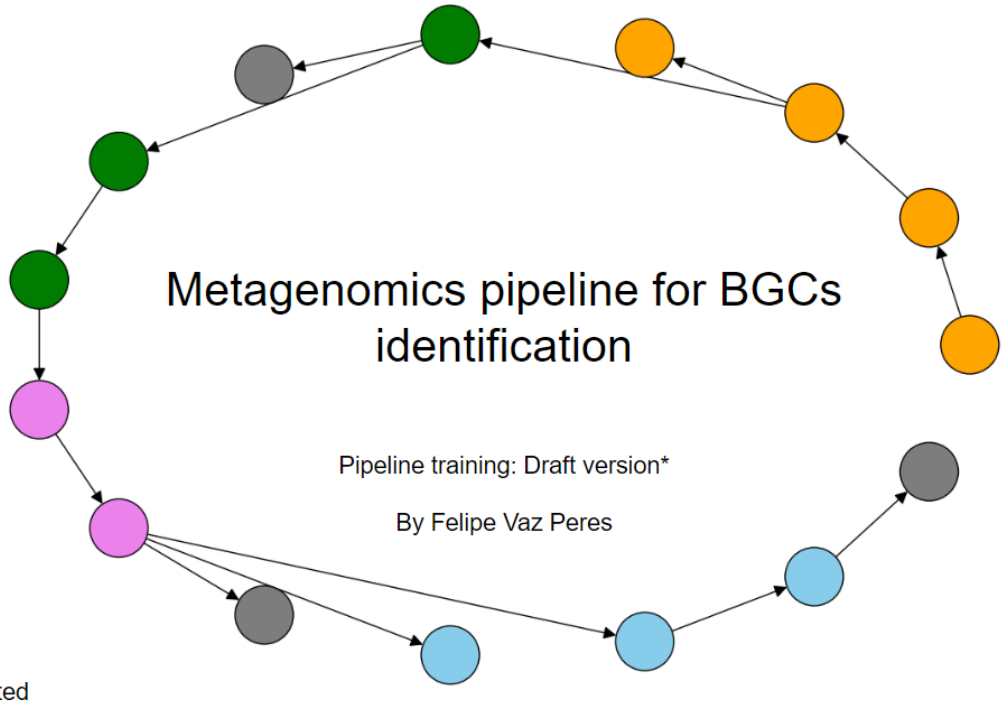

* non-automated

CURRENT WORKFLOW

https://github.com/felipevzps/seabed-symphony

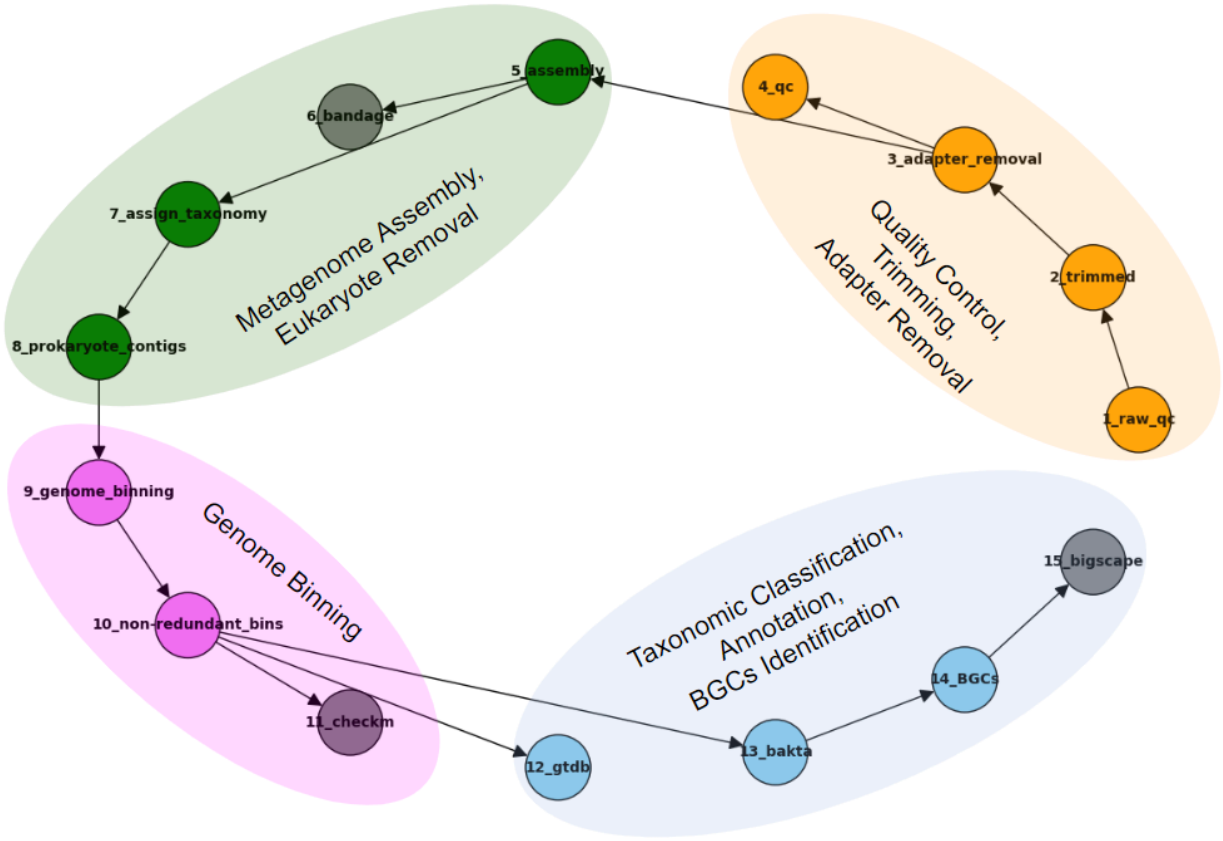

TABLE OF CONTENTS

DOWNLOAD AND INSTALL MINICONDA

```
# download latest version of conda (can be in your dtu home directory)
wget -b https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

# install it
bash Miniconda3-latest-Linux-x86_64.sh

# accept terms of use
```

SETUP THE ENVIRONMENTS

```
# go to your scratch folder - /work3/user/ (make sure you already have it!)
cd /work3/fevape/

# clone the seabed-symphony repository from github
git clone https://github.com/felipevzps/seabed-symphony.git

# installing required softwares and environments
cd seabed-symphony/environments/

conda env create -f antismash.yaml
conda env create -f bakta.yaml
conda env create -f bigscape.yaml
conda env create -f checkm.yaml
conda env create -f graphmb.yaml
conda env create -f gtdbtk.yaml
conda env create -f metagenomics.yaml
conda env create -f whokaryote.yaml

# NOTE: The installation of all environments may take a few minutes.
```

RUNNING THE PIPELINE

Now that all environments are installed, you can run the pipeline. The DTU Cluster uses the LSF to manage job submissions, so all scripts need to be submitted with the header containing submission instructions, for example:

```
nano job_test.sh

# -- script --
#!/bin/sh

#BSUB -q hpc
#BSUB -J job_test
#BSUB -n 1
#BSUB -R "span[hosts=1]"
#BSUB -R "rusage[mem=4GB]"
#BSUB -W 00:01
#BSUB -o job_test.sh.o%J
#BSUB -e job_test.sh.e%J

echo `hostname`
date
sleep 20
date
# -- end script –

# save and close the job_test.sh file
```

Now you can submit this script to the cluster:

```
# switch to another node (dynamically assigned by the system)
# REMEMBER: (you can’t run jobs in the login node)
linuxsh

# how to run (the syntax is bsub < "name_of_script.sh")
bsub < job_test.sh
```

This script will display the current date, pause for 20 seconds, and then display the date again. This information will be recorded in the "**output file**". In my case, the file created is named **job_test.sh.o20689747**.

```
# let’s see the first lines
head job_test.sh.o20689747

n-62-31-3                              # this is the hostname (or de node)
Sat Mar 23 20:54:43 CET 2024           # the date
Sat Mar 23 20:55:03 CET 2024           # the date after 20 seconds!
```

This documentation serves as a guide for running the pipeline for BGC identification. All jobs need to include instructions for the **queue** name to be used, the **number of cores**, the **amount of RAM**, and the **estimated execution time**. However, in this document, I am only showing the execution line of the software and its specific parameters.

You can check the instruction header of each job on the GitHub repository, e.g. script to submit seqkit stats:
https://github.com/felipevzps/seabed-symphony/blob/main/playground/1_raw_qc/scenarioB/seqkit_scenarioB_stats.sh

It is recommended that you copy the scripts for submission to your scratch folder (/work3/user) and only make modifications to the names of the inputs and outputs of the script, for example:

```
cd /work3/user/

mkdir new-analysis && cd new-analysis

ls ../seabed-symphony/playground/

# create the directory for each step
mkdir 10_non-redundant_bins 12_gtdb 14_BGCs 1_raw_qc 3_adapter_removal 5_assembly
7_assign_taxonomy 9_genome_binning 11_ckeckm 13_bakta 15_bigscape 2_trimmed 4_qc
6_bandage 8_prokaryote_contigs

# I recommend to store raw datasets in the 0_raw dir
mkdir 0_raw

# add your own data to 0_raw dir

# now, you can copy the scripts to each directory
cd 1_raw_qc
cp ../../seabed-symphony/playground/1_raw_qc/scenarioB/seqkit_scenarioB_stats.sh .

# you are free to rename the script as you wish!
mv seqkit_scenarioB_stats.sh seqkit_new-analysis_stats.sh

# NOTE: remember to update the inputs and outputs of seqkit_new-analysis_stats.sh
with your own files.
```

OBS: In this document, the lines in red in all scripts should be modified to use the names of your files (desired inputs and outputs).

QUALITY CONTROL

NANOPLOT (RAW READS)

- Histogram of read length
- Bivariate plot of length against base call quality (dot, kernel density)
- Cumulative yield plot

Run NanoPlot

```
# -- script --
source activate metagenomics

/usr/bin/time -v NanoPlot -t 5 --fastq ../0_raw/input.fastq --plots dot --legacy
dot --N50 -o nanoplot_output
# -- end script --

# run nanoplot.sh
bsub < nanoplot.sh
```

General summary - example (nanoplot_output/NanoStats.txt)

```
Mean read length: 3,467.6
Mean read quality: 13.6
Median read length: 2,769.0
Median read quality: 15.1
Number of reads: 500,000.0
Read length N50: 5,133.0
STDEV read length: 3,255.6
```

SEQKIT (RAWREADS)

Ultrafast toolkit for FASTA/Q file manipulation

Run SeqKit

```
# -- script --
source activate metagenomics

/usr/bin/time -v seqkit stats ../0_raw/input.fastq >> input_stats.txt
# -- end script --

# run seqkit_stats.sh
bsub < seqkit_stats.sh
```

General summary - example (input_stats.txt)

```
file                format format type num_seqs  sum_len  min_len avg_len max_len
../0_raw/input.fastq FASTQ  DNA  1,336,282 4,674,459,047 38 3,498.1 204,582
```

FILTLONG

Filtlong is a tool for filtering long reads by quality. It can take a set of long reads and produce a smaller, better subset. It uses both read length (longer is better) and read identity (higher is better) when choosing which reads pass the filter.

QC recommendations[1]

For ONT read QC, I recommend throwing out reads too short to be useful.

- A low threshold (e.g. **discarding reads below 1 kbp**) will **ensure that any small plasmid reads are retained**.
- If your pre-QC read set has a poor N50, a higher threshold (e.g. **10 kbp**) may be appropriate, **but small plasmids might be excluded**.

There is a trade-off between post-QC depth and N50:

- Low thresholds favour depth and high thresholds favour N50.

Both depth and N50 are important for assembly, so choose a value appropriate for your reads – e.g. if you have plenty of depth you can use a higher threshold. I also recommended discarding the lowest quality ONT reads. You can rely on the ONT sequencer/basecaller to do this filtering (e.g. only using 'pass' reads and not 'fail' reads).

Run Filtlong

```
# -- script --
source activate metagenomics

mkdir soft_filter

/usr/bin/time -v filtlong --min_length 1000 ../0_raw/input.fastq >>
soft_filter/input.trimmed.fastq
# -- end script --

# run filtlong.sh
bsub < filtlong.sh
```

[1] https://github.com/rrwick/Perfect-bacterial-genome-tutorial/wiki/Tutorial-(hard)

Run Filtlong again to throw out the worst 10% of reads

```
# -- script --
source activate metagenomics

/usr/bin/time -v filtlong --keep_percent 90 soft_filter/input.trimmed.fastq >>
soft_filter/input.trimmed.highquality.fastq

rm -f soft_filter/input.trimmed.fastq
# -- end script --

# run filtlong_highquality.sh
bsub < filtlong_highquality.sh
```

PORECHOP

Finding and removing adapters from ONT reads

Run Porechop

```
# -- script --
source activate metagenomics

mkdir soft_filter

/usr/bin/time -v porechop -i
../../2_trimmed/dataset/soft_filter/input.trimmed.highquality.fastq -o
soft_filter/input.trimmed.highquality.noadapter.fastq --threads 5
# -- end script --

# run porechop.sh
bsub < porechop.sh
```

NANOPLOT (TRIMMED READS)

Run NanoPlot again (to compare raw with trimmed reads)

```
# -- script --
source activate metagenomics

/usr/bin/time -v NanoPlot -t 5 --fastq
../../3_adapter_removal/dataset/soft_filter/input.trimmed.highquality.noadapter.fastq --plots dot --legacy dot --N50 -o nanoplot_input
# -- end script --

# run nanoplot.sh
bsub < nanoplot.sh
```

SEQKIT (TRIMMED READS)

Run SeqKit again (to compare raw with trimmed reads)

```
# -- script --
source activate metagenomics

/usr/bin/time -v seqkit stats
../../3_adapter_removal/dataset/soft_filter/input.trimmed.highquality.noadapter.fastq >> seqkit_dataset_stats.txt
# -- end script --

# run seqkit_stats.sh
bsub < seqkit_stats.sh
```

METAGENOME ASSEMBLY

We are using meta flye to assemble genomes from metagenomic datasets

The main differences are that "regular flye" assumes a relatively uniform coverage of the assembled genome and makes certain desicions based on that.

The metagenome mode is more general in this respect, and works well for assembly of complex microbial communities with **highly non-uniform coverage and richer repeat content**. It is sensitive to very short sequences and underrepresented organisms at low read coverage (as low as 3x).

Run meta flye

```
# -- script --
source activate metagenomics

/usr/bin/time -v flye --meta --nano-hq
../../3_adapter_removal/dataset/soft_filter/input.trimmed.highquality.noadapter.fastq --out-dir dataset --iterations 2 --threads 10
# -- end script --

# run flye_assembly.sh
bsub < flye_assembly.sh
```

General summary - example (flye_assembly_dataset.sh.e*)

```
INFO: Assembly statistics:
Total length: 23440031
Fragments: 1201
Fragments N50: 25696
Largest frg: 285942
Scaffolds: 0
Mean coverage: 5
```

BANDAGE

Bandage facilitates interaction with de Bruijn graphs made by *de novo* assemblers such as flye. It displays the graph in a graphical user interface (GUI).

Paper: https://academic.oup.com/bioinformatics/article/31/20/3350/196114

Documentation: https://github.com/rrwick/Bandage

Download and run Bandage

```
# Download and install bandage in your computer - follow the instructions
https://rrwick.github.io/Bandage/

# Open bandage
./Bandage

# Or open it by double clicking the executable (if using Windows)

# 1 - Select "Load Graph"
# 2 - Find the file "assembly_graph.gfa"
# 3 - Press "Draw Graph"

# NOTE:
The file "assembly_graph.gfa" are located on the meta flye output directory in the
cluster
So you must transfer this file to your computer before running Bandage.
```

ASSIGN TAXONOMY

Whokaryote uses a **random forest classifier** that uses gene-structure based features and optionally Tiara predictions to predict whether a contig is from a eukaryote or from a prokaryote.

Why not kraken2? → Kraken2 is good at assigning taxonomy to metagenomic contigs up to the species level, but use k-mers (Kraken2) and thus require the use of large pre-existing databases. Uncultivated organisms are frequently missed because they are not yet in a database. **Good point for unknown organisms in marine sediments**.

Whokaryote features

- **intergenic distance**: Genes in prokaryotes are often packed closely together on the genome in co-regulated operons that are transcribed to a single polycistronic mRNA. Therefore, we expected genes on eukaryotic contigs to generally have a higher intergenic distance .
- **gene density**: We also expected genes on euk contigs to have a lower gene density than prokaryotic contigs
- **gene length**
- **ratio genes in same orientation**
- **RBS motif ratio**: ribosome-binding site (RBS) these motifs are prokaryote-specific, we expected that on average, Prodigal would find such RBS motifs more frequently on prokaryotic contigs than on eukaryotic contigs.
- **k-mer counts (Tiara)**: Tiara, a classifier that uses k-mer counts as features for deep learning models - Whokaryote+Tiara

Run whokaryote

```
# -- script --
source activate whokaryote

/usr/bin/time -v whokaryote.py --contigs ../5_assembly/dataset/assembly.fasta
--outdir dataset --threads 4
# -- end script --

# run whokaryote.sh
bsub < whokaryote.sh
```

EXTRACT PROKARYOTE CONTIGS

I developed extractContigsFromWhokaryote.py to extract prokaryote/eukaryote contigs from multi-fasta assembly (classified by whokaryote+tiara). **It is faster than the whokaryote approach**.

Run my script to extract prokaryote contigs from Whokaryote output

```
# -- script --
source activate metagenomics

/usr/bin/time -v .././extractContigsFromWhokaryote.py -i
../../5_assembly/dataset/dataset/assembly.fasta -p
../../7_assign_taxonomy/dataset/scenarioB/prokaryote_contig_headers.txt -o
prokaryote_dataset
# -- end script --

# run extractProkaryoteContigsFromWhokaryote.sh
bsub < extractProkaryoteContigsFromWhokaryote.sh
```

GENOME BINNING

Binning refers to the separation of metagenomic sequences into "bins" or groups representing distinct microbial populations.

Our aim is to group sequences that likely belong to the same organism or related organisms, based on characteristics such as nucleotide composition, read coverage, genetic similarity, and genomic context. To achieve this, we are using 4 genome binning tools (MetaBAT2, MaxBin2, CONCOCT and GraphMB).

MAPPING READS TO ASSEMBLY

In this step we generate the mapping file to be used to estimate the each bin coverage.

Run mapping

```
# creating directories to for genome binning tool
mkdir 9_genome_binning && cd 9_genome_binning
mkdir CONCOCT  GraphMB  mapping  MaxBin2  MetaBAT2

# go to mapping dir
cd mapping

# -- script --
source activate metagenomics

in=`ls -1
../../../3_adapter_removal/dataset/soft_filter/input.trimmed.highquality.noadapter.
fastq`
assembly=`ls -1
../../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fasta`

# Reformat long reads to max 600 length
# trd: Trim the names of reads after the first whitespace
/usr/bin/time -v reformat.sh in=$in out=mapped_reads_dataset.fasta fastareadlen=600
trd
/usr/bin/time -v bbmap.sh ref=$assembly in=mapped_reads_dataset.fasta
out=mapped_dataset.sam threads=10
# -- end script --

# run mappingReadsToAssembly_dataset.sh
bsub < mappingReadsToAssembly_dataset.sh
```

Convert SAM to BAM

```
# -- script --
source activate metagenomics

/usr/bin/time -v samtools view -b -o mapped_dataset.bam mapped_dataset.sam
/usr/bin/time -v samtools sort -o mapped_sorted_dataset.bam mapped_dataset.bam
/usr/bin/time -v samtools index mapped_sorted_dataset.bam
# -- end script --

# run samToBam_scenarioB.sh
bsub < samToBam_scenarioB.sh
```

METABAT2

It uses coverage and composition to compute a pairwise distance matrix for all contig pairs, with the composition feature based on an empirical posterior probability calculated from a set of reference genomes. A graph-based clustering algorithm is then used to bin the contigs based on their distances, where the contigs are linked according to their similarity scores.

Paper: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6662567/

Documentation: https://bitbucket.org/berkeleylab/metabat/src/master/

In this step we generate the depth file with bins coverage → necessary to genome binning tools.

```
# go to MetaBAT2 dir
cd MetaBAT2

# -- script --
source activate metagenomics

bam=`ls -1 ../../mapping/dataset/mapped_sorted_dataset.bam`

/usr/bin/time -v jgi_summarize_bam_contig_depths --outputDepth depth.txt $bam
# -- end script --

# run bamDepths.sh
bsub < bamDepths.sh
```

Run MetaBAT2

```
# -- script --
source activate metagenomics

assembly=`ls -1
../../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fasta`

mkdir bins

/usr/bin/time -v metabat2 -i $assembly -a depth.txt -o bins/bins -t 5
# -- end script --

# run metabat.sh
bsub < metabat.sh
```

MAXBIN2

MaxBin2 recovers genomes from co-assembly of multiple metagenomic samples by exploiting contig coverage levels across multiple metagenomic datasets.

Paper: https://academic.oup.com/bioinformatics/article/32/4/605/1744462

Documentation:
https://denbi-metagenomics-workshop.readthedocs.io/en/latest/binning/maxbin.html

It uses the depth file created before when we run MetaBAT2!

Run MaxBin2

```
# go to MaxBin2 dir
cd MaxBin2

# -- script --
source activate metagenomics

reads=`ls -1
../../../3_adapter_removal/dataset/soft_filter/input.trimmed.highquality.noadapter.
fastq`
assembly=`ls -1
../../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fasta`
depth=`ls -1 ../../MetaBAT2/dataset/depth.txt`

mkdir bins

/usr/bin/time -v run_MaxBin.pl -contig $assembly -abund $depth -out bins/bins
-thread 5
# -- end script --

# run maxbin.sh
bsub < maxbin.sh
```

CONCOCT

Unsupervised binning of metagenomic contigs by using nucleotide composition - kmer frequencies - and coverage data for multiple samples.

Paper: https://www.nature.com/articles/nmeth.3103

Documentation: https://concoct.readthedocs.io/en/latest/

Run CONCOCT

```
# go to CONCOCT dir
cd CONCOCT

# -- script --
source activate metagenomics

assembly=`ls -1
../../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fasta`

# Cut contigs into smaller parts
/usr/bin/time -v cut_up_fasta.py $assembly -c 10000 -o 0 --merge_last -b
contigs_10K.bed > contigs_10K.fa

# Generate table with coverage depth
/usr/bin/time -v concoct_coverage_table.py contigs_10K.bed
../../mapping/dataset/mapped_sorted_dataset.bam > coverage_table.tsv

# Run COCOCT
/usr/bin/time -v concoct --composition_file contigs_10K.fa --coverage_file
coverage_table.tsv -b concoct_output/ -t 5

# Merge subcontig clustering into original contig clustering:
/usr/bin/time -v merge_cutup_clustering.py concoct_output/clustering_gt1000.csv >
concoct_output/clustering_merged.csv

# Extract bins as individual FASTA
mkdir bins
/usr/bin/time -v extract_fasta_bins.py $assembly
concoct_output/clustering_merged.csv --output_path bins
# -- end script --

# run concoct.sh
bsub < concoct.sh
```

GRAPHMB

GraphMB takes advantage of graph machine learning algorithms and the assembly graph generated during assembly. It has been tested on (meta)flye assemblies.

Paper: https://academic.oup.com/bioinformatics/article/38/19/4481/6668279?login=false

Documentation: https://github.com/MicrobialDarkMatter/GraphMB

Spotted Issue

I found some errors trying to run GraphMB, as the following one:

```
# Error:
  File
"/home/felipe.peres/.conda/envs/metagenomics/lib/python3.8/site-packages/graphmb/main.py", line 499, in main
    vae_embs, _ = train_ccvae.run_model_ccvae(dataset, args, logger, 0,
ValueError: too many values to unpack (expected 2)
```

After debugging the source code, I managed to fix this and two other errors that I found. So to facilitate this step, you can run the corrected GraphMB by copying the fixed source code directly from my directory on the cluster.

```
# go to the lib directory of your conda environments
cd ~/bin/miniconda3/envs/graphmb/lib/python3.8/site-packages/

# remove the graphmb directory (source code with errors)
rm -rf graphmb

# copy my fixed version of graphmb to your own dir
cp -r
/zhome/56/5/209982/bin/miniconda3/envs/graphmb/lib/python3.8/site-packages/graphmb
.

# done!
```

Prepare the files before running GraphMB

```
# go to GraphMB dir
cd GraphMB

# create an assembly dir
mkdir assembly && cd assembly

# make symbolic links to assembly graph, assembly fasta (prokaryotes) and depth
file
ln -s ../../../../5_assembly/dataset/dataset/assembly_graph.gfa .
ln -s
../../../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fas
ta .
ln -s ../../../MetaBAT2/dataset/depth.txt .
```

Run GraphMB

```
# go to GraphMB dir
cd GraphMB

# -- script --
source activate graphmb

assembly=assembly

/usr/bin/time -v graphmb --assembly $assembly --outdir bins/ --assembly_name
assembly_prokaryote.fasta --graph_file assembly_graph.gfa --depth depth.txt
--numcores 10 --contignodes --writebins
# -- end script --

# run graphmb
bsub < graphmb.sh
```

FIND NON-REDUNDANT BINS

DAS Tool is an automated method that integrates the results of a flexible number of binning algorithms to calculate an optimized, non-redundant set of bins from a single assembly

Manual file formatting

```
# create directory to run DAS Tool
mkdir 10_non-redundant_bins && cd 10_non-redundant_bins
mkdir dataset && cd dataset

# Using helper script to convert contigs2bin
mkdir sample_data

# make sure you have runFastaToContig2Bin.sh and Fasta_to_Contig2Bin.sh in this
directory

# just run runFastaToContig2Bin.sh to prepare the input files to DAS Tool (create
contigs to bin files)
bash runFastaToContig2Bin.sh

# we have to do an additional step to get graphmb contigs to bin
cd sample_data

# copy the GraphMB contigs to bins and rename it (GraphMB.contigs2bin.tsv)
cp ../../../9_genome_binning/GraphMB/dataset/bins/graphmb_0_best_contig2bin.tsv
GraphMB.contigs2bin.tsv

# make sure that ALL files only have contigs and bins (e.g we don't want headers)

# you can easily see if we have headers:
head *

==> CONCOCT.contigs2bin.tsv <==
contig_1        CONCOCT.11
contig_10       CONCOCT.11




==> GraphMB.contigs2bin.tsv <==
@Version:0.9.0
@SampleID:SAMPLEID
@@SEQUENCEID    BINID
contig_1        723
contig_1        723
```

```
==> MaxBin2.contigs2bin.tsv <==
contig_1080     bins.001
contig_1124     bins.001

==> MetaBAT2.contigs2bin.tsv <==
contig_1965     bins.1
contig_341      bins.1

# for me, only GraphMB.contigs2bin.tsv have headers, so we have to open it and
delete the first 3 lines:
vi GraphMB.contigs2bin.tsv
# delete the first 3 lines (containing headers)

# make sure you removed the header fom all contigs to bins files
# done!
```

Run DAS Tool to generate non-redundant bins from bins inferred by MetaBAT, MaxBin, CONCOCT and GraphMB.

```
# -- script --
source activate metagenomics

assembly=`ls -1
../../8_prokaryote_contigs/dataset/prokaryote_dataset/assembly_prokaryote.fasta`

mkdir output

/usr/bin/time -v DAS_Tool -i
sample_data/MetaBAT2.contigs2bin.tsv,sample_data/MaxBin2.contigs2bin.tsv,sample_dat
a/CONCOCT.contigs2bin.tsv,sample_data/GraphMB.contigs2bin.tsv -l
MetaBAT2,MaxBin2,CONCOCT,GraphMB -c $assembly -o output/DASToolRun -t 5
--write_bin_evals --write_bins --score_threshold 0.5
# -- end script --

# run dastool.sh
bsub < dastool.sh
```

EVALUATE BINS QUALITY

We are using CheckM. It provides a set of tools for assessing the quality of genomes recovered from isolates, single cells, or metagenomes.

Paper: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4484387/

Documentation: https://github.com/Ecogenomics/CheckM

Run CheckM

```
# go to CheckM dir
cd 11_ckeckm/

# -- script --
source activate checkm

bins=../../10_non-redundant_bins/scenarioB/output/DASToolRun_DASTool_bins/

/usr/bin/time -v checkm lineage_wf -x fa -t 10 --tab_table $bins output/ -f
output/checkm.tsv
# -- end script --

# run checkm.sh
bsub < checkm.sh
```

TAXONOMIC CLASSIFICATION

GTDB-Tk is a software toolkit for assigning objective taxonomic classifications to bacterial and archaeal genomes based on the Genome Database Taxonomy GTDB. It is designed to work with recent advances that allow hundreds or thousands of metagenome-assembled genomes (MAGs) to be obtained directly from environmental samples. It can also be applied to isolate and single-cell genomes.

Get familiar with the approach:
https://academic.oup.com/bioinformatics/article/38/23/5315/6758240?login=false

Install GTDB-Tk2 reference Database

```
# go to your work folder and create a database dir
cd /work3/user

mkdir database && cd database
mkdir gtdb && cd gtdb

# download and setup GTDB-Tk reference data - This can take a long time (for me it
took 6h50 hours)
wget -b
https://data.gtdb.ecogenomic.org/releases/release214/214.0/auxillary_files/gtdbtk_r
214_data.tar.gz

# decompress the database
tar -xvzf gtdbtk_r214_data.tar.gz

# setup the path to databse
conda env config vars set GTDBTK_DATA_PATH="/work3/fevape/database/gtdb/release214"
```

Run GTDB-Tk2 → Running Classify_wf genomes
https://ecogenomics.github.io/GTDBTk/commands/classify.html

```
# -- script --
source activate gtdbtk-2.3.2

bins=../../10_non-redundant_bins/dataset/output/DASToolRun_DASTool_bins/

/usr/bin/time -v gtdbtk classify_wf --genome_dir $bins --out_dir output --mash_db
mash_db --extension fa --cpus 30
# -- end script --

# run gtdb.sh
bsub < gtdb.sh
```

GENOME ANNOTATION

Bakta is a tool for the rapid & standardized annotation of bacterial genomes and plasmids from both isolates and MAGs.

Install database

```
# load environment
conda activate bakta

# go to your database folder created before
cd /work3/user/database

mkdir bakta && cd bakta

# To download the most recent compatible database version we recommend to use the
internal database download & setup tool
bakta_db download --output bakta --type full

conda deactivate

# export the database path to your user variables
export BAKTA_DB=/work3/user/database/bakta/db
```

Run bakta annotation

```
# -- script --
source activate bakta

genome_dir=../../10_non-redundant_bins/dataset/output/DASToolRun_DASTool_bins/
genome=`ls -1 $genome_dir | head -n $LSB_JOBINDEX | tail -n 1`
output=`basename ${genome/.fa}`

/usr/bin/time -v bakta --db /work3/fevape/database/bakta/db/ --prefix $output
--output output/$output --meta --threads 10 --keep-contig-headers
$genome_dir/$genome
# -- end script --

# run bakta.sh
bsub < bakta.sh
```

BGCs IDENTIFICATION

The antiSMASH framework allows the detection of clusters of co-occurring biosynthesis genes in genomes, called **B**iosynthetic **G**ene **C**lusters (**BGCs**). BGCs often contain all the genes required for the biosynthesis of one or more **N**atural **P**roducts (**NPs**), also known as **specialized** or **secondary metabolites**.

Install antiSMASH databases

```
# load environment
conda activate antismash

# go to your database folder created before
cd /work3/user/database

mkdir antismash && cd antismash

# download databases
download-antismash-databases --database-dir antismash
```

Run antiSMASH

```
# -- script --
source activate antismash

bakta_annot=../../13_bakta/dataset/output/
genome=`ls -1 $bakta_annot | head -n $LSB_JOBINDEX | tail -n 1`
genbank=../../13_bakta/dataset/output/${genome}/${genome}.gbff

mkdir -p output
mkdir output/$genome

/usr/bin/time -v antismash $genbank --databases /work3/fevape/database/antismash -t
bacteria -c 20 --output-dir output/$genome --output-basename $genome --clusterhmmer
--cb-general --cb-knownclusters --pfam2go
# -- end script --

# run antismash.sh
bsub < antismash.sh
```

BGCs VISUALIZATION

BiG-SCAPE provide a set of tools to explore the diversity of biosynthetic gene clusters (BGCs) across large numbers of genomes by constructing BGC sequence similarity networks, grouping BGCs into gene cluster families and exploring gene cluster diversity linked to enzyme phylogenies.

```
Install PFAM databases
# load environment
conda activate bigscape

# go to your database folder created before
cd /work3/user/database

mkdir pfam && cd pfam

wget -b https://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/Pfam-A.hmm.gz

# decompress
gunzip Pfam-A.hmm.gz

# prepare HMM database for hmmscan
hmmpress Pfam-A.hmm

conda deactivate bigscape
```

Prepare genbank files before running bigscape

```
# run linkAndChangeClustersName.sh to prepare the input files

mkdir 15_bigscape && cd 15_bigscape
mkdir dataset && cd dataset
mkdir gbks && cd gbks

# make sure you have a copy of this script (it is available on my github repo)
#
https://github.com/felipevzps/seabed-symphony/blob/main/playground/15_bigscape/link
AndChangeClustersName.sh

bash linkAndChangeClustersName.sh

# done!
# back dataset dir
cd ..
```

Run bigscape

```
# -- script --
source activate bigscape

genbank=gbks

/usr/bin/time -v bigscape -i $genbank -o output --pfam_dir
/work3/fevape/database/pfam -c 20 --verbose --include_singletons --mode auto
--mibig
# -- end script --

# run bigscape.sh
bsub < bigscape.sh
```

ADVANCED JOB SUBMISSION

One of the advantages of running jobs on a cluster is the possibility of creating **job chains**, where a job will be executed as soon as the execution of a previous job is completed. This is really cool because we can adapt the scripts and create a job chain without having to wait for a job to finish.

For instance, considering a workflow A, B, C, where B depends on A and C depends on the output of B, an elegant way to ensure the execution of all three jobs in sequence would be:

```
bsub -w "done(JOB_ID)" < script.sh
```

```
# First, submit job A
bsub < script_job_A.sh

# message from the cluster
Job <100> is submitted to queue <hpc>

# Then, execute job B and request it to be executed only when job A finishes.
bsub -w "done(100)" < script_job_B.sh

# message from the cluster
Job <101> is submitted to queue <hpc>

# Do the same with job C (execute only when job B finishes)
bsub -w "done(101)" < script_job_C.sh

# message from the cluster
Job <102> is submitted to queue <hpc>
```