Metagenomics Workshop Documentation

Release 1

Alex Sczyrba

Jul 14, 2021

CONTENTS

1 Assembling data **3**
- 1.1 Prerequisites . . . 3
- 1.2 Part 1 - Quality control and filtering of the raw sequence files . . . 4
- 1.3 Part 2 - Assembly and Co-assembly . . . 10

2 Assembly analysis **15**
- 2.1 MGnify hands-on exercises . . . 15

3 Viral detection and classification **19**
- 3.1 Prerequisites . . . 19
- 3.2 1. Identification of putative viral sequences . . . 20
- 3.3 2. Detection of viral taxonomic markers . . . 22
- 3.4 3. Viral taxonomic assignment . . . 22

4 MAG generation **25**
- 4.1 Prerequisites . . . 25
- 4.2 Generating metagenome assembled genomes . . . 26

These are the materials that will be used for the "Analysing metagenomic assemblies using MGnify" workshop, carried out as part of the BiATA (Bioinformatics: from Algorithms to Applications) 2021 conference.

We strongly recommend navigating to the 'Prerequisites' sections of each chapter below and completing the download and setup steps before the practical sessions. Depending on your available network it can take some time (1-2 hrs) to download the necessary files.

CHAPTER

ONE

ASSEMBLING DATA

- Sequence quality control (quality/adapter trimming, host decontamination)
- Assembly/co-assembly
- Evaluating an assembly
- Submitting primary assemblies to the ENA

1.1 Prerequisites

This tutorial requires Bandage which can be installed as per the github instructions for your system https://github.com/rrwick/Bandage#pre-built-binaries

For this tutorial you will need to make a working directory to store your data in:

```
mkdir -p ~/BiATA/session1/data
chmod -R 777 ~/BiATA
export DATADIR=~/BiATA/session1/data/session1
```

In this directory, download the tarball from http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/

```
cd  ~/BiATA/session1/data/
wget -q http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/
↪session1.tgz
tar -xzvf session1.tgz
```

Now makes sure that you have pulled the docker container

```
docker pull microbiomeinformatics/biata-qc-assembly:v2021
```

Finally, start the docker container in the following way:

```
xhost +
docker run --rm -it  -e DISPLAY=$DISPLAY  -v $DATADIR:/opt/data -v /tmp/.X11-unix:/tmp/.
↪X11-unix:rw -e DISPLAY=unix$DISPLAY microbiomeinformatics/biata-qc-assembly:v2021
```

Note that if running Docker windows, WSL must be disabled by selecting "Turn Windows features or or off" when Docker starts.

1.2 Part 1 - Quality control and filtering of the raw sequence files


Learning Objectives - in the following exercises you will learn how to assess the quality of short read sequences, identify the presence of adapter sequences, and remove both adapters and low quality sequences. You will also learn how to construct a reference database for decontamination.


First go to your working area. The data that you downloaded has been mounted in `/opt/data` in the docker container.

```
cd /opt/data
ls
```

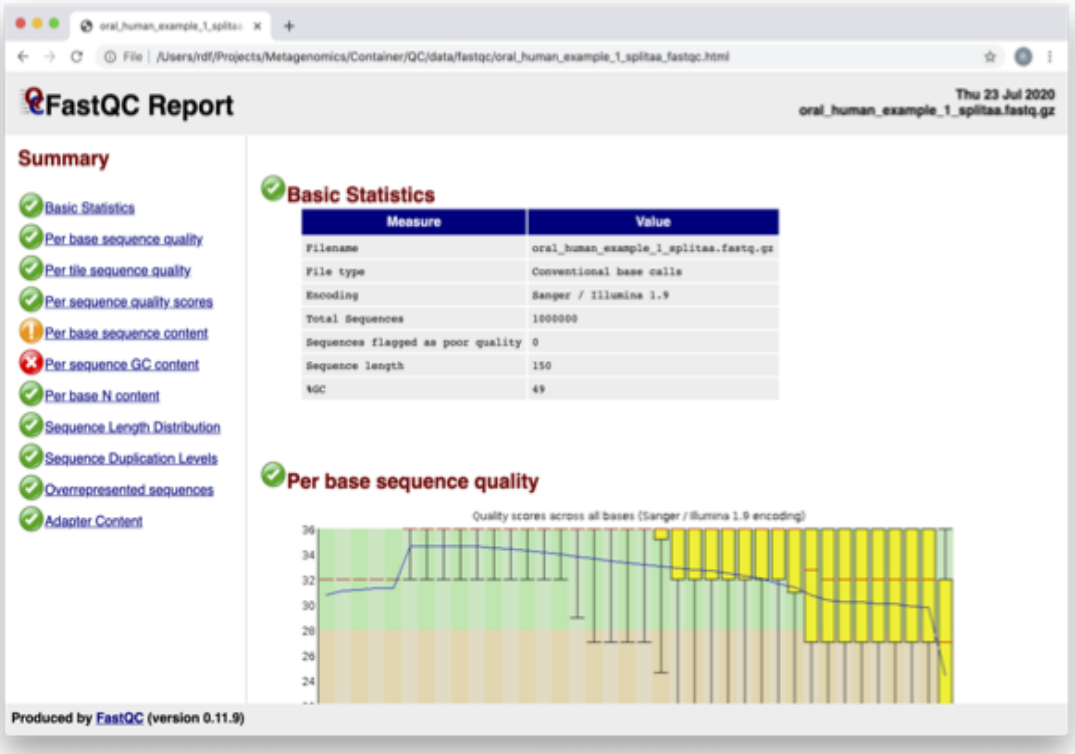

Here you should see the same contents as you had from downloading and uncompressing the session data. As we write into this directory, we should be able to see this from inside the container, and on the filesystem of the computer running this container. We will use this to our advantage as we go through this practical. Unless stated otherwise, all of the following commands should be executed in the terminal running the Docker container.


Generate a directory of the fastqc results

```
cd /opt/data
mkdir fastqc_results
cd fastqc_results
fastqc ../oral_human_example_1_splitaa.fastq.gz
fastqc ../oral_human_example_2_splitaa.fastq.gz
mv ../*html .
mv ../*zip .
```


Now on your **local** computer, go to the browser, and `File -> Open File`. Use the file navigator to select the following file

~/BiATA/session1/data/fastqc_results/oral_human_example_1_splitaa_fastqc.html



Spend some time looking at the 'Per base sequence quality'.

For each position a BoxWhisker type plot is drawn. The elements of the plot are as follows:

- The central red line is the median value
- The yellow box represents the inter-quartile range (25-75%)
- The upper and lower whiskers represent the 10% and 90% points
- The blue line represents the mean quality

The y-axis on the graph shows the quality scores. The higher the score the better the base call. The background of the graph divides the y axis into very good quality calls (green), calls of reasonable quality (orange), and calls of poor quality (red). The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the orange area towards the end of a read.

What does this tell you about your sequence data? Where do the errors start?

In the pre-processed files we see two warnings, as shown on the left side of the report. Navigate to the "Per bases sequence content"

At around 15-19 nucleotides, the DNA composition becomes very even but at the 5' end of the sequence there are distinct differences. Why do you think that is?

Open up the FastQC report corresponding to the reversed reads.

Are there any significant differences between the forward and reverse reads?

For more information on the FastQC report, please consult the 'Documentation' available from this site: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

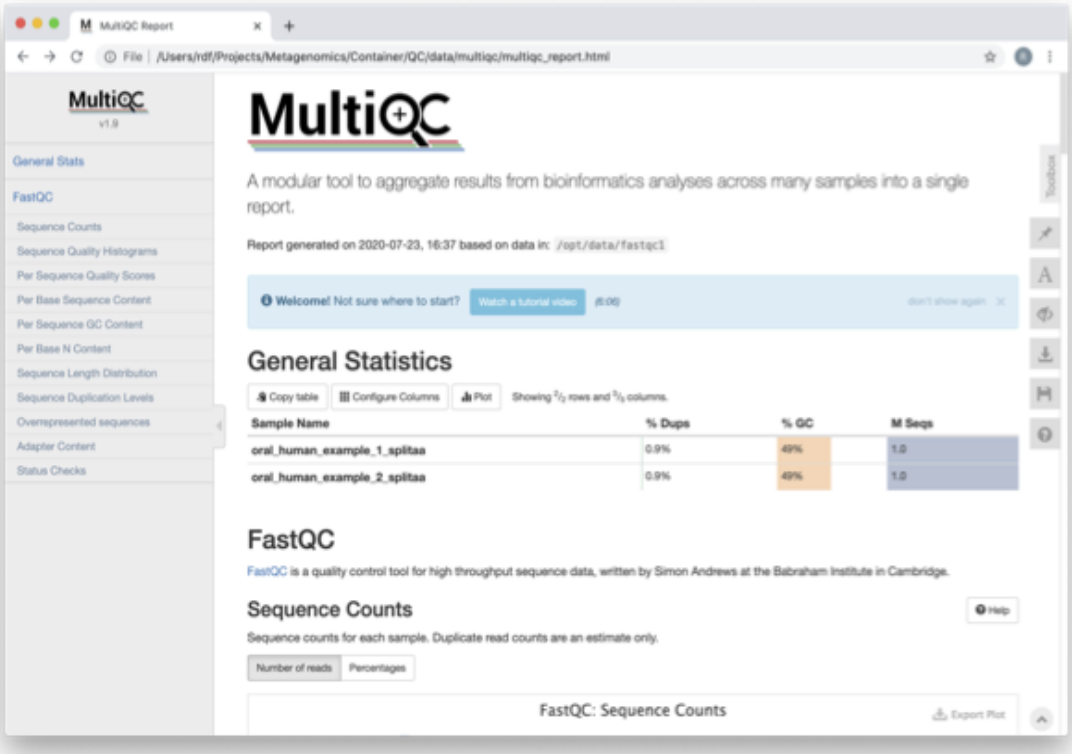
We are currently only looking at two files but often we want to look at many files. The tool multiqc aggregates the FastQC results across many samples and creates a single report for easy comparison. Here we will demonstrate the use of this tool

```
cd /opt/data
mkdir multiqc_results
multiqc fastqc_results -o multiqc_results
```

In this case, we provide the folder containing the fastqc results to multiqc and the -o allows us to set the output directory for this summarised report.

Now on your **local** computer, open the summary report from MultiQC. To do so, go to your browser, and use `File -> Open File`. Use the file navigator to select the following file

~/BiATA/session1/data//multiqc_results/multiqc_report.html



Scroll down through the report. The sequence quality histograms show the following results from each file as two separate lines. The 'Status Checks' show a matrix of which samples passed check and which ones have problems.

What fraction of reads are duplicates?

So, far we have looked at the raw files and assessed their content, but we have not done anything about removing sequences with low quality scores or adapters. So, lets start this process. The first step in the process is to make a database relevant for decontaminating the sample. It is always good to routinely screen for human DNA (which may come from the host and/or staff performing the experiment). However, if the sample is say from mouse, you would want to download the the mouse genome instead.

In the following exercise, we are going to use two "genomes" already downloaded for you in the decontamination folder. To make this tutorial quicker and smaller in terms of file sizes, we are going to use PhiX (a common spike in) and just chromosome 10 from human.

```
cd /opt/data/decontamination
```

For the next step we need one file, so we want to merge the two different fasta files. This is simply done using the command line tool cat.

```
cat phix.fasta GRCh38_chr10.fasta > GRCh38_phix.fasta
```

Now we need to build a bowtie index for them:

```
bowtie2-build GRCh38_phix.fasta GRCh38_phix.index
```

 It is possible to automatically download a pre-indexed human genome in Bowtie2 format using the following command (but do not do this now, as this will take a while to download):

```
kneaddata_database –download human_genome bowtie2
```

 Now we are going to use the *GRCh38_phix* database and clean-up our raw sequences. kneaddata is a helpful wrapper script for a number of pre-processing tools, including Bowtie2 to screen out contaminant sequences, and Trimmomatic to exclude low-quality sequences. We also have written wrapper scripts to run these tools (see below), but using kneaddata allows for more flexibility in options.

```
cd /opt/data/
mkdir clean
```

We now need to uncompress the fastq files.

```
gunzip -c oral_human_example_2_splitaa.fastq.gz > oral_human_example_2_splitaa.fastq
gunzip -c oral_human_example_1_splitaa.fastq.gz > oral_human_example_1_splitaa.fastq

kneaddata --remove-intermediate-output -t 2 --input oral_human_example_1_splitaa.fastq --
↪input oral_human_example_2_splitaa.fastq --output /opt/data/clean --reference-db /opt/
↪data/decontamination/GRCh38_phix.index --trimmomatic-options  "SLIDINGWINDOW:4:20␣
↪MINLEN:50" --bowtie2-options "--very-sensitive --dovetail" --remove-intermediate-output
```

 The options above are:

* **–input**, Input FASTQ file. This option is given twice as we have paired-end data.
* **–output**, Output directory.
* **–reference-db**, Path to bowtie2 database for decontamination.
* **-t**, # Number of threads to use (2 in this case).
* **–trimmomatic-options**, Options for Trimmomatic to use, in quotations ("SLIDINGWINDOW:4:20 MINLEN:50" in this case). See the Trimmomatic website for more options.
* **–bowtie2-options**, Options for bowtie2 to use, in quotations. The options "–very-sensitive" and "–dovetail" set the alignment parameters to be very sensitive and sets cases where mates extend past each other to be concordant (i.e. they will be called as contaminants and be excluded).
* **–remove-intermediate-output**, Intermediate files, including large FASTQs, will be removed.

Kneaddata generates multiple outputs in the "clean" directory, containing different 4 different files for each read.

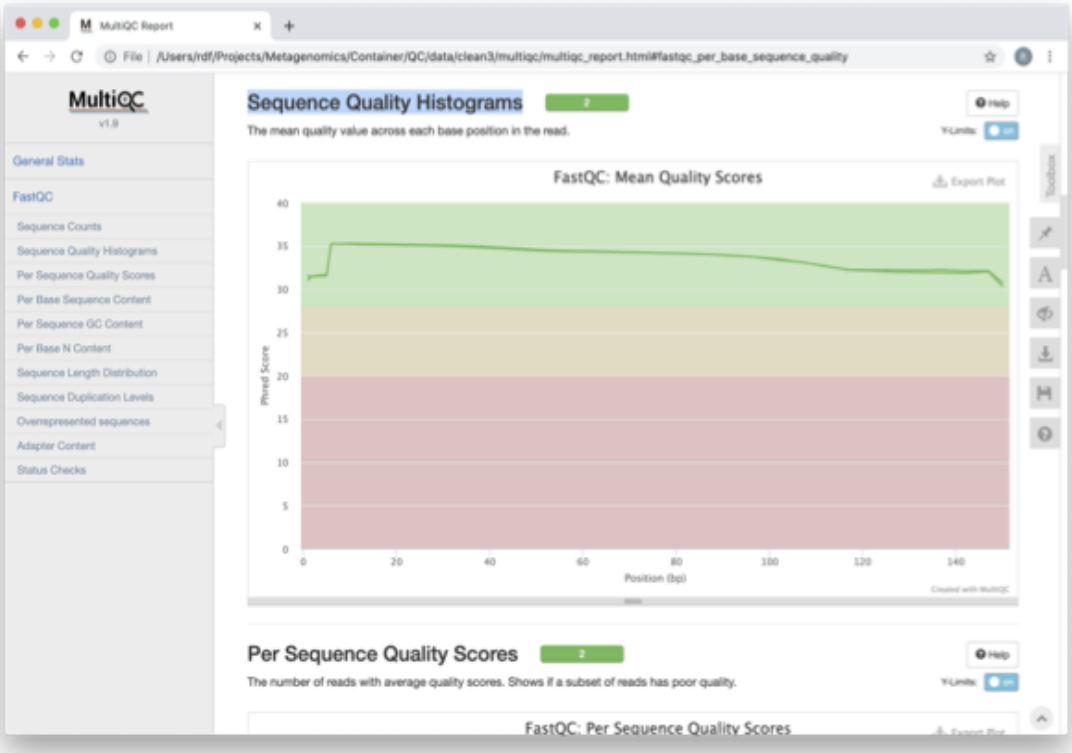
 Using what you have learned previously, generate a fastqc report for each of the *oral_human_example_1_splitaa_kneaddata_paired* files. Do this within the clean directory.

```
cd /opt/data/clean
mkdir fastqc_final
<you construct the command>
```

 Also generate a multiqc report and look at the sequence quality historgrams.

```
cd /opt/data/clean
mkdir multiqc
<you construct the command>
```

View the multiQC report as before using your browser. You should see something like this:



Open the previous MultiQC report and see have they have improved?

Did sequences at the 5' end become uniform? Why might that be? Is there anything that suggests that adapter sequences were found?

To generate a summary file of how the sequence were categorised by Kneaddata, run the following command.

```
cd /opt/data
kneaddata_read_count_table --input /opt/data/clean --output kneaddata_read_counts.txt
less kneaddata_read_counts.txt
```

What fraction of reads were deemed to be contaminating?

The reads have now be decontaminated any can be uploaded to ENA, one of the INSDC members. It is beyond the scope of this course to include a tutorial on how to submit to ENA, but there is additional information available on how to do this in this Online Training guide provided by EMBL-EBI

https://www.ebi.ac.uk/training/online/course/ebi-metagenomics-portal-submitting-metagenomics-da/considerations-submitting-metagenomic-data

1.3 Part 2 - Assembly and Co-assembly


Learning Objectives - in the following exercises you will learn how to perform a metagenomic assembly and to start some basic analysis of the output. Subsequently, we will demonstrate the application of co-assembly. Note, due to the complexity of metagenomics assembly, we will only be investigating very simple example datasets as these often take days of CPU time and 100s of GB of memory. Thus, do not think that there is an issue with the assemblies.

Once you have quality filtered your sequencing reads (see Part 1 of this session), you may want to perform *de novo* assembly in addition to, or as an alternative to a read-based analyses. The first step is to assemble your sequences into contigs. There are many tools available for this, such as MetaVelvet, metaSPAdes, IDBA-UD, MegaHIT. We generally use metaSPAdes, as in most cases it yields the best contig size statistics (i.e. more continguous assembly) and has been shown to be able to capture high degrees of community diversity (Vollmers, et al. PLOS One 2017). However, you should consider the pros and cons of different assemblers, which not only includes the accuracy of the assembly, but also their computational overhead. Compare these factors to what you have available. For example, very diverse samples with a lot of sequence data uses a lot of memory with SPAdes. In the following practicals we will demonstrate the use of metaSPAdes on a small sample and the use of MegaHIT for performing co-assembly.


Using the sequences that you have previously QC-ed, run metaspades. To make things faster, we are going to turn-off metaspades own read error correction method, by specifying the command –only-assembler.

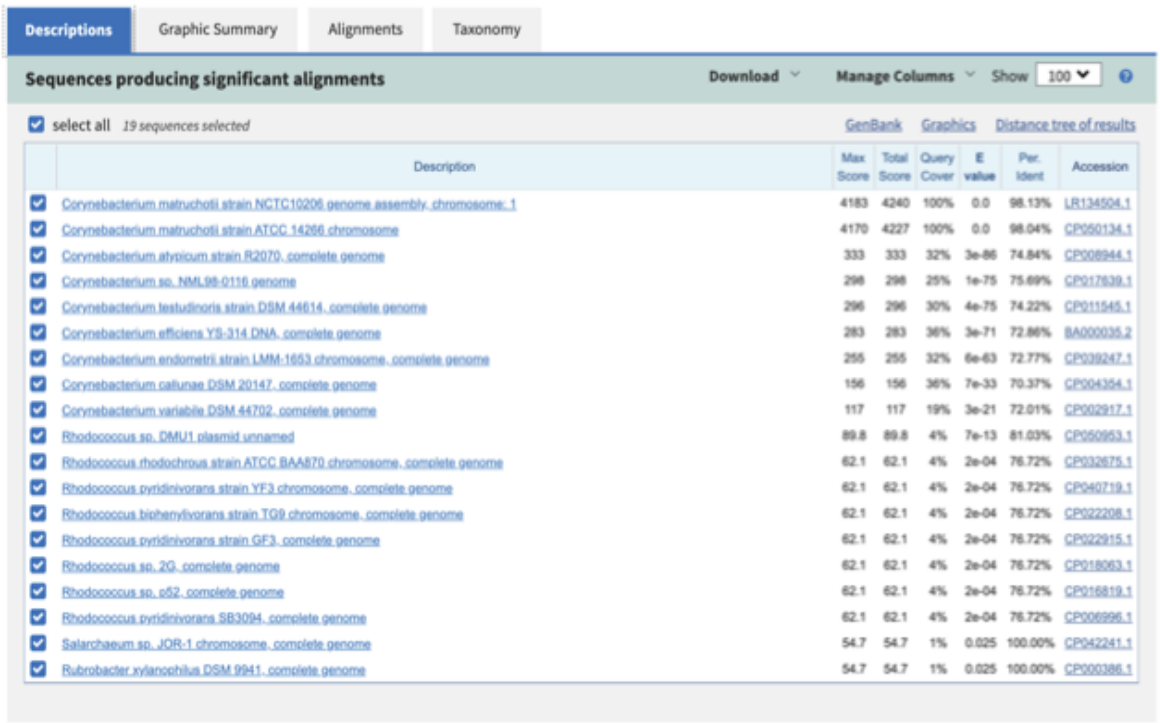
```
cd /opt/data
mkdir assembly
metaspades.py \
        -t 2  \
        --only-assembler \
        -m 10 \
        -1 /opt/data/clean/oral_human_example_1_splitaa_kneaddata_paired_1.fastq \
        -2 /opt/data/clean/oral_human_example_1_splitaa_kneaddata_paired_2.fastq \
        -o /opt/data/assembly
```


This takes about 1 hour to complete.


Once this completes, we can investigate the assembly. The first step is to simply look at the contigs.fasta file.

Now take the first 40 lines of the sequence and perform a blast search at NCBI (https://blast.ncbi.nlm.nih.gov/Blast.cgi, choose Nucleotide:Nucleotide from the set of options). Leave all other options as default on the search page. To select the first 40 lines of sequence perform the following:

```
head -41 contigs.fasta
```



Which species do you think this sequence may be coming from? Does this make sense as a human oral bacteria? Are you surprised by this result at all?

Now let us consider some statistics about the entire assembly

```
cd /opt/data/assembly
assembly_stats scaffolds.fasta
```

This will output two simple tables in JSON format, but it is fairly simple to read. There is a section that corresponds to the scaffolds in the assembly and a section that corresponds to the contigs.

What is the length of longest and shortest contigs?

What is the N50 of the assembly? Given that are input sequences were ~150bp long paired-end sequences, what does this tell you about the assembly?

N50 is a measure to describe the quality of assembled genomes that are fragmented in contigs of different length. We can apply this with some caution to metagenomes, where we can use it to crudely assess the contig length that covers 50% of the total assembly. Essentially the longer the better, but this only makes sense when thinking about alike metagenomes. Note, N10 is the minimum contig length to cover 10 percent of the metagenome. N90 is the minimum contig length to cover 90 percent of the metagenome.

In addition to evaluating the contiguity the assemblies, we can ask what fraction of the diversity in the samples was assembled. We can answer this question by quantifying the number of reads that map to the assembly. BWA expects that the read names in the forward and reverse reads are the same so we will first remove the read identifiers and make sure that they are ordered correctly.

```
sed 's/\/1//g' ../clean/oral_human_example_1_splitaa_kneaddata_paired_1.fastq > ../clean/
↪oral_human_example_1_splitaa_kneaddata_paired_noidentifiers_1.fastq
sed 's/\/2//g' ../clean/oral_human_example_1_splitaa_kneaddata_paired_2.fastq > ../clean/
↪oral_human_example_1_splitaa_kneaddata_paired_noidentifiers_2.fastq
repair.sh in=../clean/oral_human_example_1_splitaa_kneaddata_paired_noidentifiers_1.
↪fastq in2=../clean/oral_human_example_1_splitaa_kneaddata_paired_noidentifiers_2.fastq␣
↪out=../clean/oral_human_example_1_splitaa_kneaddata_paired_noidordered_1.fastq out2=../
↪clean/oral_human_example_1_splitaa_kneaddata_paired_noidordered_2.fastq
```


To calculate the percent reads mapping to the assembly using the flagstat output generated in the previous step, calculate the number of primary alignments (mapped - secondary - supplementary). Then divide the number of primary alignments by the sum of forward and reverse reads to get the fraction of reads mapped.

```
bwa index scaffolds.fasta
bwa mem -t 2 scaffolds.fasta ../clean/oral_human_example_1_splitaa_kneaddata_paired_
↪noidordered_1.fastq ../clean/oral_human_example_1_splitaa_kneaddata_paired_noidordered_
↪2.fastq | samtools view -bS - | samtools sort -@ 2 -o oral_human_example_1_splitaa.sam␣
↪-
samtools flagstat oral_human_example_1_splitaa.sam > oral_human_example_1_splitaa_
↪flagstat.txt
```


To get the total number of reads in the forward read, run the command below and divide by 4. Repeat for the reverse read.

```
wc -l ../clean/oral_human_example_1_splitaa_kneaddata_paired_noidordered_1.fastq
wc -l ../clean/oral_human_example_1_splitaa_kneaddata_paired_noidordered_2.fastq
```


What percent of the reads were incorporated into the assembly? What factors can affect the percent of reads mapping to the assembly?


Bandage (a Bioinformatics Application for Navigating De novo Assembly Graphs Easily), is a program that creates interactive visualisations of assembly graphs. They can be useful for finding sections of the graph, such as rRNA, or to try to find parts of a genome. Note, you can install Bandage on your local system - see the prerequisites section. With Bandage, you can zoom and pan around the graph and search for sequences, plus much more. The following guide allows you to look at the assembly graph. Normally, I would recommend looking at the ‘ assembly_graph.fastg, but our assembly is quite fragmented, so we will load up the assembly_graph_after_simplification.gfa.


Open Bandage

In the the Bandage GUI perform the following

> Select File->Load graph
>
> Navigate to ~/BiATA/session1/data/session1/assembly and select on assembly_graph_after_simplification.gfa

Once loaded, you need to draw the graph. To do so, under the “Graph drawing” panel on the left side perform the following:

> Set Scope to ‘Entire graph’
>
> The click on Draw graph


Use the sliders in the main panel to move around and look at each distinct part of the assembly graph.

Can you find any large, complex parts of the graph? If so, what do they look like.

In this particular sample, we believe that strains related to the species *Rothia dentocariosa,* a Gram-positive, round- to rod-shaped bacteria that is part of the normal community of microbes residing in the mouth and respiratory tract, should be present in our sample. While this is a tiny dataset, lets try to see if there is evidence for this genome. To do so, we will search the *R. dentocariosa* genome against the assembly graph.

To do so, go to the "BLAST" panel on the left side of the GUI.

Step 1 - Select Create/view BLAST search, this will open a new window

Step 2 - select build Blast database

Step 3 - Load from FASTA file -> navigate to the genome folder /opt/data/genome and select GCA_000164695.fasta

Step 4 - modify the blast filters to 95% identity

Step 6 - run blast

Step 7 - close this window

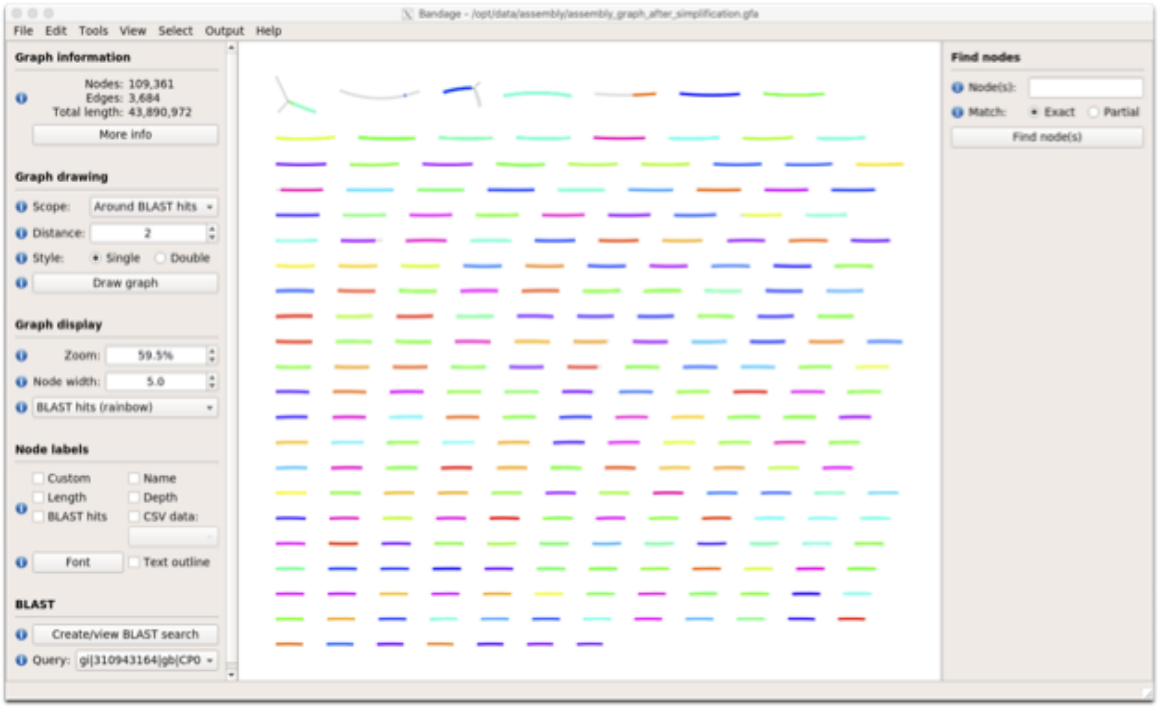
To visualise just these hits, go back to "Graph drawing" panel.

Set Scope to 'Around BLAST hits'

Set Distance 2

The click on Draw graph

You should then see something like this:



In the following steps of this exercise, we will look at performing co-assembly of multiple datasets. Due to computational limitations, we can only look a example datasets. However, the principles are the same. We have also pre-calculated some assemblies for you. In the co-assembly directory, there are already 2 assemblies. We have a single paired-end assembly.

```
megahit -1 clean_other/oral_human_example_1_splitac_kneaddata_paired_1.fastq -2 clean_
↪other/oral_human_example_1_splitac_kneaddata_paired_1.fastq -o coassembly/assembly1 -t␣
↪2 --k-list 23,51,77
```

 Now run the assembly_stats on the contigs for this assembly.

```
cd /opt/data
assembly_stats coassembly/assembly1/final.contigs.fa
```

 How do these differ to the ones you generated previously? What may account for these differences?

 We have also generated the first coassembly using MegaHIT. This was produced using the following command. To specify the files, we put all of the forward file as a comma separated list, and all of the reversed as a comma separated list, which should be ordered that same in both, such that the mate pairs match up.

```
cd /opt/data
megahit -1 clean_other/oral_human_example_1_splitac_kneaddata_paired_1.fastq,clean_other/
↪oral_human_example_1_splitab_kneaddata_paired_1.fastq -2 clean_other/oral_human_
↪example_1_splitac_kneaddata_paired_1.fastq,clean_other/oral_human_example_1_splitab_
↪kneaddata_paired_2.fastq -o coassembly/assembly2 -t 2 --k-list 23,51,77
```

 Now perform another co-assembly:

```
megahit -1 clean_other/oral_human_example_1_splitab_kneaddata_paired_1.fastq,clean_other/
↪oral_human_example_1_splitac_kneaddata_paired_1.fastq,clean/oral_human_example_1_
↪splitaa_kneaddata_paired_1.fastq -2 clean_other/oral_human_example_1_splitab_kneaddata_
↪paired_2.fastq,clean_other/oral_human_example_1_splitac_kneaddata_paired_2.fastq,clean/
↪oral_human_example_1_splitaa_kneaddata_paired_2.fastq -o coassembly/assembly3 -t 2 --k-
↪list 23,51,77
```

 This takes about 20-30 minutes. Also, if you are using a laptop, make sure that it does not go into standby mode.

 You should now have three different assemblies, two provide and one generated by yourselves. Now let us compare the assemblies.

```
cd /opt/data
assembly_stats coassembly/assembly1/final.contigs.fa
assembly_stats coassembly/assembly2/final.contigs.fa
assembly_stats coassembly/assembly3/final.contigs.fa
```

 We only have contigs.fa from MegaHIT, so the contigs and scaffold sections are the same.

 Has the assembly improved? If so how?

CHAPTER

TWO

ASSEMBLY ANALYSIS

In the presentation we will cover:

- Overview of MGnify annotation pipeline
- Taxonomic assignment
- Functional characterisation
- Pathways/systems
- Using the contig viewer

2.1 MGnify hands-on exercises

For this session we will look at some of the data and analyses that are available from MGnify. We will navigate the resource, try out different ways to search for interesting samples/studies, and then investigate the analysis results that are available for assemblies.

2.1.1 Browsing MGnify

From the MGnify front page (https://www.ebi.ac.uk/metagenomics/) you can see various options to browse the data. There are quick links to the various data-types (e.g. amplicon, assembly, metagenomes, etc) we support, as well as a subset of the biomes that the data covers.

 Click on the "wastewater" biome icon.

 How many studies does MGnify hold that relate to wastewater?

 How many samples does that relate to?

 From the sample page, filter the rows with accession ERS1215575, and take a look at the metadata available.

 Do you know the exact location of where the sample was taken?

 What are the lat/long co-ordinates?

 Follow the link to the BioSamples record, can you find any more information about the location of the sample?

 From the tabs in the header bar, select **Text search**, and then select **Samples** below the search box. There are a number of metadata fields available to allow you to filter for a sample of interest to you. Not all are relevant to all samples. Within the hierarchy of biomes, navigate to environmental>aquatic>lentic. You should see 92 samples. Now select the depth filter.

 How many lentic samples have depth data associated with them?

 Using the sliders, can you identify a sample of a lentic water system from a depth between 47-49m?

2.1.2 MGnify assembly analysis

Now we will look at some assembly data that has been analysed by MGnify.

 Search for **MGYS00003598**, and go to this study page. This is a large study where MGnify have assembled the raw reads from an existing public study. The list of assemblies is shown at the bottom of the study page.

 How many analyses are included in this study?

 Click on the 2nd analysis link in the list **MGYA00510849**. You could alternatively search for this accession using the text search options. Have a look at the information within the **Quality control** tab.

 How many contigs are included in this analysis?

 What length is the longest contig in this dataset?

 Click on the **Taxonomic analysis** tab and examine the phylum composition in the graphs and the krona plot.

 What proportion of the total LSU rRNA predictions are eukaryotic?

 Try switching between the other available graph views.

 Which phylum contains the highest proportion of LSU rRNA predictions?

 Click on the **Functional analysis** tab. The top part of this page shows a sequence feature summary, showing the number of contigs with predicted coding sequences (pCDS), the number of pCDS with InterPro matches, etc.

 How many predicted coding sequences (pCDS) are in the assembly?

 How many pCDS have InterProScan hits?

 Scroll down the page to the InterPro match summary section.

 How many different InterPro entries are matched by the pCDS?

 Why is this number different to the number of pCDS that have InterProScan hits?

 Click on the **GO Terms** sub-tab. This shows a summary of the most common GO terms annotated to the pCDS as both bar charts, and pie charts.

 What are the top 3 biological process terms predicted for the pCDS from this assembly?

 Have a look at the information in the **Pfam** and **KO** (KEGG orthologue) sub-tabs.

 Click on the **Pathways/Systems** tab. Have a look at the data reported in the 3 sub-tabs: KEGG Module, Genome Properties, and antiSMASH.

 How many KEGG modules are reported for this assembly?

 How many of these are 100% complete (i.e. all of the constituent KOs are found)?

 How many Genome Properties of the category **DNA handling**, are found within this assembly?

 What is the most common class of biosynthetic gene cluster found in this assembly?

 How many non-ribosomal peptide synthetase gene clusters are identified by antiSMASH in this assembly?

 Click on the **Contig Viewer** tab. Load the data for the 4th contig in the list by clicking on the contig name (ERZ501066.4-NODE-4-length-276957-cov-33.799655). This contig will now be loaded into the viewer.

 How long is this contig?

 The longest pCDS in the contig appears to start at 202339. What protein is coded for?

 Looking at the antiSMASH annotations, where within the contig do any transport-related genes fall?

 Zoom into that region to see the predicted regions in more detail. Have a look at the information about the various transport-related genes.

 What region of the contig is predicted to code for a major facilitator transporter?

 There are lots of different visualisation options available within the contig viewer. Take some time now to investigate the various options, and play about with it by looking at a few different contigs and the anotations they contain.

CHAPTER

THREE

VIRAL DETECTION AND CLASSIFICATION

3.1 Prerequisites

IMPORTANT: It is strongly recommended that you complete section 1.1 below (setting up the computing environment) prior to the practical session. Depending on your available network it can take some time (1-2 hrs) to download the necessary files.

 1.1. To run this tutorial first we need to set up our computing environment in order to execute the commands as listed here. First, download and the **virify_tutorial.tar.gz** file containing all the data you will need using any of the following options:

```
wget http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/virify_
↪tutorial.tar.gz
or
rsync -av --partial --progress rsync://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_
↪courses/biata_2021/virify_tutorial.tar.gz .
```

Once downloaded, extract the files from the tarball:

```
tar -xzvf virify_tutorial.tar.gz
```

Now change into the **virify_tutorial** directory and setup the environment by running the following commands in your current terminal session:

```
cd virify_tutorial
docker load --input docker/virify.tar
docker run --rm -it -v $(pwd)/data:/opt/data virify
mkdir obs_results
```

All commands detailed below will be run from within this current working directory. Note: if there are any issues in running this tutorial, there is a separate directory **exp_results/** with pre-computed results.

3.2 1. Identification of putative viral sequences

In order to retrieve putative viral sequences from a set of metagenomic contigs we are going to use two different tools designed for this purpose, each of which employs a different strategy for viral sequence detection: **VirFinder** and **VirSorter**. VirFinder uses a prediction model based on kmer profiles trained using a reference database of viral and prokaryotic sequences. In contrast, VirSorter mainly relies on the comparison of predicted proteins with a comprehensive database of viral proteins and profile HMMs. The **VIRify pipeline** uses both tools as they provide complementary results:

- **VirFinder** performs better than VirSorter for short contigs (<3kb) and includes a prediction model suitable for detecting both eukaryotic and prokaryotic viruses (phages).
- In addition to reporting the presence of phage contigs, **VirSorter** detects and reports the presence of prophage sequences (phages integrated in contigs containing their prokaryotic hosts).

1.2 In the current working directory you will find the metagenomic assembly we will be working with (**ERR575691_host_filtered.fasta**). We will now filter the contigs listed in this file to keep only those that are 500 bp, by using the custom python script filter_contigs_len.py as follows:

```
filter_contigs_len.py -f ERR575691_host_filtered.fasta -l 0.5 -o obs_results/ERR575691_
↪host_filtered_filt500bp.fasta
```

1.3. The output from this command is a file named **ERR575691_host_filtered_filt500bp.fasta** which is located in the **obs_results** diretory. Our dataset is now ready to be processed for the detection of putative viral sequences. We will first analyse it with VirFinder using a custom R script:

```
VirFinder_analysis_Euk.R -f obs_results/ERR575691_host_filtered_filt500bp.fasta -o obs_
↪results
```

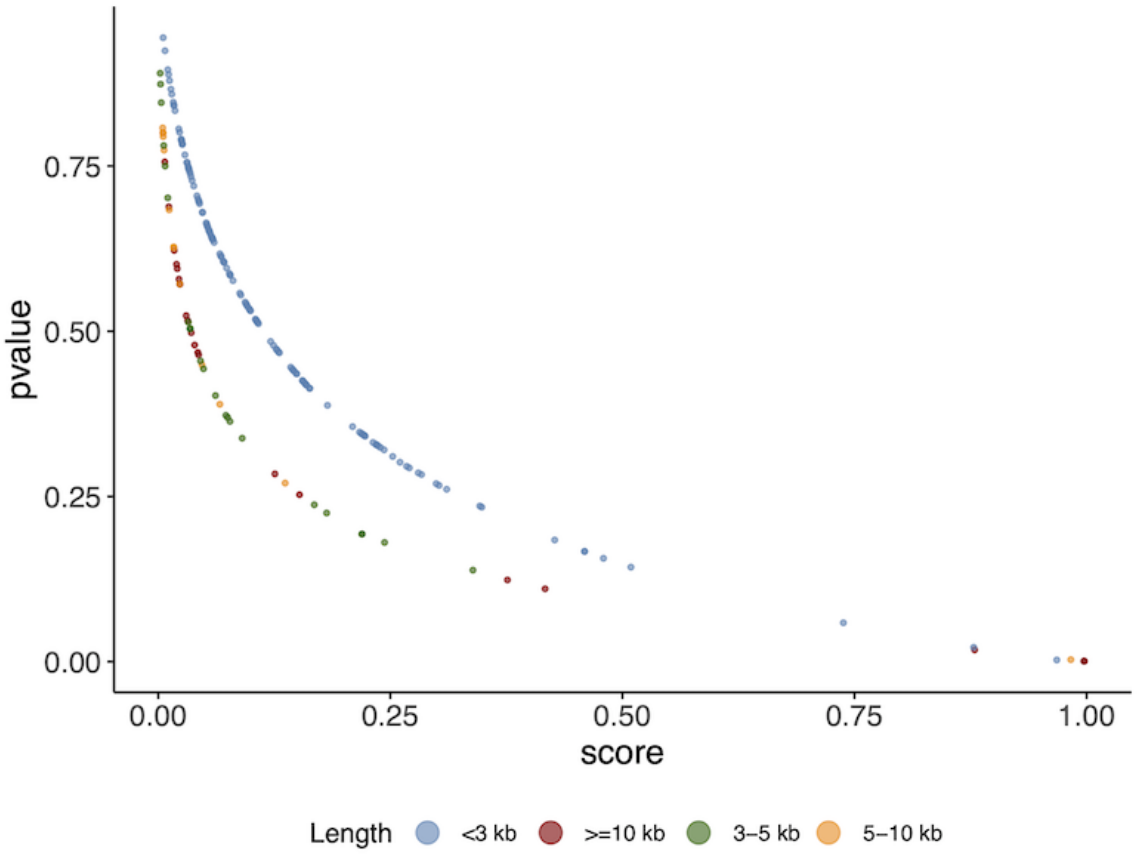
1.4. Following the execution of the R script you will see a tabular file (**obs_results/ERR575691_host_filtered_filt500bp_VirFinder_table-all.tab**) that collates the results obtained for each contig from the processed FASTA file. The next step will be to analyse the metagenomic assembly using VirSorter. To do this run:

```
wrapper_phage_contigs_sorter_iPlant.pl -f obs_results/ERR575691_host_filtered_filt500bp.
↪fasta --db 2 --wdir obs_results/virsorter_output --virome --data-dir /opt/data/
↪databases/virsorter-data
```

VirSorter classifies its predictions into different confidence categories:

- **Category 1**: "most confident" predictions
- **Category 2**: "likely" predictions
- **Category 3**: "possible" predictions
- **Categories 4-6**: predicted prophages

1.5. While VirSorter is running, we have prepared an R script so you can inspect the VirFinder results in the meantime using ggplot2. Open RStudio and load the **Analyse_VirFinder.R** script located in the **/virify_tutorial/data/scripts/** directory. Run the script (press Source on the top right corner) to generate the plot. (If you don't have RStudio, or don't care to run this you can just look at the resulting plot in the image below)



As you can see there is a relationship between the **p-value** and the **score**. A higher score or lower p-value indicates a higher likelihood of the sequence being a viral sequence. You will also notice that the results correlate with the **contig length**. The curves are slightly different depending on whether the contigs are > or < than 3kb. This is because VirFinder uses different machine learning models at these different levels of length.

1.6. Once VirSorter finishes running, we then generate the corresponding viral sequence FASTA files using a custom python script (**parse_viral_pred.py**) as follows:

```
parse_viral_pred.py -a obs_results/ERR575691_host_filtered_filt500bp.fasta -f obs_
↪results/ERR575691_host_filtered_filt500bp_VirFinder_table-all.tab -s obs_results/
↪virsorter_output -o obs_results
```

Following the execution of this command, FASTA files (*.fna) will be generated for each one of the VIRify categories mentioned above containing the corresponding putative viral sequences.

The VIRify pipeline takes the output from VirFinder and VirSorter, reporting three prediction categories:

- **High confidence**: VirSorter phage predictions from **categories 1 and 2**.
- **Low confidence**:
- Contigs that VirFinder reported with **p-value < 0.05 and score 0.9.**
- Contigs that VirFinder reported with **p-value < 0.05 and score 0.7**, but that are also reported by VirSorter in **category 3.**
- **Prophages**: VirSorter prophage predictions **categories 4 and 5.**

3.3 2. Detection of viral taxonomic markers

Once we have retrieved the putative viral sequences from the metagenomic assembly, the following step will be to analyse the proteins encoded in them in order to identify any viral taxonomic markers. To carry out this identification, we will employ a database of **profile Hidden Markov Models (HMMs)** built from proteins encoded in viral reference genomes. These profile HMMs were selected as viral taxonomic markers following a comprehensive random forest-based analysis carried out previously.

2.1. The VIRify pipeline uses **prodigal** for the detection of **protein coding sequences (CDSs)** and **hmmscan** for the alignment of the encoded proteins to each of the profile HMMs stored in the aforementioned database. We will use the custom script **Generate_vphmm_hmmer_matrix.py** to conduct these steps for each one of the FASTA files sequentially in a "for loop". In your terminal session, execute the following command:

```
for file in $(find obs_results/ -name '*.fna' -type f | grep -i 'putative'); do Generate_
↪vphmm_hmmer_matrix.py -f ${file} -o ${file%/*}; done
```

Once the command execution finishes two new files will be stored for each category of viral predictions. The file with the suffix **CDS.faa** lists the proteins encoded in the CDSs reported by prodigal, whereas the file with the suffix **hmmer_ViPhOG.tbl** contains all significant alignments between the encoded proteins and the profile HMMs, on a per-domain-hit basis.

2.2. The following command is used to parse the hmmer output and generate a new tabular file that lists alignment results in a per-query basis, which include the **alignment ratio** and absolute value of total **E-value** for each protein-profile HMM pair.

```
for file in $(find obs_results/ -name '*ViPhOG.tbl' -type f); do Ratio_Evalue_table.py -
↪i ${file} -o ${file%/*}; done
```

3.4 3. Viral taxonomic assignment

The final output of the VIRify pipeline includes a series of gene maps generated for each putative viral sequence and a tabular file that reports the taxonomic lineage assigned to each viral contig. The gene maps provide a convenient way of visualizing the taxonomic annotations obtained for each putative viral contig and compare the annotation results with the corresponding assigned taxonomic lineage. Taxonomic lineage assignment is carried out from the highest taxonomic rank (genus) to the lowest (order), taking all the corresponding annotations and assessing whether the most commonly reported one passes a pre-defined assignment threshold.

3.1. First, we are going to generate a tabular file that lists the taxonomic annotation results obtained for each protein from the putative viral contigs. We will generate this file for the putative viral sequences in each prediction category. Run the following:

```
for file in $(find obs_results/ -name '*CDS.faa' -type f); do viral_contigs_annotation.
↪py -p ${file} -t ${file%CDS.faa}hmmer_ViPhOG_informative.tsv -o ${file%/*}; done
```

3.2. Next, we will take the tabular annotation files generated and use them to create the viral contig gene maps. To achieve this, run the following:

```
for file in $(find obs_results/ -name '*annot.tsv' -type f); do Make_viral_contig_map.R -
↪t ${file} -o ${file%/*}; done
```

3.3. Finally, we will use the tabular annotation files again to carry out the taxonomic lineage assignment for each putative viral contig. Run the following command:

```
for file in $(find obs_results/ -name '*annot.tsv' -type f); do contig_taxonomic_assign.
↪py -i ${file} -o ${file%/*}; done
```

Final output results are stored in the **obs_results/** directory.

The gene maps are stored per contig in individual **PDF files** (suffix names of the contigs indicate their level of confidence and category class obtained from VirSorter). Each protein coding sequence in the contig maps (PDFs) is coloured and labeled as **high confidence** (E-value < 0.1), **low confidence** (E-value > 0.1) or **no hit**, based on the matches to the HMM profiles. Do not confuse this with the high confidence or low confidence prediction of VIRify for the **whole contig**.

Taxonomic annotation results per classification category are stored as text in the ***_tax_assign.tsv** files.

Let's inspect the results. Do:

```
cat obs_results/*tax_assign.tsv
```

You should see a list of **9 contigs** detected as viral and their taxonomic annotation in separate columns (partitioned by taxonomic rank). However, some do not have an annotation (e.g. **NODE_4…** and **NODE_5…**).

Open the gene map PDF files of the corresponding contigs to understand why some contigs were **not assigned** to a taxonomic lineage. You will see that for these cases, either there were not enough genes matching the HMMs, or there was disagreement in their assignment.

CHAPTER

FOUR

MAG GENERATION

- Generation of metagenome assembled genomes (MAGs) from assemblies
- Assessment of quality (MIMAGs)
- Taxonomic assignment

4.1 Prerequisites

For this tutorial you will need to make a working directory to store your data in.

```
mkdir -p ~/BiATA/session4/data
chmod -R 777 ~/BiATA
```

(If your path to the data folder looks different from the one in this example, please make sure there are no spaces in any of your directory names.)

In this directory, download the tarball from http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/

```
cd  ~/BiATA/session4/data
wget -q http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/
↪session4.tgz
tar xzvf session4.tgz
```

Now make sure that you have pulled the docker container:

```
docker pull microbiomeinformatics/biata-binning
```

Finally, start the docker container in the following way:

```
export DATADIR=~/BiATA/session4/data/session4
docker run --rm -it  -e DISPLAY=$DISPLAY  -v $DATADIR:/opt/data -v /tmp/.X11-unix:/tmp/.
↪X11-unix:rw  -e DISPLAY=docker.for.mac.localhost:0 microbiomeinformatics/biata-binning
```

4.2 Generating metagenome assembled genomes

Learning Objectives - in the following exercises you will learn how to bin an assembly, assess the quality of this assembly with CheckM and then visualise a placement of these genomes within a reference tree.

As with the assembly process, there are many software tools available for binning metagenome assemblies. Examples include, but are not limited to:

MaxBin: https://sourceforge.net/projects/maxbin/

CONCOCT: https://github.com/BinPro/CONCOCT

COCACOLA: https://github.com/younglululu/COCACOLA

MetaBAT: https://bitbucket.org/berkeleylab/metabat

There is no clear winner between these tools, so the best is to experiment and compare a few different ones to determine which works best for your dataset. For this exercise we will be using **MetaBAT** (specifically, MetaBAT2). The way in which MetaBAT bins contigs together is summarised in Figure 1.

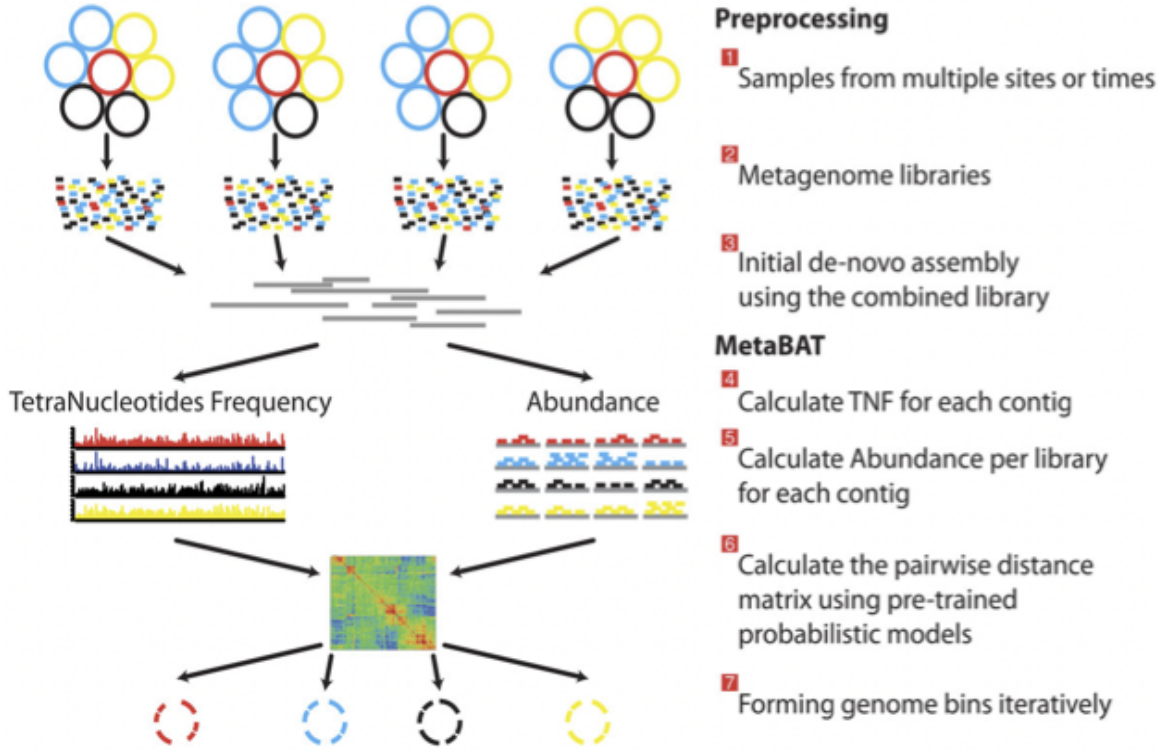


Figure 1. MetaBAT workflow (Kang, et al. *PeerJ* 2015).

Prior to running MetaBAT, we need to generate coverage statistics by mapping reads to the contigs. To do this, we can use bwa (http://bio-bwa.sourceforge.net/) and then the samtools software (http://www.htslib.org) to reformat the output. Again, this can take some time, so we have run it in advance. To repeat the process, you would run the following commands:

```
cd /opt/data/assemblies

# index the contigs file that was produced by metaSPAdes:
```

(continues on next page)

(continued from previous page)

```
bwa index contigs.fasta

# map the original reads to the contigs:
bwa mem contigs.fasta ERR011322_1.fastq ERR011322_2.fastq > input.fastq.sam

# reformat the file with samtools:
samtools view -Sbu input.fastq.sam > junk
samtools sort junk input.fastq.sam
```

We should now have the files we need for the rest of the process – the assemblies themselves (*contigs.fasta*) and a file from which we can generate the coverage stats (*input.fastq.sam.bam).*

Running MetaBAT

The `/opt/data/assemblies` directory inside the docker container should contain the following two files needed to run MetaBAT: *contigs.fasta* and *input.fastq.sam.bam*.

Create a subdirectory where files will be output:

```
cd /opt/data/assemblies/
mkdir contigs.fasta.metabat-bins2000
```

Run the following command to produce a *contigs.fasta.depth.txt* file, summarising the output depth for use with MetaBAT:

```
cd /opt/data/assemblies/
jgi_summarize_bam_contig_depths --outputDepth contigs.fasta.depth.txt input.fastq.sam.bam

# now run MetaBAT

cd /opt/data/assemblies/
metabat2 --inFile contigs.fasta --outFile contigs.fasta.metabat-bins2000/bin --abdFile␣
↪contigs.fasta.depth.txt --minContig 2000
```

Once the binning process is complete, each bin will be grouped into a multi-fasta file with a name structure of **bin.[0-9].fa**.

Move to the output directory and look at the output of the binning process.

```
cd /opt/data/assemblies/contigs.fasta.metabat-bins2000/
```

How many bins did the process produce?

How many sequences are in each bin?

What does each bin represent?

Obviously, not all bins will have the same level of accuracy since some might represent a very small fraction of a potential species present in your dataset. To further assess the quality of the bins we will use **CheckM** (https://github.com/Ecogenomics/CheckM/wiki).

Running CheckM

CheckM has its own reference database of single-copy marker genes. Essentially, based on the proportion of these markers detected in the bin, the number of copies of each and how different they are, it will determine the level of **completeness**, **contamination** and **strain heterogeneity** of the predicted genome.

Before we start, we need to configure checkM.

```
cd /opt/data
mkdir checkm_data
mv checkm_data_2015_01_16.tar.gz checkm_data
cd checkm_data
tar zxvf checkm_data_2015_01_16.tar.gz
checkm data setRoot /opt/data/checkm_data
```

This program has some handy tools not only for quality control, but also for taxonomic classification, assessing coverage, building a phylogenetic tree, etc. The most relevant ones for this exercise are wrapped into the **lineage_wf** workflow.

Move back to the top level directory

```
cd /opt/data/assemblies/
```

Now run CheckM with the following command:

```
checkm lineage_wf -x fa contigs.fasta.metabat-bins2000 checkm_output --tab_table -f MAGs_
↪checkm.tab --reduced_tree -t 4
```

Due to memory constraints (< 40 GB), we have added the option **–reduced_tree** to build the phylogeny with a reduced number of reference genomes. In case the process still runs out of memory and is killed, the result files have been provided in `/opt/data/checkm_answers`.

Once the **lineage_wf** analysis is done, the reference tree can be found in **checkm_output/storage/tree/concatenated.tre**. Additionally, you will have the taxonomic assignment and quality assessment of each bin in the file **MAGs_checkm.tab** with the corresponding level of **completeness**, **contamination** and **strain heterogeneity** (Fig. 2). A quick way to infer the overall quality of the bin is to calculate the level of **(completeness - 5*contamination)**. You should be aiming for an overall score of at least **70-80%**.

Bin Id	Marker lineage	# genomes	# markers	# marker sets	0	1	2	3	4	5+	Completeness	Contamination	Strain heterogeneity
bin.1	f__Lachnospiraceae (UID1286)	57	420	207	27	383	10	0	0	0	95.94	2.42	50
bin.2	k__Bacteria (UID203)	5449	104	58	5	22	28	44	5	0	97.39	163.71	42.63
bin.3	o__Clostridiales (UID1212)	172	263	149	9	253	1	0	0	0	96.64	0.67	100
bin.4	o__Clostridiales (UID1212)	172	263	149	65	197	1	0	0	0	75.03	0.67	0
bin.5	k__Bacteria (UID2565)	2921	152	93	13	139	0	0	0	0	88.17	0	0

Figure 2. Example output of CheckM

Before we can visualize and plot the tree we will need to convert the reference ID names used by CheckM to taxon names. We have already prepared a mapping file for renaming the tree (**rename_list.tab**). We can then do this easily with the **newick utilities** (http://cegg.unige.ch/newick_utils).

To do this, run the following command:

```
nw_rename checkm_output/storage/tree/concatenated.tre rename_list.tab > renamed.tree
```

Visualising the phylogenetic tree

We will now plot and visualize the tree we have produced. A quick and user- friendly way to do this is to use the web-based **interactive Tree of Life** (**iTOL**): http://itol.embl.de/index.shtml

iTOL only takes in newick formatted trees, so we need to quickly reformat the tree with **FigTree** (http://tree.bio.ed.ac.uk/software/figtree/).

You should be able to run FigTree as follows:

```
figtree
```

 Open the **renamed.tree** file with **FigTree** and then select from the toolbar **File -> Export Trees**. In the **Tree file format** select **Newick** and export the file as **renamed.nwk** (choose a name you will recognise if you plan to use the shared account described below).

 To use **iTOL** you will need a user account. For the purpose of this tutorial we have already created one for you with an example tree. The login is as follows:

User: *EBI_training*

Password: *EBI_training*

After you login, just click on **My Trees** in the toolbar at the top and select

IBD_checkm_tree.nwk from the **Imported trees** workspace.

Alternatively, if you want to create your own account and plot the tree yourself

follow these steps:

1) After you have created and logged in to your account go to **My Trees**

2) From there select **Upload tree files** and upload the tree you exported from **FigTree**

3) Once uploaded, click the tree name to visualize the plot

4) To colour the clades and the outside circle according to the phylum of each strain, drag and drop the files **iTOL_clades.txt** and **iTOL_ocircles.txt** into the browser window

Once that is done, all the reference genomes used by **CheckM** will be coloured according to their phylum name, while all the other ones left blank correspond to the **target genomes** we placed in the tree. Highlighting each tip of the phylogeny will let you see the whole taxon/sample name. Feel free to play around with the plot.

 Does the CheckM taxonomic classification make sense? What about the unknowns? What is their most likely taxon?