The BCSub Package: A Bayesian Semiparametric Factor Analysis Model for Subtype Identification

Jiehuan Sun, Joshua L. Warren, Hongyu Zhao
Department of Biostatistics, School of Public Health, Yale University

Overview

Gene expression profiles are commonly utilized to infer disease subtypes and many clustering methods can be adopted for this task. However, existing clustering methods may not perform well when genes are highly correlated and many uninformative genes are included for clustering. To deal with these challenges, we develop a novel clustering method in the Bayesian setting. This method, called *BCSub* (*B*ayesian *C*lustering method for *Sub*type Identification), adopts an innovative semiparametric Bayesian factor analysis model to reduce the dimension of the data to a few factor scores for clustering. Specifically, the factor scores are assumed to follow the Dirichlet process mixture model in order to induce clustering (See Sun, Warren, and Zhao (2017) for details). And, the **BCSub** package can be used to perform this analysis.

This document provides a tutorial for using the **BCSub** package. The tutorial includes information on (1) the format of the input data, (2) how to choose the number of factors, an important parameter for *BCSub*, and (3) how to obtain clustering results and visually show the clustering structure. As with any R package, detailed information on functions, along with their arguments and values, can be obtained in the help files.

Input data format

The analyses performed in this tutorial are based on a simulated dataset as obtained using the code below. Basically, the data are generated from a mixture of two multivariate normal distributions, for which the covariance matrix satisfies the factor analysis model assumption. For users who only want to try *BCSub* first without knowing the models, these code might be skipped.

```r
## simulating data for illustration ##
set.seed(1)
n = 100 ## number of subjects
G = 200 ## number of genes
SNR = 0 ## ratio of noise genes
# loading matrix with four factors
lam = matrix(0,G,4)
lam[1:(G/4),1] = runif(G/4,-3,3)
lam[(G/4+1):(G/2),2] = runif(G/4,-3,3)
lam[(G/2+1):(3*G/4),3] = runif(G/4,-3,3)
lam[(3*G/4+1):(G),4] = runif(G/4,-3,3)
# generate covariance matrix
sigma <- lam%*%t(lam) + diag(rep(1,G))
sigma <- cov2cor(sigma)
# true cluster structure
e.true = c(rep(1,n/2),rep(2,n/2))
# generate data matrix
mu1 = rep(1,G)
mu1[sample(1:G,SNR*G)] = 0
mu2 <- rep(0,G)
A = rbind(mvrnorm(n/2,mu1,sigma),mvrnorm(n/2,mu2,sigma))
```

```
colnames(A) = paste("Gene",1:G,sep="")
rownames(A) = paste("Subject",1:n,sep="")
A[1:5,1:5]
```

```
##             Gene1     Gene2     Gene3     Gene4      Gene5
## Subject1 1.830091 1.0268649 0.3218707 0.3719961  2.8273946
## Subject2 1.580951 0.3963788 1.1163194 1.3510819 -0.1752386
## Subject3 1.716357 0.8711800 1.7035161 0.2552829  1.8152167
## Subject4 1.451844 0.5717502 1.0822099 2.0340510  1.2335336
## Subject5 2.131924 1.1082908 0.8388229 0.3975739  2.4373500
```

Like most gene expression data, the input data for *BCSub* is pretty standard as shown by the variable A, which is a matrix with rows being subjects and columns being genes. And, the goal of *BCSub* is to identify subgroups (clusters) in subjects.

Number of factors

BCSub adopts a factor analysis model to reduce the dimension of the data to a few factor scores for clustering. Thus, the number of factors is an important parameter for *BCSub*, which needs to be pre-specified by the users. Here, we show how to use the *parallel* function (Drasgow and Lissak 1983) to infer the number of factors, which is based on the empirical distribution of the eigenvalues of the correlation matrix of uncorrelated normal variables. As a results of the parallel analysis, the number of factors (M) is determined to be 5, which is close to the true number of factors (i.e. 4).

```
## parallel analysis to decide the number of factors ##
ev = eigen(cor(A))
ap = parallel(subject=nrow(A),var=ncol(A),rep=100,cent=.05)
nS = nScree(x=ev$values, aparallel=ap$eigen$qevpea)
M = nS$Components[1,3] # number of factors
M
```

```
## [1] 5
```

Clustering results

After the input data (A) is prepared in the right format and the number of factors (M) is determined, it is ready to run *BCSub* function for clustering. As *BCSub* is a Bayesian approach and relies on MCMC for inference, the number of iterations and the number of samples kept for posterior inference need to be specified. For the sake of time, the MCMC is run for 1000 iterations and the first 400 samples are discarded as burn-ins in the code below. In practice, the number of iterations and the number of samples kept for posterior inference should be set to ensure the convergence of the algorithm. Note that, unlike most clustering methods, *BCSub* does not require the specification of the number of clusters, which is usually challenging in practice. In this simple simulated dataset, the true cluster structure is recovered.

```
## run BCSub for clustering ##
iters = 1000 # total number of iterations
seq = 600:1000 # posterior samples used for inference
res = BCSub(A,iter=iters,seq=seq,M=M)
e.true # true cluster structure
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
res$CL # inferred cluster structure
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

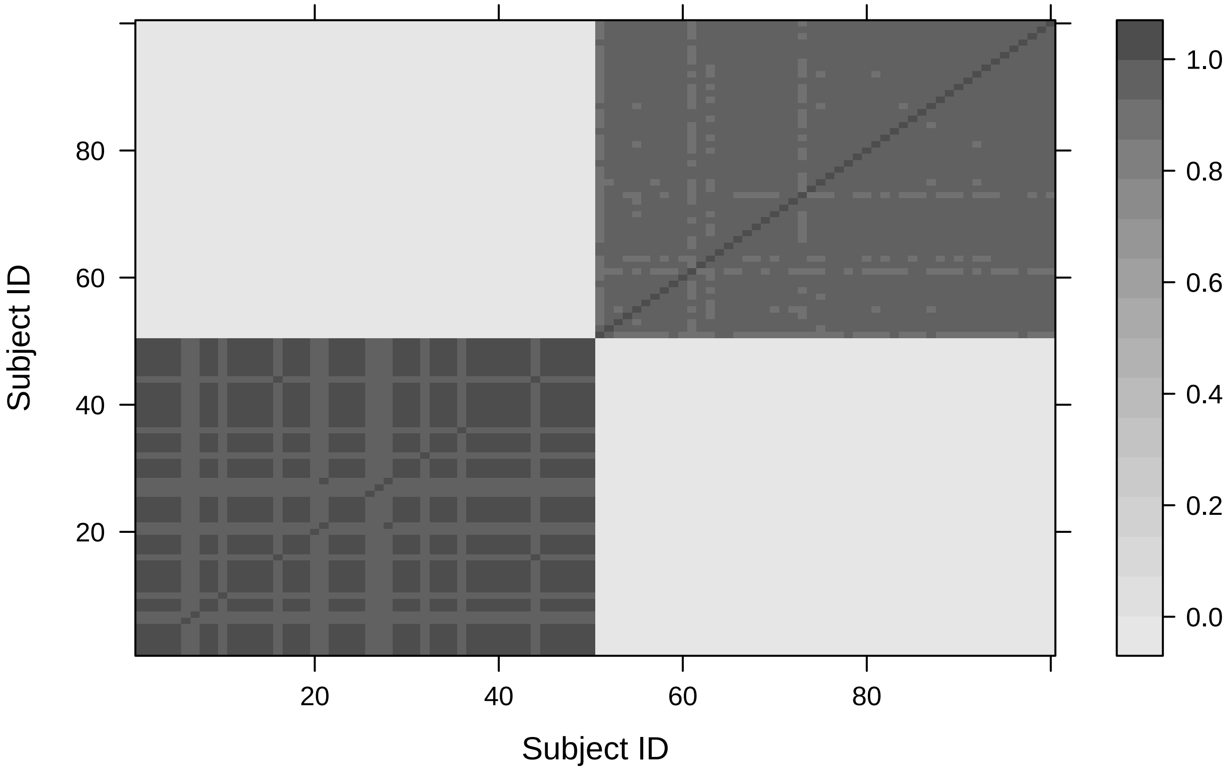
Although *BCSub* can automatically determine the number of clusters and corresponding cluster structure, it is possible to utilize the outputs of *BCSub* to produce clustering structure for a given number of clusters (this feature could be useful in practice, since researchers might have some ideas on the number of clusters that are clinically meaningful). Specifically, the posterior samples can be used to calculate the posterior similarity matrix and then Hierarchical Clustering method (*hclust*) can be used to produce clustering structure for a given number of clusters. The following code shows how to achieve this, where the desired number of clusters is 4.

```
## use hclust to get clustering results for a given number of clusters ##
sim = calSim(t(res$E[,seq])) # calculate and plot similarity matrix
K = 4 # a given number of clusters
CL = cutree(hclust(as.dist(1-sim)),k=K)
CL
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3
##  [71] 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

The posterior similarity matrix can also be visually shown so that users can have a rough idea of how many clusters there are. As shown in the figure, it is clear that there are two major clusters.

```
## plot similarity matrix ##
x = rep(1:n,times=n)
y = rep(1:n,each=n)
z = as.vector(sim)
levelplot(z~x*y,col.regions=rev(gray.colors(n^2)), xlab = "Subject ID",ylab = "Subject ID")
```



References

Drasgow, Fritz, and Robin I Lissak. 1983. "Modified Parallel Analysis: A Procedure for Examining the Latent Dimensionality of Dichotomously Scored Item Responses." *Journal of Applied Psychology* 68 (3). American Psychological Association: 363–73.

Sun, Jiehuan, Joshua L. Warren, and Hongyu Zhao. 2017. "Bayesian Semiparametric Structural Equation Models with Latent Variables." *Statistical Applications in Genetics and Molecular Biology* 0 (0). De Gruyter: 0.