Supplementary Material Section A: Community Detection Algorithms

2025-06-10

```
library(MASS)
### MASS::mvrnorm(), generate multivariate normal samples
library(matlib)
### matlib::symMat(), create a symmetric matrix from a vector
library(ICONS)
### ICONS::dense() and ICONS::plotMatrix(), extract subnetwork and heatmap plot

##############################
### Loading the functions ###
##############################

BLOCK_HADAMARD_PRODUCT <- function(A, B, p_vec){
  K <- length(p_vec)
  COL_temp <- c()
  for(k in 1 : K){
    ROW_temp <- c()
    for(kp in 1 : K){
      if(k == kp){
        ROW_temp <- cbind(ROW_temp, A[k, k] * diag(p_vec[k]) + B[k, k] * matrix(1, p_vec[k], p_vec[k]))
      } else{
        ROW_temp <- cbind(ROW_temp, B[k, kp] * matrix(1, p_vec[k], p_vec[kp]))
      }
    }
    COL_temp <- rbind(COL_temp, ROW_temp)
  }
  M <- COL_temp
  return(M)
}
```

We include a toy simulation study to illustrate the implementation of the K-medoids and SICERS algorithms.

Step 1 We first construct a population covariance matrix $\Sigma_{200\times 200}$ with $S = 200$ features, comprising $G = 3$ communities and $L_0 = 100$ singleton features, as illustrated in Figure A. The community sizes are $L_1 = L_2 = 30$ and $L_3 = 40$. For singleton features, the diagonal entries of the covariance matrix are set to 1.

```
n <- 50
### sample size
l_vec <- c(30, 30, 40)
### the known partition-size vector
l0 <- 100
### the number of singletons
```

```
R <- sum(l_vec)
S <- R + l0

A_0 <- diag(c(0.1, 0.2, 0.3))
B_0 <- symMat(c(0.9, - 0.430920, -0.353572, 0.8, - 0.333474, 0.7))
Sigma_0 <- matrix(0, S, S)
Sigma_0[1 : R, 1 : R] <- BLOCK_HADAMARD_PRODUCT(A_0, B_0, l_vec)
Sigma_0[(R + 1) : S, (R + 1) : S] <- diag(l0)
```

```
plotMatrix(z = Sigma_0, cex.axis = 1.3, cex.lab = 1.3, save.image = F)
```

```
### heatmap of the true covariance matrix based on covariates 1 to S
```

Step 2 Next, we generate a multivariate normal dataset $\mathbf{X}^*_{50\times 200}$ from $N(\mathbf{0}_{200\times 1}, \Sigma)$ with a sample size of $n = 50$. The sample covariance matrix based on $\mathbf{X}^*_{50\times 200}$ is shown in Figure B. We then randomly permute the order of the 200 features using the `sample()` function in R, resulting in a permuted dataset denoted by $\widetilde{\mathbf{X}}^*_{50\times 200}$. The corresponding sample covariance matrix, based on $\widetilde{\mathbf{X}}^*_{50\times 200}$, is shown in Figure C, which appears markedly different from the heatmap A due to the feature permutation. This representation in the heatmap C probably resembles the patterns often observed in practice, where the true community structure is obscured.

```
set.seed(20251)
data_mat <- mvrnorm(n = n, mu = rep(0, S), Sigma = Sigma_0)
### n by S
var_name <- paste("Var", seq(S), sep = "_")
colnames(data_mat) <- var_name
set.seed(20252)
name_order <- sample.int(n = S, size = S, replace = FALSE)
data_perm_mat <- data_mat[, name_order]

S_mat <- cov(data_mat)
### S by S
S_perm_mat <- cov(data_perm_mat)
### S by S
```

```
plotMatrix(z = S_mat, cex.axis = 1.3, cex.lab = 1.3, save.image = F)
```

```
### heatmap of the sample covariance matrix based on covariates 1 to S
```

```
plotMatrix(z = S_perm_mat, cex.axis = 1.3, cex.lab = 1.3, save.image = F)
```

```
### heatmap of the sample covariance matrix based permuted covariates
```

Step 3 Our objective is to recover the underlying community and singleton structure from the permuted dataset $\widetilde{\mathbf{X}}^*_{50\times 200}$ and its sample covariance heatmap C. To this end, we apply two methods: the K-medoids and SICERS methods.

For the K-medoids method, we use the built-in `kmeans()` function in R on the transposed dataset $\widetilde{\mathbf{X}}^{*,\top}$, assuming the number of clusters (communities plus singletons) is known to be 4. A pre-determined number

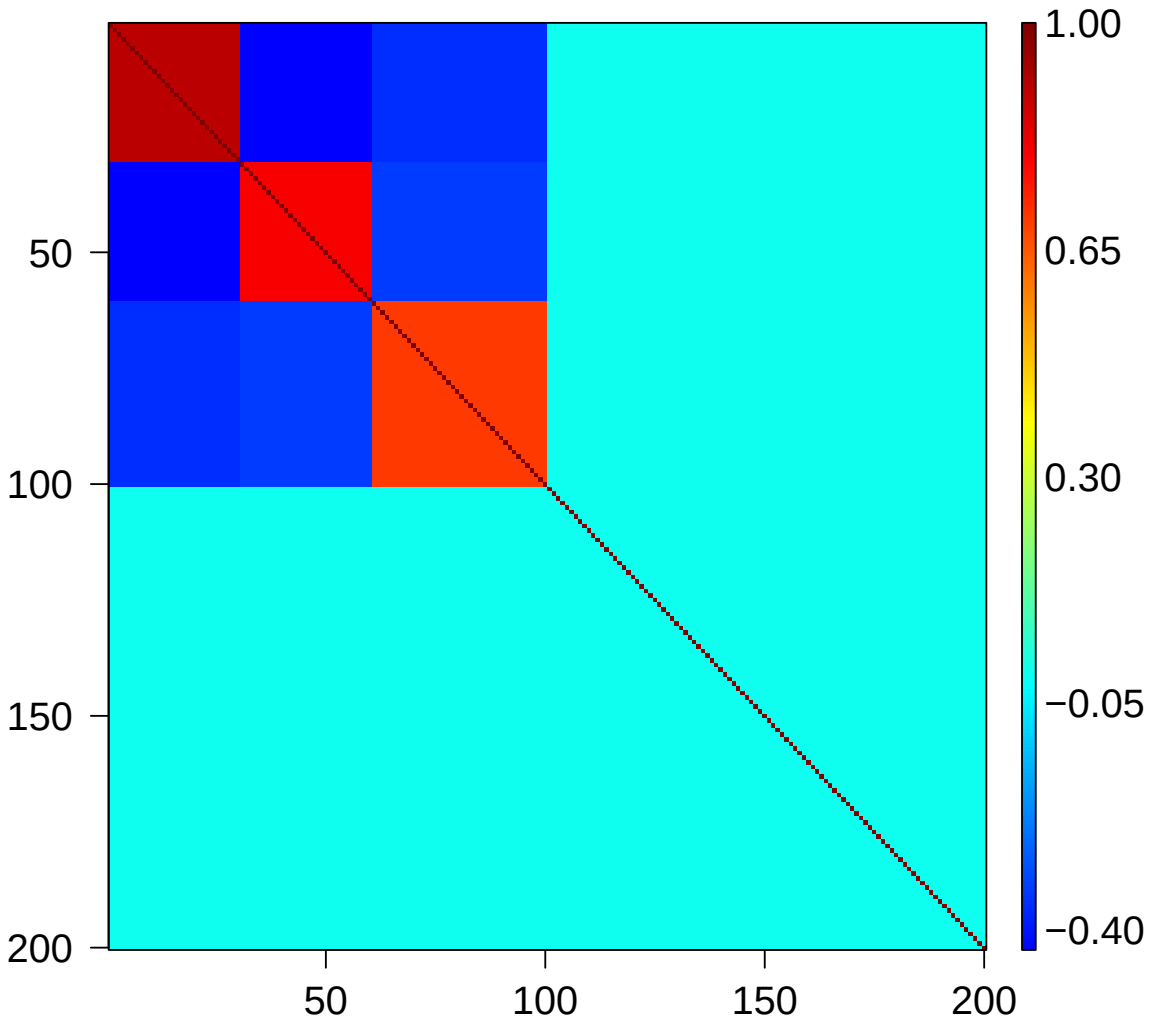

Figure 1: Figure A: the heatmap of the population covariance matrix (ground truth) with community sizes of $30, 30, 40$, alongside 100 singletons.

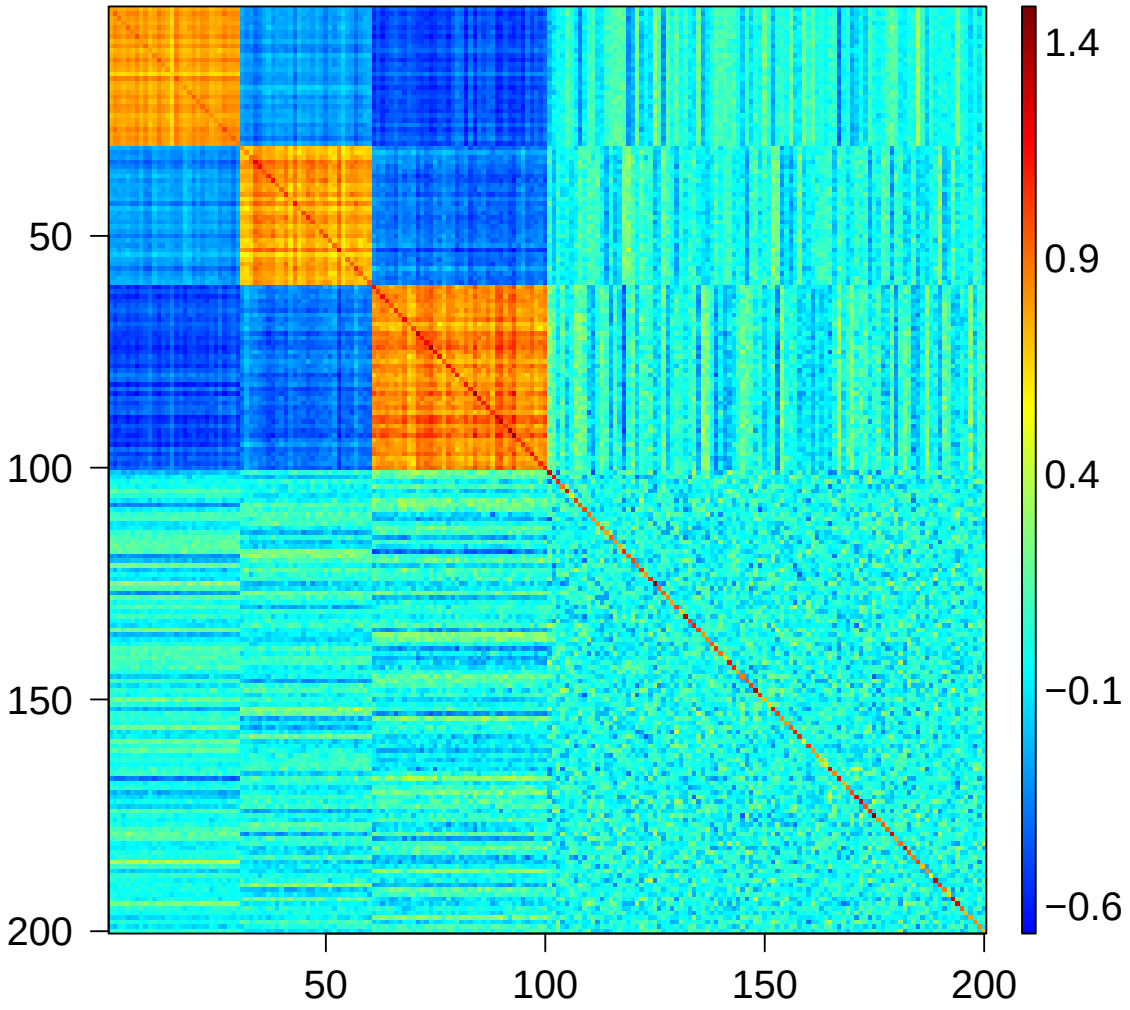

Figure 2: Figure B: the heatmap of the sample covariance matrix, based on features 1 to 200 and a sample size of 50.

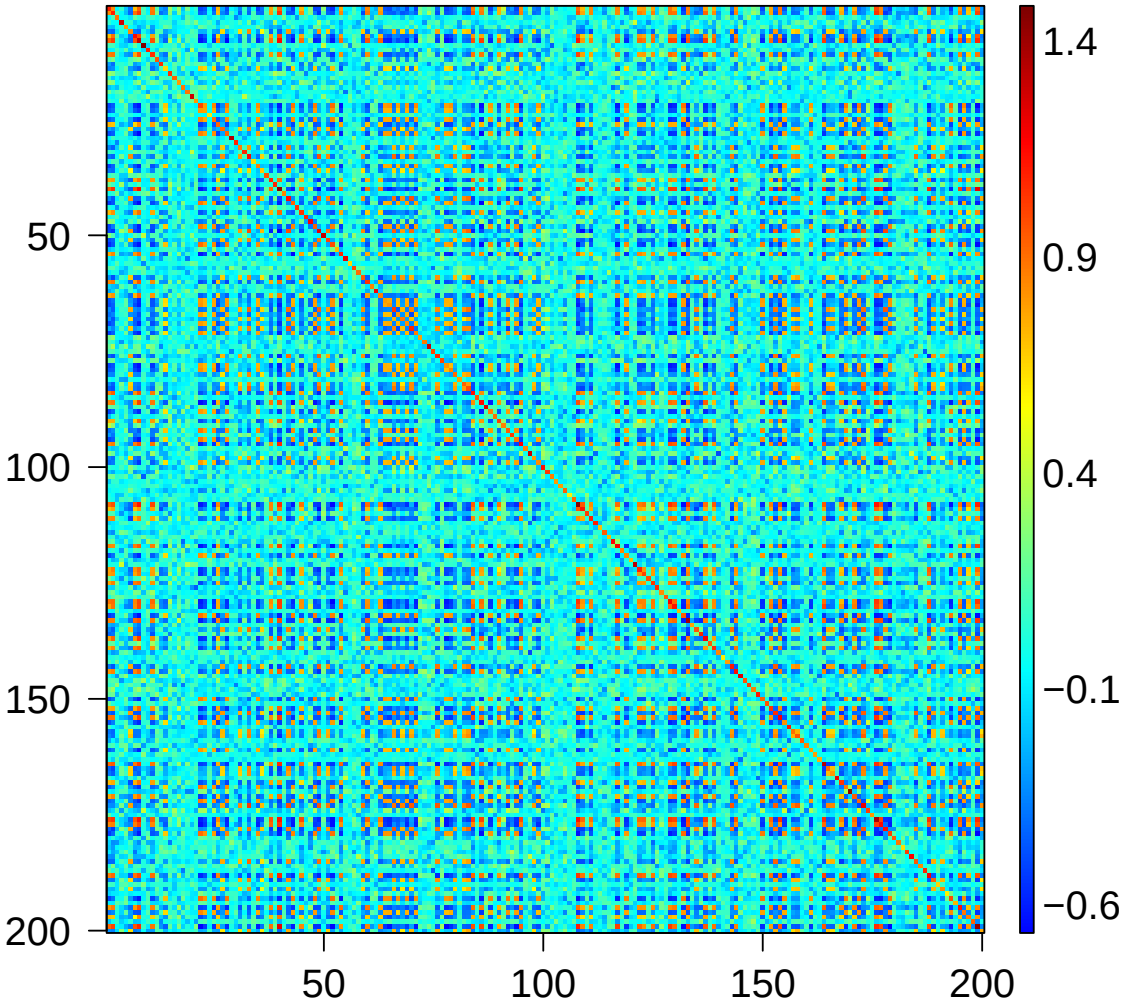

Figure 3: Figure C: the heatmap of the sample covariance matrix, based on the same dataset as in B, but with randomly permuted features.

of communities must be given. Here, we suppose we know this number, i.e., 4, due to three communities and the singleton.

```
### applying k-mean functions to data ###
set.seed(20253)
result_kmeans <- kmeans(x = t(data_perm_mat), centers = 4)
table(result_kmeans$cluster)
```

```
##
##   1   2   3   4
##  40  30 100  30
```

```
data_kmeans <- cbind(
    data_perm_mat[, result_kmeans$cluster == 1],
    data_perm_mat[, result_kmeans$cluster == 2],
    data_perm_mat[, result_kmeans$cluster == 4],
    data_perm_mat[, result_kmeans$cluster == 3])
S_mat_est_kmeans <- cov(data_kmeans)
```

```
plotMatrix(z = S_mat_est_kmeans, cex.axis = 1.3, cex.lab = 1.3, save.image = F)
```

```
### heatmap of the sample covariance matrix based permuted covariates
```

The results of the K-medoids and SICERS methods are shown in Figures D and E, respectively. Both heatmaps D and E closely resemble the original structure in Figure B, indicating successful recovery of the underlying community structure.

```
### applying ICON functions to data ###
set.seed(20254)
result_ICONS <- dense(S_perm_mat)
S_mat_est_ICONS <- result_ICONS$W_dense
### note that the diagonals are zero
diag(S_mat_est_ICONS) <- diag(S_perm_mat)[result_ICONS$Clist]
```

```
plotMatrix(z = S_mat_est_ICONS, cex.axis = 1.3, cex.lab = 1.3, save.image = F)
```

```
### heatmap of the sample covariance matrix based permuted covariates
```

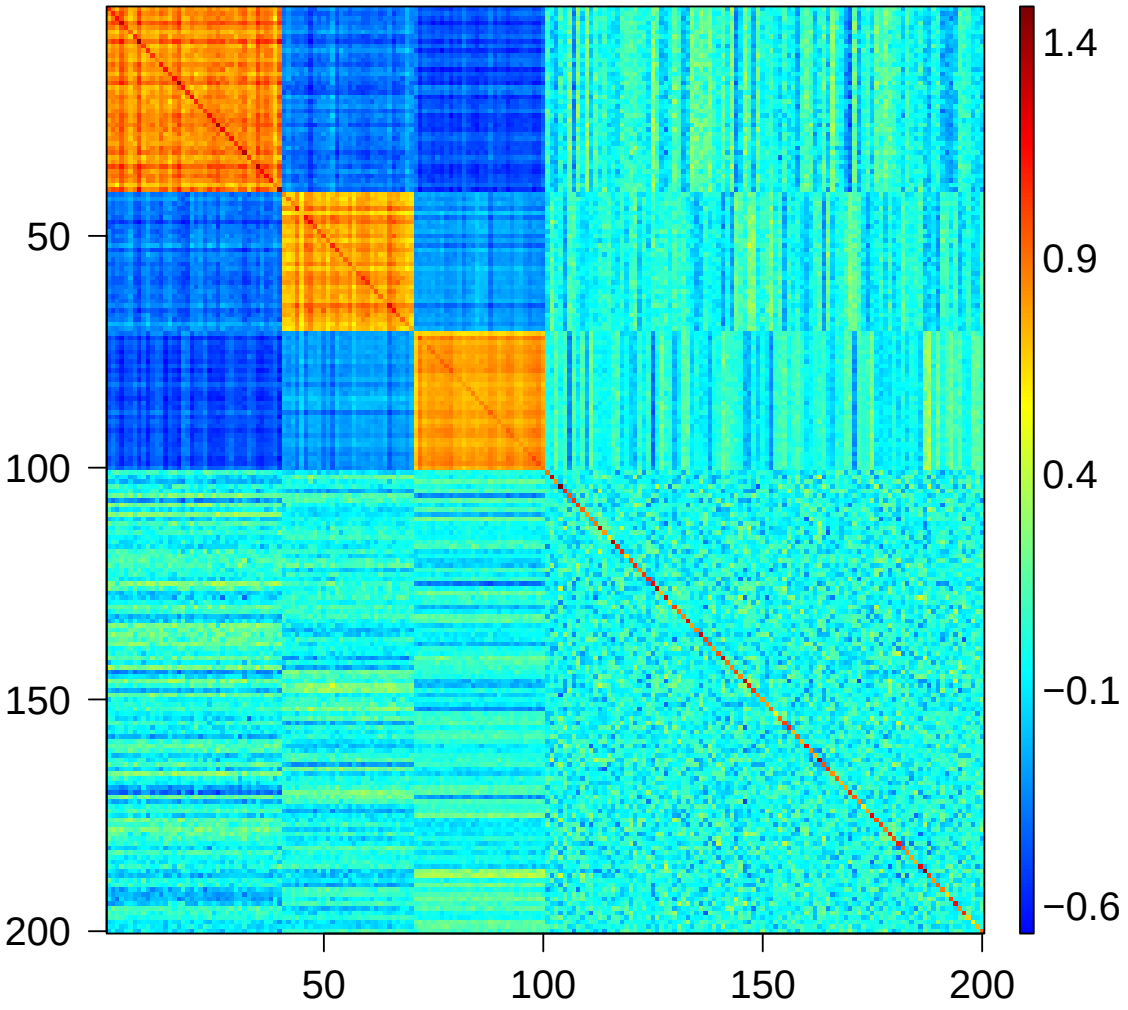

Figure 4: Figure D: the heatmap of the sample covariance matrix, based on the same dataset as in C, using the K-medoids method.

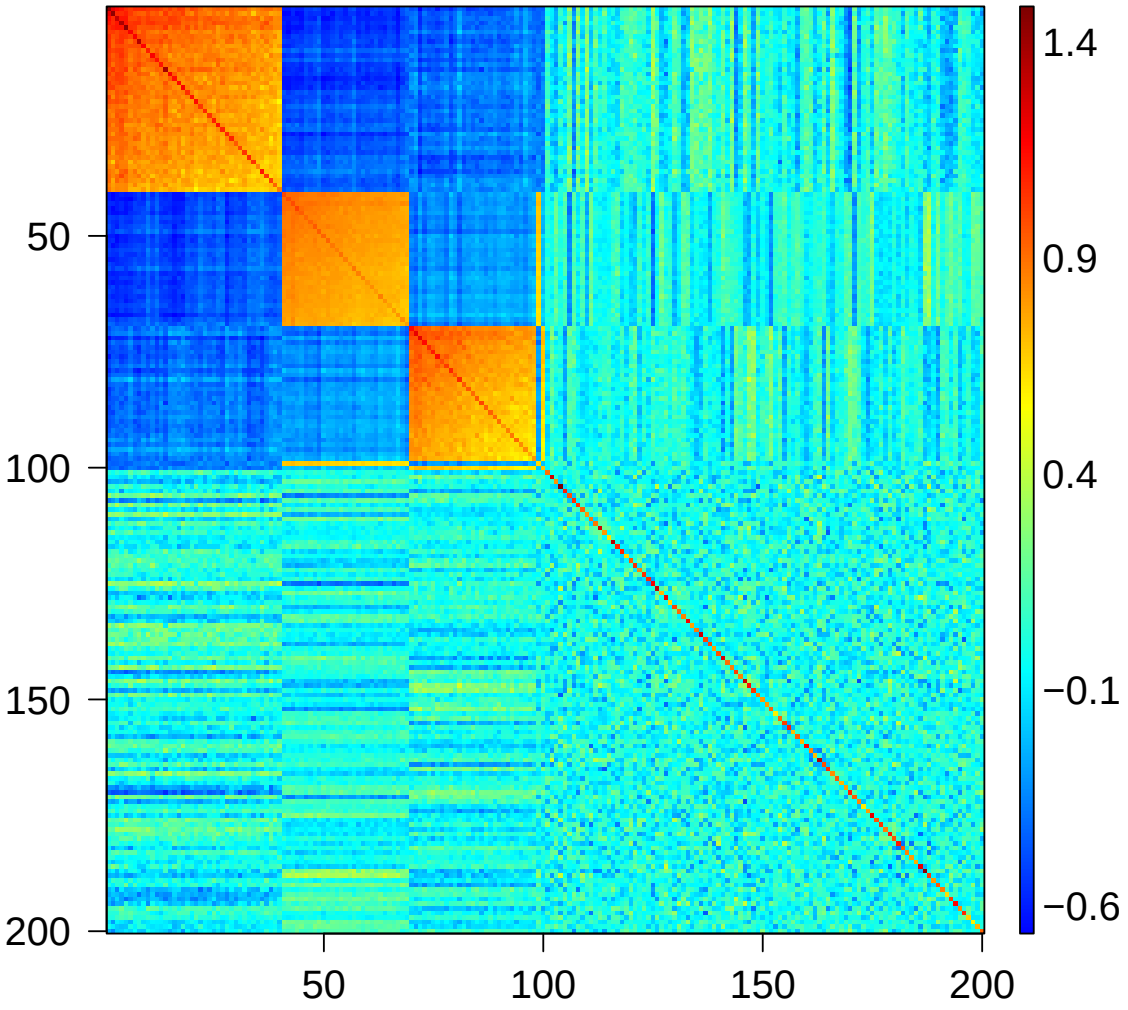

Figure 5: Figure E: the heatmap of the sample covariance matrix, based on the same dataset as in C, using the SICERS method.