Homework 6

Giang Vu

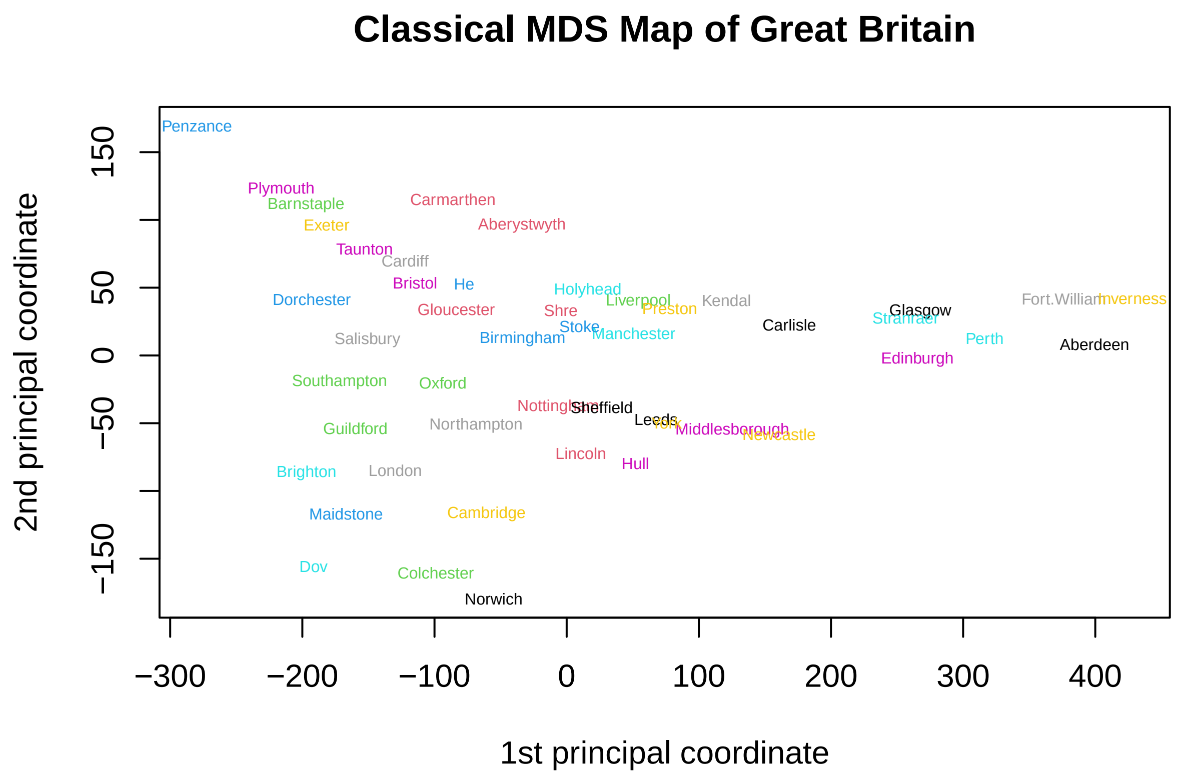
10/28/2021

Problem 6.1. Classical multidimensional scaling

1. MDS on Great Britain data

The map generated from MDS did a fairly okay job compared to the real map of Great Britain, however it's not perfect when it comes to the exact position of cities relative to each other, probably due to the limitations of 2D maps.
MDS result could generally capture the 3 main big countries in the UK, with most the cities in England grouped together on the bottom of the graph, Scotland and its towns on the right hand cluster, while the different cities of Wales are mostly on the top left.
The exact location of each city relative to each other is decent/okay compared to the real map.

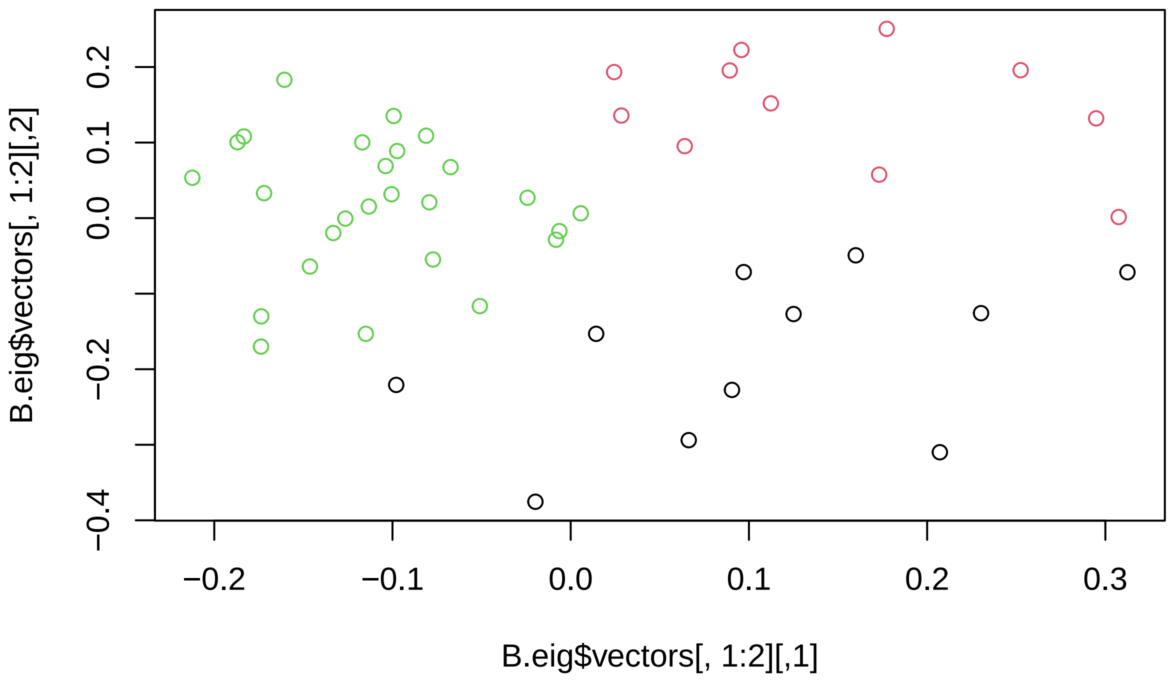
```r
gb.mds <- cmdscale(BritishTowns, k = 2, eig = T, x.ret = T, list. = T)
gb.map <- data.frame(gb.mds$points)
gb.map$towns <- row.names(gb.map)
plot(x=gb.map$X1, y = gb.map$X2,
     xlab = "1st principal coordinate",
     ylab = "2nd principal coordinate",
     main = "Classical MDS Map of Great Britain",
     cex=0)
text(x=gb.map$X1,y=gb.map$X2,
     labels=gb.map$towns,cex = 0.5,
     col = c(1:48))
```



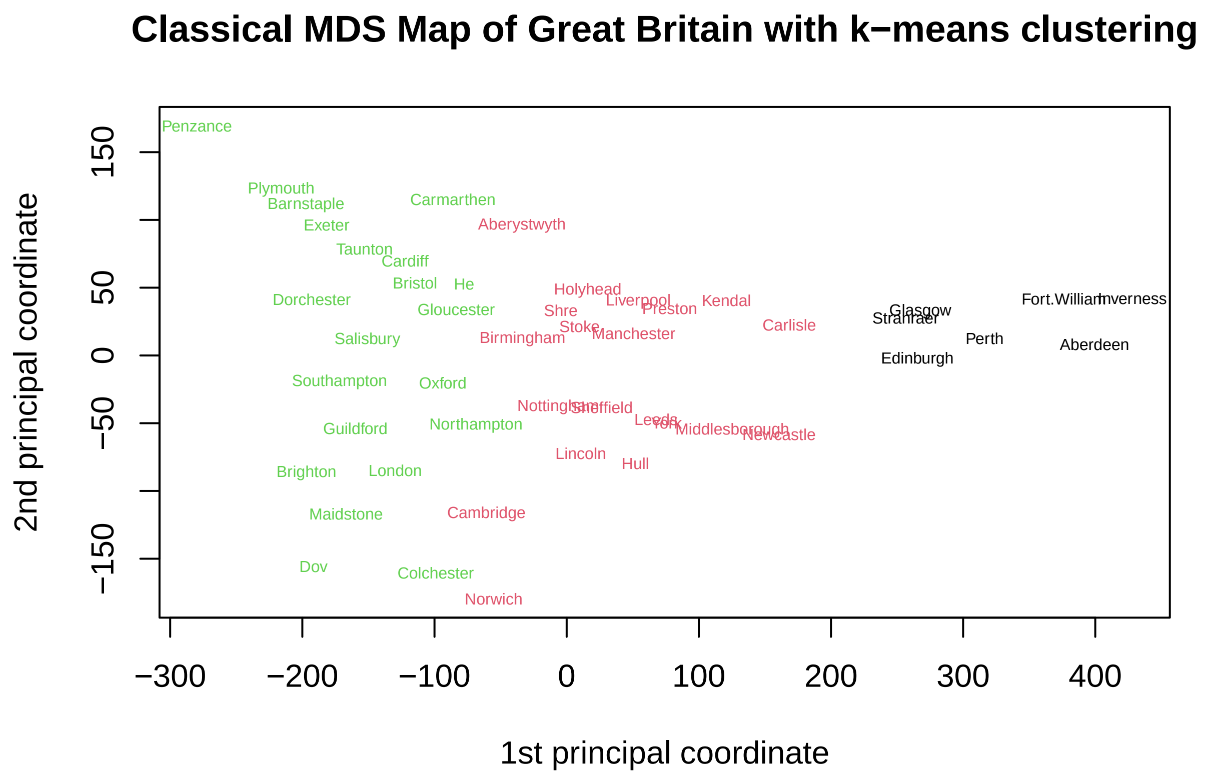
2. Clustering on Great Britain data

If we think of the proximity matrix as a weighted adjacency matrix, then we could also think of the matrix B (the doubly centered version of the proximity matrix) as the graph Laplacian, and then the results of classical MDS above gave us the first 2 eigenvectors of B, therefore we can apply clustering methods to these 2 vectors/principal coordinates as well.
I tried both k-means and Gaussian mixture modeling with a precified number of clusters $k = 3$ (3 countries in the UK). And my results showed that MClust method did better in capturing the 3 countries (especially Scotland) despite having a few errors between England and Wales. On the other hand, k-means did worse, as we can see from the plot, this method mixed up cities in all 3 countries.

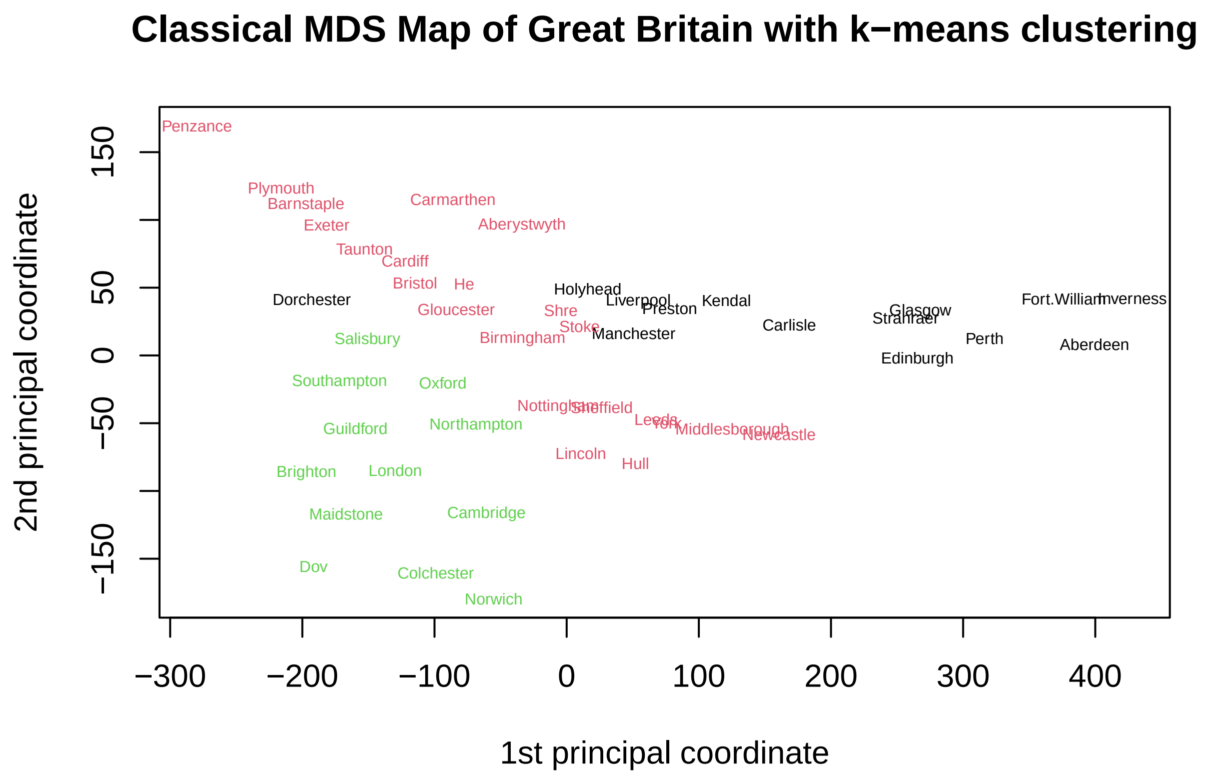
```
#extract doubly centred matrix B
B <- matrix(gb.mds$x, ncol = 48)
#treat B as Lapla and carry out LSE
B.eig <- eigen(B)
B.clust <- kmeans(B.eig$vectors[,1:2], centers = 3)
plot(B.eig$vectors[,1:2], col=B.clust$cluster)
```



```
#or cluster the matrix Y given by CMDS??
cluster <- kmeans(gb.map[,1:2], centers = 3) # kmeans
plot(x=gb.map$X1, y = gb.map$X2,
     xlab = "1st principal coordinate",
     ylab = "2nd principal coordinate",
     main = "Classical MDS Map of Great Britain with k-means clustering",
     cex=0)
text(x=gb.map$X1,y=gb.map$X2,
     labels=gb.map$towns,cex = 0.5,
     col = cluster$cluster)
```



```
cluster <- Mclust(gb.map[,1:2], G=3, verbose = F) #gaussian mix
plot(x=gb.map$X1, y = gb.map$X2,
     xlab = "1st principal coordinate",
     ylab = "2nd principal coordinate",
     main = "Classical MDS Map of Great Britain with k-means clustering",
     cex=0)
text(x=gb.map$X1,y=gb.map$X2,
     labels=gb.map$towns,cex = 0.5,
     col = cluster$classification)
```



Problem 6.2. MDS meets stochastic block models

1. Simulation study

I generated a SBM model with $n = 60$, $d = K = 2$ like in the code below. I then obtained the spectral embeddings of the adjacency matrix (2 leading eigenvectors stored in *X_ase*) and that of graph Laplacian (2 smallest eigenvectors stored in *X_lse*). The proximity matrices *delta_ase* and *delta_lse* were constructed accordingly before I applied classical MDS on them. Principal coordinates from MDS were stored in *Y_ase1*, *Y_ase2*, *Y_lse1*, *Y_lse2* for ASE and LSE, when $t = 1$ and $t = d = 2$, respectively.
I printed a few values from the X's and Y's together to compare. The first principal coordinate from MDS whenn $t = d = 2$ is the same as the MDS result when $t = 1$.
And we can see here that for ASE, the second largest eigenvector (second column in *X_ase*) is extremely close to the first principal coordinate from MDS (first column in *Y_ase2* and *Y_ase1*), with a sign flip.
For the LSE method, we can see the same phenomenon, the second smallest eigenvector (second column in *X_lse*) is actually the exact first principal coordinate from MDS (first column in *Y_lse2* and *Y_lse1*).
This phenomenon I observed from my simulation study here falls in line with my thought process in Problem 6.1, part 2 above. We can find the connection between MDS and SBM if we treat the proximity matrices (*delta_ase* and *delta_lse*) like weighted adjacency matrices, and then we will have resulting principal coordinates (*Y_ase1*, *Y_ase2*, *Y_lse1*, *Y_lse2*) from MDS to somewhat coincide with the spectral embeddings of the adjacency and Laplacian matrices (*X_ase*, *X_lse*) from SBM.

```r
# SBM model
n <- 60
rho <- log(n)/n
```

```r
aa <- 4.5
bb <- 0.25
cc <- 4.5
block.sizes = c(1/2, 1/2)*n

# above - specified inputs
########################################
# below - automated inputs/outputs

pref.matrix <- rho*rbind(c(aa, bb),
                         c(bb, cc)) #this is matrix B - prob of edge between group i and j

set.seed(1234)

my.graph <- sample_sbm(n, pref.matrix, block.sizes, directed = FALSE, loops = TRUE)


#2 leading eigenvals and vectors of adjacency matrix
adj_ed <- embed_adjacency_matrix(my.graph, 2, which="lm", scaled = F)
X_ase <- adj_ed$X

#2 smallest eigenvals and vectors of laplacian
lap_ed <- embed_laplacian_matrix(my.graph, 2, which = "sa", scaled = F)
X_lse <- lap_ed$X

#construct distance matrices
delta_ase <- dist(X_ase)
delta_lse <- dist(X_lse)

#MDS with t=1
mds_ase1 <-cmdscale(delta_ase, k = 1, eig = T, x.ret = T, list. = T)
mds_lse1 <-cmdscale(delta_lse, k = 1, eig = T, x.ret = T, list. = T)
Y_ase1 <- mds_ase1$points
Y_lse1 <- mds_lse1$points

#MDS with t=d=2
mds_ase2 <-cmdscale(delta_ase, k = 2, eig = T, x.ret = T, list. = T)
mds_lse2 <-cmdscale(delta_lse, k = 2, eig = T, x.ret = T, list. = T)
Y_ase2 <- mds_ase2$points
Y_lse2 <- mds_lse2$points

#compare Xs vs Ys
#for ASE
head(X_ase)
```

```
##             [,1]       [,2]
## [1,] -0.07401636 0.04642548
## [2,] -0.16294078 0.19817880
## [3,] -0.12088872 0.14302788
## [4,] -0.14091432 0.15505380
## [5,] -0.11441048 0.14602928
## [6,] -0.11136239 0.14567406
```

```
head(Y_ase2)
```

```
##              [,1]         [,2]
## [1,] -0.04315541  0.048100221
## [2,] -0.19088883 -0.047353747
## [3,] -0.13762312 -0.002937943
## [4,] -0.14876481 -0.023468647
## [5,] -0.14090402  0.003403329
## [6,] -0.14068199  0.006464011
```

```
head(Y_ase1)
```

```
##              [,1]
## [1,] -0.04315541
## [2,] -0.19088883
## [3,] -0.13762312
## [4,] -0.14876481
## [5,] -0.14090402
## [6,] -0.14068199
```

```
#for LSE
head(X_lse)
```

```
##             [,1]        [,2]
## [1,] -0.1290994 -0.08663586
## [2,] -0.1290994 -0.13953973
## [3,] -0.1290994 -0.13832210
## [4,] -0.1290994 -0.12849346
## [5,] -0.1290994 -0.14808933
## [6,] -0.1290994 -0.15540259
```

```
head(Y_lse2)
```

```
##              [,1]                 [,2]
## [1,] -0.08663586  0.0000000031400664
## [2,] -0.13953973  0.0000000031657830
## [3,] -0.13832210  0.0000000014477653
## [4,] -0.12849346 -0.0000000008787821
## [5,] -0.14808933  0.0000000009328740
## [6,] -0.15540259 -0.0000000001954202
```

```
head(Y_lse1)
```

```
##              [,1]
## [1,] -0.08663586
## [2,] -0.13953973
## [3,] -0.13832210
## [4,] -0.12849346
## [5,] -0.14808933
## [6,] -0.15540259
```

2. Population-level theory

First I constructed *theta* which is the node membership vector, $n = 60$ rows, and $K = 2$ columns, for each row/node, the ith column is 0 if that node isn't in the ith cluster, and 1 otherwise.
Then the population-level equivalent of the adjacency matrix, denoted P, was calculated. I then repeated what I did in previous part (simulation study) and compared the ASE and LSE embeddings versus the principal coordinates given by classical MDS.
Again, we can see here that for ASE, the second largest eigenvector (second column in X_ase) is extremely close to the first principal coordinate from MDS (first column in Y_ase2 and Y_ase1), with a sign flip.
However, for the LSE method, the similarity isn't as pronouncedd as before, the second smallest eigenvector (second column in X_lse) is not very close to the first principal coordinate from MDS (first column in Y_lse2 and Y_lse1) anymore.

```
#construct node membership vector
theta <- matrix(c(rep(1,30), rep(0,30), rep(0,30), rep(2,30)), nrow = 60, ncol = 2, byrow = F)
#population level adjacency
P <- theta%*%pref.matrix%*%t(theta)
P.graph <- graph.adjacency(P)

#2 leading eigenvals and vectors of adjacency matrix
adj_ed <- embed_adjacency_matrix(P.graph, 2, which="lm", scaled = F)
X_ase <- adj_ed$X

#2 smallest eigenvals and vectors of laplacian
lap_ed <- embed_laplacian_matrix(P.graph, 2, which = "sa", scaled = F)
X_lse <- lap_ed$X

#construct distance matrices
delta_ase <- dist(X_ase)
delta_lse <- dist(X_lse)

#MDS with t=1
mds_ase1 <-cmdscale(delta_ase, k = 1, eig = T, x.ret = T, list. = T)
mds_lse1 <-cmdscale(delta_lse, k = 1, eig = T, x.ret = T, list. = T)
Y_ase1 <- mds_ase1$points
Y_lse1 <- mds_lse1$points

#MDS with t=d=2
mds_ase2 <-cmdscale(delta_ase, k = 2, eig = T, x.ret = T, list. = T)
mds_lse2 <-cmdscale(delta_lse, k = 2, eig = T, x.ret = T, list. = T)
Y_ase2 <- mds_ase2$points
Y_lse2 <- mds_lse2$points

#compare Xs vs Ys
#for ASE
head(X_ase)
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
## [3,]    0    0
## [4,]    0    0
## [5,]    0    0
## [6,]    0    0
```

```
head(Y_ase2)
```

```
##                              [,1]        [,2]
## [1,] -0.00000000000000003095483 -0.09128709
## [2,] -0.00000000000000015516586 -0.09128709
## [3,] -0.00000000000000001760512 -0.09128709
## [4,]  0.00000000000000013331582 -0.09128709
## [5,] -0.00000000000000003148291 -0.09128709
## [6,] -0.00000000000000001066622 -0.09128709
```

```
head(Y_ase1)
```

```
##                              [,1]
## [1,] -0.00000000000000003095483
## [2,] -0.00000000000000015516586
## [3,] -0.00000000000000001760512
## [4,]  0.00000000000000013331582
## [5,] -0.00000000000000003148291
## [6,] -0.00000000000000001066622
```

```
#for LSE
head(X_lse)
```

```
##             [,1]        [,2]
## [1,]  0.07507346 -0.20092649
## [2,]  0.09954188  0.18135481
## [3,] -0.08345180 -0.04943113
## [4,]  0.01527868 -0.04059483
## [5,] -0.21854449  0.06256467
## [6,]  0.19785024 -0.10055043
```

```
head(Y_lse2)
```

```
##             [,1]        [,2]
## [1,] -0.06083961  0.22657549
## [2,]  0.18858822 -0.06415368
## [3,] -0.09640247  0.01020455
## [4,] -0.01220326  0.06251403
## [5,] -0.13699499 -0.16051552
## [6,]  0.09761827  0.22020449
```

```
head(Y_lse1)
```

```
##             [,1]
## [1,] -0.06083961
## [2,]  0.18858822
## [3,] -0.09640247
## [4,] -0.01220326
## [5,] -0.13699499
## [6,]  0.09761827
```