DSDM3 - Vignette

Introduction

In this vignette, we demonstrates how to implement the discrete sparse Dirichlet-multinoimial mixture model (DSDM^3), presented in "A Bayesian Semiparametric Mixture Model for Clustering Microbiome Data" by S Korsurat and MD Koslovsky. We begin by briefly discussing how to install the package for implementation. Next, we provide guidance on how to simulate the data used in the simulation study of the main manuscript. Last, we demonstrate how to apply the model to simulated data and data presented in the application studies.

Installation

To install the package, run the command below:

```
devtools::install_github( "skorsu/DSDM3" )
```

I have changed the GitHub repo to DSDM3. I don't know is this enough for changing the package name or not.

Simulation Data

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

```
### 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 specifying 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 dataset (`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 two elements (1) the simulated OTU table (`dat`) and (2) the cluster allocation for each observation (`c`). For example, in this case, we have simulated 100 observations with 200 taxa, 50 of which are considered signal taxa. In this 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`: Determines whether the order of observations should be shuffled. The default is `TRUE`. If set to `FALSE`, the observations from the same cluster will be grouped together.

- `caseSignal`: This is the complexity index, ranging from 1 to 5, where 1 is the most complex and 5 is the least complex. The default is 3.
- I updated this bullet point. `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 the Beta(`aPhi`, `bPhi`) distribution, while the remaining signal taxa follow the Beta(`aLambda`, `bLambda`) distribution. The detail of how to use these value for differentiate between two clusters is explained in the main manuscript under the Simulation Study section. These arguments are set to 1 as a default.

This is the brief explanation on how we use these four arguments to simulate the data and differentiate between two cluster. I think we don't have to include this in the last bullet point. What do you think? By using the simulated signal taxa marginal probability and the complexity index, we obtain the simulated data with two distinct cluster, where the vector of the signal taxa marginal probability for the first and second cluster are $\left(\left(1-\frac{\rho}{5}\right)\boldsymbol{p}_\Phi, \frac{\sum \boldsymbol{p}_\Lambda + \frac{\rho}{5}\sum \boldsymbol{p}_\Phi}{\sum \boldsymbol{p}_\Lambda}\boldsymbol{p}_\Lambda\right)$, and $\left(\left(1+\frac{\rho}{5}\right)\boldsymbol{p}_\Phi, \frac{\sum \boldsymbol{p}_\Lambda - \frac{\rho}{5}\sum \boldsymbol{p}_\Phi}{\sum \boldsymbol{p}_\Lambda}\boldsymbol{p}_\Lambda\right)$ respectively. Here, $\boldsymbol{p}_\Phi$ and $\boldsymbol{p}_\Lambda$ are the vector of the signal taxa marginal probability from the fist half and the last half respectively, and ρ is the complexity index.

The code above can be used to generate more than two clusters. 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 2 for a heatmap of the example clusters.

```r
### 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, along with posterior inference on the parameters of interest. The primary function used for implementing DSDM^3 is `ZIDM_DSDM3()`.

```
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 $N \times J$-dimentional OTU table, where each row represents the observation and each column represents the taxa (dat), the number of MCMC iterations (iter), the number of components (Kmax), the number of concentration parameters proposed to change from the original 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 before using the adaptive proposal (MHadapt), 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 clusters (i.e., Kmax $= 10$). For 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 function to determine the final cluster assignment (Dahl, Johnson, and Müller 2022).

```
### 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.

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

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.

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

We observe that the ARI equals 1, meaning our model can perfectly differentiate the observation from two clusters hidden in the simulated data. ARI can measure the similarity between two cluster assignment vectors, ranging from -1 to 1. In our case, we use ARI to measure whether the result from our model matches the truth or not. 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.

Application Data

In this section, we apply the proposed model to the HIV data 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 is the observation ID. The second and third columns represent the sexual behavior (e.g., whether the person identifies as a man who has sex with men or MSM) and HIV infection status for each observation, respectively. After removing these three columns from the imported dataset, we obtain a 155 × 60-dimensional OTU table.

```r
### Import the dataset
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 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.

```r
### Apply the model on the HIV dataset.
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. We compare the cluster assignment to the sexual behavior and HIV infection status and observe the dominant bacteria in each resulting cluster. The result align with the findings discussed in Noguera-Julian et al. (2016).

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

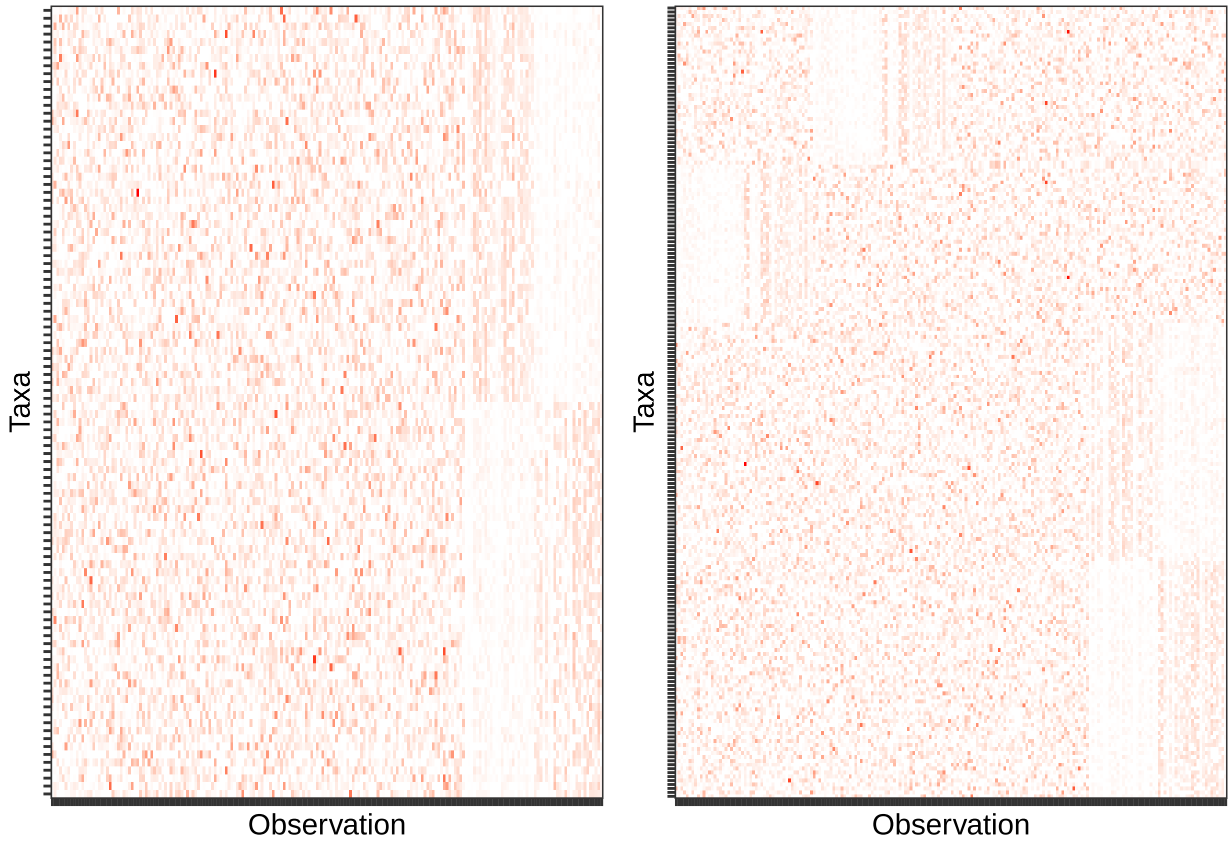

Figure 1: Heatmaps of the Simulated Data. Left: Example data set generated by the `sim_clusDat()` function to create 2 clusters. Right: Example data set generated using the `sim_clusDat()` function to create 4 clusters.

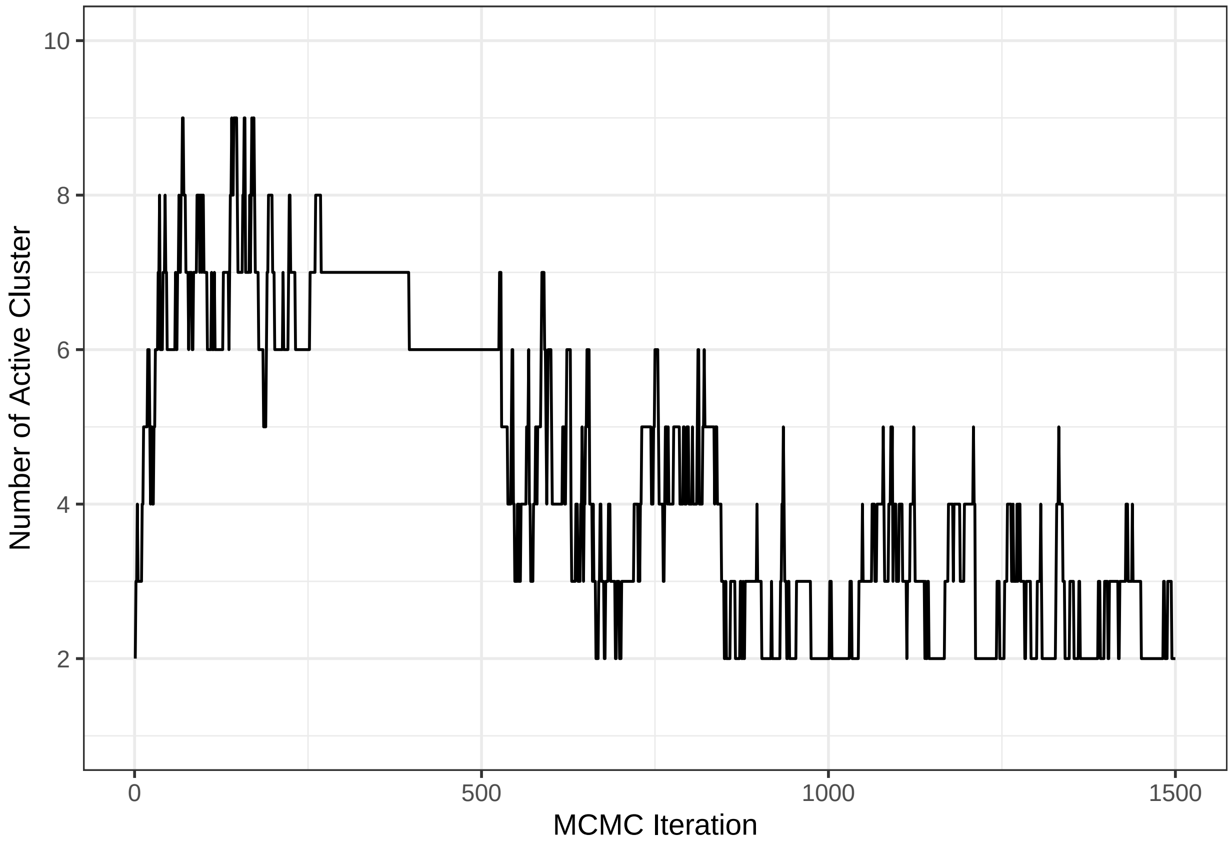

Figure 2: The line plot shows the number of active clusters for each MCMC iteration when the model is applied to the simulated data (`simDat`).

	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 dataset. The first column contains the observation ID, while the other two columns represent the metadata for each individual: sexual behavior and HIV infection status, respectively.

	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.

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.