

Getting Started with MDSMClust

Xinyue Wang

```
# Install MDSMClust from GitHub
# devtools::install_github("wxy929/MDS/MDSMClust")

library(MDSMClust)
```

Overview

The MDS for Microbiome Clustering package (v0.1.0) contains functions that conduct classical multidimensional scaling (MDS) with seven different distances or fit Dirichlet Multinomial Models (DMM) to a count matrix first, and then calculate agreement indices between any two partitions for a data set. It also contains four data sets which are used to simulate counts or analyzed as real data sets.

Dependencies

The following packages are required for functions and examples in the MDS for Microbiome Clustering package:

- SpiecEasi ($\geq 1.1.0$)
- BiocManager ($\geq 1.30.10$)
- phylr ($\geq 1.14.0$)
- fpc ($\geq 2.2.8$)
- MiSPU (≥ 1.0)
- GUniFrac (≥ 1.1)
- miProfile (≥ 1.0)
- dirmult ($\geq 0.1.3.4$)
- cluster ($\geq 2.1.0$)
- clues ($\geq 0.6.2.2$)
- DirichletMultinomial ($\geq 1.30.0$)
- MASS ($\geq 7.3.53$)
- vegan ($\geq 2.5.6$)
- reshape2 ($\geq 1.4.4$).

Among these packages, the most special one is *SpiecEasi* because you can only install it by the command “install_github(“zdk123/SpiecEasi”)“. And *phylr* and *DirichletMultinomial* are installed through *BiocManager*. Be careful with the package *miProfile*, it is not available on CRAN now and this is the only one you need to install manually before using our package. Other required packages are available on CRAN.

Examples for the Function MDS

Data Preparation

Take two real data sets as examples.

```

data("Martinez")
str(Martinez, list.len = 3)
#> List of 3
#> $ tree      :List of 6
#> ..$ edge      : int [1:20451, 1:2] 10228 10229 10229 10228 10230 10231 10232 10233 10233 10234 .
#> ..$ edge.length: num [1:20451] 0.00049 0.00206 0.00014 0.00014 0.00014 0.00014 0.00014 0.0017 0.0017 0.000
#> ..$ Nnode      : int 10225
#> .. [list output truncated]
#> ..- attr(*, "class")= chr "phylo"
#> ..- attr(*, "order")= chr "cladewise"
#> $ sampleinfo: Named chr [1:62] "Papua New Guinea" "Papua New Guinea" "Papua New Guinea" "Papua New Guinea"
#> ..- attr(*, "names")= chr [1:62] "001PNG" "002PNG" "003PNG" "004PNG" ...
#> $ otutable   : num [1:10227, 1:62] 0 0 0 0 0 573 648 145 252 178 ...
#> ..- attr(*, "dimnames")=List of 2
#> .. ..$ : chr [1:10227] "Zotu.9291" "Zotu.3366" "Zotu.9240" "Zotu.3893" ...
#> .. ..$ : chr [1:62] "001PNG" "002PNG" "003PNG" "004PNG" ...

data("Smits")
str(Smits, list.len = 3)
#> List of 3
#> $ tree      :List of 6
#> ..$ edge      : int [1:23995, 1:2] 12001 12001 12001 12002 12003 12004 12005 12006 12007 12008 ..
#> ..$ edge.length: num [1:23995] 0.00397 0.00397 0.00014 0.00014 0.00014 0.00014 0.00014 0.00014 0.00014 0.0
#> ..$ Nnode      : int 11996
#> .. [list output truncated]
#> ..- attr(*, "class")= chr "phylo"
#> ..- attr(*, "order")= chr "cladewise"
#> $ sampleinfo: Named chr [1:259] "Early Wet" "Early Wet" "Early Wet" "Early Wet" ...
#> ..- attr(*, "names")= chr [1:259] "SRR5760856" "SRR5760857" "SRR5760858" "SRR5760859" ...
#> $ otutable   : 'data.frame': 12000 obs. of 259 variables:
#> ..$ SRR5760856: num [1:12000] 0 624 204 6 24 344 414 0 269 0 ...
#> ..$ SRR5760857: num [1:12000] 0 314 110 3 4855 ...
#> ..$ SRR5760858: num [1:12000] 0 1164 1813 363 3055 ...
#> .. [list output truncated]

```

Conduct MDS and Evaluate the Clustering Results

Martinez

```

data("Martinez")

# get the tree information - it's necessary when you choose a distance which needs the tree information
tree = Martinez$tree

# get the OTU matrix - rows are samples and columns are taxa
otu.table = Martinez$otutable
otu.count = t(otu.table)
dim(otu.count)
#> [1] 62 10227

# invoke the function MDS - you need to decide the parameters first
table(Martinez$sampleinfo)
#>

```

```

#> Papua New Guinea          USA
#>           40           22
MDS(otu.count, distance = "bray", mdsrank = 5, est = "gap", K.max = 16,
group = as.numeric(as.factor(Martinez$sampleinfo)), randMethod = "MA")
#> Registered S3 methods overwritten by 'treeio':
#> method          from
#> MRCA.phylo       tidytree
#> MRCA.treedata     tidytree
#> Nnode.treedata    tidytree
#> Ntip.treedata     tidytree
#> ancestor.phylo    tidytree
#> ancestor.treedata tidytree
#> child.phylo       tidytree
#> child.treedata    tidytree
#> full_join.phylo   tidytree
#> full_join.treedata tidytree
#> groupClade.phylo  tidytree
#> groupClade.treedata tidytree
#> groupOTU.phylo    tidytree
#> groupOTU.treedata tidytree
#> is.rooted.treedata tidytree
#> nodeid.phylo      tidytree
#> nodeid.treedata   tidytree
#> nodelab.phylo     tidytree
#> nodelab.treedata  tidytree
#> offspring.phylo   tidytree
#> offspring.treedata tidytree
#> parent.phylo      tidytree
#> parent.treedata   tidytree
#> root.treedata     tidytree
#> rootnode.phylo    tidytree
#> sibling.phylo      tidytree
#> $bestk
#> [1] 10
#>
#> $ARI
#>      MA
#> 0.247169
#>
#> $cluster
#> 001PNG 002PNG 003PNG 004PNG 005PNG 006PNG 007PNG 008PNG 009PNG 010PNG 011PNG
#>      1      2      1      2      2      3      2      4      1      2      1
#> 012PNG 013PNG 014PNG 015PNG 016PNG 017PNG 018PNG 019PNG 020PNG 250PNG 251PNG
#>      5      5      1      3      5      1      2      1      2      2      1
#> 252PNG 253PNG 254PNG 255PNG 256PNG 257PNG 258PNG 259PNG 260PNG 261PNG 262PNG
#>      5      1      1      2      2      1      3      1      2      4      5
#> 263PNG 264PNG 265PNG 266PNG 267PNG 268PNG 269PNG      A      B      C      D
#>      1      2      2      3      1      3      1      6      7      8      9
#>      E      F      G      H      I      J      K      M      O      P      Q
#>     10      6     10      9      6      6      7      8     10      7      7
#>      R      S      T      U      V      W      X
#>      9      8      9      9      8      8      8

```

Smits

```
data("Smits")

# get the tree information - it's necessary when you choose a distance which needs the tree information
tree = Smits$tree

# get the OTU matrix - rows are samples and columns are taxa
otu.table = as.matrix(Smits$otutable)
otu.count = t(otu.table)
dim(otu.count)
#> [1] 259 12000

# invoke the function MDS - you need to decide the parameters first
table(Smits$sampleinfo)
#>
#> Early Wet Late Dry
#> 62 197

# detailed output can be found in the html file
# s = MDS(otu.count, distance = "PhILR", tree = tree, mdsrank = 5, est = "si", K.max = 8, group = as.nu
# s$bestk
# s$ARI
# table(s$cluster)
```

Examples for the Function DMM - Martinez

```
data(Martinez)
tree = Martinez$tree
otu.table = Martinez$otutable
otu.count = t(otu.table)

# DMM(otu.count, group = as.numeric(as.factor(Martinez$sampleinfo)), k = 5, trim = T, threshold = 5)
```

Example Code to Reproduce DM Simulation in the paper

```
rm(list = ls())

library(SpiecEasi)
library(phylr)
library(fpc)
library(MiSPU)
library(GUniFrac)
library(miProfile)
library(dirmult)
library(cluster)
library(clues)
library(DirichletMultinomial)
library(MASS)
library(vegan)
library(mvtnorm)
```

```

library(reshape2)
library(doSNOW)
library(MDSMClust)

# pre_DM_RData -----
ini.seed = 2020
n = 400
signal.cluster.zero = 0
signal.zero = 0
readdepth = 4000

data("DM_throat_info")
throat.tree = throat.info$tree
tree.otu.id = throat.tree$tip.label
tree.par.pi = throat.info$par.pi
tree.par.theta = throat.info$par.theta
otu.cluster = throat.info$otu.cluster
m = length(tree.par.pi)      # 281
n1 = n2 = n3 = n4 = n/4      # 100

groups = c(rep(0,n1), rep(1,n2), rep(2,n3), rep(3,n4))
set.seed(ini.seed)

otu.perc.cluster = NULL
for(i in 1:20){
  otu.perc.cluster = c(otu.perc.cluster, sum(tree.par.pi[which(otu.cluster==i)], na.rm = TRUE) )
}

## simulate the difference - change three clusters
# common -> DM_Common.RData
signal.cluster.pos = c(6, 17, 7)
signal.pos = 1.4

tree.par.pi.grp1 = tree.par.pi
tree.par.pi.grp2 = tree.par.pi
tree.par.pi.grp3 = tree.par.pi
tree.par.pi.grp1[which(otu.cluster==signal.cluster.pos[1])] = tree.par.pi.grp1[which(otu.cluster==signal.cluster.pos[1])]
tree.par.pi.grp2[which(otu.cluster==signal.cluster.pos[2])] = tree.par.pi.grp2[which(otu.cluster==signal.cluster.pos[2])]
tree.par.pi.grp3[which(otu.cluster==signal.cluster.pos[3])] = tree.par.pi.grp3[which(otu.cluster==signal.cluster.pos[3])]

# rare -> DM_Rare.RData
signal.cluster.pos = c(19, 20, 14)
signal.pos = 1.4

tree.par.pi.grp1 = tree.par.pi
tree.par.pi.grp2 = tree.par.pi
tree.par.pi.grp3 = tree.par.pi
tree.par.pi.grp1[which(otu.cluster==signal.cluster.pos[1])] = tree.par.pi.grp1[which(otu.cluster==signal.cluster.pos[1])]

```

```

tree.par.pi.grp2[which(otu.cluster==signal.cluster.pos[2])] = tree.par.pi.grp2[which(otu.cluster==signal.cluster.pos[2])]
tree.par.pi.grp3[which(otu.cluster==signal.cluster.pos[3])] = tree.par.pi.grp3[which(otu.cluster==signal.cluster.pos[3])]

# random -> DM_Random.RData
signal.pos = 1.4
set.seed(ini.seed)
id = sample(1:281, 60)
id1 = which(names(tree.par.pi) %in% tree.otu.id[id[1:20]])
id2 = which(names(tree.par.pi) %in% tree.otu.id[id[21:40]])
id3 = which(names(tree.par.pi) %in% tree.otu.id[id[41:60]])

tree.par.pi.grp1 = tree.par.pi
tree.par.pi.grp2 = tree.par.pi
tree.par.pi.grp3 = tree.par.pi
tree.par.pi.grp1[id1] = tree.par.pi.grp1[id1] * signal.pos * (1/sqrt(sum(tree.par.pi[id1], na.rm = TRUE)))
tree.par.pi.grp2[id2] = tree.par.pi.grp2[id2] * signal.pos * (1/sqrt(sum(tree.par.pi[id2], na.rm = TRUE)))
tree.par.pi.grp3[id3] = tree.par.pi.grp3[id3] * signal.pos * (1/sqrt(sum(tree.par.pi[id3], na.rm = TRUE)))

# Simulation in One of Three Possible Scenarios -----

# load("DM_Common.RData") from above code
rare = FALSE # TRUE if consider the rarefication

set.seed(2024) # seeds from 2024 to 2523 - avg of 500 seeds reported in the paper
# simulate data from Dirichlet-multinomial distribution
otu.countsim = rbind( simPop(J=n1, n=readdepth[1:100], pi=tree.par.pi, theta=tree.par.theta)$data,
                      simPop(J=n2, n=readdepth[101:200], pi=tree.par.pi.grp1, theta=tree.par.theta)$data,
                      simPop(J=n3, n=readdepth[201:300], pi=tree.par.pi.grp2, theta=tree.par.theta)$data,
                      simPop(J=n4, n=readdepth[301:400], pi=tree.par.pi.grp3, theta=tree.par.theta)$data)

otu.propsim = otu.countsim/rowSums(otu.countsim)
colnames(otu.propsim) = tree.otu.id
colnames(otu.countsim) = tree.otu.id

# rarefication
if(rare){
  readdepth.rarefied = min(readdepth)
  otu.countsim.rarefied = matrix(0, ncol = ncol(otu.countsim), nrow = nrow(otu.countsim))
  for(i in 1:n){
    otu.countsim.rarefied[i, ] = t(rmultinom(n=1, size = readdepth.rarefied, prob = otu.propsim[i, ]))
  }
  colnames(otu.countsim.rarefied) = tree.otu.id

  otu.propsim = otu.countsim.rarefied/rowSums(otu.countsim.rarefied)
  colnames(otu.propsim) = tree.otu.id
  otu.countsim = otu.countsim.rarefied

  print("Rarified")
}

# then take the simulated "otu.count" as the input to the "MDS" function

```