

Figure 1 (left plot) Simulation Code

Overview

This doc shows reproducible simulation code to generate the Figure 1 (left plot) result in manuscript *Community Detection with Heterogeneous Block Covariance Model*. Each value in Figure 1 is the average of 100 simulation results, which was executed by leveraging the [George Washington University cloud computing platform](#). If the purpose is to test the code in local computing environment (i.e. personal laptop), we recommend to use few (<5) simulations to execute the code base.

Install packages

Install the HBCM package from github and other helper packages.

```
# install helper package
packages_helper <- c(
  'devtools', "parallel",
  "foreach", "doParallel",
  "kernlab", "matrixcalc"
)
install.packages(packages_helper, repos = "http://cran.us.r-project.org")

# install hbcm package
devtools::install_github("xiangli2pro/hbcm")

# load all required packages
lapply(c('hbcm', packages_helper), require, char = TRUE)

# set random seed to reproduce the result
set.seed(2022)
```

Setup simulation parameters

Figure 1 (left plot) in the manuscript experiments σ taking different values in $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For the demo purpose, we set $\sigma = 6$.

```
hsigma_value <- 6
hparam_func <- list(
  lambda_func = function(p) rep(1, p),
  sigma_func = function(p) rep(hsigma_value, p)
)
```

Other parameters do not change.

```
# data size and cluster number
centers <- 3 # number of Classes
n <- 1000 # number of Observations
p <- 1000 # number of Genes

# mean vector of normal distribution
mu <- rep(0, centers)

# class-level covariance matrix
off_diag <- 0.5
omega <- diag(rep(1, centers))
for (i in 1:centers) {
  for (j in 1:centers) {
    if (i != j) {
      omega[i, j] <- off_diag
    }
  }
}

# equally distributed class
ppi <- rep(1 / centers, centers)
labels <- sample(c(1:centers), size = p, replace = TRUE, prob = ppi)
```

Generate a list of simulation data sets. Note that the manuscript used 100 simulations, but for the test purpose and limited computing resource, we recommend to use fewer simulations.

```
# set up the number of simulation data
size <- 1
```

```
# generate data
data_list <- hbcmm::data_gen(n, p, centers, mu, omega, labels, size, hparam_func)

# save data
# save(data_list, labels, file = 'sim_data_npk.rda')
```

Estimate clusters

First use spectral clustering model to get initial estimation of the clusters, then plug in the initial estimation to HBCM model to perform the estimation.

Execute `?hbcmm::heterogbcm` to see the details of input and output of the HBCM function.

```
# get the list of simulation data
X_list <- data_list$x_list

# apply spectral clustering for each simulation data in the list
system.time(
  spec_labels <- lapply(
    X_list,
    function(x) rSpecc(abs(cor(x)), centers = centers)$Data),
  gcFirst = FALSE
)

# set up parallel environment
registerDoParallel(detectCores())

# apply HBCM for each simulation data in the list
system.time(
  hbcmm_res <- foreach(
    m = c(1:size), .errorhandling = "pass",
    .packages = c("MASS", "Matrix", "matrixcalc", "kernlab", "RSpectra")
  ) %dopar%
  hbcmm::heterogbcm(
    scale(X_list[[m]], center = TRUE, scale = FALSE), # standardize the data
    centers = centers, # set number of clusters
    tol = 1e-3, # iteration stop criteria
    iter = 100, # max iteration steps
    iter_init = 3, # iteration steps for estimation of hsigma and hlambd
    labels = spec_labels[[m]], # initial label estimation
    verbose = FALSE # whether or not to print out the iteration message
```

```

    ),
    gcFirst = FALSE
  )

# save data
# save(
#   spec_labels, hbcm_res,
#   labels,
#   file = 'sim_cluster_npk.rda'
# )

```

Result

```

# cluster estimated by spectral cluster for the first simulation data
spec_labels[[1]]

```

```

[1] 1 3 2 3 2 3 2 2 3 1 2 2 2 3 3 2 1 3 3 1 1 3 3 2 2 1 3 3 3 1 3 3 2 3 2 2 1
[38] 3 1 1 3 3 3 3 1 2 1 3 3 1 2 2 1 3 3 3 2 3 3 2 3 3 1 1 2 2 3 1 2 3 1 1 1 1
[75] 3 2 2 1 3 3 3 3 1 1 2 3 2 1 3 1 2 2 3 3 3 2 2 1 3 1 1 2 1 1 3 2 1 2 3 1 1
[112] 3 3 2 1 3 2 3 3 1 2 3 2 2 3 1 3 3 2 1 3 3 2 3 1 2 1 1 3 1 3 1 1 3 1 3 3 1
[149] 1 3 3 1 1 1 1 1 2 2 1 1 2 1 1 2 1 3 2 2 3 2 1 3 1 2 1 1 1 2 3 2 1 2 3 3 3
[186] 3 3 3 1 3 2 1 2 1 1 1 3 3 1 1 3 2 2 3 2 1 3 3 1 2 1 1 1 2 2 3 3 1 1 2 1 1
[223] 2 1 2 3 2 1 3 3 2 2 3 1 3 2 1 1 2 2 3 3 1 2 2 2 2 3 1 3 3 2 2 2 1 3 2 1 1
[260] 2 1 1 3 3 3 3 2 2 2 2 3 1 1 3 1 2 2 2 3 2 1 3 3 1 1 3 1 2 3 1 1 2 1 2 2 3
[297] 2 2 2 1 3 3 2 3 2 3 2 3 1 3 2 3 2 2 2 3 2 1 2 2 2 3 2 2 2 2 2 3 2 3 1 2 2
[334] 1 1 1 2 1 2 1 2 1 3 2 3 2 2 1 2 1 3 1 2 3 3 3 1 2 3 2 2 3 3 1 3 3 1 3 2 1
[371] 3 2 3 3 1 1 1 3 3 2 1 3 1 1 3 1 3 1 2 2 2 1 3 3 1 2 3 3 3 3 3 2 3 2 2 3 2
[408] 1 3 2 3 1 1 2 2 2 3 3 3 3 3 2 1 3 3 1 1 1 1 1 1 2 3 3 2 1 3 1 3 2 2 3 3
[445] 1 1 1 1 2 2 1 3 1 3 1 3 3 1 2 1 3 3 1 1 2 1 3 1 3 2 1 1 2 1 2 1 3 1 1 1 1
[482] 1 3 1 3 2 3 3 1 3 2 1 2 3 1 1 1 3 3 2 1 1 2 1 3 3 2 3 1 3 1 3 2 1 1 2 2 1
[519] 2 3 2 1 2 1 3 1 1 1 1 3 1 3 3 1 3 1 2 3 1 3 2 3 2 3 1 2 2 2 2 3 3 2 1 3 1
[556] 2 3 1 1 2 2 1 3 2 1 3 2 2 1 1 3 1 1 2 2 3 2 1 2 1 3 2 2 3 1 1 3 3 3 3 2 2
[593] 2 2 3 2 2 3 3 2 2 3 2 2 3 3 3 3 2 1 2 1 1 3 1 2 1 2 1 3 3 1 2 2 3 3 3 2
[630] 1 3 3 1 1 2 3 3 3 2 1 1 2 3 3 1 3 3 2 2 3 1 2 2 2 1 3 1 2 2 1 2 1 3 1 1 3
[667] 2 2 3 3 2 2 1 3 3 3 3 3 2 1 3 1 3 2 3 2 3 2 1 2 3 3 2 3 2 2 1 3 2 2 1 3 2
[704] 2 1 1 2 3 2 1 3 3 2 3 1 2 1 1 2 1 1 1 3 1 3 1 3 2 1 2 1 2 3 1 2 3 1 3 1 1
[741] 3 1 2 1 3 3 1 1 2 3 2 2 3 2 2 3 3 2 1 2 3 1 1 1 2 1 3 3 1 1 1 1 2 3 1 2 3
[778] 1 3 3 2 3 2 2 1 1 1 3 1 2 1 3 3 3 2 2 3 1 2 3 1 1 2 2 3 3 1 3 1 3 3 1 1 2
[815] 1 3 1 3 2 3 1 3 3 1 2 2 3 2 3 1 1 3 2 2 1 1 1 3 2 3 3 1 2 2 1 2 2 2 1 3 3
[852] 3 3 1 3 1 1 2 2 2 1 2 3 2 2 2 1 2 2 1 1 1 2 3 3 1 2 2 3 3 3 2 3 3 2 2 2 3

```

```

[889] 3 2 1 1 3 1 2 2 1 2 3 3 2 2 1 2 2 2 3 2 2 2 1 1 2 1 3 3 2 2 2 1 2 1 3 3 1
[926] 3 2 1 2 3 1 3 3 3 1 2 1 3 1 1 1 3 3 1 3 1 1 3 2 2 2 3 2 2 2 2 1 3 3 1 3 3
[963] 1 3 3 1 1 1 2 3 3 3 2 1 1 2 2 2 2 2 1 1 3 2 2 3 1 1 3 2 3 2 2 1 1 1 1 2 1
[1000] 2

```

```

# cluster estimated by HBCM for the first simulation data
hbcm_res[[1]]$cluster

```

```

[1] 1 3 2 3 2 3 2 2 3 1 2 2 2 3 3 2 1 3 3 1 1 3 3 2 2 1 3 3 3 1 3 3 2 3 2 2 1
[38] 3 1 1 3 3 3 3 1 2 1 3 3 1 2 2 1 3 3 3 2 3 3 2 3 3 1 1 2 2 3 3 2 3 1 1 1 1
[75] 3 2 2 1 3 3 3 3 1 1 2 3 2 1 3 1 2 2 3 3 3 2 2 1 3 1 1 2 1 1 3 2 1 2 3 1 1
[112] 3 3 2 1 3 2 3 3 1 2 3 2 2 3 1 3 3 2 1 3 3 2 3 1 2 1 1 3 1 3 1 1 3 1 3 3 1
[149] 1 3 3 1 1 1 1 1 1 2 1 1 2 1 1 2 1 3 2 2 3 2 1 3 1 2 1 1 1 2 3 2 1 2 3 3 3
[186] 3 3 3 1 3 2 1 2 1 1 1 3 3 1 1 3 2 2 3 2 1 3 3 1 2 1 1 1 2 2 3 3 1 1 2 1 1
[223] 2 1 2 3 2 1 3 3 2 2 3 1 3 2 1 1 2 2 3 3 1 2 2 2 2 3 1 3 3 2 2 2 1 3 2 1 1
[260] 2 1 1 3 3 3 3 2 2 2 2 3 1 1 3 1 2 2 2 3 2 1 3 3 1 1 3 1 2 3 1 1 2 1 2 2 3
[297] 2 2 1 1 3 3 2 3 2 3 2 3 1 3 2 3 2 2 2 3 2 1 2 2 2 3 2 2 2 2 2 3 2 3 1 2 2
[334] 1 1 1 2 1 2 1 2 1 3 2 3 2 2 1 2 1 3 2 2 3 3 3 1 2 3 2 2 3 3 1 3 3 1 3 2 1
[371] 3 2 3 3 1 1 1 3 3 2 1 3 1 1 2 1 3 1 2 2 2 1 3 3 1 2 3 3 3 3 3 2 3 2 2 3 2
[408] 1 3 2 3 1 1 2 2 2 2 3 3 3 3 3 2 1 3 3 1 1 1 1 1 1 2 3 3 2 1 3 1 3 2 2 3 3
[445] 1 1 1 1 2 2 1 3 1 3 1 3 3 1 2 1 3 3 1 1 2 1 3 1 3 2 1 1 2 1 2 1 3 1 1 1 1
[482] 1 3 1 3 2 3 3 1 3 2 1 2 3 1 1 1 3 3 2 1 1 2 1 3 3 2 3 1 3 1 3 2 1 1 1 2 1
[519] 2 3 2 2 2 1 3 1 1 1 1 3 1 3 3 1 3 1 2 3 1 3 2 3 2 3 1 2 2 2 2 3 3 2 1 3 1
[556] 2 3 1 3 2 2 1 3 2 1 2 2 2 1 1 3 1 1 2 2 3 2 1 2 1 3 2 2 3 1 1 3 3 3 3 2 2
[593] 2 2 3 2 2 3 3 2 2 3 2 2 3 3 3 3 3 2 1 2 1 1 3 1 2 1 2 1 3 3 1 2 2 3 3 3 2
[630] 1 3 3 1 1 2 3 3 3 2 1 1 2 3 3 1 3 3 2 2 3 1 2 2 2 1 3 1 2 2 1 2 1 3 1 1 3
[667] 2 2 3 3 2 2 1 3 3 3 3 3 2 1 3 1 3 2 3 2 3 2 1 2 3 3 2 3 2 2 1 3 2 2 3 3 2
[704] 2 1 1 2 3 2 1 3 3 2 3 1 2 1 1 2 1 1 1 3 1 3 1 3 2 1 2 1 2 3 1 2 3 1 3 1 1
[741] 3 1 2 1 3 3 1 1 2 3 2 2 3 2 2 3 3 2 1 2 3 1 1 1 2 1 3 3 1 1 2 1 2 3 1 2 3
[778] 1 3 3 2 3 2 2 1 1 1 3 1 2 1 3 3 3 2 2 3 1 2 3 1 1 2 2 3 3 1 3 1 3 3 1 1 2
[815] 1 3 1 3 2 3 1 3 3 1 2 2 3 2 3 1 1 1 2 2 1 1 1 3 3 3 3 1 2 2 1 1 2 2 1 3 3
[852] 3 3 1 3 1 1 2 2 2 1 2 3 2 2 2 1 2 2 1 1 1 2 3 3 1 2 2 3 3 3 2 3 3 2 2 2 3
[889] 3 2 1 1 3 1 2 2 1 2 3 3 2 2 1 2 2 2 3 2 2 2 1 1 2 1 3 3 2 2 2 1 2 1 3 3 1
[926] 3 2 1 2 3 1 3 3 3 1 2 1 3 1 1 1 3 3 1 3 1 1 1 2 2 2 3 2 2 2 2 3 3 3 1 3 3
[963] 1 3 3 1 1 1 2 3 3 3 2 1 1 2 2 2 2 2 1 1 3 2 2 3 1 1 3 2 3 2 2 1 1 1 2 2 1
[1000] 2

```

```

# adjRand to evaluate how good is the estimation to truth
matchLabel(labels, spec_labels[[1]])$adjRand

```

```

[1] 0.8807712

```

```
# adjRand to evaluate how good is the estimation to truth  
matchLabel(labels, hbcm_res[[1]]$cluster)$adjRand
```

```
[1] 0.9235103
```