Package ‘MMUPHin’

March 23, 2024

Type Package

Title Meta-analysis Methods with Uniform Pipeline for Heterogeneity in Microbiome Studies

Version 1.16.0

Author Siyuan Ma

Maintainer Siyuan MA <syma.research@gmail.com>

Description MMUPHin is an R package for meta-analysis tasks of microbiome cohorts. It has function interfaces for:
   a) covariate-controlled batch- and cohort effect adjustment,
   b) meta-analysis differential abundance testing,
   c) meta-analysis unsupervised discrete structure (clustering) discovery, and
   d) meta-analysis unsupervised continuous structure discovery.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.1

VignetteBuilder knitr

SystemRequirements glpk (>= 4.57)

Depends R (>= 3.6)

Imports Maaslin2, metafor, fpc, igraph, ggplot2, dplyr, tidyr, stringr, cowplot, utils, stats, grDevices

Suggests testthat, BiocStyle, knitr, rmarkdown, magrittr, vegan, phyloseq, curatedMetagenomicData, genefilter

biocViews Metagenomics, Microbiome, BatchEffect

git_url https://git.bioconductor.org/packages/MMUPHin

git_branch RELEASE_3_18

git_last_commit 61bc5dc

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-03-22
R topics documented:

- add_back_covariates .............................................. 3
- adjust_batch ....................................................... 3
- adjust_EB .......................................................... 5
- aprior .............................................................. 5
- AST ................................................................. 6
- back_transform_abd ............................................... 6
- bprior .............................................................. 7
- catchToList .......................................................... 7
- check_batch ......................................................... 8
- check_covariates .................................................. 8
- check_covariates_random ........................................ 9
- check_D ............................................................ 9
- check_exposure .................................................... 10
- check_feature_abd ............................................... 10
- check_metadata ................................................... 11
- check_options ..................................................... 11
- check_options_continuous ..................................... 12
- check_pseudo_count ............................................ 12
- check_rank ......................................................... 13
- check_samples .................................................. 13
- check_samples_D ............................................... 14
- construct_design ............................................... 14
- construct_ind .................................................... 15
- continuous_discover .......................................... 15
- CRC_abd .......................................................... 17
- CRC_meta .......................................................... 18
- create_table_maaslin ......................................... 20
- diagnostic_adjust_batch ...................................... 20
- diagnostic_continuous_discover .............................. 21
- diagnostic_discrete_discover ................................. 21
- discrete_discover ............................................... 22
- fill_dimnames ..................................................... 23
- fit_EB .............................................................. 24
- fit_shrink .......................................................... 24
- fit_stand_feature ............................................... 25
- it_sol ............................................................... 25
- lm_meta ............................................................ 26
- LOG ................................................................. 26
- Maaslin2_wrapper ............................................... 29
- match_control .................................................... 29
- normalize_features ............................................. 30
- relocate_scale .................................................. 30
- rename_maaslin ................................................ 31
- rma_wrapper ...................................................... 32
- set_pseudo ......................................................... 32
- shorten_name ..................................................... 33
add_back_covariates

Add back covariate effects to batch-corrected feature abundance data

Description

Add back covariate effects to batch-corrected feature abundance data

Usage

add_back_covariates(adj_data, l_stand_feature, l_ind)

Arguments

adj_data feature-by-sample matrix of batch-adjusted feature abundances (but without covariate effects), as returned by relocate_scale.

l_stand_feature list of per-feature standardization fits, as returned by fit_stand_feature.

l_ind list of indicator matrices, as returned by construct_ind.

Value

feature-by-sample matrix of batch-adjusted feature abundances with covariate effects retained.

adjust_batch

Zero-inflated empirical Bayes adjustment of batch effect in compositional feature abundance data

Description

adjust_batch takes as input a feature-by-sample matrix of microbial abundances, and performs batch effect adjustment given provided batch and optional covariate variables. It returns the batch-adjusted abundance matrix. Additional options and parameters can be passed through the control parameter as a list (see details).

Usage

adjust_batch(feature_abd, batch, covariates = NULL, data, control)
adjust_batch

Arguments

feature_abd feature-by-sample matrix of abundances (proportions or counts).
batch name of the batch variable. This variable in data should be a factor variable and will be converted to so with a warning if otherwise.
covariates name(s) of covariates to adjust for in the batch correction model.
data data frame of metadata, columns must include batch and covariates (if specified).
control a named list of additional control parameters. See details.

Details

control should be provided as a named list of the following components (can be a subset).

zero_inflation logical. Indicates whether or not a zero-inflated model should be run. Default to TRUE (zero-inflated model). If set to FALSE then the correction will be similar to ComBat as provided in the sva package.
pseudo_count numeric. Pseudo count to add feature_abd before the methods’ log transformation. Default to NULL, in which case adjust_batch will set the pseudo count automatically to half of minimal non-zero values in feature_abd.
diagnostic_plot character. Name for the generated diagnostic figure file. Default to "adjust_batch_diagnostic.pdf". Can be set to NULL in which case no output will be generated.
maxit integer. Maximum number of iterations allowed for the method’s iterative algorithm. Default to 1000.
verbose logical. Indicates whether or not verbose information will be printed.

Value

a list, with the following components:

feature_abd_adj feature-by-sample matrix of batch-adjusted abundances, normalized to the same per-sample total abundance as feature_abd.
control list of additional control parameters used in the function call.

Author(s)

Siyuan Ma, <siyuanma@g.harvard.edu>

Examples

data("CRC_abd", "CRC_meta")
CRC_abd_adj <- adjust_batch(feature_abd = CRC_abd,
  batch = "studyID",
  covariates = "study_condition",
  data = CRC_meta)$feature_abd_adj
**adjust_EB**

*Perform batch adjustment on standardized feature abundances, based on EB shrinked per-batch location and scale parameters*

**Description**

Perform batch adjustment on standardized feature abundances, based on EB shrinked per-batch location and scale parameters

**Usage**

```r
adjust_EB(s_data, l_params_shrink, l_stand_feature, batchmod, n_batch, l_ind)
```

**Arguments**

- `s_data`: feature-by-sample matrix of standardized abundances.
- `l_params_shrink`: list of shrinked parameters, as returned by fit_shrink.
- `l_stand_feature`: list of per-feature standardization fits, as returned by fit_stand_feature.
- `batchmod`: design matrix for batch variables.
- `n_batch`: number of batches in the data.
- `l_ind`: list of indicator matrices, as returned by construct_ind.

**Value**

feature-by-sample matrix of batch-adjusted feature abundances.

**aprior**

*EB prior estimation for scale parameters*

**Description**

EB prior estimation for scale parameters

**Usage**

```r
aprior(delta_hat, na.rm = FALSE)
```

**Arguments**

- `delta_hat`: frequentist per-batch scale estimations.
- `na.rm`: whether or not missing values should be removed.
**Value**

shape hyper parameter

---

**AST**

*AST transformation (modified from Maaslin2 and is different)*

---

**Description**

AST transformation (modified from Maaslin2 and is different)

**Usage**

`AST(x)`

**Arguments**

- `x` vector of abundance to be transformed.

**Value**

transformed vector of abundance.

---

**back_transform_abd**

*Transform batch adjusted feature abundances back to the original scale in feature_abd*

---

**Description**

Transform batch adjusted feature abundances back to the original scale in feature_abd

**Usage**

`back_transform_abd(adj_data, feature_abd, type_feature_abd)`

**Arguments**

- `adj_data` feature-by-sample matrix of batch-adjusted feature abundances with covariate effects retained.
- `feature_abd` original feature-by-sample matrix of abundances (proportions or counts).
- `type_feature_abd` type of feature abundance table (counts or proportions). If counts, the final output will be rounded into counts as well.

**Value**

feature-by-sample matrix of batch-adjusted feature abundances, with covariate effects retained and scales consistent with original abundance matrix.
**$bprior$**

**Description**

EB prior estimation for scale parameters

**Usage**

```
bprior(delta_hat, na.rm = FALSE)
```

**Arguments**

- `delta_hat`: frequentist per-batch location estimations.
- `na.rm`: whether or not missing values should be removed.

**Value**

scale hyper parameter

---

**catchToList**

**Utility for catching warning/error messages**

**Description**

Utility for catching warning/error messages

**Usage**

```
catchToList(expr)
```

**Arguments**

- `expr`: an expression to run that can generate potential errors/warnings

**Value**

a list, capturing both the return value of the expression, as well as generated errors/warnings (NULL if no errors/warnings)
### check_batch

**Check batch variable**

**Description**
Check batch variable

**Usage**

\[\text{check\_batch}(x, \text{min\_n\_batch} = 2)\]

**Arguments**

- **x**: batch variable.
- **min\_n\_batch**: min. number of batches (for MMUPHin functions to run).

**Value**
if no errors then the batch variables (factorized if not already)

### check_covariates

**Check covariates**

**Description**
Check covariates

**Usage**

\[\text{check\_covariates}(\text{data\_covariates}, \text{batch})\]

**Arguments**

- **data\_covariates**: data frame of covariates.
- **batch**: batch variable.

**Value**
vector of indicators per batch for if/which covariates can be fitted within the batches
check_covariates_random

Check random covariates

Description

Check random covariates

Usage

check_covariates_random(data_covariates, batch)

Arguments

data_covariates
data frame of random covariates.
batch batch variable.

Value

vector of indicators per batch for if/which random covariates can be fitted within the batches

check_D

Check dissimilarity object

Description

Make sure that the input is a dissimilarity object

Usage

check_D(D)

Arguments

D dissimilarity object.

Value

returns an error if D is not a dissimilarity. Otherwise D as a matrix.
**check_exposure**  
*Check exposure variable*

**Description**
Check exposure variable

**Usage**
```
check_exposure(exposure, batch)
```

**Arguments**
- `exposure`  exposure variable.
- `batch`  batch variable.

**Value**
vector of indicators per batch for whether or not the exposure can be fitted within the batches

---

**check_feature_abd**  
*Check feature abundance table*

**Description**
Given a feature abundance table, make sure that a) it has no missing values, b) all values are non-negative, c) it is either proportions (all no greater than 1) or counts (all integers).

**Usage**
```
check_feature_abd(feature_abd)
```

**Arguments**
- `feature_abd`  feature-by-sample matrix of abundances (proportions or counts).

**Value**
returns an error if any of the check fails. Otherwise either "counts" or "proportions"
**check_metadata**

*Check that metadata data frame has all the variables and not missing*

**Description**

Check that metadata data frame has all the variables and not missing

**Usage**

`check_metadata(data, variables, no_missing = TRUE)`

**Arguments**

- `data` data frame of metadata.
- `variables` name of variables (batch, covariates, etc.) to check

**Value**

data reduced to include only those specified in variables

---

**check_options**

*Utility for checking options*

**Description**

Utility for checking options

**Usage**

`check_options(x, x_name, options)`

**Arguments**

- `x` the specified value
- `x_name` name of the specified value
- `options` allowed options

**Value**

error if x is not in options. Otherwise returns x.
check_options_continuous

*Utility for checking continuous options*

**Description**
Utility for checking continuous options

**Usage**

```r
check_options_continuous(x, x_name, range)
```

**Arguments**

- `x` : the specified numeric value
- `x_name` : name of the specified value
- `range` : allowed range

**Value**

error if `x` is not within `range` (boundaries excluded). Otherwise returns `x`.

---

check_pseudo_count

*Utility for checking pseudo count*

**Description**
Utility for checking pseudo count

**Usage**

```r
check_pseudo_count(x)
```

**Arguments**

- `x` : the specified pseudo count

**Value**

error if pseudo count is smaller than zero. Otherwise returns `x`. 
check_rank

Check if a design matrix is full rank

Description
Check if a design matrix is full rank

Usage
check_rank(design)

Arguments
design design matrix.

Value
TRUE/FALSE for whether or not the design matrix is full rank.

check_samples

Check that sample numbers and names match between a feature table and a metadata data frame

Description
Sample names (column names of the feature table, row names of the metadata data frame) must be matching exactly. Note that this dictates that they cannot be NULL because by design data (a data frame) should have non-empty row names.

Usage
check_samples(feature_abd, data)

Arguments
feature_abd feature-by-sample matrix of abundances (proportions or counts).
data data frame of metadata.

Value
matched sample names
### check_samples_D

Check that sample numbers and names match between a dissimilarity matrix and a metadata data frame

**Description**

Sample names (row/column names of the D matrix, row names of the metadata data frame) must be matching exactly. Note that this dictates that they cannot be NULL because by design data (a data frame) should have non-empty row names.

**Usage**

```r
check_samples_D(D, data)
```

**Arguments**

- `D` sample-by-sample matrix of dissimilarities (proportions or counts).
- `data` data frame of metadata.

**Value**

matched sample names

### construct_design

Construct a design model matrix given a metadata data frame, with the option to exclude the intercept.

**Description**

Construct a design model matrix given a metadata data frame, with the option to exclude the intercept.

**Usage**

```r
construct_design(data, with_intercept = TRUE)
```

**Arguments**

- `data` metadata data frame.
- `with_intercept` should intercept terms be included in the model

**Value**

design matrix.
construct_ind

Create indicator matrices for which feature/batch/samples to adjust. This is relevant for zero_inflation is TRUE and only non-zero values are adjusted.

Description

Create indicator matrices for which feature/batch/samples to adjust. This is relevant for zero_inflation is TRUE and only non-zero values are adjusted.

Usage

```r
construct_ind(feature_abd, n_batch, design, zero_inflation)
```

Arguments

- `feature_abd`: feature-by-sample matrix of abundances (proportions or counts).
- `n_batch`: number of batches in the data.
- `design`: design matrix.
- `zero_inflation`: zero inflation flag.

Value

- list of indicator matrices needed by fitting in adjust_batch.

continuous_discover

Unsupervised meta-analytical discovery and validation of continuous structures in microbial abundance data

Description

`continuous_discover` takes as input a feature-by-sample matrix of microbial abundances. It first performs unsupervised continuous structure discovery (PCA) within each batch. Loadings of top PCs from each batch are then mapped against each other to identify "consensus" loadings that are reproducible across batches with a network community discovery approach with `igraph`. The identified consensus loadings/scores can be viewed as continuous structures in microbial profiles that are recurrent across batches and valid in a meta-analytical sense. `continuous_discover` returns, among other output, the identified consensus scores for continuous structures in the provided microbial abundance profiles, as well as the consensus PC loadings which can be used to assign continuous scores to any sample with the same set of microbial features.

Usage

```r
continuous_discover(feature_abd, batch, data, control)
```
Arguments

feature_abd feature-by-sample matrix of abundances (proportions or counts).
batch name of the batch variable. This variable in data should be a factor variable and will be converted to so with a warning if otherwise.
data data frame of metadata, columns must include batch.
control a named list of additional control parameters. See details.

Details

control should be provided as a named list of the following components (can be a subset).

  normalization character. Similar to the normalization parameter in Maaslin2 but only "TSS" and "NONE" are allowed. Default to "TSS" (total sum scaling).
  transform character. Similar to the transform parameter in Maaslin2 but only "AST" and "LOG" are allowed. Default to "AST" (arcsine square root transformation).
  pseudo_count numeric. Pseudo count to add feature_abd before the transformation. Default to NULL, in which case pseudo count will be set automatically to 0 if transform="AST", and half of minimal non-zero values in feature_abd if transform="LOG".
  var_perc_cutoff numeric. A value between 0 and 1 that indicates the percentage variability explained to cut off at for selecting top PCs in each batch. Across batches, the top PCs that in total explain more than var_perc_cutoff of the total variability will be selected for meta-analytical continuous structure discovery. Default to 0.8 (PCs included need to explain at least 80 total variability).
  cos_cutoff numeric. A value between 0 and 1 that indicates cutoff for absolute cosine coefficients between PC loadings to construct the method’s network with. Once the top PC loadings from each batch are selected, cosine coefficients between each loading pair are calculated which indicate their similarity. Loading pairs with absolute cosine coefficients surpassing cos_cutoff are then considered as associated with each other, and represented as an edge between the pair in a PC loading network. Network community discovery can then be performed on this network to identified densely connected “clusters” of PC loadings, which represent meta-analytically recurrent continuous structures.
  cluster_function function. cluster_function is used to perform community structure discovery in the constructed PC loading network. This can be any of the network cluster functions provided in igraph. Default to cluster_optimal. Note that this option can be slow for larger datasets, in which case cluster_fast_greedy is recommended.
  network_plot character. Name for the generated network figure file. Default to "clustered_network.pdf". Can be set to NULL in which case no output will be generated.
  plot_size_cutoff integer. Clusters with sizes smaller than or equal to plot_size_cutoff will be excluded in the visualized network. Default to 2 - visualized clusters must have at least three nodes (PC loadings).
  diagnostic_plot character. Name for the generated diagnostic figure file. Default to "continuous_diagnostic.pdf". Can be set to NULL in which case no output will be generated.
  verbose logical. Indicates whether or not verbose information will be printed.
Value

- `consensus_scores` matrix of identified consensus continuous scores. Columns are the identified consensus scores and rows correspond to samples in `feature_abd`.
- `consensus_loadings` matrix of identified consensus loadings. Columns are the identified consensus scores and rows correspond to features in `feature_abd`.
- `mat_vali` matrix of validation cosine coefficients of the identified consensus loadings. Columns correspond to the identified consensus scores and rows correspond to batches.
- `network, communities, mat_cos` components for the constructed PC loading network and community discovery results. `network` is a `igraph` graph object for the constructed network of associated PC loadings. `communities` is a `communities` object for the identified consensus loading clusters in `network` (output from `control$cluster_function`). `mat_cos` is the matrix of cosine coefficients between all selected top PCs from all batches.
- `control` list of additional control parameters used in the function call.

Author(s)

Siyuan Ma, <siyuanma@g.harvard.edu>

Examples

```r
data("CRC_abd", "CRC_meta")
fit_continuous <- continuous_discover(feature_abd = CRC_abd, 
  batch = "studyID", 
  data = CRC_meta)
```

Description

Species level relative abundance profiles of CRC and control patients in the five public studies used in Thomas et al. (2019). These were accessed through `curatedMetagenomicData`.

Usage

```r
data(CRC_abd)
```

Format

A feature-by-sample matrix of species-level profiles

Source

`curatedMetagenomicData`
References


Examples

data(CRC_abd)
# features included
rownames(CRC_abd)
# These are relative abundances
apply(CRC_abd, 2, sum)
# The following were used to generate the object
# library(curatedMetagenomicData)
# library(phylseq)
# library(genefilter)
# datasets <- curatedMetagenomicData(
#   c("FengQ_2015.metaphlan_bugs_list.stool" ,
#      "HanniganGD_2017.metaphlan_bugs_list.stool",
#      "VogtmannE_2016.metaphlan_bugs_list.stool",
#      "YuJ_2015.metaphlan_bugs_list.stool",
#      "ZellerG_2014.metaphlan_bugs_list.stool"),
#   dryrun = FALSE)
# Construct phylseq object from the five datasets
# physeq <-
#   # Aggregate the five studies into ExpressionSet
#   mergeData(datasets) %>%
#   # Convert to phylseq object
#   ExpressionSet2phyloseq() %>%
#   # Subset samples to only CRC and controls
#   subset_samples(study_condition %in% c("CRC", "control")) %>%
#   # Subset features to species
#   subset_taxa(!is.na(Species) & is.na(Strain)) %>%
#   # Normalize abundances to relative abundance scale
#   transform_sample_counts(function(x) x / sum(x)) %>%
#   # Filter features to be of at least 1e-5 relative abundance in five
#   # samples
#   filter_taxa(kOverA(5, 1e-5), prune = TRUE)
# CRC_abd <- otu_table(physeq)$Data

---

CRC_meta

Sample metadata of five public CRC studies

Description

Metadata information of CRC and control patients in the five public studies used in Thomas et al. (2019). These were accessed through curatedMetagenomicData.
Usage

data(CRC_meta)

Format

A data.frame of per-sample metadata information

Source

curatedMetagenomicData

References


Examples

data(CRC_meta)
# has CRC and control samples across five studies
table(CRC_meta$studyID, CRC_meta$study_condition)
# The following were used to generate the object
# library(curatedMetagenomicData)
# library(phyloseq)
# library(genefilter)
# datasets <- curatedMetagenomicData(
#   c("FengQ_2015.metaphlan_bugs_list.stool" ,
#   "HanniganGD_2017.metaphlan_bugs_list.stool",
#   "VogtmannE_2016.metaphlan_bugs_list.stool",
#   "YuJ_2015.metaphlan_bugs_list.stool",
#   "ZellerG_2014.metaphlan_bugs_list.stool"),
#   dryrun = FALSE)
# Construct phyloseq object from the five datasets
# physeq <-
#   # Aggregate the five studies into ExpressionSet
#   mergeData(datasets) %>%
#   # Convert to phyloseq object
#   ExpressionSet2phyloseq() %>%
#   # Subset samples to only CRC and controls
#   subset_samples(study_condition %in% c("CRC", "control")) %>%
#   # Subset features to species
#   subset_taxa(!is.na(Species) & is.na(Strain)) %>%
#   # Normalize abundances to relative abundance scale
#   transform_sample_counts(function(x) x / sum(x)) %>%
#   # Filter features to be of at least 1e-5 relative abundance in five samples
#   filter_taxa(kOverA(5, 1e-5), prune = TRUE)
# CRC_meta <- data.frame(sample_data(physeq))
# CRC_meta$studyID <- factor(CRC_meta$studyID)
create_table_maaslin  
Utility for generating empty Maaslin2 results table

Description
Utility for generating empty Maaslin2 results table

Usage
create_table_maaslin(features, exposure, lvl_exposure)

Arguments
- **features**: name of the features fitted to Maaslin2.
- **exposure**: the exposure variable.
- **lvl_exposure**: levels of the exposure variable, if a factor.

Value
a table for each feature-exposure value pair; reference level of exposure, if a factor, is taken out because is absorbed into the intercept term in Maaslin2 regression

---

diagnostic_adjust_batch

Diagnostic visualization for adj_batch function

Description
Diagnostic visualization for adj_batch function

Usage
diagnostic_adjust_batch(feature_abd, feature_abd_adj, var_batch, gamma_hat, gamma_star, output)

Arguments
- **feature_abd**: original feature-by-sample matrix of abundances (proportions or counts).
- **feature_abd_adj**: feature-by-sample matrix of batch-adjusted feature abundances, with covariate effects retained and scales consistent with original abundance matrix.
- **var_batch**: the batch variable (should be a factor).
- **gamma_hat**: estimated per feature-batch gamma parameters.
- **gamma_star**: shrinked per feature-batch gamma parameters.
- **output**: output file name
**diagnostic_continuous_discover**

*Diagnostic visualization for continuous.discover function*

**Value**

(invisibly) the ggplot2 plot object

**Usage**

```r
diagnostic_continuous_discover(mat_vali, lvl_batch, cos_cutoff, output)
```

**Arguments**

- `mat_vali` : matrix of maximum correlations between the cluster-specific consensus loadings and top PC loadings from each batch
- `lvl_batch` : unique batches in the data
- `cos_cutoff` : the specified cosine coefficient cutoff
- `output` : output file name

**Value**

the invisible ggplot2 plot object

**diagnostic_discrete_discover**

*Diagnostic visualization for discrete.discover function*

**Description**

Diagnostic visualization for discrete.discover function

**Usage**

```r
diagnostic_discrete_discover(stats_internal, stats_external, lvl_batch, output)
```

**Arguments**

- `stats_internal` : list of internal evaluation summary statistics
- `stats_external` : list of external validation summary statistics
- `lvl_batch` : unique batches in the data
- `output` : output file name
discrete_discover

Unsupervised meta-analytical discovery and validation of discrete clustering structures in microbial abundance data

Description

discrete_discover takes as input sample-by-sample dissimilarity measurements (generated from microbial abundance profiles), and performs unsupervised clustering within each batch across a range of cluster numbers. It then evaluates the support for each cluster number with both internal (i.e., samples within the batch) and external (i.e., samples in other batches) data. Internal evaluation is realized with prediction.strength and external evaluation is based on a generalized version of the same method. discrete_discover generates as output the evaluation statistics for each cluster number. A cluster number with good support from both internal and external evaluations provides meta-analytical evidence for discrete structures in the microbial abundance profiles.

Usage

discrete_discover(D, batch, data, control)

Arguments

D sample-by-sample dissimilarity measurements. Should be provided as a dist object.
batch name of the batch variable. This variable in data should be a factor variable and will be converted to so with a warning if otherwise.
data data frame of metadata, columns must include batch.
control a named list of additional control parameters. See details.

Details

control should be provided as a named list of the following components (can be a subset).

k_max integer. Maximum number of clusters to evaluate. discrete_discover will evaluate clustering structures corresponding to cluster numbers ranging from 2 to k_max. Default to 10.

cluster_function an interface function. This function will be used for unsupervised clustering for discrete structure evaluation. This corresponds to the clustermethod parameter in prediction.strength, and similarly, should also follow the specifications as detailed in clusterboot. Default to claraCBI

classify_method character. Classification method used to assign observations in the method’s internal and external evaluation stage. Corresponds to the classification parameter in prediction.strength, and can only be either “centroid” or “knn”. Default to “centroid”.

M integer. Number of random iterations to partition the batch during method’s internal evaluation. Corresponds to the M parameter in prediction.strength. Default to 30.
fill_dimnames

**nnk** integer. Number of nearest neighbors if classify_method="knn". Corresponds to the nnk parameter in prediction.strength. Default to 1.

**diagnostic_plot** character. Name for the generated diagnostic figure file. Default to "discrete_diagnostic.pdf". Can be set to NULL in which case no output will be generated.

**verbose** logical. Indicates whether or not verbose information will be printed.

**Value**

a list, with the following components:

- **internal_mean**, **internal_se** matrices of internal clustering structure evaluation measurements (prediction strengths). Columns and rows corresponds to different batches and different numbers of clusters, respectively. **internal_mean** and **internal se**, as the names suggest, are the mean and standard error of prediction strengths for each batch/cluster number.

- **external_mean**, **external_se** same structure as **internal_mean** and **internal_se**, but records external clustering structure evaluation measurements (generalized prediction strength).

- **control** list of additional control parameters used in the function call.

**Author(s)**

Siyuan Ma, <siyuanma@g.harvard.edu>

**Examples**

```r
data("CRC_abd", "CRC_meta")
# Calculate Bray-Curtis dissimilarity between the samples
library(vegan)
D <- vegdist(t(CRC_abd))
fit_discrete <- discrete_discover(D = D,
                                batch = "studyID",
                                data = CRC_meta)
```

**fill_dimnames**

*Fill in artificial row/column names to a matrix or data frame, if they are missing*

**Description**

Fill in artificial row/column names to a matrix or data frame, if they are missing

**Usage**

```
fill_dimnames(x, row_prefix, col_prefix)
```

**Arguments**

- **x** matrix or data frame
- **row_prefix** prefix for the artificial row names
- **col_prefix** prefix for the artificial column names
**Value**

x but with the missing dimension names filled in

---

**fit_EB**

*Parametric estimation of per-batch location and scale parameters, and Empirical Bayes estimation of their priors*

---

**Description**

Parametric estimation of per-batch location and scale parameters, and Empirical Bayes estimation of their priors

**Usage**

```
fit_EB(s_data, l_stand_feature, batchmod, n_batch, l_ind)
```

**Arguments**

- `s_data` feature-by-sample matrix of standardized abundances.
- `l_stand_feature` list of per-feature standardization fits, as returned by `fit_stand_feature`.
- `batchmod` design matrix for batch variables.
- `n_batch` number of batches in the data.
- `l_ind` list of indicator matrices, as returned by `construct_ind`.

**Value**

list of parameter estimations.

---

**fit_shrink**

*A posteriori shrink per-batch location and scale parameters towards their EB priors*

---

**Description**

A posteriori shrink per-batch location and scale parameters towards their EB priors

**Usage**

```
fit_shrink(s_data, l_params, batchmod, n_batch, l_ind, control)
```
Arguments

- **s_data**: feature-by-sample matrix of standardized abundances.
- **l_params**: list of parameter fits, as returned by fit_EB.
- **batchmod**: design matrix for batch variables.
- **n_batch**: number of batches in the data.
- **l_ind**: list of indicator matrices, as returned by construct_ind.
- **control**: list of control parameters (passed on to it_sol)

Value

list of shrinked per-batch location and scale parameters.

**fit_stand_feature**: *Fit lm and standardize all features*

Description

Fit lm and standardize all features

Usage

```r
fit_stand_feature(s_data, design, l_ind)
```

Arguments

- **s_data**: feature-by-sample matrix of abundances (proportions or counts).
- **design**: design matrix.
- **l_ind**: list of indicator matrices, as returned by construct_ind.

Value

list of two componet: the standardized feature abundance matrix, and a list of per-feature standard-
ization fits.
**it_sol**

*Iteratively solve for one feature’s shrinked location and scale parameters*

**Description**

Iteratively solve for one feature’s shrinked location and scale parameters

**Usage**

```r
it_sol(s_data, g_hat, d_hat, g_bar, t2, a, b, control)
```

**Arguments**

- `s_data` : the feature’s standardized abundances.
- `g_hat` : the feature’s location parameter frequentist estimations.
- `d_hat` : the feature’s scale parameter frequentist estimations.
- `g_bar` : EB estimation of location hyper parameters.
- `t2` : EB estimation of location hyper parameters.
- `a` : EB estimation of scale hyper parameters.
- `b` : EB estimation of scale hyper parameters.
- `control` : list of control parameters

**Value**

matrix of shrinked location and scale parameters.

---

**lm_meta**

*Covariate adjusted meta-analytical differential abundance testing*

**Description**

`lm_meta` runs differential abundance models on microbial profiles within individual studies/batches, and aggregates per-batch effect sizes with a meta-analysis fixed/random effects model. It takes as input a feature-by-sample microbial abundance table and the accompanying meta data data frame which should includes the batch indicator variable, the main exposure variable for differential abundance testing, and optional covariates and random covariates. The function first runs Maaslin2 models on the exposure with optional covariates/random covariates in each batch. The per-batch effect sizes are then aggregated with rma.uni and reported as output. Additional parameters, including those for both Maaslin2 and rma.uni can be provided through control (see details).
Usage

```r
lm_meta(
  feature_abd,  
  batch,        
  exposure,     
  covariates = NULL, 
  covariates_random = NULL, 
  data,         
  control
)
```

Arguments

- `feature_abd` feature-by-sample matrix of abundances (proportions or counts).
- `batch` name of the batch variable. This variable in data should be a factor variable and will be converted to so with a warning if otherwise.
- `exposure` name of the exposure variable for differential abundance testing.
- `covariates` names of covariates to adjust for in Maaslin2 differential abundance testing models.
- `covariates_random` names of random effects grouping covariates to adjust for in Maaslin2 differential abundance testing models.
- `data` data frame of metadata, columns must include exposure, batch, and covariates and covariates_random (if specified).
- `control` a named list of additional control parameters. See details.

Details

- `control` should be provided as a named list of the following components (can be a subset).

  - `normalization` character. `normalization` parameter for Maaslin2. See `Maaslin2` for details and allowed values. Default to "TSS" (total sum scaling).
  - `transform` character. `transform` parameter for Maaslin2. See `Maaslin2` for details and allowed values. Default to "AST" (arcsine square root transformation).
  - `analysis_method` character. `analysis_method` parameter for Maaslin2. See `Maaslin2` for details and allowed values. Default to "LM" (linear modeling).
  - `rma_method` character. `method` parameter for rma.uni. See `rma.uni` for details and allowed values. Default to "REML" (restricted maximum-likelihood estimator).
  - `output` character. Output directory for intermediate Maaslin2 output and the optional forest plots. Default to "MMUPHin_lm_meta".
  - `forest_plot` character. Suffix in the name for the generated forest plots visualizing significant meta-analytical differential abundance effects. Default to "forest.pdf". Can be set to NULL in which case no output will be generated.
  - `rma_conv` numeric. Convergence threshold for rma.uni (corresponds to control$threshold. See `rma.uni` for details. Default to 1e-4.
rma_maxit  integer. Maximum number of iterations allowed for rma.uni (corresponds to control$maxiter. See rma.uni for details. Default to 1000.

verbose  logical. Indicates whether or not verbose information will be printed.

Value

a list, with the following components:

meta_fits data frame of per-feature meta-analytical differential abundance results, including columns for effect sizes, p-values and q-values, heterogeneity statistics such as $\tau^2$ and $I^2$, as well as weights for individual batches. Many of these statistics are explained in detail in rma.uni.

maaslin_fits list of data frames, each one corresponding to the fitted results of Maaslin2 in an individual batch. See Maaslin2 on details of these output.

control list of additional control parameters used in the function call.

Author(s)

Siyuan Ma, <siyuanma@g.harvard.edu>

Examples

data("CRC_abd", "CRC_meta")
fit_meta <- lm_meta(feature_abd = CRC_abd, exposure = "study_condition", batch = "studyID", covariates = c("gender", "age"), data = CRC_meta$meta_fits

LOG transformation (modified from Maaslin2 and is different)

Description

LOG transformation (modified from Maaslin2 and is different)

Usage

LOG(x)

Arguments

x vector of abundance to be transformed.

Value

transformed vector of abundance.
Maaslin2_wrapper  

Wrapper function for Maaslin2

Description

Wrapper function for Maaslin2

Usage

Maaslin2_wrapper(feature_abd, data, exposure, covariates = NULL, covariates_random = NULL, output = tempdir(), normalization = "TSS", transform = "AST", analysis_method = "LM")

Arguments

- `feature_abd`: feature*sample matrix of feature abundance.
- `data`: data frame of metadata.
- `exposure`: name of exposure variable.
- `covariates`: name of covariates.
- `covariates_random`: name of random covariates.
- `output`: directory for Maaslin2.
- `normalization`: normalization parameter for Maaslin2.
- `transform`: transformation parameter for Maaslin2.
- `analysis_method`: analysis method parameter for Maaslin2.

Value

A data frame recording per-feature coefficients, p-values, etc. from running Maaslin2.

match_control  

Match user-specified control parameters with default, and modify if needed

Description

Match user-specified control parameters with default, and modify if needed

Usage

match_control(default, control)
Arguments

default : list of default control parameters
control : list of user-provided control parameters

Value

list of control parameters, set to user provided values if specified and default otherwise

---

normalize_features

*Normalize feature abundance table (modified from Maaslin2)*

Description

Normalize feature abundance table (modified from Maaslin2)

Usage

`normalize_features(features, normalization = "NONE", pseudo_count = 0)`

Arguments

- **features**: feature-by-sample matrix of abundances (proportions or counts).
- **normalization**: normalization method.
- **pseudo_count**: pseudo count to be added to feature_abd.

Value

normalized abundance table.

---

relocate_scale

*Relocate and scale feature abundances to correct for batch effects, given shrinked per-batch location and scale parameters*

Description

Relocate and scale feature abundances to correct for batch effects, given shrinked per-batch location and scale parameters

Usage

`relocate_scale(s_data, l_params_shrink, batchmod, n_batch, l_ind)`
rename_maaslin

Arguments

s_data feature-by-sample matrix of standardized abundances.
1_params_shrink list of shinked parameters, as returned by fit_shrink.
batchmod design matrix for batch variables.
n_batch number of batches in the data.
l_ind list of indicator matrices, as returned by construct_ind.

Value

feature-by-sample matrix of batch-adjusted feature abundances (but without covariate effects).

Description

Utility for temporarily renaming samples/features for Maaslin2 run to bypass the rare cases where unconventional names can cause exceptions.

Usage

rename_maaslin(old_names, prefix)

Arguments

old_names vector of names.
prefix prefix for the replacement (new numbered names).

Value

vector of new names - numbered vector with same length as old names and with the specified prefix.
### rma_wrapper

**Wrapper for fitting fixed/random effects meta-analysis model using metafor**

### Description

Wrapper for fitting fixed/random effects meta-analysis model using metafor

### Usage

```r
rma_wrapper(maaslin_fits, method = "REML", output = tempdir(),
forest_plot = NULL, rma_conv = 1e-06, rma_maxit = 1000,
verbose = TRUE)
```

### Arguments

- **maaslin_fits**: list of Maaslin2 result data frames, outputted from Maaslin2Wrapper.
- **method**: meta-analysis model to run, options provided in metafor::rma.
- **output**: directory for the output forest plots.
- **forest_plot**: logical. should forest plots be generated for the significant associations.
- **rma_conv**: rma threshold control.
- **rma_maxit**: rma maximum iteration control.
- **verbose**: should verbose information be printed.

### Value

a data frame recording per-feature meta-analysis association results. (coefficients, p-values, etc.)

### set_pseudo

**Set pseudo count for an abundance matrix. Pseudo count is currently set to half of minimum non-zero values**

### Description

Set pseudo count for an abundance matrix. Pseudo count is currently set to half of minimum non-zero values

### Usage

```r
set_pseudo(features)
```

### Arguments

- **features**: feature-by-sample matrix of abundances (proportions or counts).
shorten_name

Value

the pseudo count

Description

Utility for shorter names Useful when plotting per-feature figures where feature names could be cutoff

Usage

shorten_name(x, cutoff = 3, replacement = "..")

Arguments

x vector of names
cutoff number of maximum string length before start cutting off the middle

Value

vector of new names with .. replacing the middle part if name is longer than cutoff

standardize_feature

Centralize (by design matrix) and standardize (by pooled variance across all batches) feature abundances for empirical Bayes fit

Description

Centralize (by design matrix) and standardize (by pooled variance across all batches) feature abundances for empirical Bayes fit

Usage

standardize_feature(y, i_design, n_batch)

Arguments

y vector of non-zero abundance of a single feature (if zero-inflated is true).
i_design design matrix for the feature; samples with zeros are taken out (if zero-inflated is true).
n_batch number of batches in the data.
Value

a list with component: y_stand for vector of centralized and standardized feature abundance, and stand_mean/varpooled for the location and scale factor (these are used later to back transform the batch-shrinked feature abundance).

---

**transform_features**  
*Transform feature abundance table (modified from Maaslin2)*

---

**Description**

Transform feature abundance table (modified from Maaslin2)

**Usage**

transform_features(features, transform = "NONE", pseudo_count = 0)

**Arguments**

- features: feature-by-sample matrix of abundances (proportions or counts).
- transform: transformation method.
- pseudo_count: pseudo count to be added to feature_abundance.

**Value**

transformed abundance table.

---

**TSS**  
*TSS normalization (modified from Maaslin2)*

---

**Description**

TSS normalization (modified from Maaslin2)

**Usage**

TSS(x)

**Arguments**

x: vector of abundance to be normalized.

**Value**

normalized vector of abundance.
Species level feature abundance data of two public vaginal studies

Description

Species level relative abundance profiles of vaginal samples in the two public studies provided in curatedMetagenomicData.

Usage

data(vaginal_abd)

Format

A feature-by-sample matrix of species-level profiles

Source

curatedMetagenomicData

References


Examples

data(vaginal_abd)
# features included
rownames(vaginal_abd)
# These are relative abundances
apply(vaginal_abd, 2, sum)
# The following were used to generate the object
# library(curatedMetagenomicData)
# library(phyloseq)
# datasets <- curatedMetagenomicData( # "*metaphlan_bugs_list.vagina*", # dryrun = FALSE)
# Construct phyloseq object from the five datasets
# physeq <-
# Aggregate the five studies into ExpressionSet
# mergeData(datasets) %>%
# Convert to phyloseq object
# ExpressionSet2phyloseq() %>%
# Subset features to species
# subset_taxa(!is.na(Species) & is.na(Strain)) %>%
# Normalize abundances to relative abundance scale
# transform_sample_counts(function(x) x / sum(x)) %>%
### vaginal_meta

Sample metadata of two public vaginal studies

#### Description

Metadata information of vaginal samples in the two public studies provided in `curatedMetagenomicData`.

#### Usage

```r
data(vaginal_meta)
```

#### Format

A `data.frame` of per-sample metadata information

#### Source

`curatedMetagenomicData`

#### References


#### Examples

```r
data(vaginal_meta)
# has vaginal samples across two studies
table(vaginal_meta$studyID, vaginal_meta$body_site)
# The following were used to generate the object
# library(curatedMetagenomicData)
# library(phyloseq)
# datasets <- curatedMetagenomicData(  
#   "*metaphlan_bugs_list.vagina*",
#   dryrun = FALSE)
# Construct phyloseq object from the five datasets
# physeq <-
#   # Aggregate the five studies into ExpressionSet
#   mergeData(datasets) %>%
#   # Convert to phyloseq object
#   ExpressionSet2phyloseq() %>%
#   # Subset features to species
#   subset_taxa(!is.na(Species) & is.na(Strain)) %>%
#   # Normalize abundances to relative abundance scale
#   transform_sample_counts(function(x) x / sum(x)) %>%
```
# Filter features to be of at least 1e-5 relative abundance in two samples
# filter_taxa(kOverA(2, 1e-5), prune = TRUE)
# vaginal_meta <- data.frame(sample_data(physeq))
# vaginal_meta$studyID <- factor(vaginal_meta$studyID)

visualize_continuous_discover

Visualization of the clustered network for the continuous.discover function

Description

Visualization of the clustered network for the continuous.discover function

Usage

visualize_continuous_discover(graph_pc, membership_loading, size_communities, plot_size_cutoff, short_names, output)

Arguments

graph_pc: the full pc network constructed from correlated PCs
membership_loading: membership of PC loadings from community discovery
size_communities: ordered (largest to smallest) size of the identified communities
plot_size_cutoff: cluster size cutoff (for cluster to be included in the visualized PC network)
short_names: shorter names of the loadings
output: output file name

Value

an invisible list of the subsetted network and memberships (to reproduce the plot)
Index

* datasets
  CRC_abd, 17
  CRC_meta, 18
  vaginal_abd, 35
  vaginal_meta, 36

* internal
  add_back_covariates, 3
  adjust_EB, 5
  aprior, 5
  AST, 6
  back_transform_abd, 6
  bprior, 7
  catchToList, 7
  check_batch, 8
  check_covariates, 8
  check_covariates_random, 9
  check_D, 9
  check_exposure, 10
  check_feature_abd, 10
  check_metadata, 11
  check_options, 11
  check_options_continuous, 12
  check_pseudo_count, 12
  check_rank, 13
  check_samples, 13
  check_samples_D, 14
  construct_design, 14
  construct_ind, 15
  create_table_maaslin, 20
  diagnostic_adjust_batch, 20
  diagnostic_continuous_discover, 21
  diagnostic_discrete_discover, 21
  fill_dimnames, 23
  fit_EB, 24
  fit_shrink, 24
  fit_stand_feature, 25
  it_sol, 26
  LOG, 28
  Maaslin2_wrapper, 29
  match_control, 29
  normalize_features, 30
  relocate_scale, 30
  rename_maaslin, 31
  rma_wrapper, 32
  set_pseudo, 32
  shorten_name, 33
  standardize_feature, 33
  transform_features, 34
  TSS, 34
  visualize_continuous_discover, 37

  add_back_covariates, 3
  adjust_batch, 3
  adjust_EB, 5
  aprior, 5
  AST, 6
  back_transform_abd, 6
  bprior, 7
  catchToList, 7
  check_batch, 8
  check_covariates, 8
  check_covariates_random, 9
  check_D, 9
  check_exposure, 10
  check_feature_abd, 10
  check_metadata, 11
  check_options, 11
  check_options_continuous, 12
  check_pseudo_count, 12
  check_rank, 13
  check_samples, 13
  check_samples_D, 14
  claraCBI, 22
  cluster_fast_greedy, 16
  cluster_optimal, 16
  clusterboot, 22
  communities, 17
construct_design, 14
construct_ind, 15
continuous_discover, 15
CRC_abd, 17
CRC_meta, 18
create_table_maaslin, 20
curatedMetagenomicData, 17–19, 35, 36
diagnostic_adjust_batch, 20
diagnostic_continuous_discover, 21
diagnostic_discrete_discover, 21
discrete_discover, 22
dist, 22
fill_dimnames, 23
fit_EB, 24
fit_shrink, 24
fit_stand_feature, 25
it_sol, 26
lm_meta, 26
LOG, 28
Maaslin2, 16, 26–28
Maaslin2_wrapper, 29
match_control, 29
normalize_features, 30
prediction.strength, 22, 23
relocate_scale, 30
rename_maaslin, 31
rma.uni, 26–28
rma_wrapper, 32
set_pseudo, 32
shorten_name, 33
standardize_feature, 33
transform_features, 34
TSS, 34
vaginal_abd, 35
vaginal_meta, 36
visualize_continuous_discover, 37