Package: TaxSEA (via r-universe)

October 26, 2026

Type Package

Title Taxon Set Enrichment Analysis

Version 1.5.8

Description TaxSEA makes metagenomic data easier to interpret. Rather than reading species one at a time, it tests whether a-priori defined sets of taxa shift together, drawing on public databases (BugSigDB, MiMeDB, GutMGene, mBodyMap, BacDive and GMRepoV2) and on sets collated from the literature. Two approaches are provided. TaxSEA takes taxon names or NCBI IDs with a rank per taxon, such as a fold change or correlation coefficient, and tests each set with a Kolmogorov-Smirnov test. ssTaxSEA scores every set in a single sample as the mean centered log-ratio of its members, giving a per-sample value to use in any downstream model. Both work with any taxonomic profiling technology, including 16S rRNA gene sequencing, shotgun metagenomics and metatranscriptomics.

License GPL-3

Encoding UTF-8

LazyData false

VignetteBuilder knitr

biocViews Microbiome, Metagenomics, Sequencing, GeneSetEnrichment, RNASeq

URL https://github.com/feargalr/taxsea, https://feargalr.github.io/TaxSEA/

BugReports https://github.com/feargalr/taxsea/issues

Depends R (>= 4.5.0)

Suggests BiocStyle, S4Vectors, bugsigdbr, fgsea, knitr, mia, rmarkdown, SummarizedExperiment, testthat

Imports methods, stats, utils

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

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

Date/Publication 2026-10-26 06:37:55 UTC

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

RemoteRef HEAD

RemoteSha 9831356494cf3096773098dea7216393db07dffb

Contents

get_ncbi_taxon_ids 2
get_taxon_sets 3
NCBI_ids 3
ssTaxSEA 4
taxon_rank_sets 7
TaxSEA 9
TaxSEA_db 11
TaxSEA_test_data 12

Index **13**

`get_ncbi_taxon_ids` *Retrieve NCBI Taxonomy IDs for a list of taxon names*

Description

This function takes a vector of taxon names and returns a vector of NCBI taxonomy IDs by querying the NCBI Entrez API.

Usage

```
get_ncbi_taxon_ids(taxon_names)
```

Arguments

`taxon_names`	A character vector of taxon names

Value

A character vector of NCBI taxonomy IDs corresponding to the input taxon names

Examples

```
taxon_names <- c("Escherichia coli", "Staphylococcus aureus")
taxon_ids <- get_ncbi_taxon_ids(taxon_names)
```

get_taxon_sets *Retrieve Taxon Sets from TaxSEA Library*

Description

Retrieve from the TaxSEA database which taxon sets (metabolite producers and disease signatures) contain a taxon of interest.

Usage

```
get_taxon_sets(taxon_to_fetch)
```

Arguments

`taxon_to_fetch` The taxon to search for in the TaxSEA database.

Value

A character vector containing the names of taxonomic sets where the specified taxon is present.

Examples

```
# Retrieve sets for Bifidobacterium longum
get_taxon_sets(taxon_to_fetch = "Bifidobacterium_longum")
```

NCBI_ids *NCBI IDs Dataset*

Description

A lookup from taxon names to NCBI taxonomy IDs, used to translate the names supplied to `TaxSEA()` into the IDs that `TaxSEA_db` is built on. Names are present both with spaces and with underscores, and each ID is also present as its own name so that IDs can be supplied directly. Species names, former names and IDs of the `BacDive_` set members were added from the BacDive harvest without changing any existing entry.

Usage

```
data(NCBI_ids)
```

Format

A named list where:

names Taxon names (e.g. "Bifidobacterium breve", "Bifidobacterium_breve") or NCBI IDs

values NCBI taxonomy IDs, as character

Source

NCBI Taxonomy; BacDive (https://bacdive.dsmz.de).

Examples

```
data(NCBI_ids)
# Can look up either with or without spaces
NCBI_ids["Bifidobacterium_breve"]
NCBI_ids["Bifidobacterium breve"]
```

---

ssTaxSEA *Single-Sample Taxon Set Enrichment Analysis*

Description

Computes a per-sample score for each taxon set as the mean centered log-ratio (CLR) of the set's members within that sample.

Usage

```
ssTaxSEA(
  counts,
  lookup_missing = FALSE,
  min_set_size = 5,
  max_set_size = 300,
  custom_db = NULL,
  pseudocount = NULL,
  scale = c("counts", "proportion"),
  reference = c("all", "top_k"),
  top_k = 20
)
```

Arguments

counts	A numeric matrix, data.frame, or SummarizedExperiment/TreeSummarizedExperiment object. For matrix/data.frame input: rows are taxa, columns are samples, and row names must be taxon names (e.g. species names or NCBI IDs). For SummarizedExperiment input: the first assay is used and rownames() provide taxon identifiers.
lookup_missing	Logical indicating whether to fetch missing NCBI IDs via the NCBI API. Default is FALSE.
min_set_size	Minimum size of taxon sets to include. Default is 5.
max_set_size	Maximum size of taxon sets to include. Default is 300.
custom_db	A user-provided list of taxon sets. If NULL (default), the built-in TaxSEA database is used (excluding BugSigDB).

pseudocount	Numeric value added to every value before the log transform, to handle zeros. It is on the scale set by scale: if NULL (default), 0.5 for scale = "counts" and 1e-6 for scale = "proportion". This choice affects the scores, so report it alongside your results.
scale	Either "counts" (default) or "proportion". With "counts" the input is used as supplied and must be counts or another count-like measure. With "proportion" each sample is first divided by its total, so counts, proportions and percentages (for example MetaPhlAn output) are all accepted, and pseudocount is a proportion. See Details.
reference	Either "all" (default) or "top_k". What each sample's log values are centered on: the mean over all taxa (the standard CLR), or the mean over that sample's top_k most abundant detected taxa. See Details.
top_k	Number of most abundant detected taxa used as the reference when reference = "top_k". Default is 20. A sample with fewer detected taxa uses all of its detected taxa.

Details

For each sample the counts are centered log-ratio transformed, and the score for a taxon set is the mean of the CLR values of the set's members:

$$score_{S,j} = \frac{1}{|S|} \sum_{i \in S} clr_{ij}$$

The CLR is computed across *all* taxa supplied in counts, before any subsetting to set members. This matters: restricting the matrix to set members first would make the geometric mean that the CLR divides by depend on the sets being tested, which manufactures apparent signal in sets that have none.

Because the score is a within-sample quantity, ssTaxSEA does not need a cohort and is well defined for a single sample.

Value

A numeric matrix of scores with taxon sets as rows and samples as columns. Higher values indicate that the set's members are more abundant in that sample, relative to the average taxon in that same sample (or, with reference = "top_k", relative to that sample's most abundant taxa).

Choosing a reference

The standard CLR (reference = "all") centers each sample on the mean log value of every taxon in the table, zeros included. That reference moves with richness: when many rare taxa drop out, as after antibiotics or chemotherapy, the reference falls and every surviving taxon's score rises, even if its abundance relative to the other survivors has not changed. Many taxon sets share the same surviving members, so hundreds of scores can rise together. Adjusting for richness in a downstream model does not remove this. The reference also depends on which rows are in the table, so filtering taxa differently changes every score.

reference = "top_k" centers each sample on the mean log value of its top_k most abundant detected taxa. Rare taxa are still scored and still count towards their sets; they just no longer move

everyone else's score. Losing rare taxa, or adding or removing rare and all-zero rows, leaves the reference unchanged. Zeros never enter the reference: a sample with fewer than `top_k` detected taxa uses all of them. The score then reads as abundance relative to the sample's dominant taxa.

A pseudocount set from each sample's own data would move with the rare tail in the same way, which is why `pseudocount` is a fixed value. On the count scale a pseudocount of 0.5 reads is a different proportion in samples of different depth; `scale = "proportion"` with a small fixed pseudocount avoids that. The default of 1e-6 is near the detection limit of deep shotgun profiles; raise it if your smallest detected proportions are larger. For relative abundance input, such as MetaPhlAn percentages, use `scale = "proportion"`.

Interpreting the scores

Scores are comparable **across samples within a taxon set**: a higher score in sample A than sample B means the set's members make up more of sample A's community.

Scores are **not** comparable across taxon sets. A set of abundant taxa will score higher than a set of rare taxa in every sample, regardless of biology, simply because its members are more abundant. When visualising the matrix, center or scale the rows first (for example `t(scale(t(scores)))`), and when comparing groups, compare a single set's scores between groups rather than comparing different sets to each other.

No p-values are returned. The score is a descriptive statistic; to test a hypothesis, compare scores across samples using a test appropriate to your design (for example `wilcox.test` or a linear model on one row of the returned matrix).

See Also

`TaxSEA` for group-level enrichment.

Examples

```
# Toy count matrix: 30 taxa x 8 samples
set.seed(42)
counts <- matrix(rpois(240, lambda = 10), nrow = 30, ncol = 8)
rownames(counts) <- paste0("Taxon_", seq_len(30))
colnames(counts) <- paste0("Sample_", seq_len(8))

# Raise one set's members in the last four samples
counts[1:6, 5:8] <- counts[1:6, 5:8] * 5

sets <- list(
  elevated_set = paste0("Taxon_", 1:6),
  control_set  = paste0("Taxon_", 20:27)
)

scores <- ssTaxSEA(counts, custom_db = sets, min_set_size = 3)
dim(scores)        # sets x samples
round(scores, 2)

# The planted signal shows up as higher scores in samples 5-8
rowMeans(scores[, 1:4]) - rowMeans(scores[, 5:8])
```

```
# A reference that is not moved by changes in rare taxa, for
# relative abundance input
pct <- sweep(counts, 2, colSums(counts), "/") * 100
ssTaxSEA(pct, custom_db = sets, min_set_size = 3,
         scale = "proportion", reference = "top_k", top_k = 10)
```

`taxon_rank_sets` *Taxon Rank Set Enrichment Analysis*

Description

Groups species by taxonomic ranks and performs TaxSEA enrichment analysis at each rank level. Returns a named list of data frames, one per taxonomic rank (excluding species).

Usage

```
taxon_rank_sets(taxon_ranks, lineage_df, min_set_size = 5, max_set_size = 100)
```

Arguments

`taxon_ranks`	A named numeric vector of log2 fold changes. Names should be feature identifiers matching the `species` column in `lineage_df`, or matching `rownames()` of a SummarizedExperiment/TreeSummarizedExperiment.
`lineage_df`	Either a data frame or a `SummarizedExperiment`/`TreeSummarizedExperiment` object. **Data frame input:** Must include a `species` column and one or more taxonomic rank columns (e.g., kingdom, phylum, class, order, family, genus). The `species` column is used for matching and is excluded from the enrichment analysis. **SummarizedExperiment input:** Taxonomy is extracted from `rowData()` and feature identifiers from `rownames()`. If the `mia` package is installed, `taxonomyRanks()` is used to identify taxonomic rank columns; otherwise all `rowData()` columns are used. The Species rank column is automatically excluded from the analysis. Requires the `SummarizedExperiment` package.
`min_set_size`	Minimum number of species in a set to include in the analysis. Default is 5.
`max_set_size`	Maximum number of species in a set to include in the analysis. Default is 100.

Value

A named list of data frames, one per taxonomic rank. Each data frame contains columns: taxonSetName, median_rank_of_set_members, PValue, Test_statistic, and FDR. Ranks that produce no valid sets (e.g., due to size filtering) are included as empty data frames with a message.

Examples

```
# --- Example 1: Data frame input ---
# Create a lineage data frame (e.g., parsed from curatedMetagenomicData)
# The 'species' column must match the names in taxon_ranks.

lineage_df <- data.frame(
  species = c("Cutibacterium_acnes", "Klebsiella_pneumoniae",
              "Propionibacterium_humerusii", "Moraxella_osloensis",
              "Enhydrobacter_aerosaccus", "Staphylococcus_capitis",
              "Staphylococcus_epidermidis", "Staphylococcus_aureus",
              "Escherichia_coli", "Enterobacter_cloacae",
              "Pseudomonas_aeruginosa", "Acinetobacter_baumannii",
              "Lactobacillus_rhamnosus", "Lactobacillus_acidophilus",
              "Bifidobacterium_longum", "Bifidobacterium_breve"),
  kingdom = rep("Bacteria", 16),
  phylum = c("Actinobacteria", "Proteobacteria",
             "Actinobacteria", "Proteobacteria",
             "Proteobacteria", "Firmicutes",
             "Firmicutes", "Firmicutes",
             "Proteobacteria", "Proteobacteria",
             "Proteobacteria", "Proteobacteria",
             "Firmicutes", "Firmicutes",
             "Actinobacteria", "Actinobacteria"),
  class = c("Actinobacteria", "Gammaproteobacteria",
            "Actinobacteria", "Gammaproteobacteria",
            "Alphaproteobacteria", "Bacilli",
            "Bacilli", "Bacilli",
            "Gammaproteobacteria", "Gammaproteobacteria",
            "Gammaproteobacteria", "Gammaproteobacteria",
            "Bacilli", "Bacilli",
            "Actinobacteria", "Actinobacteria"),
  order = c("Propionibacteriales", "Enterobacterales",
            "Propionibacteriales", "Pseudomonadales",
            "Rhodospirillales", "Bacillales",
            "Bacillales", "Bacillales",
            "Enterobacterales", "Enterobacterales",
            "Pseudomonadales", "Pseudomonadales",
            "Lactobacillales", "Lactobacillales",
            "Bifidobacteriales", "Bifidobacteriales"),
  family = c("Propionibacteriaceae", "Enterobacteriaceae",
             "Propionibacteriaceae", "Moraxellaceae",
             "Rhodospirillaceae", "Staphylococcaceae",
             "Staphylococcaceae", "Staphylococcaceae",
             "Enterobacteriaceae", "Enterobacteriaceae",
             "Pseudomonadaceae", "Moraxellaceae",
             "Lactobacillaceae", "Lactobacillaceae",
             "Bifidobacteriaceae", "Bifidobacteriaceae"),
  genus = c("Cutibacterium", "Klebsiella",
            "Cutibacterium", "Moraxella",
            "Enhydrobacter", "Staphylococcus",
            "Staphylococcus", "Staphylococcus",
            "Escherichia", "Enterobacter",
```

```
            "Pseudomonas", "Acinetobacter",
            "Lactobacillus", "Lactobacillus",
            "Bifidobacterium", "Bifidobacterium"),
  stringsAsFactors = FALSE
)

set.seed(42)
fc <- setNames(rnorm(16), lineage_df$species)
results <- taxon_rank_sets(fc, lineage_df, min_set_size = 2)
names(results)
results$family

# --- Example 2: SummarizedExperiment / TreeSummarizedExperiment input ---

library(mia)
data(GlobalPatterns, package = "mia")
tse <- GlobalPatterns

# Run differential abundance (e.g., ALDEx2) to get fold changes
# aldex_out <- ... (your DA analysis)
# fc <- aldex_out$effect
# names(fc) <- rownames(aldex_out)

# Run taxon rank set enrichment directly from the TSE
results <- taxon_rank_sets(fc, tse, min_set_size = 5)
names(results)  # Kingdom, Phylum, Class, Order, Family, Genus
results$Family
```

TaxSEA *TaxSEA: Taxon Set Enrichment Analysis*

Description

Modular TaxSEA implementation supporting enrichment (KS) and ORA (Fisher). Provide either `taxon_ranks` for enrichment or `input_taxa` for ORA.

Usage

```
TaxSEA(
  taxon_ranks = NULL,
  input_taxa = NULL,
  mode = NULL,
  lookup_missing = FALSE,
  min_set_size = 5,
  max_set_size = 300,
  custom_db = NULL,
  bugsigdb = FALSE
)
```

Arguments

taxon_ranks	Named numeric vector of statistics (e.g., log2 fold changes). Required for enrichment.
input_taxa	Character vector of taxa to treat as "hits"/selected taxa. Required for ORA.
mode	Character. One of "enrichment" or "ora". If NULL, inferred from which input is provided.
lookup_missing	Logical indicating whether to fetch missing NCBI IDs. Default is FALSE.
min_set_size	Minimum size of taxon sets to include in the analysis. Default is 5.
max_set_size	Maximum size of taxon sets to include in the analysis. Default is 300.
custom_db	A user-provided list of taxon sets. If NULL (default), the built-in database is used.
bugsigdb	Logical; whether to augment the built-in database with BugSigDB signatures, which are downloaded (and cached) at run time via bugsigdbr. Default is FALSE. Each set is tested against all the other taxa covered by the sets being analysed, and BugSigDB adds many extra taxa, so including it changes the p-values of every set, not only the BugSigDB ones. BugSigDB is also updated independently of TaxSEA. Set to TRUE to include it, and record the BugSigDB version used. If the download fails, a warning is given and the analysis continues without BugSigDB. Ignored when custom_db is supplied.

Value

A list of data frames with taxon set results.

Examples

```
data("TaxSEA_test_data")
res <- TaxSEA(taxon_ranks = TaxSEA_test_data)
head(res$All_databases)


# Include BugSigDB signatures (downloaded at run time)
res_bsdb <- TaxSEA(taxon_ranks = TaxSEA_test_data, bugsigdb = TRUE)
head(res_bsdb$BugSigDB)


# ORA example (toy): treat taxa with positive values as "hits"
hits <- names(TaxSEA_test_data)
res_ora <- TaxSEA(input_taxa = hits, mode = "ora")
head(res_ora$All_databases)
```

TaxSEA_db *TaxSEA Database A dataset containing taxon sets. Each item in the list is a taxon set, and each member within a taxon set is a taxon.*

Description

TaxSEA Database A dataset containing taxon sets. Each item in the list is a taxon set, and each member within a taxon set is a taxon.

Usage

```
data(TaxSEA_db)
```

Format

A list of vectors. Each vector contains character strings representing taxa as NCBI taxonomy IDs.

Details

Set names start with their source: `MiMeDB_`, `GutMGene_`, `GMRepoV2_`, `mBodyMap_`, `BacDive_` and so on.

The `BacDive_` sets describe measured phenotypes of cultured strains (oxygen tolerance, Gram stain, substrate use, enzyme activities, growth temperature, salt and pH ranges and others), aggregated to species over every strain BacDive holds. They are built by the bacdive_harvester pipeline (https://github.com/feargalr/bacdive_harvester); see `inst/scripts/make_BacDive_sets.R`. Only sets with at least 3 members are included. Names read `BacDive_<Family>_<value>`, optionally followed by an assay context (e.g. `_fermentation`, `_reduction`) and `_negative` for species tested and found negative, e.g. `BacDive_Oxygen_facultative_anaerobe`, `BacDive_Uses_nitrate_reduction`.

Per-set provenance (source family, number of species and strains behind each set) and a record of the build (harvester release, BacDive client version, harvest dates) are shipped with the package:

```
system.file("extdata", "BacDive_set_provenance.tsv", package = "TaxSEA")
system.file("extdata", "BacDive_build_info.tsv", package = "TaxSEA")
```

Source

See the README. BacDive sets: Schober et al. (2025) BacDive in 2025: the core database for prokaryotic strain data, via https://bacdive.dsmz.de.

Examples

```
data(TaxSEA_db)
all_sets <- names(TaxSEA_db)
GABA_producers<-TaxSEA_db[["MiMeDB_producers_of_GABA"]]
```

TaxSEA_test_data *TaxSEA Test Data*

Description

A dataset containing taxon ranks and taxon IDs.

Usage

```
data(TaxSEA_test_data)
```

Format

A data frame with two columns:

rank Character vector representing taxon ranks

id Character vector representing taxon IDs

A data frame with columns 'rank' and 'id' representing taxon ranks and taxon IDs, respectively.

Source

See READ ME.

Examples

```
data(TaxSEA_test_data)
test_results <- TaxSEA(TaxSEA_test_data)
```

Index

∗ **datasets**
NCBI_ids, 3
TaxSEA_db, 11
TaxSEA_test_data, 12

get_ncbi_taxon_ids, 2
get_taxon_sets, 3

NCBI_ids, 3

ssTaxSEA, 4

taxon_rank_sets, 7
TaxSEA, *6*, 9
TaxSEA_db, 11
TaxSEA_test_data, 12

wilcox.test, *6*