Pipeline for metagenomic data processing and analysis of cecum samples sequenced at 7 million reads. Inputs are raw fastq files, output include assembled MAGs and their resulting proteomes.

#This code was performed on terminal via UNM Center for Advanced Research Computing.

**#1.)Download raw fastq files onto supercomputer xena- scratch account, first I logged into xena-scratch, then uploaded the raw fastq’s**

#Upload rawfastq’s to xena-scratch (scp means secure copy)

scp -r /Users/vlab/Desktop/GUTS\ Fastq/MetaGenomic\ Fastqs\ E1\ vs\E2\Cecum\GUTS/raw\_fastq.tar [cmertz@xena.alliance.unm.edu](mailto:cmertz@xena.alliance.unm.edu):xena-scratch

#unzip the raw fastq’s -> note\* my files were returned as a .tar format, so our unzip.slurm script

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=16

#SBATCH --mem=60G

#SBATCH --job-name=unzip

#SBATCH --partition=singleGPU

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=[cmertz@unm.edu](mailto:cmertz@unm.edu)

cd $SLURM\_SUBMIT\_DIR

tar -xf raw\_fastq.tar

#submit job

sbatch unzip\_tar.slurm

#Now your tar files are in fastq.gz format in a new folder xena-scratch/fastq/fastq-raw

#now make a new slurm script to unzip the .gz and put your slurm script in the same folder as your fastq-raw

#unzip the fastq.gz file using slurmscript unzip.slurm:

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=16

#SBATCH --mem=60G

#SBATCH --job-name=unzip

#SBATCH --partition=singleGPU

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=[cmertz@unm.edu](mailto:cmertz@unm.edu)

cd $SLURM\_SUBMIT\_DIR

gunzip \*.gz

**#2.) Remove human genome contaminants using METAWRAP package general information.** [**https://github.com/bxlab/metaWRAP/blob/master/Usage\_tutorial.md**](https://github.com/bxlab/metaWRAP/blob/master/Usage_tutorial.md)

**(note: you only have to index the human genome once! We already did this, so no need to do it again, go ahead and skip to step 3 to generate QC report.)**

1. To download human genome onto your supercomputer go to this link, and find this Header: **Making host genome index for bmtagger**

https://github.com/bxlab/metaWRAP/blob/master/installation/database\_installation.md

1. To unzip the entire human genome will take more memory than xena can do so, we need to get more memory to and get on big memory node. To check out whats available on to us crom CARC go to:

<https://github.com/UNM-CARC/webinfo/blob/main/systems_information.md>

* lets use bimem-1TB

#to change nodes and get into bigmem1TB

srun -p bigmem-1TB --pty bash

1. Now, to open metawrap you have to use miniconda module

#load module minoconda3

module load miniconda3

#to load metawrap

source activate metawrap

1. Now follow tutorial line by line at **Making host genome index for bmtagger,**

First, lets download and merge the human genome hg38: (you have to do this line by line)

#make directory

mkdir BMTAGGER\_INDEX

#get into your new directory

cd BMTAGGER\_INDEX

#download genome using this cool hyperlink

wget ftp://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/\*fa.gz

#unzip genome files

gunzip \*fa.gz

#concatonate your files, take every file ending in .fa and combine into one file named hg38

cat \*fa > hg38.fa

# remove everything ending in chr(chomosomes) and ends in .fa

rm chr\*.fa

#Now lets index the human genome. Note that the file names of the indeces must be exactly as specified for metaWRAP to recognize them! Also note that indexing will take considerable memory and time (here I pass 100GB of RAM as a -M parameter).

bmtool -d hg38.fa -o hg38.bitmask

#we changed 100000 to 500000 because the –M is megabites and we ant to use as much megabites as we can and this node lets us!

srprism mkindex -i hg38.fa -o hg38.srprism -M 500000

#this command did not work, so we went into BMTAGGER\_INDEX directory (ls) to see what was going on. This was the output:

hg38.bitmask hg38.srprism.amp hg38.srprism.imp hg38.srprism.pmp hg38.srprism.ss hg38.srprism.ssd

hg38.fa hg38.srprism.idx hg38.srprism.map hg38.srprism.rmp hg38.srprism.ssa

#we deleted all of the srprism to just be left with hg38bitmask and hg38.fa

rm hg38.srprism.amp hg38.srprism.idx hg38.srprism.imp hg38.srprism.map hg38.srprism.pmp hg38.srprism.rmp hg38.srprism.ss hg38.srprism.ssa hg38.srprism.ssd

#Next in our BMTAGGER\_INDEX directory we wrote a slurmscript to index the human genome for us. Called it bmtagger.slurm

vim i

#make slurmscript

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=unzip

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

srprism mkindex -i hg38.fa -o hg38.srprism -M 600000

#to submit slurm script

sbatch bmtagger.slurm

#to see where your job is in queque

squeue

**#3.) Generate QC report for the raw data and clean it up “clean reads”. Use Metawrap for this. (You have several options- run as job through slurm script with one line of code, run the one line of code interactively, or you can do it in a slurm script written out one by one. I’ll show you all the options to try!**

[**https://github.com/bxlab/metaWRAP/blob/master/Usage\_tutorial.md**](https://github.com/bxlab/metaWRAP/blob/master/Usage_tutorial.md)

#Important note: it is very picky on your sample ID, it wont work if you have underscores, so we had to rename our samples like this:

**#rename them like this (old is on left, new name on right- just make sure you are in the correct directory (xena-scratch/ fastq/ fastq-raw**

mv E2\_D3\_2\_CEC\_R1.fastq E2D32CEC\_1.fastq

#Place the raw unzipped sequencing reads into a new folder xena-scratch/fastq/fastq-raw

#make a directory for the QC’s that trim the reads and remove human contamination called HUMAN\_READ\_QC

mkdir HUMAN\_READ\_QC

#Step 1: Run metaWRAP-Read\_qc to trim the reads and remove human contamination (in just one line of code in slurm script)

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=readQC

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=[cmertz@unm.edu](mailto:cmertz@unm.edu)

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

for F in fastq-raw/\*\_1.fastq; do R=${F%\_\*}\_2.fastq; BASE=${F##\*/}; SAMPLE=${BASE%\_\*}; metawrap read\_qc -1 $F -2 $R -t 64 -o HUMAN\_READ\_QC/$SAMPLE & done

*Escape*

:w human\_read\_qc.slurm

q!

sbatch human\_read\_qc.slurm

**# if this works yay! If this doesn’t work boo- but try again interactively, not through slurm. So, log onto xena-scratch. Then cd into the correct dir- “fastq”**

srun -p bigmem-1TB --pty bash

#Now, to open metawrap you have to use miniconda module

#load module minoconda3

module load miniconda3

#to load metawrap

source activate metawrap

#now run this single line of code

for F in fastq-raw/\*\_1.fastq; do R=${F%\_\*}\_2.fastq; BASE=${F##\*/}; SAMPLE=${BASE%\_\*}; metawrap read\_qc -1 $F -2 $R -t 64 -o HUMAN\_READ\_QC/$SAMPLE & done

**#if that works yay! If not try to do it one at a time and submit a slurmscript like this: (this is what ended up working for me and David Robinson)**

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=readQC

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap read\_qc -1 fastq-raw/E1D11CEC\_1.fastq -2 fastq-raw/E1D11CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D11CEC

metawrap read\_qc -1 fastq-raw/E1D49CEC\_1.fastq -2 fastq-raw/E1D49CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D49CEC

metawrap read\_qc -1 fastq-raw/E1D13CEC\_1.fastq -2 fastq-raw/E1D13CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D13CEC

metawrap read\_qc -1 fastq-raw/E2D24CEC\_1.fastq -2 fastq-raw/E2D24CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D24CEC

metawrap read\_qc -1 fastq-raw/E1D2A11CEC\_1.fastq -2 fastq-raw/E1D2A11CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D2A11CEC

metawrap read\_qc -1 fastq-raw/E2D114CEC\_1.fastq -2 fastq-raw/E2D114CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D114CEC

metawrap read\_qc -1 fastq-raw/E1D2A12CEC\_1.fastq -2 fastq-raw/E1D2A12CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D2A12CEC

metawrap read\_qc -1 fastq-raw/E1D47CEC\_1.fastq -2 fastq-raw/E1D47CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D47CEC

metawrap read\_qc -1 fastq-raw/E2D14CEC\_1.fastq -2 fastq-raw/E2D14CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D14CEC

metawrap read\_qc -1 fastq-raw/E2D115CEC\_1.fastq -2 fastq-raw/E2D115CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D115CEC

metawrap read\_qc -1 fastq-raw/E1D2A7CEC\_1.fastq -2 fastq-raw/E1D2A7CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D2A7CEC

metawrap read\_qc -1 fastq-raw/E2D28CEC\_1.fastq -2 fastq-raw/E2D28CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D28CEC

metawrap read\_qc -1 fastq-raw/E2D32CEC\_1.fastq -2 fastq-raw/E2D32CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D32CEC

metawrap read\_qc -1 fastq-raw/E1D48CEC\_1.fastq -2 fastq-raw/E1D48CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E1D48CEC

metawrap read\_qc -1 fastq-raw/E2D210CEC\_1.fastq -2 fastq-raw/E2D210CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D210CEC

metawrap read\_qc -1 fastq-raw/E2D37CEC\_1.fastq -2 fastq-raw/E2D37CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D37CEC

metawrap read\_qc -1 fastq-raw/E2D36CEC\_1.fastq -2 fastq-raw/E2D36CEC\_2.fastq -t 64 -o HUMAN\_READ\_QC/E2D36CEC

*Escape*

:w human\_read\_qc.slurm

q!

sbatch human\_read\_qc.slurm

**#to check on progress:** slurm-347026.out

**cat** slurm-347026.out

#check out QC reports and save them to your own computer.

sftp [cmertz@xena.alliance.unm.edu](mailto:cmertz@xena.alliance.unm.edu)

sftp> cd xena-scratch

sftp> ls

sftp> cd fastq

sftp> ls

sftp> get -r HUMAN\_READ\_QC

**#we are in E1D1CEC folder:**

**#How to see how many reads were taken out**

**Less host\_reads\_1.fastq**

**Grep -o ‘@’ host\_reads\_1.fastq | wc -1**

**#tells us count**

**Less final\_pure\_reads\_1.fastq**

**Grep -o ‘@’ final\_pure\_reads\_1.fastq | wc -1**

**#4.) Remove Host (Deer Mouse) genome from our clean no human reads: use metawrpas tutorial if you get stuck https://github.com/bxlab/metaWRAP/blob/master/installation/database\_installation.md**

1. download the Peromyscus maniculatus genome from NCBI <https://www.ncbi.nlm.nih.gov/data-hub/genome/GCF_003704035.1/> one with the green check mark- “HU\_Pman\_2.1.3”. Then download this genome to xena account because we need to remove this mouse material from our dataset.

#download peromyscus genome and put it on xena scratch

scp -r /Users/connermertz/Dropbox\ \(Personal\)/Mac/Downloads/Peromyscus\_genome\_ncbi.zip cmertz@xena.alliance.unm.edu:xena-scratch

#make directory for genome

Peromyscus\_genome

#move genome into new directory

mv Peromyscus\_genome\_ncbi.zip Peromyscus\_genome

#get into bigmem

srun -p bigmem-1TB --pty bash

#unzip the musmus genome

unzip MusMus\_genome\_ncbi\_dataset.zip

#look at it

ls

#you will have to change directories a few times to find it

cd ncbi\_dataset

#look at it

ls

#change directory again

cd data

#look at it

ls

#now you are given two options: GCF\_003704035.1 or GCA\_003704035.3 these are simply different reference databases one from GenBank and one from Refseq. We chose the one from Refseq because it seemed more up to date.

cd GCF\_003704035.1

#when we ls we get a .fna file that looks like this: “GCF\_000001635.27\_GRCm39\_genomic.fna”

#create slurmscript to index the unzipped genome, it is named musmusconfig.slurm, here we pasted the .fna file after the –d and after the –i. We also named our .bitmask file and .srprism file musmus

#make new directory for your slurm script and unzipped genome file GCF\_003704035.1, I moved them both into a directory called peromyscus\_config

vim

i

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=peromyscus\_config

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

bmtool -d GCF\_003704035.1\_HU\_Pman\_2.1.3\_genomic.fna -o peromyscus.bitmask srprism mkindex -i GCF\_003704035.1\_HU\_Pman\_2.1.3\_genomic.fna -o peromyscus.srprism -M 800000

*Escape*

:w peromyscus\_config.slurm

:q!

#submit job

sbatch peromyscus\_config.slurm

#to check on job

ls

#find the slurm job

cat slurm-348561.out

#once that is done move into BMTAGGER dir in xena-scratch using mv or cp. I used cp to just copy it to a new location.

cp peromyscus.srprism.amp /users/cmertz/xena-scratch/BMTAGGER\_INDEX

#remove peromyscus from no human clean reads

1. use human\_read\_qc.slurm fastq-raw change to HUMAN\_CLEAN\_READS\_WITH\_PERO, name output NO\_HUMAN\_NO\_PERO\_QC and add -x peromyscus to the end of each line

peromyscus\_readqc.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=readQC\_nohuman\_noPeromyscus

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D11CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D11CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D11CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D49CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D49CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D49CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D13CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D13CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D13CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D24CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D24CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D24CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A11CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A11CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D2A11CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D114CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D114CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D114CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A12CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A12CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D2A12CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D47CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D47CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D47CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D14CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D14CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D14CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D115CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D115CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D115CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A7CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D2A7CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D2A7CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D28CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D28CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D28CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D32CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D32CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D32CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D48CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E1D48CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E1D48CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D210CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D210CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D210CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D37CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D37CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D37CEC -x peromyscus

metawrap read\_qc -1 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D36CEC\_1.fastq -2 HUMAN\_CLEAN\_READS\_WITH\_PERO/E2D36CEC\_2.fastq -t 64 -o NO\_HUMAN\_NO\_PERO\_QC/E2D36CEC -x peromyscus

#These are the final trimmed and de-contaminated reads:

final\_pure\_reads\_1.fastq

final\_pure\_reads\_2.fastq

#Move over the final QC'ed reads into a new folder

mkdir FINAL\_CLEAN\_READS

for i in NO\_HUMAN\_NO\_PERO\_QC/\*; do

b=${i#\*/}

mv ${i}/final\_pure\_reads\_1.fastq FINAL\_CLEAN\_READS/${b}\_1.fastq

mv ${i}/final\_pure\_reads\_2.fastq FINAL\_CLEAN\_READS/${b}\_2.fastq

done

1. now cd into your FINAL\_CLEAN\_READS directory and concatenate with cat command

cat \*\_1.fastq > ALL\_READS\_1.fastq

#now cross assemble the reverse reads but first concatenate the reverse reads together

cat \*\_2.fastq > ALL\_READS\_2.fastq

**#now assemble the reads!**

#https://github.com/bxlab/metaWRAP/blob/master/Usage\_tutorial.md

#metawrap tutorial: usage\_tutorial

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_ASSEMBLY

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap assembly -1 FINAL\_CLEAN\_READS/ALL\_READS\_1.fastq -2 FINAL\_CLEAN\_READS/ALL\_READS\_2.fastq -m 800 -t 64 --metaspades -o PERO\_ASSEMBLY

\*just make sure the ALL\_READS\_1.fastq and ALL\_READS\_2.fastq are in the FINAL\_CLEAN\_READS folder!!!

#check to see if job completed

- once the assembly is complete check if the job actually completrd. If it didn’t complete, submit the same slurmscript again (don’t change anything, simply re submit it)

- if completed all the way, look at the job output cat 353656.out

- this gives you a step by step of what occurred. The Quast is a quality assessment tool, and gives a QC report.

To see the QC report you will need to sftp into your folder.

sftp [cmertz@xena.alliance.unm.edu](mailto:cmertz@xena.alliance.unm.edu)

get assembly\_report.html

1. **5.) STEP 4 FROM TUTORIAL BIN THE CO-ASSEMBLY WITH THREE DIFFERENT ALGORITHMS WITH THE BINNING.** WE WILL BIN 3 times, first with metabat2, then with maxbin2, and then concoct. A quick note here: FINAL\_CLEAN\_READS/\*fastq this needs to have your cleaned reads- but those reads are not concatenated with cat command!

Here is the slurm script:

[cmertz@xena fastq]$ cat PERO\_metabat\_BIN.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_metabat\_BIN

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap binning -o INITIAL\_BINNING -t 64 -a ASSEMBLY/final\_assembly.fasta --metabat2 FINAL\_CLEAN\_READS/\*fastq

#next step is to repeat the same script but with maxbin2

Here is the slurm script:

[cmertz@xena fastq]$ cat PERO\_metabat\_BIN.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_maxbin2\_BIN

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap binning -o INITIAL\_BINNING -t 64 -a ASSEMBLY/final\_assembly.fasta --maxbin2 FINAL\_CLEAN\_READS/\*fastq

#next step is to repeat the same script but with concoct (note: this one is notorious to not work. I am trying it for fun but will move on if it doesn’t work.)

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_concoct\_BIN

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap binning -o INITIAL\_BINNING -t 64 -a ASSEMBLY/final\_assembly.fasta --concoct FINAL\_CLEAN\_READS/\*fastq

\*\*\*\*\*IMPORTANT\*\*\*\*\*\*\* delete the slurm output file that was made with concoct because its gigantic and will crash your account! So we deleted slurm-354388.out

Rm slurm-354388.out

**#6.) Consolidate bin sets with the Bin\_refinement module:**

* Note: make sure you downloaded the CheckM database (see metaWRAP database instrucitons)
* Now what you have metabat2, maxbin2, and concoct bins, lets consolidate them into a single, stronger bin set! If you used your own binning software, feel free to use any 3 bin sets. If you have more than 3, you can run them in groups. For example if you have 5 bin sets, try consolidating 1+2+3 and 4+5, and then consolidate again between the outputs.
* When you do your refinement, make sure to put some thought into the minimum compleiton (-c) and maximum contamination (-x) parameters that you enter. During refinement, metaWRAP will have to chose the best version of each bin between 7 different versions of each bin. It will dynamically adjust to prioritize the bin quality that you desire. Consider this example: bin\_123 comes in four versions in terms of completion/contamination: 95/15, 90/10, 80/5, 70/5. Which one is the best version? The high completion but high contamination, or the less complete but more pure bin? This is subjective and depends on what you value in a bin and on what your purposes for bin extraction are.
* By default, the minimum completion if 70%, and maximum contamination is 5%. However, because of the relatively poor depth of these demonstration samples, we will set minimum completion to 50% and maximum contamination to 10%, but feel free to be much more picky. Parameters like -c 90 -x 5 are not unreasonable in some data (but you will get fewer bins, of course).

#Run metaWRAP's Bin\_refinement module: but first you need to [install checkM](https://github.com/bxlab/metaWRAP/blob/master/installation/database_installation.md) (only do this once):

mkdir MY\_CHECKM\_FOLDER

# Now manually download the database: do this one line of code at a time!

cd MY\_CHECKM\_FOLDER

wget https://data.ace.uq.edu.au/public/CheckM\_databases/checkm\_data\_2015\_01\_16.tar.gz

tar -xvf \*.tar.gz

rm \*.gz

cd ../

# Now you need to tell CheckM where to find this data before running anything:

checkm data setRoot # CheckM will prompt to to chose your storage location

# On newer versions of CheckM, you would run:

checkm data setRoot /users/cmertz/xena-scratch/MY\_CHECKM\_FOLDER

#If it asks you for a location just put /users/cmertz/xena-scratch/MY\_CHECKM\_FOLDER in again and press enter.

**#Run metaWRAP's Bin\_refinement module:**

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=Bin\_refinement

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap bin\_refinement -o BIN\_REFINEMENT -t 64 -A INITIAL\_BINNING/metabat2\_bins/ -B INITIAL\_BINNING/maxbin2\_bins/ -C INITIAL\_BINNING/concoct\_bins/ -c 50 -x 10

\*special note: David and I used parameters for the completion to be greater than 50% and the contamination to be less than 10% to make a medium-quality MAG because (The Genome Standards Consortium et al., 2017). This ensures CHECKM can still perform quality control, but we wont lose all of our bins.

* to check out the Check M output look at the slurm out file. I also saved this. We need to report these stats in the paper. Or you can check out the stats by going into:

cd BIN\_REFINEMENT

ls

cat metawrap\_50\_10\_bins.stats

**#7.) Quantify the bins Step 7: Find the abundances of the draft genomes (bins) across the samples. https://github.com/bxlab/metaWRAP/blob/master/Usage\_tutorial.md**

* We would like to know how the extracted genomes are distributed across the samples, and in what abundances each bin is present in each sample. The Quant\_bin module can give us this information. It used Salmon - a tool conventionally used for transcript quantitation - to estimate the abundance of each scaffold in each sample, and then computes the average bin abundances.
* NOTE: In order to run this module, it is highly recomended to use the non-reassembled bins (the bins produced by the Bin\_Refinment module) and provide the entire non-binned assembly with the -a option. This will give more accurate bin abundances that are in context of the entire community.

Lets run the Quant\_bins module:

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_quant\_bins

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate metawrap

metawrap quant\_bins -b BIN\_REFINEMENT/metawrap\_50\_10\_bins -o QUANT\_BINS -a ASSEMBLY/final\_assembly.fasta FINAL\_CLEAN\_READS/\*fastq

cat the slurm output to make sure it worked and…..

CONGRATULATIONS! You are done with Metawrap tutorial

**#8.) Now we are going to use CAT to assign taxonomy-**

[cmertz@xena fastq]$ cat PERO\_CAT\_bins.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=PERO\_CAT\_bins

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate CAT

CAT bins -b BIN\_REFINEMENT/metawrap\_50\_10\_bins -d /users/cmertz/xena-scratch/CAT\_prepare\_20210107/2021-01-07\_CAT\_database -t /users/cmertz/xena-scratch/CAT\_prepare\_20210107/2021-01-07\_taxonomy -n 64 -s .fa

-check out your output in QUANT\_BINS folder:

Sftp and get

get bin\_abundance\_table.tab

\*note- (rename .tab as txt so you can open it)these are in per million reads!!!

**#9.) CAT BAT classification- Assign Taxomomy** : cat CAT\_ADD\_NAMES\_PERO.slurm

<https://github.com/dutilh/CAT>

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=RNA\_CAT\_classification

#SBATCH --partition=bigmem-1TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate CAT

CAT add\_names -i out.BAT.bin2classification.txt -o METAGENOMIC\_TAXONOMY\_classification.txt -t /users/cmertz/xena-scratch/CAT\_prepare\_20210107/2021-01-07\_taxonomy --only\_official --exclude\_scores

#check the output out in this file:

Named: METAGENOMIC\_TAXONOMY\_classification.txt

#get DRAM data folder from Henrietta onto xena (only need to do this once)

Rsync -aP ‘media/vlab/drive2/ALPS\_Metagenomics/POLAR\_LAKES\_META/DRAM\_data’ cmertz@xena.unm.edu:xena-scratch

Special note- if connection gets lost or broken pipe, then all you have to do is type in the same line of code into terminal (up tab) then press enter and it will resume.

**#10.) Use prodigal to translate DNA to protein**

<https://github.com/hyattpd/Prodigal>

#install prodigal onto xena-scratch (only do this once)

$ make install

$ prodigal -h

#Make a new working directory for metagenomics and move your bins ONE BIN AT A TIME! (from /BIN\_refinement/) into the new prodigal working directory

mkdir prodigal\_metagenomics

cd BIN\_REFINEMENT

ls

cp bin.10.fa /users/cmertz/xena-scratch/prodigal\_metagenomics

#now translate your bins into protein sequences one bin at a time. Be sure to name your output file to include the bin name you are working with, the first one was named: bin.10.protein.translations.faa

#here is the script for the first bin:

[cmertz@xena prodigal\_metagenomics]$ cat prodigal.bin10.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=prodigal.bin10

#SBATCH --partition=bigmem-3TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate prodigal

prodigal -i bin.10.fa -o gene.coords.gbk -a bin.10.protein.translations.faa

*Escape*

:w prodigal.bin10.slurm

:q!

#submit slurmscript!

sbatch prodigal.bin10.slurm

#download translation onto computer to upload onto GapMind for Amino acid biosynthesis

Sftp> get bin.10.protein.translations.faa

#now to do all bins at the same time use this!

[cmertz@xena prodigal\_metagenomics]$ cat prodigal\_allbins.slurm

#!/usr/bin/bash

#SBATCH --time=48:00:00

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=64

#SBATCH --mem=800G

#SBATCH --job-name=prodigal\_allbins

#SBATCH --partition=bigmem-3TB

#SBATCH --mail-type=BEGIN,END,FAIL

#SBATCH --mail-user=cmertz@unm.edu

cd $SLURM\_SUBMIT\_DIR

module load miniconda3

source activate prodigal

prodigal -i bin.1.fa -o gene.coords.gbk -a bin.1.protein.translations.faa

prodigal -i bin.2.fa -o gene.coords.gbk -a bin.2.protein.translations.faa

prodigal -i bin.3.fa -o gene.coords.gbk -a bin.3.protein.translations.faa

prodigal -i bin.4.fa -o gene.coords.gbk -a bin.4.protein.translations.faa

prodigal -i bin.5.fa -o gene.coords.gbk -a bin.5.protein.translations.faa

prodigal -i bin.6.fa -o gene.coords.gbk -a bin.6.protein.translations.faa

prodigal -i bin.7.fa -o gene.coords.gbk -a bin.7.protein.translations.faa

prodigal -i bin.8.fa -o gene.coords.gbk -a bin.8.protein.translations.faa

prodigal -i bin.9.fa -o gene.coords.gbk -a bin.9.protein.translations.faa

prodigal -i bin.10.fa -o gene.coords.gbk -a bin.10.protein.translations.faa

prodigal -i bin.11.fa -o gene.coords.gbk -a bin.11.protein.translations.faa

prodigal -i bin.12.fa -o gene.coords.gbk -a bin.12.protein.translations.faa

prodigal -i bin.13.fa -o gene.coords.gbk -a bin.13.protein.translations.faa

prodigal -i bin.14.fa -o gene.coords.gbk -a bin.14.protein.translations.faa

prodigal -i bin.15.fa -o gene.coords.gbk -a bin.15.protein.translations.faa

prodigal -i bin.16.fa -o gene.coords.gbk -a bin.16.protein.translations.faa

prodigal -i bin.17.fa -o gene.coords.gbk -a bin.17.protein.translations.faa

prodigal -i bin.18.fa -o gene.coords.gbk -a bin.18.protein.translations.faa

prodigal -i bin.19.fa -o gene.coords.gbk -a bin.19.protein.translations.faa

prodigal -i bin.20.fa -o gene.coords.gbk -a bin.20.protein.translations.faa

prodigal -i bin.21.fa -o gene.coords.gbk -a bin.21.protein.translations.faa

prodigal -i bin.22.fa -o gene.coords.gbk -a bin.22.protein.translations.faa

prodigal -i bin.23.fa -o gene.coords.gbk -a bin.23.protein.translations.faa

prodigal -i bin.24.fa -o gene.coords.gbk -a bin.24.protein.translations.faa

prodigal -i bin.25.fa -o gene.coords.gbk -a bin.25.protein.translations.faa

prodigal -i bin.26.fa -o gene.coords.gbk -a bin.26.protein.translations.faa

prodigal -i bin.27.fa -o gene.coords.gbk -a bin.27.protein.translations.faa

prodigal -i bin.28.fa -o gene.coords.gbk -a bin.28.protein.translations.faa

prodigal -i bin.29.fa -o gene.coords.gbk -a bin.29.protein.translations.faa

prodigal -i bin.30.fa -o gene.coords.gbk -a bin.30.protein.translations.faa

prodigal -i bin.31.fa -o gene.coords.gbk -a bin.31.protein.translations.faa

prodigal -i bin.32.fa -o gene.coords.gbk -a bin.32.protein.translations.faa

prodigal -i bin.33.fa -o gene.coords.gbk -a bin.33.protein.translations.faa

prodigal -i bin.34.fa -o gene.coords.gbk -a bin.34.protein.translations.faa

prodigal -i bin.35.fa -o gene.coords.gbk -a bin.35.protein.translations.faa

prodigal -i bin.36.fa -o gene.coords.gbk -a bin.36.protein.translations.faa

prodigal -i bin.37.fa -o gene.coords.gbk -a bin.37.protein.translations.faa

prodigal -i bin.38.fa -o gene.coords.gbk -a bin.38.protein.translations.faa

prodigal -i bin.39.fa -o gene.coords.gbk -a bin.39.protein.translations.faa

prodigal -i bin.40.fa -o gene.coords.gbk -a bin.40.protein.translations.faa

prodigal -i bin.41.fa -o gene.coords.gbk -a bin.41.protein.translations.faa

prodigal -i bin.42.fa -o gene.coords.gbk -a bin.42.protein.translations.faa

prodigal -i bin.43.fa -o gene.coords.gbk -a bin.43.protein.translations.faa

prodigal -i bin.44.fa -o gene.coords.gbk -a bin.44.protein.translations.faa

prodigal -i bin.45.fa -o gene.coords.gbk -a bin.45.protein.translations.faa

prodigal -i bin.46.fa -o gene.coords.gbk -a bin.46.protein.translations.faa

prodigal -i bin.47.fa -o gene.coords.gbk -a bin.47.protein.translations.faa

prodigal -i bin.48.fa -o gene.coords.gbk -a bin.48.protein.translations.faa

prodigal -i bin.49.fa -o gene.coords.gbk -a bin.49.protein.translations.faa

**#11.) Find genes of interest using- GapMind for Amino acid biosynthesis**

<https://papers.genomics.lbl.gov/cgi-bin/gapView.cgi>

#Upload your proteome in fasta format (.faa). Remember it only allows 100,000 sequences at a time

#we did this one bin at a time so for bin 1 I uploaded prodigal output: bin.1.protein.translations.faa

#scroll to the downloads portion of the output on the webpage. You want to download the Candidates link. For a mac right click-> save link as-> change .cand to .txt -> save and then open in excel.

#Download the candidates file as an excel or csv

Bin1: https://papers.genomics.lbl.gov/cgi-bin/gapView.cgi

8.) Quantify with Kallisto

#log into xena scratch

#load miniconda3

module load miniconda3

#make conda environment

conda create -n kallisto

#open environment

source activate kallisto

#install kallisto

conda install -c bioconda kallisto

#make index of fasta file of the assembly

#get onto big mem

srun -p bigmem-1TB –-pty bash

#load miniconda3

module load miniconda3

#open environment

source activate kallisto

#Put your assembled transcript and clean Fastq files into the same folder. I put them in the directory CLEAN\_READS\_BEFORE\_RIBO\_REMOVAL

#get into that environment and folder so your “handle” should look like this:

(kallisto) [cmertz@xena-bigmem02 CLEAN\_READS\_BEFORE\_RIBO\_REMOVAL]

#Now that we put our assembled transcripts into the right directory, run this code to index your transcript file transcripts.fasta and name it kalistoindextranscripts.idx

kallistco index -i kalistoindextranscripts.idx transcripts.fasta

#quantify one by one

kallisto quant -i kalistoindextranscripts.idx -t 60 -o 2852quant 2852\_1.fastq 2852\_2.fastq

#repeat for each fasta file

9.) lets export our kalisto results onto our computer!!

#get into your

sftp cmertz@xena.alliance.unm.edu

#find your kalistco files

sftp> cd xena-scratch

ls

cd CLEAN\_READS\_BEFORE\_RIBO\_REMOVAL

ls

#get your kalistco file named “2856quant” onto your desktop

sftp> get -r 2856quant

#8.) Make files compatible with phyloseq, you will need to create a column that keeps the order of things 1-your number of contigs

- be careful, your kalistco files eg. “2856quant” will be too large to manipulate in excel so you will need to format them in R.