HUDM6122 Homework_03

Chenguang Pan

2023-02-23

0.1 Ex 4.1

Consider 51 objects O1, . , O51 assumed to be arranged along a straight line with the jth object being located at a point with coordinate j. Define the similarity sij between object i and object j as... and then apply classical multidimensional scaling to the resulting dissimilarity matrix. Explain the shape of the derived two-dimensional solution.

MY SOLUTION:
First, I define the function of dissimilarities δ_{ij} as follows, where the input paramter X is a n*1 matrix that contains all the coordination of 51 objects.

```
> D_dis <- function(X) {
+   n <- dim(X)[1] # to have the length of any input n*1 matrix
+   S <- matrix(0, n, n) # to make a n*n empty matrix
+   D <- matrix(0, n, n) # to make a n*n empty matrix
+   # to make the similarity matrix by conditions
+   for (i in c(1:n)) {
+     for (j in c(1:n)){
+       if (i == j){S[i,j] <- 9}
+       else if (abs(i-j) >= 1 & abs(i-j) <= 3){S[i,j] <- 8}
+       else if (abs(i-j) >= 4 & abs(i-j) <= 6){S[i,j] <- 7}
+       else if (abs(i-j) >= 7 & abs(i-j) <= 9){S[i,j] <- 6}
+       else if (abs(i-j) >= 10 & abs(i-j) <= 12){S[i,j] <- 5}
+       else if (abs(i-j) >= 13 & abs(i-j) <= 15){S[i,j] <- 4}
+       else if (abs(i-j) >= 16 & abs(i-j) <= 18){S[i,j] <- 3}
+       else if (abs(i-j) >= 19 & abs(i-j) <= 21){S[i,j] <- 2}
+       else if (abs(i-j) >= 22 & abs(i-j) <= 24){S[i,j] <- 1}
+       else if (abs(i-j) >= 25){S[i,j] <- 0}
+     }
+   } # similarity matrix finished!
+   # using the elements in the Similarity matrix to generate Dissimilarities Matrix
+   for (i in c(1:n)) {
+     for (j in c(1:n)) {
+       D[i,j] <- sqrt(S[i,i] + S[j,j] - 2*S[i,j])
+     }
+   } # dissimilarity matrix finished!
+   return(D)
+ }
```

Next, I randomly generate a n*1 matrix with 51 integers by using `sample()` function. And plug this vector to the dissimilarity function above.

```
> obs_ <- function(n, # the number of objects you wanna generate
+                      replace=TRUE, # randomly sample with replacement or not
+                      start =1,  # range from start to end; 1-200 by default
+                      end =200){
+   number_vec <- sample(c(start:end),n,replace=replace)
+   # change the number array to matrix
+   return(matrix(number_vec,n,1))
+ }
```

The functions above looks good. I try to generate 51 observations and plug them into the dissimilarity matrix function to get the required D matrix.

```
> set.seed(2023)
> # generate 51 observations
> observations <- obs_(51)
> # plug the n*1 matrix into the dissimilarity martix function
> D <- D_dis(observations)
> # select a part of the dissimilarity matrix
> D[1:5,1:5]
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 0.000000 1.414214 1.414214 1.414214 2.000000
[2,] 1.414214 0.000000 1.414214 1.414214 1.414214
[3,] 1.414214 1.414214 0.000000 1.414214 1.414214
[4,] 1.414214 1.414214 1.414214 0.000000 1.414214
[5,] 2.000000 1.414214 1.414214 1.414214 0.000000
```

This dissimilarity matrix looks good. Then I run the classical multidimensional scaling to this resulting matrix. Note, this is a non-Euclidean case. Some of the eigenvalue may be negative.

```
> d_mds <- cmdscale(D, k=50, eig = T)
> lam <- d_mds$eig
> lam
 [1]  1.260857e+02  6.593531e+01  1.817103e+01  7.820841e+00  7.610276e+00
 [6]  7.378184e+00  7.018378e+00  5.283354e+00  3.436749e+00  3.096003e+00
[11]  3.089512e+00  2.697496e+00  1.939351e+00  1.457605e+00  1.427548e+00
[16]  1.401274e+00  1.341983e+00  1.162903e+00  9.675241e-01  9.495632e-01
[21]  9.380593e-01  8.766031e-01  7.925229e-01  7.664407e-01  7.041554e-01
[26]  6.921047e-01  5.994198e-01  5.448786e-01  4.750354e-01  4.738073e-01
[31]  4.017636e-01  3.996796e-01  3.607046e-01  3.046375e-01  2.892578e-01
[36]  2.756293e-01  2.664975e-01  2.588366e-01  2.539959e-01  2.509620e-01
[41]  1.209368e-01  9.981512e-02  1.776357e-15 -9.449958e-02 -1.060236e-01
[46] -3.067296e-01 -3.298370e-01 -6.738433e-01 -7.105516e-01 -2.002901e+00
[51] -2.074323e+00
> cumsum(abs(lam))/sum(abs(lam))
 [1] 0.4428488 0.6744323 0.7382541 0.7657231 0.7924526 0.8183668 0.8430174
 [8] 0.8615740 0.8736449 0.8845189 0.8953701 0.9048445 0.9116561 0.9167756
[15] 0.9217895 0.9267112 0.9314246 0.9355091 0.9389073 0.9422424 0.9455372
[22] 0.9486160 0.9513996 0.9540916 0.9565647 0.9589956 0.9611009 0.9630147
[29] 0.9646832 0.9663473 0.9677584 0.9691622 0.9704291 0.9714991 0.9725150
[36] 0.9734831 0.9744191 0.9753283 0.9762204 0.9771018 0.9775266 0.9778771
[43] 0.9778771 0.9782091 0.9785814 0.9796588 0.9808172 0.9831840 0.9856796
[50] 0.9927144 1.0000000
> cumsum(abs(lam^2))/sum(abs(lam^2))
```

```
 [1] 0.7608520 0.9689196 0.9847222 0.9876495 0.9904214 0.9930267 0.9953842
 [8] 0.9967201 0.9972854 0.9977441 0.9982010 0.9985492 0.9987292 0.9988309
[15] 0.9989284 0.9990224 0.9991086 0.9991733 0.9992181 0.9992613 0.9993034
[22] 0.9993402 0.9993702 0.9993983 0.9994221 0.9994450 0.9994622 0.9994764
[29] 0.9994872 0.9994979 0.9995057 0.9995133 0.9995195 0.9995240 0.9995280
[36] 0.9995316 0.9995350 0.9995382 0.9995413 0.9995443 0.9995450 0.9995455
[43] 0.9995455 0.9995459 0.9995465 0.9995510 0.9995562 0.9995779 0.9996021
[50] 0.9997941 1.0000000
```