QIIME-1.9.1 Using Atmosphere

Rationale and background:

QIIME (canonically pronounced *chime)*: **Quantitative Insights Into Microbial Ecology** (http://www.qiime.org)

> *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*

QIIME is an open-source bioinformatics pipeline for performing microbiome analysis from raw DNA sequencing data. QIIME is designed to take users from raw sequencing data generated on the Illumina or other platforms through publication quality graphics and statistics. This includes demultiplexing and quality filtering, OTU picking, taxonomic assignment, and phylogenetic reconstruction, and diversity analyses and visualizations. QIIME has been applied to studies based on billions of sequences from tens of thousands of samples.

Introduction

This tutorial will orient you to using the QIIME software package (version 1.9.1) installed on Atmosphere. This particular demo is adapted from the Illumina tutorial on the QIIME site.

This tutorial will take users through steps of:

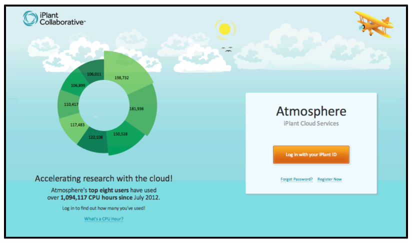
1. Launching the QIIME-1.9.1 Atmosphere image
2. Running QIIME-1.9.1 on an test data

Please work through the tutorial and add your comments on the bottom of this page. Or send comments per email to upendra@cyverse.org. Thank you.

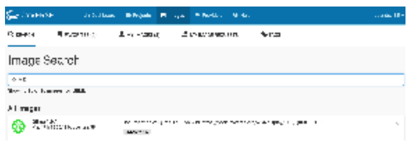
Learn about allocations
Learn about CyVerse's allocation policies here.

Part 1: Connect to an instance of an Atmosphere Image (Virtual Machine)

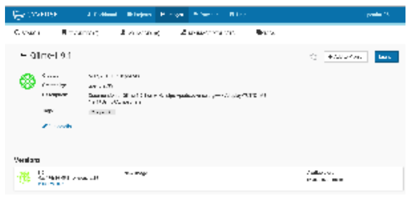
Step 1. Go to https://atmo.iplantcollaborative.org and log in with your CyVerse credentials.


iPlant Collaborative
Atmosphere
iPlant Cloud Services
Log in with your iPlant ID
Forgot Password? Register Now
Accelerating research with the cloud!
Atmosphere's top eight users have used over 1,094,117 CPU hours since July 2012.
Log in to find out how many you've used!
What's a CPU Hour?

Step 2. Click on the **Launch New Instance** button and search for **Qiime-1.9.1**



Step 3. Select the image **Qiime-1.9.1** and click **Launch Instance**. It will take ~10-15 minutes for the cloud instance to be launched.



Note: *Instances can be configured for different amounts of CPU, memory, and storage depending on user needs. This tutorial can be accomplished with the small instance size,* ***medium1 (4 CPUs, 8 GB memory, 80 GB root)***

Part 2: Set up a Qiime-1.9.1 run using the Terminal window

Step 1. Open the **Terminal**. Add the ssh details along with your IP address to connect the instance through the terminal. **Remember to put your actually iPlant username in place of the text 'username' and 'IPaddress' in this next line of code:**

```
$ ssh <username>@<IPaddress>
```

Step 2. You will find tutorial data in "/opt/qiime_illumina_test_data" folder. List its contents with the **ls** command.

```
$ ls /opt/qiime_illumina_test_data
barcodes.fastq.gz  forward_reads.fastq.gz  map-bad.tsv  map.tsv  precomputed-output
uc_fast_params.txt
```

We'll change to the `illumina` directory for the remaining steps.

```
$ cd /opt/qiime_illumina_test_data
```

Part 3: Run Qiime-1.9.1

Step 1: 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 current tutorial 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 to see an example of the data (or view the publis hed 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. First we will use a perfectly normal mapping file to do test this

```
$ validate_mapping_file.py -o ~/qiime_illumina_test/vmf-map/ -m map.tsv
# No errors or warnings were found in mapping file.
```

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.

For the sake of illustrating what errors in a mapping file might look like, we've created a bad mapping file (`map-bad.tsv`). We'll next call `valida te_mapping_file.py` on the file `map-bad.tsv`.

```
$ validate_mapping_file.py -o ~/qiime_illumina_test/vmf-map-bad/ -m map-bad.tsv
# Errors and/or warnings detected in mapping file.  Please check the log and html file
for details.
```

```
$ cat ~/qiime_illumina_test/vmf-map-bad/map-bad.tsv.log
# Errors and warnings are written as a tab separated columns, with the first column
showing the error or warning, and the second column contains the location of the error
or warning, written as row,column, where 0,0 is the top left header item (SampleID).
Problems not specific to a particular data cell will be listed as having 'no
location'.
Errors -----------------------------
Duplicate barcode AGCTGACTAGTC found.   1,1
Duplicate barcode AGCTGACTAGTC found.   6,1
Warnings ---------------------------
Found invalid character in DaysSinceExperimentStart! header field.      0,9
```

Step 2: 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 to `split_libraries_fastq.py` command which 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).

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

The above command creates 3 new files which we can view:

```
$ ls ~/qiime_illumina_test/slout
histograms.txt  seqs.fna  split_library_log.txt
```

We can examine the results of the split libraries commands by using the **less**.

```
$ less ~/qiime_illumina_test/slout/split_library_log.txt
```

You can also see how many sequencing reads passed after demultiplexing and quality filtering using the `count_seqs.py`.

```
$ count_seqs.py -i ~/qiime_illumina_test/slout/seqs.fna
# 186333  : slout/seqs.fna (Sequence lengths (mean +/- std): 132.2422 +/- 9.8806)
# 186333  : Total
```

Step 3: 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. The QIIME demo here uses Open-reference OTU picking (though it takes some time but this is currently QIIME's preferred method). Discussion of these methods can be found in Rideout et. al (2014). Note that this command takes the slout/`seqs.fna` file that was generated in the previous step. 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.

This step can take long time to complete since we are working on a small machine. With a larger instance you can take advantage of parallelization.

```
$ pick_open_reference_otus.py -o ~/qiime_illumina_test/otus/ -i
~/qiime_illumina_test/slout/seqs.fna -p uc_fast_params.txt
```

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 compute some summary statistics of the OTU table we can run the biom summarize-table command.

```
$ biom summarize-table -i
~/qiime_illumina_test/otus/otu_table_mc2_w_tax_no_pynast_failures.biom
Num samples: 34
Num observations: 3426
Total count: 182140
Table density (fraction of non-zero values): 0.122
Counts/sample summary:
 Min: 1114.0
 Max: 11449.0
 Median: 4966.000
 Mean: 5357.059
 Std. dev.: 3333.813
 Sample Metadata Categories: None provided
 Observation Metadata Categories: taxonomy
Counts/sample detail:
L3S242: 1114.0
L3S341: 1234.0
L3S360: 1240.0
L3S313: 1304.0
L3S294: 1472.0
L3S378: 1558.0
L2S309: 1842.0
L5S155: 1982.0
L5S240: 2127.0
L5S174: 2176.0
L5S203: 2352.0
L5S104: 2505.0
L5S222: 2771.0
L2S357: 3016.0
L2S222: 4035.0
L2S204: 4131.0
L2S155: 4880.0
L2S382: 5052.0
L2S175: 5437.0
L2S240: 7054.0
L1S257: 7165.0
L6S68: 7281.0
L1S281: 7548.0
L6S20: 7882.0
L1S8: 8320.0
L1S140: 8477.0
L6S93: 8509.0
L1S76: 8837.0
L1S105: 9065.0
L4S112: 9705.0
L1S57: 9793.0
L1S208: 9862.0
L4S63: 10965.0
L4S137: 11449.0
```

The key piece of information you need to pull from this output is the depth of sequencing (Counts/sample) that should be used in diversity analyses (step 4). Many of the analyses that follow require that there are an equal number of sequences in each sample, so you need to review the output 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.

Step 4: Run diversity analyses

Now we run the `core_diversity_analyses.py` script which does the diversity analyses that users are generally interested in. The main output that users will interact with is the otus/`index.html` file, which provides links into the different analysis results. Note that in this step we're passing `-e` (depth of sampling) that should be used for diversity analyses. Based on reviewing the above output from `biom summarize-table, 1114 seems to be e value`. 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).

```
$ core_diversity_analyses.py -o ~/qiime_illumina_test/cdout/ -i
~/qiime_illumina_test/otus/otu_table_mc2_w_tax_no_pynast_failures.biom -m map.tsv -t
~/qiime_illumina_test/otus/rep_set.tre -e 1114
```

Step 5: Run emperor

After this runs, you can reload the Emperor plots that you accessed from the above `cdout/index.html` links.

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

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

If you want to run the Qiime 1.9.1 on Atmosphere with Jupyter notebook you can do so using this protocol - QIIME-1.9.1 Using Atmosphere on Jupyter Notebook