition
$values
[1] 2.154 0.788 0.616 0.443

$vectors
       [,1]   [,2]   [,3]   [,4]
[1,] -0.527  0.457  0.249  0.672
[2,] -0.503  0.412 -0.614 -0.447
[3,] -0.443 -0.758 -0.368  0.306
[4,] -0.523 -0.215  0.652 -0.505


------------
$L
       [,1]   [,2]
[1,] -0.773  0.406
[2,] -0.739  0.365
[3,] -0.650 -0.673
[4,] -0.767 -0.190

------------
$h^2
      [,1]
[1,] 0.762
[2,] 0.679
[3,] 0.875
[4,] 0.625

------------
$e
      [,1]  [,2]  [,3]  [,4]
[1,] 0.238 0.000 0.000 0.000
[2,] 0.000 0.321 0.000 0.000
[3,] 0.000 0.000 0.125 0.000
[4,] 0.000 0.000 0.000 0.375

------------
$Residual Matrix
        [,1]    [,2]    [,3]    [,4]
[1,]  0.0000 -0.2272  0.0345 -0.0504
[2,] -0.2272  0.0000  0.0787 -0.1466
[3,]  0.0345  0.0787  0.0000 -0.2161
[4,] -0.0504 -0.1466 -0.2161  0.0000

------------
$Variance Explained:
 0.735
```

```
> factor_analysis.mle(S=(R[-c(5:6),-c(5:
6)]),m=1, rotation = 'none')
------------
Factor Analysis, MLE method
------------
$L
      [,1]
[1,] 0.708
[2,] 0.630
[3,] 0.485
[4,] 0.653

------------
$h^2
      [,1]
[1,] 0.502
[2,] 0.397
[3,] 0.236
[4,] 0.426

------------
$e
      [,1]  [,2]  [,3]  [,4]
[1,] 0.498 0.000 0.000 0.000
[2,] 0.000 0.603 0.000 0.000
[3,] 0.000 0.000 0.764 0.000
[4,] 0.000 0.000 0.000 0.574

------------
$Residual Matrix
         [,1]     [,2]     [,3]     [,4]
[1,]  0.00000  0.04570 -0.08025  0.00282
[2,]  0.04570  0.00000  0.00688 -0.06085
[3,] -0.08025  0.00688  0.00000  0.09382
[4,]  0.00282 -0.06085  0.09382  0.00000

------------
$Variance Explained:
     [,1]
[1,] 0.39

------------
$Total Variance Explained: 0.39
```

```
> library(psych)

> principal(R[-c(5:6),-c(5:6)],1,TRUE,rotate
 = 'varimax')
Principal Components Analysis
Call: principal(r = R[-c(5:6), -c(5:6)], nfa
ctors = 1, residuals = TRUE,
    rotate = "varimax")
Standardized loadings (pattern matrix) based
 upon correlation matrix
   PC1   h2   u2 com
1 0.77 0.60 0.40   1
2 0.74 0.55 0.45   1
3 0.65 0.42 0.58   1
4 0.77 0.59 0.41   1

                PC1
SS loadings    2.15
Proportion Var 0.54

Mean item complexity =  1
Test of the hypothesis that 1 component is s
ufficient.

The root mean square of the residuals (RMSR)
 is  0.16

Fit based upon off diagonal values = 0.82
```

```
> principal(R[-c(5:6),-c(5:6)],2,TRUE,rotate
 = 'varimax')
Principal Components Analysis
Call: principal(r = R[-c(5:6), -c(5:6)], nfa
ctors = 2, residuals = TRUE,
    rotate = "varimax")
Standardized loadings (pattern matrix) based
 upon correlation matrix
   RC1  RC2   h2   u2 com
1 0.86 0.16 0.76 0.24 1.1
2 0.81 0.17 0.68 0.32 1.1
3 0.09 0.93 0.88 0.12 1.0
4 0.48 0.63 0.63 0.37 1.9

                       RC1  RC2
SS loadings           1.62 1.32
Proportion Var        0.41 0.33
Cumulative Var        0.41 0.74
Proportion Explained  0.55 0.45
Cumulative Proportion 0.55 1.00

Mean item complexity =  1.3
Test of the hypothesis that 2 components are
 sufficient.

The root mean square of the residuals (RMSR)
 is  0.15

Fit based upon off diagonal values = 0.86
```

```
> factor_analysis.mle(S=(R[-c(5:6),-c(5:
6)]),m=1, rotation = 'varimax')
------------
Factor Analysis, MLE method
------------
$L
      [,1]
[1,] 0.708
[2,] 0.630
[3,] 0.485
[4,] 0.653

------------
$h^2
      [,1]
[1,] 0.502
[2,] 0.397
[3,] 0.236
[4,] 0.426

------------
$e
      [,1]  [,2]  [,3]  [,4]
[1,] 0.498 0.000 0.000 0.000
[2,] 0.000 0.603 0.000 0.000
[3,] 0.000 0.000 0.764 0.000
[4,] 0.000 0.000 0.000 0.574

------------
$Residual Matrix
         [,1]     [,2]     [,3]     [,4]
[1,]  0.00000  0.04570 -0.08025  0.00282
[2,]  0.04570  0.00000  0.00688 -0.06085
[3,] -0.08025  0.00688  0.00000  0.09382
[4,]  0.00282 -0.06085  0.09382  0.00000

------------
$Variance Explained:
     [,1]
[1,] 0.39

------------
$Total Variance Explained: 0.39
```

```
> principal(R,4,TRUE,rotate = 'varimax')
Principal Components Analysis
Call: principal(r = R, nfactors = 4, residua
ls = TRUE, rotate = "varimax")
Standardized loadings (pattern matrix) based
 upon correlation matrix
    RC1   RC4   RC2   RC3   h2    u2 com
1  0.42  0.70 -0.23 -0.19 0.76 0.239 2.1
2  0.10  0.92 -0.01  0.21 0.90 0.104 1.1
3  0.72  0.19  0.36  0.31 0.79 0.210 2.1
4  0.85  0.19 -0.26  0.04 0.83 0.170 1.3
5 -0.06 -0.12  0.92 -0.16 0.88 0.119 1.1
6  0.13  0.06 -0.15  0.93 0.92 0.083 1.1

                       RC1  RC4  RC2  RC3
SS loadings           1.46 1.42 1.12 1.08
Proportion Var        0.24 0.24 0.19 0.18
Cumulative Var        0.24 0.48 0.67 0.85
Proportion Explained  0.29 0.28 0.22 0.21
Cumulative Proportion 0.29 0.57 0.79 1.00

Mean item complexity =  1.5
Test of the hypothesis that 4 components are
 sufficient.

The root mean square of the residuals (RMSR)
 is  0.09

Fit based upon off diagonal values = 0.89
```

```
> principal(R,3,TRUE,rotate = 'varimax')
Principal Components Analysis
Call: principal(r = R, nfactors = 3, residua
ls = TRUE, rotate = "varimax")
Standardized loadings (pattern matrix) based
 upon correlation matrix
    RC1   RC3   RC2   h2   u2 com
1  0.85 -0.13 -0.14 0.76 0.24 1.1
2  0.74  0.11 -0.07 0.57 0.43 1.1
3  0.51  0.46  0.54 0.77 0.23 3.0
4  0.71  0.28  0.00 0.59 0.41 1.3
5 -0.24 -0.21  0.86 0.85 0.15 1.3
6  0.05  0.92 -0.15 0.88 0.12 1.1

                       RC1  RC3  RC2
SS loadings           2.11 1.22 1.08
Proportion Var        0.35 0.20 0.18
Cumulative Var        0.35 0.55 0.74
Proportion Explained  0.48 0.28 0.25
Cumulative Proportion 0.48 0.75 1.00

Mean item complexity =  1.5
Test of the hypothesis that 3 components are
 sufficient.

The root mean square of the residuals (RMSR)
 is  0.11

Fit based upon off diagonal values = 0.86
```

```
> factor_analysis.mle(S=R,m=3, rotation = 'v
arimax')
------------
Factor Analysis, MLE method
------------
$L
        [,1]   [,2]    [,3]
[1,]  0.9939 0.0625  0.0573
[2,]  0.4662 0.2111  0.2679
[3,]  0.2010 0.9758  0.0485
[4,]  0.4293 0.3161  0.3297
[5,] -0.2084 0.1343 -0.5054
[6,]  0.0238 0.2268  0.4779

------------
$h^2
      [,1]
[1,] 0.995
[2,] 0.334
[3,] 0.995
[4,] 0.393
[5,] 0.317
[6,] 0.280

------------
$e
      [,1]  [,2]  [,3]  [,4]  [,5] [,6]
[1,] 0.005 0.000 0.000 0.000 0.000 0.00
[2,] 0.000 0.666 0.000 0.000 0.000 0.00
[3,] 0.000 0.000 0.005 0.000 0.000 0.00
[4,] 0.000 0.000 0.000 0.607 0.000 0.00
[5,] 0.000 0.000 0.000 0.000 0.683 0.00
[6,] 0.000 0.000 0.000 0.000 0.000 0.72

------------
$Residual Matrix
          [,1]      [,2]      [,3]      [,4]
      [,5]      [,6]
[1,] -1.11e-16  3.89e-05  4.08e-09 -6.65e-06
 -5.64e-06 -2.24e-05
[2,]  3.89e-05  0.00e+00 -3.10e-05 -4.60e-03
  1.25e-02  1.75e-02
[3,]  4.08e-09 -3.10e-05  0.00e+00  3.57e-05
  7.51e-06 -1.39e-06
[4,] -6.65e-06 -4.60e-03  3.57e-05  0.00e+00
 -1.13e-02 -1.02e-02
[5,] -5.64e-06  1.25e-02  7.51e-06 -1.13e-02
  0.00e+00  1.65e-03
[6,] -2.24e-05  1.75e-02 -1.39e-06 -1.02e-02
  1.65e-03  0.00e+00

------------
$Variance Explained:
      [,1]
[1,] 0.246
[2,] 0.195
[3,] 0.112

------------
$Total Variance Explained: 0.552
```

### *Question 11.1*

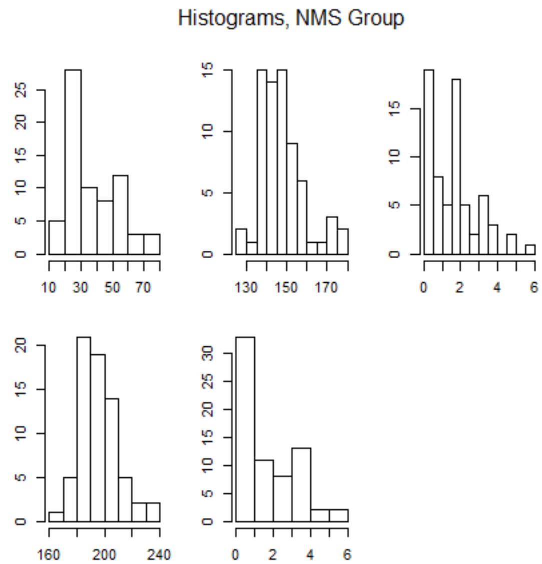
```
> xbar1 = c(3,6)
> xbar2 = c(5,8)
> Spooled = matrix(c(1,1,1,2),2,2)
> (part1 = t(as.matrix(xbar1 - xbar2)) %*% s
olve(Spooled))
     [,1] [,2]
[1,]   -2    0
> (part2 = 0.5*t(as.matrix(xbar1 - xbar2)) %*% solv
e(Spooled)%*%as.matrix(xbar1 + xbar2))
     [,1]
[1,]   -8
```

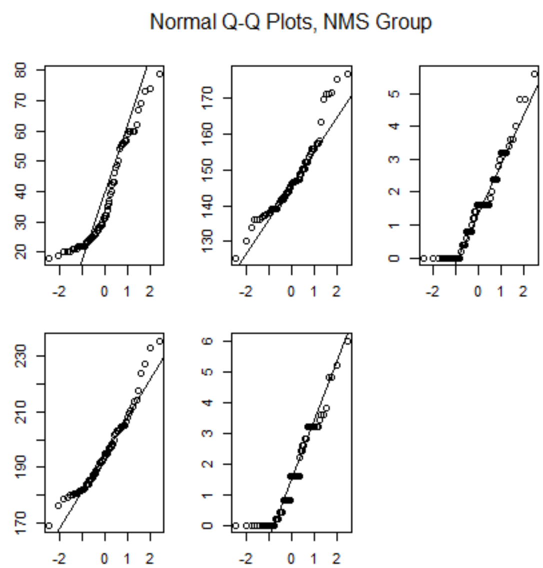
## *Question 11.23*

```
> X = matrix(c(18,152.0,1.6,198.4,0.0,0,19,138.0,0.4,180.8,1.6,0,
+            20,144.0,0.0,186.4,0.8,0,20,143.6,3.2,194.8,0.0,0,
+            20,148.8,0.0,217.6,0.0,0,21,141.6,0.8,181.6,0.8,0,
+            21,136.0,1.6,180.0,0.8,0,21,137.6,1.6,185.6,3.2,0,
+            22,140.4,3.2,182.0,3.2,0,22,137.2,0.0,181.8,0.2,0,
+            22,125.4,1.0,169.2,0.0,0,22,142.4,4.8,185.6,0.0,0,
+            22,150.4,0.0,214.4,3.2,0,22,145.6,1.6,203.6,5.2,0,
+            23,147.2,3.2,196.8,1.6,0,23,139.2,1.6,179.2,0.0,0,
+            24,169.6,0.0,204.8,0.0,0,24,139.2,1.6,176.0,3.2,0,
+            24,153.6,0.0,212.0,0.8,0,25,146.8,0.0,194.8,3.2,0,
+            25,139.2,1.6,198.4,3.2,0,25,136.0,1.6,181.6,2.4,0,
+            26,138.8,1.6,191.6,0.0,0,26,150.4,0.0,205.2,0.4,0,
+            26,139.0,1.4,178.6,0.2,0,27,133.8,0.2,180.8,0.0,0,
+            27,139.0,1.8,190.4,1.6,0,28,136.0,1.6,193.2,3.6,0,
+            28,146.4,0.8,195.6,2.8,0,29,145.2,4.8,194.2,3.8,0,
+            29,146.4,0.8,208.2,0.2,0,29,138.0,2.8,181.2,0.4,0,
+            30,148.8,1.6,196.4,1.6,0,31,137.2,0.0,184.0,0.0,0,
+            31,147.2,0.0,197.6,0.8,0,32,144.0,0.0,185.8,0.2,0,
+            32,156.0,0.0,192.8,2.4,0,34,137.0,0.2,182.4,0.0,0,
+            35,143.2,2.4,184.0,1.6,0,36,141.6,0.8,187.2,1.6,0,
+            37,152.0,1.6,189.2,2.8,0,39,157.4,3.4,227.0,2.6,0,
+            40,141.4,0.6,209.2,1.6,0,42,156.0,2.4,195.2,3.2,0,
+            43,150.4,1.6,180.0,0.8,0,43,142.4,1.6,188.8,0.0,0,
+            46,158.0,2.0,192.0,3.2,0,48,130.0,3.6,190.0,0.4,0,
+            49,152.2,1.4,200.0,4.8,0,49,150.0,3.2,206.6,2.2,0,
+            50,146.4,2.4,191.6,2.8,0,54,146.0,1.2,203.2,1.6,0,
+            55,140.8,0.0,184.0,1.6,0,56,140.4,0.4,203.2,1.6,0,
+            56,155.8,3.0,187.8,2.6,0,56,141.6,0.8,196.8,1.6,0,
+            57,144.8,0.8,188.0,0.8,0,57,146.8,3.2,191.6,0.0,0,
+            59,176.8,2.4,232.8,0.8,0,60,171.0,1.8,202.0,3.6,0,
+            60,163.2,0.0,224.0,0.0,0,60,171.6,1.2,213.8,3.4,0,
+            60,146.4,4.0,203.2,4.8,0,62,146.8,3.6,201.6,3.2,0,
+            67,154.4,2.4,205.2,6.0,0,69,171.2,1.6,210.4,0.8,0,
+            73,157.2,0.4,204.8,0.0,0,74,175.2,5.6,235.6,0.4,0,
+            79,155.0,1.4,204.4,0.0,0,23,148.0,0.8,205.4,0.6,1,
+            25,195.2,3.2,262.8,0.4,1,25,158.0,8.0,209.8,12.2,1,
+            28,134.4,0.0,198.4,3.2,1,29,190.2,14.2,243.8,10.6,1,
+            29,160.4,18.4,222.8,31.2,1,31,227.8,90.2,270.2,83.0,1,
+            34,211.0,3.0,250.8,5.2,1,35,204.8,12.8,254.4,11.2,1,
+            36,141.2,6.8,194.4,21.6,1,39,157.4,3.4,227.0,2.6,1,
+            42,166.4,0.0,226.0,0.0,1,43,191.8,35.4,243.6,40.8,1,
+            44,156.8,0.0,203.2,0.0,1,44,202.8,29.2,246.4,24.8,1,
+            44,165.2,18.4,254.0,46.4,1,45,162.0,5.6,224.4,8.8,1,
+            45,138.4,0.8,176.8,4.0,1,45,158.4,1.6,214.4,0.0,1,
+            46,155.4,1.8,201.2,6.0,1,46,214.8,9.2,290.6,0.6,1,
+            47,185.0,19.0,274.4,7.6,1,48,236.0,20.0,328.0,0.0,1,
+            57,170.8,24.0,228.4,33.6,1,57,165.6,16.8,229.2,15.6,1,
+            58,238.4,8.0,304.4,6.0,1,58,164.0,0.8,216.8,0.8,1,
+            58,169.8,0.0,219.2,1.6,1,59,199.8,4.6,250.2,1.0,1), ncol
= 6, byrow = TRUE)
```

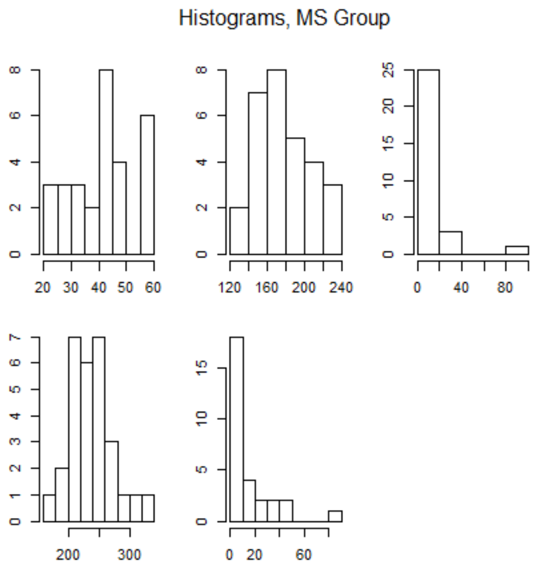
```
> x1 = X[X[,6]==0,][,c(1:5)]
> x2 = X[X[,6]==1,][,c(1:5)]
```

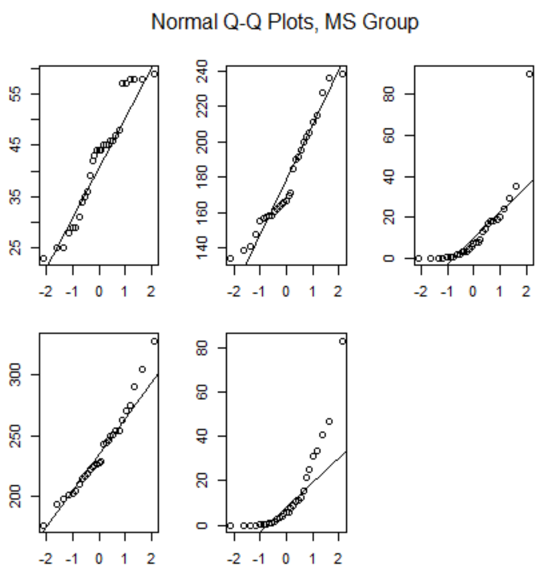
```
> combine_plots(x1, "NMS Group")
```



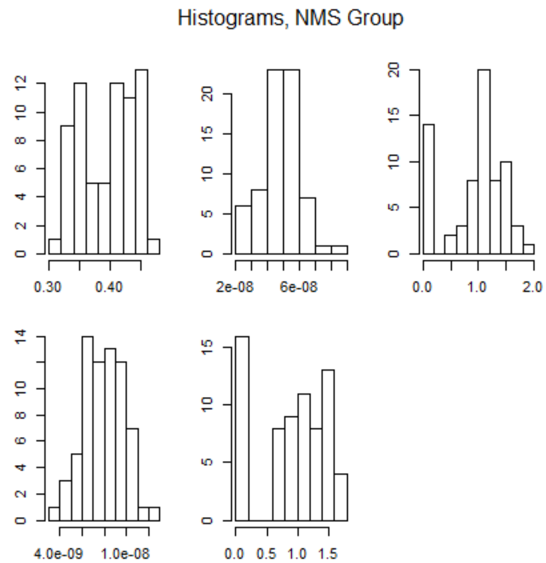



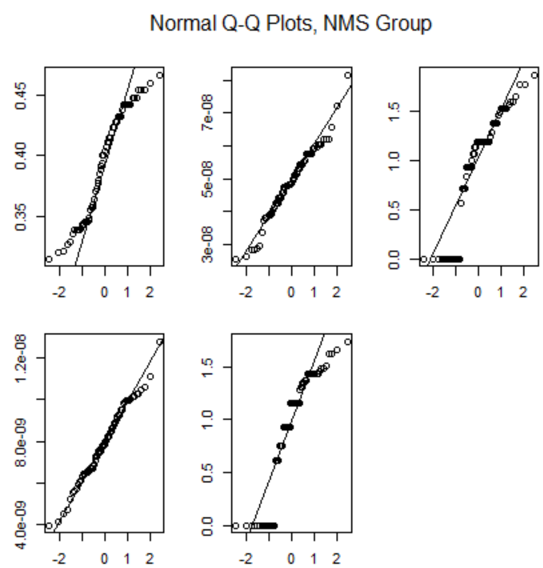
```
> combine_plots(x2, "MS Group")
```



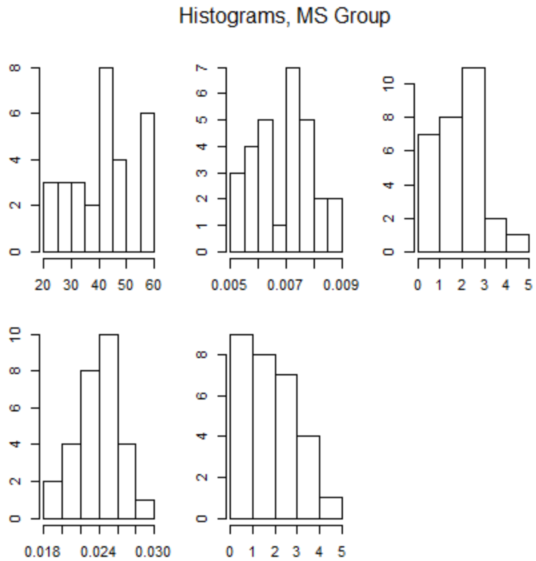



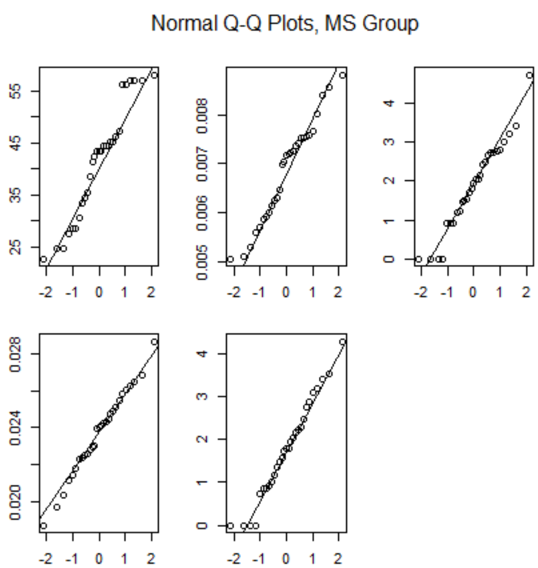

```
> trans1 = transform_data(x1)
    Y1     Y2     Y3     Y4     Y5
-0.264 -3.378  0.361 -3.542  0.310
> x1.trans = trans1[[2]]
> combine_plots(x1.trans, "NMS Group")
```



```
> trans2 = transform_data(x2)
    Y1     Y2     Y3     Y4     Y5
 0.996 -0.966  0.345 -0.686  0.330
> x2.trans = trans2[[2]]
> combine_plots(x2.trans, "MS Group")
```




```
> compute_line(x1,x2)

------------
Computing the Fisher's Line Discriminant
Function
------------
Mean of X1:
[1]  37.99 147.29   1.56 195.60   1.62
Mean of X2:
[1]  42.1 178.3  12.3 236.9  13.1
Pooled Covariance:
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] 232.0  83.0 -2.1  93.3  -6.4
[2,]  83.0 325.9 72.6 341.8  32.6
[3,]  -2.1  72.6 93.8  69.4  87.1
[4,]  93.3 341.8 69.4 475.4  25.3
[5,]  -6.4  32.6 87.1  25.3 104.1

------------
yhat =
(0.0234 -0.0345 0.2103 -0.0839 -0.2535) x + 23.2

yhat      >= 0      obs belongs to population 1;
           <  0      otherwise
------------
      predict.binary
actual  0  1
     0 22  7
     1  3 66
Apparent Error Rate: 0.102
------------
```

## Define Functions

```
> draw.ellipse.for.mean = function (mu, S, c2,
npoints = 100, showcentre=T, showvectors = T,...){
+   angle = function (x, y){
+     returnValue(atan2(x,y))
+   }
+
+   draw.ellipse = function (a = 1, b = 1, theta = 0,
        xc = 0, yc = 0, newplot = T, npoints = 100, ...){
+     t <- seq(0, 2 * pi, length = npoints + 1)
+     xp <- a*cos(t)*cos(theta) - b*sin(t)*sin(theta)+xc
+     yp <- a*cos(t)*sin(theta) + b*sin(t)*sin(theta)+yc
+     if (newplot){
+       plot = plot(xp, yp, asp = 1,type = "l", ...)
+     }
+     else lines(xp, yp, ...)
+   }
+
+   if (nrow(S) == 2 && isSymmetric.matrix(S) &&
        ncol(S) == 2) {
+     ei = eigen(S)
+     print(ei)
+     hlen <- sqrt(c2*ei$val)
+     theta <- angle(ei$vec[1, 1], ei$vec[2, 1])
+     draw.ellipse(hlen[1], hlen[2], theta, mu[1], mu[2])
+     cat('Centers: \n')
+     print(mu)
+     cat('Axis: \n')
+     print(hlen[1]*ei$vectors[,1])
+     print(hlen[2]*ei$vectors[,2])
+     cat('Angle:',theta)
+     cat('Length:\n')
+     print(hlen)
+     if(showvectors){
+       library(graphics)
+       arrows(x0=mu[1],y0=mu[2], x1=mu[1]+abs(cos
              (theta)*hlen[1]),y1=mu[2]
              +abs(sin(theta)*hlen[1]),col = 6,
              code = 2, length = 1/7)
+       usr = par('usr')
+       correct = abs(usr[2] - usr[1])/20
+       text(mu[1]+abs(cos(theta)*hlen[1])+correct,
          mu[2]+abs(sin(theta)*hlen[1]),'e1')
+       minor.angle=angle(ei$vectors[1,2],ei$vectors[2,2])
+       correct = abs(usr[4] - usr[3])/20
+       text(mu[1]+abs(cos(theta)*hlen[2])-correct,
          mu[2]+abs(sin(theta)*hlen[2]),'e2')
+       arrows(x0=mu[1],y0=mu[2], x1=mu[1]
              -abs(cos(minor.angle)*hlen[2]),
              y1=mu[2]+abs(sin(minor.angle)
                 *hlen[2]),col = 4, code = 2,
              length=1/15)
+     }
+     if (showcentre){
+       points(mu[1], mu[2],pch = 3, col = 2)
+     }
+   }
+ }
```

```
> factor_analysis.pca = function(S, m){
+   ei = eigen(S)
+   print(ei)
+   cat('\n------------\n$L\n')
+   L = ei$vectors %*% diag(sqrt(ei$values))[,c(1:m)]
+   print(L)
+   h = apply(L**2, 1, sum)
+   cat('\n------------\n$h^2\n')
+   print(matrix(h))
+   e = diag(diag(S) - h)
+   cat('\n------------\n$e\n')
+   print(e)
+   residual = S - L %*% t(L) - e
+   cat('\n------------\n$Residual Matrix\n')
+   print(residual)
+   var.explained = sum(ei$values[1:m])/ncol(S)
+   cat('\n------------\n$Variance Explained:\n',
        var.explained)
+ }
```

```
> factor_analysis.mle = function(S, m, rotation){
+   factor = factanal(factors = m, covmat = S,
                      rotation = rotation)
+   cat('------------\nFactor Analysis, MLE method\n')
+   L = matrix(factor$loading,ncol = m)
+   cat('------------\n$L\n')
+   print(L)
+   h = apply(L**2, 1, sum)
+   cat('\n------------\n$h^2\n')
+   print(matrix(h))
+   e = diag(diag(S) - h)
+   cat('\n------------\n$e\n')
+   print(e)
+   residual = S - L %*% t(L) - e
+   cat('\n------------\n$Residual Matrix\n')
+   print(residual)
+   var.explained = apply(L**2,2,sum) / sum(diag(S))
+   cat('\n------------\n$Variance Explained:\n')
+   print(matrix(var.explained))
+   cat('\n------------\n$Total Variance Explained:
', sum(var.explained))
+   if(m>2){
+     biplot.factanal <- function (fa.fit){
+       x = fa.fit$scores[,1:2]
+       y = fa.fit$loadings[,1:2]
+       biplot(x,y)
+     }
+   }
+ }
```

```
> pca.correlation = function(S){

+   ei = eigen(S)
+   n = dim(S)[1]
+   for(i in c(1:n)){
+     for(j in c(1:n)){
+       cor = ei$vectors[j,i] * sqrt(ei$values[i])
/ sqrt(S[j,j])
+       cat('\ny =',i,'\tx =', j, '\tcor =', cor)
+     }
+     cat('\n----------------')
+   }
+ }
```

```
> pca.compute = function(S){
+   cat('----------------\nThe PCA Procedure using
        the Covariance Matrix\n----------------\n')
+   ei = eigen(S)
+   print(ei)
+   n = dim(S)[1]
+   cat('----------------\nVariance of the Components')
+   for(i in c(1:n)){
+     cat('\nVar(Y',i,') = ',(ei$vectors[,i])**2
          %*% diag(S), sep = '')
+   }
+
+   cat('\n----------------\nVariance Explained
         by the Components\n')
+   var_explained = cum_var_explained = NULL
+   for(i in c(1:n)){
+     var_explained = c(var_explained,
                        ei$values[i]/sum(ei$values))
+     cum_var_explained = c(cum_var_explained,
                             sum(var_explained))
+   }
+   all_var_explained = cbind(var_explained,
                               cum_var_explained)
+   colnames(all_var_explained) = c('Proportion',
                                     'Cumulative')
+   rownames(all_var_explained) = paste('Y',
                               c(1:n),':', sep = '')
+   print(all_var_explained)
+   plot(var_explained, type = 'b', xaxt = 'n',
         ylab = 'Variance Explained',
         xlab = 'Component')
+   axis(1,at=c(1:n),labels=c(1:n))
+   if(n>2){
+     biplot(princomp(S, cor = TRUE))
+   }
+   cat('----------------\n')
+ }
```

```
> combine_plots = function(x, title = NULL,...){

+   n = ncol(x)
+   if(n==1){}
+   else if(n==2){
+     par(mfrow = c(1,2),mar = c(2,2,2,2),
          oma = c(0,0,2,0))
+   }else if (n<=4){
+     par(mfrow = c(2,2),mar = c(2,2,2,2),
          oma = c(0,0,2,0))
+   }else if (n<=6){
+     par(mfrow = c(2,3),mar = c(2,2,2,2),
          oma = c(0,0,2,0))
+   }else if (n<=9){
+     par(mfrow = c(3,3),mar = c(2,2,2,2),
          oma = c(0,0,2,0))
+   }else if(n>9){
+     warning("Too many variables, only plot the
               first 9 variables")
+     par(mfrow = c(3,3),mar = c(2,2,2,2),
          oma = c(0,0,2,0))
+   }
+   if(is.null(title)){
+     t = 'Histograms'
+   }else{
+     t = paste('Histograms',title, sep = ', ')
+   }
+   names = paste('x', c(1:n), sep = '')
+   library(graphics)
+   for(i in c(1:n)){
+     hist(x[,i], main=NULL, xlab = names[i],
           ylab = NULL,...)
+   }
+   mtext(t, outer = TRUE, cex = 1)
+
+   if(is.null(title)){
+     t = 'Normal Q-Q Plots'
+   }else{
+     t = paste('Normal Q-Q Plots',title, sep = ', ')
+   }
+   par(par())
+   for(i in c(1:n)){
+     qqnorm(x[,i], main=NULL, xlab = names[i],
             ylab = NULL,...)
+     qqline(x[,i])
+   }
+   mtext(t, outer = TRUE, cex = 1)
+   autoimage::reset.par()
+ }
```

```
> transform_data = function(x,...){
+   if(all(x>0)){
+     summary(trans <- car::powerTransform((x)~1))
+   }else{
+     summary(trans <- car::powerTransform((x+0.001
                                          +min(x))~1))
+   }
+   x.trans = NULL
+   print(trans$lambda)
+   for(i in c(1:ncol(x))){
+    x.trans = cbind(x.trans, x[,i]**trans$lambda[i])
+   }
+   returnValue(list(trans$lambda, x.trans))
+ }
```

```
> compute_line = function(x1,x2){
+   cat("\n------------\nComputing the Fisher's Line
    Discriminant Function\n------------\nMean of     X1:\n")
+   xbar1 = apply(x1,2,mean)
+   print(xbar1)
+   cat('Mean of X2:\n')
+   xbar2 = apply(x2,2,mean)
+   print(xbar2)
+   n1 = nrow(x1)
+   n2 = nrow(x2)
+   p1 = ncol(x1)
+   p2 = ncol(x2)
+   cov1 = cov(x1)
+   cov2 = cov(x2)
+   cat('Pooled Covariance:\n')
+   Spooled = ((n1-1)*cov1+(n2-1)*cov2)/(n1+n2-2)
+   print(Spooled)
+   part1 = t(as.matrix(xbar1 - xbar2)) %*% solve(Spooled)
+   part2 = 0.5*t(as.matrix(xbar1 - xbar2)) %*% sol
    ve(Spooled)%*%as.matrix(xbar1 + xbar2)
+   cat('\n------------\n')
+   cat('yhat = (',paste(round(part1, digits = 4)),
    ') x - (', part2,')\n\n', sep = ' ')
+   cat('yhat\t>= 0\tobs belongs to population 1;\n
    \t<  0\totherwise')
+   cat('\n------------\n')
+   X = rbind(x1,x2)
+   actual = c(rep(1,n1),rep(0,n2))
+   seperate = function(row){
+     returnValue(t(matrix(part1)) %*% matrix(row) - part2)
+   }
+   predict.continuous = apply(X, 1, seperate)
+   predict.binary = as.numeric(predict.continuous>=0)
+   table = table(actual,predict.binary)
+   print(table)
+   apparent_error_rate = 1 - sum(diag(table))/nrow(X)
+   cat('Apparent Error Rate:', apparent_error_rat
    e,'\n------------\n')
+ }
```

```
> cov_to_cor = function(S){
+   stopifnot(isSymmetric.matrix(S))
+   p = diag(1, nrow(S))
+   for(i in c(1:nrow(S))){
+     for(j in i+1:nrow(S)){
+       if(j<=nrow(S)){
+         p[i,j] = S[i,j] / sqrt(S[i,i]) / sqrt(S[j,j])
+         p[j,i] = p[i,j]
+       }
+     }
+   }
+   returnValue(p)
+ }
```