Example Workflows for the Package RDPutils

John Quensen

June 29, 2014

Introduction

The package provides means to construct phyloseq objects from the output of RDP tools for clustering and classfying DNA sequences from high-throughput amplicon sequencing projects.

The Ribosomal Database Project (RDP) provides both web-based and command line tools for processing rRNA gene sequences from Bacteria, Archae, and Fungi as well as functional genes. Web-based tools and tutorials for using them are available at http://rdp.cme.msu.edu/index.jsp. The command line tools are available at https://github.com/rdpstaff/RDPTools. Processing can take either of two approaches. In the "supervised" approach, sequences from multiple samples are binned by classifying them using a database for Bacteria or Archae or Fungi, or a user's own database, and further processing is usually done in a spreadsheet program. In the "unsupervised" approach, sequences are clustered into OTUs based on their degree of similarity. RDP provides additional tools to parse the cluster files into OTU tables that can be imported into R, and to retrieve representative sequences for each cluster.

Phyloseq is an R/Bioconductor package that includes a variety of wrappers for quick exploratory data analysis of sequencing data, but perhaps its most convenient feature is that it provides a means of rapid and flexible sub-setting of data from a large experiment. A phyloseq object has slots for an OTU table, a classification table, a sample data table, and a tree of the sequences representing each OTU. I recommend phyloseq because it organizes all data for an experiment and R because it provides a flexible means of analyzing that data.

The functions in this package reformat RDP output so that it is possible to fill phyloseq slots. In the case of the supervised approach, the RDP classifier result is reformatted directly into a phyloseq object containing an OTU table and a classification table. For the unsupervised approach, the OTU table can be the result from the RDP R-formatter tool, or from a function in this package. The key to filling the classification table and tree slots is to rename the representative sequences to correspond to the OTU names; this package includes functions to do this. Once renamed, the representative sequences can be classified and treed, and the results used to fill phyloseq classification table and tree slots. For either approach, a sample data table is most easily constructed in a spreadsheet program.

Below are example workflows for both supervised and unsupervised approaches.

A Note:

Input to some of the functions in this package are text files read from disk, normally in practice from your R working directory. But for the examples to work, it is necessary to use the system.file command to get the path plus file name in the R installation. This is assigned to a variable, and that variable is used as a

function argument in place of what a user would normally enter enclosed in parentheses. For example, while the following works as an example:

```
> clstr.file <- system.file("extdata", "all_seq_complete.clust",
+                           package="RDPutils")
> otu <- clstr2otu(clstr_file=clstr.file, dist=0.03, OutFile=TRUE,
+                  file.name= "dist_03.clust")
```

a user would normally type

```
> otu <- clstr2otu(clstr_file = "all_seq_complete.clust", dist = 0.03,
+                  OutFile = FALSE, file.name = "dist_03.clust")
```

Workflow for Cluster Results

RDP's unsupervised method involves aligning rRNA sequences using the structure-aware Infernal aligner, calculating distances between sequences, and then clustering them, by default, by the complete linkage method. The result is a large "clust" file giving the numbers and names of every sequence in every cluster for every sample for several distances, from 0 to 0.15 by default. The R-formatter tool parses the cluster file into tab-delimited OTU tables for each distance with samples as rows and OTUs as columns. These can be easily imported into R with the base function `read.table`.

Cluster files may also be parsed into an OTU table for a specific distance with `clstr2otu`. This function also gives the option of excising the portion of the larger cluster file that corresponds to the same specific distance, as in this example:

```
> library(RDPutils)
> clstr.file <- system.file("extdata", "all_seq_complete.clust",
+                           package="RDPutils")
> otu <- clstr2otu(clstr_file=clstr.file, dist=0.03,
+                  OutFile=TRUE, file.name= "dist_03.clust")
> otu[ , 1:5]
```

```
         OTU_000 OTU_001 OTU_002 OTU_003 OTU_004
Sample_1       1       3       1       3       1
Sample_2       1       3       4       1       0
Sample_3       1       4       2       1       1
Sample_4       0       5       1       1       2
```

Here, `clstr.file` is the name of the cluster file, retrieved from the examples provided for RDPutils. Again, normally, the user would enter the cluster file name `all_seq_complete.clust` within quotation marks in its place. Since `OutFile` is `TRUE`, a cluster file for the distance 0.03 is written to the file `dist_03.clust`. The result, `otu`, is of course the OTU table. The sample names have been simplified by removing common prefixes and suffixes introduced by the RDP pipeline.

Compare this to importing the text file from the R-formatter:

```
> rformatfile <- system.file("extdata", "rformat_dist_0.03.txt",
+                            package="RDPutils")
> otu.rdp <- read.table(rformatfile, header=TRUE, row.names=1,
+                       sep="\t")
> otu.rdp[ , 1:5]
```

```
         OTU_000 OTU_001 OTU_002 OTU_003 OTU_004
Sample_1       1       3       1       3       1
Sample_4       0       5       1       1       2
Sample_3       1       4       2       1       1
Sample_2       1       3       4       1       0
```

The result is the same, except for the order of the samples, which of course can be sorted:

```
> otu.rdp <- otu.rdp[sort(rownames(otu.rdp)), ]
> otu.rdp[ , 1:5]
```

```
         OTU_000 OTU_001 OTU_002 OTU_003 OTU_004
Sample_1       1       3       1       3       1
Sample_2       1       3       4       1       0
Sample_3       1       4       2       1       1
Sample_4       0       5       1       1       2
```

In order to assign taxonomy to the OTUs, it is necessary to rename the representative sequences with the OTU names. After mid-2013, the tools to retrieve representative sequences include the cluster (or OTU) number in the fasta IDs. If retrieved with the on-line tool, they have the form:
`>HC9D00P01BCTC4 cluster_id=1,size=2`
If retrieved with the command line tool in RDPTools, they have the form:
`>HC9D00P01BCTC4 prefered=false,cluster=0,clustsize=2`
In these cases, it is possible to make an association between the representative sequence IDs and the OTU names with the function `assoc_repseq_IDs_with_otus_by_fasta`:

```
> repseq.file <- system.file("extdata",
+                 "all_seq_complete.clust_rep_seqs.fasta",
+                  package="RDPutils")
> assoc.table <- assoc_repseq_IDs_with_otus_by_fasta(repseq_file =
+                                                    repseq.file)
> head(assoc.table)
```

```
       RepSeq.ID     OTU Cluster # Cluster size
1 HC9D00P01AW8TJ OTU_000         0            3
2 HC9D00P01AXCPC OTU_001         1           15
3 HC9D00P01AUPHY OTU_002         2            8
4 HC9D00P01A480B OTU_003         3            6
5 HC9D00P01AV8BT OTU_004         4            4
6 HC9D00P01A6EHN OTU_005         5            4
```

If the fasta IDs of the representative sequences do not meet the conditions above, the association between OTUs and representative sequences can be obtained from the cluster file. This method is more involved and

takes longer to run. First it is necessary to get a list of the machine names of the representative sequences and use that list as input to the function `assoc_repseq_IDs_with_otus_by_clstr`:

```
> repseq.file <- system.file("extdata",
+                            "all_seq_complete.clust_rep_seqs.fasta",
+                            package="RDPutils")
> rep.seqs <- get_repseq_IDs_from_fasta(repseq_file = repseq.file)
> clstr <- system.file("extdata", "dist_03.clust", package="RDPutils")
> assoc.table <- assoc_repseq_IDs_with_otus_by_clstr(clstr_file=clstr,
+                                                    rep_seqs=rep.seqs)
> head(assoc.table)
```

```
       RepSeq.ID     OTU Cluster #           Sample Sequences in Sample
1 HC9D00P01AW8TJ OTU_000         0 aligned_Sample_1                   1
2 HC9D00P01AXCPC OTU_001         1 aligned_Sample_2                   3
3 HC9D00P01AUPHY OTU_002         2 aligned_Sample_3                   2
4 HC9D00P01A480B OTU_003         3 aligned_Sample_1                   3
5 HC9D00P01AV8BT OTU_004         4 aligned_Sample_4                   2
6 HC9D00P01A6EHN OTU_005         5 aligned_Sample_1                   2
```

Once the `assoc.table` is made, it can be used to rename the representative sequences. But first, it is necessary to trim the IDs of the representative sequences to contain only the machine names.

```
> repseq.file <- system.file("extdata",
+                            "all_seq_complete.clust_rep_seqs.fasta",
+                            package="RDPutils")
> trim_fasta_names(repseq_file = repseq.file,
+                  trimmed_names = "names_trimmed.fasta", strip = FALSE)
```

```
[1] "Fasta file with trimmed names written to file names_trimmed.fasta."
```

```
> rename_fasta(in_file = "names_trimmed.fasta", out_file = "renamed.fasta",
+              rename_table = assoc.table)
```

```
[1] "Fasta sequences renamed with OTU names; written to file renamed.fasta."
```

After renaming, the representative sequences can be classified and the detailed fixrank format from the classifier converted into a phyloseq `tax_table` with the function `make_tax_table`.

```
> my.in.file <- system.file("extdata", "fixrank_classified.txt", package="RDPutils")
> my.tax.table <- make_tax_table(in_file = my.in.file, confidence=0.5)
> head(my.tax.table)
```

```
Taxonomy Table:     [6 taxa by 6 taxonomic ranks]:
        domain     phylum                  class
OTU_000 "Bacteria" "Proteobacteria"        "Alphaproteobacteria"
OTU_001 "Bacteria" "unclassified_Bacteria" "unclassified_Bacteria"
```

```
OTU_002 "Bacteria" "Proteobacteria"        "Alphaproteobacteria"
OTU_003 "Bacteria" "unclassified_Bacteria" "unclassified_Bacteria"
OTU_004 "Bacteria" "Proteobacteria"        "Alphaproteobacteria"
OTU_005 "Bacteria" "Proteobacteria"        "Alphaproteobacteria"
        order                   family
OTU_000 "Rhizobiales"           "unclassified_Rhizobiales"
OTU_001 "unclassified_Bacteria" "unclassified_Bacteria"
OTU_002 "Rhizobiales"           "unclassified_Rhizobiales"
OTU_003 "unclassified_Bacteria" "unclassified_Bacteria"
OTU_004 "Rhizobiales"           "Hyphomicrobiaceae"
OTU_005 "Rhizobiales"           "unclassified_Rhizobiales"
        genus
OTU_000 "unclassified_Rhizobiales"
OTU_001 "unclassified_Bacteria"
OTU_002 "unclassified_Rhizobiales"
OTU_003 "unclassified_Bacteria"
OTU_004 "Rhodoplanes"
OTU_005 "unclassified_Rhizobiales"
```

The renamed representative sequences can also be treed, with FastTree for example, and the result included in a phyloseq object. Once this is done, a phyloseq object can be created. In the example below, sam.data is a simple dataframe including information about each sample, and `rep.tree.nwk` is a tree of the renamed representative sequences in Newick format. Nexus format is also acceptable. The `read_tree` function converts either to a phylo class object.

```
> data(sam.data)
> sam.data
```

```
         Treatment Replicate Variable_A Variable_B
Sample_1         A         1       0.54       0.72
Sample_2         A         2       0.62       0.65
Sample_3         B         1       0.23       0.11
Sample_4         B         2       0.20       0.12
```

```
> rep.tree <- system.file("extdata", "rep.tree.nwk", package="RDPutils")
> my.tree <- read_tree(rep.tree)
> my.otu <- otu_table(as.matrix(t(otu)), taxa_are_rows=TRUE, errorIfNULL=TRUE)
> my.data <- sample_data(sam.data)
> my.expt <- phyloseq(my.otu, my.tax.table, my.data, my.tree)
> my.expt
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 443 taxa and 4 samples ]
sample_data() Sample Data:       [ 4 samples by 4 sample variables ]
tax_table()   Taxonomy Table:    [ 443 taxa by 6 taxonomic ranks ]
phy_tree()    Phylogenetic Tree: [ 443 tips and 441 internal nodes ]
```

Workflow for Classifier Results

The RDP classifier can be used for multiple samples and reports results in two different formats – a hierarchical output and a sequence–by–sequence or detailed output. When using the command line classifier, the sequence–by–sequence output is given by the `-o` parameter (required) and the hierarchical output is given by the `-h` parameter. The function `hier2phyloseq` converts hierarchical output for multiple samples to a phyloseq object containing an OTU table and a taxonomy table. It does not work for a single sample. It is best used with the output from the command line classifier because then the results can be binned according to confidence level by setting the format parameter to `filterbyconf`. For example, if sequences are identified to order level but not to family, they are assigned to family 'unclass_order,' etc. An appropriate classifier command is:

```
java -Xmx1g -jar C:/rdp_classifier_2.7/dist/classifier.jar classify -g fungallsu -c 0.5
-f filterbyconf -o test_classified.txt -h test_hier.txt Sample_1.fasta Sample_2.fasta
```

issued from the directory containing the fasta files.

As an example:

```
> hier.file <- system.file("extdata", "test_hier.txt", package="hier")
> expt <- hier2phyloseq(hier_file=hier.file)
> expt
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 717 taxa and 18 samples ]
tax_table()   Taxonomy Table:    [ 717 taxa by 7 taxonomic ranks ]
```

This example is for 28S rRNA gene sequences and the aim is to analyze fungal communities. Domains other than fungi have been included in the data base to improve classification. The domains represented in the data can be determined thus:

```
> rank_names(expt)
```

```
[1] "rootrank" "domain"   "phylum"   "class"    "order"    "family"   "genus"
```

```
> unique(tax_table(expt)[ ,"domain"])
```

```
Taxonomy Table:     [7 taxa by 1 taxonomic ranks]:
         domain
ID_11409 "Fungi"
ID_12567 "Animalia"
ID_12619 "Eukaryota incertae sedis"
ID_12574 "Viridiplantae"
ID_12603 "Alveolata"
ID_-1    "unclass_Root"
ID_12629 "Amoebozoa"
```

The data can be subset to include only the fungal sequences thus:

```
> fungi <- subset_taxa(expt, domain=="Fungi")
> fungi
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 685 taxa and 18 samples ]
tax_table()   Taxonomy Table:    [ 685 taxa by 7 taxonomic ranks ]
```

and the percentage of non-fungal sequences is determined as:

```
> expt.sum <- sum(otu_table(expt))
> fungi.sum <- sum(otu_table(fungi))
> 100*((expt.sum-fungi.sum)/expt.sum)
```

```
[1] 19.21827
```