Package ‘biomformat’

June 4, 2026

Version 1.41.0

Date 2026-03-15

License GPL-2

Title An interface package for the BIOM file format

Type Package

BugReports https://github.com/joey711/biomformat/issues

URL https://github.com/joey711/biomformat/, http://biom-format.org/

Description This is an R package for interfacing with the BIOM file format. This package includes basic tools for reading biom-format files, accessing and subsetting data tables from a biom object (which is more complex than a single table), as well as limited support for writing a biom-object back to a biom-format file. The design of this API is intended to match the python API and other tools included with the biom-format project, but with a decidedly ``R flavor'' that should be familiar to R users. This includes S4 classes and methods, as well as extensions of common core functions/methods.

Imports jsonlite (>= 0.9.16), Matrix (>= 1.7-0)

Depends R (>= 4.1), methods

Suggests testthat (>= 0.10), knitr (>= 1.10), BiocStyle (>= 1.6), rmarkdown (>= 0.7), SummarizedExperiment, S4Vectors, tibble, rhdf5

VignetteBuilder knitr

Collate 'allClasses.R' 'allPackage.R' 'IO-methods.R' 'BIOM-class.R' 'SE-methods.R' 'tidy-methods.R' 'validity-methods.R'

biocViews ImmunoOncology, DataImport, Metagenomics, Microbiome

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.3

git_url https://git.bioconductor.org/packages/biomformat

git_branch devel

git_last_commit a7f85e0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-06-04

Author Paul J. McMurdie [aut, cre],
Joseph N. Paulson [aut]

Maintainer Paul J. McMurdie <`joey711@gmail.com`>

Contents

`biomformat-package` *biomformat: An R interface to the BIOM file format*

Description

This is an R package for interfacing with the biom-format. It includes basic utilities for reading and writing BIOM format using R, and native R methods for interacting with biom-data that maps to functionality in other languages that support biom-format, like python.

Author(s)

Paul J. McMurdie II <`mcmurdie@stanford.edu`>

References

`http://www.biom-format.org/`

See Also

Useful links:

- `https://github.com/joey711/biomformat/`
- `http://biom-format.org/`
- Report bugs at `https://github.com/joey711/biomformat/issues`

`as.data.frame.biom` *Coerce a biom object to a long-format tidy data frame*

Description

Returns one row per (feature, sample) pair with columns `feature_id`, `sample_id`, and `count` (or `value` for character-type BIOM data), plus any sample and observation metadata columns appended via a left-join.

Usage

```
## S3 method for class 'biom'
as.data.frame(x, ...)
```

Arguments

`x`	A `biom-class` object.
`...`	Ignored; present for S3 generic compatibility.

Details

Column name conflicts between the base triplet and joined metadata are resolved automatically by `merge` using the suffixes `""` and `"_sample"` (for sample metadata) or `"_feature"` (for observation metadata).

The implementation is pure base R – no tidyverse dependency. For a tibble-wrapped version see `as_tibble.biom`.

Value

A `data.frame` in long (tidy) format with at least three columns: `feature_id`, `sample_id`, `count`. Additional columns from `sample_metadata` and `observation_metadata` are appended when present.

See Also

`biom_data`, `as_tibble.biom`

Examples

```
biom_file <- system.file("extdata", "rich_dense_otu_table.biom",
                         package = "biomformat")
x <- read_biom(biom_file)
df <- as.data.frame(x)
head(df)
dim(df)   # 30 rows (5 features x 6 samples), 14 columns
```

`as_tibble.biom` *Coerce a biom object to a tidy tibble*

Description

A thin wrapper around `as.data.frame.biom` that returns a `tibble` instead of a plain `data.frame`. Requires the **tibble** package (in `Suggests`).

Usage

```
as_tibble.biom(x, ...)
```

Arguments

`x`	A `biom-class` object.
`...`	Passed to `as_tibble`.

Value

A `tibble` in long (tidy) format. See `as.data.frame.biom` for column details.

See Also

`as.data.frame.biom`

Examples

```
biom_file <- system.file("extdata", "rich_dense_otu_table.biom",
                         package = "biomformat")
x <- read_biom(biom_file)
if (requireNamespace("tibble", quietly = TRUE)) {
  # Call the method directly to ensure correct S3 dispatch
  tbl <- as_tibble.biom(x)
  tbl
}
```

`biom` *Build and return an instance of the biom-class.*

Description

This is for instantiating a biom object within R (`biom-class`), and assumes relevant data is already available in R. This is different than reading a biom file into R. If you are instead interested in importing a biom file into R, you should use the `read_biom` function. This function is made available (exported) so that advanced-users/developers can easily represent analogous data in this structure if needed. However, most users are expected to instead rely on the `read_biom` function for data import, followed by accessor functions that extract R-friendly subsets of the data stored in the biom-format derived list.

Usage

```
biom(x)

## S4 method for signature 'list'
biom(x)
```

Arguments

x	(REQUIRED). A named list conforming to conventions arising from the `fromJSON` function reading a biom-format file with default settings. See `read_biom` for more details about data import and `biom-class` for more details about accessor functions that extract R-friendly subsets of the data and metadata stored in `x`.

Details

`biom()` is a constructor method. This is the main method suggested for constructing an experiment-level (`biom-class`) object from its component data.

Value

An instance of the `biom-class`.

See Also

Function to create a biom object from R data, `make_biom`.

Definition of the `biom-class`.

The `read_biom` import function.

Function to write a biom format file from a biom object, `write_biom`

Accessor functions like `header`.

Examples

```
#
# import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
x = read_biom(biom_file)
show(x)
print(x)
header(x)
biom_data(x)
biom_shape(x)
nrow(x)
ncol(x)
observation_metadata(x)
sample_metadata(x)
```

`biom-class` *The biom format data class.*

Description

This class inherits from the `list-class`, with validity checks specific to the definition to the biom-format. Effectively this means the list must have certain index names, some elements of which must have a specific structure or value. For further details see the biom-format definition. Importantly, this means other special properties of lists, like operations with `$` and single- or double-square-braces are also supported; as-is the `apply`-family function that can operate on lists. Note that some features of the biom-format can be essentially empty, represented by the string `"null"` in the file. These fields are returned as `NULL` when accessed by an accessor function.

See Also

The constructor, `biom`

Accessor functions:

`header`, `biom_shape`, `nrow`, `ncol`, `matrix_element_type`, `biom_data`, `observation_metadata`, `sample_metadata`

Examples

```
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
x = read_biom(biom_file)
header(x)
biom_shape(x)
nrow(x)
ncol(x)
rownames(x)
colnames(x)
matrix_element_type(x)
biom_data(x)
observation_metadata(x)
sample_metadata(x)
```

`biom_data` *Access main data observation matrix data from `biom-class`.*

Description

Retrieve and organize main data from `biom-class`, represented as a matrix with index names.

Usage

```
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,missing,missing'
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,character,missing'
```

```
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,character,ANY'
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,ANY,character'
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,numeric,missing'
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,missing,numeric'
biom_data(x, rows, columns, parallel = FALSE)

## S4 method for signature 'biom,numeric,numeric'
biom_data(x, rows, columns, parallel = FALSE)
```

Arguments

`x`	(Required). An instance of the `biom-class`.
`rows`	(Optional). The subset of row indices described in the returned object. For large datasets, specifying the row subset here, rather than after creating the whole matrix first, can improve speed/efficiency. Can be vector of index numbers (`numeric-class`) or index names (`character-class`).
`columns`	(Optional). The subset of column indices described in the returned object. For large datasets, specifying the column subset here, rather than after creating the whole matrix first, can improve speed/efficiency. Can be vector of index numbers (`numeric-class`) or index names (`character-class`).
`parallel`	(Optional). Logical. Defunct; retained for backward compatibility only. Passing `TRUE` emits a deprecation warning and has no effect. Will be removed in a future version. @return A matrix containing the main observation data, with index names. The type of data (numeric or character) will depend on the results of `matrix_element_type(x)`. The class of the matrix returned will depend on the sparsity of the data, and whether it has numeric or character data. For now, only numeric data can be stored in a `Matrix-class`, which will be stored sparsely, if possible. Character data will be returned as a vanilla `matrix-class`.

Examples

```
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
min_sparse_file  = system.file("extdata", "min_sparse_otu_table.biom", package = "biomformat")
rich_dense_file  = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
rich_sparse_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
rich_dense_char  = system.file("extdata", "rich_dense_char.biom", package = "biomformat")
rich_sparse_char  = system.file("extdata", "rich_sparse_char.biom", package = "biomformat")
# Read the biom-format files
x1 = read_biom(min_dense_file)
x2 = read_biom(min_sparse_file)
x3 = read_biom(rich_dense_file)
x4 = read_biom(rich_sparse_file)
x5 = read_biom(rich_dense_char)
```

```
x6 = read_biom(rich_sparse_char)
# Extract the data matrices
biom_data(x1)
biom_data(x2)
biom_data(x3)
biom_data(x4)
biom_data(x5)
biom_data(x6)
```

`biom_shape` *The matrix dimensions of a `biom-class` object.*

Description

The matrix dimensions of a `biom-class` object.

Usage

```
biom_shape(x)

## S4 method for signature 'biom'
biom_shape(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

A length two `integer-class` vector indicating the `nrow` and `ncol` of the main data matrix stored in `x`.

See Also

`biom-class`

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
biom_shape(x)
```

`biom_to_SummarizedExperiment` *Convert a biom object to a SummarizedExperiment*

Description

Converts a `biom-class` object into a `SummarizedExperiment`, placing the count/value matrix in `assay(., "counts")`, sample metadata in `colData()`, and observation/feature metadata in `rowData()`.

Usage

```
biom_to_SummarizedExperiment(x)
```

Arguments

`x`	A `biom-class` object.

Details

Both `colData` and `rowData` are coerced to `DataFrame`. When a biom object carries no sample or observation metadata (i.e. the corresponding accessor returns `NULL`), an empty `DataFrame` with the correct row names is substituted so that the resulting `SummarizedExperiment` is always fully valid.

Value

A `SummarizedExperiment` with:

`assay("counts")`	The feature-by-sample count (or value) matrix returned by `biom_data`.
`colData()`	Per-sample metadata from `sample_metadata`, or an empty `DataFrame`.
`rowData()`	Per-feature metadata from `observation_metadata`, or an empty `DataFrame`.

See Also

`read_biom`, `biom_data`, `sample_metadata`, `observation_metadata`

Examples

```
biom_file <- system.file("extdata", "rich_sparse_otu_table.biom",
                         package = "biomformat")
x <- read_biom(biom_file)
if (requireNamespace("SummarizedExperiment", quietly = TRUE)) {
  se <- biom_to_SummarizedExperiment(x)
  se
  # S4 coercion syntax also works:
  se2 <- as(x, "SummarizedExperiment")
  identical(SummarizedExperiment::assay(se), SummarizedExperiment::assay(se2))
}
```

`colnames,biom-method` *Method extensions to* `colnames` *for* `biom-class` *objects.*

Description

See the general documentation of `colnames` method for expected behavior.

Usage

```
## S4 method for signature 'biom'
colnames(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

The number of columns in `x`. A length 1 `integer-class`.

See Also

`nrow`

`colnames`

`biom_shape`

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
colnames(x)
```

`header` *Extract the header from a* `biom-class` *object as a list.*

Description

Extract the header from a `biom-class` object as a list.

Usage

```
header(x)

## S4 method for signature 'biom'
header(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

A list containing the header data. That is, all the required elements that are not the main data or index metadata.

Examples

```
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
x = read_biom(biom_file)
header(x)
```

`make_biom` *Create a* `biom-class` *from* `matrix-class` *or* `data.frame`.

Description

This function creates a valid instance of the `biom-class` from standard base-R objects like `matrix-class` or `data.frame`. This makes it possible to export any contingency table data represented in R to the biom-format, regardless of its source. The object returned by this function is appropriate for writing to a `.biom` file using the `write_biom` function. The sparse biom-format is not (yet) supported.

Usage

```
make_biom(
  data,
  sample_metadata = NULL,
  observation_metadata = NULL,
  id = NULL,
  matrix_element_type = "int"
)
```

Arguments

`data`	(Required). `matrix-class` or `data.frame`. A contingency table. Observations / features / OTUs / species are rows, samples / sites / libraries are columns.
`sample_metadata`	(Optional). A `matrix-class` or `data.frame` with the number of rows equal to the number of samples in `data`. Sample covariates associated with the count data. This should look like the table returned by `sample_metadata` on a valid instance of the `biom-class`.
`observation_metadata`	(Optional). A `matrix-class` or `data.frame` with the number of rows equal to the number of features / species / OTUs / genes in `data`. This should look like the table returned by `observation_metadata` on a valid instance of the `biom-class`.
`id`	(Optional). Character string. Identifier for the project.
`matrix_element_type`	(Optional). Character string. Either 'int' or 'float'

Details

The BIOM file format (canonically pronounced biome) is designed to be a general-use format for representing biological sample by observation contingency tables. BIOM is a recognized standard for the Earth Microbiome Project and is a Genomics Standards Consortium candidate project. Please see the biom-format home page for more details.

Value

An object of `biom-class`.

References

`http://biom-format.org/`

See Also

`write_biom`

`biom-class`

`read_biom`

Examples

```
# import with default parameters, specify a file
biomfile = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
x = read_biom(biomfile)
data = biom_data(x)
data
smd = sample_metadata(x)
smd
omd = observation_metadata(x)
omd
# Make a new biom object from component data
y = make_biom(data, smd, omd)
# Won't be identical to x because of header info.
identical(x, y)
# The data components should be, though.
identical(observation_metadata(x), observation_metadata(y))
identical(sample_metadata(x), sample_metadata(y))
identical(biom_data(x), biom_data(y))
## Quickly show that writing and reading still identical.
# Define a temporary directory to write .biom files
tempdir = tempdir()
write_biom(x, biom_file=file.path(tempdir, "x.biom"))
write_biom(y, biom_file=file.path(tempdir, "y.biom"))
x1 = read_biom(file.path(tempdir, "x.biom"))
y1 = read_biom(file.path(tempdir, "y.biom"))
identical(observation_metadata(x1), observation_metadata(y1))
identical(sample_metadata(x1), sample_metadata(y1))
identical(biom_data(x1), biom_data(y1))
```

`matrix_element_type` *Access class of data in the matrix elements of a* `biom-class` *object*

Description

Access class of data in the matrix elements of a `biom-class` object

Usage

```
matrix_element_type(x)

## S4 method for signature 'biom'
matrix_element_type(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

A `character-class` string indicating the class of the data stored in the main observation matrix of `x`, with expected values `"int"`, `"float"`, `"unicode"`.

See Also

`biom-class`

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
matrix_element_type(x)
```

`ncol,biom-method` *Method extensions to* `ncol` *for* `biom-class` *objects.*

Description

See the general documentation of `ncol` method for expected behavior.

Usage

```
## S4 method for signature 'biom'
ncol(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

The number of columns in `x`. A length 1 `integer-class`.

See Also

`nrow`

`ncol`

`biom_shape`

Examples

```
# import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
ncol(x)
```

`nrow,biom-method` *Method extensions to* `nrow` *for* `biom-class` *objects.*

Description

See the general documentation of `nrow` method for expected behavior.

Usage

```
## S4 method for signature 'biom'
nrow(x)
```

Arguments

`x`	(Required). An instance of the `biom-class`.

Value

The number of rows in `x`. A length 1 `integer-class`.

See Also

`ncol`

`nrow`

`biom_shape`

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
nrow(x)
```

`observation_metadata` *Access observation (row) meta data from* `biom-class`*.*

Description

Retrieve and organize meta data from `biom-class`, represented as a `data.frame` (if possible) or a list, with proper index names.

Usage

```
observation_metadata(x, rows, parallel = FALSE)

## S4 method for signature 'biom,missing'
observation_metadata(x, rows, parallel = FALSE)

## S4 method for signature 'biom,character'
observation_metadata(x, rows, parallel = FALSE)

## S4 method for signature 'biom,numeric'
observation_metadata(x, rows, parallel = FALSE)
```

Arguments

`x`	(Required). An instance of the `biom-class`.
`rows`	(Optional). The subset of row indices described in the returned object. For large datasets, specifying the row subset here, – rather than first creating the complete data object – can improve speed/efficiency. This parameter can be vector of index numbers (`numeric-class`) or index names (`character-class`).
`parallel`	(Optional). Logical. Defunct; retained for backward compatibility only. Passing `TRUE` emits a deprecation warning and has no effect. Will be removed in a future version.

Value

A `data.frame` or `list` containing the meta data, with index names. The precise form of the object returned depends on the metadata stored in `x`. A `data.frame` is created if possible.

Examples

```
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
min_sparse_file = system.file("extdata", "min_sparse_otu_table.biom", package = "biomformat")
rich_dense_file = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
rich_sparse_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
rich_dense_char  = system.file("extdata", "rich_dense_char.biom", package = "biomformat")
rich_sparse_char  = system.file("extdata", "rich_sparse_char.biom", package = "biomformat")
# Read the biom-format files
x1 = read_biom(min_dense_file)
x2 = read_biom(min_sparse_file)
x3 = read_biom(rich_dense_file)
x4 = read_biom(rich_sparse_file)
x5 = read_biom(rich_dense_char)
```

```
x6 = read_biom(rich_sparse_char)
# Extract metadata
observation_metadata(x1)
observation_metadata(x2)
observation_metadata(x3)
observation_metadata(x3, 2:4)
observation_metadata(x3, 2)
observation_metadata(x3, c("GG_OTU_3", "GG_OTU_4", "whoops"))
observation_metadata(x4)
observation_metadata(x5)
observation_metadata(x6)
```

`read_biom` *Read a biom-format file, returning a* `biom-class`.

Description

Import the data from a biom-format file into R, represented as an instance of the `biom-class`; essentially a `list` with special constraints that map to the biom-format definition.

Usage

```
read_biom(biom_file)
```

Arguments

`biom_file`	(Required). A character string indicating the file location of the biom formatted file. This is a HDF5 or JSON formatted file specific to biological datasets. The format is formally defined at the biom-format definition and depends on the versioning.

Details

The BIOM file format (canonically pronounced biome) is designed to be a general-use format for representing biological sample by observation contingency tables. BIOM is a recognized standard for the Earth Microbiome Project and is a Genomics Standards Consortium candidate project. Please see the biom-format home page for more details.

It is tempting to include an argument identifying the biom-format version number of the data file being imported. However, the biom-format version number is a required field in the biom-format definition. Rather than duplicate this formal specification and allow the possibility of a conflict, the version number of the biom format will be referred to only by the "format" field in the biom formatted data, or its representation in R.

Value

An instance of the `biom-class`.

References

`http://biom-format.org/`

See Also

Function to create a biom object from R data, `make_biom`.

Definition of the `biom-class`.

Function to write a biom format file from a biom object, `write_biom`

Accessor functions like `header`.

Examples

```
# # # import with default parameters, specify a file
biom_file <- system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
biom_file
read_biom(biom_file)
biom_file <- system.file("extdata", "min_sparse_otu_table.biom", package = "biomformat")
biom_file
read_biom(biom_file)
## The previous examples use system.file() because of constraints in specifying a fixed
##   path within a reproducible example in a package.
## In practice, however, you can simply provide "hard-link"
## character string path to your file:
# mybiomfile <- "path/to/my/biomfile.biom"
# read_biom(mybiomfile)
```

`read_hdf5_biom` *Read in a biom-format v2 (HDF5) file, returning a* `list`.

Description

This function reads a BIOM v2 HDF5 file directly. In normal use, you should call `read_biom`, which automatically routes to this function when it detects the HDF5 magic bytes. Call this function directly only when you are certain the file is HDF5 format.

Usage

```
read_hdf5_biom(biom_file)
```

Arguments

`biom_file`	(Required). A character string path to an HDF5-format BIOM file.

Details

The `rhdf5` package is required at runtime. If it is not available, or if the underlying HDF5 system libraries are absent, a graceful R-level warning is thrown instead of a fatal C-level abort.

Value

A named list representing the BIOM data, suitable for passing to `biom()`.

References

http://biom-format.org/

See Also

read_biom, make_biom, biom-class, write_biom, write_hdf5_biom.

Examples

```
biom_file <- system.file("extdata", "rich_sparse_otu_table_hdf5.biom", package = "biomformat")
x = read_hdf5_biom(biom_file)
x = biom(x)
outfile = tempfile()
write_biom(x, outfile)
y = read_biom(outfile)
identical(observation_metadata(x),observation_metadata(y))
identical(sample_metadata(x),sample_metadata(y))
identical(biom_data(x), biom_data(y))
```

`rownames,biom-method` *Method extensions to* rownames *for* biom-class *objects.*

Description

See the general documentation of `rownames` method for expected behavior.

Usage

```
## S4 method for signature 'biom'
rownames(x)
```

Arguments

`x`	(Required). An instance of the biom-class.

Value

The number of columns in `x`. A length 1 integer-class.

See Also

nrow

rownames

biom_shape

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
rownames(x)
```

`sample_metadata` *Access meta data from* `biom-class`.

Description

Retrieve and organize meta data from `biom-class`, represented as a `data.frame` (if possible, or a list) with proper index names.

Usage

```
sample_metadata(x, columns, parallel = FALSE)

## S4 method for signature 'biom,missing'
sample_metadata(x, columns, parallel = FALSE)

## S4 method for signature 'biom,character'
sample_metadata(x, columns, parallel = FALSE)

## S4 method for signature 'biom,numeric'
sample_metadata(x, columns, parallel = FALSE)
```

Arguments

`x`	(Required). An instance of the `biom-class`.
`columns`	(Optional). The subset of column indices described in the returned object. For large datasets, specifying the column subset here, rather than after creating the whole matrix first, can improve speed/efficiency. Can be vector of index numbers (`numeric-class`) or index names (`character-class`).
`parallel`	(Optional). Logical. Defunct; retained for backward compatibility only. Passing `TRUE` emits a deprecation warning and has no effect. Will be removed in a future version.

Value

A `data.frame` or `list` containing the meta data, with index names. The precise form of the object returned depends on the metadata stored in `x`. A `data.frame` is created if possible.

Examples

```
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
min_sparse_file = system.file("extdata", "min_sparse_otu_table.biom", package = "biomformat")
rich_dense_file = system.file("extdata", "rich_dense_otu_table.biom", package = "biomformat")
rich_sparse_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
min_dense_file  = system.file("extdata", "min_dense_otu_table.biom", package = "biomformat")
rich_dense_char  = system.file("extdata", "rich_dense_char.biom", package = "biomformat")
rich_sparse_char  = system.file("extdata", "rich_sparse_char.biom", package = "biomformat")
# Read the biom-format files
x1 = read_biom(min_dense_file)
x2 = read_biom(min_sparse_file)
x3 = read_biom(rich_dense_file)
x4 = read_biom(rich_sparse_file)
x5 = read_biom(rich_dense_char)
```

```
x6 = read_biom(rich_sparse_char)
# Extract metadata
sample_metadata(x1)
sample_metadata(x2)
sample_metadata(x3)
sample_metadata(x3, 1:4)
sample_metadata(x4)
sample_metadata(x5)
sample_metadata(x6)
```

`show,biom-method` *Method extensions to show for biom objects.*

Description

See the general documentation of `show` method for expected behavior.

Usage

```
## S4 method for signature 'biom'
show(object)
```

Arguments

`object`	biom-class object

See Also

`show`

Examples

```
# # # import with default parameters, specify a file
biom_file = system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
(x = read_biom(biom_file) )
show(x)
```

`write_biom` *Write a biom-format v1 file, returning a* `biom-class`.

Description

Write a biom-format v1 file, returning a `biom-class`.

Usage

```
write_biom(x, biom_file)
```

Arguments

`x`	(Required). A biom object that is going to be written to file as a proper biom formatted file, adhering to the biom-format definition.
`biom_file`	(Required). A character string indicating the file location of the biom formatted file. This is a JSON formatted file specific to biological datasets. The format is formally defined at the biom-format definition

Details

`write_biom()` serialises the entire BIOM object to a single JSON string via `toJSON`. Because R character strings are limited to $2^{31} - 1$ bytes, this function will fail with a *"character strings are limited to 2^31-1 bytes"* error for very large datasets. If your dataset has more than a few thousand samples or features, use `write_hdf5_biom` instead — the HDF5 format has no such size constraint.

Value

Nothing. The first argument, `x`, is written to a file.

References

`http://biom-format.org/`

See Also

Function to create a biom object from R data, `make_biom`.

Definition of the `biom-class`.

The `read_biom` import function.

Accessor functions like `header`.

Examples

```
biom_file <- system.file("extdata", "rich_sparse_otu_table.biom", package = "biomformat")
x = read_biom(biom_file)
outfile = tempfile()
write_biom(x, outfile)
y = read_biom(outfile)
identical(x, y)
```

`write_hdf5_biom` *Write a biom object to an HDF5 (BIOM v2) file.*

Description

Serialises a `biom-class` object to the BIOM v2 HDF5 format. Both the sample-major and observation-major compressed-sparse representations required by the spec are written, along with sample and observation metadata.

Usage

```
write_hdf5_biom(x, biom_file)
```

Arguments

`x`	(Required). A `biom-class` object.
`biom_file`	(Required). Character string path to the output file. Any existing file at that path is overwritten.

Details

The `rhdf5` package is required. If it is not installed a clear error is thrown. In normal use you should prefer `write_biom` for JSON (BIOM v1) output; use this function only when HDF5 output is explicitly needed.

Value

`biom_file` (invisibly).

References

`http://biom-format.org/`

See Also

`read_hdf5_biom`, `write_biom`, `biom-class`, `make_biom`.

Examples

```
biom_file <- system.file("extdata", "rich_sparse_otu_table.biom",
                          package = "biomformat")
x <- read_biom(biom_file)
outfile <- tempfile(fileext = ".biom")
write_hdf5_biom(x, outfile)
y <- read_biom(outfile)
identical(biom_data(x), biom_data(y))
```

Index