BIOM Documentation

Release 1.0.0

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" --
```

CHAPTER 2

BIOM version

The latest official version of the biom-format project is 1.0.0 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-1.0.0, 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-1.0.0, uncompress the file, and change to the resulting directory:

```
wget https://github.com/downloads/biom-format/biom-format/biom-format-1.0.0.tgz
tar -xvzf biom-format-1.0.0.tgz
cd $HOME/code/biom-format-1.0.0
```

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.