Package ‘rbiom’

May 29, 2020

Type Package

Title Read/Write, Transform, and Summarize 'BIOM' Data

Version 1.0.2

Description A toolkit for working with Biological Observation Matrix ('BIOM') files. Features include reading/writing all 'BIOM' formats, rarefaction, alpha diversity, beta diversity (including 'UniFrac'), summarizing counts by taxonomic level, and sample subsetting. Standalone functions for reading, writing, and subsetting phylogenetic trees are also provided. All CPU intensive operations are encoded in C with multi-thread support.

URL https://cmmr.github.io/rbiom/index.html

BugReports https://github.com/cmmr/rbiom/issues

License AGPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppParallel

Imports magrittr, methods, plyr, Rcpp, RcppParallel, R.utils, rjson, slam, openxlsx

Suggests ape, reshape2, rhdf5, testthat

RoxygenNote 7.1.0

NeedsCompilation yes

Author Daniel P. Smith [aut, cre, cph] (<https://orcid.org/0000-0002-2479-2044>)

Maintainer Daniel P. Smith <dansmith@orst.edu>

Repository CRAN

Date/Publication 2020-05-29 12:30:09 UTC

R topics documented:

`alpha.div` *Estimate the diversity of each sample.*

Description

Estimate the diversity of each sample.

Usage

```
alpha.div(biom, rarefy = FALSE)
```

Arguments

biom
: A `matrix`, `simple_triplet_matrix`, or `BIOM` object, as returned from read.biom. For matrices, the rows and columns are assumed to be the taxa and samples, respectively.

rarefy
: Control how/whether rarefactions are done prior to alpha diversity computations. Options are:

- `FALSE` Use each sample's current set of observations without applying any rarefaction. (Default)
- `TRUE` Automatically select and apply a single rarefaction.
- `"multi"` Automatically select and apply multiple rarefactions.
- ***integer vector*** Rarefy at the specified depth(s).

Value

A data frame of four diversity values for each sample in `biom`. The column names are **Sample**, **Depth** and the diversity metrics: **OTUs**, **Shannon**, **Chao1**, **Simpson**, and **InvSimpson**. The row names are the sample names, except when multiple rarefactions are done.

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

ad <- alpha.div(biom)
head(ad)

biom <- subset(biom, `Body Site` == "Saliva" & Age < 26)
ad   <- alpha.div(biom, "multi")
boxplot(Shannon ~ Depth, data=ad, xlab="Reads", ylab="Diversity")
```

`beta.div` *Make a distance matrix of samples vs samples.*

Description

Make a distance matrix of samples vs samples.

Usage

```
beta.div(biom, method, weighted = TRUE, tree = NULL)
```

Arguments

biom — A matrix, simple_triplet_matrix, or BIOM object, as returned from read.biom. For matrices, the rows and columns are assumed to be the taxa and samples, respectively.

method — The distance algorithm to use. Options are: **“manhattan”**, **“euclidean”**, **“bray-curtis”**, **“jaccard”**, and **“unifrac”**. Non-ambiguous abbrevations of the method names are also accepted. A phylogentic tree must be present in biom or explicitly provided via tree= to use the UniFrac methods.

weighted — Take relative abundances into account. When weighted=FALSE, only presence/absence is considered.

tree — A phylo object representing the phylogenetic relationships of the taxa in biom. Will be taken from the tree embedded in the biom object if not explicitly specified. Only required for computing UniFrac distance matrices.

Value

A distance matrix.

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)
biom <- select(biom, 1:10)

dm <- beta.div(biom, 'unifrac')

as.matrix(dm)[1:4,1:4]
plot(hclust(dm))
```

counts *Get the abundance counts.*

Description

Get the abundance counts.

Usage

```
counts(biom)
```

Arguments

biom — A BIOM object, as returned from read.biom.

Value

A numeric matrix of the sample abundance counts in biom.

See Also

Other accessor functions: info(), metadata(), nsamples(), ntaxa(), phylogeny(), sample.names(), sequences(), taxa.names(), taxa.ranks(), taxonomy()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

counts(biom)[1:4,1:5]
```

---

info *Get biom's misc information.*

Description

Get biom's misc information.

Usage

```
info(biom)
```

Arguments

biom	A BIOM object, as returned from read.biom.

Value

A data frame of the metadata in biom.

See Also

Other accessor functions: counts(), metadata(), nsamples(), ntaxa(), phylogeny(), sample.names(), sequences(), taxa.names(), taxa.ranks(), taxonomy()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

info(biom)
```

metadata *Get the sample metadata.*

Description

Get the sample metadata.

Usage

```
metadata(biom)
```

Arguments

biom	A BIOM object, as returned from read.biom.

Value

A data frame of the metadata in biom.

See Also

Other accessor functions: counts(), info(), nsamples(), ntaxa(), phylogeny(), sample.names(), sequences(), taxa.names(), taxa.ranks(), taxonomy()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

metadata(biom)[1:4,1:3]
```

`nsamples` *Number of samples in a BIOM.*

Description

Number of samples in a BIOM.

Usage

```
nsamples(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

The number of samples present.

See Also

Other accessor functions: `counts()`, `info()`, `metadata()`, `ntaxa()`, `phylogeny()`, `sample.names()`, `sequences()`, `taxa.names()`, `taxa.ranks()`, `taxonomy()`

Examples

```
    library(rbiom)

    infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
    biom <- read.biom(infile)

    nsamples(biom)
```

`ntaxa` *Number of taxa in a BIOM.*

Description

Number of taxa in a BIOM.

Usage

```
ntaxa(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

The number of taxa present.

See Also

Other accessor functions: `counts()`, `info()`, `metadata()`, `nsamples()`, `phylogeny()`, `sample.names()`, `sequences()`, `taxa.names()`, `taxa.ranks()`, `taxonomy()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

ntaxa(biom)
```

`phylogeny` *Get the phylogenetic tree.*

Description

Get the phylogenetic tree.

Usage

```
phylogeny(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

A `phylo` class object of the tree in `biom`.

See Also

Other accessor functions: `counts()`, `info()`, `metadata()`, `nsamples()`, `ntaxa()`, `sample.names()`, `sequences()`, `taxa.names()`, `taxa.ranks()`, `taxonomy()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

summary(phylogeny(biom))
```

---

print *Summarize the contents of a BIOM object*

Description

Summarize the contents of a BIOM object

Usage

```
## S3 method for class 'BIOM'
print(x, ...)
```

Arguments

x	A BIOM object, as returned from read.biom.
...	Not used.

Value

NULL (invisibly)

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

print(biom)
```

rarefy — *Subset counts so that all samples have the same number of observations.*

Description

Subset counts so that all samples have the same number of observations.

Usage

```
rarefy(biom, depth = NULL, seed = 0)
```

Arguments

`biom` A `matrix`, `simple_triplet_matrix`, or `BIOM` object, as returned from read.biom. For matrices, the rows and columns are assumed to be the taxa and samples, respectively.

`depth` The number of observations to keep, per sample. If set to `NULL`, a depth will be automatically selected. Samples that have fewer than this number of observations will be dropped. If called on data with non-integer abundances, values will be re-scaled to integers between 1 and `depth` such that they sum to `depth`.

`seed` An integer to use for seeding the random number generator. If you need to create different random rarefactions of the same `BIOM` object, set this seed value to a different number each time.

Value

A `matrix`, `simple_triplet_matrix`, or `BIOM` object, depending on the input object type. The type of object provided is the same type that is returned. The retained observations are randomly selected, based on a seed value derived from the `BIOM` object. Therefore, rarefying the same biom to the same depth will always produce the same resultant rarification.

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)
range(slam::col_sums(biom$counts))

biom <- rarefy(biom, depth=1000)
range(slam::col_sums(biom$counts))
```

rbiom *rbiom: Read/Write, Transform, and Summarize BIOM Data*

Description

A toolkit for working with Biological Observation Matrix (BIOM) files. Features include reading/writing all BIOM formats, rarefaction, alpha diversity, beta diversity (including UniFrac), summarizing counts by taxonomic level, and sample subsetting. Standalone functions for reading, writing, and subsetting phylogenetic trees are also provided. All CPU intensive operations are encoded in C with multi-thread support.

Multithreading

Many `rbiom` functions support multithreading:

The default behavior of these function is to run on as many cores as are available in the local compute environment. If you wish to limit the number of simultaneous threads, set `RcppParallel`'s `numThreads` option. For instance:

```
RcppParallel::setThreadOptions(numThreads = 4)
```

read.biom *Extracts counts, metadata, taxonomy, and phylogeny from a biom file.*

Description

Extracts counts, metadata, taxonomy, and phylogeny from a biom file.

Usage

```
read.biom(src, tree = "auto", prune = FALSE)
```

Arguments

`src` Input data as either a file path, URL, or JSON string. `read.biom` can read BIOM files formatted according to both the version 1.0 (JSON) and 2.1 (HDF5) specifications as well as classical tabular format. URLs must begin with `http://`, `https://`, `ftp://`, or `ftps://`. JSON files must have `{` as their first non-whitespace character. Compressed (gzip or bzip2) BIOM files are also supported. NOTE: to read HDF5 formatted BIOM files, the BioConductor R package `rhdf5` must be installed.

`tree` The default value of `auto` will read the tree from the BIOM file specified in `src`, if present. The value `TRUE` will do the same, but will generate an error message if a tree is not present. Setting `tree=FALSE` will return a `BIOM` object without any tree data. You may also provide a file path, URL, or Newick string to load that tree data into the final `BIOM` object.

`prune` Should samples and taxa with zero observations be discarded?

Value

A BIOM class object containing the parsed data. This object can be treated as a list with the following named elements:

counts A numeric slam sparse matrix of observation counts. Taxa (OTUs) as rows and samples as columns.

metadata A data frame containing any embedded metadata. Row names are sample IDs.

taxonomy Character matrix of taxonomic names, if given. Row names are taxa (OTU) IDs. Column rows are named Kingdom, Phylum, Class, Order, Family, Genus, Species, and Strain, or TaxLvl.1, TaxLvl.2, ... , TaxLvl.N when more than 8 levels of taxonomy are encoded in the biom file.

phylogeny An object of class phylo defining the phylogenetic relationships between the taxa. Although the official specification for BIOM only includes phylogenetic trees in BIOM version 2.1, if a BIOM version 1.0 file includes a phylogeny entry with newick data, then it will be loaded here as well. The **ape** package has additional functions for working with phylo objects.

sequences A named character vector, where the names are taxonomic identifiers and the values are the sequences they represent. These values are not part of the official BIOM specification, but will be read and written when defined.

info A list of other attributes defined in the BIOM file, such as id, type, format, format_url, generated_by, date, matrix_type, matrix_element_type, Comment, and shape

metadata, taxonomy, and phylogeny are optional components of the BIOM file specification and therefore will be empty in the returned object when they are not provided by the BIOM file.

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

summary(biom)

# Taxa Abundances
as.matrix(biom$counts[1:4,1:4])

top5 <- names(head(rev(sort(slam::row_sums(biom$counts))), 5))
biom$taxonomy[top5,c('Family', 'Genus')]
as.matrix(biom$counts[top5, 1:6])

# Metadata
table(biom$metadata$Sex, biom$metadata$`Body Site`)
sprintf("Mean age: %.1f", mean(biom$metadata$Age))

# Phylogenetic tree
tree <- biom$phylogeny
top5.tree <- rbiom::subtree(tree, top5)
ape::plot.phylo(top5.tree)
```

`read.fasta` *Parse a fasta file into a named character vector.*

Description

Parse a fasta file into a named character vector.

Usage

```
read.fasta(file, ids = NULL)
```

Arguments

`file`	A file with fasta-formatted sequences. Can optionally be compressed with gzip, bzip2, xz, or lzma.
`ids`	Character vector of IDs to retrieve. The default, `NULL`, will retrieve everything.

Value

A named character vector in which names are the fasta headers and values are the sequences.

`read.tree` *Read a newick formatted phylogenetic tree.*

Description

A phylogenetic tree is required for computing UniFrac distance matrices. You can load a tree either from a file or by providing the tree string directly. This tree must be in Newick format, also known as parenthetic format and New Hampshire format.

Usage

```
read.tree(src)
```

Arguments

`src`	Input data as either a file path, URL, or Newick string. URLs must begin with `http://`, `https://`, `ftp://`, or `ftps://`. Newick strings must have `(` as their first non-whitespace character. Compressed (gzip or bzip2) Newick files are also supported.

Value

A `phylo` class object representing the tree.

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree <- read.tree(infile)

tree <- read.tree("
    (t9:0.99,((t5:0.87,t2:0.89):0.51,(((t10:0.16,(t7:0.83,t4:0.96)
    :0.94):0.69,(t6:0.92,(t3:0.62,t1:0.85):0.54):0.23):0.74,t8:0.1
    2):0.43):0.67);")
```

`sample.names` *Get the sample names.*

Description

Get the sample names.

Usage

```
sample.names(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

A character vector of the sample IDs / names in `biom`.

See Also

Other accessor functions: `counts()`, `info()`, `metadata()`, `nsamples()`, `ntaxa()`, `phylogeny()`, `sequences()`, `taxa.names()`, `taxa.ranks()`, `taxonomy()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

sample.names(biom)
```

`select` *Reduce samples to a specific list*

Description

Reduce samples to a specific list

Usage

```
select(biom, samples = NULL, nTop = NULL, nRandom = NULL, seed = 0)
```

Arguments

`biom` A BIOM object, as returned from read.biom.

`samples` Sample names, indices, or a logical vector identifying the samples to keep. The latter two should be based on the order of sample names given by `colnames(biom$counts)`.

`nTop` Selects this number of samples, taking the sample with the most observations first, then the sample with the second-most observations, etc. If `nTop` is higher than the number of samples in the dataset, the entire dataset will be returned. See note.

`nRandom` Randomly selects this number of samples. If higher than the number of samples in the dataset, the entire dataset will be returned. See note.

`seed` Random seed, used when selecting `nRandom` samples.

Note: Generally, you will specify only one of the filters: `samples`, `nTop`, or `nRandom`. However, specifying multiple filters is allowed; they will be applied in the order listed above.

Value

A `BIOM` object.

See Also

`subset`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

ex1 <- select(biom, c('HMP14', 'HMP22', 'HMP03'))
ex2 <- select(biom, c(32, 11, 28, 16, 46, 5))
ex3 <- select(biom, 1:50 %% 6 == 0)
ex4 <- select(biom, nRandom = 10)
ex5 <- select(biom, nTop = 5)
```

```
ex6 <- select(biom, samples = 10:40, nTop = 20, nRandom = 10)
```

sequences *DNA sequence associated with each taxonomic identifier.*

Description

DNA sequence associated with each taxonomic identifier.

Usage

```
sequences(biom)
```

Arguments

biom	A BIOM object, as returned from read.biom.

Value

A named character vector of sequences in biom. If this data is not present, then returns NULL.

See Also

Other accessor functions: counts(), info(), metadata(), nsamples(), ntaxa(), phylogeny(), sample.names(), taxa.names(), taxa.ranks(), taxonomy()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

sequences(biom)[1:4]

# Write to a compressed fasta file in the temporary directory:
seqs <- sequences(biom)
conn <- bzfile(file.path(tempdir(), "Sequences.fa.bz2"), "w")
cat(sprintf(">%s\n%s", names(seqs), seqs), file=conn, sep="\n")
close(conn)

# You can also use the write.fasta function for this task:
write.fasta(biom, file.path(tempdir(), "Sequences.fa.gz"))
```

subset *Subset samples using the BIOM object's metadata*

Description

Subset samples using the BIOM object's metadata

Usage

```
## S3 method for class 'BIOM'
subset(x, ...)
```

Arguments

`x`	A BIOM object, as returned from read.biom.
`...`	Test to run on the metadata to identify samples to retain.

Value

A `BIOM` object.

See Also

`select`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

ex1 <- subset(biom, Age > 30)
ex2 <- subset(biom, `Body Site` %in% c("Saliva", "Stool"))
ex3 <- subset(biom, Age < 25 & BMI > 22)
```

subtree *Create a subtree by specifying tips to keep.*

Description

Create a subtree by specifying tips to keep.

Usage

```
subtree(tree, tips)
```

Arguments

`tree`	A phylo object, as returned from read.tree.
`tips`	A character, numeric, or logical vector of tips to keep.

Value

A `phylo` object for the subtree.

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree <- read.tree(infile)

leafs   <- tips(tree)
subtree <- subtree(tree, head(leafs))
```

`taxa.names` *Get the taxa names.*

Description

Get the taxa names.

Usage

```
taxa.names(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

A character vector of the taxa IDs / names in `biom`.

See Also

Other accessor functions: `counts`(), `info`(), `metadata`(), `nsamples`(), `ntaxa`(), `phylogeny`(), `sample.names`(), `sequences`(), `taxa.ranks`(), `taxonomy`()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

taxa.names(biom) %>% head()
```

---

taxa.ranks *Get the taxa ranks.*

Description

Get the taxa ranks.

Usage

```
taxa.ranks(biom)
```

Arguments

`biom`	A `BIOM` object, as returned from read.biom.

Value

A character vector of the taxa ranks in `biom`.

See Also

Other accessor functions: `counts()`, `info()`, `metadata()`, `nsamples()`, `ntaxa()`, `phylogeny()`, `sample.names()`, `sequences()`, `taxa.names()`, `taxonomy()`

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

taxa.ranks(biom)
```

taxa.rollup *Generate a matrix of samples by taxa, at the specified taxonomic rank.*

Description

Generate a matrix of samples by taxa, at the specified taxonomic rank.

Usage

```
taxa.rollup(biom, rank = "OTU", map = NULL, lineage = FALSE, sparse = FALSE)
```

Arguments

biom
: A `matrix`, `simple_triplet_matrix`, or `BIOM` object, as returned from read.biom. For matrices, the rows and columns are assumed to be the taxa and samples, respectively.

rank
: The taxonomic rank. E.g. **"OTU"**, **"Phylum"**, etc. May also be given numerically: 0 for OTU, 1 for the highest level (i.e. Kingdom), and extending to the number of taxonomic ranks encoded in the original biom file. See example below to fetch the names of all available ranks.

map
: A character matrix defining the value that each taxa IDs is assigned for each taxonomic rank. If `map=NULL` and `biom` is a `BIOM` class object, the map will be automatically loaded from `biom$taxonomy`. `map` must not be `null` when `biom` is a `matrix` or `simple_triplet_matrix`. See the example below for an example of `map`'s structure.

lineage
: Include all ranks in the name of the taxa. For instance, setting to `TRUE` will produce `Bacteria; Actinobacteria; Coriobacteriia; Coriobacteriales`. Whereas setting to `FALSE` (the default) will return simply `Coriobacteriales`. You want to set this to TRUE if you have genus names (such as *Incertae_Sedis*) that map to multiple higher level ranks.

sparse
: If true, returns a sparse matrix as described by `slam::simple_triplet_matrix`, otherwise returns a normal R matrix object. Sparse matrices will likely be considerably more memory efficient in this scenario.

Value

A numeric matrix with samples as column names, and taxonomic identifiers as row names.

Examples

```
    library(rbiom)

    infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
    biom <- read.biom(infile)

    colnames(biom$taxonomy)
```

```
phyla <- taxa.rollup(biom, 'Phylum')
phyla[1:4,1:6]

# Custom matrices should be formatted like so:
counts <- as.matrix(biom$counts)
map    <- biom$taxonomy

counts[1:3,1:6]
map[1:3,1:4]

phyla <- taxa.rollup(counts, 'Phylum', map=map)
phyla[1:3,1:6]
```

taxonomy *Get the taxonomy table.*

Description

Get the taxonomy table.

Usage

```
taxonomy(biom)
```

Arguments

biom	A BIOM object, as returned from read.biom.

Value

A character matrix of the named taxonomies in biom.

See Also

Other accessor functions: counts(), info(), metadata(), nsamples(), ntaxa(), phylogeny(), sample.names(), sequences(), taxa.names(), taxa.ranks()

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)

taxonomy(biom)[1:4,]
```

tips *Names of a phylogenetic tree's tips/leafs.*

Description

Names of a phylogenetic tree's tips/leafs.

Usage

```
tips(x)
```

Arguments

x	A phylo object, as returned from read.tree..

Value

A character vector with the leaf names.

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree <- read.tree(infile)

leafs   <- tips(tree)
subtree <- subtree(tree, head(leafs))
```

unifrac *Compute Weighted and Unweighted UniFrac distance matrices.*

Description

This is the function called internally by beta.div, but is made visible here so you can use it with matrices and trees without having to first convert them to BIOM objects.

Usage

```
unifrac(biom, weighted = TRUE, tree = NULL)
```

Arguments

`biom`	A `matrix`, `simple_triplet_matrix`, or `BIOM` object, as returned from read.biom. For matrices, the rows and columns are assumed to be the taxa and samples, respectively.
`weighted`	Use weighted UniFrac, which takes abundance into account rather than simply presence/absence.
`tree`	A `phylo` object providing a phylogenetic tree for the taxa names in `biom`. If `tree=NULL`, then the tree will be loaded from `biom`, if encoded there.

Value

A distance matrix of class `dist`.

Examples

```
library(rbiom)

infile <- system.file("extdata", "hmp50.bz2", package = "rbiom")
biom <- read.biom(infile)
biom <- select(biom, 1:10)

dm <- unifrac(biom)
plot(hclust(dm), cex=.8)
as.matrix(dm)[1:4,1:4]

# Using a custom matrix and tree
mtx <- matrix(sample.int(12*20), ncol=20)
dimnames(mtx) <- list(LETTERS[1:12], letters[1:20])
tree <- ape::as.phylo(hclust(dist(mtx)))

dm <- unifrac(mtx, tree=tree)
as.matrix(dm)[1:4,1:4]
```

`write.biom` *Write counts, metadata, taxonomy, and phylogeny to a biom file.*

Description

Write counts, metadata, taxonomy, and phylogeny to a biom file.

Usage

```
write.biom(biom, file, format = "json")
```

Arguments

`biom`	The BIOM object to save to the file.
`file`	Path to the output file.
`format`	Options are **"tab"**, **"json"**, and **"hdf5"**, corresponding to classic tabular format, biom format version 1.0 and biom version 2.1, respectively. Abbreviations are also accepted. See `http://biom-format.org/documentation/` for details. NOTE: to write HDF5 formatted BIOM files, the BioConductor R package `rhdf5` must be installed.

Value

On success, returns `NULL` invisibly.

`write.fasta` *Write sequences from a BIOM object to a file in fasta format.*

Description

Write sequences from a BIOM object to a file in fasta format.

Usage

```
write.fasta(seqs, outfile)
```

Arguments

`seqs`	A named character vector where names are sequence names and values are the sequences. Also accepts a `BIOM` object which contains sequences.
`outfile`	Path to the output fasta file. Files ending in `.gz` or `.bz2` will be compressed accordingly.

Value

On success, returns `NULL` invisibly.

write.tree *Write a newick formatted phylogenetic tree.*

Description

Write a newick formatted phylogenetic tree.

Usage

```
write.tree(tree = NULL, file = NULL)
```

Arguments

`tree`	A `phylo` object, as returned from read.tree. Also accepts a `BIOM` object if it has a phylogentic tree.
`file`	Filename or connection to write the newick file to (optional).

Value

If file is NULL, the newick string as a character vector. Otherwise, the return value from `writeChar`, typically invsible(NULL).

Examples

```
library(rbiom)

infile <- system.file("extdata", "newick.tre", package = "rbiom")
tree   <- read.tree(infile)
newick <- write.tree(tree)
```

write.xlsx *Write data and summary information to a Microsoft Excel-compatible workbook.*

Description

Write data and summary information to a Microsoft Excel-compatible workbook.

Usage

```
write.xlsx(biom, outfile, depth = NULL, seed = 0)
```

Arguments

`biom`	The BIOM object to save to the file.
`outfile`	Path to the output xlsx file.
`depth`	Depth to rarefy to. See `rarefy` function for details. Only use `depth` with `BIOM` files of type 'OTU table' and integer count values.
`seed`	Random seed to use in rarefying. See `rarefy` function for details.

Value

On success, returns `NULL` invisibly.

Note

Any data frame attributes on `biom` will be included as separate worksheets. An attribute named 'Reads Per Step' is treated specially and merged with the usual 'Reads Per Sample' tab - if provided, its row names should match those in `biom` exactly.

Index