Homework3

Yi Chen

2/6/2020

Homework3

Part A

1 symeertrize the confusion probability matrix

```
confusion <-  matrix(c(97,4,4,7,2,9,87,8,37,9,8,16,93,12,12,11,59,17,96,12,9,15
,26,12,86),nrow = 5,byrow = T)
confusion
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   97    4    4    7    2
## [2,]    9   87    8   37    9
## [3,]    8   16   93   12   12
## [4,]   11   59   17   96   12
## [5,]    9   15   26   12   86
```

```
symmetric_confusion <- (confusion + t(confusion))/2
symmetric_confusion
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 97.0  6.5  6.0  9.0  5.5
## [2,]  6.5 87.0 12.0 48.0 12.0
## [3,]  6.0 12.0 93.0 14.5 19.0
## [4,]  9.0 48.0 14.5 96.0 12.0
## [5,]  5.5 12.0 19.0 12.0 86.0
```

2 transform into dissimilarity

```
#dissimilarity <- abs(symmetric_confusion - apply(symmetric_confusion,1,max))
dissimilarity <- abs(symmetric_confusion - max(symmetric_confusion))
dissimilarity
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.0 90.5 91.0 88.0 91.5
## [2,] 90.5 10.0 85.0 49.0 85.0
## [3,] 91.0 85.0  4.0 82.5 78.0
## [4,] 88.0 49.0 82.5  1.0 85.0
## [5,] 91.5 85.0 78.0 85.0 11.0
```

Note, after this step some diag elements in matrix is still not zero.

3 check the traingular inequality. Since the matrix is symmetric, we only need to check the low half of the matrix.

```
# all combination of 3 points out of 5
comb_3 <- combn(5,3)

dis_satisfied <- c()
for (i in 1:ncol(comb_3)){
  if ((dissimilarity[comb_3[,i][1],comb_3[,i][2]] > dissimilarity[comb_3[,i][1
],comb_3[,i][3]] + dissimilarity[comb_3[,i][2],comb_3[,i][3]]) |
      (dissimilarity[comb_3[,i][1],comb_3[,i][2]] < dissimilarity[comb_3[,i][1
],comb_3[,i][3]] - dissimilarity[comb_3[,i][2],comb_3[,i][3]])){
          dis_satisfied <- c(dis_satisfied,dissimilarity[comb_3[,i][1],comb_3[,
i][2]] - (dissimilarity[comb_3[,i][1],comb_3[,i][3]] + dissimilarity[comb_3[,i]
[2],comb_3[,i][3]]))
      }
}
dis_satisfied
```

```
## NULL
```

There is no break of triangle inequality.

4

```
diag(dissimilarity) <- 0
dissimilarity
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.0 90.5 91.0 88.0 91.5
## [2,] 90.5  0.0 85.0 49.0 85.0
## [3,] 91.0 85.0  0.0 82.5 78.0
## [4,] 88.0 49.0 82.5  0.0 85.0
## [5,] 91.5 85.0 78.0 85.0  0.0
```

Part B

5 square the matrix

```
d_square <- dissimilarity^2
d_square
```

```
##         [,1]    [,2]    [,3]    [,4]    [,5]
## [1,]    0.00 8190.25 8281.00 7744.00 8372.25
## [2,] 8190.25    0.00 7225.00 2401.00 7225.00
## [3,] 8281.00 7225.00    0.00 6806.25 6084.00
## [4,] 7744.00 2401.00 6806.25    0.00 7225.00
## [5,] 8372.25 7225.00 6084.00 7225.00    0.00
```

double-center

```
b <- d_square
for (i in 1:ncol(d_square)){
  for (j in 1:ncol(d_square)){
    b[i,j] <- -0.5 * (d_square[i,j] - colMeans(d_square)[j] - rowMeans(d_squar
e)[i] + mean(d_square))
  }
}
b
```

```
##           [,1]    [,2]      [,3]      [,4]    [,5]
## [1,]  3735.350 -1114.4  -824.275  -977.775  -818.9
## [2,] -1114.400  2226.1 -1050.900   939.100  -999.9
## [3,]  -824.275 -1050.9  2897.100  -928.025   -93.9
## [4,]  -977.775   939.1  -928.025  2053.100 -1086.4
## [5,]  -818.900  -999.9   -93.900 -1086.400  2999.1
```

Part C

7 PCA

```
Bcomp <- eigen(b)
wts <- diag(sqrt(Bcomp$values))
principal <- Bcomp$vectors %*% wts
round(principal,2)
```

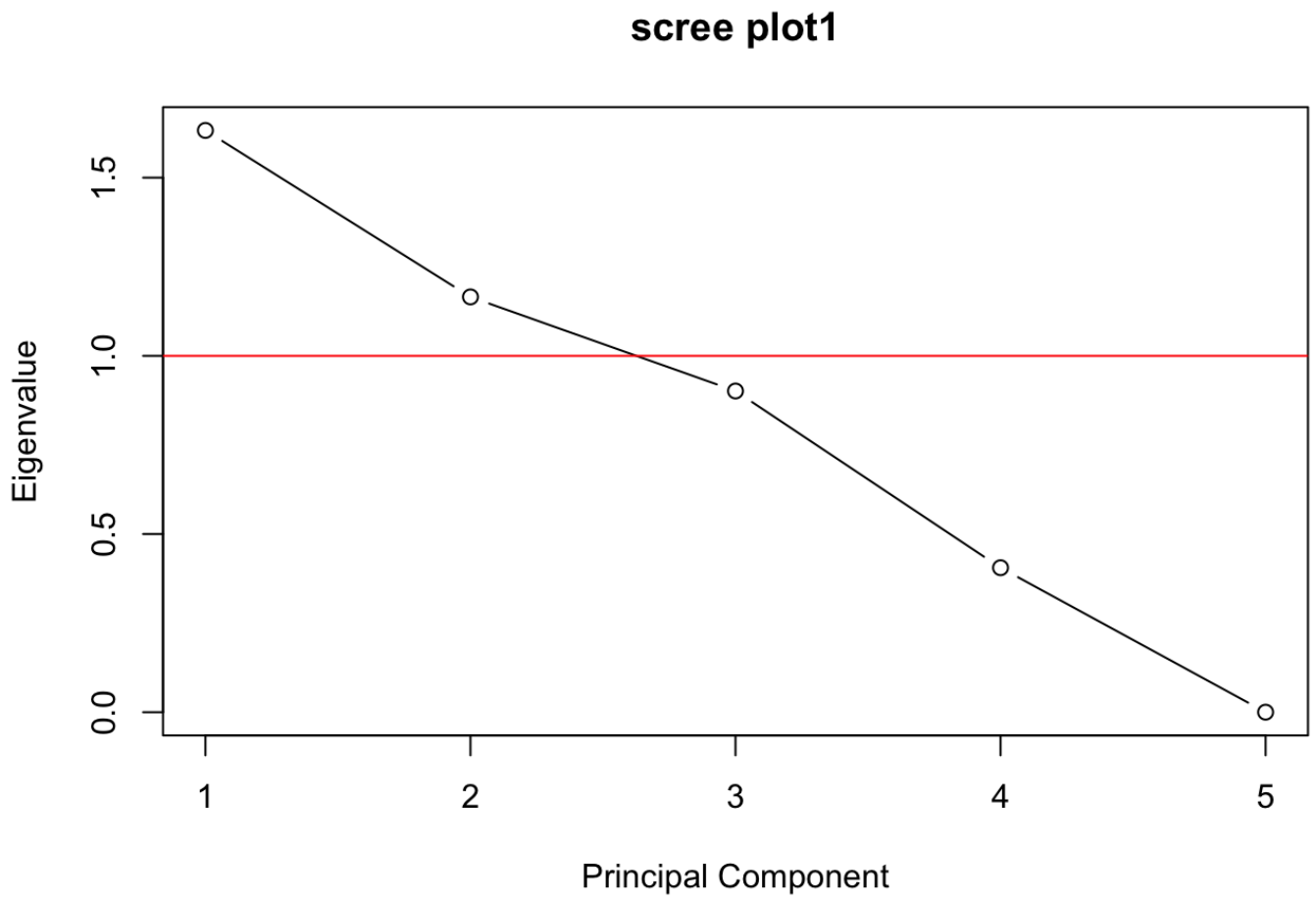
```
##        [,1]   [,2]   [,3]   [,4] [,5]
## [1,] -30.49  52.96   0.34   0.92    0
## [2,]  40.58   1.89   3.17  23.79    0
## [3,] -22.33 -28.19 -40.02   1.49    0
## [4,]  37.66   3.66  -1.33 -24.89    0
## [5,] -25.43 -30.32  37.84  -1.30    0
```

Let check the scree plot.

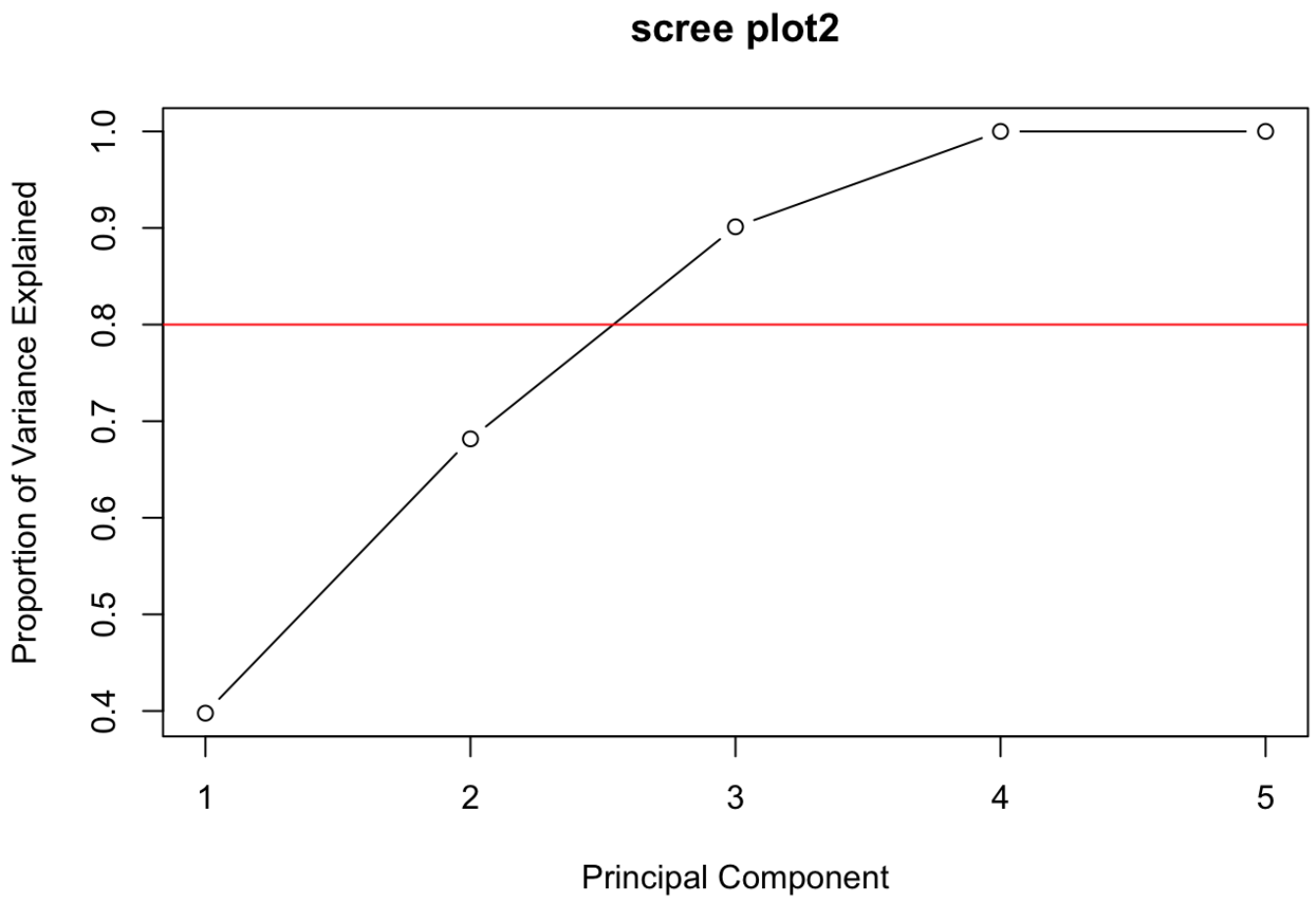
```
princ <- princomp(x = b,cor = T, scores = T)
print(princ$sdev,digits = 2)
```

```
##  Comp.1  Comp.2  Comp.3  Comp.4  Comp.5
## 1.6e+00 1.2e+00 9.0e-01 4.1e-01 1.2e-08
```

```
plot(princ$sdev, xlab = "Principal Component",ylab = "Eigenvalue",type = "b", m
ain = 'scree plot1')
abline(h=1,col='Red')
```



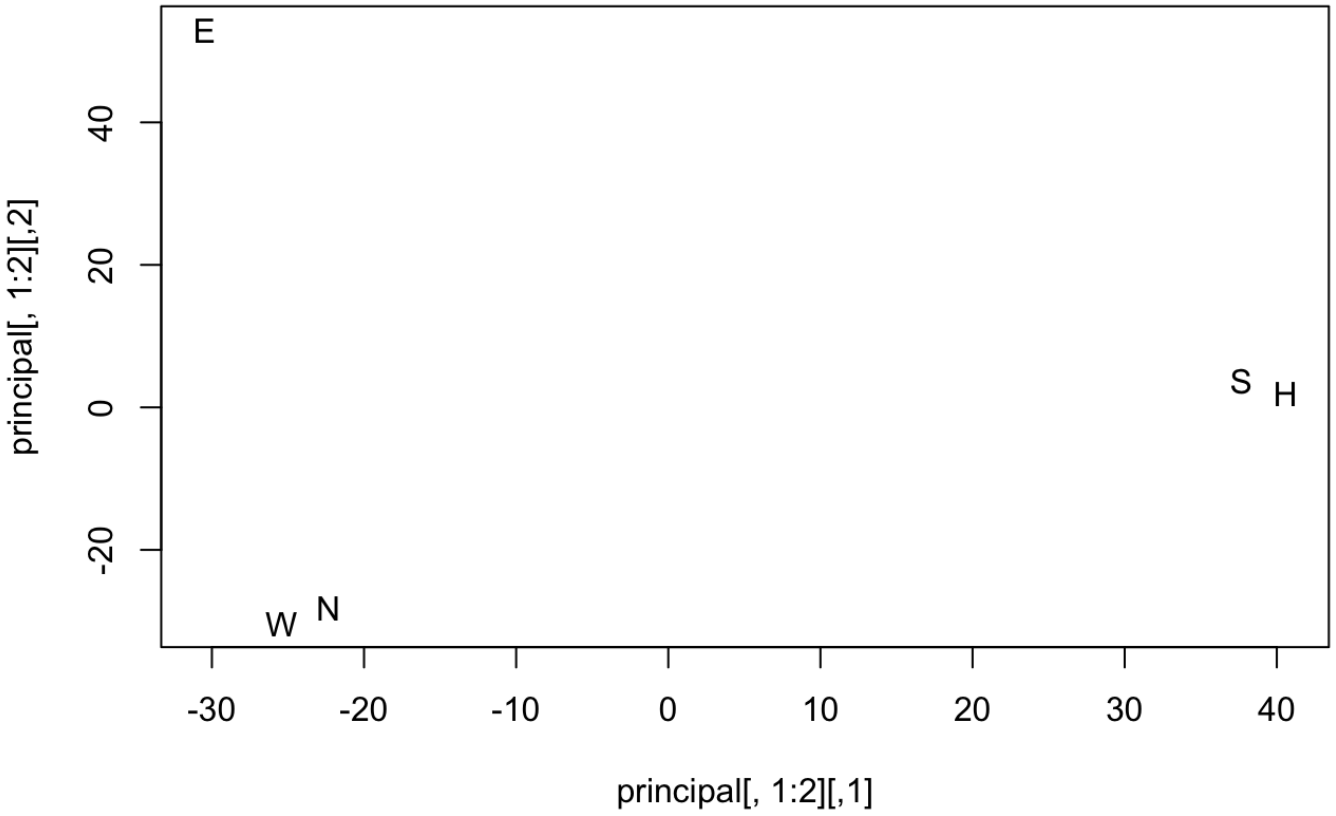

```
plot(cumsum(princ$sdev)/sum(princ$sdev), xlab = "Principal Component",ylab = "P
roportion of Variance Explained",type = "b", main = 'scree plot2')
abline(h=0.8,col='Red')
```



Thus, we only need 2 dimentsions.

8 Two-dimension expression

```
plot(principal[,1:2],pch="")
points <- c("E","H",'N',"S","W")
text(principal[,1:2],points)
```



Firstly, same to Shepard, W and N, S and H are closer like a cluster. While, E is single point. If result from Shepard rotate counterclockwisely in 90 degree, it is very similar to my result.