Package 'MicrobiomeStat'

January 9, 2026

Type Package

Title Statistical Methods for Microbiome Compositional Data

Version 1.3

Date 2026-01-05

Author Xianyang Zhang [aut],
Jun Chen [aut, cre],
Huijuan Zhou [ctb],
Linsui Deng [ctb]

Maintainer Jun Chen <chen.jun2@mayo.edu>

Description A suite of methods for powerful and robust microbiome data analysis addressing zero-inflation, phylogenetic structure and compositional effects. Includes the LinDA method for differential abundance analysis (Zhou et al. (2022)<doi:10.1186/s13059-022-02655-5>), the BMDD (Bimodal Dirichlet Distribution) method for accurate modeling and imputation of zero-inflated microbiome sequencing data (Zhou et al. (2025)<doi:10.1371/journal.pcbi.1013124>) and compositional sparse CCA methods for microbiome multi-omics data integration (Deng et al. (2024) <doi:10.3389/fgene.2024.1489694>).

Depends R (>= 3.5.0)

Imports ggplot2, matrixStats, parallel, stats, utils, Matrix, statmod, MASS, ggrepel, lmerTest, foreach, modeest, dplyr, mlrMBO, Rcpp, ParamHelpers, smoof, lhs, mlr, BBmisc

LinkingTo Rcpp, RcppArmadillo

Suggests DiceKriging, randomForest

NeedsCompilation yes

SystemRequirements NLopt library (optional, for high-performance BMDD mode)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Repository CRAN

Date/Publication 2026-01-09 00:00:02 UTC

Contents

---

bmdd *Bimodal Dirichlet Distribution Estimation*

Description

Estimates parameters of the bimodal Dirichlet distribution using a variational EM algorithm. Automatically selects the optimal implementation: high-performance C++ (NLopt) when possible, or R when covariates are needed.

Usage

```
bmdd(W, type = c("count", "proportion"),
     Z = NULL, formula.Z = NULL, U = NULL, formula.U = NULL,
     Z.standardizing = TRUE, U.standardizing = TRUE,
     alp.eta = FALSE, alp.kap = FALSE,
     pi.xi = FALSE, pi.zeta = FALSE,
     para.alp.init = NULL, para.pi.init = NULL, gam.init = NULL,
     iterlim = 500, tol = 1e-6, trace = FALSE,
     method = c("auto", "nlopt", "R"),
     inner.loop = TRUE, inner.iterlim = 20, inner.tol = 1e-6,
     alp.iterlim = 100, alp.tol = 1e-6,
     alp.min = 1e-3, alp.max = 1e3, ...)
```

Arguments

`W`	Matrix of observed data (m taxa × n samples)
`type`	Data type: "count" or "proportion"
`Z`	Optional matrix of covariates for alpha (forces R implementation)
`formula.Z`	Optional formula for Z covariates
`U`	Optional matrix of covariates for pi (forces R implementation)
`formula.U`	Optional formula for U covariates
`Z.standardizing`	Standardize Z covariates (default TRUE)

U.standardizing	Standardize U covariates (default TRUE)
alp.eta	Model alpha0 as function of Z (default FALSE)
alp.kap	Model alpha1 as function of Z (default FALSE)
pi.xi	Model pi as function of U (default FALSE)
pi.zeta	Model pi variance as function of U (default FALSE)
para.alp.init	Initial alpha values
para.pi.init	Initial pi values
gam.init	Initial gamma values
iterlim	Maximum iterations (default 500)
tol	Convergence tolerance (default 1e-6)
trace	Print progress (default FALSE)
method	Force method: "auto", "nlopt", or "R" (default "auto")
inner.loop	Use inner loop for NLopt (default TRUE)
inner.iterlim	Inner loop max iterations (default 20)
inner.tol	Inner loop tolerance (default 1e-6)
alp.iterlim	Alpha optimization iterations (default 100)
alp.tol	Alpha tolerance (default 1e-6)
alp.min	Minimum alpha (default 1e-3)
alp.max	Maximum alpha (default 1e3)
...	Additional arguments (ignored)

Details

Automatically chooses best implementation:

- **NLopt C++**: When no covariates. 50-90x faster using L-BFGS-B with analytical gradients. Scientifically equivalent to R ($r > 0.999$).
- **R**: When covariates needed or explicitly requested. Supports full covariate modeling.

Requires NLopt library for high-performance mode:

- macOS: `brew install nlopt`
- Linux: `sudo apt-get install libnlopt-dev`

Refer to https://github.com/zhouhj1994/BMDD for detailed examples about zero imputation and posterior sample generation.

Value

A list containing:

gamma	Estimated gamma parameters (bimodality indicators)
pi	Estimated pi parameters (mixing proportions)
beta	Estimated posterior Dirichlet parameters
alpha0	Estimated alpha0 parameters (mode 0)
alpha1	Estimated alpha1 parameters (mode 1)
converge	Logical indicating convergence
iter	Number of iterations
method	Method used: "nlopt" or "R"

References

Zhou, H., Chen, J., & Zhang, X. (2025). BMDD: A probabilistic framework for accurate imputation of zero-inflated microbiome sequencing data. PLoOS Computational Biology, 21(10), e1013124.

Examples

```
## Not run:
# Simulate data
m <- 100  # taxa
n <- 50   # samples
W <- matrix(rpois(m*n, 100), m, n)

# Auto-select method (uses NLopt for speed)
result <- bmdd(W, type = "count")

# Access results
head(result$beta)    # Posterior parameters
head(result$gamma)   # Bimodality indicators
result$method        # Shows which method was used

# Force specific method
result_nlopt <- bmdd(W, method = "nlopt")  # Force NLopt
result_r <- bmdd(W, method = "R")          # Force R

# With covariates (automatically uses R)
Z <- matrix(rnorm(m * 2), m, 2)
result_cov <- bmdd(W, Z = Z, alp.eta = TRUE)

## End(Not run)
```

`bmdd.nlopt` *NLopt C++ Implementation of BMDD*

Description

High-performance implementation using NLopt L-BFGS-B optimizer in C++. Provides 50-90x speedup over R implementation for cases without covariates. This is a low-level function; most users should use `bmdd()` instead.

Usage

```
bmdd.nlopt(W, type = c("count", "proportion"),
           para.alp.init = NULL, para.pi.init = NULL, gam.init = NULL,
           iterlim = 500, tol = 1e-6, trace = FALSE,
           inner.loop = TRUE, inner.iterlim = 20, inner.tol = 1e-6,
           alp.iterlim = 100, alp.tol = 1e-6,
           alp.min = 1e-3, alp.max = 1e3)
```

Arguments

`W`	Matrix of observed data (m taxa × n samples)
`type`	Data type: "count" or "proportion"
`para.alp.init`	Initial alpha values
`para.pi.init`	Initial pi values
`gam.init`	Initial gamma values
`iterlim`	Maximum iterations (default 500)
`tol`	Convergence tolerance (default 1e-6)
`trace`	Print progress (default FALSE)
`inner.loop`	Use inner loop optimization (default TRUE)
`inner.iterlim`	Inner loop max iterations (default 20)
`inner.tol`	Inner loop tolerance (default 1e-6)
`alp.iterlim`	Alpha optimization iterations (default 100)
`alp.tol`	Alpha tolerance (default 1e-6)
`alp.min`	Minimum alpha (default 1e-3)
`alp.max`	Maximum alpha (default 1e3)

Details

This function provides direct access to the NLopt C++ implementation. Most users should use `bmdd()` instead, which automatically selects the optimal implementation.

Does not support covariates. For covariate modeling, use `bmdd()` with method="R".

Value

A list containing estimated parameters (same as bmdd)

References

Zhou, H., Chen, J., & Zhang, X. (2025). BMDD: A probabilistic framework for accurate imputation of zero-inflated microbiome sequencing data. PLoOS Computational Biology, 21(10), e1013124.

See Also

bmdd for the main user interface

Examples

```
## Not run:
# Simulate data
m <- 100
n <- 50
W <- matrix(rpois(m*n, 100), m, n)

# Direct NLopt usage (advanced)
result <- bmdd.nlopt(W, type = "count")

# Recommended: use bmdd() instead
result <- bmdd(W, type = "count")  # auto-selects NLopt

## End(Not run)
```

---

cscca *Compositional Sparse Canonical Correlation Analysis*

Description

A compositional sparse canonical correlation analysis (csCCA) framework for integrating microbiome data with other high-dimensional omics data.

Usage

```
cscca(
  Y,
  View.ind,
  lambda.seq,
  a.old = NULL,
  View.type = NULL,
  eps.stop = 1e-04,
  max.step = 30,
  eps = 1e-04,
  T.step = 1
)
```

Arguments

Y	a n*(K*p) matrix representing the observations.
View.ind	a (K*p) integer vector indicating the classes of features. The features with the same View.ind is in the same class.
lambda.seq	a K vector consisting of hyper-parameters.
a.old	Optional initial value for the coefficient vector a.new.
View.type	a K vector encoding the structure type of each feature class. There are two choices: "O" (Omics Data),"C" (Compositional Data).
eps.stop	a numerical value controlling the convergence.
max.step	an integer controlling the maximum step for interaction.
eps	a numerical value controlling the convergence.
T.step	an integer controlling the maximum step for interaction.

Value

a.new the estimated coefficient vector.

References

1. Deng, L., Tang, Y., Zhang, X., et al. (2024). Structure-adaptive canonical correlation analysis for microbiome multi-omics data. Frontiers in Genetics, 15, 1489694.

2. Chen, J., Bushman, F. D., Lewis, J. D., et al. (2013). Structure-constrained sparse canonical correlation analysis with an application to microbiome data analysis. Biostatistics, 14(2), 244–258.

Examples

```
## Not run:
library(dplyr)

n <- 200
p <- q <- 100
sigma.nu <- 5
sigma.eps <- 1
omega_X <- 0.85*c(rep(1/10,9),-9/10,rep(0,p-10))
omega_Y <- 0.85*c(seq(0.08,0.12,length = 10),rep(0,q-10))
Data1 <- DGP_OC(seed=10,n,p,q,sigma.nu,sigma.eps,omega_X,omega_Y)

library(mlrMBO)
Res.sCCA.CV <- cscca.CV(Y=Data1$Y,View.ind=Data1$View.ind,
                        View.type=c("O","O"),
                        show.info = TRUE)


Res.CsCCA.CV <- cscca.CV(Y=Data1$Y,View.ind=Data1$View.ind,
                                  View.type=c("O","C"),
                                  show.info = TRUE)

Res.sCCA <- cscca(Y=Data1$Y,View.ind=Data1$View.ind,
```

```
                    lambda.seq=Res.sCCA.CV$lam.opt.trgt,
                    View.type=c("O","O"))
Res.CsCCA <- cscca(Y=Data1$Y,View.ind=Data1$View.ind,
                    lambda.seq=Res.CsCCA.CV$lam.opt.trgt,
                    View.type=c("O","C"))

## End(Not run)
```

---

cscca.CV *Compositional Sparse Canonical Correlation Analysis (Cross Valication Version)*

Description

The cross validation version of a compositional sparse canonical correlation analysis (sCCA) framework for integrating microbiome data with other high-dimensional omics data.

Usage

```
cscca.CV(
  Y,
  View.ind,
  View.type = NULL,
  eps.stop = 1e-04,
  max.step = 30,
  eps = 1e-04,
  T.step = 10,
  n_fold = 5,
  seed.sam.ind = NULL,
  show.info = FALSE,
  hp.lower = NULL,
  hp.upper = NULL,
  hp.eta.lower = NULL,
  hp.eta.upper = NULL,
  eta.warm.stat.mat = NULL,
  opt_n_design = 30,
  opt_n_iter = 20,
  Criterion = "cov",
  des.init = NULL,
  is.refit = F,
  is.refix.eta = T,
  opt_n_design.eta_warm = 30,
  opt_n_iter.eta_warm = 20,
  is.opt.hyper = TRUE,
  hyper_n_grid = 20,
  ...
)
```

Arguments

Y	a n*(K*p) matrix representing the observations.
View.ind	a (K*p) integer vector indicating the classes of features. The features with the same View.ind is in the same class.
View.type	a K vector encoding the structure type of each feature class. There are two choices: "O" (Omics Data),"C" (Compositional Data).
eps.stop	a numerical value controlling the convergence.
max.step	an integer controlling the maximum step for interaction.
eps	a numerical value controlling the convergence.
T.step	an integer controlling the maximum step for interaction.
n_fold	an integer representing the number of folds for cross validation.
seed.sam.ind	a vector of the seeds for sampling.
show.info	a bool suggesting whether to show information through the hyperparameter optimization.
hp.lower	a numerical value or K vector specifying the lower bound of the hyper-parameter.
hp.upper	a numerical value or K vector specifying the upper bound of the hyper-parameter.
hp.eta.lower	a numerical value or K vector specifying the lower bound of the hyper-parameter for eta.
hp.eta.upper	a numerical value or K vector specifying the upper bound of the hyper-parameter for eta.
eta.warm.stat.mat	a matrix providing statistics for warm start of eta.
opt_n_design	an integer controlling the number of design points in the hyperparameter optimization.
opt_n_iter	an integer controlling the number of iterations in the hyperparameter optimization.
Criterion	a character indicating the criterion we choose for cross validation.
des.init	an initial design for hyperparameter optimization.
is.refit	a bool suggesting whether to refit the model using the optimal hyper-parameters.
is.refix.eta	a bool suggesting whether eta is fixed during refitting.
opt_n_design.eta_warm	an integer controlling the number of design points for eta warm-start optimization.
opt_n_iter.eta_warm	an integer controlling the number of iterations for eta warm-start optimization.
is.opt.hyper	a bool suggesting whether to optimize the hyper-parameters.
hyper_n_grid	an integer controlling the grid size for hyperparameter search.
...	additional arguments passed to the internal optimization procedures.

Value

A list containing the following elements: (1) a.hat.opt.trgt: The coefficient vector estimated with the optimal hyper-parameter vector; (2) lam.opt.trgt: The optimal hyper-parameter vector.

References

1. Deng, L., Tang, Y., Zhang, X., et al. (2024). Structure-adaptive canonical correlation analysis for microbiome multi-omics data. Frontiers in Genetics, 15, 1489694.

2. Chen, J., Bushman, F. D., Lewis, J. D., et al. (2013). Structure-constrained sparse canonical correlation analysis with an application to microbiome data analysis. Biostatistics, 14(2), 244–258.

Examples

```
## Not run:
library(dplyr)

n <- 200
p <- q <- 100
sigma.nu <- 5
sigma.eps <- 1
omega_X <- 0.85*c(rep(1/10,9),-9/10,rep(0,p-10))
omega_Y <- 0.85*c(seq(0.08,0.12,length = 10),rep(0,q-10))
Data1 <- DGP_OC(seed=10,n,p,q,sigma.nu,sigma.eps,omega_X,omega_Y)

library(mlrMBO)
Res.sCCA.CV <- cscca.CV(Y=Data1$Y,View.ind=Data1$View.ind,
                        View.type=c("O","O"),
                        show.info = TRUE)


Res.CsCCA.CV <- cscca.CV(Y=Data1$Y,View.ind=Data1$View.ind,
                                  View.type=c("O","C"),
                                  show.info = TRUE)

Res.sCCA <- cscca(Y=Data1$Y,View.ind=Data1$View.ind,
                    lambda.seq=Res.sCCA.CV$lam.opt.trgt,
                    View.type=c("O","O"))
Res.CsCCA <- cscca(Y=Data1$Y,View.ind=Data1$View.ind,
                    lambda.seq=Res.CsCCA.CV$lam.opt.trgt,
                    View.type=c("O","C"))
print(Res.sCCA.CV$Cri.opt.trgt)
print(Res.CsCCA.CV$Cri.opt.trgt)

## End(Not run)
```

DGP_OC *Data Generating Process (Omics Data versus Compositional data)*

Description

Data Generating Process (Omics Data versus Compositional data)

Usage

```
DGP_OC(seed = 10, n, p, q, sigma.nu, sigma.eps, omega_X, omega_Y)
```

Arguments

seed	an integer for the initial seed.
n	an integer representing the sample size.
p	an integer representing the feature size of the omics dataset.
q	an integer representing the feature size of the compositional dataset.
sigma.nu	a numerial value representing the strength of correlation.
sigma.eps	a numerical value representing the strength of noise.
omega_X	a p vector representing the coefficient for the omics data.
omega_Y	a q vector representing the coefficient for the compositional data.

Value

A list containing the following elements: (a) Y: a n*(2p) matrix representing the full observations; (b) View.ind: a 2p integer vector indicating the classes of features. The features with the same View.ind is in the same class; (c) omega a 2p vector representing the true coefficients.

Examples

```
library(dplyr)
n <- 200
p <- q <- 100
sigma.nu <- 5
sigma.eps <- 1
omega_X <- 0.85*c(rep(1/10,9),-9/10,rep(0,p-10))
omega_Y <- 0.85*c(seq(0.08,0.12,length = 10),rep(0,q-10))
Data1 <- DGP_OC(seed=10,n,p,q,sigma.nu,sigma.eps,omega_X,omega_Y)
```

---

linda *Linear (Lin) Model for Differential Abundance (DA) Analysis of High-dimensional Compositional Data*

Description

The function implements a simple, robust and highly scalable approach to tackle the compositional effects in differential abundance analysis of high-dimensional compositional data. It fits linear regression models on the centered log2-ratio transformed data, identifies a bias term due to the transformation and compositional effect, and corrects the bias using the mode of the regression coefficients. It could fit mixed-effect models for analysis of correlated data.

Usage

```
linda(
  feature.dat,
  meta.dat,
  formula,
```

```
  feature.dat.type = c('count', 'proportion'),
  prev.filter = 0,
  mean.abund.filter = 0,
  max.abund.filter = 0,
  is.winsor = TRUE,
  outlier.pct = 0.03,
  adaptive = TRUE,
  zero.handling = c('pseudo-count', 'imputation'),
  pseudo.cnt = 0.5,
  corr.cut = 0.1,
  p.adj.method = "BH",
  alpha = 0.05,
  n.cores = 1,
  verbose = TRUE
)
```

Arguments

`feature.dat`	a matrix of counts/proportions, row - features (OTUs, genes, etc) , column - samples.
`meta.dat`	a data frame containing the sample meta data. If there are NAs, the corresponding samples will be removed in the analysis.
`formula`	a character string for the formula. The formula should conform to that used by `lm` (independent data) or `lmer` (correlated data). For example: `formula = '~x1*x2+x3+(1|id)'`. At least one fixed effect is required.
`feature.dat.type`	the type of the feature data. It could be "count" or "proportion".
`prev.filter`	the prevalence (percentage of non-zeros) cutoff, under which the features will be filtered. The default is 0.
`mean.abund.filter`	the mean relative abundance cutoff, under which the features will be filtered. The default is 0.
`max.abund.filter`	the max relative abundance cutoff, under which the features will be filtered. The default is 0.
`is.winsor`	a logical value indicating whether winsorization should be performed to replace outliers (high values). The default is TRUE.
`outlier.pct`	the expected percentage of outliers. These outliers will be winsorized. The default is 0.03.
`adaptive`	a logical value indicating whether the approach to handle zeros (pseudo-count or imputation) will be determined based on the correlations between the log(sequencing depth) and the explanatory variables in `formula` when `feature.dat` is 'count'. If TRUE and the correlation p-value for any explanatory variable is smaller than or equal to `corr.cut`, the imputation approach will be used; otherwise, the pseudo-count approach will be used.

`zero.handling` a character string of 'pseudo-count' or 'imputation' indicating the zero handling method used when `feature.dat` is 'count'. If 'pseudo-count', `apseudo.cnt` will be added to each value in `feature.dat`. If 'imputation', then we use the imputation approach using the formula in the referenced paper. Basically, zeros are imputed with values proportional to the sequencing depth. When `feature.dat` is 'proportion', this parameter will be ignored and zeros will be imputed by half of the minimum for each feature.

`pseudo.cnt` a positive numeric value for the pseudo-count to be added if `zero.handling` is 'pseudo-count'. Default is 0.5.

`corr.cut` a numerical value between 0 and 1, indicating the significance level used for determining the zero-handling approach when `adaptive` is TRUE. Default is 0.1.

`p.adj.method` a character string indicating the p-value adjustment approach for addressing multiple testing. See R function `p.adjust`. Default is 'BH'.

`alpha` a numerical value between 0 and 1 indicating the significance level for declaring differential features. Default is 0.05.

`n.cores` a positive integer. If `n.cores > 1` and formula is in a form of mixed-effect model, `n.cores` parallels will be conducted. Default is 1.

`verbose` a logical value indicating whether the trace information should be printed out.

Value

A list with the elements

`variables` A vector of variable names of all fixed effects in `formula`. For example: `formula = '~x1*x2+x3+(1|id)'`. Suppose `x1` and `x2` are numerical, and `x3` is a categorical variable of three levels: a, b and c. Then the elements of `variables` would be `('x1', 'x2', 'x3b', 'x3c', 'x1:x2')`.

`bias` numeric vector; each element corresponds to one variable in `variables`; the estimated bias of the regression coefficients due to the compositional effect.

`output` a list of data frames with columns 'baseMean', 'log2FoldChange', 'lfcSE', 'stat', 'pvalue', 'padj', 'reject', 'df'; `names(output)` is equal to `variables`; the rows of the data frame corresponds to taxa. Note: if there are taxa being excluded due to `prev.cut`, the number of the rows of the output data frame will be not equal to the number of the rows of `otu.tab`. Taxa are identified by the rownames. If the rownames of `otu.tab` are NULL, then `1 : nrow(otu.tab)` is set as the rownames of `otu.tab`.

baseMean: 2 to the power of the intercept coefficients (normalized by one million)

log2FoldChange: bias-corrected coefficients

lfcSE: standard errors of the coefficients

stat: `log2FoldChange / lfcSE`

pvalue: `2 * pt(-abs(stat), df)`

padj: `p.adjust(pvalue, method = p.adj.method)`

reject: `padj <= alpha`

df: degrees of freedom. The number of samples minus the number of explanatory variables (intercept included) for fixed-effect models; estimates from R package `lmerTest` with Satterthwaite method of approximation for mixed-effect models.

`covariance` a list of data frames; the data frame records the covariances between a regression coefficient with other coefficients; `names(covariance)` is equal to `variables`; the rows of the data frame corresponds to taxa. If the length of `variables` is equal to 1, then the `covariance` is NULL.

`otu.tab.use` the OTU table used in the abundance analysis (the `otu.tab` after the preprocessing: samples that have NAs in the variables in `formula` or have less than `lib.cut` read counts are removed; taxa with prevalence less than `prev.cut` are removed and data is winsorized if `!is.null(winsor.quan)`; and zeros are treated, i.e., imputed or pseudo-count added).

`meta.use` the meta data used in the abundance analysis (only variables in `formula` are stored; samples that have NAs or have less than `lib.cut` read counts are removed; numerical variables are scaled).

`wald` a list for use in Wald test. If the fitting model is a linear model, then it includes

beta: a matrix of the biased regression coefficients including intercept and all fixed effects; the culumns correspond to taxa

sig: the standard errors; the elements corresponding to taxa

X: the design matrix of the fitting model

bias: the estimated biases of the regression coefficients including intercept and all fixed effects

If the fitting model is a linear mixed-effect model, then it includes

beta: a matrix of the biased regression coefficients including intercept and all fixed effects; the culumns correspond to taxa

beta.cov: a list of covariance matrices of `beta`; the elements corresponding to taxa

rand.cov: a list with covariance matrices of variance parameters of random effects; the elements corresponding to taxa; see more details in the paper of 'lmerTest'

Joc.beta.cov.rand: a list of a list of Jacobian matrices of `beta.cov` with respect to the variance parameters; the elements corresponding to taxa

bias: the estimated biases of the regression coefficients including intercept and all fixed effects

Author(s)

Huijuan Zhou, Jun Chen, Xianyang Zhang

References

Zhou, H., He, K., Chen, J., Zhang, X. (2022). LinDA: linear models for differential abundance analysis of microbiome compositional data. Genome biology, 23(1), 95.

Examples

```
data(smokers)

ind <- smokers$meta$AIRWAYSITE == 'Throat'
otu.tab <- as.data.frame(smokers$otu[, ind])
depth <- colSums(otu.tab)
meta <- cbind.data.frame(Smoke = factor(smokers$meta$SMOKER[ind]),
                         Sex = factor(smokers$meta$SEX[ind]),
                         Site = factor(smokers$meta$SIDEOFBODY[ind]),
                         SubjectID = factor(smokers$meta$HOST_SUBJECT_ID[ind]))

# Differential abundance analysis using the left throat data
ind1 <- meta$Site == 'Left' & depth >= 1000
linda.obj  <- linda(otu.tab[, ind1], meta[ind1, ], formula = '~Smoke+Sex',
           feature.dat.type = 'count',
           prev.filter = 0.1, is.winsor = TRUE, outlier.pct = 0.03,
           p.adj.method = "BH", alpha = 0.1)




linda.plot(linda.obj, c('Smokey', 'Sexmale'),
           titles = c('Smoke: n v.s. y', 'Sex: female v.s. male'),
           alpha = 0.1, lfc.cut = 1, legend = TRUE, directory = NULL,
            width = 11, height = 8)

rownames(linda.obj $output[[1]])[which(linda.obj $output[[1]]$reject)]


# Differential abundance analysis pooling both the left and right throat data
# Mixed effects model is used

ind  <- depth >= 1000
linda.obj <- linda(otu.tab[, ind], meta[ind, ], formula = '~Smoke+Sex+(1|SubjectID)',
           feature.dat.type = 'count',
           prev.filter = 0.1, is.winsor = TRUE, outlier.pct = 0.03,
           p.adj.method = "BH", alpha = 0.1)



# For proportion data
otu.tab.p <- t(t(otu.tab) / colSums(otu.tab))
ind1 <- meta$Site == 'Left' & depth >= 1000
lind.obj  <- linda(otu.tab[, ind1], meta[ind1, ], formula = '~Smoke+Sex',
           feature.dat.type = 'proportion',
           prev.filter = 0.1, is.winsor = TRUE, outlier.pct = 0.03,
           p.adj.method = "BH", alpha = 0.1)
```

`linda.plot` *Plot LinDA Results*

Description

The function produces the effect size plot of the differential features and volcano plot based on the output from `linda`.

Usage

```
linda.plot(
  linda.obj,
  variables.plot,
  titles = NULL,
  alpha = 0.05,
  lfc.cut = 1,
  legend = FALSE,
  directory = NULL,
  width = 11,
  height = 8
)
```

Arguments

`linda.obj`	return from function `linda`.
`variables.plot`	vector; variables whose results are to be plotted. For example, suppose the return value `variables` is equal to `('x1', 'x2', 'x3b', 'x3c', 'x1:x2')`, then one could set `variables.plot = c('x3b', 'x1:x2')`.
`titles`	vector; titles of the effect size plot and volcano plot for each variable in `variables.plot`. Default is NULL. If NULL, the titles will be set as `variables.plot`.
`alpha`	a numerical value between 0 and 1; cutoff for `padj`.
`lfc.cut`	a positive numerical value; cutoff for `log2FoldChange`.
`legend`	TRUE or FALSE; whether to show the legends of the effect size plot and volcano plot.
`directory`	character; the directory to save the figures, e.g., `getwd()`. Default is NULL. If NULL, figures will not be saved.
`width`	the width of the graphics region in inches. See R function `pdf`.
`height`	the height of the graphics region in inches. See R function `pdf`.

Value

A list of `ggplot2` objects.

`plot.lfc`	a list of effect size plots. Each plot corresponds to one variable in `variables.plot`.
`plot.volcano`	a list of volcano plots. Each plot corresponds to one variable in `variables.plot`.

Author(s)

Huijuan Zhou, Jun Chen, Xianyang Zhang

References

Zhou, H., He, K., Chen, J., Zhang, X. (2022). LinDA: linear models for differential abundance analysis of microbiome compositional data. Genome biology, 23(1), 95.

Examples

```
data(smokers)
ind <- smokers$meta$AIRWAYSITE == 'Throat' & smokers$meta$SIDEOFBODY == 'Left'
otu.tab <- as.data.frame(smokers$otu[, ind])
depth <- colSums(otu.tab)
meta <- cbind.data.frame(Smoke = factor(smokers$meta$SMOKER[ind]),
                         Sex = factor(smokers$meta$SEX[ind]))

ind  <- depth >= 1000
linda.obj <- linda(otu.tab[, ind], meta[ind, ], formula = '~Smoke+Sex',
           feature.dat.type = 'count',
           prev.filter = 0.1, is.winsor = TRUE, outlier.pct = 0.03,
           p.adj.method = "BH", alpha = 0.1)

linda.plot(linda.obj, c('Smokey', 'Sexmale'),
           titles = c('Smoke: n v.s. y', 'Sex: female v.s. male'),
           alpha = 0.1, lfc.cut = 1, legend = TRUE, directory = NULL,
            width = 11, height = 8)
```

---

linda.wald.test *Wald test for bias-corrected regression coefficients*

Description

The function implements Wald test for bias-corrected regression coefficients generated from the linda function. One can utilize the function to perform ANOVA-type analyses. For example, if you have a cateogrical variable with three levels, you can test whether all levels have the same effect.

Usage

```
linda.wald.test(
  linda.obj,
  L,
  model = c("LM", "LMM"),
  alpha = 0.05,
  p.adj.method = "BH"
)
```

Arguments

linda.obj	return from the linda function.

L	A matrix for testing Lb = 0, where b includes the intercept and all fixed effects from runing linda. Thus the number of columns of L must be equal to length(variables)+1, where variables is from linda.obj, which does not include the intercept.
model	'LM' or 'LMM' indicating the model fitted in {linda} is linear model or linear mixed-effect model.
alpha	significance level for testing Lb = 0.
p.adj.method	P-value adjustment approach. See R function p.adjust. The default is 'BH'.

Value

A data frame with columns

Fstat	Wald statistics for each taxon
df1	The numerator degrees of freedom
df2	The denominator degrees of freedom
pvalue	1 - pf(Fstat, df1, df2)
padj	p.adjust(pvalue, method = p.adj.method)
reject	padj <= alpha

Author(s)

Huijuan Zhou <huijuanzhou2019@gmail.com> Jun Chen <Chen.Jun2@mayo.edu> Xianyang Zhang <zhangxiany@stat.tamu.edu>

References

Zhou, H., He, K., Chen, J., Zhang, X. (2022). LinDA: linear models for differential abundance analysis of microbiome compositional data. Genome biology, 23(1), 95.

Examples

```
data(smokers)

ind <- smokers$meta$AIRWAYSITE == 'Throat'
otu.tab <- as.data.frame(smokers$otu[, ind])
depth <- colSums(otu.tab)
meta <- cbind.data.frame(Smoke = factor(smokers$meta$SMOKER[ind]),
                         Sex = factor(smokers$meta$SEX[ind]),
                         Site = factor(smokers$meta$SIDEOFBODY[ind]),
                         SubjectID = factor(smokers$meta$HOST_SUBJECT_ID[ind]))
ind  <- depth >= 1000
linda.obj <- linda(otu.tab[, ind], meta[ind, ], formula = '~Smoke+Sex+(1|SubjectID)',
           feature.dat.type = 'count',
           prev.filter = 0.1, is.winsor = TRUE, outlier.pct = 0.03,
           p.adj.method = "BH", alpha = 0.1)
# L matrix (2x3) is designed to test the second (Smoke) and the third (Sex) coefficient to be 0.
# For a categorical variable > two levels, similar L can be designed to do ANOVA-type test.
```

```
L <- matrix(c(0, 1, 0, 0, 0, 1), nrow = 2, byrow = TRUE)
result <- linda.wald.test(linda.obj, L, 'LMM', alpha = 0.1)
```

`smokers` *Microbiome data from the human upper respiratory tract (left and right throat)*

Description

A dataset containing "otu", "tax", meta", "genus", family"

Usage

```
data(smokers)
```

Format

A list with elements

otu otu table, 2156 taxa by 290 samples

tax taxonomy table, 2156 taxa by 7 taxonomic ranks

meta meta table, 290 samples by 53 sample variables

genus 304 by 290

family 113 by 290

Source

https://qiita.ucsd.edu/ study ID:524 Reference: Charlson ES, Chen J, Custers-Allen R, Bittinger K, Li H, et al. (2010) Disordered Microbial Communities in the Upper Respiratory Tract of Cigarette Smokers. PLoS ONE 5(12): e15216.

Index