Package 'chem16S'

January 16, 2025

Date 2025-01-16

Version 1.2.0

Title Chemical Metrics for Microbial Communities

Author Jeffrey Dick [aut, cre] (<https://orcid.org/0000-0002-0687-5890>)

Maintainer Jeffrey Dick <j3ffdick@gmail.com>

Depends R (>= 3.5.0)

Imports plyr, ggplot2, rlang, reshape2, phyloseq, canprot (>= 2.0.0)

Suggests tinytest, knitr, patchwork, ggpmisc, rmarkdown

Description Combines taxonomic classifications of high-throughput 16S rRNA gene sequences with reference proteomes of archaeal and bacterial taxa to generate amino acid compositions of community reference proteomes. Calculates chemical metrics including carbon oxidation state ('Zc'), stoichiometric oxidation and hydration state ('nO2' and 'nH2O'), H/C, N/C, O/C, and S/C ratios, grand average of hydropathicity ('GRAVY'), isoelectric point ('pI'), protein length, and average molecular weight of amino acid residues. Uses precomputed reference proteomes for archaea and bacteria derived from the Genome Taxonomy Database ('GTDB'). Also includes reference proteomes derived from the NCBI Reference Sequence ('RefSeq') database and manual mapping from the 'RDP Classifier' training set to 'RefSeq' taxonomy as described by Dick and Tan (2023) <doi:10.1007/s00248-022-01988-9>. Processes taxonomic classifications in 'RDP Classifier' format or OTU tables in 'phyloseq-class' objects from the Bioconductor package 'phyloseq'.

Encoding UTF-8

License GPL-3

VignetteBuilder knitr

URL https://github.com/jedick/chem16S

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-16 05:50:02 UTC

Contents

`chem16S-package` *Chemical metrics for microbial communities*

Description

Functions and data to calculate chemical metrics for reference proteomes for microbial (archaeal and bacterial) communities. Amino acid compositions of community reference proteomes are generated by combining reference proteomes of taxa (derived from GTDB or RefSeq) with taxonomic classifications of 16S rRNA gene sequences.

Details

- `read_RDP` - Read and filter an RDP Classifier table
- `map_taxa` - Map RDP Classifier assignments to the NCBI taxonomy
- `get_metrics` - Get chemical metrics for community reference proteomes
- `get_metadata` - Example function for retrieving sample metadata
- `plot_metrics` - Plot metrics with symbols and colors based on metadata
- `physeq` - Functions designed to analyze `phyloseq-class` objects

History:

1. Work begun in 2021. The combination of RefSeq reference proteomes with taxonomic abundances to compute community-level chemical metrics was described by Dick and Tan (2023). **chem16S** originated as code in the **JMDplots** package (`https://github.com/jedick/JMDplots`).
2. Development in 2022. Dick and Meng (2023) compared community Z_C with redox potential measurements from local to global scales. The term “community reference proteomes” was first applied, and **chem16S** was split into a separate package.
3. Late 2022. GTDB r207 was added as a reference database.
4. June–July 2023. Integration with **phyloseq** and addition of vignettes: *Chemical metrics of reference proteomes*, *Integration of chem16S with phyloseq*, and *Plotting two chemical metrics*. Default reference database changed to GTDB r207.
5. April 2024. Updated to GTDB r214.
6. July 2024. Updated to GTDB r220.

Options set in package chem16S

chem16S sets an option using the global options mechanism in R. This option will be set when package **chem16S** (or its namespace) is loaded if not already set.

manual_mappings A data frame of mappings between RDP and NCBI (RefSeq) taxonomies, which is read from 'extdata/manual_mappings.csv'. The columns include RDP.rank, RDP.name, NCBI.rank, NCBI.name, and notes. This option is made available so the user can modify the manual mappings used by map_taxa at runtime.

Files in RefDB/RefSeq_206

NOTE: None of the '*.R' files in the 'extdata' directories are included in the package submitted to CRAN; see GitHub or Zenodo for these files.

This directory contains two sets of files: 1) scripts to process source RefSeq sequence files to generate amino acid compositions of species-level reference proteomes and taxonomic names; 2) script and output for amino acid compositions of higher-level taxa. The files are based on RefSeq release 206 of 2021-05-21 (O'Leary et al., 2016).

'README.txt' Description of steps to generate reference proteomes of species-level taxa (including downloads and shell commands).

'gencat.sh' Helper script to extract microbial protein records from the RefSeq catalog.

'genome_AA.R' R code to sum the amino acid compositions of all proteins for each bacterial, archaeal, and viral species in the NCBI Reference Sequence database. NOTE: To save space in this package, the output file ('genome_AA.csv') is stored in the RefDB/RefSeq_206 directory of the JMDplots package on GitHub (https://github.com/jedick/JMDplots. The first five columns are: protein ("refseq"), organism (taxonomic id), ref (organism name), abbrv (empty), chains (number of protein sequences for this organism). Columns 6 to 25 have the counts of amino acids.

'taxonomy.R' R code for processing taxonomic IDs; the output file is 'taxonomy.csv'. The columns are NCBI taxonomic ID (taxid), and names at different taxonomic rank (species, genus, family, order, class, phylum, superkingdom).

'taxon_AA.R' Functions to create the files listed below:

'taxon_AA.csv.xz' Average amino acid composition of reference proteomes for all species in each genus, family, order, class, phylum, and superkingdom.

Files in RefDB/GTDB_220

'taxon_AA.R' Functions to process GTDB source files (Parks et al., 2022) and produce the following output file:

'taxon_AA.csv.xz' Average amino acid composition of reference proteomes for all species in each genus, family, order, class, phylum, and domain. In both this file and the corresponding file for RefSeq (see above), the protein, organism, ref, and abbrv columns contain the rank, taxon name, number of species used to generate the amino acid composition of this taxon, and parent taxon. chains is 1, denoting a single polypeptide chain, so the amino acid composition represents the average per-protein amino acid composition in this taxon, and the sum of amino acid counts is the average protein length.

Files in extdata/metadata

'BGPF13.csv' Metadata for Heart Lake Geyser Basin, Yellowstone (Bowen De León et al., 2012).

'HLA+16.csv' Metadata for the Baltic Sea (Herlemann et al., 2016).

'SMS+12.csv' Metadata for Bison Pool, Yellowstone (Swingley et al., 2012).

Files in extdata/RDP

Output of RDP Classifier with the default training set.

'pipeline.R' Pipeline for sequence data processing (uses external programs fastq-dump, vsearch, seqtk, RDP Classifier). This was used to make the files in both 'RDP' and 'RDP-GTDB_220' (the latter with GTDB = TRUE in the script).

'BGPF13.tab.xz' Heart Lake Geyser Basin.

'HLA+16.tab.xz' Baltic Sea.

'SMS+12.tab.xz' Bison Pool.

Files in extdata/RDP-GTDB_220

Output of RDP Classifer trained with 16S rRNA sequences from GTDB release 220 (doi:10.5281/zenodo.7633099).

'BGPF13.tab.xz' Heart Lake Geyser Basin.

'HLA+16.tab.xz' Baltic Sea.

'SMS+12.tab.xz' Bison Pool.

Files in extdata/DADA2-GTDB_220

Identification and taxonomic classification of sequences using DADA2 with GTDB r220.

'FEN+22' Analysis of data from Fonseca et al. (2022) for marine sediment from the Humboldt Sulfuretum. 'pipeline.R' has the commands used to process the 16S rRNA gene sequence data and was adapted by Jeffrey Dick from the DADA2 pipeline tutorial (Callahan, 2020). 'SraRunInfo.csv' was obtained from the NCBI Sequence Read Archive (SRA) (https://www.ncbi.nlm.nih.gov/sra/?term=PRJNA251688). 'sample_data.csv' has data obtained from NCBI BioSample records for BioProject PRJNA251688. '*.png' are several plots created while running the DADA2 pipeline. 'ps_FEN+22.rds' contains the phyloseq object with (including otu_table, sample_data, and refseq objects) created at the end of the DADA2 pipeline.

'ZFZ+23' Analysis of data from Zhang et al. (2023) for hot springs in the Qinghai-Tibet Plateau. 'pipeline.R' has the commands used to process the 16S rRNA gene sequence data and was adapted by Jeffrey Dick from the DADA2 pipeline tutorial (Callahan, 2020). 'SraRunInfo.csv' was obtained from the NCBI Sequence Read Archive (SRA) (https://www.ncbi.nlm.nih.gov/sra/?term=PRJNA860942). 'sample_data.csv' has data obtained from NCBI BioSample records for BioProject PRJNA860942. '*.png' are several plots created while running the DADA2 pipeline. 'ps_ZFZ+23.rds' contains the phyloseq object with (including otu_table, sample_data, and refseq objects) created at the end of the DADA2 pipeline.

'mouse/pipeline.R' Pipeline adapted from the DADA2 pipeline tutorial (Callahan, 2020) for processing the mouse example dataset (see chem16S-data).

References

Bowen De León K, Gerlach R, Peyton BM, Fields MW. 2013. Archaeal and bacterial communities in three alkaline hot springs in Heart Lake Geyser Basin, Yellowstone National Park. *Frontiers in Microbiology* **4**: 330. doi:10.3389/fmicb.2013.00330

Callahan B. 2020. DADA2 Pipeline Tutorial (1.16). `https://benjjneb.github.io/dada2/tutorial.html`, accessed on 2023-06-14.

Dick JM, Meng D. 2023. Community- and genome-based evidence for a shaping influence of redox potential on bacterial protein evolution. *mSystems* **8**(3): e00014-23. doi:10.1128/msystems.00014-23

Dick JM, Tan J. 2023. Chemical links between redox conditions and estimated community proteomes from 16S rRNA and reference protein sequences. *Microbial Ecology* **85**: 1338–1355. doi:10.1007/s00248022019889

Fonseca A, Espinoza C, Nielsen LP, Marshall IPG, Gallardo VA. 2022. Bacterial community of sediments under the Eastern Boundary Current System shows high microdiversity and a latitudinal spatial pattern. *Frontiers in Microbiology* **13**: 1016418. doi:10.3389/fmicb.2022.1016418

Herlemann DPR, Lundin D, Andersson AF, Labrenz M, Jürgens K. 2016. Phylogenetic signals of salinity and season in bacterial community composition across the salinity gradient of the Baltic Sea. *Frontiers in Microbiology* **7**: 1883. doi:10.3389/fmicb.2016.01883

O'Leary NA et al. 2016. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research* **44**: D733-D745. doi:10.1093/nar/gkv1189

Parks DH, Chuvochina M, Rinke C, Mussig AJ, Chaumeil P-A, Hugenholtz P. 2022. GTDB: an ongoing census of bacterial and archaeal diversity through a phylogenetically consistent, rank normalized and complete genome-based taxonomy. *Nucleic Acids Research* **50**: D785–D794. doi:10.1093/nar/gkab776

Swingley WD, Meyer-Dombard DR, Shock EL, Alsop EB, Falenski HD, Havig JR, Raymond J. 2012. Coordinating environmental genomics and geochemistry reveals metabolic transitions in a hot spring ecosystem. *PLOS One* **7**(6): e38108. doi:10.1371/journal.pone.0038108

Zhang H-S, Feng Q-D, Zhang D-Y, Zhu G-L, Yang L. 2023. Bacterial community structure in geothermal springs on the northern edge of Qinghai-Tibet plateau. *Frontiers in Microbiology* **13**: 994179. doi:10.3389/fmicb.2022.994179

`chem16S-data` *‘phyloseq-class‘ objects generated using the DADA2 Pipeline Tutorial*

Description

Objects in `phyloseq-class` format created by following the DADA2 Pipeline Tutorial (Callahan, 2020).

Details

The example dataset for gut communities in a single mouse was taken from the mothur MiSeq SOP. It is an extract of the complete dataset reported by Schloss et al. (2012). The output files were generated by using three different training sets to assign taxonomy to genus level:

'mouse.silva' silva_nr99_v138.1: Silva 138.1 prokaryotic SSU taxonomic training data formatted for DADA2

'mouse.RDP' rdp_train_set_18: RDP taxonomic training data formatted for DADA2 (RDP trainset 18/release 11.5)

'mouse.GTDB_220' GTDB_bac120_arc53_ssu_r220: DADA2 formatted 16S rRNA gene sequences for both bacteria & archaea (GTDB release 220)

Author(s)

Jeffrey M. Dick, following the DADA2 Pipeline Tutorial by Callahan (2020).

References

Callahan B (2020). DADA2 Pipeline Tutorial (1.16). `https://benjjneb.github.io/dada2/tutorial.html`, accessed on 2023-06-14. Publication year taken from `https://code.bioconductor.org/browse/dada2/blob/RELEASE_3_11/DESCRIPTION`.

Schloss PD, Schubert AM, Zackular JP, Iverson KD, Young VB, Petrosino JF. 2012. Stabilization of the murine gut microbiome following weaning. *Gut Microbes* **3**(4): 383–393. doi:10.4161/gmic.21008

Examples

```
# Load one data set
data(mouse.silva)
# Summarize the object (phyloseq-class)
mouse.silva
```

---

chemlab *Formatted labels for chemical metrics*

Description

Formatted labels for indicating chemical metrics on plots.

Usage

```
chemlab(varname)
```

Arguments

varname	character, the name of a variable

Details

This function provides formatted labels for chemical metrics and related variables. `varname` can be any one of the chemical metrics described in `calc_metrics`.

Value

For most values of `varname`, a language object suitable for plotting (see `plotmath`). For labels where no special mathematical notation is required, a character object. If `varname` is not among the chemical metrics described in `calc_metrics`, the value of `varname` itself.

See Also

`cplab`

Examples

```
chemlab("Zc")
```

`get_metadata` *Template function for metadata*

Description

Simple function that processes metadata files.

Usage

```
get_metadata(file, metrics)
```

Arguments

`file`	character, metadata file (CSV format)
`metrics`	data frame, output of `get_metrics`

Details

This function is provided as a simple example of reading metadata about 16S rRNA datasets. The basic metadata for all datasets include ‘`name`’ (study name), ‘`Run`’ (SRA or other run ID), ‘`BioSample`’ (SRA BioSample), and ‘`Sample`’ (sample description). Each data set contains environmental data, but the specific variables differ between datasets. Therefore, this function processes the environmental data for different datasets and appends ‘`pch`’ and ‘`col`’ columns that can be used for visualization and identifying trends between groups of samples.

Samples are identified by the `Run` column (case-sensitive) in both `file` and `metrics`. It is possible that metrics for one or more samples are not available (for example, if the number of RDP classifications is below `mincount` in `get_metrics`). The function therefore removes metadata for unavailable samples. Then, the samples in `metrics` are placed in the same order as they appear in metadata. Ordering the samples is important because the columns in RDP Classifier output are not necessarily in the same order as the metadata.

Value

If `metrics` is NULL, a data frame read from `file` with columns ‘`pch`’ and ‘`col`’ appended. Otherwise, a list with two components, `metadata` and `metrics`. In this list, `metrics` is the value from the `metrics` argument and `metadata` is a data frame read from `file` with columns ‘`pch`’ and ‘`col`’ appended and excluding any rows that correspond to unavailable samples in `metrics`.

Examples

```
# Get metrics for the Bison Pool dataset (Swingley et al., 2012)
RDPfile <- system.file("extdata/RDP/SMS+12.tab.xz", package = "chem16S")
RDP <- read_RDP(RDPfile)
map <- map_taxa(RDP, refdb = "RefSeq_206")
metrics <- get_metrics(RDP, map, refdb = "RefSeq_206")
# Read the metadata file
mdatfile <- system.file("extdata/metadata/SMS+12.csv", package = "chem16S")
mdat <- get_metadata(mdatfile, metrics)

# Print sample names
mdat$metadata$Sample # 1 2 3 4 5
# Plot Zc of community reference proteomes at sites 1-5
Site <- mdat$metadata$Sample
Zc <- mdat$metrics$Zc
plot(Site, Zc, ylab = "Carbon oxidation state", type = "b")

# Find the columns of the RDP file for samples 1 to 5
match(mdat$metadata$Run, colnames(RDP)) # 6 5 7 8 9
# NOTE: for this dataset, samples 1 and 2 are in columns 6 and 5, respectively
```

`get_metrics` *Calculate chemical metrics of community reference proteomes*

Description

Combines taxonomic classifications with reference proteomes for taxa to get amino acid compositions of community reference proteomes. Amino acid compositions are used to calculate chemical metrics.

Usage

```
get_metrics(RDP, map, refdb = "GTDB_220", taxon_AA = NULL, groups = NULL,
  return_AA = FALSE, zero_AA = NULL, metrics = c("Zc", "nO2", "nH2O"))
```

Arguments

`RDP` data frame, taxonomic abundances produced by `read_RDP` or `ps_taxacounts`

`map` data frame, taxonomic mapping produced by `map_taxa`

`refdb` character, name of reference database (‘`GTDB_220`’ or ‘`RefSeq_206`’)

`taxon_AA` data frame, amino acid compositions of taxa, used to bypass `refdb` specification `groups` list of indexing vectors, samples to be aggregated into groups

`return_AA` logical, return the amino acid composition for each sample instead of the chemical metrics?

`zero_AA` character, three-letter abbreviation(s) of amino acid(s) to assign zero counts for calculating chemical metrics

`metrics` character, the chemical metrics to calculate

Details

`get_metrics` calculates selected chemical metrics for the community reference proteome in each sample. The community reference proteome is computed from the amino acid compositions of reference proteomes for taxa (obtained from the reference database in `refdb`), multiplied by taxonomic abundances given in `RDP`. `RDP` may include results from the RDP Classifier (read using `read_RDP`) or derived from the OTU table of a `phyloseq-class` object (see `ps_taxacounts`). `map` defines the taxonomic mapping between `RDP` and `refdb`. Then, chemical metrics are calculated from the amino acid composition of the community reference proteome. The default chemical metrics are carbon oxidation state (Z_C), stoichiometric oxidation state ($n\mathrm{O}_2$), and stoichiometric hydration state ($n\mathrm{H}_2\mathrm{O}$). See `calc_metrics` for other available metrics.

`groups`, if given, is a list of one or more indexing vectors (with logical or numeric values) corresponding to samples whose taxonomic classifications are aggregated into groups before calculating amino acid compositions and chemical metrics.

Value

A data frame with one row for each sample, corresponding to columns 5 and above of `RDP`. The sample names are in the first column, which is named `Run` by default, or `group` if the `groups` argument is provided. The remaining columns have numeric values and are named for each of the calculated `metrics`.

See Also

`calc_metrics`, `plot_metrics`

Examples

```
## First two examples are for RDP Classifier with default training set
## and mapping to NCBI taxonomy with RefSeq reference proteomes

# Get chemical metrics for all samples in a dataset
RDPfile <- system.file("extdata/RDP/BGPF13.tab.xz", package = "chem16S")
RDP <- read_RDP(RDPfile)
map <- map_taxa(RDP, refdb = "RefSeq_206")
# This is a data frame with 14 rows and Run, Zc, nO2, and nH2O columns
(metrics <- get_metrics(RDP, map, refdb = "RefSeq_206"))

# Read the metadata file
mdatfile <- system.file("extdata/metadata/BGPF13.csv", package = "chem16S")
# Create list with metadata and metrics in same sample order
mdat <- get_metadata(mdatfile, metrics)
```

```
# Calculate metrics for aggregated samples of Archaea and Bacteria
groups <- list(A = mdat$metadata$domain == "Archaea",
  B = mdat$metadata$domain == "Bacteria")
# This is a data frame with 2 rows and group, Zc, nO2, and nH2O columns
get_metrics(RDP, map, refdb = "RefSeq_206", groups = groups)

# Classifications were made using the RDP Classifer retrained with GTDB r220
RDPfile.GTDB <- system.file("extdata/RDP-GTDB_220/BGPF13.tab.xz", package = "chem16S")
RDP.GTDB <- read_RDP(RDPfile.GTDB)
# These use the default option of refdb = "GTDB_220"
map.GTDB <- map_taxa(RDP.GTDB)
metrics.GTDB <- get_metrics(RDP.GTDB, map.GTDB)

# Plot Zc from GTDB vs RefSeq
xylim <- range(metrics$Zc, metrics.GTDB$Zc)
plot(metrics$Zc, metrics.GTDB$Zc, xlim = xylim, ylim = xylim, type = "n")
lines(xylim, xylim, lty = 2, col = 8)
points(metrics$Zc, metrics.GTDB$Zc, pch = mdat$metadata$pch, bg = mdat$metadata$col)
md.leg <- mdat$metadata[1:2, ]
legend("bottomright", md.leg$domain, pch = md.leg$pch, pt.bg = md.leg$col)
title(quote(italic(Z)[C]~"from GTDB vs RefSeq"))

# To exclude tryptophan, tyrosine, and phenylalanine
# from the calculation of chemical metrics
get_metrics(RDP.GTDB, map.GTDB, zero_AA = c("Trp", "Tyr", "Phe"))
```

get_metric_byrank *Chemical metrics for taxa aggregated to a given rank*

Description

Calculates a single chemical metric for taxa in each sample aggregated to a specified rank.

Usage

```
get_metric_byrank(RDP, map, refdb = "GTDB_220", taxon_AA = NULL,
  groups = NULL, zero_AA = NULL, metric = "Zc", rank = "genus")
```

Arguments

RDP	data frame, taxonomic abundances produced by read_RDP or ps_taxacounts
map	data frame, taxonomic mapping produced by map_taxa
refdb	character, name of reference database (‘GTDB_220’ or ‘RefSeq_206’)
taxon_AA	data frame, amino acid compositions of taxa, used to bypass refdb specification
groups	list of indexing vectors, samples to be aggregated into groups
zero_AA	character, three-letter abbreviation(s) of amino acid(s) to assign zero counts for calculating chemical metrics

`metric`	character, chemical metric to calculate
`rank`	character, amino acid compositions of all lower-ranking taxa (to genus) are aggregated to this rank

Details

This function adds up amino acid compositions of taxa up to the specified rank and returns a data frame samples on the rows and taxa on the columns. Because amino acid composition for genera have been precomputed from species-level genomes in a reference database, chemical metrics for genera are constant. In contrast, chemical metrics for higher-level taxa is variable as they depend on the reference genomes as well as relative abundances of children taxa.

The value for `rank` should be one of '`rootrank`', '`domain`', '`phylum`', '`class`', '`order`', '`family`', or '`genus`'. For all ranks other than '`genus`', the amino acid compositions of all lower-ranking taxa are weighted by taxonomic abundance and summed in order to calculate the chemical metric at the specified rank. If the rank is '`genus`', then no aggregation is done (because it is lowest-level rank available in the classifications), and the values of the `metric` for all genera in each sample are returned. If the rank is '`rootrank`', then the results are equivalent to community reference proteomes (i.e., `get_metrics`).

The `RDP`, `map`, '`refdb`', and `groups` arguments are the same as described in `get_metrics`. See `calc_metrics` for available metrics.

Value

A data frame of numeric values with row names corresponding to samples and column names corresponding to taxa.

References

Dick JM, Shock E. 2013. A metastable equilibrium model for the relative abundances of microbial phyla in a hot spring. *PLOS One* **8**: e72395. doi:10.1371/journal.pone.0072395

See Also

`get_metrics`

Examples

```
# Plot similar to Fig. 1 in Dick and Shock (2013)
# Read example dataset
RDPfile <- system.file("extdata/RDP-GTDB_220/SMS+12.tab.xz", package = "chem16S")
RDP <- read_RDP(RDPfile)
# Get mapping to reference database
map <- map_taxa(RDP)
# Calculate phylum-level Zc
phylum_Zc <- get_metric_byrank(RDP, map, rank = "phylum")
# Keep phyla present in at least two samples
n_values <- colSums(!sapply(phylum_Zc, is.na))
phylum_Zc <- phylum_Zc[n_values > 2]
# Swap first two samples to get them in the right location
# (MG-RAST accession numbers for these samples are not in spatial order)
```

```
phylum_Zc <- phylum_Zc[c(2, 1, 3, 4, 5), ]
matplot(phylum_Zc, type = "b", xlab = "Sampling site (hot -> cool)", ylab = "Zc")
title("Phylum-level Zc at Bison Pool hot spring")
```

map_taxa *Map taxonomic names to NCBI or GTDB taxonomy*

Description

Maps taxonomic names to NCBI (RefSeq) or GTDB taxonomy by automatic matching of taxonomic names, with manual mappings for some groups.

Usage

```
map_taxa(taxacounts = NULL, refdb = "GTDB_220",
  taxon_AA = NULL, quiet = FALSE)
```

Arguments

taxacounts	data frame with taxonomic name and abundances
refdb	character, name of reference database ('GTDB_220' or 'RefSeq_206')
taxon_AA	data frame, amino acid compositions of taxa, used to bypass refdb specification
quiet	logical, suppress printed messages?

Details

This function maps taxonomic names to the NCBI (RefSeq) or GTDB taxonomy. taxacounts should be a data frame generated by either read_RDP or ps_taxacounts. Input names are made by combining the taxonomic rank and name with an underscore separator (e.g. 'genus_ Escherichia/Shigella'). Input names are then matched to the taxa listed in 'taxon_AA.csv.xz' found under 'RefDB/RefSeq_206' or 'RefDB/GTDB_220'. The protein and organism columns in these files hold the rank and taxon name extracted from the RefSeq or GTDB database. Only exactly matching names are automatically mapped.

For mapping to the NCBI (RefSeq) taxonomy, some group names are manually mapped as follows (see Dick and Tan, 2023):

RDP training set	**NCBI**
genus_Escherichia/Shigella	genus_Escherichia
phylum_Cyanobacteria/Chloroplast	phylum_Cyanobacteria
genus_Marinimicrobia_genera_incertae_sedis	species_Candidatus Marinimicrobia bacterium
class_Cyanobacteria	phylum_Cyanobacteria
genus_Spartobacteria_genera_incertae_sedis	species_Spartobacteria bacterium LR76
class_Planctomycetacia	class_Planctomycetia
class_Actinobacteria	phylum_Actinobacteria
order_Rhizobiales	order_Hyphomicrobiales
genus_Gp1	genus_Acidobacterium

genus_Gp6	genus_Luteitalea
genus_GpI	genus_Nostoc
genus_GpIIa	genus_Synechococcus
genus_GpVI	genus_Pseudanabaena
family_Family II	family_Synechococcaceae
genus_Subdivision3_genera_incertae_sedis	family_Verrucomicrobia subdivision 3
order_Clostridiales	order_Eubacteriales
family_Ruminococcaceae	family_Oscillospiraceae

To avoid manual mapping, GTDB can be used for both taxonomic assignemnts and reference proteomes. 16S rRNA sequences from GTDB release 220 are available for the RDP Classifier (doi:10.5281/zenodo.7633099) and **dada2** (doi:10.5281/zenodo.13984843). Example files created using the RDP Classifier are provided under ‘`extdata/RDP-GTDB_220`’. An example dataset created with DADA2 is `data(mouse.GTDB_220)`; this is a `phyloseq-class` object that can be processed with functions described at `physeq`.

Change `quiet` to TRUE to suppress printing of messages about manual mappings, most abundant unmapped groups, and overall percentage of mapped names.

Value

Integer vector with length equal to number of rows of `taxacounts`. Values are rownumbers in the data frame generated by reading `taxon_AA.csv.xz`, or NA for no matching taxon. Attributes `unmapped_groups` and `unmapped_percent` have the input names of unmapped groups and their percentage of the total classification count.

References

Dick JM, Tan J. 2023. Chemical links between redox conditions and estimated community proteomes from 16S rRNA and reference protein sequences. *Microbial Ecology* **85**: 1338–1355. doi:10.1007/s00248022019889

Examples

```
# Partial mapping from RDP training set to NCBI taxonomy
file <- system.file("extdata/RDP/SMS+12.tab.xz", package = "chem16S")
# Use drop.groups = TRUE to exclude root- and domain-level
# classifications and certain non-prokaryotic groups
RDP <- read_RDP(file, drop.groups = TRUE)
map <- map_taxa(RDP, refdb = "RefSeq_206")
# About 24% of classifications are unmapped
sum(attributes(map)$unmapped_percent)

# 100% mapping from GTDB training set to GTDB taxonomy
file <- system.file("extdata/RDP-GTDB_220/SMS+12.tab.xz", package = "chem16S")
RDP.GTDB <- read_RDP(file)
map.GTDB <- map_taxa(RDP.GTDB)
stopifnot(all.equal(sum(attributes(map.GTDB)$unmapped_percent), 0))
```

physeq *Process ‘phyloseq-class‘ objects to calculate chemical metrics*

Description

Calculate chemical metrics of community reference proteomes from taxonomic abundances in phyloseq-class objects and make plots of individual metrics or two metrics against each other.

Usage

```
ps_taxacounts(physeq, split = TRUE)
ps_metrics(physeq, split = TRUE, refdb = "GTDB_220", quiet = FALSE, ...)
plot_ps_metrics(physeq, metrics = c("Zc", "nO2", "nH2O"), x = "samples",
  color = NULL, shape = NULL, title = NULL,
  scales = "free_y", nrow = 1, sortby = NULL, ...)
plot_ps_metrics2(physeq, metrics = c("Zc", "nH2O"), color = NULL,
  shape = NULL, title = NULL, refdb = "GTDB_220", quiet = FALSE)
```

Arguments

physeq	‘phyloseq-class‘, a **phyloseq** object containing an ‘otu_table‘ and ‘tax_table‘
split	logical, calculate metrics for each sample (TRUE) or the entire set (FALSE)
refdb	character, name of reference database (passed to get_metrics)
quiet	logical, suppress printed messages? (passed to map_taxa)
...	additional arguments passed to ps_metrics and thence to get_metrics
metrics	character, chemical metrics to calculate
x	map this variable to the horizontal axis
color	map this variable to colors
shape	map this variable to shapes
title	main title for the plot
scales	‘free_y’ or ‘fixed’ for different vertical scales or same scale in each panel
nrow	number of rows of panels (passed to facet_wrap)
sortby	subset of metrics argument, sort x-index by the mean values of these metrics

Details

ps_taxacounts returns a data frame with lowest-level classifications (from genus to phylum) and abundances for each OTU in the physeq object.

ps_metrics calculates chemical metrics of community reference proteomes from the lowest-level classifications after taxonomic mapping with map_taxa.

plot_ps_metrics plots individual chemical metrics or multiple chemical metrics, each in their own panel. This function is modified from plot_richness; see that help page for further details about graphics-related arguments.

plot_ps_metrics2 plot two chemical metrics against each other.

The metrics argument is passed to get_metrics and selects the metrics to calculate. split, refdb, and quiet can also be used as additional arguments in plot_ps_metrics and plot_ps_metrics2.

Value

For ps_taxacounts, a data frame with samples in rows, and columns taxid (sample name taken from the OTU table obtained from otu_table(physeq)), lineage (all taxonomic names from the taxonomy table obtained from tax_table(physeq), separated by semicolons), name (lowest-level taxonomic name), rank (rank of lowest-level taxon), followed by the abundances from the OTU table.

For ps_metrics, a data frame with samples in rows, and columns corresponding to each of the metrics.

For plot_ps_metrics and plot_ps_metrics2, ggplot objects.

Author(s)

The author of this Rd file and the ps_taxacounts and ps_metrics functions is Jeffrey Dick. plot_ps_metrics is based on the plot_richness function by Paul J. McMurdie and was modified from that function by Jeffrey Dick.

Examples

```
# Example 1: Humboldt Sulfuretum (Fonseca et al., 2022)
# This file has the classifications made with DADA2 using 16S rRNA sequences from GTDB r220
file <- system.file("extdata/DADA2-GTDB_220/FEN+22/ps_FEN+22.rds", package = "chem16S")
physeq <- readRDS(file)
# Get lowest-level (to genus) classification for each OTU
taxacounts <- ps_taxacounts(physeq)
# Show numbers of assignments at each taxonomic level
table(taxacounts$rank)
# Map taxonomic names to GTDB r220
map <- map_taxa(taxacounts, refdb = "GTDB_220")
# Taxonomy and reference proteomes are from different versions of GTDB,
# so there is a small percentage of unmapped classifications
sum(attr(map, "unmapped_percent"))
# Calculate chemical metrics
ps_metrics(physeq, refdb = "GTDB_220")

# Example 2: GlobalPatterns dataset from phyloseq
data(GlobalPatterns, package = "phyloseq")
# Plot metrics grouped by sample type and sorted by mean Zc;
# refdb = "RefSeq" uses manual mappings from the RDP to NCBI taxonomy
p <- plot_ps_metrics(GlobalPatterns, x = "SampleType",
                     sortby = "Zc", refdb = "RefSeq_206")
# Change orientation of x-axis labels
p + ggplot2::theme(axis.text.x = ggplot2::element_text(
  angle = 45, vjust = 1, hjust = 1))
```

`plot_metrics` *Plot chemical metrics of community reference proteomes*

Description

Functions to plot chemical metrics of community reference proteomes.

Usage

```
plot_metrics(mdat,
  xvar = "Zc", yvar = "nH2O",
  xlim = NULL, ylim = NULL,
  xlab = chemlab(xvar), ylab = chemlab(yvar),
  plot.it = TRUE, add = FALSE, identify = FALSE,
  points = TRUE, lines = FALSE, title = TRUE,
  cex = 1, pt.open.col = 1, pch1 = 1, pch2 = 21,
  return = "data", extracolumn = NULL
)
```

Arguments

`mdat`	list, output by `get_metadata`
`xvar`	character, name of x variable
`yvar`	character, name of y variable
`xlim`	numeric, x axis limits
`ylim`	numeric, y axis limits
`xlab`	x axis label
`ylab`	y axis label
`plot.it`	logical, make a plot?
`add`	logical, add to existing plot?
`identify`	logical, run `identify` for interactive identification of points?
`points`	logical, plot points?
`lines`	logical, plot lines?
`title`	character, plot title
`cex`	numeric, point size
`pt.open.col`	color of border for open point symbols (pch > `20`)
`pch1`	numeric, symbol for samples in group 1
`pch2`	numeric, symbol for samples in group 2
`return`	character, indicates whether to return ‘`data`’ values or group ‘`means`’
`extracolumn`	character, the name of one or more extra columns (from `get_metadata`) to include in the output

Details

`plot_metrics` plots the values of Z_C and $n\mathrm{H_2O}$ (or other variables indicated by `xvar` and `yvar`) provided in `mdat$metrics`. Symbol shape and color (`pch` and `col`) are taken from `mdat$metadata`.

If `pch1` and `pch2` are provided, then samples are classified into two groups according to the value of `mdat$metadata$pch`. Mean values of the chemical metrics for each group are plotted with star-shaped symbols.

Value

For `plot_metrics`, a data frame with columns for study name and Run IDs (‘`name`’, ‘`Run`’), chemical metrics (taken from `mdat$metrics`), and symbols and colors for plotting points (‘`pch`’, ‘`col`’).

References

Herlemann, D. P. R., Lundin, D., Andersson, A. F., Labrenz, M. and Jürgens, K. (2016) Phylogenetic signals of salinity and season in bacterial community composition across the salinity gradient of the Baltic Sea. *Front. Microbiol.* **7**, 1883. doi:10.3389/fmicb.2016.01883

Examples

```
# Make a plot for the Baltic Sea salinity gradient
# (data from Herlemann et al., 2016)
RDPfile <- system.file("extdata/RDP/HLA+16.tab.xz", package = "chem16S")
RDP <- read_RDP(RDPfile)
map <- map_taxa(RDP, refdb = "RefSeq_206")
metrics <- get_metrics(RDP, map, refdb = "RefSeq_206")
mdatfile <- system.file("extdata/metadata/HLA+16.csv", package = "chem16S")
mdat <- get_metadata(mdatfile, metrics)
pm <- plot_metrics(mdat)
# Add a legend
legend <- c("< 6 PSU", "6-20 PSU", "> 20 PSU")
pch <- c(24, 20, 21)
pt.bg <- c(3, NA, 4)
legend("bottomright", legend, pch = pch, col = 1, pt.bg = pt.bg, bg = "white")
# Classify samples with low and high salinity
ilo <- mdat$metadata$salinity < 6
ihi <- mdat$metadata$salinity > 20
# Add convex hulls
canprot::add_hull(pm$Zc[ilo], pm$nH2O[ilo],
  col = adjustcolor("green3", alpha.f = 0.3), border = NA)
canprot::add_hull(pm$Zc[ihi], pm$nH2O[ihi],
  col = adjustcolor("blue", alpha.f = 0.3), border = NA)

# Show points for all samples and larger star-shaped points
# for mean values of high- and low-salinity samples
plot_metrics(mdat, pch1 = 21, pch2 = 24)

# Plot nO2 instead of Zc
plot_metrics(mdat, xvar = "nO2")

# Make a plot for only Proteobacteria
```

```
RDP <- read_RDP(RDPfile, lineage = "Proteobacteria")
map <- map_taxa(RDP, refdb = "RefSeq_206")
metrics <- get_metrics(RDP, map, refdb = "RefSeq_206")
mdatfile <- system.file("extdata/metadata/HLA+16.csv", package = "chem16S")
mdat <- get_metadata(mdatfile, metrics)
mdat$metadata$name <- paste(mdat$metadata$name, "(Proteobacteria)")
plot_metrics(mdat)
```

read_RDP *Read and filter RDP Classifier output*

Description

Reads RDP Classifier output and optionally extracts lineages, removes classifications below a certain count, or truncates classifications to lowest taxonomic level.

Usage

```
read_RDP(file, lineage = NULL, mincount = 200,
  lowest.level = NULL, drop.groups = FALSE, quiet = FALSE)
```

Arguments

`file`	character, file name
`lineage`	character, regular expression for filtering lineages
`mincount`	integer, samples with less than this number of RDP classifications are excluded
`lowest.level`	character, truncate classifications at this taxonomic level
`drop.groups`	logical, remove some taxonomic groups deemed to be uninformative?
`quiet`	logical, suppress printed messages?

Details

`read_RDP` reads and filters RDP results for all samples in a study. The input file should be created by RDP Classifer using the -h option to create a hierarchical listing. Classifications for multiple samples can be combined into a single file using the merge-count command of RDP Classifier.

Only rows (lineages) with count > 0 for at least one sample are retained.

If `drop.groups` is TRUE: Sequences with classifications at only the root or domain level (Root, Archaea, or Bacteria) are omitted because they provide poor taxonomic resolution. Sequences classified to the class Chloroplast or genera Chlorophyta or Bacillariophyta are also omitted because they have little correspondence with the NCBI taxonomy. These actions were hard-coded in earlier versions of chem16S (based on using the RDP Classifier taxonomy) but were made optional with adoption of the GTDB.

Then, only columns (samples) with classification count >= `mincount` are retained. All remaining sequences (those classified to genus or higher levels) can be used for mapping to the NCBI taxonomy.

The lineage text of the RDP Classifier looks like “Root;rootrank;Archaea;domain;Diapherotrites; phylum;Diapherotrites Incertae Sedis AR10;genus;”, so you can use `lineage = "Archaea"` to select the archaeal classifications or `lineage = "genus"` to select genus-level classifications.

Use the `lowest.level` argument to truncate the classifications to a level higher than genus. This argument does not reduce the number of classifications, but only trims the RDP lineages to the specified level. This may create duplicate lineages, for which the classification counts are summed, and only unique lineages are present in the returned data frame.

Change `quiet` to TRUE to suppress printing of messages about percentage classification to genus level, omitted sequences, and final range of total counts among all samples.

Value

Data frame with columns inherited from `file` (from RDP output files: `taxid`, `rank`, `lineage`, `name`, and one column of classification counts for each sample). If any sample has less than `mincount` total counts, that column is omitted from the result. The rows in the result are subset from those in `file` by filtering operations.

NB: taxids in RDP files are not NCBI taxids.

Examples

```
# An example file
file <- system.file("extdata/RDP/SMS+12.tab.xz", package = "chem16S")

# Default settings
r0 <- read_RDP(file)

# Suppress messages
r1 <- read_RDP(file, quiet = TRUE)

# Increase minimum count threshold
r2 <- read_RDP(file, mincount = 500)
# This should be 3 (i.e., 3 samples have less than 500 counts)
ncol(r1) - ncol(r2)

# Keep only Archaea
r3 <- read_RDP(file, lineage = "Archaea")
# This should be TRUE
all(grepl("Archaea", r3$lineage))

# Truncate taxonomy at phylum
r4 <- read_RDP(file, lowest.level = "phylum")
# This should be TRUE
all(sapply(strsplit(r4$lineage, ";"), tail, 1) == "phylum")
```

Index