**Workflow for analyzing swine fecal metagenomic samples**

Current as of March 9, 2023.

This workflow assumes that all the necessary mamba (or conda) environments have been installed and is being run on the Biocluster. All of the packages used below can be found on Bioconda with installation instructions there.

<https://bioconda.github.io/>

**Downloading files from the SRA using the SRA Toolkit.**

<https://github.com/ncbi/sra-tools>

SRA Toolkit conda package: <https://bioconda.github.io/recipes/sra-tools/README.html>

Here is the link to all of our weaning pig metagenomic samples on the SRA.

https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA629856

Use the filter on the left hand side of the page to find the metagenomic samples. You can filter by “Instrument” (choose NovaSeq 6000 to display these samples. You can also filter by “Host\_Age” as well. Pick two ages, e.g. 14 d and 70 d. That way there will be an even split between pre- and post-weaning samples. Then filter by “Sex” and choose only the males or females. That should give you about 25 or 26 samples. Then selection those samples in the display in the middle of the page.

Finally hit the “Accession List” button in the “Selected” row. Save that file and upload it into the Biocluster folder where you plan to do these analyses.

https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA629856

Then on the Biocluster use the SRA Toolkit to download those samples you selected in the SRR\_Acc\_List.txt file.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate sra\_tools

prefetch --option-file SRR\_Acc\_List.txt

This will generate a folder for each sample containing an .sra file. Use the fasterq-dump function in the SRA Toolkit to get the fastq files from these .sra files.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate sra\_tools

fasterq-dump SRR\*/SRR\*

Now you should have 25 or 26 raw paired-end fastq files to use for the rest of this workflow.

**QUALITY-FILTERING AND TRIMMING**

Use **fastp** to remove low quality reads, adapters, and produce read quality statistics. This script will trim all the fastq.gz in the folder using a for loop. Note: this script assumes that the paired-end files end in \_1.fastq.gz and \_2.fastq.gz. If your files are different (e.g. R1\_fastq.gz) then adjust accordingly.

This script removes the first three and last three nucleotides if they are below a quality score threshold of 15. Reads are also being trimmed over a sliding window of 4 nucleotides when the average quality score is below 15. Any read that is less than 100 bases is also removed.

<https://github.com/OpenGene/fastp>

fastp conda package: <https://bioconda.github.io/recipes/fastp/README.html>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate fastp

fastp -v

mkdir paired\_trimmed

mkdir unpaired\_trimmed

for i in \*\_1.fastq.gz

do

prefix=$(basename $i \_1.fastq.gz)

j=${prefix}\_2.fastq.gz

fastp -i $i -I $j -o paired\_trimmed/${prefix}\_trimmed\_R1.fastq.gz \

-O paired\_trimmed/${prefix}\_trimmed\_R2.fastq.gz \

--unpaired1 unpaired\_trimmed/${prefix}\_unpaired\_R1.fastq.gz \

--unpaired2 unpaired\_trimmed/${prefix}\_unpaired\_R2.fastq.gz \

--length\_required 100 \

--cut\_tail --cut\_front --cut\_mean\_quality 15 \

--qualified\_quality\_phred 15 cut\_window\_size 4 \

--detect\_adapter\_for\_pe --report\_title="${prefix}" \

--html=${prefix}.html --json=${prefix}.json --thread=40;done

**REMOVING HOST AND PHIX SEQUENCES**

For host-associated samples there is often contamination from the host which varies depending on the sample type and microbial biomass to host cell ratio.

It is also a good idea to check for PhiX sequences in your metagenomic files (Illumina). The two steps can be combined.

Use Bowtie2 to align your *paired\_trimmed* reads against the pig genome (**Sscrofa11.1**) Bowtie2 requires that you first build an index of the fasta file you want to align your reads against. You can download the latest pig reference genome here:

<https://www.ncbi.nlm.nih.gov/genome/?term=sus+scrofa> and choose *Download sequences in FASTA format for genome*

wget <https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/003/025/GCF_000003025.6_Sscrofa11.1/GCF_000003025.6_Sscrofa11.1_genomic.fna.gz>

Then unzip the file.

gunzip GCF\_000003025.6\_Sscrofa11.1\_genomic.fna.gz

Download the PhiX genome:

wget <ftp://ftp.ncbi.nlm.nih.gov/genomes/Viruses/enterobacteria_phage_phix174_sensu_lato_uid14015/NC_001422.fna>

Combine the pig and PhiX genomes:

cat NC\_001422.fna GCF\_000003025.6\_Sscrofa11.1\_genomic.fna > Pig\_PhiX\_genomes.fna

Bowtie2 conda package: <https://bioconda.github.io/recipes/bowtie2/README.html>

To create a Bowtie2 index use the following:

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bowtie2

mkdir pig

bowtie2-build Pig\_PhiX\_genomes.fna pig/pig

**Mapping the paired-end reads to the indexed pig genome:**

Then run Bowtie2 from the *paired\_trimmed* directory. \*make sure that the full path of the indexed pig genome is indicated as highlighted below.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

# ENTER COMMANDS HERE:

mamba activate bowtie2

mkdir bowtie2\_output

for i in \*\_trimmed\_R1.fastq.gz

do

prefix=$(basename $i \_trimmed\_R1.fastq.gz)

r2=${prefix}\_trimmed\_R2.fastq.gz

bowtie2 -x /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/pig/pig

-1 $i -2 $r2

-S ${prefix}.sam

-p 40;

done

#End of file

The output of this script is a SAM (sequence alignment map) file for each sample. You will need to extract from this file the reads that didn’t align to the pig genome. We use a program called SAMtools to convert the SAM files into BAM (binary alignment map) files with only the unmapped reads. Note: the SAMtools script below can be changed to retain only those reads that aligned by altering the flags.

Conda package: <https://bioconda.github.io/recipes/samtools/README.html>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate samtools

mkdir unmapped\_BAM\_files

for i in \*.sam

do

prefix=$(basename $i .sam)

samtools view -@ 20 -bS -f 12 -F 256 $i > unmapped\_BAM\_files/${prefix}\_unmapped.bam; done

done

We have to sort the sequences in the BAM files so that the next program (BEDTools) can output the paired fastq files.

Run this script out of the unmapped\_BAM\_files folder.

cd unmapped\_BAM\_files

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate samtools

mkdir Sorted\_BAM

for i in \*.bam

do

prefix=$(basename $i .bam)

samtools sort -@ 20 -n $i > Sorted\_BAM/${prefix}.sorted.bam; done

**Now you can use BEDtools to output the paired fastq reads that did not align to the pig genome.**

Conda package: <https://bioconda.github.io/recipes/bedtools/README.html>

Run this script out of the sorted\_bam folder. By now we are a few directories into our main project directory so it would be easier to have the files closer to the main project directory.

This script will make a directory in “Holman\_et\_al\_2021” called “final\_paired” and the paired-end fastq files from the BAM file will be output here. Remember to change the highlighted file path for match your directory.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate bedtools

mkdir /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/Holman\_et\_al\_2021/final\_paired

for i in \*.bam

do

prefix=$(basename $i \_unmapped.sorted.bam)

bedtools bamtofastq -i $i \

-fq /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/Holman\_et\_al\_2021/final\_paired/${prefix}\_R1.fastq \

-fq2 /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/Holman\_et\_al\_2021/final\_paired/${prefix}\_R2.fastq; done

Within the “final\_paired” folders are your paired reads in fastq format that did not align to the pig genome and that you will use for the rest of the analyses.

Because there are often lots of fastq files generated after the previous BEDtools step it is usually a good idea to zip the fastq files to save space. A program called pigz is a parallel implementation of gzip and can zip hundreds of files relatively quickly.

To install with mamba run the following after setting up a new mamba environment called pigz

mamba install -c conda-forge pigz

Then run :

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate pigz

pigz \*fastq -p 40

**TAXONOMY**

One of the more useful programs for assigning taxonomy to the reads is “Kaiju”. This program translates the reads into their amino acid sequences first and then using an k-mer approach to assign taxonomy.

<https://github.com/bioinformatics-centre/kaiju>

Conda package: <https://bioconda.github.io/recipes/kaiju/README.html>

\*\*note as of Nov 10, 2022, there is an issue on the Biocluster with the kaiju-makedb script which prevents the nr.gz file from being downloaded completely from NCBI. However, it is possible to download the nr.gz file from NCBI manually first and then run the kaiju-makedb command.

First make a directory for the Kaiju database to be downloaded to.

mkdir kaijudb

cd kaijudb

Then make another directory for the nr.gz file to be downloaded to:

mkdir nr\_euk

cd nr\_euk

Then download the nr.gz file. This is a subset of the NCBI BLAST nr database containing all proteins belonging to Archaea, Bacteria and Viruses.

wget <ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nr.gz>

and the protein accessions:

wget https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/prot.accession2taxid.gz

Then run go up one directory to kaijudb and download the taxonomy file:

cd ..

wget <https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate kaiju

kaiju-makedb --no-download -s nr\_euk -t 40

Only the *kaiju\_db\_\*.fmi, nodes.dmp*, and *names.dmp* are needed to taxonomy assignment. These files have to be in your working directory or you need to indicate their full path name where they can be located.

In this example the files are all in the “final\_paired” folder. “z” is the number of CPUs to use and “v” tells it to be verbose in its output as it runs.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mkdir Kaiju\_output

mamba activate kaiju

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

kaiju -t nodes.dmp \

-f kaiju\_db\_nr\_euk.fmi \

-i $i -j $j \

-v -o Kaiju\_output/${prefix}.out \

-z 40;

done

Then create a taxonomic summary of the Kaiju output for each sample. The script below will produce a “species” level summary. The “-u” flag omits the unclassified reads from the relative abundance calculation. If you want to aggregate the results at the “genus” or “phylum” level just change the “-r species” to “-r genus” or “-r phylum”.

##$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate kaiju

for i in \*.out;

do

prefix=$(basename $i .out)

kaiju2table -t nodes.dmp \

-n names.dmp \

-r species \

-u \

-o ${prefix}.species.summary.tsv $i;

done

**Merging multiple Kaiju summary files**

Arun wrote this handy R script for merging all of the individual summary.tsv files into one file.

library(tidyr)

# Read all species level files into lists

data\_list <- lapply( list.files(path="Species/", pattern="\*species.summary.tsv", full.names = T),

FUN=function(x) { read.delim(x, header=T) } )

# Merge lists into long form dataframe

data\_long <- Reduce(function(x, y) merge(x, y, all=TRUE), data\_list)

# Some cleanup

data\_long$reads <- NULL

data\_long$file <- gsub(".out","", data\_long$file)

# Convert long form to wide dataframe

data\_wide <- spread( data\_long, file, percent)

# Order by taxon\_name

data\_wide <- data\_wide[ order(data\_wide$taxon\_name), ]

# Write results to csv

write.csv( data\_wide, file="Kaiju\_merged\_species.csv", row.names = F)

**Kraken2 and Bracken**

Kraken2 is an alternative to using Kaiju. It is similar to Kaiju but uses the nucleotide sequences directly rather than translating them into amino acids first.

<https://github.com/DerrickWood/kraken2/wiki>

Conda package: <https://bioconda.github.io/recipes/kraken2/README.html>

As with Kaiju, a database must be downloaded and created. This will download archaeal, bacterial, and viral genomes from RefSeq.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 80

mamba activate kraken2

kraken2-build --standard --use-ftp --threads 80 --db kraken2\_db

**1.** This method is best if you don’t plan on using Bracken on your Kraken2 output. However, I would recommend using Bracken as it can improve on the taxonomic assignments. To do this, skip to the following Kraken2 script and move to step 2 below.

**--db** = Kraken2 database location. Here we are using the non-redundant nucleotide database created by Martin Gauthier on October 29, 2021 and located here: /isilon/common/reference/databases/kraken2\_ncbi\_nt. Check to see if a new version of the database is available.

**--confidence** = confidence score. This is essentially the number of kmers from a single read that mapped to a particular taxon divided by the number of kmers in total from that read. If this proportion is greater than the confidence score (e.g. 0.5) than the read is classified as that taxon.

**--report-zero-counts** is used here so that reports can be aggregated as all taxa will be included in the report, not just those with “hits”.

**--use-mpa-style** results inone taxon per line with the full taxonomic assignment.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate kraken2

mkdir Kraken2\_output

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

kraken2 --use-names

--threads 40

--db /isilon/common/reference/databases/kraken2\_ncbi\_nt/kraken2\_ncbi\_nt\_20211029

--confidence 0.5

--report Kraken2\_output/Kraken2\_${prefix}.report.txt

--use-mpa-style

--report-zero-counts

--paired $i $j >Kraken2\_output/${prefix}.kraken;

done

Python scripts for combining multiple Kraken2 reports can be found here:

https://github.com/jenniferlu717/KrakenTools

Download the combine\_mpa.py script into Kraken2\_output folder and then run:

python combine\_mpa.py -i \*.report.txt -o Combined\_Kraken2.reports.txt

**2.** Taxonomy in Kraken2 output can be further refined/improved by using a program called Bracken.

<https://ccb.jhu.edu/software/bracken/index.shtml?t=manual>

Conda package : https://bioconda.github.io/recipes/bracken/README.html

To do this first run Kraken2 with the following parameters:

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate kraken2

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

mkdir Kraken2\_output

j=${prefix}\_R2.fastq.gz

kraken2 --use-names

--threads 40

--db /isilon/common/reference/databases/kraken2\_ncbi\_nt/kraken2\_ncbi\_nt\_20211029

--confidence 0.5

--output Kraken2\_output/${prefix}\_output.txt

--report Kraken2\_output/${prefix}\_report.txt

--paired $i $j;

done

cd Kraken2\_output

**Bracken**

Use the report.txt files from Kraken2 as input for Bracken.

-r is the length of the reads used in the Kraken2 script above.

–t = “threshold [Default = 10] and specifies the minimum number of reads required for a classification at the specified rank. Any classifications with less than the specified threshold will not receive additional reads from higher taxonomy levels when distributing reads for abundance estimation.”

-l is the taxonomic rank; S = species, G = genus, and so on.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bracken

for i in \*\_report.txt

do

prefix=$(basename $i \_report.txt)

bracken -d /isilon/common/reference/databases/kraken2\_ncbi\_nt/kraken2\_ncbi\_nt\_20211029-i $i -r 150 -t 10 -l S -o ${prefix}\_bracken\_report\_species.txt;

done

**ANTIMICROBIAL RESISTANCE GENE SCREENING**

<https://github.com/arpcard/rgi>

Conda package: [https://bioconda.github.io/recipes/rgi/README.htm**l**](https://bioconda.github.io/recipes/rgi/README.html)

Download and load the CARD data.

wget https://card.mcmaster.ca/latest/data

tar -xvf data ./card.json

mamba activate rgi

rgi load --card\_json card.json –local

rgi card\_annotation -i card.json > card\_annotation.log 2>&1

rgi load -i card.json --card\_annotation card\_database\_v3.2.6.fasta --local

#change the highlighted portion to match the current version of CARD that was downloaded.

Running the CARD-RGI:

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate rgi

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

rgi bwt --read\_one $i \

--read\_two $j --aligner kma \

--output\_file ${prefix}\_CARD\_RGI \

--clean --threads 40 \

--local;done

**METABOLIC PREDICTION USING HUMAnN 3.0**

HUMAnN (HMP Unified Metabolic Analysis Network) 3.0 is designed to profile “the abundance of microbial metabolic pathways and other molecular functions from metagenomic sequencing data.” As the name indicates it is designed for metagenomic samples derived from humans but it works on pig gut samples too.

<https://huttenhower.sph.harvard.edu/humann>

Conda package: <https://bioconda.github.io/recipes/humann/README.html>

First you need to download ChocoPhlAn which is an annotated pangenome database created by the developers of HUMAnN. You also need to download the UniRef90 database and something called “utility\_mapping”. The full path to where you want to download these databases must always be provided (highlighted in yellow). The databases only have to be downloaded the first time you use HUMAnN.

As of December 21, 2021, the Bowtie2 module in HUMAnN requires that a program called tbb is downgraded to an earlier version.

Within your humann conda environment run:

conda install tbb=2020.2

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate humann

humann\_databases --download chocophlan full /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/ --update-config yes

humann\_databases --download uniref uniref90\_diamond /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/

humann\_databases --download utility\_mapping full /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/ --update-config yes

HUMAnN doesn’t use paired-end sequence data and instead uses sequence separately. Therefore it is recommended that the pair-end files are concatenated instead.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

cat $i $j >${prefix}\_cat.fastq.gz; done

Running HUMAnN. Note: this can take a long time to run depending on the number of metagenomes.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate humann

for i in \*\_cat.fastq.gz

do

prefix=$(basename $i \_cat.fastq)

humann -i $i -o ${prefix}\_humann3\_output --threads 40 --remove-temp-output --nucleotide-database /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/chocophlan --protein-database /isilon/lacombe-rdc/users/holmandb/Metagenome\_example/uniref;done

#End of file

Copy the “pathabundance.tsv” files into a new folder called Pathway\_abundance\_files.These are the MetaCyc pathway abundance files. The raw pathabundance.tsv contains in the first column MetaCyc pathways and a bacterial species. The 2nd column is abundance for each MetaCyc pathway for bacterial species.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate human

mkdir Pathway\_abundance\_files

for i in \*\_cat.fastq.gz

do

prefix=$(basename $r1 \_cat.fastq)

cp ${prefix}\_humann3\_output/${prefix}\_pathabundance.tsv Pathway\_abundance\_files/;done

#End of file

Then join all of the pathabundance.csv into one csv file using the human\_join\_tables command.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate human

humann\_join\_tables -i Pathway\_abundance\_files -o Joined\_pathabundance.tsv --file\_name pathabundance

Then normalize to relative abundance values.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate human

humann\_renorm\_table --input Joined\_pathabundance.tsv --output Joined\_pathabundance\_relab.tsv --units relab

Finally split the Joined\_pathabundance.tsv table into “stratified” and “unstratified” tables. A “stratified” table has each pathway broken down (stratified) into the bacterial species contributing to it and the “unstratified” table collapses it into one pathway. Generally speaking for downstream analyses the unstratified table is the one we will want to use since each pathway will only have one entry.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate human

mkdir Split\_humann\_pathwayabundance\_tables

humann\_split\_stratified\_table --input Joined\_pathabundance\_relab.tsv --output Split\_humann\_pathwayabundance\_tables

The other file type of interest in HUMAnN output is the “genefamilies.tsv”. This file contains the abundance of each gene family by bacterial species within the metagenome sample. Abundance values represent reads per kilobase (RPK). The tables can be handled similar to the pathabundance.tsv files.

**CARBOHYDRATE-ACTIVE ENZYMES (CAZymes)**

Carbohydrate-active enzymes (CAZymes) as the name implies are those enzymes that synthesize or degrade carbohydrates. CAZymes are divided into several classes: glycoside hydrolases (GHs), glycosyltransferases (GTs), polysaccharide lyases (PLs), carbohydrate esterases (CEs), carbohydrate-binding modules (CBMs), and auxiliary activities (AAs). Within each class are a number of families. For example, there are currently 174 families within the GH class.

Perhaps the easiest way to profile the “CAZyome” using short metagenomic reads is with the dbCAN2 database and DIAMOND.

As of the February, 2023, the most current version of the dbCAN2 database is CAZyDB.08062022.fa.

wget <https://bcb.unl.edu/dbCAN2/download/CAZyDB.08062022.fa>

This is an amino acid sequence database so we will have to use DIAMOND which is a faster reimplementation of BLASTx. To use DIAMOND the paired-end fastq files have to be converted to fasta files. seqtk can do this.

seqtk conda package: <https://bioconda.github.io/recipes/seqtk/README.html>

DIAMOND conda package: <https://bioconda.github.io/recipes/diamond/README.html>

Another conda package called parallel is also required. Install this package within your DIAMOND conda env.

Mamba activate diamond

mamba install -c conda-forge parallel

https://anaconda.org/conda-forge/parallel

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 10

mamba activate seqtk

for i in \*.fastq.gz

do

prefix=$(basename $i .fastq.gz)

seqtk seq -a $i > ${prefix}.fasta;done

First make a database that DIAMOND can read.

diamond makedb --in CAZyDB.08062022.fa -d dbCAN2

Then use DIAMOND to translate the nucleotide reads to amino acid sequences and then align against the dbCAN2 database above. This script will align the forward and reverse reads against the dbCAN2 database, find which “hits” are found in both reads, and then count these to produce a summary file with one entry per sample.

The python scripts get-paired-hits-AK.py and count-up.py need to be in the folder you run this bash script from.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate diamond

#run DIAMOND

for x in `ls \*.fasta`; do echo "diamond blastx --db dbCAN2.dmnd --query $x --out $x.90\_dbCAN2.DIAMOND --threads 20 --id 90 --query-cover 90 -f 6"; done >> DIAMOND-dbCAN2\_90.sh

cat DIAMOND-dbCAN2\_90.sh | parallel

#get the paired hits

for x in `ls \*R1.fasta.90\_dbCAN2.DIAMOND`; do echo "python get-paired-hits-AK.py $x ${x%R1\*}R2.fasta.90\_dbCAN2.DIAMOND > $x.90\_dbCAN2.DIAMOND.pairs"; done > get-pairs-dbCAN2\_90.sh

cat get-pairs-dbCAN2\_90.sh | parallel

#count the hits

python count-up.py \*90\_dbCAN2.DIAMOND.pairs

mv summary-count.tsv dbCAN2.counts\_90.tsv

**METAGENOME-ASSEMBLED GENOMES**

Use the same files that were used as initial input for Kaiju. It is usually best to use both a co-assembly of the metagenomes as well as an individual assembly for each sample. This will help capture as many unique MAGs as possible.

To do a co-assembly of all fastq files first concatenate them.

cat file\*R1.fastq.gz > coassembly\_R1.fastq.gz

cat file\*R2.fastq.gz > coassembly\_R2.fastq.gz

Then use MEGAHIT to coassembly all the reads from all samples. MEGAHIT conda package: <https://bioconda.github.io/recipes/megahit/README.html>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 80

mamba activate megahit

megahit -1 coassembly\_R1.fastq \

-2 coassembly\_R2.fastq \

-o Megahit\_coassembly \

--out-prefix Coassembly \

-t 80 \

--min-contig-len 1000

After you’ve made your assembly(ies) you need to build an index of the assembly(ies) so that Bowtie2 can map the reads from each sample back to this assembly.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bowtie2

mkdir coassembly

bowtie2-build Megahit\_coassembly/Coassembly.final.contigs.fa coassembly/coassembly

Then map each sample to the assembly with Bowtie2:

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mkdir bowtie2\_output\_for\_metabat

mamba activate bowtie2

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

bowtie2 -x coassembly/coassembly \

-1 $i -2 $j \

-S bowtie2\_output\_for\_metabat/${prefix}.sam -p 40;

done

Use SAMtools to convert and sort the SAM files that were output to BAM files.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mkdir Contigs\_mapped

mamba activate samtools

for i in \*.sam

do

prefix=$(basename $i .sam)

samtools sort $i > Contigs\_mapped/${prefix}\_sorted.bam;

done

Use MetaBAT2 to bin contigs from the coassembly into metagenome-assembled genomes (MAGs) by sample. It requires that you direct it to the coassembly contigs (e.g final.contigs.fa) from MEGAHIT and the BAM files above. This script will create a “depth” file with the coverage for each contig.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate metabat2

jgi\_summarize\_bam\_contig\_depths --outputDepth Coassembly\_depth.txt \*.bam

Then this depth file is used to put the contigs with a minimum length of 2000 bp into “bins” (MAGs).

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 80

mamba activate metabat2

mkdir Coassembled\_bins

for i in \*\_sorted.bam

do

prefix=$(basename $i \_sorted.bam)

metabat2 -i <full path name to coassembly contig file>/Coassembly.final.contigs.fa \

-a Coassembly\_depth.txt \

-o Coassembled\_bins/${prefix}/${prefix}.bin \

-t 80 -m 2000 -v;

done

The next step is run CheckM2 to assess completeness and contamination of each bin.

The “.bam” is used here just to grab the sample names which are needed for CheckM2 to recognize the bin folders. Otherwise, the BAM files are not needed here. The

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 20

mamba activate checkm2

mkdir CheckM2\_output\_coassembled\_bin

for r1 in \*.bam

do

prefix=$(basename $r1 .bam)

checkm2 predict --threads 20 -x fa --input Coassembled\_bins/${prefix} --output-directory CheckM2\_output\_coassembled\_bin

**OPTIONAL**: sometimes if you have too many sequences it can pose a problem for assemblers. Therefore, it becomes necessary to normalize the read coverage. This can be done with a program called BBNorm within the BBTools/BBMap package. Run this on the coassembly\_R1.fastq.gz and coassembly\_R2.fastq.gz from above.

<https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbnorm-guide/>

Conda package: <https://bioconda.github.io/recipes/bbmap/README.html>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bbmap

bbnorm.sh in=coassembly\_R1.fastq.gz \

in2=coassembly\_R2.fastq.gz \

out=normalized\_coassembly\_R1.fastq.gz \

out2=normalized\_coassembly\_R2.fastq.gz \

target=100 min=5 threads=40 -Xmx1000g

Then use the normalized\_coassembly\_R1.fastq.gz and normalized\_coassembly\_R2.fastq.gz files as input for MEGAHIT and continue as above.

**Individual assemblies**

Each sample can also be assembled individually using most of the same steps as above. First each metagenomic sample is assembled using MEGAHIT.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mkdir Indiv\_assemblies

mamba activate megahit

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

megahit -1 $i -2 $j \

-o ${prefix} \

--out-prefix ${prefix} \

-t 40 \

--min-contig-len 1000

mv ${prefix}/${prefix}.contigs.fa Indiv\_assemblies/

rm -r ${prefix};

done

Then each sample is mapped back to its own assembly using Bowtie2.

First an Bowtie2 index needs to be created for each assembly. Run this within the “Indiv\_assemblies” folder.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bowtie2

for i in \*.contigs.fa

do

prefix=$(basename $i .contigs.fa)

mkdir ${prefix}

bowtie2-build $i ${prefix}/${prefix}; done

Then run the next script one directory above **Indiv\_assemblies**

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate bowtie2

mkdir Bowtie2\_indiv\_assembly\_mapped

for i in \*\_R1\_paired.fastq.gz

do

prefix=$(basename $i \_R1\_fastq.gz)

j=${prefix}\_R2.fastq.gz

bowtie2 -x Indiv\_assemblies/${prefix}/${prefix}

-1 $i -2 $j

-S Bowtie2\_indiv\_assembly\_mapped/${prefix}.sam -p 40;

done

Convert and sort the SAM files with SAMtools.

cd Bowtie2\_indiv\_assembly\_mapped

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate samtools

for i in \*.sam

do

prefix=$(basename $i .sam)

samtools sort $i > ${prefix}\_sorted.bam

done

Then create a depth file for each sample using MetaBAT2.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 80

mamba activate metabat2

for i in \*\_sorted.bam

do

prefix=$(basename $i \_sorted.bam)

jgi\_summarize\_bam\_contig\_depths --outputDepth ${prefix}\_depth.txt $i; done

MetaBAT2 is run again using the above depth.txt files to place the contigs into bins.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

cd ..

mkdir Indiv\_assembled\_bins

mamba activate metabat2

for i in Indiv\_assemblies/\*.contigs.fa

do

prefix=$(basename $i .contigs.fa)

metabat2 -t 40 -m 2000 \

-i Indiv\_assemblies/${prefix}.contigs.fa \

-a Bowtie2\_indiv\_assembly\_map/${prefix}\_depth.txt \

-o Indiv\_assembled\_bins/${prefix}/${prefix}.individ.bin;

done

Now run CheckM2 on these bins.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mkdir CheckM2\_output

mamba activate checkm

for i in Indiv\_assemblies/\*.contigs.fa

do

prefix=$(basename $i .contigs.fa)

checkm2 predict --threads 20 -x fa --input Indiv\_assembled\_bins/${prefix} --output-directory CheckM2\_output\_individual\_bins

**Combining individually assembled and co-assembled bins**

According to Bowers et al. (2017), high-quality draft MAGs are those >90% complete and <5% contamination. Generally speaking, these are the guidelines that the research community adheres to. I typically only focus on these high-quality MAGs for downstream analyses.

<https://www.nature.com/articles/nbt.3893>

To do this I sort through the CheckM output by completeness and contamination. Once I’ve done this I make a new folder called “All\_bins” or something similar.

mkdir All\_bins

Then I move all of the bins from the Coassembled\_bins folder to All\_bins

mv Coassembled\_bins/\*/\*.fa All\_bins

And all the individually-assembled bins as well.

mv Indiv\_assembled\_bins/\*/\*.fa All\_bins

However, we only want to focus on the high-quality MAGs.I make another new folder called High\_quality\_bins and a txt file (e.g. High\_quality\_bins.txt) with the names of all of the bins from the sorted CheckM2 output that are >90% complete and with <5% contamination.

mkdir High\_quality\_bins

Then I moved all of the bins meeting that criteria in the All\_bins folder into the High\_quality\_bins folder.

rsync -av --remove-source-files --files-from=Indiv\_bins\_high\_quality.txt All\_bins/ High\_quality\_bins

**Dereplication of MAGs**

Many of the MAGs will be redundant and share >99% average nucleotide identity (ANI) with other MAGs. It is best at this stage to remove the redundant MAGs so that only one MAG is left as a representative for that particular MAG. The MAGs can be “dereplicated” with a program called dRep.

<https://github.com/MrOlm/drep>

dRep conda package: <https://bioconda.github.io/recipes/drep/README.html>

Here the “comp 90” and “con 5” parameters are telling it to use 90% completeness and 5% contamination as the cutoffs. Since we have already filtered our MAGs this should have no effect. “strW” is the strain heterogeneity weight. It is used although with contamination, completeness, and N50 when deciding which bin should be the “winning bin” when there are multiple bins sharing an ANI of >99%.

dRep first clusters the bins at a user-defined identity (primary clustering; pa) which here we are using 90%. Then the bins within each of these primary cluster are clustered again (secondary clustering; sa) using ANI. Here we are using 99% for our secondary clustering. MAGs with ≥95% ANI can be considered to be from the same bacterial species and those with ≥99% ANI are from the same bacterial strain.

Note: when there are thousands of bins to be dereplicated it is generally necessary to use fastANI for secondary clustering (S\_algorithm).

dRep will also run CheckM1; however, the previous CheckM2 results can provided allowing the program to skip this step. In this case the checkM\_results.csv file should have the following headers: genome,completeness,contamination,strain\_heterogeneity. As of March 9, 2023 CheckM1 is still being used by dRep. Because CheckM2 no longer reports “strain\_heterogeneity” I would just put in 0’s in in the strain\_heterogeneity column since we aren’t using that information anyway.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate drep

dRep dereplicate -g High\_quality\_bins \

-comp 90 -con 5 -p 40 \

-strW 1 -pa 0.90 -sa 0.99 \

--S\_algorithm fastANI \

--multiround\_primary\_clustering \

--greedy\_secondary\_clustering \

--run\_tertiary\_clustering \

--genomeInfo checkM\_results.csv dRep\_output

Within the dRep\_output folder there will be another folder called “dereplicated\_genomes”. These are the MAGs that have <99% ANI and will be used for all downstream analyses.

**ASSEMBLY STATISTICS OF MAGS AND CO-ASSEMBLED METAGENOME**

QUAST can be used to assess the assembly quality of both the MAGs and co-assembled metagenome.

<http://bioinf.spbau.ru/quast>

QUAST conda package: <https://bioconda.github.io/recipes/quast/README.html>

For MAGs :

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate quast

quast.py dRep\_output/dereplicated\_genomes -o QUAST\_replicated\_MAGs;done

For the co-assembled metagenome:

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate quast

metaquast.py Megahit\_coassembly/Coassembly.final.contigs.fa -t 40

**TAXONOMY OF MAGS**

The first thing we probably want to do with our dereplicated MAGs is assign taxonomy to them. This can be done using GTDB-Tk.

<https://github.com/Ecogenomics/GTDBTk>

GTDB-Tk conda package: <https://bioconda.github.io/recipes/gtdbtk/README.html>

This program assigns taxonomy to genomes using a combination of ANI to reference genomes in the GTDB, relative evolutionary divergence, and placement in the reference tree.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate gtdbtk

gtdbtk classify\_wf --genome\_dir dRep\_output/dereplicated\_genomes

-x fa --out\_dir GTDBtk\_output --cpus 40

The gtdbtk.bac120.summary.tsv has the taxonomy for bacteria and gtdbtk.ar53.summary.tsv contains the archaeal taxonomy. The 2nd column has the classification for each bin. Species ending with “sp006954405” for example are uncultured and so the designation is used by the GTDB developers to keep track of potential new species in the GTDB.

If a genome lacks a designation at the species level (i.e. “s\_\_”) it \*may\* potentially represent a new species. GTDB-Tk only provides ANI values for genomes with an ANI of 95% or higher to any of the genomes in the GTDB. A full summary of the various columns is available here: <https://ecogenomics.github.io/GTDBTk/files/summary.tsv.html>

**DRAM (Distilled and Refined Annotation of Metabolism)**

[**https://github.com/shafferm/DRAM**](https://github.com/shafferm/DRAM)

DRAM conda package: <https://bioconda.github.io/recipes/dram/README.html>

DRAM is a tool for predicting and annotating genes within genomes using CAZymes and KEGG. We have access to the full version of KEGG through the Ottawa RDC so we can give DRAM the locations of the required KEGG files during the database setup step.

Presently, it is also necessary to manually download the viral.1.protein.faa.gz file required by DRAM since there is an error in the FTP URL that DRAM tries to download from.

This is the correct one:

wget http://ftp.ncbi.nlm.nih.gov/refseq/release/viral/viral.1.protein.faa.gz

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate dram

DRAM-setup.py prepare\_databases \

--verbose --threads 40 \

--gene\_ko\_link\_loc /isilon/ottawa-rdc/reference/KEGG/ftp.kegg.net/kegg/genes/links/genes\_ko.list.gz \

--kegg\_loc /isilon/ottawa-rdc/reference/KEGG/ftp.kegg.net/kegg/genes/fasta/prokaryotes.pep.gz \

--viral\_loc viral.1.protein.faa.gz \

--output\_dir DRAM\_data

**Annotation**

Once the DRAM database has been set up move on to annotating the dereplicated MAGs. You can also provide the GTDB taxonomy file and it will be included in some of the output files downstream. Use the same headers in the gtdbtk.bac120.summary.tsv and paste in the data for the archaeal MAGs as well.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

Mamba activate dram

DRAM.py annotate -i 'dRep\_output/dereplicated\_genomes/\*.fa'\

-o DRAM\_annotated\_MAGs --verbose --threads 40 \

--gtdb\_taxonomy gtdbtk.bac120.ar53.summary.tsv

**Distillation**

This last step will summarize the annotations and provide several files with useful information about the metabolic and carbohydrate activity potential of the genomes. The “genomes\_per\_product” flag should be included if you have more than 600 MAGs since the heatmap in the product.html output file cannot handle more than that many MAGs in one heatmap.

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate dram

DRAM.py distill -i DRAM\_annotated\_MAGs/annotations.tsv \

-o MAG\_DRAM\_distilled\_summaries \

--trna\_path DRAM\_annotated\_MAGs/trnas.tsv \

--rrna\_path DRAM\_annotated\_MAGs/rrnas.tsv \

--genomes\_per\_product 575

**ANTIMICROBIAL RESISTANCE GENE SCREENING**

The MAGs can be screened for antimicrobial resistance genes similar to the unassembled metagenomic reads using the CARD-RGI. This bash script courtesy of Arun will set up an array on the Biocluster greatly reducing the computing time required.

Create a folder where the log files for each job in the array will be generated.

mkdir CARD\_RGI\_dRep\_MAGs\_logs

#!/bin/bash

# Specify name to be used to identify this run

#$ -N CARD\_RGI\_dRep\_MAGs

# This sets the task range in the array and step size

#$ -t 1-1150:1

# This sets the maximum number of concurrent tasks

#$ -tc 100

# Specify the number of processors for the single job

#$ -pe smp 1

# Change directory to the current

#$ -cwd

# Specify that bash shell should be used to process this script

#$ -S /bin/bash

# Specify the output file:: Make sure this folder exists

#$ -o $JOB\_NAME\_logs/$TASK\_ID.out

# Specify the error file:: Make sure this folder exists

#$ -e $JOB\_NAME\_logs/$TASK\_ID.err

#########################################################################

## Section 2: Definitions #

#########################################################################

# Specify MAGS directory

MAGS\_DIR="/isilon/lacombe-rdc/users/holmandb/Metagenome\_example/dRep\_output/dereplicated\_genomes"

# Specify the filenames

INPUTFILES=($(ls $MAGS\_DIR/\*))

# Pull data file name from list defined above according to job id

INPUTFILENAME="${INPUTFILES[$SGE\_TASK\_ID - 1]}"

# Specify destination directory to store output files: Make sure this folder exists

DESTINATION\_DIR="/isilon/lacombe-rdc/users/holmandb/Metagenome\_example/dRep\_output/dereplicated\_genomes/CARD\_RGI\_output"

#########################################################################

## Section 3: Executing the program #

#########################################################################

# Run the program

mamba activate rgi

rgi main --input\_sequence $INPUTFILENAME --output\_file $DESTINATION\_DIR/output\_$(basename $INPUTFILENAME .fa) --input\_type contig --alignment\_tool DIAMOND --num\_threads 1 --clean

**TO BE ADDED: HOW TO COMBINE THE MULTIPLE FILES OUTPUT FROM RGI-CARD**

**PHYLOGENETICS/PHYLOGENOMICS**

The phylogeny of the MAGs is often something that we are interested in as well. PhyloPhlAn 3.0 is designed to handle hundreds of thousands of genomes and aligns 400 universal marker genes from the MAGs used as input, producing a phylogenetic tree at the end.

<https://huttenhower.sph.harvard.edu/phylophlan/>

PhyloPhlAn conda package: <https://bioconda.github.io/recipes/phylophlan/README.html>

#$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 112

mamba activate phylophlan

phylophlan -d phylophlan \

-i dRep\_output/dereplicated\_genomes \

-o Phylophlan\_output \

--db\_type a \

-f supermatrix\_aa.cfg \

--nproc 112 \

--diversity low \

--fast \

--verbose \

--genome\_extension fa

In the Phylophlan\_output folder the RAxML\_bestTree.dereplicated\_genomes\_refined.tre is the Newick tree that you can use as input in any tree viewer program (e.g. iTOL).

**TO BE ADDED: HOW TO USE iTOL TO ADD METADATA TO THE TREE.**

**DETERMING THE RELATIVE ABUNDANCE OF MAGS IN SAMPLES**

The relative abundance of each of the dereplicated MAGs in each of the unassembled metagenome samples can be determined using CoverM.

<https://github.com/wwood/CoverM>

CoverM conda package: <https://bioconda.github.io/recipes/coverm/README.html>

$ -S /bin/bash

#$ -V

#$ -cwd

#$ -o $JOB\_NAME.$JOB\_ID.out

#$ -e $JOB\_NAME.$JOB\_ID.err

#$ -pe smp 40

mamba activate coverm

for i in \*\_R1.fastq.gz

do

prefix=$(basename $i \_R1.fastq.gz)

j=${prefix}\_R2.fastq.gz

coverm genome -1 $i -2 $j \

--genome-fasta-directory dRep\_output/dereplicated\_genomes \

--genome-fasta-extension fa \

--min-covered-fraction 1 \

--threads 40 -v \

--output-file ${prefix}\_coverM\_output.txt; done

**STATISTICAL ANALYSIS USING MAASLIN2**

MaAsLin2 is an R-based package that can identify differentially abundant features (e.g. species, ARGs, CAZymes, etc.) as long as the input is relative abundance (aka total sum scaling) values.

<https://huttenhower.sph.harvard.edu/maaslin/>

To install, run the following in RStudio:

if(!requireNamespace("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("Maaslin2")

Then load the Maaslin2 library:

library(Maaslin2)

The first step is to import a tab-delimited txt file (data file) into R. The first column should be “Name” and then the list of features (species/MAGs/CAZymes, etc.). Each of the next columns should be a sample name and then the relative abundance for each “feature” within that sample.

I typically remove any feature with a percent relative abundance of less than 0.1% within the group of samples being compared. This helps preserve statistical power and features below than this cutoff are less likely to be biologically meaningful.

input\_data\_MAGs = read.table(file = "P:/Lacombe\_projects/Weaning\_microbiome\_study\_Alberta\_Ag/Analysis/MAGs/MAGs\_pre\_vs\_postweaning.txt", header = TRUE, sep = "\t",row.names = 1, stringsAsFactors = FALSE)

The second step involves importing another tab-delimited file with all of the metadata for each sample. The first column should be labeled “ID” and subsequent columns labeled for each relevant metadata category.

Note: he data and metadata files can contain samples that are not in both. However, any sample not included in both files will be excluded from the analysis.

input\_metadata\_MAGs = read.table(file = "P:/Lacombe\_projects/Weaning\_microbiome\_study\_Alberta\_Ag/Analysis/MAGs/Metadata.txt", header = TRUE, sep = "\t",row.names = 1,stringsAsFactors = FALSE)

The last step here is to run a multivariable regression model using Maaslin2. In this example, pre-weaned and post-weaned pigs are being compared to identify MAGs that are differentially abundant between the two groups. The relative abundance values were obtained by aligning the metagenomic sequences from each sample to a group of MAGs that were assembled and binned from these same samples.

Here, Maaslin2 will generate a new folder in your working directory called “Pre\_vs\_post\_weaning\_MAGs” and use the input\_data\_MAGs and input\_metadata\_MAGs files from above. The only fixed effect is weaning status, referred to as “Wean” in the input\_metadata\_MAGs metadata file. For more complex experiments it is possible to include multiple fixed factors, e.g. ‘Wean’ and ‘Time’.

The “reference” parameter can be included when you want to specify what you want to compare the other level of the category against. In the example below, we are setting “Pre” as the reference “level” within the “Wean” category.

“max\_significance” is the corrected p-value to use as a cutoff for the “significant\_results.tsv” that is generated. I almost always use 0.05 as the cutoff unless there are very few or no significantly differentially abundant features at that level. Then I would increase this cutoff to 0.10.

Usually our data will already be normalized by total-sum scaling (relative abundance) so we indicate this as “NONE” in the “normalization” parameter.

The “standardize” parameter is only used if we have a continuous metadata category like pH.

fit\_data <- Maaslin2(

input\_data\_MAGs, input\_metadata\_MAGs, 'Pre\_vs\_post\_weaning\_MAGs',

fixed\_effects = c('Wean'),

reference = "Wean,Pre",

max\_significance = 0.05,

normalization = 'NONE',

standardize = FALSE)

In the generated output folder, here that is “Pre\_vs\_post\_weaning\_MAGs”, there are a number of files, the most important is the “significant\_results.csv” file. This file contains only those features (e.g. MAGs) that are significantly (FDR ≤ 0.05) different between the groups. FDR = false discovery rate = p-values corrected for multiple comparisons using the Benjamini-Hochberg (BH) procedure.

These features are listed in the first column, the metadata category associated with the feature is in the 2nd column, and the level of that metadata category in the 3rd column. Here that will be “Pre” since we set that as the reference level in the script above. The “coef” column is the coefficient is the effect size for the difference. Positive coefficient values indicate features that are more relatively abundant in the reference level (i.e. Pre) and negative coefficient values are those features more relatively abundant in the other levels (i.e. Post). The higher the value (positive or negative) the greater the association with that particular category level.

stderr = the standard error from the multivariable regression model.

N = number of samples used as input in the model.

N.not.0 = the number of samples within “N” that have values greater than 0 for a particular feature (e.g. MAG).

pval = the uncorrected p-values.

qval = these are the BH-corrected p-values. These are the p-values that should be reported in a manuscript.

In this example there were 48 MAGs that were more relatively abundant in pre-weaned pigs and 51 MAGs that were more relatively abundant in post-weaned pigs.

If you are unsure of whether a positive or negative coefficient means that a feature is associated with a particular category level, Maaslin2 also generates box plots that visually show the values for each significantly differentially abundant feature. Here that is the Wean.pdf file. I always find it good to check this just so I can confirm which category level is being used as the reference.

Maaslin2 can also include “random effects”, e.g. subject, when same animals are sampled repeatedly over time. This involves adding this term to the model above:

random\_effects = c("subject"))