Package ‘MMUPHin’

April 12, 2022

Type Package

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

Version 1.8.2

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 6.1.1

VignetteBuilder knitr

Depends R (>= 3.6)

Imports Maaslin2, metafor, fpc, igraph, ggplot2, dplyr, tidyr, 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_14

git_last_commit 6c224c9

git_last_commit_date 2022-03-31

Date/Publication 2022-04-12

R topics documented:

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)
```

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.

conv numeric. Convergence threshold for the method's iterative algorithm for shrinking batch effect parameters. Default to 1e-4.

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
```

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-analyitical 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

```
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. Defaul 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

a list, with the following components:

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

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

`CRC_abd` *Species level feature abundance data of five public CRC studies*

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

```
data(CRC_abd)
```

Format

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

Source

`curatedMetagenomicData`

References

Thomas, Andrew Maltez, Paolo Manghi, Francesco Asnicar, Edoardo Pasolli, Federica Armanini, Moreno Zolfo, Francesco Beghini et al. "Metagenomic analysis of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation." Nature medicine 25, no. 4 (2019): 667.

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(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_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

Thomas, Andrew Maltez, Paolo Manghi, Francesco Asnicar, Edoardo Pasolli, Federica Armanini, Moreno Zolfo, Francesco Beghini et al. "Metagenomic analysis of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation." Nature medicine 25, no. 4 (2019): 667.

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)
```

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.

nnk integer. Numbber 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

```
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)
```

---

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

```
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 `"LOG"` (log 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"` (estricted 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-analyitical 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-analyitical differential abundance results, including columns for effect sizes, p-values and q-values, heterogeneity statistics such as τ^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 a 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
```

---

vaginal_abd *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

Pasolli, Edoardo, Lucas Schiffer, Paolo Manghi, Audrey Renson, Valerie Obenchain, Duy Tin Truong, Francesco Beghini et al. "Accessible, curated metagenomic data through ExperimentHub." Nature methods 14, no. 11 (2017): 1023.

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)) %>%
  # Filter features to be of at least 1e-5 relative abundance in two samples
#   filter_taxa(kOverA(2, 1e-5), prune = TRUE)
# vaginal_abd <- otu_table(physeq)@.Data
```

vaginal_meta *Sample metadata of two public vaginal studies*

Description

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

Usage

```
data(vaginal_meta)
```

Format

A data.frame of per-sample metadata information

Source

curatedMetagenomicData

References

Pasolli, Edoardo, Lucas Schiffer, Paolo Manghi, Audrey Renson, Valerie Obenchain, Duy Tin Truong, Francesco Beghini et al. "Accessible, curated metagenomic data through ExperimentHub." Nature methods 14, no. 11 (2017): 1023.

Examples

```
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)
```

Index