Package 'miMeta'

May 1, 2024

Type Package

Title Meta-analysis of microbiome association studies

Version 0.1.0

Author Zhoujingpeng Wei, Guanhua Chen, Zheng-Zheng Tang

Maintainer Zhoujingpeng <zwei74@wisc.edu>

Description This R package implements the new methods for meta-analysis of microbiome association studies that respect the unique features of microbiome data such as compositionality.

Data 2024-05-01

Reference Wei Z, Chen G, Tang ZZ. Melody: Meta-analysis of Microbiome Association Studies for Discovering Generalizable Microbial Signatures. Submitted.

License GPL (>= 3) | file LICENSE

Imports MASS,
brglm2,
abess,
UpSetR,
dplyr,
ggplot2,
doParallel,
parallel,
foreach

RoxygenNote 7.2.3

Suggests knitr,
rmarkdown

BugReports https://github.com/ZjpWei/miMeta/issues

URL https://github.com/ZjpWei/miMeta

VignetteBuilder knitr

Encoding UTF-8

LazyData true

Depends R (>= 4.3.0)

R topics documented:

`CRC_data` *Datasets from two metagenomics studies of colorectal cancer (CRC)*

Description

The "CRC_data" includes "CRC_abd" and "CRC_meta". "CRC_abd" is a sample-by-feature matrix of relative abundance counts from the two studies including 267 species under order "Clostridiales". The "CRC_meta" is a data frame including the sample-level variables from the two studies.

Usage

```
data(CRC_data)
```

Format

An object of class `list` of length 2.

Source

<https://github.com/zellerlab/crc_meta>

References

Wirbel, Jakob et al. Nat Med. 2019 Apr;25(4):679-689.

Examples

```
library("miMeta")
data("CRC_data")
CRC_abd <- CRC_data$CRC_abd
CRC_meta <- CRC_data$CRC_meta
```

melody *Meta-analysis for selecting microbial signatures associated with covariate of interests.*

Description

Melody is a meta-analysis method designed to account for the unique features of compositional microbiome data in selecting microbial signatures.

Usage

```
melody(
  rel.abd,
  covariate.interest,
  covariate.adjust = NULL,
  cluster = NULL,
  depth.filter = 0,
  prev.filter = 0.1,
  ref = NULL,
  tune.path = c("gsection", "sequence"),
  tune.size.sequence = NULL,
  tune.size.range = NULL,
  tune.type = c("BIC", "HBIC", "KBIC", "EBIC"),
  tol = 0.001,
  NMAX = 20,
  parallel.core = NULL,
  output.best.one = TRUE,
  verbose = FALSE
)
```

Arguments

`rel.abd` A list of matrices for relative abundance counts. Each element of the list pertains to one study, and the name of each element must be the study ID. Each matrix must have sample IDs as row names and microbial feature IDs as column names. No missing value is allowed.

`covariate.interest` A list of matrices for single or multiple covariates of interest. Each element of the list pertains to one study, and the name of each element must be the study ID. In a matrix of a given study, each row is a sample, and each column is a covariate of interest and can only be a numeric vector. The set of covariates can be different among studies. In a given study, the order of samples should be matched with the order in "rel.abd". No missing value is allowed.

`covariate.adjust` A list of data frames for covariates that will be adjusted for in the model. Each element of the list pertains to one study, and the name of each element must be the study ID. In a data frame of a given study, each row is a sample, and each column is a covariate to be adjusted and can only be a factor or numeric vector. The set of covariates can be different among studies. In a given study, the order of samples should be matched with the order in "rel.abd". No missing value is allowed. Default is NULL (not adjusting for any covariates)

`cluster` A list of vectors of variable that define the sample cluster for studies with correlated samples. Each element of the list pertains to one study with correlated samples, and the name of each element must be the study ID. The order of samples should be matched with the order in "rel.abd" in the corresponding study. For example, the values of this variable are subject IDs if each subject has multiple correlated samples (e.g., measured in a longitudinal study). Default is NULL (all samples in all studies are independent, so no need to provide this input.).

`depth.filter` A cutoff value to remove samples with sequencing depth less than or equal to the cutoff. Default is 0.

`prev.filter` A cutoff value remove microbial features with prevalence (proportion of nonzero observations) less than or equal to the cutoff. This cutoff is applied to each study. So, a feature could be removed in a subset of the studies. The cutoff value must be in the range of 0-1. Default is 0.1.

`ref` A feature ID or a vector of feature IDs. If a feature ID is provided, this microbial feature will be used as the reference in all studies when generating summary statistics. If a vector of feature IDs is provided, each feature in the vector will be the reference for a study. Therefore, the length of the vector should be equal to the number of studies and have study IDs as names. Default is NULL (the function will automatically pick reference for each study).

`tune.path` Method to choose the optimal subset size in the best subset selection. If tune.path is "sequence", we solve the best subset selection problem for each size in "tune size sequence". If tune.path is "gsection", we solve the best subset selection problem with the size ranged in tune.size.range, where the considered size sequence within the range is determined by golden section search. Default is "gsection".

`tune.size.sequence`
An integer vector containing the subset sizes to be considered. Only used when tune.path is "sequence". Default is 0:round(K/2), where K is number of microbial features.

`tune.size.range`
An integer vector with two elements that define the range of the size. Default is c(0, K/2)

`tune.type` Type of information criterion for choosing the best subset size. Available options are "BIC", "HBIC", "KBIC" and "EBIC". Default is "BIC".

`tol` Convergence tolerance in the search for the best delta tuning parameters. Default is 1e-3.

`NMAX` The maximum number of iterations in the search for the best delta tuning parameters. Default is 20.

`parallel.core` The number of cores to be concurrently used for generating summary statistics. It's an integer between 1 and cl - 1 (cl is the total core number). Default is cl - 1.

`ouput.best.one` Whether only output the best model. Default is TRUE.

`verbose:` Whether to print verbose information. Default is FALSE. (see details in Value)

Details

Melody first generates summary statistics (microbiome association coefficient estimates and their variances) for individual studies. Melody then combines the summary statistics across studies to select microbial signatures. In particular, the selection of signature is operated through a best-subset selection. There are two sets of tuning parameters in the best subset selection including the subset

size (i.e. the number of microbial features selected) and delta's which are introduced to recover the absolute-abundance coefficients from the relative-abundance coefficients. (details in reference Wei et al.)

Value

Output a list with each component for a covariate of interest.

If output.best.one=TRUE (default), the component includes the following elements to deliver the meta-analysis results under the best model over the subset sizes considered.

`coef`	A coefficient vector that contains the meta association coefficient estimate (at the absolute-abundance level) under the best subset size. The vector names are microbial features IDs. The microbial features with nonzero coefficients are the selected signatures.
`delta`	A vector that contains the best values of the delta tuning parameters under the best subset size. The vector names are in the format "<study ID>_<reference microbial feature ID>"
`dev`	The value of the deviance for the best subset size.
`ic`	The value of the information criterion (the type of information criterion is specified in tune.type) for the best subset size.

If output.best.one=FALSE, the component includes the following elements to deliver the meta-analysis results under all models over the subset sizes considered. (depends on tune.path, see details in Arguments)

`coef`	A coefficient matrix with each column contains the meta association coefficient estimate (at the absolute-abundance level) under a subset size. The row names are microbial features IDs and the column names are the subset sizes considered.
`delta`	A matrix with each column contains the best values of the delta tuning parameters under a subset size. The row names are in the format "<study ID>_<reference microbial feature ID>" and the column names are the subset sizes considered.
`dev`	A vector contains the values of the deviance for the subset sizes considered.
`ic`	A vector contains the values of the information criterion (the type of information criterion is specified in tune.type) for the subset sizes considered.

If verbose=TRUE, print out information about the progress of the meta-analysis. In addition, it will generate an intersection plot showing the number of microbial features shared among studies and scatter plots for the meta-coefficient estimates for the top 5 covariates of interest associated with most microbial features.

Author(s)

Zhoujingpeng Wei, Guanhua Chen, Zheng-Zheng Tang

References

Wei Z, Chen G, Tang ZZ. Melody: Meta-analysis of Microbiome Association Studies for Discovering Generalizable Microbial Signatures. Submitted.

See Also

`melody.get.summary`, `melody.meta.summary`, `melody.merge.summary`

Examples

```
library("miMeta")
data("CRC_data")
CRC_abd <- CRC_data$CRC_abd
CRC_meta <- CRC_data$CRC_meta

########## Generate summary statistics ##########
rel.abd <- list()
covariate.interest <- list()
for(d in unique(CRC_meta$Study)){
  rel.abd[[d]] <- CRC_abd[CRC_meta$Sample_ID[CRC_meta$Study == d],]
  disease <- as.numeric(CRC_meta$Group[CRC_meta$Study == d] == "CRC")
  names(disease) <- CRC_meta$Sample_ID[CRC_meta$Study == d]
  covariate.interest[[d]] <- data.frame(disease = disease)
}

meta.result <- melody(rel.abd = rel.abd, covariate.interest = covariate.interest)
```

`melody.get.summary` *Generate Melody summary statistics for individual studies.*

Description

This function directly takes the output object from the "melody.null.model" function and constructs summary statistics for covariates of interest in each study. The output can be directly used by the function "melody.meta.summary" to perform meta-analysis.

Usage

```
melody.get.summary(
  null.obj,
  covariate.interest,
  cluster = NULL,
  parallel.core = NULL,
  verbose = FALSE
)
```

Arguments

`null.obj`	The output of function "melody.null.model".
`...`	See function melody. In covariate.interest and cluster, the order of samples should be matched with the order in rel.abd used in the "melody.null.model" function.

Value

Output a list with each component for a study. The component includes the following elements.

`ref`	Reference feature ID for the study.

est	A matrix contains the relative-abundance association coefficient estimates for the study. The row names are microbial feature IDs and the column names are the covariates of interest IDs.
var	A matrix contains the variances of the relative-abundance association coefficient estimates for the study. The row names are microbial feature IDs and the column names are the covariates of interest IDs.
n	Sample size for the study.

See Also

melody.null.model, melody.meta.summary, melody

Examples

```
library("miMeta")
data("CRC_data")
CRC_abd <- CRC_data$CRC_abd
CRC_meta <- CRC_data$CRC_meta

########## Generate summary statistics ##########
rel.abd <- list()
covariate.interest <- list()
for(d in unique(CRC_meta$Study)){
  rel.abd[[d]] <- CRC_abd[CRC_meta$Sample_ID[CRC_meta$Study == d],]
  disease <- as.numeric(CRC_meta$Group[CRC_meta$Study == d] == "CRC")
  names(disease) <- CRC_meta$Sample_ID[CRC_meta$Study == d]
  covariate.interest[[d]] <- data.frame(disease = disease)
}

 null.obj <- melody.null.model(rel.abd = rel.abd)

 summary.stats <- melody.get.summary(null.obj = null.obj, covariate.interest = covariate.interest)
```

---

melody.meta.summary *Meta-analyze summary statistics across studies*

Description

This function directly takes the summary statistics output from the "melody.get.summary" function and combines the summary statistics across studies for selecting microbial signatures associated with each covariate of interest.

Usage

```
melody.meta.summary(
  summary.stats,
  tune.path = c("gsection", "sequence"),
  tune.size.sequence = NULL,
  tune.size.range = NULL,
  tune.type = c("BIC", "HBIC", "KBIC", "EBIC"),
```

```
  output.best.one = TRUE,
  tol = 0.001,
  NMAX = 20,
  verbose = FALSE
)
```

Arguments

summary.stats	The output of function "melody.get.summary".
...	See function melody.

Value

Same output as the function "melody".

See Also

melody.null.model, melody.get.summary, melody

Examples

```
library("miMeta")
data("CRC_data")
CRC_abd <- CRC_data$CRC_abd
CRC_meta <- CRC_data$CRC_meta

########## Generate summary statistics ##########
rel.abd <- list()
covariate.interest <- list()
for(d in unique(CRC_meta$Study)){
  rel.abd[[d]] <- CRC_abd[CRC_meta$Sample_ID[CRC_meta$Study == d],]
  disease <- as.numeric(CRC_meta$Group[CRC_meta$Study == d] == "CRC")
  names(disease) <- CRC_meta$Sample_ID[CRC_meta$Study == d]
  covariate.interest[[d]] <- data.frame(disease = disease)
}

null.obj <- melody.null.model(rel.abd = rel.abd)

summary.stats <- melody.get.summary(null.obj = null.obj, covariate.interest = covariate.interest)

########## Meta-analysis ##########
meta.result <- melody.meta.summary(summary.stats = summary.stats)
```

---

melody.null.model *Get information from the null model.*

Description

The "melody.null.model" function computes the estimated mean microbial feature proportions and residuals of the relative abundance under the null model of no associations for each study (so this function does not need the covariate.interest information). The output of this function will be fed into the "melody.get.summary" function to construct summary statistics for covariates of interest in each study.

Usage

```
melody.null.model(
  rel.abd,
  covariate.adjust = NULL,
  depth.filter = 0,
  prev.filter = 0.1,
  ref = NULL,
  parallel.core = NULL
)
```

Arguments

...	See function melody.

Value

Output a list with each component for a study. The component includes the following elements.

ref	Reference feature ID for the study.
p	A matrix of the estimated mean microbial feature proportions for the study.
res	A matrix of the residuals of the relative abundance for the study.
N	A vector of microbiome sequencing depth for the study.
X	A data frame of the cleaned covariate.adjust for the study.
rm.sample.idx	The index of the removed samples for the study.

See Also

melody.get.summary, melody.meta.summary, melody

Examples

```
library("miMeta")
data("CRC_data")
CRC_abd <- CRC_data$CRC_abd
CRC_meta <- CRC_data$CRC_meta

########## Generate summary statistics ##########
rel.abd <- list()
for(d in unique(CRC_meta$Study)){
  rel.abd[[d]] <- CRC_abd[CRC_meta$Sample_ID[CRC_meta$Study == d],]
}

null.obj <- melody.null.model(rel.abd = rel.abd)
```