QIIME:Moving Pictures of the human microbiome

Bonnie Hurwitz

Abstract

This tutorial covers a full QIIME workflow using Illumina sequencing data and was adapted from a tutorial on the QIIME website.. This tutorial is intended to be quick to run, and as such, uses only a subset of a full Illumina Genome Analyzer II (GAIIx) run. We'll make use of the Greengenes reference OTUs, which is the default reference database used by QIIME. You can determine which version of Greengenes is being used by running print_qiime_config.py. This will be Greengenes, unless you've configured QIIME to use a different reference database by default.The data used in this tutorial are derived from the Moving Pictures of the Human Microbiome study, where two human subjects collected daily samples from four body sites: the tongue, the palm of the left hand, the palm of the right hand, and the gut (via fecal samples obtained by swapping used toilet paper). These data were sequenced using the barcoded amplicon sequencing protocol described in Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample. A more recent version of this protocol that can be used with the Illumina HiSeq 2000 and MiSeq can be found here.

Citation: Bonnie Hurwitz QIIME:Moving Pictures of the human microbiome. **protocols.io** dx.doi.org/10.17504/protocols.io.d5288d
Published: 25 Nov 2015

Guidelines

This tutorial is run using the suite of python tools in QIIME.

Before start

Make sure you have QIIME installed or are using the QIIME virtual machine. This tutorial is specifically designed to run on the HPC at the University of Arizona.

Protocol

Getting started

Step 1.
We'll begin by downloading the tutorial data.

cmd COMMAND

```
wget ftp://ftp.microbio.me/qiime/tutorial_files/moving_pictures_tutorial-1.9.0.tgz || curl -O ftp://ftp.microbio.me/qiime/tutorial_files/moving_pictures_tutorial-1.9.0.tgz
```

NOTES

Bonnie Hurwitz 25 Nov 2015

The data used in this tutorial are derived from the Moving Pictures of the Human Microbiome study, where two human subjects collected daily samples from four body sites: the tongue, the palm of the left hand, the palm of the right hand, and the gut (via fecal samples obtained by swapping used toilet paper). These data were sequenced using the barcoded amplicon sequencing protocol described in Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample. A more recent version of this protocol that can be used with the Illumina HiSeq 2000 and MiSeq can be found here.

Getting started

Step 2.

Unzip the data files with the tutorial information

cmd COMMAND

```
tar -xzf moving_pictures_tutorial-1.9.0.tgz
```

Getting started

Step 3.

We'll change to the moving_pictures_tutorial-1.9.0/illumina directory for the remaining steps

cmd COMMAND

```
cd moving_pictures_tutorial-1.9.0/illumina
```

Step 4.

Make sure the QIIME software is loaded on the hpc.

cmd COMMAND

```
module load qiime
```

Validate Mapping File

Step 5.

Check our mapping file for errors. The QIIME mapping file contains all of the per-sample metadata, including technical information such as primers and barcodes that were used for each sample, and information about the samples, including what body site they were taken from. In this data set we're looking at human microbiome samples from four sites on the bodies of two individuals at mutliple time points. The metadata in this case therefore includes a subject identifier, a timepoint, and a body site for each sample. You can review the map.tsv file at the link in the previous cell to see an example of the data (or view the published Google Spreadsheet version, which is more nicely formatted).In this step, we run validate_mapping_file.py to ensure that our mapping file is compatible with QIIME.

cmd COMMAND

```
validate_mapping_file.py -o vmf-map/ -m map.tsv
```

NOTES

Bonnie Hurwitz 06 Nov 2015

In this case there were no errors, but if there were we would review the resulting HTML summary to find out what errors are present. You could then fix those in a spreadsheet program or text editor and rerun validate_mapping_file.py on the updated mapping file.

Bonnie Hurwitz 25 Nov 2015

If you need to create a mapping file form scratch use the format noted here

Validate Mapping File

Step 6.

Create your own mapping file. In some circumstances, users may need to generate a mapping file that does not contain barcodes and/or primers. To generate such a mapping file, fields for “BarcodeSequence” and “LinkerPrimerSequence” can be left empty. An example of such a file is below (note that the tabs are still present for the empty “BarcodeSequence” and “LinkerPrimerSequence” fields):

cmd COMMAND

```
validate_mapping_file.py -m  -o check_id_output/ -p -b
```

NOTES

Bonnie Hurwitz 25 Nov 2015

Example mapping file:

#SampleID BarcodeSequence LinkerPrimerSequence Treatment DOB Description

#Example mapping file for the QIIME analysis package. These 9 samples are from a study of the effects of

#exercise and diet on mouse cardiac physiology (Crawford, et al, PNAS, 2009).

PC.354 Control 20061218 Control_mouse__I.D._354

PC.355 Control 20061218 Control_mouse__I.D._355

PC.356 Control 20061126 Control_mouse__I.D._356

PC.481 Control 20070314 Control_mouse__I.D._481

PC.593 Control 20071210 Control_mouse__I.D._593

PC.607 Fast 20071112 Fasting_mouse__I.D._607

PC.634 Fast 20080116 Fasting_mouse__I.D._634

PC.635 Fast 20080116 Fasting_mouse__I.D._635

PC.636 Fast 20080116 Fasting_mouse__I.D._636

Demultiplexing and quality filtering sequences

Step 7.

Demultiplexing and quality filtering sequences. We next need to demultiplex and quality filter our sequences (i.e. assigning barcoded reads to the samples they are derived from). In general, you should get separate fastq files for your sequence and barcode reads. Note that we pass these files while still gzipped. split_libraries_fastq.py can handle gzipped or unzipped fastq files. The default strategy in QIIME for quality filtering of Illumina data is described in Bokulich et al (2013).

cmd COMMAND

```
split_libraries_fastq.py -o slout/ -i forward_reads.fastq.gz -b barcodes.fastq.gz -
m map.tsv
```

NOTES

Bonnie Hurwitz 25 Nov 2015

Note you can skip this step if you are starting from fasta or fastq files that have already been demultiplexed. See step 8.

Uploading existing data

Step 8.

Sometimes, you would like to use QIIME with an existing dataset that has already been split into individual fastq or fasta files.

In order for the downstream modules of QIIME to associate sequences with particular samples, these

demultiplexed sequences need to be labeled in such a way that the SampleID (see mapping file format) and sequence number are incorporated into the fasta label (see example in annotation below).

cmd COMMAND

```
#! /usr/bin/perl

use strict;

my $file = shift @ARGV;
my $sample = shift @ARGV;
my $out = "slout/seqs.fna";

open (F, $file) || die;
open (OUT, ">>$out") || die "Cannot open outfile";

my $count = 0;
while () {
  chomp $_;
  if ($_ =~ /^>/) {
    $count++;
    print OUT ">$sample" . "_" . "$count\n";
  }
  else {
    print OUT "$_\n";
  }
}
```

NOTES

Bonnie Hurwitz 25 Nov 2015
The first sequence in a fasta file that is associated with sample PC.634 should look like:

>PC.634_1

CTGGGCCGTGTCTCAGTCCCAATGTGGCCGTTTACCCTCTCAGGCCGGCTACGCATCATCGCCTTGGTGGG
C

Bonnie Hurwitz 25 Nov 2015
Note that the command below will append to a file called seqs.fna. You will need to run this on each sample. You can use a for loop like so:
for sample `cat samples`; do
./convert_to_qiime.pl $sample.fna $sample
done

Check the data

Step 9.
We can see how many sequences we ended up with using count_seqs.py.

cmd COMMAND

```
count_seqs.py -i slout/seqs.fna
```

OTU Picking

Step 10.
OTU picking: using an open-reference OTU picking protocol by searching reads against the Greengenes database.Now that we have demultiplexed sequences, we're ready to cluster these sequences into OTUs. There are three high-level ways to do this in QIIME. We can use *de novo*, *closed-*

reference, or *open-reference OTU picking*. Open-reference OTU picking is currently our preferred method. Discussion of these methods can be found in Rideout et. al (2014).
Here we apply open-reference OTU picking. **Note that this command takes the seqs.fna file that was generated in the previous step**. We're also specifying some parameters to the pick_otus.py command, which is internal to this workflow. Specifically, we set enable_rev_strand_match to True, which allows sequences to match the reference database if either their forward or reverse orientation matches to a reference sequence. This parameter is specified in the*parameters file* which is passed as -p. You can find information on defining parameters files here.

DURATION
00:15:00

cmd COMMAND
```
pick_open_reference_otus.py -o otus/ -i slout/seqs.fna -p ../uc_fast_params.txt
```

NOTES
Bonnie Hurwitz 06 Nov 2015
The primary output that we get from this command is the *OTU table*, or the number of times each operational taxonomic unit (OTU) is observed in each sample. QIIME uses the Genomics Standards Consortium Biological Observation Matrix standard (BIOM) format for representing OTU tables. You can find additional information on the BIOM format here, and information on converting these files to tab-separated text that can be viewed in spreadsheet programs here. Several OTU tables are generated by this command. The one we typically want to work with is otus/otu_table_mc2_w_tax_no_pynast_failures.biom. This has singleton OTUs (or OTUs with a total count of 1) removed, as well as OTUs whose representative (i.e., centroid) sequence couldn't be aligned with PyNAST. It also contains taxonomic assignments for each OTU as *observation metadata*.The open-reference OTU picking command also produces a phylogenetic tree where the tips are the OTUs. The file containing the tree is otus/rep_set.tre, and is the file that should be used with otus/otu_table_mc2_w_tax_no_pynast_failures.biom in downstream phylogenetic diversity calculations. The tree is stored in the widely used newick format.To view the output of this command, look at the index.html file in the output directory.
Bonnie Hurwitz 25 Nov 2015

To view the output of this command, look at the index.html file in the output directory.

OTU Summary

Step 11.
To compute some summary statistics of the OTU table we can run the following command.

cmd COMMAND
```
biom summarize-table -i otus/otu_table_mc2_w_tax_no_pynast_failures.biom
```

NOTES
Bonnie Hurwitz 06 Nov 2015
The key piece of information you need to pull from this output is the depth of sequencing that should be used in diversity analyses. Many of the analyses that follow require that there are an equal number of sequences in each sample, so you need to review the *Counts/sample detail* and decide what depth you'd like. Any samples that don't have at least that many sequences will not be included in the analyses, so this is always a trade-off between the number of sequences you throw away and the number of samples you throw away. For some perspective on this, see Kuczynski 2010.

Run Diversity Analysis

Step 12.

Run diversity analyses. Here we're running the core_diversity_analyses.py script which applies many of the "first-pass" diversity analyses that users are generally interested in. The main output that users will interact with is the index.html file, which provides links into the different analysis results.

DURATION

00:15:00

cmd COMMAND

```
core_diversity_analyses.py -o cdout/ -i otus/otu_table_mc2_w_tax_no_pynast_failures.biom -
m map.tsv -t otus/rep_set.tre -e 1114 -c SampleType,DaysSinceExperimentStart
```

NOTES

Bonnie Hurwitz 06 Nov 2015

You may see a RuntimeWarning generated by this command. As the warning indicates, it's not something that you should be concerned about in this case. QIIME (and scikit-bio, which implements a lot of QIIME's core functionality) will *sometimes* provide these types of warnings to help you figure out if your analyses are valid, but you should always be thinking about whether a particular test or analysis is relevant for your data. Just because something can be passed as input to a QIIME script, doesn't necessarily mean that the analysis it performs is appropriate.

Bonnie Hurwitz 25 Nov 2015

Note that in this step we're passing -e which is the sampling depth that should be used for diversity analyses. I chose 1114 here, based on reviewing the above output from biom summarize-table. This value will be study-specific, so don't just use this value on your own data (though it's fine to use that value for this tutorial).

Bonnie Hurwitz 25 Nov 2015

To get the correct graphic produced, you will need to create a file called:

vi ~/.config/matplotlib/matplotlibrc

and add the following to the file "backend : agg"

Step 13.

The results above treat all samples independently, but sometimes (for example, in the taxonomic summaries) it's useful to categorize samples by their metadata. We can do this by passing categories (i.e., headers from our mapping file) to core_diversity_analyses.py with the -c parameter. Becausecore_diversity_analyses.py can take a long time to run, it has a --recover_from_failure option, which can allow it to be rerun from a point where it previously failed in some cases (for example, if you accidentally turned your computer off while it was running). This option can also be used to add categorical analyses if you didn't include them in your initial run. Next we'll rerun core_diversity_analyses.py with two sets of categorical analyses: one for the"SampleType category, and one for the DaysSinceExperimentStart category. Remember the --recover_from_failure option: it can save you a lot of time.

cmd COMMAND

```
core_diversity_analyses.py -o cdout/ --recover_from_failure -
c "SampleType,DaysSinceExperimentStart" -
i otus/otu_table_mc2_w_tax_no_pynast_failures.biom -m map.tsv -t otus/rep_set.tre -e 1114
```

Step 14.

One thing you may notice in the PCoA plots generated by core_diversity_analyses.py is that the samples don't cluster perfectly by SampleType. This is unexpected, based on what we know about the human microbiome. Since this is a time series, let's explore this in a little more detail integrating a

time axis into our PCoA plots. We can do this by re-running Emperor directly, replacing our previously generated PCoA plots. (Emperor is a tool for the visualization of PCoA plots with many advanced features that you can explore in the Emperor tutorial. If you use Emperor in your research you should be sure to cite it directly, as with the other tools that QIIME wraps, such as uclust and RDPClassifier.)After this runs, you can reload the Emperor plots that you accessed from the above cdout/index.html links. Try making the samples taken duringAntibioticUsage invisible.

cmd COMMAND

```
make_emperor.py -i cdout/bdiv_even1114/weighted_unifrac_pc.txt -
o cdout/bdiv_even1114/weighted_unifrac_emperor_pcoa_plot -m map.tsv --
custom_axes DaysSinceExperimentStart

make_emperor.py -i cdout/bdiv_even1114/unweighted_unifrac_pc.txt -
o cdout/bdiv_even1114/unweighted_unifrac_emperor_pcoa_plot -m map.tsv --
custom_axes DaysSinceExperimentStart
```

NOTES

Bonnie Hurwitz 06 Nov 2015

IMPORTANT: Removing points from a PCoA plot, as is suggested above for data exploration purposes, is not the same as computing PCoA without those points. If after running this, you'd like to remove the samples taken during AntibioticUsage from the analysis, you can do this withfilter_samples_from_otus_table.py, which is discussed here. As an exercise, try removing the samples taken during AntibioticUsage from the OTU table and re-running core_diversity_analyses.py. You should output the results to a different directory than you created above (e.g., cdout_no_abx).

Bonnie Hurwitz 25 Nov 2015

After this runs, you can reload the Emperor plots that you accessed from the above cdout/index.html links. Try making the samples taken during AntibioticUsage invisible