DSDM3 - Vignette

Introduction

In this vignette, we demonstrate how to implement the discrete sparse Dirichlet-multinomial mixture model (DSDM^3) presented in "A Bayesian Semiparametric Mixture Model for Clustering Microbiome Data" by S Korsurat and MD Koslovsky. We start by providing guidance on how to simulate the data used in the simulation study of the main manuscript. We then demonstrate how to apply the model to both the simulated data and the HIV data featured in the application study.

Required Libraries

Below are the required packages used in this vignette.

```r
library( DSDM3 ) ## Our package.
library( ggplot2 ) ## For creating the heatmap visualizing the simulated data.
library( gridExtra ) ## For creating the heatmap visualizing the simulated data.
library( xtable ) ## For generating the tables in this vignette.
library( kableExtra ) ## For generating the tables in this vignette.
```

Simulate Data

In this section, we demonstrate how to simulate data similar to that used in the main manuscript using the `sim_clusDat()` function, which generates two clusters of zero-inflated Dirichlet-multinomial distributed data.

```r
### Simulate the data
simDat <- sim_clusDat( N = 100, Jnoise = 150, Jsignal = 50, pZero = 0.35,
                       ZSumNoise = 12500, ZSumSignal = 2500, seed = 1 )
```

The `sim_clusDat()` function requires the specification of several parameters: the number of observations (`N`), the number of taxa that are not differentiated across clusters, or the noise taxa, (`Jnoise`), the number of taxa that are differentiated between clusters, or the signal taxa, (`Jsignal`), the proportion of zeros expected in the simulated data set (`pZero`), the sequencing depth for both noise and signal taxa (`ZSumNoise` and `ZSumSignal`, respectively), and a random seed (`seed`). Note that the `sim_clusDat()` function splits the observations into two clusters evenly. The output from the `sim_clusDat()` function is a list consisting of two elements: (1) the simulated OTU table (`dat`) and (2) the cluster allocation for each observation (`c`). In this case, we have simulated 100 observations with 200 taxa, 50 of which are considered signal taxa. In the simulated data set, we expect that around 35% of the counts are zero. The assumed sequencing depth for each observation is 15,000, with 12,500 for the noise taxa and 2,500 for the signal taxa.

Note that there are six other default arguments, which are listed below:

- `shuffle`: This argument determines whether the order of observations should be shuffled. The default is `TRUE`. If set to `FALSE`, observations from the same cluster will be grouped together in the data set.
- `caseSignal`: This is a complexity index for the data generation, ranging from 1 to 5, where 1 is the most complex and 5 is the least complex. The default is 3.
- `aPhi`, `bPhi`, `aLambda`, and `bLambda`: These parameters are used to simulate the marginal probability for each signal taxa. The first ⌈`Jsignal`/2⌉ signal taxa follow a Beta(`aPhi`, `bPhi`) distribution, while the remaining signal taxa follow a Beta(`aLambda`, `bLambda`) distribution. The details regarding how to use these values to differentiate between the two clusters are explained in the main manuscript under the Simulation Study section. These arguments are set to 1 by default.

The code above can be used to generate more than two clusters in the data. For example, if we want to generate 200 observations with four different clusters, where the cluster sizes are 60, 60, 40, and 40, respectively, we can use the code below. See Figure 1 for a heatmap of the example data sets.

```
### Extend the code for simulating 4 clusters.

#### Generate the first 120 observations, 60 from each cluster,
simDat_1 <- sim_clusDat( N = 120, Jnoise = 150, Jsignal = 50, pZero = 0.35,
                         ZSumNoise = 12500, ZSumSignal = 2500, seed = 1,
                         shuffle = FALSE )

#### Generate the other 80 observations, 40 from each cluster,
simDat_2 <- sim_clusDat( N = 80, Jnoise = 150, Jsignal = 50, pZero = 0.35,
                         ZSumNoise = 12500, ZSumSignal = 2500, seed = 2,
                         shuffle = FALSE )

#### Rearrange the order of the taxa to differentiate among these 4 clusters
d2r1 <- cbind( simDat_2$dat[ 1:40, 151:200 ], simDat_2$dat[ 1:40, -(151:200) ] )
d2r2 <- cbind( simDat_2$dat[ 41:80, 1:50 ], simDat_2$dat[ 1:40, 151:200 ],
               simDat_2$dat[ 1:40, 51:150 ] )

simDat_4clus <- rbind( simDat_1$dat, d2r1, d2r2 )

#### Optional: Shuffle the order of the observations
set.seed( 1 )
index <- sample( 1:200 )
simDat_4clus <- simDat_4clus[ index, ]
simDat_4clus_c <- c( simDat_1$c, simDat_2$c + 2 )[ index ]
```

Implementation

In this section, we demonstrate how to implement DSDM^3 on the simulated data using the `ZIDM_DSDM3()` function.

```
resultMod <- ZIDM_DSDM3( dat = simDat$dat, iter = 1500, Kmax = 10, nxi_split = 10,
                         theta = 1, s2 = 0.1, s2MH = 1e-3, MHadapt = 500,
                         thin = 1, seed = 1 )
```

This function requires us to specify the $N \times J$-dimensional OTU table, where each row represents an observation and each column represents a taxon (`dat`), the number of MCMC iterations (`iter`), the number of potential components (`Kmax`), the number of concentration parameters proposed to change from the original

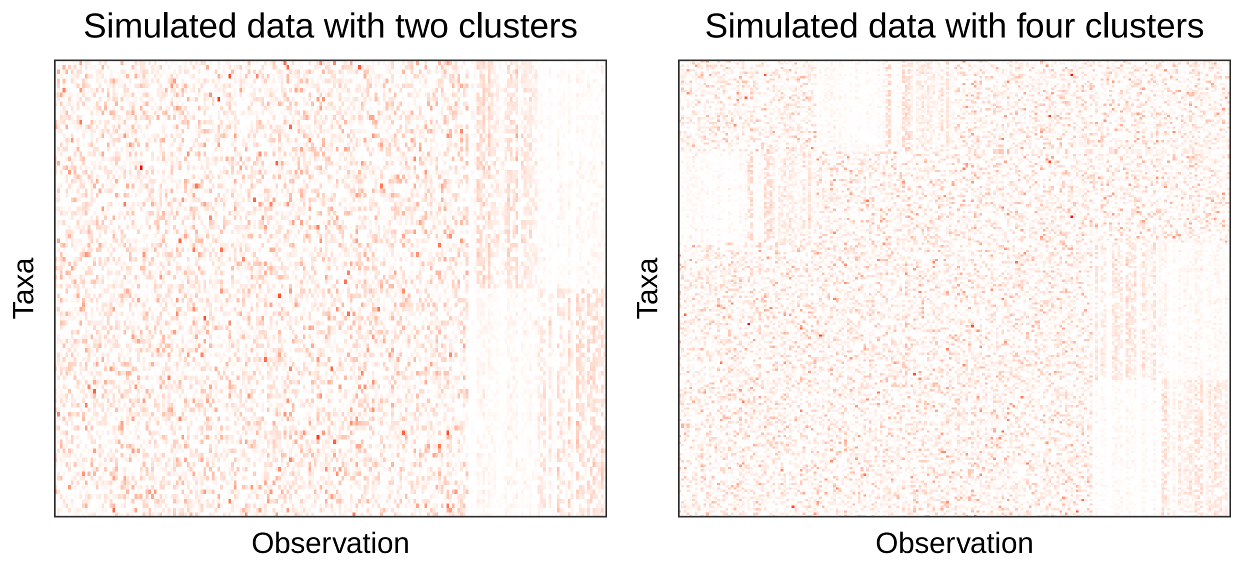

Figure 1: Heatmaps of the Simulated Data. Left figure represents the simulated data set generated using the `sim_clusDat()` function to create 2 clusters and the right figure represents the simulated data set generated using the `sim_clusDat()` function to create 4 clusters.

cluster in a split step of the Split-Merge update (`nxi_split`), the sparsity concentration parameter (`theta`), the variance of the cluster concentration parameter (`s2`), the variance for the adaptive Metropolis-Hastings (AMH) step when updating the cluster concentration parameter (`s2MH`), the number of MCMC iterations run before using the adaptive proposal (`MHadapt`), the amount of thinning (`thin`), and the random seed (`seed`). Note that the `ZIDM_DSDM3()` function initializes all observations in the same cluster. In this example, we will apply the model to the simulated data from above (`simDat$dat`). We run the model for 15,000 iterations, where the first 500 iterations use a non-adaptive proposal for the AMH step. We set the variance of the cluster concentration parameter to 0.1 and the variance of the adaptive Metropolis-Hasting to 1×10^{-3}. We limit the clusters to no more than 10 (i.e., `Kmax` $= 10$). For the Split-Merge step, the proposed cluster concentration parameters are obtained by using the cluster concentration parameters corresponding to the original cluster with 10 random taxa having new cluster concentration parameters (i.e., `nxi_split` $= 10$).

The output from the `ZIDM_DSDM3()` function is a list object. To obtain the final cluster assignment, we use the `finalCLUS()` function. For implementation, we need to specify the number of iterations to consider as burn-in. We utilize the `salso()` function from the `salso` package with the variation of information loss to determine the final cluster assignment (Dahl, Johnson, and Müller 2022).

```r
### Obtain a vector of the final cluster assignment.
clusResult <- finalCLUS( resultMod, burn_in = 500, seed = 1 )
```

Posterior Inference

Prior to performing inference on the cluster allocation, we assess the convergence of the model by plotting the number of active clusters in each MCMC iteration using the `uniqueCLUS()` function (Figure 2). We observe that the model converges around iteration 750.

```r
### Obtain a vector of the number of active cluster for each MCMC iteration.
clusMCMC <- uniqueCLUS( resultMod )
```

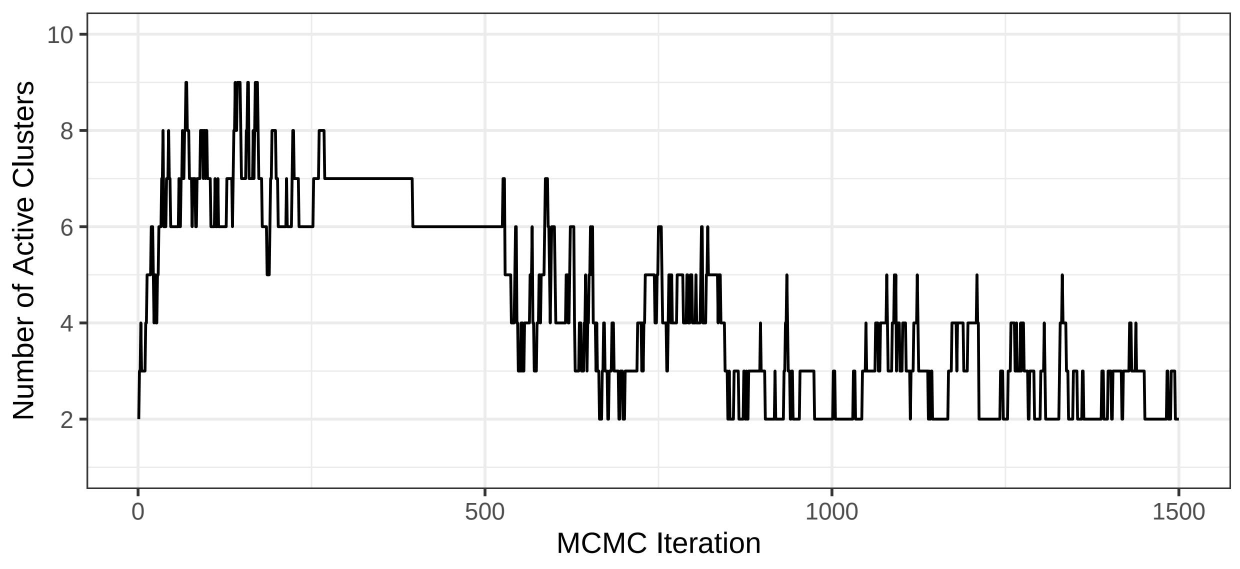

Figure 2: Simulation Results: Trace plot for the number of active clusters in each MCMC iteration when the model is applied to the simulated data.

Next, we assess the performance of the resulting cluster allocation compared to the true cluster assignment (if applicable) using the Adjusted Rand Index (ARI) with the `ariCLUS()` function. ARI measures the similarity between two cluster assignment vectors, ranging from -1 to 1. The higher the ARI, the more similar the result to the truth. ARI below 0 indicates that the clustering result is worse than what would be expected by random chance.

```
### Calculate the ARI
ariCLUS( clusResult, simDat$c )
#> [1] 1
```

We observe that the ARI equals 1, demonstrating that our model successfully differentiated between the two clusters.

Application Data

In this section, we apply the proposed model to the HIV dataset analyzed in the main manuscript collected by Noguera-Julian et al. (2016), which can be accessed in the `selbal` package (Rivera-Pinto et al. 2018). We first load the data into the R environment. The first column represents the observation ID. The second and third columns represent the sexual behavior (i.e., whether the person identifies as a man who has sex with men; MSM) and HIV infection status for each observation, respectively. See Table 1 for an example of the metadata. After removing these three columns from the imported data set, we obtain a 155×60-dimensional OTU table for inference.

```
### Import the data set
data( "selbalHIV" )
metaHIV <- selbalHIV[ , 1:3 ] ### Obtain the metadata
otuHIV <- selbalHIV[ , -(1:3) ] ### Obtain the OTU table
```

We then apply the proposed model to the HIV data, running the MCMC sampler for 25,000 MCMC iterations without thinning. We implement the AMH step after the first 2,500 iterations. The variance of the cluster

	ID	Sexual Preference	HIV Status
1	Sample_001	nonMSM	Pos
2	Sample_002	nonMSM	Pos
3	Sample_003	MSM	Pos
4	Sample_004	MSM	Neg
5	Sample_005	MSM	Neg
6	Sample_006	MSM	Neg

Table 1: Example of the metadata for the HIV data set. The first column contains the observation ID, while the other two columns represent sexual behavior and HIV infection status, respectively.

concentration parameter is set to 1, and the variance of the AMH step is set to 1×10^{-3}. We limit the model to a maximum of 20 clusters. Additionally, if a split in the cluster space is proposed, the new cluster concentration parameters are derived from those of the original cluster, with 5 randomly selected taxa having different concentration parameters.

```
### Apply the model on the HIV data set.
HIVMod <- ZIDM_DSDM3( dat = otuHIV, iter = 25000, Kmax = 20, nxi_split = 5,
                      theta = 1, s2 = 1, s2MH = 1e-3, MHadapt = 2500,
                      thin = 1, seed = 1 )
```

We obtain the final cluster assignment by using `finalCLUS()` function. In Table 2, we compare the cluster assignment to the observed sexual behavior and HIV infection status. We observed that the dominant bacteria in each resulting cluster aligns with the findings discussed in Noguera-Julian et al. (2016).

```
### Obtain the final cluster assignment for the HIV data set.
HIVclus <- finalCLUS( HIVMod, burn_in = 5000, seed = 1 )
```

	Cluster	Healthy: non-MSM	HIV: non-MSM	Healthy: MSM	HIV: MSM
1	Cluster 1	4	51	1	13
2	Cluster 2	0	2	8	19
3	Cluster 3	0	2	14	41

Table 2: Distribution of patients and sexual preferences within the resulting clusters estimated with the proposed model.

Reference

Dahl, David B, Devin J Johnson, and Peter Müller. 2022. "Search algorithms and loss functions for Bayesian clustering." *Journal of Computational and Graphical Statistics* 31 (4): 1189–1201.

Noguera-Julian, Marc, Muntsa Rocafort, Yolanda Guillén, Javier Rivera, Maria Casadellà, Piotr Nowak, Falk Hildebrand, et al. 2016. "Gut microbiota linked to sexual preference and HIV infection." *EBioMedicine* 5: 135–46.

Rivera-Pinto, Javier, Juan Jose Egozcue, Vera Pawlowsky-Glahn, Raul Paredes, Marc Noguera-Julian, and M Luz Calle. 2018. "Balances: A new perspective for microbiome analysis." *MSystems* 3 (4): 10–1128.