

An Introduction to *OTUbase*

Modified: 8 September 2010. Compiled: October 24, 2023

```
> library("OTUbase")
```

The *OTUbase* package provides an organized structure for OTU (Operational Taxonomic Unit) data analysis. In addition, it provides a similar structure for general read-taxonomy classification type data. *OTUbase* provides some basic functions to analyze the data as well.

1 A simple workflow

This section walks through a simple workflow using a small example dataset. It demonstrates the main features of *OTUbase*. The data used for this example comes from a dataset described in "Microbial diversity in the deep sea and the underexplored 'rare biosphere'" by Sogin et al. (PNAS 2006). The complete dataset is available through PNAS. A random set of 1000 sequences was taken from this dataset.

1.1 Sample meta data

Sample metadata is collected along with the sample. This data may include any number of different pieces of information about the sample. In the example dataset, the meta data is provided in Table 1 of Sogin et al. This file is named 'sample_metadata.txt'. To be easily read by *OTUbase*, this file is in the form of an `AnnotatedDataFrame`.

1.2 Sequence preprocessing

This section describes the preprocessing steps necessary to generate the files used by *OTUbase*.

1.2.1 Sequence trimming and filtering

Many OTU data projects will begin with raw sequence reads from a next generation pyrosequencer. These reads may include primers, barcodes, and/or adapters that are not part of the actual read. The first step in the analysis pipeline is to trim the primers and barcodes from the read. A number of tools are able to do this. Commands in *Mothur* are 'trim.seqs()' and 'filter.seqs()'. For this workflow we are assuming that these steps have already been done. When the barcodes are trimmed off the reads, a separate file

is generally created that links the read with a sample identification. Mothur creates this file automatically and gives it a '.groups' extension. Each line of this file contains a read ID and the ID of the sample the read belongs to, separated by a tab. OTUbase requires this groups file.

1.3 Taxonomic classification and OTU generation

There are two main approaches used to analyse amplicon data. An OTU approach involves first clustering the sequences together by similarity into OTUs or Operational Taxonomic Units. These OTUs can then be used in richness calculations and in comparing two samples. An alternate approach attempts to classify each sequence into an existing taxonomy. The RDP classifier, for example, uses a Markov model to sort sequences into genus level classifications.

The data produced by these two approaches is slightly different. The OTU approach results in a list of sequences belonging to each OTU. The classification approach results in each sequence having a classification. OTUbase is able to use either of these types of data.

The data processing involved in OTU generation is described by Pat Schloss on the Mothur web site. Those interested can find the OTU generation steps for Sogin's data at .

The data processing involved in the RDP taxonomic classification is somewhat simpler and less computationally demanding. Details can be found on the RDP website .

1.3.1 Reducing the dataset to unique sequences

To decrease the computation time involved in both techniques, duplicate sequences in the dataset are removed first. These sequences can then be added back in after the processing is complete. Mothur removes the duplicate sequences with the command 'unique.seqs()' which also automatically generates a file that keeps track of which sequences have duplicates (called a name file).

In this workflow the duplicate sequences have been removed using Mothur. The name file is called 'sogin.names'

1.4 Importing files into an OTUbase object

OTUbase is able to automatically import a number of files generated during the data processing. These files include the sample file (the group file produced by Mothur), the OTU file (the list file produced by Mothur), and the meta data (in AnnotatedDataFrame format). In addition OTUbase inherits *ShortRead* which allows the user to include a fasta file and a quality file. OTUbase also recognizes the RDP taxonomic classification files that are in the 'fixed' format.

```
> dirPath <- system.file("extdata/Sogin_2006", package="OTUbase")
```

Usually `dirPath` will be the directory path containing the files that will be read by OTUbase.

Because there are two main approaches to data analysis (OTU and Taxonomic classification) we will look at both in parallel. To read in OTU related data the function `readOTUset()` is used. Likewise, to read in classification data the function `readTAXset()` is used.

```
> soginOTU <- readOTUset(dirPath=dirPath, level="0.03", samplefile="sogin.groups")
> soginOTU
```

```
Class: OTUsetF
Number of Sequences: 1000 reads
Sequence Width: 56..100 cycles
Number of OTUs: 399
Number of Samples: 8
sampleData: T          ncol: 6
assignmentData: F
```

The level is the OTU classification level desired (many clustering levels may be present in one otufile). The default is '0.03'. The samplefile connects the read ID to the sample it belongs to. The fastafilename and the associated quality file are optional. Their inclusion may make the reading of the data significantly slower. The otufilename must be in Mothur format. The sampleADF is the sample meta data file.

```
> soginTAX <- readTAXset(dirPath=dirPath, fastafilename='sogin.fasta', sampleADF='sampleADFs')
> soginTAX
```

```
Class: TAXsetF
Number of Sequences: 1000 reads
Sequence Width: 56..100 cycles
Number of Samples: 8
sampleData: T          ncol: 6
assignmentData: F
```

The `readTAXset` function only differs from the `readOTUset` function slightly. Notably different is the absence of an otufilename and the presence of a taxfile (in this case the RDP fixed output). Also included in the `readTAXset` function is the namefile. This file is the Mothur names file and should be included when the dataset has been reduced to unique sequences.

1.5 Accessing data in OTUbase objects

OTUbase provides a number of accessor functions that allow the user to easily access the data contained in the OTUbase object. `sread`, `quality`, and `id` are inherited from the *ShortRead* package and allow access to the sequence, the quality, and the sequence id. In addition, `sampleID`, `sData`, and `aData` provide access to the sample ID, the sample meta data, and the assignment meta data respectively (when available).

```
> head(id(soginOTU))
```

```
BStringSet object of length 6:
```

```
width seq
[1] 14 D4WT9DQ06DVGFR
[2] 14 D4WT9DQ05C6YNI
[3] 14 D4WT9DQ12HNQY2
[4] 14 D4WT9DQ01AP0UQ
[5] 14 D4WT9DQ09FLPTJ
[6] 14 D27LU0R02A82DK
```

```
> head(sread(soginOTU))
```

```
DNASTringSet object of length 6:
```

```
width seq
[1] 60 TGCCTTTGACATCCTCGGAACGGT...GGTGCCTTCGGGAACCGAGAGAC
[2] 71 TGGACTTGACATGTTAGTGTA AAC...AGCTTGCTCAAAGACACTATCAC
[3] 58 CGGGCTTGAAGTGCAAGCGACAAC...GATTTCCGCAAGGACGCTTGTAG
[4] 64 TGGTCTTGACATCCCGGGAATCTC...CCTCATTAGAGGAGCCTGGTGAC
[5] 60 AGGACTTGACATCCAGAGAACTCG...GGTGCCTTCGGGA ACTCTGTGAC
[6] 59 ATCCCTTGACATCCTGCGAACTTT...TGGTGCCTTCGGGAACGCAGTGAC
```

```
> head(sampleID(soginOTU))
```

```
[1] "53R" "53R" "115R" "FS312" "112R" "FS312"
```

```
> head(sData(soginOTU))
```

	Site	Lat_N	Long_W	Depth	Temperature
53R	Labrador seawater	58.3	-29.133	1,400	3.5
55R	Oxygen minimum	58.3	-29.133	500	7.1
112R	Lower deep water	50.4	-25.000	4,121	2.3
115R	Oxygen minimum	50.4	-25.000	550	7.0
137	Labrador seawater	60.9	-38.516	1,710	3.0
138	Labrador seawater	60.9	-38.516	710	3.5

Cells

53R	6.4 × 10 ⁴
55R	1.8 × 10 ⁵
112R	3.9 × 10 ⁴
115R	1.5 × 10 ⁵
137	3.3 × 10 ⁴
138	5.2 × 10 ⁴

There are a couple accessors specific to OTUset or TAXset. To access the OTU IDs stored in OTUset objects, `otuID` is used. Likewise, to access the taxonomic classifications stored in TAXset objects, `tax` is used.

```
> head(otuID(soginOTU))
```

```
[1] "otu221" "otu250" "otu116" "otu385" "otu59" "otu95"
```

```

> head(tax(soginTAX))

  root root_score  domain domain_score  phylum
1 Root          1.0  Bacteria          0.91 Proteobacteria
2 Root          1.0  Bacteria          0.91 Actinobacteria
3 Root          1.0  Bacteria          0.96 Actinobacteria
4 Root          1.0  Bacteria          1.00 Proteobacteria
5 Root          1.0  Bacteria          1.00 Proteobacteria
6 Root          1.0  Bacteria          1.00 Proteobacteria
  phylum_score      class class_score
1          0.62  Gammaproteobacteria    0.46
2          0.10      Actinobacteria    0.10
3          0.16      Actinobacteria    0.16
4          1.00  Deltaproteobacteria    1.00
5          0.98  Gammaproteobacteria    0.97
6          1.00  Gammaproteobacteria    1.00
  order order_score  family
1 Oceanospirillales    0.07  Halomonadaceae
2 Bifidobacteriales    0.04  Bifidobacteriaceae
3 Bifidobacteriales    0.06  Bifidobacteriaceae
4 Desulfobacterales    1.00  Desulfobulbaceae
5 Oceanospirillales    0.54  Oceanospirillaceae
6   Thiotrichales      1.00  Francisellaceae
  family_score      genus genus_score
1          0.06  Modicisalibacter    0.03
2          0.04   Metascardovia     0.04
3          0.06   Parascardovia     0.02
4          1.00   Desulfocapsa     1.00
5          0.51  Oceanospirillum    0.44
6          0.63   Francisella       0.63

```

1.6 First data analysis steps

Now that the data is in the OTUbase object, we can now generate tables and figures that help analyze it. One of the first steps in many analyses is the generation of an abundance table. There is an OTUbase method that does this.

```

> abundOTU <- abundance(soginOTU, weighted=F, collab='Site')
> head(abundOTU)

```

```

      s
o      Lower deep water Oxygen minimum Labrador seawater
otul      0              0              2
otul0     2              0              0
otul100   1              0              0
otul101   0              0              0
otul102   0              0              0

```

```

otu103          0          0          0
  s
o      Labrador seawater Labrador seawater Oxygen minimum
otu1          1          1          0
otu10         0          0          0
otu100        0          0          0
otu101        0          0          0
otu102        1          0          0
otu103        0          0          0
  s
o      Bag City Marker 52
otu1          0          0
otu10         0          0
otu100        1          0
otu101        2          0
otu102        0          0
otu103        1          2

```

```

> abundTAX <- abundance(soginTAX, weighted=F, taxCol='genus', collab='Site')
> head(abundTAX)

```

```

  s
o      Lower deep water Oxygen minimum
Abiotrophia          0          0
Acetivibrio          0          0
Acinetobacter        0          0
Actibacter           0          0
Aestuariicola        0          0
Agromonas            0          0
  s
o      Labrador seawater Labrador seawater
Abiotrophia          0          1
Acetivibrio          0          0
Acinetobacter        0          1
Actibacter           0          0
Aestuariicola        0          0
Agromonas            0          0
  s
o      Labrador seawater Oxygen minimum Bag City
Abiotrophia          0          0          0
Acetivibrio          0          0          1
Acinetobacter        0          0          0
Actibacter           0          0          1
Aestuariicola        0          0          1
Agromonas            1          0          0
  s

```

	Marker 52
Abiotrophia	0
Acetivibrio	0
Acinetobacter	0
Actibacter	3
Aestuariicola	0
Agromonas	4

It should be noted that the abundance method for TAXset objects requires one extra piece of information, the column of the classification desired. The abundance can be generated from any of them (genus, family, etc). Other options are also available in the abundance methods. For example, the abundance can be generated based on any column in the assignment data. For more on the abundance method please see the help documentation.

One of the strengths of OTUbase is that by being in the R environment it can take advantage of a number of available data analysis packages. One of these packages is *vegan*. *vegan* is an R package that provides many tools to analyze ecological type data. It includes diversity estimation and cluster analysis.

Using the functions provided by *vegan* and the abundance table previously generated:

```
> estrichOTU <- apply(abundOTU, 2, estimateR)
> estrichOTU
```

```

s
  Lower deep water Oxygen minimum
S.obs      57.000000      44.000000
S.chao1    192.125000     176.000000
se.chao1    60.975679      74.630319
S.ACE      283.067602     154.791782
se.ACE       7.841178       8.026768

s
  Labrador seawater Labrador seawater
S.obs      49.000000      48.000000
S.chao1    254.000000     159.42857
se.chao1    110.794024      53.88188
S.ACE      230.006548     320.32000
se.ACE       5.532269      11.21429

s
  Labrador seawater Oxygen minimum Bag City
S.obs      37.000000      29.000000 110.00000
S.chao1    145.750000     191.500000 384.61538
se.chao1    62.860021     104.644533  96.01630
S.ACE      147.625000     182.685606 579.22305
se.ACE       3.619804       2.553564 16.82521

s
  Marker 52
```

```

S.obs      130.00000
S.chao1    308.00000
se.chao1   54.70480
S.ACE      392.65701
se.ACE     12.75554

> estrichTAX <- apply(abundTAX, 2, estimateR)
> estrichTAX

```

```

s
  Lower deep water Oxygen minimum
S.obs      48.00000      34.000000
S.chao1    118.00000     64.000000
se.chao1   33.36542     17.882622
S.ACE      202.70270     77.238156
se.ACE     10.26136      5.658764

```

```

s
  Labrador seawater Labrador seawater
S.obs      41.00000      35.000000
S.chao1    173.00000    122.750000
se.chao1   74.62941     52.070518
S.ACE      167.54231    132.763430
se.ACE     4.56640      5.203785

```

```

s
  Labrador seawater Oxygen minimum   Bag City
S.obs      29.000000     22.000000   91.000000
S.chao1    67.000000    107.500000  253.750000
se.chao1   23.913688     59.310721   62.402381
S.ACE      98.141354    194.379259  290.053022
se.ACE     7.521336      9.549327   9.670711

```

```

s
  Marker 52
S.obs      88.000000
S.chao1    157.789474
se.chao1   26.353821
S.ACE      190.277732
se.ACE     8.276204

```

The vegan function `vegdist` and `hclust` have been combined into one `OTUbase` wrapper for convenience. This allows the user to quickly cluster the samples. This clustering can be done using a number of different distance and clustering methods.

```
> clusterSamples(soginOTU, distmethod='jaccard', clustermethod='complete', colla
```

```
Call:
hclust(d = d, method = clustermethod)
```



```
Cluster method   : complete
Distance         : jaccard
Number of objects: 8
```

```
> clusterSamples(soginTAX, taxCol='genus', distmethod='jaccard', clustermethod='
```

```
Call:
hclust(d = d, method = clustermethod)
```

```
Cluster method   : complete
Distance         : jaccard
Number of objects: 8
```

The user is encouraged to explore many functions available through *vegan* and other R packages. Commonly useful ones can then be brought into *OTUbase* to make their use more efficient.

2 Advanced features

A number of other functions are available. While the implementation is incomplete, `subOTUset()` is a function that allows the user to extract any OTUs or samples from the dataset to be analyzed separately. This makes it possible to remove one or more OTUs or samples from the analysis. Eventually this will be implemented using the more traditional '[' notation.

```
> soginReduced <- subOTUset(soginOTU, samples=c("137", "138", "53R", "55R"))
> soginReduced
```

```
Class: OTUsetF
Number of Sequences: 222 reads
Sequence Width: 56..100 cycles
Number of OTUs: 127
Number of Samples: 4
sampleData: T      ncol: 6
assignmentData: F
```

3 The structure of an OTUbase object

OTUbase objects include a number of possible slots. Inherited from *ShortRead* are `sread`, `id`, and `quality`. These slots, along with `otuID`, `tax`, and `sampleID` are all of identical length and order. For example, the first row in the `id` slot is connected to the first rows in the `sread`, `quality`, `otuID`, and `sampleID` slots. In other words, the first `id` represents the first sequence that has a quality described by the first row of the `quality` slot; it is a member of the `otu` listed in the `otuID` slot and a member of the `sample` listed in the first row of the `sample` slot.

In addition there are two AnnotatedDataFrames. The sampleData data frame is linked to the sampleID slot through the sample IDs. The assignmentData data frame is linked to the otuID slot through the OTU IDs.

There are slight differences in the OTUset objects and the TAXset objects. In the TAXset objects, the assignmentData data frame is not explicitly linked to the tax slot.

4 Conclusions and directions for development

OTUbase provides an organization and structure for OTU data and taxonomic classification data produced during the analysis of amplicon sequences. This allows the user to quickly and easily analyze amplicon data.

While the structure and a few basic functions are available withing OTUbase, there are a large number of possible improvements and extensions that have yet to be developed. OTUbase provides a structure for the data but functions for downstream analysis are not yet included. Future development will include a better integration of OTUbase with other available R packages such as vegan and the inclusion of a wider variety of functions for data analysis.

5 References

Sogin, M., H. Morrison, J. Huber, D. Welch, S. Huse, P. Neal, J. Arrieta, and G. Herndl. 2006. Microbial diversity in the deep sea and the underexplored "rare biosphere." *Proc. Natl. Acad. Sci. U. S. A.* 103:12115-12120

Schloss PD, et al. (2009) Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol* 75:7537-7541

Wang, Q, G. M. Garrity, J. M. Tiedje, and J. R. Cole. 2007. Naïve Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy. *Appl Environ Microbiol.* 73(16):5261-7

```
> toLatex(sessionInfo())
```

- **R Under development (unstable) (2023-10-22 r85388),**
x86_64-pc-linux-gnu
- **Locale:** LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C,
LC_TIME=en_US.UTF-8, LC_COLLATE=en_US.UTF-8,
LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8,
LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C,
LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8,
LC_IDENTIFICATION=C
- **Time zone:** America/New_York
- **TZcode source:** system (glibc)
- **Running under:** Ubuntu 22.04.3 LTS
- **Matrix products:** default
- **BLAS:** /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
- **LAPACK:**
/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
- **Base packages:** base, datasets, graphics, grDevices, methods, stats, stats4, utils
- **Other packages:** Biobase 2.63.0, BiocGenerics 0.49.0, BiocParallel 1.37.0,
Biostrings 2.71.0, GenomeInfoDb 1.39.0, GenomicAlignments 1.39.0,
GenomicRanges 1.55.0, IRanges 2.37.0, lattice 0.22-5, MatrixGenerics 1.15.0,
matrixStats 1.0.0, OTUbase 1.53.0, permute 0.9-7, Rsamtools 2.19.0,
S4Vectors 0.41.0, ShortRead 1.61.0, SummarizedExperiment 1.33.0,
vegan 2.6-4, XVector 0.43.0
- **Loaded via a namespace (and not attached):** abind 1.4-5, bitops 1.0-7,
cluster 2.1.4, codetools 0.2-19, compiler 4.4.0, crayon 1.5.2,
DelayedArray 0.29.0, deldir 1.0-9, GenomeInfoDbData 1.2.11, grid 4.4.0,
hwriter 1.3.2.1, interp 1.1-4, jpeg 0.1-10, latticeExtra 0.6-30, MASS 7.3-60.1,
Matrix 1.6-1.1, mgcv 1.9-0, nlme 3.1-163, parallel 4.4.0, png 0.1-8,
RColorBrewer 1.1-3, Rcpp 1.0.11, RCurl 1.98-1.12, S4Arrays 1.3.0,
SparseArray 1.3.0, splines 4.4.0, tools 4.4.0, zlibbioc 1.49.0

Table 1: The output of `sessionInfo` on the build system after running this vignette.