Package: Macarron (via r-universe)

October 8, 2026

Version 1.17.0

Title Prioritization of potentially bioactive metabolic features from epidemiological and environmental metabolomics datasets

Depends R (>= 4.5.0), SummarizedExperiment

Imports BiocParallel, DelayedArray, WGCNA, ff, data.table, dynamicTreeCut, Maaslin2, plyr, stats, psych, logging, methods, utils

Suggests knitr, BiocStyle, optparse, testthat (>= 2.1.0), rmarkdown, markdown

Description Macarron is a workflow for the prioritization of potentially bioactive metabolites from metabolomics experiments. Prioritization integrates strengths of evidences of bioactivity such as covariation with a known metabolite, abundance relative to a known metabolite and association with an environmental or phenotypic indicator of bioactivity. Broadly, the workflow consists of stratified clustering of metabolic spectral features which co-vary in abundance in a condition, transfer of functional annotations, estimation of relative abundance and differential abundance analysis to identify associations between features and phenotype/condition.

VignetteBuilder knitr

License MIT + file LICENSE

URL http://huttenhower.sph.harvard.edu/macarron

Encoding UTF-8

biocViews Sequencing, Metabolomics, Coverage, FunctionalPrediction, Clustering

BugReports https://forum.biobakery.org/c/microbial-community-profiling/macarron

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libicu-dev libuv1-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-04-28 12:58:20 UTC

RemoteUrl https://github.com/bioc/Macarron

RemoteRef HEAD

RemoteSha 9abb534c5e7039be874bdd89d1645805e23fdcb8

Contents

calAVA 2
calES 3
calQval 4
chem_taxonomy 6
decorateID 6
findMacMod 7
Macarron 8
makeDisMat 11
prepInput 12
prioritize 13
showBest 14

Index **16**

`calAVA` *Calculate abundance versus anchor (AVA) of metabolic features.*

Description

AVA of a feature is the ratio of its abundance and the most abundant metabolite in the same module i.e. the "anchor". Anchor is an annotated/known feature if available or just the most abundant metabolic feature. For every feature, mean abundance in each phenotype or condition is calculated and the maximum is considered for AVA calculation. Singletons are assigned an AVA of 1.

Usage

```
calAVA(se, mod.assn, metadata_variable = 1, anchor_annotation = 2)
```

Arguments

`se`	SummarizedExperiment object created using Macarron::prepInput().
`mod.assn`	the output of Macarron::findMacMod().
`metadata_variable`	name or index of metadata column identifying phenotypes/conditions to be used for evaluating AVA. Default: Column 1 of metadata dataframe. Note: metadata_variable must be consistent across distance matrix, ava, q-value and effect-size calculations.
`anchor_annotation`	name or index of column containing common names of the annotated metabolite. Default: Column 2 of annotation dataframe.

Value

mac.ava abundance versus anchor values of metabolic features

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
mets.ava <- Macarron::calAVA(se = mbx,
                             mod.assn = modules.assn)
```

---

calES *Calculate effect size of differential abundance of metabolic features.*

Description

Effect size of a metabolic feature is the difference in mean log2 transformed abundances in test and control (reference) samples. For the specified metadata variable, effect size is calculated for all test categories against the reference category.

Usage

```
calES(se, mac.qval)
```

Arguments

se	SummarizedExperiment object created using Macarron::prepInput().
mac.qval	the output of Macarron::calQval().

Value

mac.es effect sizes of metabolic features in phenotypes of interest.

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
mets.qval <- Macarron::calQval(se = mbx,
                               mod.assn = modules.assn)
mets.es <- Macarron::calES(se = mbx,
                           mac.qval = mets.qval)
```

`calQval` *Calculate q-value of differential abundance of metabolic features.*

Description

This function uses the MaAsLin2 package for estimating q-value of differential abundance. Multiple fixed and random effects can be specified for fitting the multiple regression model. Default analysis method is "LM". Can be run on multiple cores. metadata_variable and ref (reference group) should be the same as the one specified for effect size calculation.

Usage

```
calQval(
  se,
  mod.assn,
  metadata_variable = 1,
  fixed_effects = NULL,
  random_effects = NULL,
  reference = NULL,
  output_folder = NULL,
  cores = 1,
  plot_heatmap = FALSE,
  plot_scatter = FALSE,
  heatmap_first_n = 50
)
```

Arguments

se	SummarizedExperiment object created using Macarron::prepInput().
mod.assn	the output of Macarron::findMacMod().
metadata_variable	name or index of metadata column identifying phenotypes/conditions to be used for differential abundance testing. Default: Column 1 of metadata dataframe Note: metadata_variable must be consistent across ava, q-value and effect-size calculations.
fixed_effects	fixed effects, comma delimited e.g. c("metadata1","metadata2"). Default: all columns in metadata.
random_effects	random effects, comma delimited. Default: NULL.
reference	a reference level/group in each metadata column with more than 3 levels, semi-colon delimited for multiple variables e.g. c("metadata1,ref1";"metadata2,ref2"). Default: alphabetically first phenotype/condition will be used as reference. Note: Reference must be specified for metadata with more than 2 levels.
output_folder	the name of the output folder where all MaAsLin2 results will be written. Default: maaslin2_output
cores	the number of R processes to run in parallel.
plot_heatmap	Maaslin2 option-Generate a heatmap for the significant associations. Default: TRUE
plot_scatter	Maaslin2 option-Generate scatter plots for the significant associations. Default: FALSE
heatmap_first_n	Maaslin2 option-Generate heatmap for top n significant associations. Default: 50

Value

mac.qval q-value of metabolic features in phenotypes of interest.

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
```

```
mets.qval <- Macarron::calQval(se = mbx,
                               mod.assn = modules.assn)
```

`chem_taxonomy` *Chemical taxonomy lookup table*

Description

A dataset mapping metabolite IDs (HMDB or PubChem) to chemical subclass and class, used by `decorateID()`.

Usage

```
data(chem_taxonomy)
```

Format

A data frame with 145837 rows and 4 variables:

hmdb_id HMDB accession

pubchem_compound_id PubChem CID (character, may be NA if only HMDB is known)

sub_class Chemical subclass (character, may be NA)

class Chemical class (character, may be NA)

`decorateID` *Create a chemical taxonomy table for annotated metabolic features. Uses a pregenerated chemical taxonomy table shipped with the package to assign chemical subclass and class to annotated features.*

Description

Create a chemical taxonomy table for annotated metabolic features. Uses a pregenerated chemical taxonomy table shipped with the package to assign chemical subclass and class to annotated features.

Usage

```
decorateID(input_annotations, chemical_taxonomy = NULL)
```

Arguments

input_annotations
: a dataframe (features x annotations) containing the available feature annotations. ^^Column 1 must contain standard annotations such as HMDB ID or PubChem CID for the subset of identified/annotated metabolic features.

chemical_taxonomy
: Optional data frame with columns hmdb_id, pubchem_compound_id, sub_class, and class. If NULL (default), the built-in chem_taxonomy dataset is loaded via data().

Value

A data frame with three columns: ID (same type as the first column of input_annotations), Sub_Class and Class, for the subset of features whose IDs are present in chemical_taxonomy.

Examples

```
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
input_taxonomy <- decorateID(annotations_df)
```

---

findMacMod — *Cluster metabolic features based on covarying abundances into modules*

Description

Cluster metabolic features based on covarying abundances into modules

Usage

```
findMacMod(
  se,
  w,
  input_taxonomy,
  standard_identifier = 1,
  min_module_size = NULL,
  evaluateMOS = TRUE
)
```

Arguments

se	SummarizedExperiment object created using Macarron::prepInput().
w	distance matrix from function Macarron::makeDisMat().

`input_taxonomy` chemical taxonomy file with 3 columns specifying annotation, subclass and class of annotated features. Can be created using the decorateID.R utility of Macarron. Annotation specified with "standard_identifier" and annotation in the first column of the chemical taxonomy file must match.

`standard_identifier` name or index of column containing HMDB or PubChem IDs. Default: Column 1 in annotation dataframe.

`min_module_size` minimum module size to be used for module identification with dynamicTreeCut::cutreeDynamic(). Default is cube root of number of prevalent features.

`evaluateMOS` examine measure of success for modules identified using min_module_size, min_module_size + 5, min_module_size + 10, min_module_size - 5, min_module_size - 10

Value

mod.assn metabolic features clustered into "modules" based on covarying abundances and measures of success.

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
```

`Macarron` *Macarron*

Description

Macarron

Usage

```
Macarron(
  input_abundances,
  input_annotations,
  input_metadata,
  input_taxonomy,
  output = "Macarron_output",
  metadata_variable = 1,
  min_prevalence = 0.7,
  execution_mode = "serial",
  standard_identifier = 1,
  anchor_annotation = 2,
  min_module_size = NULL,
  fixed_effects = NULL,
  random_effects = NULL,
  reference = NULL,
  cores = 1,
  plot_heatmap = FALSE,
  plot_scatter = TRUE,
  heatmap_first_n = 50,
  show_best = TRUE,
  priority_threshold = 0.9,
  per_module = 10,
  per_phenotype = 1000,
  only_characterizable = TRUE
)
```

Arguments

input_abundances	a comma-delimited file or dataframe (features x samples) containing metabolic feature intensities (abundances).
input_annotations	a comma-delimited file or dataframe (features x annotations) containing available feature annotations.
input_metadata	a comma-delimited file or dataframe (samples x metadata) containing sample metadata.
input_taxonomy	a comma-delimited file or dataframe containing the chemical class and subclass information of annotated features.
output	name of the folder where Macarron output files will be written. Default: "Macarron_output".
metadata_variable	Name or index of the column that identifies the phenotypes/conditions in the study. Default: Column 1 of metadata dataframe.
min_prevalence	prevalence threshold (percentage). Default = 0.7.
execution_mode	BiocParallel execution mode. Options: "serial" or "multi" Default = "serial".

`standard_identifier`
: Name or index of column containing HMDB or PubChem IDs. Default: Column 1 in annotation dataframe.

`anchor_annotation`
: Name or index of column containing common names of the annotated metabolite. Default: Column 2 of annotation dataframe.

`min_module_size`
: Integer that defines the size of the smallest covariance module. Default: Cube root of number of prevalent metabolic features.

`fixed_effects`
: Covariates for linear modeling with MaAsLin2. Default: All columns of metadata dataframe.

`random_effects`
: Random effects for linear modeling with MaAsLin2. Default: NULL.

`reference`
: Reference category (factor) in categorical metadata covariates containing three or more levels. Must be provided as a string of 'covariate,reference' semi-colon delimited for multiple covariates.

`cores`
: MaAsLin2 option-The number of R processes to be run in parallel.

`plot_heatmap`
: MaAslin2 option-Generate a heatmap for the significant associations. Default: TRUE

`plot_scatter`
: MaAslin2 option-Generate scatter plots for the significant associations. Default: FALSE

`heatmap_first_n`
: MaAslin2 option-Generate heatmap for top n significant associations. Default = 50

`show_best`
: write 1000 or fewer highly prioritized metabolic features into a separate file. Default: TRUE

`priority_threshold`
: cut-off of priority score for showing highly prioritized features. Default = 0.9

`per_module`
: show first n highly prioritized features in a module. Default = 10

`per_phenotype`
: show highly prioritized n features per phenotype/condition. Default = 1000

`only_characterizable`
: show highly prioritized features in modules which contain at least one annotated metabolite. Default = TRUE

Value

mac.result dataframes containing metabolic features listed according to their priority (potential bioactivity) in a phenotype of interest.

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
mets.prioritized <- Macarron::Macarron(input_abundances = prism_abundances,
                                       input_annotations = prism_annotations,
```

```
                                    input_metadata = prism_metadata,
                                    input_taxonomy = met_taxonomy)
```

`makeDisMat` *Create a biweight midcorrelation (WGCNA::bicor()) based distance matrix.*

Description

Create a biweight midcorrelation (WGCNA::bicor()) based distance matrix.

Usage

```
makeDisMat(
  se,
  metadata_variable = 1,
  min_prevalence = 0.7,
  nthreads = 0,
  execution_mode = "serial",
  optimize.for = c("runtime", "memory")
)
```

Arguments

`se`	SummarizedExperiment object created using Macarron::prepInput().
`metadata_variable`	metadata column identifying phenotypes/conditions to be used to evaluate prevalence of features. Default = Column 1 of metadata dataframe.
`min_prevalence`	prevalence threshold (percentage). Default = 0.7.
`nthreads`	number of processors
`execution_mode`	"serial" or "multi" processing with BiocParallel. Default: "serial" (recommended for laptops). "multi" may be used when running Macarron on a cluster.
`optimize.for`	runtime or memory.
	Features present (i.e. not NA) in "min_prevalence" of samples in each category of a "metadata_variable" will be considered e.g. if min_prevalence is 0.7 and metadata_variable has 2 categories A and B, union of (i) features present in at least 70 and (ii) features present in at least 70 Correlation between feature abundances are is calculated using WGCNA::bicor().

Value

w distance matrix where distance = 1-bicor^3

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
```

`prepInput` *Create a SummarizedExperiment object*

Description

Create a SummarizedExperiment object

Usage

```
prepInput(input_abundances, input_annotations, input_metadata)
```

Arguments

`input_abundances`	a dataframe (features x samples) containing metabolic feature intensities (abundances).
`input_annotations`	a dataframe (features x annotations) containing the available feature annotations. ^^Column 1 must contain standard annotations such as HMDB ID or Pubchem CID for the subset of identified/annotated features. ^^Column 2 must contain metabolite name. ^^Column 3 must contain a continuous numeric chemical property such as m/z or shift/ppm.
`input_metadata`	a dataframe (samples x metadata) containing sample metadata. ^^Row names must identify samples. ^^Column 1 must identify phenotypes or conditions (categorical metadata) associated with the samples. Must not contain NA. Rows with no specified phenotype/condition will be removed.

Value

SummarizedExperiment object

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
```

`prioritize` *Rank metabolic features and prioritize based on predicted bioactivity.*

Description

Metabolic features are ranked based on AVA, and q-value and effect size of differential abundance. The harmonic mean of these three ranks is calculated and used as the meta-rank to prioritize potentially bioactive features in a phenotype (or condition). Top-ranked features have good relative abundance, and are significantly perturbed in the specified environment/phenotype.

Usage

```
prioritize(se, mod.assn, mac.ava, mac.qval, mac.es)
```

Arguments

`se`	SummarizedExperiment object created using Macarron::prepInput()
`mod.assn`	the output of Macarron::findMacMod()
`mac.ava`	the output of Macarron::calAVA()
`mac.qval`	the output of Macarron::calQval()
`mac.es`	the output of Macarron::calES()

Value

mac.result - metabolic features listed according to priority

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
```

```
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
mets.ava <- Macarron::calAVA(se = mbx,
                             mod.assn = modules.assn)
mets.qval <- Macarron::calQval(se = mbx,
                               mod.assn = modules.assn)
mets.es <- Macarron::calES(se = mbx,
                           mac.qval = mets.qval)
mets.prioritized <- Macarron::prioritize(se = mbx,
                                         mod.assn = modules.assn,
                                         mac.ava = mets.ava,
                                         mac.qval = mets.qval,
                                         mac.es = mets.es)
```

`showBest` *View highly prioritized bioactives grouped by modules.*

Description

Modules are listed in the order of priority. Only the top-ranked n features in each module are shown. The priority of a module is the ratio of number of features in it that are ranked higher than the cut-off and the size of the module. This utility function makes it easier to understand default prioritization results of large datasets where a few hundred metabolic features are highly-prioritized.

Usage

```
showBest(
  mac.result,
  priority_threshold = 0.9,
  per_module = 10,
  per_phenotype = 1000,
  only_characterizable = TRUE
)
```

Arguments

`mac.result`	the output of Macarron::Macarron() or Macarron::prioritize().
`priority_threshold`	cut-off of priority score. Default = 0.9.
`per_module`	show first n highly prioritized features in a module. Default = 10

per_phenotype show highly prioritized n features per phenotype/condition. Default = 1000

only_characterizable
show highly prioritized features in modules which contain at least one annotated metabolite. Default = TRUE

Value

best.mets -highly-prioritized bioactives in each module in each phenotype

Examples

```
prism_abundances = system.file("extdata", "demo_abundances.csv", package="Macarron")
abundances_df = read.csv(file = prism_abundances, row.names = 1)
prism_annotations = system.file("extdata", "demo_annotations.csv", package="Macarron")
annotations_df = read.csv(file = prism_annotations, row.names = 1)
prism_metadata = system.file("extdata", "demo_metadata.csv", package="Macarron")
metadata_df = read.csv(file = prism_metadata, row.names = 1)
met_taxonomy = system.file("extdata", "demo_taxonomy.csv", package="Macarron")
taxonomy_df = read.csv(file = met_taxonomy)
mbx <- Macarron::prepInput(input_abundances = abundances_df,
                           input_annotations = annotations_df,
                           input_metadata = metadata_df)
w <- Macarron::makeDisMat(se = mbx)
modules.assn <- Macarron::findMacMod(se = mbx,
                                     w = w,
                                     input_taxonomy = taxonomy_df)
mets.ava <- Macarron::calAVA(se = mbx,
                             mod.assn = modules.assn)
mets.qval <- Macarron::calQval(se = mbx,
                               mod.assn = modules.assn)
mets.es <- Macarron::calES(se = mbx,
                           mac.qval = mets.qval)
mets.prioritized <- Macarron::prioritize(se = mbx,
                                         mod.assn = modules.assn,
                                         mac.ava = mets.ava,
                                         mac.qval = mets.qval,
                                         mac.es = mets.es)
best.mets <- Macarron::showBest(mac.result = mets.prioritized)
```

Index

∗ **datasets**
chem_taxonomy, 6

calAVA, 2
calES, 3
calQval, 4
chem_taxonomy, 6

decorateID, 6

findMacMod, 7

Macarron, 8
makeDisMat, 11

prepInput, 12
prioritize, 13

showBest, 14