PhyloToAST Documentation

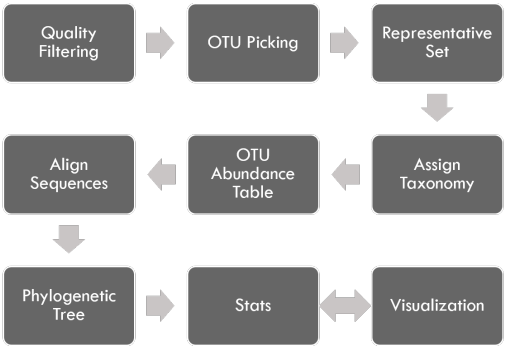
Release 1.3.0

Shareef Dabdoub

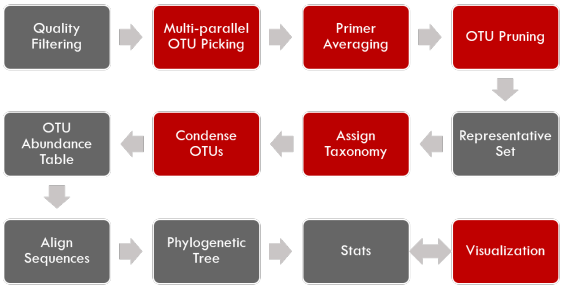
March 09, 2016

Contents

The PhyloToAST project is a collection of python scripts that modifies the original QIIME [1] pipeline:



by adding or modifying several steps (as seen below) including support for PBS-based cluster-computing, multiple primer support [2], enhanced support for species-specific analysis, and additional visualization tools.



Download and install PhyloToAST package.

[1] **QIIME allows analysis of high-throughput community sequencing data**. J Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D Bushman, Elizabeth K Costello, Noah Fierer, Antonio Gonzalez Pena, Julia K Goodrich, Jeffrey I Gordon, Gavin A Huttley, Scott T Kelley, Dan Knights, Jeremy E Koenig, Ruth E Ley, Catherine A Lozupone, Daniel McDonald, Brian D Muegge, Meg Pirrung, Jens Reeder, Joel R Sevinsky, Peter J Turnbaugh, William A Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld and Rob Knight; Nature Methods, 2010; doi: 10.1038/nmeth.f.303

[2] **Target Region Selection Is a Critical Determinant of Community Fingerprints Generated by 16S Pyrosequencing**. Kumar PS, Brooker MR, Dowd SE, Camerlengo T (2011) Target Region Selection Is a Critical Determinant of Community Fingerprints Generated by 16S Pyrosequencing. PLoS ONE 6(6): e20956. doi: 10.1371/journal.pone.0020956

CHAPTER 1

PhyloToAST Scripts

1.1 API

API scripts of PhyloToAST.

1.1.1 biom_calc module

This module provides methods for calculating various metrics with regards to each OTU in an input OTU abundance table.

arcsine_sqrt_transform

Takes the proportion data from relative_abundance() and applies the variance stabilizing arcsine square root transformation:

$$X = sin^{-1}(\sqrt(p))$$

```
usage: phylotoast.biom_calc.arcsine_sqrt_transform(rel_abd)
```

rel_abd:
Refers to a dictionary keyed on SampleIDs, and the values are dictionaries keyed on OTUID's and their values represent the relative abundance of that OTUID in that SampleID. rel_abd is the output of relative_abundance() function.

return:
Returns a dictionary keyed on SampleIDs, and the values are dictionaries keyed on OTUID's and their values represent the transformed relative abundance of that OTUID in that SampleID.

---

mean_otu_pct_abundance

Calculate the mean OTU abundance percentage.

```
usage: phylotoast.biom_calc.mean_otu_pct_abundance(rel_abd, otuIDs)
```

rel_abd:
Refers to a dictionary keyed on SampleIDs, and the values are dictionaries keyed on OTUID's and their values represent the relative abundance of that OTUID in that SampleID. rel_abd is the output of relative_abundance() function.

otuIDs:
A list of OTUID's for which the percentage abundance needs to be measured.

return:
A dictionary of OTUID and their percent relative abundance as key/value pair.

MRA

Calculate the mean relative abundance.

```
usage: phylotoast.biom_calc.MRA(biomf)
```

biomf:
A BIOM file.

return:
A dictionary keyed on OTUID's and their mean relative abundance for a given number of sampleIDs.

---

raw_abundance

Calculate the total number of sequences in each OTU or SampleID.

```
usage: phylotoast.biom_calc.raw_abundance(biomf, sampleIDs=None, sample_abd=True)
```

biomf:
A BIOM file.

sampleIDs:
A list of column id's from BIOM format OTU table. By default, the list has been set to None.

sample_abd:
A boolean operator to provide output for OTUID's or SampleID's. By default, the output will be provided for SampleID's.

return:
Returns a dictionary keyed on either OTUID's or SampleIDs and their respective abundance as values.

---

relative_abundance

Calculate the relative abundance of each OTUID in a Sample.

```
usage: phylotoast.biom_calc.relative_abundance(biomf)
```

biomf:
A BIOM format.

return:
Returns a dictionary keyed on SampleIDs, and the values are dictionaries keyed on OTUID's and their values represent the relative abundance of that OTUID in that SampleID.

---

transform_raw_abundance

Function to transform the total abundance calculation for each sample ID to another format based on user given transformation function.

```
usage: phylotoast.biom_calc.transform_raw_abundance(biomf, fn=math.log10, sampleIDs=None, sample_abd=
```

biomf:
A BIOM file.

fn:
Mathematical function which is used to transform smax to another format. By default, the function has been given as base 10 logarithm.

sampleIDs:
A list of column id's from BIOM format OTU table. By default, the list has been set to None.

sample_abd:
A boolean operator to provide output for OTUID's or SampleID's. By default, the output will be provided for SampleID's.

return:
Returns a dictionary similar to output of raw_abundance function but with the abundance values modified by the mathematical operation. By default, the operation performed on the abundances is base 10 logarithm.

1.1.2 otu_calc module

assign_otu_membership

Determines the number of OTUs associated with samples using fuzzy sets with membership amount determined by relative abundance.

```
usage: phylotoast.otu_calc.assign_otu_membership(biom)
```

biomf:
BIOM format file.

return:
Returns a dictionary of FuzzySet of SampleID's with OTUID and relative abundance as its elements.

---

fuzzy_lookup

Return the intersection of a fuzzy set and a collection of keys (presumably a subset).

```
usage: phylotoast.otu_calc.fuzzy_lookup(orig, keys)
```

orig:
FuzzySet of SampleID with OTUID and relative abundances.

keys:
Genus-species taxonomic identifier.

return:
Returns a new FuzzySet of genus-species identifier and relative abundance for the given list of keys.

---

load_core_file

For core OTU data file, returns Genus-species identifier for each data entry.

```
usage: phylotoast.otu_calc.load_core_file(core_fp)
```

core_fp:
A file containing core OTU data.

return:
Returns genus-species identifier based on identified taxonomical level.

---

otu_name_biom

Given an OTU row from a BIOM table, determine a Genus-species identifier from the taxonomic specifier (see otu_name() method).

```
usage: phylotoast.otu_calc.otu_name_biom(biom_row)
```

biom_row:
Row entry of a BIOM file containing full taxonomy.

return:
Returns the genus-species identifier.

---

otu_name

Determine a simple Genus-species identifier for an OTU, if possible. If OTU is not identified to the species level, name it as Unclassified (familly/genus/etc...).

```
usage: phylotoast.otu_calc.otu_name(tax)
```

tax:
QIIME-style taxonomy identifiers, e.g. ['k__Bacteria', u'p__Firmicutes', u'c__Bacilli', ...]

return:
Returns genus-species identifier based on identified taxonomical level.

---

print_membership

Given an entry from a Sample dictionary (see assign_otu_membership) of fuzzy otu membership, pretty-print the members as percentages.

```
usage: phylotoast.otu_calc.print_membership(entry)
```

entry:
SampleID's from the output dict of assign_otu_membership().

return:
Returns OTU name and percentage relative abundance as membership for the given list of SampleID's.

---

sdi

Calculate the Shannon Diversity Index.

$$H = -sum(p * ln(p))$$

where p is the relative abundance of a single OTU in the set.

```
usage: phylotoast.otu_calc.sdi(fset)
```

fset:
The set of OTUs and their relative abundance values.

return:
The Shannon Diversity Index.

Note
Equitability Index ($E_H = H/H_{max}$) could be easily calculated from the returned array by:

$$diversities = sdi(fset)$$
$$equitabilities = diversities/max(diversities)$$

1.1.3 util module

ensure_dir

Check to make sure the supplied directory path does not exist, if so, create it.

```
usage: phylotoast.util.ensure_dir(d)
```

d:
It is the full path to a directory.

return:
Does not return anything, but creates a directory path if it doesn't exist already.

---

file_handle

Takes either a file path or an open file handle, checks validity and returns an open file handle or raises an appropriate Exception.

```
usage: phylotoast.util.file_handle(fnh, mode='rU')
```

fnh:
It is the full path to a file, or open file handle.

mode:
The way in which this file will be used, for example to read or write or both. By default, file will be opened in rU mode.

return:
Returns an opened file for appropriate usage.

---

gather_categories

Find the user specified categories in the map and create a dictionary to contain the relevant data for each type within the categories. Multiple categories will have their types combined such that each possible combination will have its own entry in the dictionary.

```
usage: phylotoast.util.gather_categories(imap, header, categories=None)
```

imap:
The input mapping file data keyed by SampleID.

header:
The header line from the input mapping file. This will be searched for the user-specified categories.

categories:
The list of user-specified categories from the mapping file.

return:
A sorted dictionary keyed on the combinations of all the types found within the user-specified categories. Each entry will contain an empty DataCategory namedtuple. If no categories are specified, a single entry with the key 'default' will be returned.

---

parseFASTA

Parse the records in a FASTA-format file by first reading the entire file into memory.

```
usage: phylotoast.util.parseFASTA(fastaFNH)
```

fastaFNH:
The data source from which to parse the FASTA records. Expects the input to resolve to a collection that can be iterated through, such as a list or an open file handle.

return:
FASTA records containing entries for id, description and data.

---

parse_map_file

Opens a QIIME mapping file and stores the contents in a dictionary keyed on SampleID (default) or a user-supplied one. The only required fields are SampleID, BarcodeSequence, LinkerPrimerSequence (in that order), and Description (which must be the final field).

```
usage: phylotoast.util.parse_map_file(mapFNH)
```

mapFNH:
Either the full path to the map file or an open file handle.

return:
A tuple of header line for mapping file and a map associating each line of the mapping file with the appropriate sample ID (each value of the map also contains the sample ID). An OrderedDict is used for mapping so the returned map is guaranteed to have the same order as the input file.

---

parse_taxonomy_table

Greengenes provides a file each OTU a full taxonomic designation. This method parses that file into a map with (key,val) = (OTU, taxonomy).

```
usage: phylotoast.util.parse_taxonomy_table(idtaxFNH)
```

idtaxFNH:
Either the full path to the map file or an open file handle.

return:
A map associating each OTU ID with the taxonomic specifier. An OrderedDict is used so the returned map is guaranteed to have the same order as the input file.

---

parse_unifrac

Parses the unifrac results file into a dictionary.

```
usage: phylotoast.util.parse_unifrac(unifracFN)
```

unifracFN:
The path to the unifrac results file.

return:
A dictionary with keys: 'pcd' (principle coordinates data) which is a dictionary of the data keyed by sample ID, 'eigvals' (eigenvalues), and 'varexp' (variation explained).

---

parse_unifrac_v1_8

Function to parse data from older version of unifrac file obtained from Qiime version 1.8 and earlier.

```
usage: phylotoast.util.parse_unifrac_v1_8(unifrac, file_data)
```

unifrac:
The path to the unifrac results file.

file_data
Unifrac data lines after stripping whitespace characters.

return:
A dictionary with keys: 'pcd' (principle coordinates data) which is a dictionary of the data keyed by sample ID, 'eigvals' (eigenvalues), and 'varexp' (variation explained).

---

parse_unifrac_v1_9

Function to parse data from newer version of unifrac file obtained from Qiime version 1.9 and later.

```
usage: phylotoast.util.parse_unifrac_v1_9(unifrac, file_data)
```

unifrac:
The path to the unifrac results file.

file_data
Unifrac data lines after stripping whitespace characters.

return:
A dictionary with keys: 'pcd' (principle coordinates data) which is a dictionary of the data keyed by sample ID, 'eigvals' (eigenvalues), and 'varexp' (variation explained).

---

split_phylogeny

Return either the full or truncated version of a QIIME-formatted taxonomy string.

```
usage: phylotoast.util.split_phylogeny(p, level='s')
```

p:
A QIIME-formatted taxonomy string: k__Foo; p__Bar; ...

level:
The different level of identification are kingdom (k), phylum (p), class (c),order (o), family (f), genus (g) and species (s). The default level of identification is species.

return:
A QIIME-formatted taxonomy string up to the classification given by param level.

---

storeFASTA

Parse the records in a FASTA-format file by first reading the entire file into memory.

```
usage: phylotoast.util.storeFASTA(fastaFNH)
```

fastaFNH:
The data source from which to parse the FASTA records. Expects the input to resolve to a collection that can be iterated through, such as a list or an open file handle.

return:
FASTA records containing entries for id, description and data.

---

write_map_file

Given a list of mapping items (in the form described by the parse_mapping_file method) and a header line, write each row to the given input file with fields separated by tabs.

```
usage: phylotoast.util.write_map_file(mapFNH, items, header)
```

mapFNH:
Either the full path to the map file or an open file handle.

items:
The list of row entries to be written to the mapping file.

header:
The descriptive column names that are required as the first line of the mapping file.

return:
None.

1.1.4 graph_util

plot_kde

Plot a smoothed (by kernel density estimate) histogram.

```
usage: phylotoast.graph_util.plot_kde(data, ax, title=None, color='r', fill_bt=True)
```

data:
An array containing the data to be plotted.

ax:
The Axes object to draw to.

title:
The plot title.

color:
The color of the histogram line and fill. Note that the fill will be plotted with an alpha of 0.35.

fill_bt:
Specify whether to fill the area beneath the histogram line.

1.2 Other Scripts

Various PhyloToAST scripts.

1.2.1 assign_taxonomy_by_blast _result.py

Assign taxonomy to a rep set of OTUs that were chosen by BLAST from an annotated database.

```
    usage: assign_taxonomy_by_blast_result.py [-h] -i REP_SET_FP -t ID_TO_TAXONOMY_FP [-o ASSIGNED_T
```

Required arguments

-i REP_SET_FP, **--rep_set_fp** REP_SET_FP
The set of representative sequences.

-t ID_TO_TAXONOMY_FP, **--id_to_taxonomy_fp** ID_TO_TAXONOMY_FP
Path to tab-delimited file mapping sequences to assigned taxonomy.

Optional arguments

-o ASSIGNED_TAXONOMY_FP, **--assigned_taxonomy_fp** ASSIGNED_TAXONOMY_FP
The path to the result file. By default outputs to assigned_taxonomy.txt

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.2 barcode_filter.py

From an input FASTA file, filter all sequences with barcodes matching those in an input mapping file.

```
usage: barcode_filter.py [-h] -i INPUT_FASTA_FN -m MAPPING_FN [-q QUALITY_FN] [-o OUTPUT_PREFIX]
```

Required arguments

-i INPUT_FASTA_FN, **--input_fasta_fn** INPUT_FASTA_FN
The sequence data file to be filtered.

-m MAPPING_FN, **--mapping_fn** MAPPING_FN
The mapping file containing the barcodes you want filtered sequenced to contain.

Optional arguments

-q QUALITY_FN, **--quality_fn** QUALITY_FN
The quality data file. If you plan to use quality data with split_libraries.py, you have to filter the quality data as well.

-o OUTPUT_PREFIX, **--output_prefix** OUTPUT_PREFIX
The prefix for the output filtered data

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.3 biom_relative_abundance.py

Convert a BIOM file of OTU abundance data into a CSV of relative abundance data.

```
usage: biom_relative_abundance.py [-h] [-i INPUT_BIOM_FP] [-o OUTPUT_TSV_FP] [--stabilize_varian
```

Required arguments

-i INPUT_BIOM_FP, **--input_biom_fp** INPUT_BIOM_FP
The BIOM file path.

Optional arguments

-o OUTPUT_CSV_FP, **--output_csv_fp** OUTPUT_CSV_FP
A CSV table of relative OTU abundance data.

--stabilize_variance
Apply the variance-stabilizing arcsine square root transformation to the OTU proportion data.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.4 condense_workflow.py

This workflow script will run all three steps of the OTU condensing pipeline automatically with the default output file settings.

```
usage: condense_workflow.py [-h] -i ASSIGNED_TAXONOMY_FN -r REP_SET_FN -s SEQS_OTUS_FN [-L {k,p,
```

Required arguments

-i ASSIGNED_TAXONOMY_FN, **--assigned_taxonomy_fn** ASSIGNED_TAXONOMY_FN
The taxonomy file output by the assign_taxonomy script.

-r REP_SET_FN, **--rep_set_fn** REP_SET_FN
The set of representative sequences.

-s SEQS_OTUS_FN, **--seqs_otus_fn** SEQS_OTUS_FN
The list of OTU IDs and their associated sequence IDs.

Optional arguments

-L {k,p,c,o,f,g,s}, **--phylogenetic_level** {k,p,c,o,f,g,s}
Set the phylogenetic level at which to define OTUs for condensing and downstream processing. Defaults to species level.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.5 filter_rep_set.py

Step 2 of the condensing process. Filter the representative sequence set to include only those sequences that map to unique OTUs.

```
usage: filter_rep_set.py [-h] -r REP_SET_FN -u UNIQUE_OTUS_FN [-o OUTPUT_FILTERED_REP_SET_FN] [-
```

Required arguments

-r REP_SET_FN, **--rep_set_fn** REP_SET_FN
The set of representative sequences.

-u UNIQUE_OTUS_FN, **--unique_otus_fn** UNIQUE_OTUS_FN
The condensed assigned taxonomy file.

Optional arguments

-o OUTPUT_FILTERED_REP_SET_FN, **--output_filtered_rep_set_fn** OUTPUT_FILTERED_REP_SET_FN
The filtered representative set. By default outputs to condensed_rep_set.fna

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.6 multi_parallel_pick_otus.py

Generate PBS scripts for submission to the OSC to run the QIIME parallel blast pick OTUs script on multiple input sequence data sets.

```
usage: osc_parallel_pick_otus.py [-h] -i INPUT_FNA [INPUT_FNA ...] [-t WALLTIME] [-n JOB_NAME] [
```

Required arguments

-i INPUT_FNA [INPUT_FNA ...], **--input_fna** INPUT_FNA [INPUT_FNA ...]
The names of the sequence files that will be have PBS scripts generated to process them. The expected input is from the split_sequence_data.py script (e.g. 0.fna, 1.fna, ..., n.fna).

-t WALLTIME, **--walltime** WALLTIME
The maximum running time to specify to the OSC queuing system for each script.

-n JOB_NAME, **--job_name** JOB_NAME
A descriptive name for the job script that will appear when checking the job status. Max length is 15 characters, but '_#' will be appended to the name you provide to differentiate among all the jobs, so this parameter will be truncated if necessary to accommodate for the number of input files.

-h, --help
Show the help message and exit

-v, --verbose
This will cause the program to print the full path for each output file to the command line. This can be used for informational purposes or to pipe (|) to the PBS multi-submission script to automate job submission as soon as the scripts are created.

1.2.7 otu_condense.py

Step 1 of the condensing process. Take a taxonomy table from the assign_taxonomy QIIME script and prune all redundant taxonomy strings

```
usage: otu_condense.py [-h] -i INPUT_ASSIGNED_TAXONOMY [-p PRUNED_OUTPUT_FILE] [-n NON_UNIQUE_OU
```

Required arguments

-i INPUT_ASSIGNED_TAXONOMY, **--input_assigned_taxonomy** INPUT_ASSIGNED_TAXONOMY
The taxonomy file output by the assign_taxonomy script.

Optional arguments

-p PRUNED_OUTPUT_FILE, **--pruned_output_file** PRUNED_OUTPUT_FILE
The output file for the pruned taxonomy list. Defaults to condensed_assigned_taxonomy.txt

-n NON_UNIQUE_OUTPUT_FILE, **--non_unique_output_file** NON_UNIQUE_OUTPUT_FILE
The file will contain a list of pruned OTU IDs associated with the OTU IDs they replaced. Defaults to nonunique_otu_matrix.txt

-l {k,p,c,o,f,g,s}, **--phylogenetic_level** {k,p,c,o,f,g,s}
Set the phylogenetic level at which to define OTUs for condensing and downstream processing. Defaults to species level.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.8 otu_to_tax_name.py

Convert a list of OTU IDs to a list of OTU IDs paired with Genus_species identifiers.

```
usage: otu_to_tax_name.py [-h] -i OTU_ID_FP -t TAXONOMY_FP [-o OUTPUT_FP]
```

Required arguments

-i OTU_ID_FP, **--otu_id_fp** OTU_ID_FP
Either a text file containing a list (one per line) of OTU IDs, or a tab-separated (classic) BIOM-format file.

-t TAXONOMY_FP, **--taxonomy_fp** TAXONOMY_FP
A file associating OTU ID with a full taxonomic specifier.

Optional arguments

-o OUTPUT_FP, **--output_fp** OUTPUT_FP
For a list input, a new file containing a list of OTU IDs and their corresponding short taxonomic identifiers separated by tabs. For a BIOM file input, a new mapping file with all the OTU IDs replaced by the short identifier.

-h, --help
Show the help message and exit

1.2.9 pick_otus_condense.py

Step 3 of the condensing process. Condense the QIIME pick_otus.py script output by moving the sequences associated with non-unique OTUs to OTU IDs that were identified as unique.

```
usage: pick_otus_condense.py [-h] -s SEQS_OTUS -n NON_UNIQUE_OTU_MATRIX [-o CONDENSED_SEQS_OTUS_
```

Required arguments

-s SEQS_OTUS, **--seqs_otus** SEQS_OTUS
The list of OTU IDs and their associated sequence IDs.

-n NON_UNIQUE_OTU_MATRIX, **--non_unique_otu_matrix** NON_UNIQUE_OTU_MATRIX
The list of unique OTU IDs associated with the OTU IDs they replaced.

-o CONDENSED_SEQS_OTUS_FILE, **--condensed_seqs_otus_file** CONDENSED_SEQS_OTUS_FILE
The condensed set of OTU IDs and the matching sequences. By default outputs to condensed_seqs_otus.txt

Optional arguments

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.10 multi_qsub.py

Submit multiple PBS job scripts to the queuing system (qsub) and store the output job IDs.

```
usage: multi_qsub.py [-h] [-t] job_scripts [job_scripts ...]
```

Required arguments

job_scripts
The job script files to submit to the queuing system.

Optional arguments

-h, --help
Show the help message and exit

-t, --test
Only print each of the qsub commands instead of actually running the commands.

1.2.11 merge_otu_results.py

Distributing sequence data across the cluster for OTU picking results in a set of result files that need to be merged into a single pick otus result.

```
usage: merge_otu_results.py [-h] [-o OUTPUT_FN] [-v] pick_otus_results [pick_otus_results ...]
```

Required arguments

pick_otus_results
The result files from multiple runs of a pick otus script that need to be merged.

Optional arguments

-o OUTPUT_FN, **--output_fn** OUTPUT_FN
The name of the file the merged results will be written to.

-h, --help
Show the help message and exit.

-v, --verbose
Print detailed information about script operation.

1.2.12 primer_average.py

Combine multi-primer pick OTUs results files into a single results file while at the same time averaging sequence counts per sample for OTUs shared between the primer-set results. See reference: Kumar PS et al. (2011) doi:10.1371/journal.pone.0020956

```
usage: primer_average.py [-h] --p1 P1 --p2 P2 [-o OUTPUT_FP] [-v]
```

Required arguments

--p1 P1
Primer-set 1 seqs_otus results files.

--p2 P2
Primer-set 2 seqs_otus results files.

Optional arguments

-o OUTPUT_FP, **--output_fp** OUTPUT_FP
The combined seqs_otus file that has been averaged by shared OTU entries. Default: combined_seqs_otus.txt

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.13 prune_otus.py

Parse the OTU-sequence data in two steps. First remove any OTUs that occur in less than a user-defined percent of samples (default 5%). Second, remove any OTUs that make up less than a user-defined percentage of the overall sequences (default 0.01%)

```
usage: prune_otus.py [-h] -i SEQS_OTUS_FN -t ID_TO_TAXONOMY_FN [-p PERCENT_OF_SAMPLES] [-s PERCENT_OF
```

Required arguments

-i SEQS_OTUS_FN, **--seqs_otus_fn** SEQS_OTUS_FN
The output from the pick OTUs step, e.g. seqs_otus.txt

-t ID_TO_TAXONOMY_FN, **--id_to_taxonomy_fn** ID_TO_TAXONOMY_FN
Path to tab-delimited file mapping sequences to assigned taxonomy.

Optional arguments

-p PERCENT_OF_SAMPLES, **--percent_of_samples** PERCENT_OF_SAMPLES
OTUs that occur in less than this percent of samples will be removed. Default is 5 percent.

-s PERCENT_OF_SEQUENCES, **--percent_of_sequences** PERCENT_OF_SEQUENCES
OTUs that occur in less than this percent of total sequences will be removed. Default is 0.01 percent.

-l {k,p,c,o,f,g,s}, **--phylogenetic_level** {k,p,c,o,f,g,s}
Set the phylogenetic level at which to join OTUs for consideration in pruning. Default is 'g'(group).

-o OUTPUT_PRUNED_OTUS_FN, **--output_pruned_otus_fn** OUTPUT_PRUNED_OTUS_FN
The main output file that will contain the remaining OTUs and sequence IDs.

--output_removed_otus_fn OUTPUT_REMOVED_OTUS_FN
The file to write out the set of OTUs that were removed by the filter.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.14 split_sequence_data.py

Split an input FASTA-formatted sequence file into a user-specified number of smaller files such that the sequence data is evenly distributed among them.

```
usage: split_sequence_data.py [-h] -i INPUT_FASTA_FN [-n NUM_OUTPUT_FILES] [-o OUTPUT_DIR] [-v]
```

Required arguments

-i INPUT_FASTA_FN, **--input_fasta_fn** INPUT_FASTA_FN
The sequence data file to be split up into a series of smaller files.

-n NUM_OUTPUT_FILES, **--num_output_files** NUM_OUTPUT_FILES
The number of files the input data should be split into.

Optional arguments

-o OUTPUT_DIR, **--output_dir** OUTPUT_DIR
The location to write the split data files.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.2.15 transpose_biom.py

Transpose a BIOM-format file so that the matrix is sample by species.

```
usage: transpose_biom.py [-h] -i INPUT_BIOM_FP -m MAPPING [-c MAP_CATEGORY] -o OUTPUT_BIOM_FP [-
```

Required arguments

-i INPUT_BIOM_FP, **--input_biom_fp** INPUT_BIOM_FP
The BIOM-format file.

-m MAPPING, **--mapping** MAPPING
The mapping file specifying group information for each sample.

-o OUTPUT_BIOM_FP, **--output_biom_fp** OUTPUT_BIOM_FP
The BIOM-format file to write.

Optional arguments

-c MAP_CATEGORY, **--map_category** MAP_CATEGORY
A mapping category, such as TreatmentType, that will be used to split the data into separate BIOM files; one for each value found in the category.

-h, --help
Show the help message and exit

-v, --verbose
Print detailed information about script operation.

1.3 Visualization

PhyloToAST scripts used for visualizing data.

1.3.1 iTol.py

Create files appropriate for use in the iTOL visualization program by using the abundance data from a biom-format file and groups specified in a QIIME mapping file. The program also modifies a Newick-format phylogenetic tree file to use proper taxonomic names instead of OTU IDs for useful display in iTOL.

```
usage: iTol.py [-h] -i OTU_TABLE -m MAPPING [-t INPUT_TREE] [-e OUTPUT_TRE] [-o OUTPUT_ITOL_TABLE] [-
```

Required arguments

-i OTU_TABLE, **--otu_table** OTU_TABLE
The biom-format file with OTU-Sample abundance data.

-m MAPPING, **--mapping** MAPPING
The mapping file specifying group information for each sample.

Optional arguments

-t INPUT_TREE, **--input_tree** INPUT_TREE
A phylogenetic tree in Newick format to be modified by exchanging the OTU ID node names for taxonomic names.

-e OUTPUT_TRE, **--output_tre** OUTPUT_TRE
The output Newick-format tree (.tre) file

-o OUTPUT_ITOL_TABLE, **--output_itol_table** OUTPUT_ITOL_TABLE
Other than a phylogenetic tree, the main input to iTOL is a dataset file containing some representation of the abundance of every OTU across the specified data groups. This program provides multiple calculation methods. See the –analysis_metric option for details.

-c MAP_CATEGORIES, **--map_categories** MAP_CATEGORIES
Any mapping categories, such as treatment type, that will be used to group the data in the output iTOL table. For example, one category with three types will result in three data columns in the final output. Two categories with three types each will result in six data columns. Default is no categories and all the data will be treated as a single group.

-a {MRA,NMRA,raw}, **--analysis_metric** {MRA,NMRA,raw}
Specifies which metric is calculated on the abundance data in the OTU table. Available options: MRE - mean relative abundance (Abundance data is normalized by total sample abundance, then averaged across OTU), NMRE - normalized mean relative abundance (MRE normalized by the total MRE across the groups as specified in –map_categories), raw (outputs the actual sequence abundance data for each OTU).

--stabilize_variance
Apply the variance-stabilizing arcsine square root transformation to the OTU proportion data. Recommended for usage with `-a NMRA` or `-a MRA`.

-h, --help
Show the help message and exit.

Workflow for generating useful phylogenetic trees using PhyloToAST

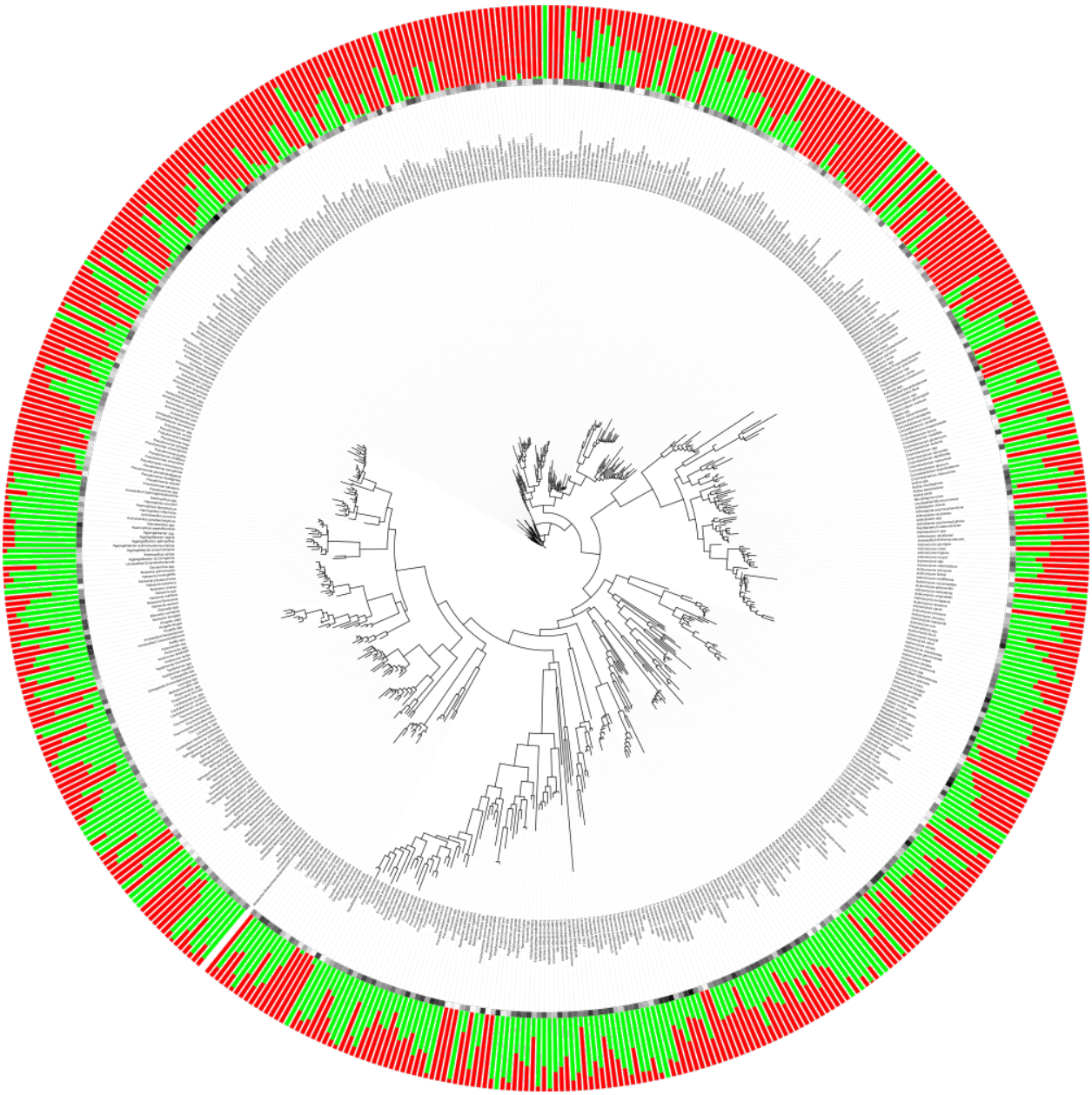
Step 1 : Obtain .tre file from QIIME's make_phylogeny.py script.

Step 2 : Run `iTol.py` script with `-a NMRA` analysis metric. This file will denote the multibar graph around the circular phylogenetic tree.

Step 3 : Run `iTol.py` script with `-a raw` analysis metric. This file will denote the gradient graph around the circular phylogenetic tree.

Step 4 : Upload modified .tre file from `iTol.py` script to iTOL website. Add your dataset files and obtain the final phylogenetic tree figure.

Example output image



NOTE : Please refer to iTOL help page for changing dataset parameters.

1.3.2 LDA.py

Create an LDA plot from sample-grouped OTU data. It is necessary to remove the header cell '#OTU ID' before running this program.

```
usage: LDA.py [-h] -i BIOM_TSV -m MAP_FP -g GROUP_BY [GROUP_BY ...] -c COLOR_BY [--dpi DPI] [--save_
```

Required arguments

-i BIOM_TSV, **--biom_tsv** BIOM_TSV
Sample-OTU abundance table in TSV format with the arcsin sqrt transform already applied.

-m MAP_FP, **--map_fp** MAP_FP
Metadata mapping file.

Optional arguments

-g GROUP_BY [GROUP_BY ...], **--group_by** GROUP_BY [GROUP_BY ...]
Any mapping categories, such as treatment type, that will be used to group the data in the output iTol table. For example, one category with three types will result in three data columns in the final output. Two categories with three types each will result in six data columns. Default is no categories and all the data will be treated as a single group.

-c COLOR_BY, **--color_by** COLOR_BY
A column name in the mapping file containing hexadecimal (#FF0000) color values that will be used to color the groups. Each sample ID must have a color entry.

--dpi DPI
Set plot quality in Dots Per Inch (DPI). Larger DPI will result in larger file size.

--save_lda_input SAVE_LDA_INPUT
Save a CSV-format file of the transposed LDA-input table to the file specifed by this option.

--plot_title PLOT_TITLE
Plot title. Default is no title.

-o OUT_FP, **--out_fp** OUT_FP
The path and file name to save the plot under. If specified, the figure will be saved directly instead of opening a window in which the plot can be viewed before saving.

-h, --help
Show the help message and exit.

1.3.3 PCoA.py

Create a series of 2D or 3D PCoA plots where the marker size varies by relative abundance of a particular OTU.

```
usage: PCoA.py [-h] -i COORD_FP -m MAP_FP -b COLORBY [-o OUT_FN] [-d {2,3}][-t TITLE] [--save] [-c MA
```

Required Arguments

-i COORD_FP, **--coord_fp** COORD_FP
Input principal coordinates filepath (i.e., resulting file from principal_coordinates.py).

-m MAP_FP, **--map_fp** MAP_FP
Input metadata mapping file-path.

-b COLORBY, **--colorby** COLORBY
Metadata categories (column headers) to color by in the plots.

Optional Arguments

-d {2,3}, **--dimensions** {2,3}
Choose whether to plot 2D or 3D.

-c COLORS, **--colors** COLORS
A file containing user defined colors in hex values or matplotlib named colors, each on separate line. If user color list is not sufficient or ont defined, program will use Qualitative Set1 scheme from brewer colors. More information on matplotlib named colors.

-s POINT_SIZE, **--point_size** POINT_SIZE
Specify the size of the circles representing each of the samples in the plot.

--pc_order PC_ORDER
Choose which Principle Coordinates are displayed and in which order, for example: 1,2 (Note the lack of any spaces around the comma).

--x_limits X_LIMITS X_LIMITS
Specify limits for the x-axis instead of automatic setting based on the data range. Should take the form: –x_limits -0.5 0.5

--y_limits Y_LIMITS Y_LIMITS
Specify limits for the y-axis instead of automatic setting based on the data range. Should take the form: –y_limits -0.5 0.5

--z_limits Z_LIMITS Z_LIMITS
Specify limits for the z-axis instead of automatic setting based on the data range. Should take the form: –z_limits -0.5 0.5

-t TITLE, **--title** TITLE
Title of the plot.

--dpi DPI
Set plot quality in Dots Per Inch (DPI). Larger DPI will result in larger file size.

-o OUT_FP, **--out_fp** OUT_FP
The path and file name to save the plot under. If specified, the figure will be saved directly instead of opening a window in which the plot can be viewed before saving.

-h, **--help**
Show the help message and exit.

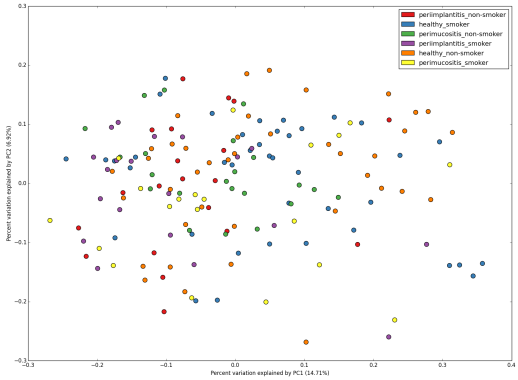
Workflow for generating PCoA plots using PhyloToAST

Step 1 : Obtain unifrac principal coordinates file from QIIME's beta_diversity_through_plots.py script.

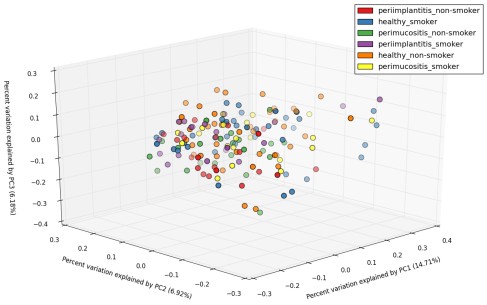
Step 2 : Run `PCoA.py` script with `-t Path to the tree file` parameters.

Example plots

2D PCoA plot with 2 metadata categories - DiseaseState and SmokingStatus.



3D PCoA plot with 2 metadata categories - DiseaseState and SmokingStatus.



1.3.4 PCoA_bubble.py

Create a series of Principle Coordinate plots for each OTU in an input list where the plot points are varied in size by the relative abundance of the OTU (relative to either Sample or the total contribution of the OTU to the data set.

```
usage: PCoA_bubble.py [-h] -i OTU_TABLE -u UNIFRAC -d NAMES_COLORS_IDS_FN -m MAPPING -c MAP_CATEGORY
```

Required Arguments

-i OTU_TABLE, **--otu_table** OTU_TABLE
The biom-format file with OTU-Sample abundance data.

-u UNIFRAC, **--unifrac** UNIFRAC
Principle coordinates analysis file. Eg. unweighted_unifrac_pc.txt

-d NAMES_COLORS_IDS_FN, **--names_colors_ids_fn** NAMES_COLORS_IDS_FN
The name of an input data file containing three items: Line 1: a tab-separated list of display names for the different types in the specified mapping category (–mapping_category), Line 2: a matching tab-separated list of hexadecimal colors for each of the category types, Lines 3-end: a tab-separated pair specifying OTU ID and OTU Name. Each entry will get a separate PCoA plot under a file with the name of the OTU.

-m MAPPING, **--mapping** MAPPING
The mapping file specifying group information for each sample.

-c MAP_CATEGORY, **--map_category** MAP_CATEGORY
Any mapping category, such as treatment type, that will be used to group the data in the output plots. For example, one category with three types will result in three different point sets in the final output.

Optional Arguments

-o OUTPUT_DIR, **--output_dir** OUTPUT_DIR
The directory to output the PCoA plots to.

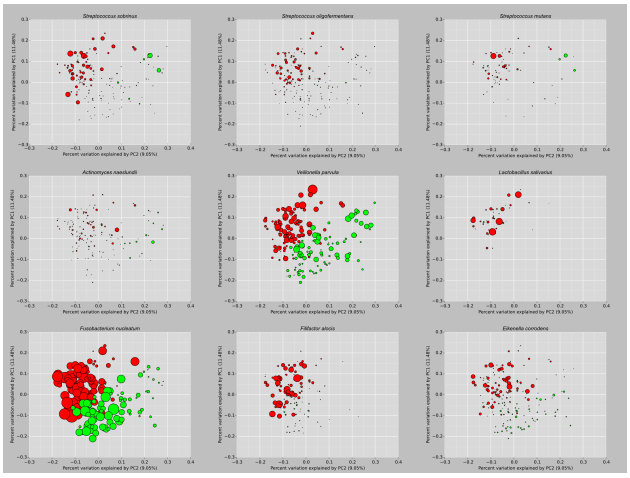
--scaling_factor SCALING_FACTOR
Species relative abundance is multiplied by this factor in order to make appropriate visible bubbles in the output plots. Default is 10000.

-v, --verbose
Displays species name as each is being plotted and stored to disk.

-h, --help
Show this help message and exit

Example plot

PCoA bubble plots of subgingival microbiome pathogens of smokers [1].



Citation:

1.3.5 diversity.py

Calculate and plot for two sample categories: Shannon diversity, Chao1 diversity, and a Jaccard similiarity distance matrix heatmap.

```
usage: usage: diversity.py [-h][-c {shannon,chao1,jaccard} [{shannon,chao1,jaccard} ...]] [-p IMAGE_T
```

Required arguments

map_file
QIIME mapping file.

[1] **The Subgingival Microbiome of Clinically Healthy Current and Never Smokers**. Matthew R Mason, Philip M Preshaw, Haikady N Nagaraja, Shareef M Dabdoub, Anis Rahman and Purnima S Kumar; doi: 10.1038/ismej.2014.114

biom_file
BIOM table file name

category
Specific category from the mapping file.

plot_title
The name of a PDF file the pathway map will be written to.

out_dir
The directory all plots will be saved to.

Optional arguments

-h, --help
show this help message and exit

-c {shannon,chao1,jaccard} [{shannon,chao1,jaccard} ...]
Choose the type of calculation needed. Default value is "None", which will output all 3 types of calculations.

-p IMAGE_TYPE, **--image_type** IMAGE_TYPE
The type of image to save: PNG, SVG, etc.

Citations:

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Symbols

D

E

F

H