20230816\_QIIME\_code\_and\_HPC\_primer\_(sorta)

2023-08-16

* This is a companion doc to the batchfile “qiime\_processing\_batch\_file.batch”, which contains the code to run a QIIME job on the HPC.
* I have run everything on the Norman servers, though a similar batch job can be run on the OUHSC Mustang server as well with some modifications (and after QIIME is installed to the server–I don’t have a tutorial for that, though the faculty (currently Phil O’Neil [Philip-ONeill@ouhsc.edu](mailto:Philip-ONeill@ouhsc.edu)) running the servers may be willing to globally install packages if you ask).
  + The OSCER servers in Norman operate on SGE, while the OUHSC Mustang servers use LSF.
    - More info can be found at: <https://www.med.upenn.edu/hpc/assets/user-content/documents/SGE_to_LSF_User_Migration_Guide.pdf>
* For our purposes, the only difference between the two is syntax. Let’s look at the batchfile header for SGE on the Norman OSCER servers, which uses qsub to submit jobs:

#!/bin/bash  
#SBATCH --ntasks=1  
#SBATCH --time=10:00:00  
#SBATCH --job-name=Assembly  
#SBATCH --mem=5G  
#SBATCH --output=Assembly\_%J\_stdout.txt  
#SBATCH --error=Assembly\_%J\_stderr.txt  
#SBATCH --mail-user=  
#SBATCH --mail-type=ALL  
#SBATCH --chdir=

* The commands in the header are fairly self-explanitory, so I won’t go into too many details here. But there are a few things you should be aware of, depending on the server/cluster/nodes you have access to
  + You need to include the first line of code (#!/bin/bash)
  + Make sure each line starts with “#”
  + “–ntasks” sets the number of computing *nodes* that will be used by the program. Each node contains several processors (which can also be coded into the batchfile, though it is not shown here). If you use more processors (and/or nodes) than one, you’ll likely need to make sure that the program’s code itself specifies this.
    - I’ll show a code snippet of this later
  + Time and mem specify how much time the run should take and how much RAM is needed for these computations, respectively. Each node/processor has a maximum amount of RAM it can pull from, so requesting as much as you need can be challenging if you aren’t sure of run parameters
    - Note that there’s nothing wrong, per se, with requesting more than you think you’ll need, just make sure you aren’t requesting more memory than is available on a given node; moreover, the more resources requested, the harder it can be to obtain server time
* You’ll need a few files for your run:
  + Raw sequencing files
  + File with sample ID and file path to forward and reverse reads (you have single rather than pair-end reads, I’m sure this code can still work for you with some modifications)
    - It’s named fmt\_filepath.txt
  + The base pair length of your sequencing region
    - Here it is around 275 bp
  + The primers used for sequencing
    - We used F: ACTCCTACGGGAGGCAGCAG | R: GGACTACHVGGGTWTCTAAT
* One other thing to note before we get into the code: I’m using the greengenes database for this run. It’s required to get gene annotation predictions form Picrust, but the database hasn’t been updated since 2013. If you want a newer database, you can use SILVA (<https://www.arb-silva.de/>)
  + If you use SILVA, you’ll need to look up how to download and process the database so that it is in a correct format for your needs; Google is your friend!
* The full batch file to run QIIME on raw 16S sequences looks like this:
  + See the comments in the code for more info on how to alter the run parameters

#!/bin/bash  
#SBATCH --ntasks=1  
#SBATCH --time=10:00:00  
#SBATCH --job-name=Assembly  
#SBATCH --mem=5G  
#SBATCH --output=Assembly\_%J\_stdout.txt  
#SBATCH --error=Assembly\_%J\_stderr.txt  
#SBATCH --mail-user=  
#SBATCH --mail-type=ALL  
#SBATCH --chdir=  
  
module load python  
module load QIIME2  
  
#fmt\_filepath.txt is in here. This basically is a tab-separated text file containing sample names in column 1, forward read file in column 2, and reverse read file in column 3  
#I usually pull the sequence names via the command line using ls to output my file. You can also assemble the file in excel (though I'd still use ls and copy/paste the file names to save some time)  
qiime tools import \  
 --type 'SampleData[PairedEndSequencesWithQuality]' \  
 --input-path fmt\_filepath.txt \  
 --output-path paired-end-demux.qza \  
 --input-format PairedEndFastqManifestPhred33V2  
  
qiime demux summarize \  
 --i-data paired-end-demux.qza \  
 --o-visualization demux-summary-1.qzv  
  
#you'll want to replace 5 and 300 with numbers appropriate for the length of your sequencing region  
qiime dada2 denoise-paired \  
 --p-trim-left-f 5 \  
 --p-trim-left-r 5 \  
 --p-trunc-len-f 300 \  
 --p-trunc-len-r 300 \  
 --i-demultiplexed-seqs paired-end-demux.qza \  
 --o-representative-sequences rep-seqs-1.qza \  
 --o-table table-1.qza \  
 --o-denoising-stats stats-1.qza  
  
  
qiime metadata tabulate \  
 --m-input-file stats-1.qza \  
 --o-visualization denoising-stats-1.qzv  
  
qiime feature-table summarize \  
 --i-table table-1.qza \  
 --o-visualization table.qzv \  
  
qiime feature-table tabulate-seqs \  
 --i-data rep-seqs-1.qza \  
 --o-visualization rep-seqs.qzv  
  
###  
#preparing the greengenes alignment file  
#if using SILVA, replace this code chunk with the one you wrote for using the SILVA database  
  
#download greengenes files  
wget ftp://greengenes.microbio.me/greengenes\_release/gg\_13\_5/gg\_13\_8\_otus.tar.gz  
tar -xzvf gg\_13\_8\_otus.tar.gz  
  
qiime tools import \  
 --type 'FeatureData[Sequence]' \  
 --input-path /scratch/suginoka/karen\_mice/gg\_13\_8\_otus/rep\_set/99\_otus.fasta \  
 --output-path 13\_8\_99\_otus.qza  
  
qiime tools import \  
 --type 'FeatureData[Taxonomy]' \  
 --input-format HeaderlessTSVTaxonomyFormat \  
 --input-path /scratch/suginoka/karen\_mice/gg\_13\_8\_otus/taxonomy/99\_otu\_taxonomy.txt \  
 --output-path 13\_8\_99\_ref-taxonomy.qza  
  
  
#make sure to change the f and r primers to reflect the primers used in your experiment!  
#the wide bp range is just to make sure the region isn't too small or too large, but lies within the expected range of our read lengths  
qiime feature-classifier extract-reads \  
 --i-sequences /scratch/suginoka/karen\_mice/13\_8\_99\_otus.qza \  
 --p-f-primer ACTCCTACGGGAGGCAGCAG \  
 --p-r-primer GGACTACHVGGGTWTCTAAT \  
 --p-min-length 100 \  
 --p-max-length 500 \  
 --o-reads ref-seqs.qza\  
 --verbose \  
 &> primer\_trimming.log   
  
  
qiime feature-classifier fit-classifier-naive-bayes \  
 --i-reference-reads ref-seqs.qza \  
 --i-reference-taxonomy /scratch/suginoka/karen\_mice/13\_8\_99\_ref-taxonomy.qza \  
 --o-classifier classifier.qza  
  
#End of greengenes database processing  
###  
  
# Classify rep seqs  
qiime feature-classifier classify-sklearn \  
--i-classifier classifier.qza \  
--i-reads rep-seqs-1.qza \  
--o-classification classified\_rep\_seqs.qza  
  
# Tabulate the features, their taxonomy and the confidence of taxonomy assignment  
qiime metadata tabulate \  
--m-input-file classified\_rep\_seqs.qza \  
--o-visualization classified\_rep\_seqs.qzv  
  
  
qiime tools export \  
 --input-path table-1.qza \  
 --output-path exported-otu-table  
   
qiime tools export \  
 --input-path classified\_rep\_seqs.qza \  
 --output-path exported-feature-table  
   
#in the resulting folder:  
cd exported-otu-table  
biom convert -i feature-table.biom -o table.from\_biom.txt --to-tsv

* That’s basically it! Using the HPC servers is a whole other thing that I don’t want to get into, but to run this job on the OSCER servers, you’d need to set up a few things:
  + Put all the sequences, batch job, and other files in a folder
  + Navigate to the folder with your data, and run:

qsub qiime\_processing\_batch\_file.batch

* This will put your job in the queue to run and return a job ID. To check on your job runs:

squeue -u $USER

* To cancel a run, replace *jobID* with the job ID given by either of the two steps above:

scancel \*jobID\*

* So, I have less experience with LSF on the OUHSC servers, but I can show how I’ve been submitting jobs:

#!/bin/bash  
#BSUB -J "mpa"  
#BSUB -o assembly\_%J\_stdout.txt  
#BSUB -e assembly\_%J\_stderr.txt  
#BSUB -n 4  
  
eval "$(conda shell.bash hook)"  
conda activate mpa  
  
metaphlan CHO57M.merged\_cat\_output.fastq\_u.fastq\_ec.fastq.fixed\_assembly.fastq\_sequence\_min1000.fastq --bowtie2out CHO57M.merged\_cat\_output.fastq\_u.fastq\_ec.fastq.fixed\_assembly.fastq\_sequence\_min1000.fastq\_metagenome.bowtie2.bz2 --input\_type fasta -x mpa\_vJan21\_CHOCOPhlAnSGB\_202103 --bowtie2db /storage01/home/ksugino/mpa\_db/ --nproc 4

* The header looks similar to OSCERs SGE setup, but I haven’t bothered to designate the time or memory needed by my programs; these are private servers for just the HHDC (at the moment at least) so there are fewer issues with memory and time allocations–they just use what’s needed
* Note that I designate 4 nodes (-n 4) and I subsequently designate the number of processors I’ve allocated in the code for metaphlan (–nproc 4)
  + I honestly don’t understand how the multithread processing works, but in short, it’s a way to put more resources towards a job so that it runs faster (though I’m sure it’s much deeper than that)