A tutorial on R package AIT

Yaru Song [1,3], Hongyu Zhao [2,3], and Tao Wang [1,3,4,*]

[1] Department of Bioinformatics and Biostatistics, Shanghai Jiao Tong University, China,
[2] Department of Biostatistics, Yale University, USA and
[3] SJTU-Yale Joint Center for Biostatistics, Shanghai Jiao Tong University, China
[4] MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University

November 26, 2019

1 Introduction

The `AIT` package provides an Adaptive Independence Test for microbiome community data.

To test the assiciation between the microbial community composition and a continuous or many-valued outcome variable, we propose a parametric test based on likelihood ratio test. The basic idea is to partition the range of the outcome variable into a few slices or bins, then test whether the community distributions are equal across the slices. Our method regularizes the log likelihood ratio with a penalty on the number of slices, and maximizes over all possible slicing schemes. To model multivariate count data, we use multinomial and Dirichlet-multinomial distribution, thus proposed two kinds of test: `AIT-MN` and `AIT-DMN`, repectively. Microbiome studies are subject to many other covariate factors, thus we extend to a new t method `AIT-DMN-a` based on `AIT-DMN` to adjust other covariates. Functions of performing test, several simulated and real dataset are provided. Core functions for the test are implemented in the `Cpp` and integrated in `R` through the `Rcpp` package (Eddelbuettel *et al.*, 2011).

2 Running `AIT`

First, we need to install package `Rcpp`, `foreach`, `GUniFrac`, `MGLM`, and `stats` before we install `AIT`. To load `AIT`, type:

```
> # install.packages("devtools")
> # devtools::install_github("YaruSong/AIT")
```

```
>
> library(AIT)
```

2.1 A simulated example without other covariate

To see how `AIT` performs the independence test between microbial community composition and a categorical variable or a continuous variable, we generate 100 individals' micribiome compositions and corresponding outcome variable. For simplicity, we set the vector of bacterial counts, $\boldsymbol{X}$, follows a Dirichlet-multinomial distribution, and sample size to be 100, the number of taxa to be 30, and the sequencing depth as 1000 for all individuals, dispersion as 0.2. We divided the 100 individuals into 4 groups with 10, 20, 30, 40 in each group, and induce abundance differences by increasing and decreasing the proportions of six taxa. For more details see Song *et al.* (2019). Simulation Procedures are as follows.

```
> rm(list=ls())
> set.seed(2019)
> samples=100
> groups.samples<-c(10,20,30,40)
> dispersion=0.2
> seq.depth=1000
> mean.0<-c(rep(0,6),rep(0.005,5),rep(0.01,5),rep(0.02,4),
+ rep(0.03,3),rep(0.04,2),c(0.08,0.1,0.11,0.165,0.22))
> mean.change<-function(pi,location,alpha){
+   mean.tmp=sort(pi)
+   mean.tmp[0+location]= mean.tmp[0+location]+alpha*mean.tmp[20+location]
+   mean.tmp[20+location]= (1-alpha)*mean.tmp[20+location]
+   return(mean.tmp)
+ }
> beta=0.6  #beta acts as the effect size
> mean.control=mean.0
> mean.case1=mean.change(mean.0,c(8,9,10),beta)
> mean.case2=mean.change(mean.0,c(5,6,7),beta)
> mean.case3=mean.change(mean.0,c(2,3,4),beta)
> mean.group<-as.matrix(rbind(mean.control,mean.case1,mean.case2,mean.case3))
> x.d=sim_x("DMN",groups.samples,seq.depth,mean.group,dispersion)
> x.m=sim_x("MN",groups.samples,seq.depth,mean.group)
```

In the above, we simulate microbiome data with function `sim_x`, parameter `method` has two options: `DMN` for Dirichlet-multinomial distribution and `MN` for multinomial distribution, and `theta` is not the parameter for `MN` in `sim_x`.

2.1.1 Y is categorical

For outcome variable Y, we set 10 levels, and divide the 10 levels into 4 groups: {0.5}, {1,1.5}, {2,2.5,3} and {3.5,4,4.5,4}, with 10 individuals in each level.

```
> y=sort(rep(1:10,10))/2
> y.d=y.m=y
```

```
> print(theta_calculate(x.d))
```

```
[1] 0.2217333
```

```
> print(theta_calculate(x.m))
```

```
[1] 0.01511535
```

Here, `theta_calculate` is a funtion of moment estimation for overdispersion `theta` in Dirichlet-multinomial distribution. If `theta`>0, we recommend you to use `AIT_DMN_p` for the later analysis; otherwise, `AIT_MN_p` is more appropriate. In order to control type I error, choose an optimal penalty lambda by function `lambda_selection`, then perform independence test using `AIT_DMN_p`, which returns the `statistics`, `p_value`, `slicing`.

```
> lambda_d= lambda_selection(method = "DMN", x.d, y.d, 1000)
> print(lambda_d)
```

```
[1] 0.5
```

```
> print(AIT_DMN_p(x.d, y.d, lambda_d, 1000, slice=TRUE))
```

```
$statistics
[1] 202.1575

$p_value
[1] 0.001

$slicing
   0.5  1 1.5  2 2.5  3 3.5  4 4.5  5 total
s1  10  0   0  0   0  0   0  0   0  0    10
```

```
s2   0 10  10  0   0  0   0  0   0  0    20
s3   0  0   0 10  10 10   0  0   0  0    30
s4   0  0   0  0   0  0  10 10  10 10    40
```

The test splits 10 levels into 4 slices, which is in accordance with our predefined groups.

2.1.2 Y is continuous

In the above simulation, Y is a categorical variable. Then we simulate a continuous variable Y, and $\boldsymbol{X}$ depends on Y only through four latent slices of Y. For more details see Song *et al.* (2019).

```
> set.seed(2019)
> y.d=sort(runif(samples, min=0.01, max=1))
> #lambda_d=lambda_selection("DMN",x.d,y.d,100)  #3.125
> lambda_d=3.125
> results=AIT_DMN_p(x.d,y.d,lambda_d,100,slice = TRUE)
> print(results$p_value)
```

```
[1] 0.01
```

```
> print(results$slicing[,ncol(results$slicing)])
```

```
s1 s2 s3 s4
10 20 30 40
```

The test also splits Y into 4 slices, which is also in accordance with our predefined latent slices.

2.2 A simulated example with other covariate

Microbiome studies are subject to many covariates, and how to properly model and adjust for covariates is critically important for the independence/association test.

We consider a simulation example as follows. The taxa abundance data were generated from the Dirichlet-multinomial model, with 100 individuals, 6 taxa, sequencing depth 500. Y had 10 levels, $0.5, 1, \cdots, 5$, with 10 samples at each level. We divided the 10 levels into 4 groups as above, and simulated 1 dimensional `covariates`. The coefficient vector `coefficent`

and the covariate vector `covariates` were drawn from $N(0,1)$, and the vector of intercepts `mean.proportion` was generated from $N(1,1)$. Besides, we set coefficient for 4 groups as $0,1,2,3$ (`gamma`). Belows are simulation Processes using function `sim_x_c`.

```
> set.seed(2019)
> p <- 6  # number of taxa
> d <- 1  # number of covariates
> samples <- 100
> seq.depth=500
> mean.proportion=rnorm(p,1,1)
> groups.samples<-c(10,20,30,40)
> gamma=t(matrix(sort(rep(0:3,p)),p,4))
> coefficent=matrix(rnorm(p),1,p)
> covariates=matrix(rnorm(samples),samples,1)
> x.dc=sim_x_c(seq.depth,mean.proportion,gamma,groups.samples,coefficent,covariates)
> y.dc=y
```

The function `sim_x_c` simulate Dirichlet-multinomial distribution subject to other covariates.

Then, choose a optimal penalty lambda with function `lambda_selection`, and perform the test with covariate adjustment function `AIT_DMN_adj_p`

```
> # lambda_da=lambda_selection("DMN",x.dc,y.dc,permutation=100,covariates=covariates) #2.75
> lambda_da=2.75
> test.results=AIT_DMN_adj_p(x.dc,y.dc,covariates,lambda_da,permutation=100,slice=TRUE)
> print(test.results)


$statistics
[1] 43.46284

$p_value
[1] 0.01

$slicing
   0.5  1 1.5  2 2.5  3 3.5  4 4.5  5 total
s1  10  0   0  0   0  0   0  0   0  0    10
s2   0 10  10  0   0  0   0  0   0  0    20
s3   0  0   0 10  10 10  10 10  10 10    70
```

As the difference between 3th and 4th group are not as clear as others, our method can't

distinguish those 2 groups. However, slightly under-slicing doesn't affect the test much, the p-value is still significant.

2.3 A real example

Yatsunenko *et al.* (2012) conducted a study to investigate whether there exist taxa abundance differences across age and geography, where fecal specimens were collected from 531 healthy infants, children, and adults living in Amazonas (Venezuela), rural Malawi, and USA metropolitan areas.

We combined the OTUs at the family level (242 families) as the dataset `taxa_family`. Besides, we also included individual's "Age", "Country", "BMI", "Gender" and "Breast.fed" information in dataset `individual_index`. We can load the data by:

```
> data(individual_index)
> data(taxa_family)
```

We restricte attention to subjects less than 40 years old. To reduce the number of taxa, we discard family-level OTUs that appeare in less than 25% of the samples in each population, and those with relative abundance <= 0.02% in all samples. Here, we illustrate the test with "Malawi".

```
> library(GUniFrac)
> index.tmp=subset(individual_index,individual_index$Age<=40&
+                    individual_index$Country=="Malawi")
> age=index.tmp$Age
> taxa_age=taxa_family[row.names(index.tmp),]
> taxa_age_sub <- taxa_age[,(colSums(taxa_age != 0) ) >nrow(taxa_age)*0.25]
> taxa_age_sub=taxa_age_sub[,apply(taxa_age_sub, 2,
+                                  function(x){max(x)})>sum(taxa_age_sub)*0.0002]
> x=Rarefy(taxa_age_sub,depth=10000)$otu.tab.rff  #rarefy to the same sequencing depth
> x=x[,colSums(x)>0]
> age.group=c(0,0.25,0.5,0.75,1,1.5,2,2.5,3)
> y=age
> for(i in 1:length(age)){
+  if(age[i]>3){
+    y[i]=ceiling(age[i])}
+  for(j in 1:(length(age.group)-1)){
+    if(age.group[j]<age[i] &age[i] <=age.group[j+1]){
+      y[i]=age.group[j+1]
+    }
```

```
+   }
+ }
```

Then, we perform the adaptive independent test:

```
>  theta_calculate(x)  # here theta>0.1, thus we use function AIT_DMN_p

[1] 0.160982

>  lambda_d=lambda_selection(method="DMN",x,y,100)
>  test.results=AIT_DMN_p(x,y,lambda_d,1000,slice=TRUE)
>  print(test.results)

$statistics
[1] 140.0354

$p_value
[1] 0.001

$slicing
   0.25 0.5 0.75  1 1.5 2 2.5 3 20 21 23 24 25 26 27 28 32 33 34 38 total
s1    8   8    6 11   9 0   0 0  0  0  0  0  0  0  0  0  0  0  0  0    42
s2    0   0    0  0   0 7  11 2  1  2  2  3  2  2  2  1  2  2  1  1    41
```

Here p-value is 0.001, indicating a strong association between community composition and age in this population. Adaptive slicing divided the samples into two slices: subjects less than 1.5, and those more than 2 years old. The slicing result is in compatible with the unsupervised learning method: heatmap from hierarchical clustering, with average linkage and Bray-Curtis distance (Song *et al.*, 2019).

References

Eddelbuettel, D., François, R., Allaire, J., Ushey, K., Kou, Q., Russel, N., Chambers, J., and Bates, D. (2011). Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, **40**(8), 1–18.

Song, Y., Zhao, H., and Wang, T. (2019). An adaptive independence test for microbiome community data. *Biometrics*, pages 1–13.

Yatsunenko, T., Rey, F. E., Manary, M. J., Trehan, I., Dominguez-Bello, M. G., Contreras, M., Magris, M., Hidalgo, G., Baldassano, R. N., Anokhin, A. P., *et al.* (2012). Human gut microbiome viewed across age and geography. *Nature*, **486**(7402), 222–227.