A HTS Pipeline using USEARCH, Qiime, and FAST

Zewei Song

# SECTION 0. SET UP WORKING ENVIRONMENT

## 0.1 SET UP USER PATH

In general, you can put the USEARCH executable and all FAST script in a single folder, and set the path as a path environment variable. By doing these, we can call these program anywhere in our computer. But you can also copy all these program to your working folder and use them.

***Download Program:***

USEARCH: <http://www.drive5.com/usearch/download.html>

Submit your email address, and the website will send a link to your mailbox. Beware to choose the right platform for you. USEARCH will be a single file, with a version number. I always rename it to “usearch” for convenient.

FAST: <https://github.com/ZeweiSong/FAST>

This is a GitHub repository with several script I wrote to pre-process the sequencing files, dereplicate, and handling the Qiime style OTU map. You can download the entire repository using “Download ZIP” button on the right and unzip the content to the same folder as USEARCH executable. You need to install Python 2.7 in order to use these scripts (<https://www.python.org/>). Python 3 should also work without any problem, but I haven’t test it under Python 3.

Trimmomatic: <http://www.usadellab.org/cms/index.php?page=trimmomatic>

Trimmomatic has to be in your working folder. You also need to install latest Java to make it work.

***Path Variable:***

For Windows user, add the path of the tool folder to the Path Variable of under User Variable. Following this to find the Path variable: Control Panel\System and Security\System\Advanced system setting\Environmental variables\User variables for User\_name

For Linux user, modify you .bashrc to include the folder path. Usually you can find this file under your home directory (~). Use vi ~/.bashrc to make any modification. Usually you would like to save the tool folder to /usr/local/bin.`

Use echo $PATH to check you current path variable.

## 0.2 SET UP WORKING FOLDER

Make a new folder, and under it make a new folder names “Raw”. Save all your R1 read FASTQ files to the “Raw” folder.

If you did not set up the path variable for the programs, copy USEARCH executable, and all FAST scripts including any subfolders to your working folder.

This is any example of the file structure you should have:

Directory of D:\Example

04/22/2015 04:39 PM <DIR> .

04/22/2015 04:39 PM <DIR> ..

04/13/2015 11:28 AM 2,762 add\_labels.py

04/21/2015 08:38 AM 1,912 add\_seqs\_size.py

04/16/2015 06:04 PM 2,261 convert\_fastq.py

04/13/2015 11:28 AM 1,475 correct\_fasta.py

04/17/2015 08:23 AM 1,850 count\_seqs.py

04/20/2015 03:30 PM 5,003 dereplicate.py

04/22/2015 09:16 AM 2,689 filter\_otu\_map.py

04/22/2015 10:09 AM 2,986 filter\_seqs.py

04/13/2015 11:28 AM 2,824 generate\_mapping.py

04/22/2015 09:16 AM <DIR> lib

04/22/2015 01:10 PM 1,747 make\_otu\_table.py

04/22/2015 01:04 PM 1,938 merge\_otu\_maps.py

04/22/2015 09:16 AM 2,138 merge\_seqs.py

04/22/2015 01:01 PM 2,036 parse\_uparse\_cluster.py

04/22/2015 11:43 AM 2,533 pick\_seqs.py

04/22/2015 04:38 PM <DIR> Raw

04/22/2015 04:01 PM 164 README.md

04/17/2015 08:23 AM 4,461 stat\_seqs.py

11/15/2014 04:35 PM 118,310 trimmomatic-0.32.jar

01/22/2015 03:40 PM 883,201 usearch.exe

18 File(s) 1,040,290 bytes

4 Dir(s) 496,106,565,632 bytes free

## 0.3 TEST RUN

***All Python Scripts:***

@ python add\_labels.py -h

Use -h any time to see the usage for the script.

You should be able to run all python scripts by just typing its file name without starting with the word python.

***Usearch:***

@ usearch

***Trimmomatic:***

@java -jar Trimmomatic-0.32.jar

For both Windows and Linux, use tab will autocomplete any file or folder in your current directory.

# SECTION 1. LABEL SEQUENCE FILES

## 1.1 RELABEL FASTQ FILES:

***Generate File Mapping:***

@ generate\_mapping.py -i Raw -o mapping.txt

Open the mapping file and change any labels, the script will take a good guess basing on the file name.

***Add Labels:***

@ add\_labels.py -i Raw -o Labeled -m mapping.txt -t 4

Take a look at the files in the new folder “Labeled” using more under Windows, and less under Linux.

It should look like:

@SampleA\_1;SampleName=SampleA-1-R1;barcodelabel=SampleA;

GTAGGTGAACCTGCGGAAGGATCATTATCGATTCACGGGAAAGGGATGTGCTGGCGGAT...

+

CCCCCFFFFFCFGGGGGGFGGGHFHHHFHHGHHGHHHGGGGHHHGGHGHHHHHHGGGGF...

It is still a long name compared to the raw data. The part before the first “;”, SamnpleA\_1, is labels for Qiime, and the part after the first “;” is for USEARCH. They will not have conflict if you follow this pipeline, but you can also use -label to specify only one style of labels.

## 1.2 Merge FASTA Files:

***Merge All FASTQ File into One:***

@merge\_seqs.py -i Raw -o merged.fastq

***Count the Number of Total Sequences:***

@count\_seqs.py -i merged.fastq

# SECTION 2. Quality Filter

## 2.1 Quality Trim

@ java -jar trimmomatic-0.32.jar SE merged.fastq merged.trim.fastq ILLUMINACLIP:illumina\_MiSeq\_Fungi.fa:2:30:10 SLIDINGWINDOW:5:20 MINLEN:125

## 2.2 Ambiguous Base and Homopolyers

@ filter\_seqs.py -i merged.trim.fastq -o merged.trim.N0.homop9.fastq

## 2.3 Report on the Sequence File

@ stat\_seqs.py -i merged.trim.N0.homop9.fastq -o report.txt

## 2.4 Convert to FASTA File

@ convert\_fastq.py -i merged.trim.N0.homop9.fastq -o raw.qc.fasta

# SECTION 3. OTU CLUSTERING

## 3.1 Dereplication

@ dereplicate.py -i raw.qc.fasta -o raw.qc.derep -t 4

In general, USEARCH 8.0 has a faster dereplication function, but its free version has memory limitations (2 GB in Windows, 4 GB in Linux), and it does not output a Qiime style OTU map. A 2 GB file should be dereplicated in around 10 min using the script above. You should refrain from using other program since a large chunk of memory will be needed.

You will find two new files after dereplication:

raw.qc.derep.fasta contains only the unique sequences in raw.qc.fasta (like unique.seqs in mother).

raw.qc.derep.txt contains an OTU map that is compatible with Qiime.

## 3.2 Remove Singletons

@ filter\_otu\_map.py -i raw.qc.derep.txt -o raw.qc.derep.size2.txt -min\_size 2

@ pick\_seqs.py -i raw.qc.derep.fasta -o raw.qc.derep.size2.fasta -map raw.qc.derep.size2.txt

## 3.3 Cluster OTUs

***Add Size Annotation for USEARCH***

@ add\_seqs\_size.py -i raw.qc.derep.size2.fasta -map raw.qc.derep.size2.txt -o raw.qc.derep.size2.sizeout.fasta

***Cluster with USEARCH***

@ usearch -cluster\_otus raw.qc.derep.size2.fasta -uparseout uparse.up

***Convert Uparse Output to OTU Map***

@ parse\_uparse\_cluster.py -i uparse.up -o raw.qc.derep.size2.usearch.txt

***Merged Two OTU Maps***

@ merge\_otu\_maps.py -map\_large raw.qc.derep.size2.txt -map\_small raw.qc.derep.size2.usearch.txt -o merged.otu.txt

In general, any clustering method in Qiime can also be used here, as long as you remember to merge two OTU maps each time after clustering.

## 3.4 Make OTU Table

@ make\_otu\_table.py -i merged.otu.txt -o otu\_table.txt

## 3.5 Representative Sequence

Using Qiime (you need a Qiime virtual box under Windows: <http://qiime.org/install/virtual_box.html>)

@ rep\_seq.py -i merged.otu.txt -f raw.qc.fasta