Introduction to IFAA

IFAA is a novel approach to make inference on the association of covariates with the absolute abundance (AA) of microbiome in an ecosystem.

To model the association, the following equation is used:

$$\log(\mathcal{Y}_i^k)|\mathcal{Y}_i^k > 0 = \beta^{0k} + X_i^T\beta^k + W_i^T\gamma^k + Z_i^Tb_i + \epsilon_i^k, \quad k = 1, ..., K+1,$$

where

- $\mathcal{Y}_i^k$ is the AA of taxa k in subject i in the entire ecosystem.
- X_i is the covariate matrix.
- W_i is the confounder matrix.
- Z_i is the design matrix for random effects.
- β^k is the regression coefficients that will be estimated and tested with the `IFAA()` function.

The challenge in microbiome analysis is that we can not oberve $\mathcal{Y}_i^k$. What is observed is its small proportion: $Y_i^k = C_i\mathcal{Y}_i^k$ where C_i is an unknown number between 0 and 1 that denote the observed proportion. The IFAA method can handle this challenge by identifying and employing reference taxa.

Package installation

The package "IFAA" could be installed from GitHub using the following code:

```r
require(devtools)
devtools::install_github("gitlzg/IFAA")
library(IFAA)
```

Input and Output for IFAA() function

The `IFAA()` function is the main function. The User Inputs are:

- `MicrobData`: Microbiome data matrix containing microbiome abundance with each row per sample and each column per taxon/OTU/ASV. It should contain an `"id"` variable to correspond to the `"id"` variable in the covariates data: `CovData`.
- `CovData`: Covariates data matrix containing covariates and confounders with each row per sample and each column per variable. It should also contain an `"id"` variable to correspond to the `"id"` variable in the microbiome data: `MicrobData`.
- `linkIDname`: Variable name of the `"id"` variable in both `MicrobData` and `CovData`. The two data sets will be merged by this `"id"` variable.
- `testCov`: Covariates that are of primary interest for testing and estimating the associations. It corresponds to X_i in the equation. Default is `NULL` which means all covariates are `testCov`.
- `ctrlCov`: Potential confounders that will be adjusted in the model. It corresponds to W_i in the equation. Default is `NULL` which means all covariates except those in `testCov` are adjusted as confounders.

- `testMany`: This takes logical value `TRUE` or `FALSE`. If `TRUE`, the `testCov` will contain all the variables in `CovData` provided `testCov` is set to be `NULL`. The default value is `TRUE` which does not do anything if `testCov` is not `NULL`.
- `ctrlMany`: This takes logical value `TRUE` or `FALSE`. If `TRUE`, all variables except `testCov` are considered as control covariates provided `ctrlCov` is set to be `NULL`. The default value is `TRUE` which does not do anything if `ctrlCov` is not `NULL`.
- `nRef`: The number of randomly picked reference taxa used in phase 1. Default number is `40`.
- `nPermu`: The number of permutation used in phase 1. Default number is `40`.
- `x1permut`: This takes a logical value `TRUE` or `FALSE`. If true, it will permute the variables in testCov. If false, it will use residual-permutation proposed by Freedman and Lane (1983).
- `refTaxa`: A vector of taxa names. These are reference taxa specified by the user to be used in phase 1. If the number of reference taxa is less than 'nRef', the algorithm will randomly pick extra reference taxa to make up 'nRef'. The default is `NULL` since the algorithm will pick reference taxa randomly.
- `reguMethod`: regularization approach used in phase 1 of the algorithm. Take value `"mcp"` or `"lasso"`, default is `"mcp"`.
- `fwerRate`: The family wise error rate for identifying taxa/OTU/ASV associated with `testCov` in phase 1. Default is `0.25`.
- `sequentialRun`: This takes a logical value `TRUE` or `FALSE`. Sometimes parallel jobs can not be successfully run for unknown reasons. For example, socket related errors may pop up or some slave cores return simple error instead of numerical results. In those scenarios, setting `sequentialRun = TRUE` may help, but it will take more time to run. Default is `FALSE`.
- `paraJobs`: If `sequentialRun` is `FALSE`, this specifies the number of parallel jobs that will be registered to run the algorithm. Default is `8`. If specified as `NULL`, it will automatically detect the cores to decide the number of parallel jobs.
- `standardize`: This takes a logical value `TRUE` or `FALSE`. If `TRUE`, all design matrix X in phase 1 and phase 2 will be standardized in the analyses. Default is `FALSE`.
- `nRefMaxForEsti`: The maximum number of reference taxa used in phase 2. The default is `1`.
- `bootB`: Number of bootstrap samples for obtaining confidence interval of estimates in phase 2. The default is `500`.
- `bootLassoAlpha`: The significance level in phase 2. Default is `0.05`.
- `refReadsThresh`: The threshold of non-zero sequencing reads for choosing the reference taxon in phase 2. The default is `0.2` which means at least 20% non-zero sequencing reads.
- `SDThresh`: The threshold of standard deviations of sequencing reads for choosing the reference taxon in phase 2. The default is `0.5` which means the standard deviation of sequencing reads should be at least `0.5`.
- `balanceCut`: The threshold of non-zero sequencing reads in each group of a binary variable for choosing the reference taxon in phase 2. The default number is `0.2` which means at least 20% sequencing reads are non-zero in each group.
- `seed`: Random seed for reproducibility. Default is `1`.

The output of `IFAA()` function is a list. The estimation results can extracted as the following:

- `analysisResults$estByCovList`: A list containing estimating results for all the variables in `testCov`. See details.

The covariates data including `testCov` and `ctrlCov` can be extracted in the output:

- `covariatesData`: A dataset containing covariates and confounders used in the analyses

Examples

The example datasets `dataM` and `dataC` are included in the package. They could be accessed by:

```
data(dataM)
dim(dataM)
#> [1] 20 60
dataM[1:5, 1:8]
#>   id rawCount1 rawCount2 rawCount3 rawCount4 rawCount5 rawCount6 rawCount7
#> 1  1         0         0         0         0         0         3         0
#> 2  2         0         0         0         0         0         0         0
#> 3  3         0         0         0         0         0       214         0
#> 4  4         0         0         0         0         0         2         0
#> 5  5         0         0         0         0         0        40         0

data(dataC)
dim(dataC)
#> [1] 20  6
dataC[1:5, ]
#>   id v4       v1 v5 v2 v3
#> 1  1  1 1.653901  4  1 NA
#> 2  2  2 0.362706  5  2  2
#> 3  3  1 1.496269 NA  5  2
#> 4  4  1 1.755541  5  3  3
#> 5  5  1 1.035714  5  7 NA
```

Both the microbiome data `dataM` and the covariates data `dataC` contain 20 samples (i.e., 20 rows).

- `dataM` contains 60 taxa with absolute abundances and these are gut microbiome.
- `dataC` contains 5 covariates.

Next we analyze the data to test the association between microbiome and the two variables `"v1"` and `"v2"` while adjusting for the variable `"v3"`.

```
results <- IFAA(MicrobData = dataM,
                CovData = dataC,
                linkIDname = "id",
                testCov = c("v1", "v2"),
                ctrlCov = c("v3"),
                nRef = 4,
                nPermu = 4,
                fwerRate = 0.25,
                bootB = 5)
#> There are 41 taxa without any sequencing reads and
#>         excluded from the analysis
#> Data dimensions (after removing missing data if any):
#> 13 samples
#> 18 OTU's or microbial taxa
#> 2 testCov variables in the analysis
#> [1] "These are the testCov variables:"
#> [1] "v1" "v2"
#> 1 ctrlCov variables in the analysis
#> [1] "These are the ctrlCov variables:"
#> [1] "v3"
```

```
#> 0 binary covariates in the analysis
#> 54.27 percent of microbiome sequencing reads are zero
#> Start Phase 1 association identification
#> start Original screen
#> Loading required namespace: Rmpi
#>  6 slaves are spawned successfully. 0 failed.
#> 6 parallel jobs are registered for analyzing 4 reference taxa in Phase 1a.
#> OriginDataScreen parallel setup took 6.24 seconds
#> Original screen done and took 0.1255 minutes
#> start to run permutation
#>  6 slaves are spawned successfully. 0 failed.
#> 6 parallel jobs are registered for the permutation analysis in Phase 1b.
#> Permutation done and took 0.3453333 minutes
#> Phase 1 Associaiton identification is done and used 0.588 minutes
#> Start Phase 2 parameter estimation
#> Final Reference Taxa are: rawCount9
#> Start estimation for the 1 th final reference taxon: rawCount9
#> 8 parallel jobs are registered for bootstrp in Phase 2.
#> Estimation done for the 1 th final reference taxon: rawCount9 and it took 0.08283333 minutes
#> Phase 2 parameter estimation done and took 0.08283333 minutes.
#> The entire analysis took 0.672 minutes
```

In this example, we are only interested in testing the association with `"v1"` and `"v2"` which is why `testCov=c("v1","v2")`. The variable `"v3"` is adjusted as a potential confounder in the analyses. For the sake of speed in this hypothetical example, we set small numbers for `nRef=4`, `nPermu=4` and `bootB=5`. These are just for illustration purpose here and are too small for a formal analysis to generate valid results.

The final analysis results are stored in the list `analysisResults$estByCovList`:

```
results$analysisResults$estByCovList
#> $v2
#>               Beta.LPR LowB95%CI.LPR UpB95%CI.LPR
#> rawCount29 0.04045972   0.007860232   0.05096203
#> rawCount42 0.02472210  -0.030848352   0.05009310
```

The results found the two taxa `"rawCount29"` and `"rawCount42"` associated with `"v2"`. The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to β^k in the model equation.

The interpretation is that

- Every unit increase in `"v2"` is associated with approximately 4.0% increase in the absolute abundance of `"rawCount29"` and approximately 2.5% increase in the absolute abundance of `"rawCount42"` in the entire gut ecosystem.
- There were no taxa associated with `"v1"` in the analysis.

All the analyzed covariates including `testCov` and `ctrlCov` are stored in the object: `covariatesData`:

```
results$covariatesData
#>    id          v1 v2  v3
#> 2   2  0.36270596  2   2
#> 3   3  1.49626921  5   2
#> 4   4  1.75554095  3   3
#> 6   6  1.64525227  4   4
#> 8   8 -1.57781131 24  22
#> 9   9  2.22581203 55   5
```

```
#> 10 10  0.71642615 98  67
#> 12 12  2.12230160 98   3
#> 14 14  1.99387922 93   4
#> 16 16  0.05417617 83  34
#> 18 18 -0.43426021 73  67
#> 19 19  1.46579846 68 566
#> 20 20  1.89625949 63  34
```

MZILN() function

The IFAA package also offers the MZILN() function to implement the Multivariate Zero-Inflated Logistic Normal regression model for analyzing microbiome data. The regression model for `MZILN()` can be expressed as follows:

$$\log\left(\frac{\mathcal{Y}_i^k}{\mathcal{Y}_i^{K+1}}\right)|\mathcal{Y}_i^k>0,\mathcal{Y}_i^{K+1}>0=\alpha^{0k}+\mathcal{X}_i^T\alpha^k+\epsilon_i^k,\ \ k=1,...,K,$$

where

- $\mathcal{Y}_i^k$ is the AA of taxa k in subject i in the entire ecosystem.
- $\mathcal{Y}_i^{K+1}$ is the reference taxon (specified by user).
- $\mathcal{X}_i$ is the covariate matrix for all covariates including confounders.
- α^k is the regression coefficients that will be estimated and tested by the `MZILN()` function.

Input and Output for MZILN() function

The `MZILN()` function is to implement the Multivariate Zero-Inflated Logistic Normal model. It estimates and tests the associations given a user-specified reference taxon/OTU/ASV, whereas the 'IFAA()' does not require any user-specified reference taxa. If the user-specified taxon is independent of the covariates, 'MZILN()' should generate similar results as 'IFAA()'. The User Inputs for 'MZILN()' are:

- `MicrobData`: Microbiome data matrix containing microbiome abundance with each row per sample and each column per taxon/OTU/ASV. It should contain an `"id"` variable to correspond to the `"id"` variable in the covariates data: `CovData`.
- `CovData`: Covariates data matrix containing covariates and confounders with each row per sample and each column per variable. It should also contain an `"id"` variable to correspond to the `"id"` variable in the microbiome data: `MicrobData`.
- `linkIDname`: Variable name of the `"id"` variable in both `MicrobData` and `CovData`. The two data sets will be merged by this `"id"` variable.
- `allCov`: All covariates of interest (including confounders) for estimating and testing their associations with microbiome. Default is all covariates in covData are of interest.
- `refTaxa`: A vector of taxa names (or one taxon name) specified by the user and will be used as the reference taxa.
- `reguMethod`: regularization approach used in phase 1 of the algorithm. Take value `"mcp"` or `"lasso"`, default is `"mcp"`.
- `sequentialRun`: This takes a logical value `TRUE` or `FALSE`. Sometimes parallel jobs can not be successfully run for unknown reasons. For example, socket related errors may pop up or some slave cores return simple error instead of numerical results. In those scenarios, setting `sequentialRun = TRUE` may help, but it will take more time to run. Default is `TRUE` for the MZILN function since typically users should specify one or just a few reference taxa in `refTaxa`.

- `paraJobs`: If `sequentialRun` is `FALSE`, this specifies the number of parallel jobs that will be registered to run the algorithm. Default is `8`. If specified as `NULL`, it will automatically detect the cores to decide the number of parallel jobs.
- `standardize`: This takes a logical value `TRUE` or `FALSE`. If `TRUE`, all design matrix X in phase 1 and phase 2 will be standardized in the analyses. Default is `FALSE`.
- `bootB`: Number of bootstrap samples for obtaining confidence interval of estimates in phase 2. The default is `500`.
- `bootLassoAlpha`: The significance level in phase 2. Default is `0.05`.
- `seed`: Random seed for reproducibility. Default is `1`.

The output of `MZILN()` function is a list. The estimation results can extracted as the following:

- `analysisResults$estByCovList`: A list containing estimating results for all reference taxa and all the variables in `allCov`.

All covariates data can be extracted:

- `covariatesData`: A dataset containing covariates and confounders used in the analyses

Examples

We use the same example data The example dataset as that for illustrating the IFAA function. `dataM` and `dataC` are included in the package. They could be accessed by:

```
data(dataM)
dim(dataM)
#> [1] 20 60
dataM[1:5, 1:8]
#>   id rawCount1 rawCount2 rawCount3 rawCount4 rawCount5 rawCount6 rawCount7
#> 1  1         0         0         0         0         0         3         0
#> 2  2         0         0         0         0         0         0         0
#> 3  3         0         0         0         0         0       214         0
#> 4  4         0         0         0         0         0         2         0
#> 5  5         0         0         0         0         0        40         0

data(dataC)
dim(dataC)
#> [1] 20  6
dataC[1:5, ]
#>   id v4       v1 v5 v2 v3
#> 1  1  1 1.653901  4  1 NA
#> 2  2  2 0.362706  5  2  2
#> 3  3  1 1.496269 NA  5  2
#> 4  4  1 1.755541  5  3  3
#> 5  5  1 1.035714  5  7 NA
```

Both the microbiome data `dataM` and the covariates data `dataC` contain 20 samples (i.e., 20 rows).

- `dataM` contains 60 taxa with absolute abundances and these are gut microbiome.
- `dataC` contains 5 covariates.

Next we analyze the data to test the association between microbiome and all the three variables `"v1"`, `"v2"` and `"v3"`.

```
results <- MZILN(MicrobData = dataM,
                 CovData = dataC,
```

```r
                  linkIDname = "id",
                  allCov = c("v1","v2","v3"),
                  refTaxa=c("rawCount11")
                  )
#> There are 41 taxa without any sequencing reads and
#>         excluded from the analysis
#> Data dimensions (after removing missing data if any):
#> 13 samples
#> 18 OTU's or microbial taxa
#> 3 covariates in the analysis
#> [1] "These are the covariates:"
#> [1] "v1" "v2" "v3"
#> 0 binary covariates in the analysis
#> 54.27 percent of microbiome sequencing reads are zero
#> start Original screen
#>  6 slaves are spawned successfully. 0 failed.
#> OriginDataScreen parallel setup took 6.64 seconds
#> Original screen done and took 0.05283333 minutes
#> Reference taxa are: rawCount11
#> 8 parallel jobs are registered for bootstrp in Phase 2.
#> Estimation done for the 1 th reference taxon: rawCount11 and it took 0.124 minutes
#> The entire analysis took 0.2928333 minutes
```

In this example, we are only interested in testing the associations with `"v1"`, `"v2"` and '"v3"' which is why `allCov=c("v1,"v2","v3")`.

The final analysis results are stored in the list `results$analysisResults$estByRefTaxaList$rawCount11$estByCovList`:

```r
results$analysisResults$estByRefTaxaList$rawCount11$estByCovList
#> $v2
#>              Beta.LPR LowB95%CI.LPR UpB95%CI.LPR
#> rawCount29 0.03583529  -0.002122168   0.07067514
#> rawCount42 0.02566816  -0.017277108   0.06864125
#>
#> $v3
#>                 Beta.LPR LowB95%CI.LPR  UpB95%CI.LPR
#> rawCount6  -0.0034889572   -0.01529409  0.0054946220
#> rawCount29 -0.0041563989   -0.01549365  0.0046623846
#> rawCount32 -0.0006867174   -0.01186827  0.0106614073
#> rawCount42 -0.0089560683   -0.01974328 -0.0002497733
#> rawCount45 -0.0084721161   -0.01970520  0.0009583073
#> rawCount47 -0.0034332912   -0.01697852  0.0115896848
```

The results found the two taxa `"rawCount29"` and `"rawCount42"` associated with `"v2"`, and a bunch of other taxa assoicated with "'v3'". The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to α^k in the model equation, and can be interpreted as the associations between the covariates and log-ratio of the significant taxa over the reference taxon..

The interpretation is that

- Every unit increase in `"v2"` is associated with approximately 3.6% increase in the abundance ratio of `"rawCount29"` over '"rawCount11"' and approximately 2.6% increase in the abundance ratio of `"rawCount42"` over '"rawCount11"' in the entire gut ecosystem. The interpretation is similar for the associations with '"v3"'.
- There were no taxa associated with `"v1"` in the analysis.

All the analyzed covariates are stored in the object: `covariatesData`:

```
results$covariatesData
#>     id          v1 v2  v3
#> 2    2  0.36270596  2   2
#> 3    3  1.49626921  5   2
#> 4    4  1.75554095  3   3
#> 6    6  1.64525227  4   4
#> 8    8 -1.57781131 24  22
#> 9    9  2.22581203 55   5
#> 10  10  0.71642615 98  67
#> 12  12  2.12230160 98   3
#> 14  14  1.99387922 93   4
#> 16  16  0.05417617 83  34
#> 18  18 -0.43426021 73  67
#> 19  19  1.46579846 68 566
#> 20  20  1.89625949 63  34
```