# BIOM Documentation

***Release 1.1.1***

**The BIOM Project**

December 02, 2013

# Contents

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.

The BIOM format is designed for general use in broad areas of comparative -omics. For example, in marker-gene surveys, the primary use of this format is to represent OTU tables: the observations in this case are OTUs and the matrix contains counts corresponding to the number of times each OTU is observed in each sample. With respect to metagenome data, this format would be used to represent metagenome tables: the observations in this case might correspond to SEED subsystems, and the matrix would contain counts corresponding to the number of times each subsystem is observed in each metagenome. Similarly, with respect to genome data, this format may be used to represent a set of genomes: the observations in this case again might correspond to SEED subsystems, and the counts would correspond to the number of times each subsystem is observed in each genome.

There are two components to the BIOM project: first is definition of the BIOM format, and second is development of support objects in multiple programming languages to support the use of BIOM in diverse bioinformatics applications. The version of the BIOM file format is independent of the version of the *biom-format* software.

CHAPTER 1

# Contents

## 1.1 BIOM Documentation

These pages provide format specifications and API information for the BIOM table objects.

### 1.1.1 The biom file format

The BIOM project consists of two independent tools: the *biom-format* software package, which contains software tools for working with BIOM-formatted files and the tables they represent; and the BIOM file format. As of the 1.0.0 software version and the 1.0 file format version, the version of the software and the file format are independent of one another. Version specific documentation of the file formats can be found on the following pages.

#### The biom file format: Version 1.0

The `biom` format is based on JSON to provide the overall structure for the format. JSON is a widely supported format with native parsers available within many programming languages.

Required top-level fields:

```
id                  : <string or null> a field that can be used to id a table (or null)
format              : <string> The name and version of the current biom format
format_url          : <url> A string with a static URL providing format details
type                : <string> Table type (a controlled vocabulary)
                      Acceptable values:
                       "OTU table"
                       "Pathway table"
                       "Function table"
                       "Ortholog table"
                       "Gene table"
                       "Metabolite table"
                       "Taxon table"
generated_by        : <string> Package and revision that built the table
date                : <datetime> Date the table was built (ISO 8601 format)
rows                : <list of objects> An ORDERED list of obj describing the rows
                      (explained in detail below)
columns             : <list of objects> An ORDERED list of obj  describing the columns
                      (explained in detail below)
```

```
matrix_type         : <string> Type of matrix data representation (a controlled vocabulary)
                      Acceptable values:
                       "sparse" : only non-zero values are specified
                       "dense" : every element must be specified
matrix_element_type : Value type in matrix (a controlled vocabulary)
                      Acceptable values:
                       "int" : integer
                       "float" : floating point
                       "unicode" : unicode string
shape               : <list of ints>, the number of rows and number of columns in data
data                : <list of lists>, counts of observations by sample
                       if matrix_type is "sparse", [[row, column, value],
                                                    [row, column, value],
                                                    ...]
                       if matrix_type is "dense",  [[value, value, value, ...],
                                                    [value, value, value, ...],
                                                    ...]
```

Optional top-level fields:

```
comment             : <string> A free text field containing any information that you
                       feel is relevant (or just feel like sharing)
```

The rows value is an ORDERED list of objects where each object corresponds to a single row in the matrix. Each object can currently store arbitrary keys, although this might become restricted based on table type. Each object must provide, at the minimum:

```
id                  : <string> an arbitrary UNIQUE identifier
metadata            : <an object or null> A object containing key, value metadata pairs
```

The columns value is an ORDERED list of objects where each object corresponds to a single column in the matrix. Each object can currently store arbitrary keys, although this might become restricted based on table type. Each object must provide, at the minimum:

```
id                  : <string> an arbitrary UNIQUE identifier
metadata            : <an object or null> A object containing key, value metadata pairs
```

### Example biom files

Below are examples of minimal and rich biom files in both sparse and dense formats. To decide which of these you should generate for new data types, see the section on *Tips and FAQs regarding the BIOM file format*.

#### Minimal sparse OTU table

```
{
    "id":null,
    "format": "Biological Observation Matrix 0.9.1-dev",
    "format_url": "http://biom-format.org/documentation/format_versions/biom-1.0.html",
    "type": "OTU table",
    "generated_by": "QIIME revision 1.4.0-dev",
    "date": "2011-12-19T19:00:00",
    "rows":[
            {"id":"GG_OTU_1", "metadata":null},
            {"id":"GG_OTU_2", "metadata":null},
            {"id":"GG_OTU_3", "metadata":null},
            {"id":"GG_OTU_4", "metadata":null},
            {"id":"GG_OTU_5", "metadata":null}
```

```
        ],
    "columns": [
            {"id":"Sample1", "metadata":null},
            {"id":"Sample2", "metadata":null},
            {"id":"Sample3", "metadata":null},
            {"id":"Sample4", "metadata":null},
            {"id":"Sample5", "metadata":null},
            {"id":"Sample6", "metadata":null}
        ],
    "matrix_type": "sparse",
    "matrix_element_type": "int",
    "shape": [5, 6],
    "data":[[0,2,1],
            [1,0,5],
            [1,1,1],
            [1,3,2],
            [1,4,3],
            [1,5,1],
            [2,2,1],
            [2,3,4],
            [2,4,2],
            [3,0,2],
            [3,1,1],
            [3,2,1],
            [3,5,1],
            [4,1,1],
            [4,2,1]
           ]
}
```

**Minimal dense OTU table**

```
{
    "id":null,
    "format": "Biological Observation Matrix 0.9.1-dev",
    "format_url": "http://biom-format.org/documentation/format_versions/biom-1.0.html",
    "type": "OTU table",
    "generated_by": "QIIME revision 1.4.0-dev",
    "date": "2011-12-19T19:00:00",
    "rows":[
            {"id":"GG_OTU_1", "metadata":null},
            {"id":"GG_OTU_2", "metadata":null},
            {"id":"GG_OTU_3", "metadata":null},
            {"id":"GG_OTU_4", "metadata":null},
            {"id":"GG_OTU_5", "metadata":null}
        ],
    "columns": [
            {"id":"Sample1", "metadata":null},
            {"id":"Sample2", "metadata":null},
            {"id":"Sample3", "metadata":null},
            {"id":"Sample4", "metadata":null},
            {"id":"Sample5", "metadata":null},
            {"id":"Sample6", "metadata":null}
        ],
    "matrix_type": "dense",
    "matrix_element_type": "int",
    "shape": [5,6],
    "data":  [[0,0,1,0,0,0],
```

```
                [5,1,0,2,3,1],
                [0,0,1,4,2,0],
                [2,1,1,0,0,1],
                [0,1,1,0,0,0]]
}
```

**Rich sparse OTU table**

```
{
 "id":null,
 "format": "Biological Observation Matrix 0.9.1-dev",
 "format_url": "http://biom-format.org/documentation/format_versions/biom-1.0.html",
 "type": "OTU table",
 "generated_by": "QIIME revision 1.4.0-dev",
 "date": "2011-12-19T19:00:00",
 "rows":[
    {"id":"GG_OTU_1", "metadata":{"taxonomy":["k__Bacteria", "p__Proteobacteria", "c__Gammaproteobact
    {"id":"GG_OTU_2", "metadata":{"taxonomy":["k__Bacteria", "p__Cyanobacteria", "c__Nostocophycideae
    {"id":"GG_OTU_3", "metadata":{"taxonomy":["k__Archaea", "p__Euryarchaeota", "c__Methanomicrobia",
    {"id":"GG_OTU_4", "metadata":{"taxonomy":["k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Ha
    {"id":"GG_OTU_5", "metadata":{"taxonomy":["k__Bacteria", "p__Proteobacteria", "c__Gammaproteobact
    ],
 "columns":[
    {"id":"Sample1", "metadata":{
                               "BarcodeSequence":"CGCTTATCGAGA",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"gut",
                               "Description":"human gut"}},
    {"id":"Sample2", "metadata":{
                               "BarcodeSequence":"CATACCAGTAGC",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"gut",
                               "Description":"human gut"}},
    {"id":"Sample3", "metadata":{
                               "BarcodeSequence":"CTCTCTACCTGT",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"gut",
                               "Description":"human gut"}},
    {"id":"Sample4", "metadata":{
                               "BarcodeSequence":"CTCTCGGCCTGT",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"skin",
                               "Description":"human skin"}},
    {"id":"Sample5", "metadata":{
                               "BarcodeSequence":"CTCTCTACCAAT",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"skin",
                               "Description":"human skin"}},
    {"id":"Sample6", "metadata":{
                               "BarcodeSequence":"CTAACTACCAAT",
                               "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                               "BODY_SITE":"skin",
                               "Description":"human skin"}}
            ],
 "matrix_type": "sparse",
 "matrix_element_type": "int",
 "shape": [5, 6],
 "data":[[0,2,1],
```

```
          [1,0,5],
          [1,1,1],
          [1,3,2],
          [1,4,3],
          [1,5,1],
          [2,2,1],
          [2,3,4],
          [2,5,2],
          [3,0,2],
          [3,1,1],
          [3,2,1],
          [3,5,1],
          [4,1,1],
          [4,2,1]
        ]
}
```

**Rich dense OTU table**

```
{
 "id":null,
 "format": "Biological Observation Matrix 0.9.1-dev",
 "format_url": "http://biom-format.org/documentation/format_versions/biom-1.0.html",
 "type": "OTU table",
 "generated_by": "QIIME revision 1.4.0-dev",
 "date": "2011-12-19T19:00:00",
 "rows":[
    {"id":"GG_OTU_1", "metadata":{"taxonomy":["k__Bacteria", "p__Proteobacteria", "c__Gammaproteobact
    {"id":"GG_OTU_2", "metadata":{"taxonomy":["k__Bacteria", "p__Cyanobacteria", "c__Nostocophycideae
    {"id":"GG_OTU_3", "metadata":{"taxonomy":["k__Archaea", "p__Euryarchaeota", "c__Methanomicrobia",
    {"id":"GG_OTU_4", "metadata":{"taxonomy":["k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Ha
    {"id":"GG_OTU_5", "metadata":{"taxonomy":["k__Bacteria", "p__Proteobacteria", "c__Gammaproteobact
    ],
 "columns":[
    {"id":"Sample1", "metadata":{
                            "BarcodeSequence":"CGCTTATCGAGA",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"gut",
                            "Description":"human gut"}},
    {"id":"Sample2", "metadata":{
                            "BarcodeSequence":"CATACCAGTAGC",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"gut",
                            "Description":"human gut"}},
    {"id":"Sample3", "metadata":{
                            "BarcodeSequence":"CTCTCTACCTGT",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"gut",
                            "Description":"human gut"}},
    {"id":"Sample4", "metadata":{
                            "BarcodeSequence":"CTCTCGGCCTGT",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"skin",
                            "Description":"human skin"}},
    {"id":"Sample5", "metadata":{
                            "BarcodeSequence":"CTCTCTACCAAT",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"skin",
```

```
                            "Description":"human skin"}},
    {"id":"Sample6", "metadata":{
                            "BarcodeSequence":"CTAACTACCAAT",
                            "LinkerPrimerSequence":"CATGCTGCCTCCCGTAGGAGT",
                            "BODY_SITE":"skin",
                            "Description":"human skin"}}
           ],
 "matrix_type": "dense",
 "matrix_element_type": "int",
 "shape": [5,6],
 "data":  [[0,0,1,0,0,0],
           [5,1,0,2,3,1],
           [0,0,1,4,2,0],
           [2,1,1,0,0,1],
           [0,1,1,0,0,0]]
}
```

Release versions contain three integers in the following format: `major-version.incremental-version.minor-version`. When `-dev` is appended to the end of a version string that indicates a development (or between-release version). For example, `1.0.0-dev` would refer to the development version following the 1.0.0 release.

### 1.1.2 Tips and FAQs regarding the BIOM file format

#### Should I generate sparse or dense biom files?

In general, we recommend using the sparse format for your biom files. These will be a lot smaller than the dense format biom files when your data is sparse (i.e., more than 85% of your counts are zero). This is common for OTU tables and metagenome tables, and you'll want to investigate whether it's true for your data. If you currently format your data in tab-separated tables where observations are rows and samples are columns, you can format that file to be convertible to biom format with the `convert_biom.py`. Here you can create dense and sparse formats, and see which file size is smaller. See the section on *Converting between file formats*.

#### Motivation for the BIOM format

The BIOM format was motivation by several goals. First, to facilitate efficient handling and storage of large, sparse biological contingency tables; second, to support encapsulation of core study data (contingency table data and sample/observation metadata) in a single file; and third, to facilitate the use of these tables between tools that support this format (e.g., passing of data between QIIME, MG-RAST, and VAMPS.).

##### Efficient handling and storage of very large tables

In QIIME, we began hitting limitations with OTU table objects when working with thousands of samples and hundreds of thousands of OTUs. In the near future we expect that we'll be dealing with hundreds of thousands of samples in single analyses.

The OTU table format up to QIIME 1.4.0 involved a dense matrix: if an OTU was not observed in a given sample, that would be indicated with a zero. We now primarily represent OTU tables in a sparse format: if an OTU is not observed in a sample, there is no count for that OTU. The two ways of representing this data are exemplified here.

A dense representation of an OTU table:

```
OTU ID PC.354  PC.355  PC.356
OTU0   0   0   4
OTU1   6   0   0
```

```
OTU2   1   0   7
OTU3   0   0   3
```

A sparse representation of an OTU table:

```
PC.354 OTU1 6
PC.354 OTU2 1
PC.356 OTU0 4
PC.356 OTU2 7
PC.356 OTU3 3
```

OTU table data tends to be sparse (e.g., greater than 90% of counts are zero, and frequently as many as 99% of counts are zero) in which case the latter format is more convenient to work with as it has a smaller memory footprint. Both of these representations are supported in the biom-format project via dense and sparse Table types. Generally if less than 85% of your counts are zero, a dense representation will be more efficient.

#### Encapsulation of core study data (OTU table data and sample/OTU metadata) in a single file

The JSON-format OTU table allow for storage of arbitrary amounts of sample and OTU metadata in a single file. Sample metadata corresponds to what is generally found in QIIME mapping files. At this stage inclusion of this information in the OTU table file is optional, but it may be useful for sharing these files with other QIIME users and for publishing or archiving results of analyses. OTU metadata (generally a taxonomic assignment for an OTU) is also optional. In contrast to the previous OTU table format, you can now store more than one OTU metadata value in this field, so for example you can score taxonomic assignments based on two different taxonomic assignment approaches.

#### Facilitating the use of tables between tools that support this format

Different tools, such as QIIME, MG-RAST, and VAMPS work with similar data structures that represent different types of data. An example of this is a *metagenome* table that could be generated by MG-RAST (where for example, columns are metagenomes and rows are functional categories). Exporting this data from MG-RAST in a suitable format will allow for the application of many of the QIIME tools to this data (such as generation of alpha rarefaction plots or beta diversity ordination plots). This new format is far more general than previous formats, so will support adoption by groups working with different data types and is already being integrated to support transfer of data between QIIME, MG-RAST, and VAMPS.

### File extension

We recommend that BIOM files use the `.biom` extension.

## 1.1.3 biom-format `Table` objects

The biom-format project provides rich `Table` objects to support use of the BIOM file format. The objects encapsulate matrix data (such as OTU counts) and abstract the interaction away from the programmer. This provides the immediate benefit of the programmer not having to worry about what the underlying data object is, and in turn allows for different data representations to be supported. Currently, biom-format supports a `dense` object built off of `numpy.array` (NumPy) and a `sparse` object built off of Python dictionaries.

### biom-format `table_factory` method

Generally, construction of a `Table` subclass will be through the `table_factory` method. This method facilitates any necessary data conversions and supports a wide variety of input data types.

### Description of available `Table` objects

There are multiple objects available but some of them are unofficial abstract base classes (does not use the `abc` module for historical reasons). In practice, the objects used should be the derived Tables such as `SparseOTUTable` or `DenseGeneTable`.

#### Abstract base classes

Abstract base classes establish standard interfaces for subclassed types and provide common functionality for derived types.

**`Table`** `Table` is a container object and an abstract base class that provides a common and required API for subclassed objects. Through the use of private interfaces, it is possible to create public methods that operate on the underlying datatype without having to implement each method in each subclass. For instance, `Table.iterSamplesData` will return a generator that always yields `numpy.array` vectors for each sample regardless of how the table data is actually stored. This functionality results from derived classes implementing private interfaces, such as `Table._conv_to_np`.

**`OTUTable`** The `OTUTable` base class provides functionality specific for OTU tables. Currently, it only provides a static private member variable that describes its `BIOM` type. This object was stubbed out incase future methods are developed that do not make sense with the context of, say, an MG-RAST metagenomic abundance table. It is advised to always use an object that subclasses `OTUTable` if the analysis is on OTU data.

**`PathwayTable`** A table type to represent gene pathways.

**`FunctionTable`** A table type to represent gene functions.

**`OrthologTable`** A table type to represent gene orthologs.

**`GeneTable`** A table type to represent genes.

**`MetaboliteTable`** A table type to represent metabolite profiles.

**`TaxonTable`** A table type to represent taxonomies.

#### Container classes

The container classes implement required private member variable interfaces as defined by the `Table` abstract base class. Specifically, these objects define the ways in which data is moved into and out of the contained data object. These are fully functional and usable objects, however they do not implement table type specifc functionality.

**`SparseTable`** The subclass `SparseTable` can be derived for use with table data. This object implemented all of the required private interfaces specified by the `Table` base class. The object contains a `_data` private member variable that is an instance of `biom.table.SparseDict`. It is advised to used derived objects of SparseTable if the data being operated on is sparse.

**DenseTable** The `DenseTable` object fulfills all private member methods stubbed out by the `Table` base class. The dense table contains a private member variable that is an instance of `numpy.array`. The `array` object is a matrix that contains all values including zeros. It is advised to use this table only if the number of samples and observations is reasonble. Unfortunately, it isn't reasonable to define reasonable in this context. However, if either the number of observations or the number of samples is > 1000, it would probably be a good idea to rely on a `SparseTable`.

#### Table type objects

The table type objects define variables and methods specific to a table type. These inherit from a `Container Class` and a table type base class, and are therefore instantiable. Generally you'll instantiate tables with `biom.table.table_factory`, and one of these will be passed as the `constructor` argument.

**DenseOTUTable**

**SparseOTUTable**

**DensePathwayTable**

**SparsePathwayTable**

**DenseFunctionTable**

**SparseFunctionable**

**DenseOrthologTable**

**SparseOrthologTable**

**DenseGeneTable**

**SparseGeneTable**

**DenseMetaboliteTable**

**SparseMetaboliteTable**

### 1.1.4 Converting between file formats

**The `convert_biom.py` script in the biom-format project can be used to convert between biom and tab-delimited table format**

- converting biom format tables to tab-delimited tables for easy viewing in programs such as Excel
- converting between sparse and dense biom formats

---

**Note:** The tab-delimited tables are commonly referred to as the *classic format* tables, while BIOM formatted tables are referred to as *biom tables*.

---

#### General usage examples

Convert a tab-delimited table to sparse biom format. Note that you *must* specify the type of table here:

```
convert_biom.py -i table.txt -o table.from_txt.biom --biom_table_type="otu table"
```

Convert a tab-delimited table to dense biom format:

```
convert_biom.py -i table.txt -o table.dense.biom --biom_table_type="otu table" --biom_type=dense
```

Convert biom format to tab-delimited table format:

```
convert_biom.py -i table.biom -o table.from_biom.txt -b
```

Convert dense biom format to sparse biom format:

```
convert_biom.py -i table.dense.biom -o table.sparse.biom --dense_biom_to_sparse_biom
```

Convert sparse biom format to dense biom format:

```
convert_biom.py -i table.sparse.biom -o table.dense.biom --sparse_biom_to_dense_biom
```

Convert biom format to classic format, including the `taxonomy` observation metadata as the last column of the classic format table. Because the BIOM format can support an arbitrary number of observation (or sample) metadata entries, and the classic format can support only a single observation metadata entry, you must specify which of the observation metadata entries you want to include in the output table:

```
convert_biom.py -i table.biom -o table.from_biom_w_taxonomy.txt -b --header_key taxonomy
```

Convert biom format to classic format, including the `taxonomy` observation metadata as the last column of the classic format table, but renaming that column as `ConsensusLineage`. This is useful when using legacy tools that require a specific name for the observation metadata column.:

```
convert_biom.py -i table.biom -o table.from_biom_w_consensuslineage.txt -b --header_key taxonomy --ou
```

#### Special case usage examples

##### Converting QIIME 1.4.0 and earlier OTU tables to BIOM format

If you are converting a QIIME 1.4.0 or earlier OTU table to BIOM format, there are a few steps to go through. First, for convenience, you might want to rename the `ConsensusLineage` column `taxonomy`. You can do this with the following command:

```
sed 's/Consensus Lineage/ConsensusLineage/' < otu_table.txt | sed 's/ConsensusLineage/taxonomy/' > ot
```

Then, you'll want to perform the conversion including a step to convert the taxonomy *string* from the classic OTU table to a taxonomy *list*, as it's represented in QIIME 1.4.0-dev and later:

```
convert_biom.py -i otu_table.taxonomy.txt -o otu_table.from_txt.biom --biom_table_type="otu table" --
```

### 1.1.5 Adding sample and observation metadata to biom files

Frequently you'll have an existing BIOM file and want to add sample and/or observation metadata to it. For samples, metadata is frequently environmental or technical details about your samples: the subject that a sample was collected from, the pH of the sample, the PCR primers used to amplify DNA from the samples, etc. For observations, metadata is frequently a categorization of the observation: the taxonomy of an OTU, or the EC hierarchy of a gene. You can use the `add_metadata.py` script to add this information to an existing BIOM file.

To get help with `add_metadata.py` you can call:

```
add_metadata.py -h
```

This script takes a BIOM file, and corresponding sample and/or observation mapping files. The following examples are used in the commands below. You can find these files in the `biom-format/examples` directory.

Your BIOM file might look like the following:

```
{
    "id":null,
    "format": "Biological Observation Matrix 1.0.0-dev",
    "format_url": "http://biom-format.org",
    "type": "OTU table",
    "generated_by": "some software package",
    "date": "2011-12-19T19:00:00",
    "rows":[
            {"id":"GG_OTU_1", "metadata":null},
            {"id":"GG_OTU_2", "metadata":null},
            {"id":"GG_OTU_3", "metadata":null},
            {"id":"GG_OTU_4", "metadata":null},
            {"id":"GG_OTU_5", "metadata":null}
        ],
    "columns": [
            {"id":"Sample1", "metadata":null},
            {"id":"Sample2", "metadata":null},
            {"id":"Sample3", "metadata":null},
            {"id":"Sample4", "metadata":null},
            {"id":"Sample5", "metadata":null},
            {"id":"Sample6", "metadata":null}
        ],
    "matrix_type": "sparse",
    "matrix_element_type": "int",
    "shape": [5, 6],
    "data":[[0,2,1],
            [1,0,5],
            [1,1,1],
            [1,3,2],
            [1,4,3],
            [1,5,1],
            [2,2,1],
            [2,3,4],
```

```
            [2,5,2],
            [3,0,2],
            [3,1,1],
            [3,2,1],
            [3,5,1],
            [4,1,1],
            [4,2,1]
           ]
}
```

A sample metadata mapping file could then look like the following. Notice that there is an extra sample in here with respect to the above BIOM table. Any samples in the mapping file that are not in the BIOM file are ignored.

```
#SampleID       BarcodeSequence DOB
# Some optional
# comment lines...
Sample1 AGCACGAGCCTA    20060805
Sample2 AACTCGTCGATG    20060216
Sample3 ACAGACCACTCA    20060109
Sample4 ACCAGCGACTAG    20070530
Sample5 AGCAGCACTTGT    20070101
Sample6 AGCAGCACAACT    20070716
```

An observation metadata mapping file might look like the following. Notice that there is an extra observation in here with respect to the above BIOM table. Any observations in the mapping file that are not in the BIOM file are ignored.

```
#OTUID  taxonomy        confidence
# Some optional
# comment lines
GG_OTU_0        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__       0.980
GG_OTU_1        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_2        Root;k__Bacteria        0.980
GG_OTU_3        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_4        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_5        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
```

### Adding metadata

To add sample metadata to a BIOM file, you can run the following:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_smd.biom --sample_mapping_fp sam_md.txt
```

To add observation metadata to a BIOM file, you can run the following:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_omd.biom --observation_mapping_fp obs_md.txt
```

You can also combine these in a single command to add both observation and sample metadata:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_md.biom --observation_mapping_fp obs_md.txt -
```

In the last case, the resulting BIOM file will look like the following:

```
{
    "columns": [
        {
            "id": "Sample1",
            "metadata": {
                "BarcodeSequence": "AGCACGAGCCTA",
```

```
                "DOB": "20060805"
            }
        },
        {
            "id": "Sample2",
            "metadata": {
                "BarcodeSequence": "AACTCGTCGATG",
                "DOB": "20060216"
            }
        },
        {
            "id": "Sample3",
            "metadata": {
                "BarcodeSequence": "ACAGACCACTCA",
                "DOB": "20060109"
            }
        },
        {
            "id": "Sample4",
            "metadata": {
                "BarcodeSequence": "ACCAGCGACTAG",
                "DOB": "20070530"
            }
        },
        {
            "id": "Sample5",
            "metadata": {
                "BarcodeSequence": "AGCAGCACTTGT",
                "DOB": "20070101"
            }
        },
        {
            "id": "Sample6",
            "metadata": {
                "BarcodeSequence": "AGCAGCACAACT",
                "DOB": "20070716"
            }
        }
    ],
    "data": [
        [0, 2, 1.0],
        [1, 0, 5.0],
        [1, 1, 1.0],
        [1, 3, 2.0],
        [1, 4, 3.0],
        [1, 5, 1.0],
        [2, 2, 1.0],
        [2, 3, 4.0],
        [2, 5, 2.0],
        [3, 0, 2.0],
        [3, 1, 1.0],
        [3, 2, 1.0],
        [3, 5, 1.0],
        [4, 1, 1.0],
        [4, 2, 1.0]
    ],
    "date": "2012-12-11T07:36:15.467843",
    "format": "Biological Observation Matrix 1.0.0",
```

```
    "format_url": "http://biom-format.org",
    "generated_by": "some software package",
    "id": null,
    "matrix_element_type": "float",
    "matrix_type": "sparse",
    "rows": [
        {
            "id": "GG_OTU_1",
            "metadata": {
                "confidence": "0.665",
                "taxonomy": "Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnos
            }
        },
        {
            "id": "GG_OTU_2",
            "metadata": {
                "confidence": "0.980",
                "taxonomy": "Root;k__Bacteria"
            }
        },
        {
            "id": "GG_OTU_3",
            "metadata": {
                "confidence": "1.000",
                "taxonomy": "Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnos
            }
        },
        {
            "id": "GG_OTU_4",
            "metadata": {
                "confidence": "0.842",
                "taxonomy": "Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnos
            }
        },
        {
            "id": "GG_OTU_5",
            "metadata": {
                "confidence": "1.000",
                "taxonomy": "Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnos
            }
        }
    ],
    "shape": [5, 6],
    "type": "OTU table"
}
```


### Processing metadata while adding

There are some additional parameters you can pass to this script for more complex processing.

You can tell the script to process certain metadata column values as integers (`--int_fields`), floating point (i.e., decimal or real) numbers (`--float_fields`), or as hierarchical semicolon-delimited data (`--sc_separated`).

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_md.biom --observation_mapping_fp obs_md.txt -
```

Here your resulting BIOM file will look like the following, where `DOB` values are now integers (compare to the above: they're not quoted now), `confidence` values are now floating point numbers (again, not quoted now), and

`taxonomy` values are now lists where each entry is a taxonomy level, opposed to above where they appear as a single semi-colon-separated string.


```
{
    "columns": [
        {
            "id": "Sample1",
            "metadata": {
                "BarcodeSequence": "AGCACGAGCCTA",
                "DOB": 20060805
            }
        },
        {
            "id": "Sample2",
            "metadata": {
                "BarcodeSequence": "AACTCGTCGATG",
                "DOB": 20060216
            }
        },
        {
            "id": "Sample3",
            "metadata": {
                "BarcodeSequence": "ACAGACCACTCA",
                "DOB": 20060109
            }
        },
        {
            "id": "Sample4",
            "metadata": {
                "BarcodeSequence": "ACCAGCGACTAG",
                "DOB": 20070530
            }
        },
        {
            "id": "Sample5",
            "metadata": {
                "BarcodeSequence": "AGCAGCACTTGT",
                "DOB": 20070101
            }
        },
        {
            "id": "Sample6",
            "metadata": {
                "BarcodeSequence": "AGCAGCACAACT",
                "DOB": 20070716
            }
        }
    ],
    "data": [
        [0, 2, 1.0],
        [1, 0, 5.0],
        [1, 1, 1.0],
        [1, 3, 2.0],
        [1, 4, 3.0],
        [1, 5, 1.0],
        [2, 2, 1.0],
        [2, 3, 4.0],
        [2, 5, 2.0],
        [3, 0, 2.0],
```

```
        [3, 1, 1.0],
        [3, 2, 1.0],
        [3, 5, 1.0],
        [4, 1, 1.0],
        [4, 2, 1.0]
    ],
    "date": "2012-12-11T07:30:29.870689",
    "format": "Biological Observation Matrix 1.0.0",
    "format_url": "http://biom-format.org",
    "generated_by": "some software package",
    "id": null,
    "matrix_element_type": "float",
    "matrix_type": "sparse",
    "rows": [
        {
            "id": "GG_OTU_1",
            "metadata": {
                "confidence": 0.665,
                "taxonomy": ["Root", "k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Clostridial
            }
        },
        {
            "id": "GG_OTU_2",
            "metadata": {
                "confidence": 0.98,
                "taxonomy": ["Root", "k__Bacteria"]
            }
        },
        {
            "id": "GG_OTU_3",
            "metadata": {
                "confidence": 1.0,
                "taxonomy": ["Root", "k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Clostridial
            }
        },
        {
            "id": "GG_OTU_4",
            "metadata": {
                "confidence": 0.842,
                "taxonomy": ["Root", "k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Clostridial
            }
        },
        {
            "id": "GG_OTU_5",
            "metadata": {
                "confidence": 1.0,
                "taxonomy": ["Root", "k__Bacteria", "p__Firmicutes", "c__Clostridia", "o__Clostridial
            }
        }
    ],
    "shape": [5, 6],
    "type": "OTU table"
}
```


If you have multiple fields that you'd like processed in one of these ways, you can pass a comma-separated list of field names (e.g., `--float_fields confidence,pH`).

#### Renaming (or naming) metadata columns while adding

You can also override the names of the metadata fields provided in the mapping files with the `--observation_header` and `--sample_header` parameters. This is useful if you want to rename metadata columns, or if metadata column headers aren't present in your metadata mapping file. If you pass either of these parameters, you must name all columns in order. If there are more columns in the metadata mapping file then there are headers, extra columns will be ignored (so this is also a useful way to select only the first n columns from your mapping file). For example, if you want to rename the `DOB` column in the sample metadata mapping you could do the following:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_smd.biom --sample_mapping_fp sam_md.txt --sam
```

If you have a mapping file without headers such as the following:

```
GG_OTU_0        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__       0.980
GG_OTU_1        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_2        Root;k__Bacteria        0.980
GG_OTU_3        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_4        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
GG_OTU_5        Root;k__Bacteria;p__Firmicutes;c__Clostridia;o__Clostridiales;f__Lachnospiraceae
```

you could name these while adding them as follows:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_omd.biom --observation_mapping_fp obs_md.txt
```

As a variation on the last command, if you only want to include the `taxonomy` column and exclude the `confidence` column, you could run:

```
add_metadata.py -i min_sparse_otu_table.biom -o table.w_omd.biom --observation_mapping_fp obs_md.txt
```

### 1.1.6 Sparse matrix backends

As the BIOM project evolves, so do the underlying data structures, leading to potential runtime trade offs between implementations. Currently, there are three distinct sparse matrix backends to BIOM: the `CSMat` (default as of BIOM v1.1), the `SparseMat` and the `SparseDict`. Specific differences are discussed below.

#### How to check what sparse backend is in use

To check what sparse backend is in use, simply execute `print_biom_python_config.py`. The last line shows the SparseObj type. For instance:

```
print_biom_python_config.py

System information
==================
      Platform: darwin
Python/GCC version: 2.7.2 (default, Mar 23 2012, 13:31:52)  [GCC 4.2.1 (Based on Apple Inc. build 565
Python executable: /Users/mcdonald/bin/python

Dependency versions
===================
              NumPy version:    1.6.1
biom-format library version:    1.1.0-dev
 biom-format script version:    1.1.0-dev

biom-format package information
```

```
==============================
SparseObj type: biom.csmat.CSMat
```

The last line indicates that the `CSMat` object is in use.

### Changing the sparse backend

There are two methods to change the backend that is used. The first method is by copying the `biom_config` file located under `support_files/` and placing it in your home directory as `~/.biom_config`. Then, edit ``~/.biom_config` and replace the current backend type with the desired type.

The second method is to set the environment variable `$BIOM_CONFIG_FP` to a file path of your choice, and place the following into that file:

```
python_code_sparse_backend      <BACKEND TYPE>
```

Where `<BACKEND TYPE>` is replaced by the specific backend implementation to use.

### Sparse matrix backend descriptions

Different sparse matrix backends have different performance characteristics. As BIOM changes over time, additional methods may be added that address specific runtime concerns.

#### CSMat

The default sparse backend is the `CSMat`. `CSMat` implements coordinate list, compressed sparse row and compressed sparse column formats and facilitates interaction with these representations. This backend will have the lowest memory footprint. In general, this backend should be the fastest. However, it has been observed that under certain circumstances, this backend may not perform the best.

#### SparseMat

The `SparseMat` is built using a combination of Python objects, a Cython wrapper and C++. It implements the dictionary of keys sparse matrix representation. This method performs pretty well, but has an increasing memory footprint as the number of nonzero values increases. Under some situations, specifically those that require a large number of `Table` creations, this backend has about a 15-20% reduced runtime over `CSMat`.

#### SparseDict

The `SparseDict` is the naive pure Python implementation. This was first implemented as a test backend, and it is not advised to use this object.

## 1.2 BIOM License

```
            GNU GENERAL PUBLIC LICENSE
               Version 2, June 1991

 Copyright 2007-2009 (C) 1989, 1991 Free Software Foundation, Inc.
                     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
```

of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below,

refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

```
Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
```

```
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
```

address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright 2007-2009 (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by

```
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright 2007-2009 (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

CHAPTER 2

# BIOM version

The latest official version of the biom-format project is 1.1.1 and of the BIOM file format is 1.0. Details on the file format can be found here.

CHAPTER 3

# Installing the biom-format project

To install the biom-format project, you can download the release version biom-format-1.1.1, or work with the development version. Generally we recommend working with the release version as it will be more stable, but if you want access to the latest features (and can tolerate some instability) you should work with the development version.

**The biom-format project has the following dependencies:**

- Python 2 (2.6 or later)
- numpy (1.3.0 or later)
- gcc >= 4.1.2 (optional; used for more efficient sparse table representations)
- cython >= 0.14.1 (optional; used for more efficient sparse table representations)

We'll illustrate the install process in the $HOME/code directory. You can either work in this directory on your system (creating it, if necessary, by running mkdir $HOME/code) or replace all occurrences of $HOME/code in the following instructions with your working directory. Change to this directory to start the install process:

```
cd $HOME/code
```

To install the release version, download from biom-format-1.1.1, uncompress the file, and change to the resulting directory:

```
wget ftp://thebeast.colorado.edu/pub/biom-format-releases/biom-format-1.1.1.tar.gz
tar -xvzf biom-format-1.1.1.tar.gz
cd $HOME/code/biom-format-1.1.1
```

Alternatively, to install the development version, pull it from github, and change to the resulting directory:

```
git clone git://github.com/biom-format/biom-format.git
cd $HOME/code/biom-format
```

To install (either the development or release version), follow these steps:

```
sudo python setup.py install
```

If you do not have sudo access on your system (or don't want to install the biom-format project in the default location) you'll need to install the library code and scripts in specified directories, and then tell your system where to look for those files. You can do this as follows:

```
echo "export PATH=$HOME/bin/:$PATH" >> $HOME/.bashrc
echo "export PYTHONPATH=$HOME/lib/:$PYTHONPATH" >> $HOME/.bashrc
```

```
mkdir -p $HOME/bin $HOME/lib/
source $HOME/.bashrc
python setup.py install --install-scripts=$HOME/bin/ --install-purelib=$HOME/lib/ --install-lib=$HOM
```

You should then have access to the biom-format project. You can test this by running the following command:

```
python -c "from biom import __version__; print __version__"
```

You should see the current version of the biom-format project.

Next you can run:

```
which convert_biom.py
```

You should get a file path ending with `convert_biom.py` printed to your screen if it is installed correctly.

CHAPTER 4

# Citing the BIOM project

You can cite the BIOM format as follows:

The Biological Observation Matrix (BIOM) format or: how I learned to stop worrying and love the ome-ome.
Daniel McDonald, Jose C. Clemente, Justin Kuczynski, Jai Ram Rideout, Jesse Stombaugh, Doug Wendel, Andreas Wilke, Susan Huse, John Hufnagle, Folker Meyer, Rob Knight, and J. Gregory Caporaso.
GigaScience, June 2012.

CHAPTER 5

# Development team

The biom-format project was conceived of and developed by the QIIME, MG-RAST, and VAMPS development groups to support interoperability of our software packages. If you have questions about the biom-format project you can contact gregcaporaso@gmail.com.