Genome-centric Metagenome Analysis – Standard Operating Procedure V2.1

Energy Bioengineering and Geomicrobiology

Xiaoli Dong, Marc Strous

February 15, 2019

Introduction

This SOP describes the workflow of the genome-centric, shotgun metagenome sequence data analysis procedure used in the EBG lab group at the University of Calgary. The workflow consists of several steps: sequence data acquisition and data pre-processing, assembly, short read mapping, gene prediction, gene annotation, and binning. The aim of genome-centric metagenomics is to convert the shotgun raw sequencing reads into as many as possible, nearly complete provisional bacterial whole genome sequences.

Step 1: Sequence data quality checking

After getting sequence data from sequence center, the FastQC program will be used to evaluate multiple aspects of the raw sequencing data quality, such as per base quality, per base GC content and sequence length distribution. It generates a QC report which can spot problems that originated either in the sequencer or in the starting library material. The FastQC tutorial is available from FastQC Home Page. To run fastqc, simply type:

```
# to check fastqc’s parameters
fastqc -h

#generate fastqc reports
fastqc read1.fastq read2.fastq
```

Step 2: Quality control (QC)

Sequence data must be pre-processed by Quality Control (QC) procedures before they can be utilized for omic analysis. The QC procedures usually include clipping off the technical sequences (primer, adapter), trimming off the low-quality ends, identifying and filtering out artifacts, short reads, and contamination reads. There are many publicly available QC toolkits that perform metagenome sequence data quality control. Some representative tools are Sickle, Cutadapt, Trimmomatic, and BBDuk. The following are the examples on how to do QC using BBDuk. To pre-process raw sequence data with BBDuk, a recommended specific order needs to be followed:

```
# BBDuk’s parameters are described in its shell script:
Bbduk.sh -h

#Step 1 (optional), when your reads are 151 bp instead of 150bp, this command
trims off the last base of 151bp reads because that base is usually very low
quality.

bbduk.sh -Xmx10g in=read1.fastq.gz in2=read2.fastq.gz
out=read1.lastbase_rm.fastq out2=read2.lastbase_rm.fastq ftm=5

#Step 2: trim off the partial adapter
bbduk.sh -Xmx10g in=read1.lastbase_rm.fastq in2=read2.lastbase_rm.fastq
out=read1.adapter_rm.fastq out2=read2.adapter_rm.fastq
ref=$BBMAP/resources/adapters.fa tbo tpe k=23 mink=11 hdist=1 ktrim=r

#Step 3: filter out contaminates
bbduk.sh in=read1.adapter_rm.fastq in2=read2.adapter_rm.fastq
out=read1.dec.fastq out2=read2.dec.fastq outm=contaminates.fq
ref=$BBMAP/resources/phix_adapters.fa.gz k=31 hdist=1 stats=stats.txt

#step 4: clipping off the low quality ends
bbduk.sh -Xmx10g in=read1.dec.fastq in2=read2.dec.fastq out=read1.qc.fastq
out2=read2.qc.fastq qtrim=rl trimq=15 minlength=30 entropy=0.5
```

Step 3: Read merging or combining overlapped reads (optional)

Combining overlapped reads before metagenome assembly can sometimes improve the assemblies and speed up the process for some assemblers. This is an optional step. The BBMerge tool can be used to merge the the overlapping reads:

```
bbmerge.sh in=read1.qc.fastq in2=read2.qc.fastq out=qc.merged.fastq
outu=read1.qc.unmerged.fastq outu2=read2.qc.unmerged.fastq
```

Step 4: Assembly

Genome assembly is the process that reconstructs genomes from the smaller DNA segments called reads which are generated in a sequencing experiment. There are a number of metagenome assemblers widely used for Illumina sequences such as IDBA, SOAPDenovo2, MegaHit, and metaSPAdes. The following are examples on how to use metaSPAdes and MegaHit to do the assembly. Detailed tutorials are available from their website:

```
#access megahit’s parameter
megahit -h

#do the assembly using quality controlled reads
megahit -1 read1.qc.fastq -2 read2.qc.fastq -t 8 -m 0.5 -o megahit_out --min-
contig-len 500

#do the assembly using quality control and merged reads
megahit -1 read1.qc.unmerged.fastq -2 read2.qc.unmerged.fastq -r
qc.merged.fastq -t 8 -m 0.5 -o megahit_out --min-contig-len 500

# To check metaSPAdes’s parameters
metaspades.py -h

#assembly using quality controlled reads
metaspades.py -1 read1.qc.fastq -2 read2.qc.fastq -o metaspades_out -t 8

#assembly using quality controlled and merged reads
metaspades.py -1 read1.qc.unmerged.fastq -2 read2.qc.unmerged.fastq --merged
qc.merged.fastq -o metaspades_out -t 8
```

Step 5: Assembly quality assessment (optional)

MetaQuast is a quality assessment tool for evaluating and comparing metagenome assemblies based on alignments to close reference sequences. It produces many reports, summary tables and plots. More information is available on MetaQuast home page.

```
# See the full list of options to run metaquast.py
metaquasy.py –help

# run metaquast with reference genomes
metaquast.py -m 500 -t 8 -r reference_genome1,reference_genome2,… -o
metequast_out -l megahit,metaspades contigs.megahit.fasta
contigs.metaspades.fasta

# run metaquast without reference genomes
metaquast.py -m 500 -t 8 -o metequast_out -l megahit,metaspades
contigs.megahit.fasta contigs.metaspades.fasta
```

Step 6: Remove short contigs from the assembly:

Filtering out short contigs before the annotation and binning processes can significantly speed up the downstream annotation process and improve the binning results. The command below filters out all contigs shorter than 500 bp. The script is included in the MetaErg installation.

```
# filter out contigs shorter than 500 bp
filterContigByLength.pl contigs.fasta 500 > contigs.500.fasta
```

Step 7: Mapping

Mapping is the process of aligning short reads back to reference genomes, genes, or contigs. The mapping results can be used to compute contig coverages, quantify genes, infer taxonomic diversities, functional and pathway profiles, and improve binning results. There are a number of tools available to do the short-read mapping such as bowtie2, bwa, and BBMap. The following are examples on how to use BBMap to do the short-read mapping. To see more information of BBMap, please check BBMap Guide.

```
# Step 1: make a mapping directory
mkdir mapping

# Step 2: mapping
bbmap.sh ref=contigs.500.fasta in=read1.qc.fastq in2=read2.qc.fastq
out=mapping/bbmap.sam covstats=mapping/bbmap_covstats.txt
scafstats=mapping/bbmap_scafstats.txt threads=8

# Step 3: converting sam file to bam file and sort the bam file
samtools view -b mapping/bbmap.sam | samtools sort -o mapping/bbmap_sorted.bam
-

# Step 4: indexing sorted bam file
samtools index mapping/bbmap_sorted.bam

#The step 2 to 4 should be applied to all the relevant samples before moving
to to step 5


# Step 5: calculating coverage depth for each sequence in the assembly
jgi_summarize_bam_contig_depths --outputDepth mapping/depth.txt mapping/*.bam
```

Step 8: Annotation

Many metagenome annotation systems or pipelines are publicly available such as MG-RAST, IMG/M, and EBI Metagenomics. MetaErg, a locally developed genome and metagenome annotation pipeline, is used to do the metagenome annotation in our group. For how to use MetaErg, see the box below:

```
# check MetaErg’s parameters
metaerg.pl --help

# run the annotation pipeline
metaerg.pl --mincontiglen 500 --minorflen 180 –prefix sodalake --outdir
metaerg_out --locustag cyano  --cpus 8  --gtype meta –gcode 11 –depth
mapping/depth.txt contig.500.fasta
```

Step 9: Binning

During the binning process, contigs originating from the same genome are grouped into bins. Each bin represents an individual population from a mixed microbial community. MetaBat is an automated metagenome binning software, which integrates empirical probabilistic distances of genome abundance and tetranucleotide frequency. CheckM provides a set of tools for assessing the quality of genomes recovered from isolates, single cells, or metagenomes. It provides robust estimates of genome completeness and contamination by using collocated sets of genes that are ubiquitous and single-copy within a phylogenetic lineage. Currently in EBG, MetaBat and CheckM are used together to perform metagenome contig binning, evaluate the completeness and contamination of the bins, generate abundance profiles of each bin, and extract unbinned sequences. For how to use MetaBat and CheckM, see the detailed tutorials from the MetaBat home page and the CheckM home page.

Perform binning, evaluate and select best bins

The step by step procedures on how to use MetaBat and CheckM to perform binning and to select the best bins are available in The Best Binning Practices tutorial from MetaBat's home page. The following are the basic commands to run MetaBat and CheckM from the command line:

```
# access MetaBat's parameters
metabat -h

# run metabat
metabat -i contigs.500.fasta -a mapping/depth.txt -o resA1/bin -v

# access CheckM lineage_wf command parameters
checkm  lineage_wf -h

# run CheckM
checkm lineage_wf -f resA1/CheckM.txt -t 8 -x fa resA1/ resA1/SCG

#After binning and evaluation process, resA1/Bin.*.fa files include all the
contigs belong to each bin and resA1/CheckM.txt has basic statistics of the
binning results. resA1/ SCG/storeage/tree/concatenated.tre is a newick format
phylogenetic tree file can be used to infer taxonomic origin of each bin.
```

Phylogeny of the metagenome bins

Genome classification can be obtained using the protein phylogeny in the Genome Taxonomy Database, which seeks to provide standardized microbial taxonomy inferred from a concatenated set of 120 single copy marker proteins for bacterial and 122 marker proteins for Archaea, with "gtdbtk classify_wf " command of the GTDB-Tk program. The classification provided by the program are in the files "*.bac120.summary.tsv" and "*.ar122.summary.tsv". The resulting Newick tree files "*.bac120.classify.tree" and "*.ar122.classify.tree" can be open in a tree viewer software such as Dendroscope for visualization.

```
#classify genome bins sitting inside resA1 directory generated in the binning
process
gtdbtk classify_wf –genome_dir resA1 -x fa –out_dir gtdbtk_resA1_out –prefix
sodaLake –cpus 8
```

Visualize MetaBat bins (optional)

Metagenome bins can be visualized and manually refined in VizBin software, which allows a further separation of bins if they are formed from two or more distinct clusters. MetaErg includes a script to prepare the inputs for VizBin. This script outputs two files. The "binned_concat.fasta" file includes all the binned sequences with bin ids included in the sequence headers. The "binned_annotation.list" file lists all the bin ids in the same order as the fasta sequence file. These two files can be used as inputs to the VizBin software for visualizing the interested or selected bins.

```
# Bin.*.fa files in resA1 directory contains the binned contigs of bins
getVizBinInput.pl -d resA1
```

Generate relative abundance profile of populations within a community

Firstly, a coverage profile for all sequences within a set of genome bins is generated. Secondly, the coverage profile is used to produce a table indicating the percentage of reads mapped to an assembly which are assigned to each genome bin or assigned to unbinned contigs. This is a useful indication of the relative percentage of different populations within a community. The following are the example commands:

```
# produces coverage profiles for all sequences within resA1
checkm coverage -x fa resA1 coverage.resA1.tsv mapping/*.bam

# determine the percentage of each genome bin relative to all genome bins
under consideration.
checkm profile coverage.resA1.tsv
```

Extract unbinned sequences

During the binning, not all the contigs are binned. CheckM provides utility command to extract all unbinned sequences from the total contig file. The following is one example:

```
# access checkm unbinned utility command's parameters
checkm unbinned -h

# extract unbinned sequences and get unbinned sequence statistics
checkm unbinned resA1 contigs.500.fasta unbinned.fna unbinned_stas.txt
```

Add bin id to the protein coding sequence

The predicted protein coding sequences generated using MetaErg during the annotation process can be used to build a proteomics data searching database. It is very often that people want to know the taxonomic origin or bin origin of each of the coding sequences. The MetaErg pipeline provides a script to add binning ids to all the protein coding sequence headers before they are used for constructing proteomics search databases.

```
# add bin ids to the protein coding sequences in cds.faa file generated using
MetaErg
add_binid2cds.pl -d resA1 -c metaerg_out/data/cds.faa -g
metaerg_out/data/master.gff  > cds_with_binid.faa
```

Produce MetaErg annotation for an individual metagenome bin

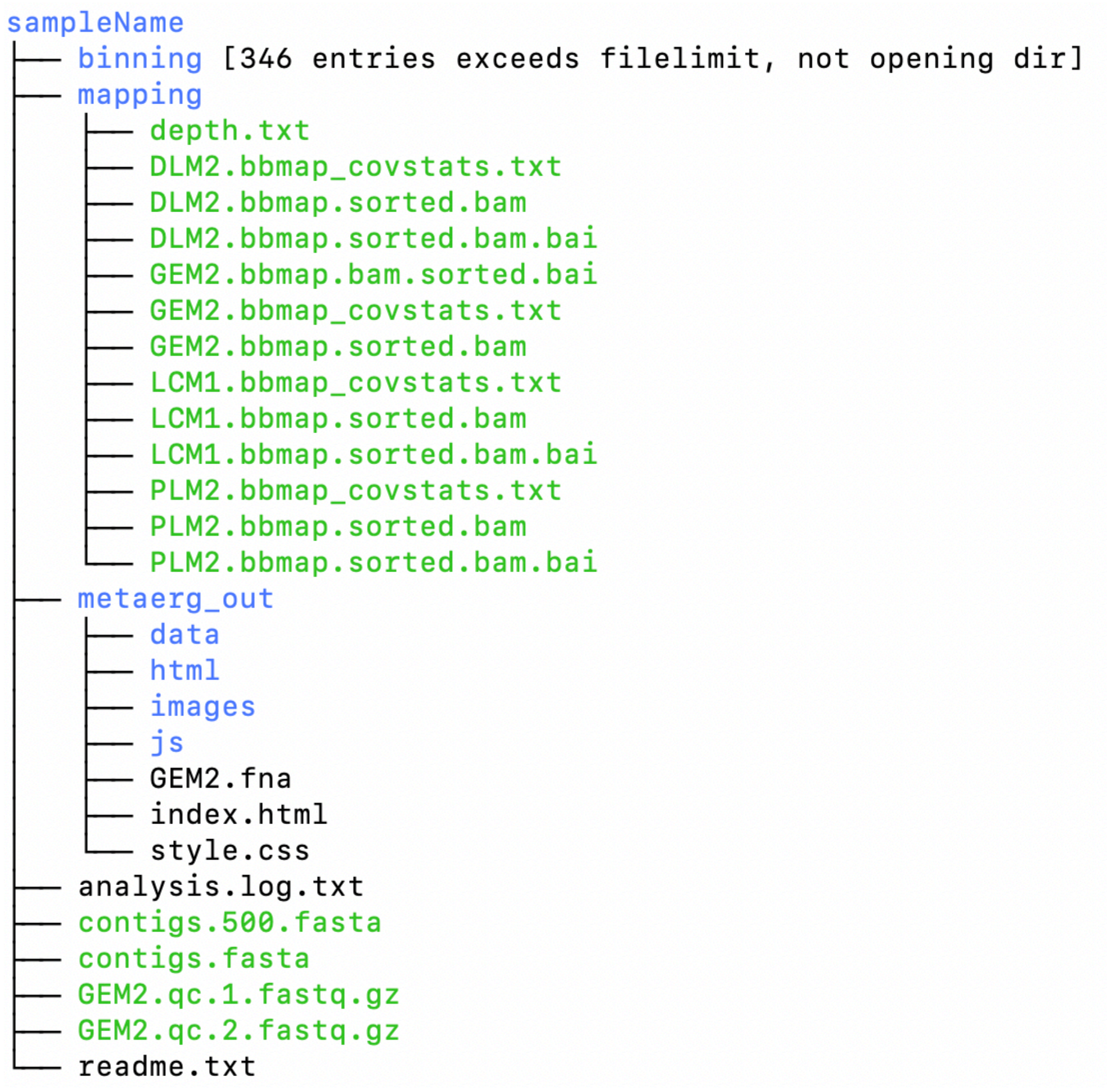
You do not need to re-run the MetaErg pipeline in order to get a metagenome bin annotated. MetaErg provides scripts to extract annotation and to generate html reports for a subset of contigs from the total annotation generated in step 8. See the example below on how to generate MetaErg output for a metagenome bin:

```
# Firstly, extracting gff subset from the metaerg output
fastaContig2Gff.pl -c resA1/Bin.1.fa -g metaerg_out/data/master.gff  >
resA1/Bin.1.gff

# Secondly,  generate metaerg output and report
output_reports.pl  -g resA1/Bin.1.gff -f resA1/Bin.1.fa -o Bin1_metaerg_out
```

Step 10: Data archiving

Metagenome sequence analysis generates a lot of intermediate output files and those files usually occupy a huge amount of storage space. To save storage space, it is necessary to delete those intermediate files and compress the large quality-controlled sequence files. In EBG, it's mandatory that all group members archive their metagenome sequence analysis results in a predefined structure, delete intermediate output files, and compress the large sequence files. The following is the example on what a final metagenome analysis results should be organized after deleting all the intermediate files.



```
sampleName
├── binning [346 entries exceeds filelimit, not opening dir]
├── mapping
│   ├── depth.txt
│   ├── DLM2.bbmap_covstats.txt
│   ├── DLM2.bbmap.sorted.bam
│   ├── DLM2.bbmap.sorted.bam.bai
│   ├── GEM2.bbmap.bam.sorted.bai
│   ├── GEM2.bbmap_covstats.txt
│   ├── GEM2.bbmap.sorted.bam
│   ├── LCM1.bbmap_covstats.txt
│   ├── LCM1.bbmap.sorted.bam
│   ├── LCM1.bbmap.sorted.bam.bai
│   ├── PLM2.bbmap_covstats.txt
│   ├── PLM2.bbmap.sorted.bam
│   └── PLM2.bbmap.sorted.bam.bai
├── metaerg_out
│   ├── data
│   ├── html
│   ├── images
│   ├── js
│   ├── GEM2.fna
│   ├── index.html
│   └── style.css
├── analysis.log.txt
├── contigs.500.fasta
├── contigs.fasta
├── GEM2.qc.1.fastq.gz
├── GEM2.qc.2.fastq.gz
└── readme.txt
```

In the directory, “analysis.log.txt” text file includes all the step-by-step commands used to get the analysis results and “readme.txt” text file describes the sample and the things that are included in each subdirectory.