

---

# **Metagenomics Workshop Documentation**

***Release 1***

**MGnify team**

**Oct 13, 2022**



# CONTENTS

<b>1</b>	<b>Overview</b>	<b>3</b>
1.1	Quality control and filtering of the raw sequence files . . . . .	3
1.2	MGnify Services . . . . .	13
1.3	Assembling data . . . . .	17
1.4	MAG generation . . . . .	21
1.5	MGnify Genomes Catalogues . . . . .	25
1.6	License . . . . .	26
1.7	Viral detection and classification . . . . .	26



These are the materials that will be used for the “Metagenomics bioinformatics (Virtual)” workshop.



## OVERVIEW

Gain knowledge of the tools, processes and analysis approaches used in the field of metagenomics.

This course will cover the metagenomics data analysis workflow from the point of newly generated sequence data. Participants will explore the use of publicly available resources and tools to manage, share, analyse and interpret metagenomics data. The content will include issues of data quality control and how to submit to public repositories. While sessions will detail marker-gene and whole-genome shotgun (WGS) approaches; the primary focus will be on assembly-based approaches. Discussions will also explore considerations when assembling genome data, the analysis that can be carried out by MGnify on such datasets, and what downstream analysis options and tools are available

Contents:

## 1.1 Quality control and filtering of the raw sequence files

### 1.1.1 Prerequisites

There are no prerequisites to run this practical for the course. All the data and tools have been downloaded and installed on the VMs.

**Skip to the next section “Quality control and filtering of the raw sequence files”**

To run this practical locally after the course: see section “Running this practical locally” at the end of this document.

### 1.1.2 Quality control and filtering of the raw sequence files

For this tutorial you will need to move into the working directory. It is important to set the variable DATADIR as stated.

```
cd /home/training/Data/Quality/files
chmod -R 777 /home/training/Data/Quality
export DATADIR=/home/training/Data/Quality/files
xhost +
```

You will see the message “access control disabled, clients can connect from any host”



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



Here you should see the contents of the working directory. These are the files we will use for the practical.

```
ls $DATADIR
```



Generate a directory of the fastqc results

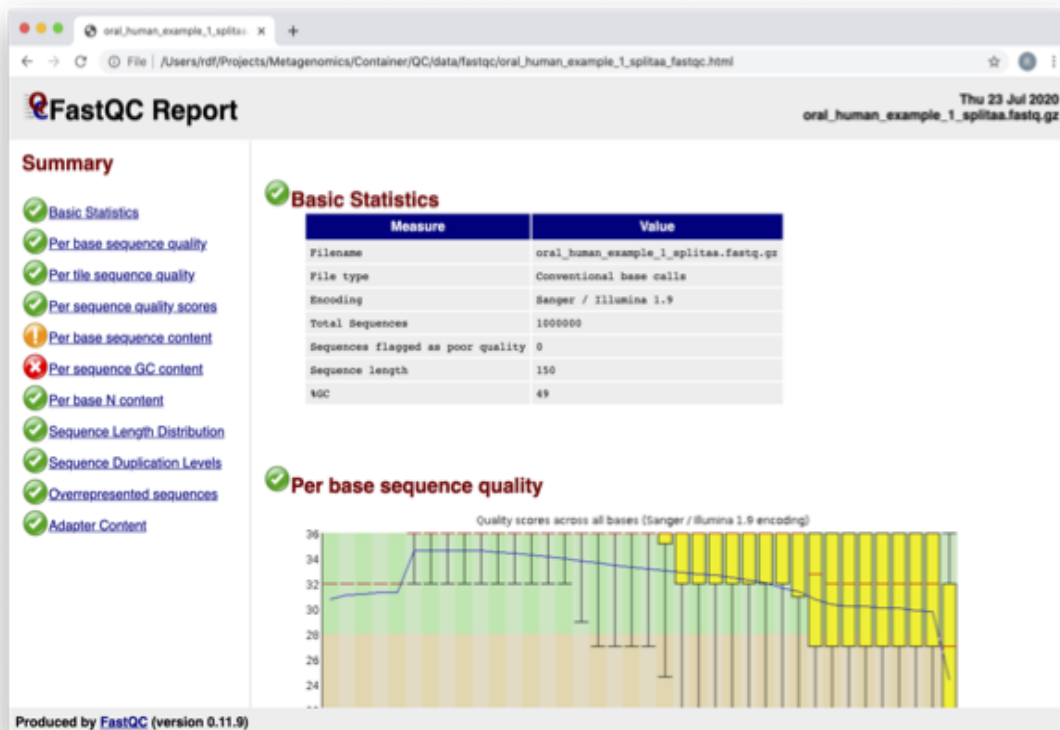
```
mkdir fastqc_results
fastqc oral_human_example_1_splitaa.fastq.gz
fastqc oral_human_example_2_splitaa.fastq.gz
mv $DATADIR/*.zip $DATADIR/fastqc_results
mv $DATADIR/*.html $DATADIR/fastqc_results
```



Now on your computer on the left hand bar, select the folder icon called 'Files'.

Navigate to Home -> Data -> Quality -> files -> fastqc\_results

Click on file 'oral\_human\_example\_1\_splitaa\_fastqc.html' which should open in the browser



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? When 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, however, 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 files?

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 $DATADIR
mkdir multiqc_results
```

(continues on next page)

(continued from previous page)

```
multiqc fastqc_results -o multiqc_results
```

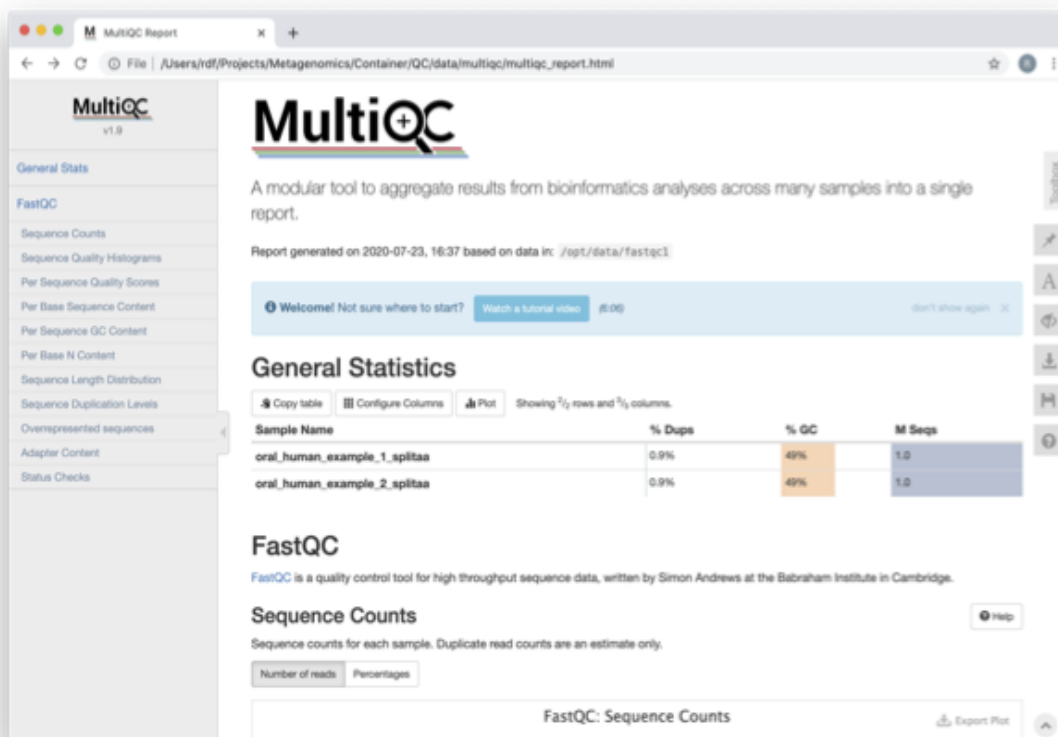
In this case, we provide the folder containing the fastqc results to multiqc and the -o allows us to set the output directory for this summarised report.



Now on your computer on the left hand bar, select the folder icon called 'Files'.

Navigate to Home → Data → Quality → files → multiqc\_results

Click on file 'multiqc\_report.html' which should open in the browser



Scroll down through the report. The sequence quality histograms show the above 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 duplicates, sequences with low quality scores or removal of the adaptors. 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.



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 $DATADIR/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 $DATADIR
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 $DATADIR/clean --reference-db
↳ $DATADIR/decontamination/GRCh38_phix.index --bowtie2-options "--very-sensitive --
↳ dovetail" --trimmomatic /home/training/Data/Quality/Trimmomatic-0.39/ --bypass-trf --
↳ trimmomatic-options "SLIDINGWINDOW:4:20 MINLEN:50"
```



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 $DATADIR/clean
mkdir fastqc_final
<you construct the commands>
mv $DATADIR/clean/*.zip $DATADIR/clean/fastqc_final
mv $DATADIR/clean/*.html $DATADIR/clean/fastqc_final
```

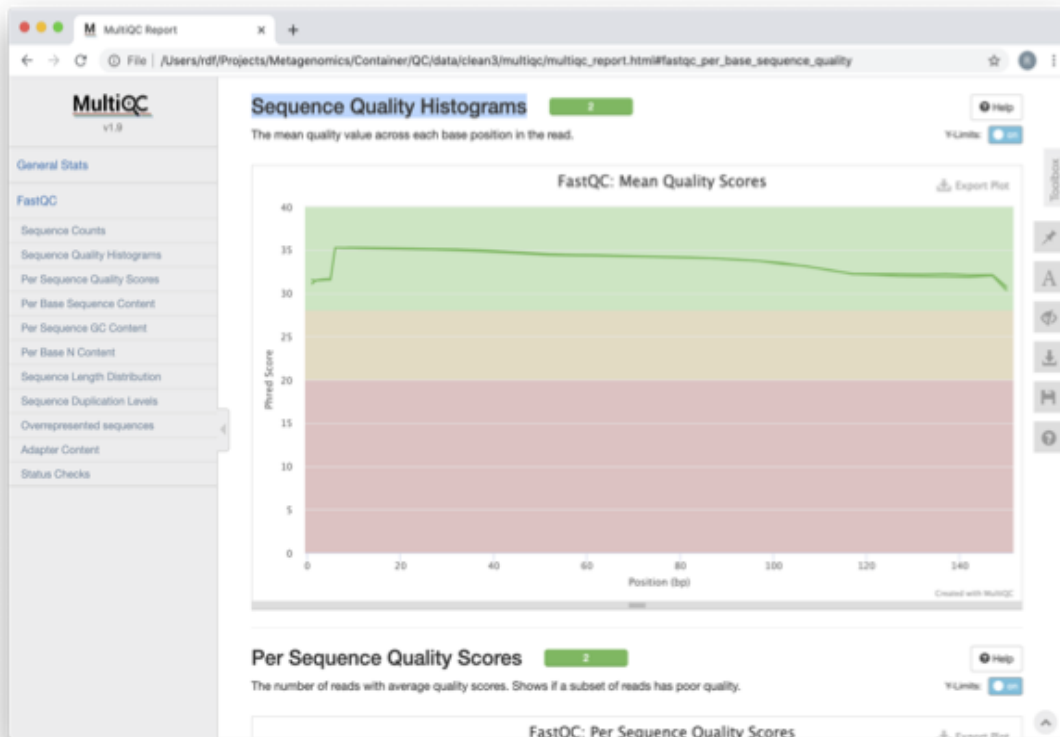


Also generate a multiqc report and look at the sequence quality histograms.

```
cd $DATADIR/clean/
mkdir multiqc_final
<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 if they have improved?



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



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

```
cd $DATADIR
kneaddata_read_count_table --input $DATADIR/clean --output kneaddata_read_counts.txt
less kneaddata_read_counts.txt
```



What fraction of reads have been 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>

### 1.1.3 Assembly PhiX decontamination



Learning Objectives - in the following exercises you will generate a PhiX blast database, and run a blast search with a subset of assembled freshwater sediment metagenomic reads, to identify contamination.

PhiX, used in the previous section of this practical, is a small bacteriophage genome typically used as a calibration control in sequencing runs. Most library preparations will use PhiX at low concentrations, however it can still appear in the sequencing run. If not filtered out, PhiX can form small spurious contigs which could be incorrectly classified as diversity.



Generate the PhiX reference blast database

```
cd $DATADIR/decontamination
makeblastdb -in phix.fasta -input_type fasta -dbtype nucl -parse_seqids -out phix_blastDB
```



Prepare the freshwater sediment example assembly file and search against the new blast database. This assembly file contains only a subset of the contigs for the purpose of this practical.

```
cd $DATADIR
gunzip -c freshwater_sediment_contigs.fa.gz > freshwater_sediment_contigs.fa
blastn -query freshwater_sediment_contigs.fa -db decontamination/phix_blastDB -task_
↪ megablast -word_size 28 -best_hit_overhang 0.1 -best_hit_score_edge 0.1 -dust yes -
↪ evaluate 0.0001 -min_raw_gapped_score 100 -penalty -5 -soft_masking true -window_size_
↪ 100 -outfmt 6 -out freshwater_blast_out.txt
```



The blast options are:

- \* **-query**, Input assembly fasta file.
- \* **-out**, Output file
- \* **-db**, Path to blast database.
- \* **-task**, Search type -“megablast”, for very similar sequences (e.g, sequencing errors)
- \* **-word\_size**, Length of initial exact match



Add headers to the blast output and look at the contents of the final output file

```
cat blast_outfmt6.txt freshwater_blast_out.txt > freshwater_blast_out_headers.txt
less freshwater_blast_out_headers.txt
```



Are the hits significant?



What are the lengths of the matching contigs? We would typically filter metagenomic contigs at a length of 500bp. Would any PhiX contamination remain after this filter?



Now that PhiX contamination was identified, it is important to remove these contigs from the assembly file before further analysis or upload to public archives.

### 1.1.4 Using Negative Controls



**Learning Objectives** - This exercise will look at the analysis of negative controls. You will assess the microbial diversity between a negative control and skin sample.

The images below show the taxonomic classification of two samples: a reagent negative control and a skin metagenomic sample. The skin sample is taken from the antecubital fossa - the elbow crease, which is moist and site of high microbial diversity. The classification was performed with kraken2. Kraken2 takes a while to run, so we have done this for you and plotted the results. An example of the command used to do this:

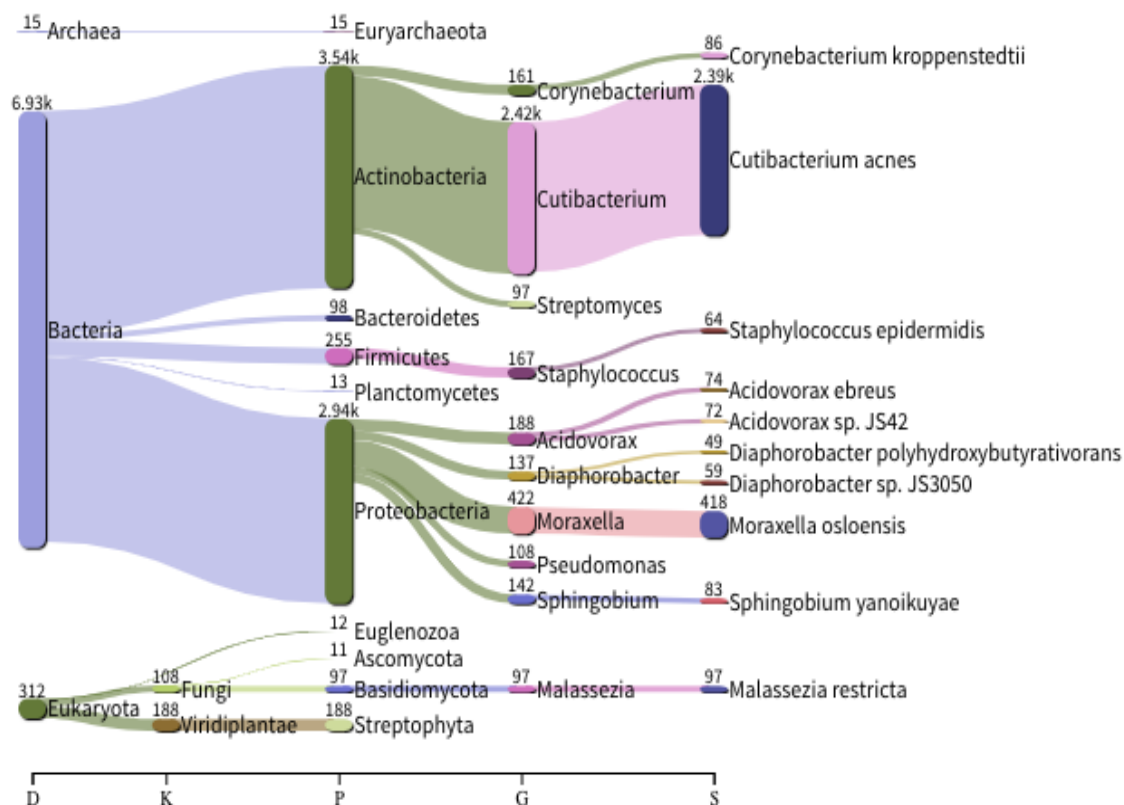
```
kraken2 -db standard_db -threshold 0.10 -threads 8 -use-names -fastq-input -report out.report -gzip-
compressed in_1.fastq.gz in_2.fastq.gz
```

See the kraken2 manual for more information: <https://github.com/DerrickWood/kraken2/wiki/Manual>

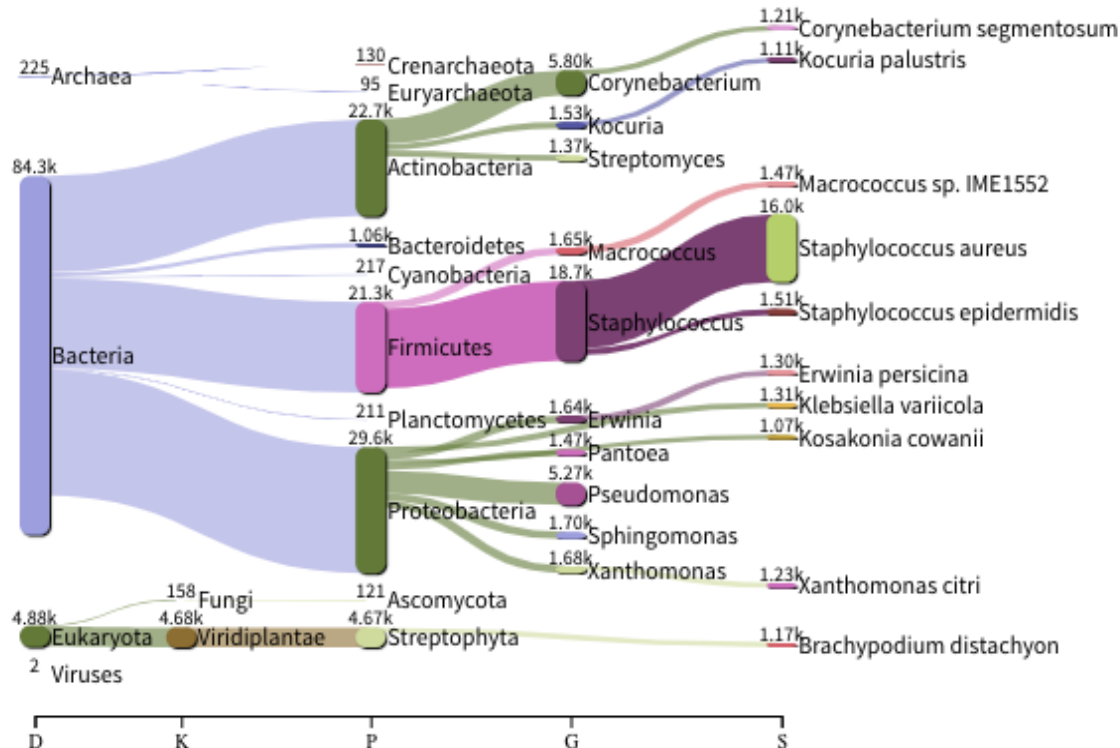
See Pavian manual for the plots: <https://ccb.jhu.edu/software/pavian/>



The following image shows the microbial abundance in the negative control



The following image shows the microbial abundance in the skin sample



Look for similarities and differences at both the phylum and genus level - labelled as 'P' and 'G' on the bottom axis.

Is there any overlap between the negative control and skin sample phylum? Can we map the negative control directly to the skin sample to remove all contaminants? If not, why?

Are there any genera in the negative control which aren't present in the skin sample? If you do a google search of this genus, where are they commonly found? With this information, where could this bacteria in the negative control have originated from?

For more practice assessing and trimming datasets, there is another set of raw reads called "skin\_example\_aa" from the skin metagenome available. These will require a fastqc or multiqc report, followed by trimming and mapping to the reference database with kneaddata. Using what you have learned previously, construct the relevant commands. Remember to check the quality before and after trimming.

Hint: Consider other trimmomatic options from the manual [http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual\\_V0.32.pdf](http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf) e.g. "ILLUMINACLIP", where \$DATADIR/NexteraPE-PE is a file of adapters.

Navigate to skin folder and run quality control

```
cd $DATADIR/skin
<construct the required commands>
```



### 1.1.5 Running this practical locally

The files can be downloaded using the tarball from [http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify\\_courses/ebi\\_2020/quality.tar.gz](http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/ebi_2020/quality.tar.gz)

The docker container has all the tools required and can be started in the following way:

```
export DATADIR={path_to_downloaded_files}
docker run --rm -it -e DISPLAY=$DISPLAY -v $DATADIR:/opt/data -v /tmp/.X11-unix:/tmp/.X11-unix:rw -e DISPLAY=unix$DISPLAY microbiomeinformatics/mgnify-ebi-2020-qc-assembly
```

**Ensure variable DATADIR is set. This is used to mount downloaded data into ``/opt/data`` in the docker container.** You can now run through the practical, replacing \$DATADIR with /opt/data in all commands.

---

**Note:** It's possible that the docker image is not available in dockerhub. In that case you can build the container using the [Dockerfile](#)

To build the container, download the Dockerfile and run “docker build -t microbiomeinformatics/mgnify-ebi-2020-qc-assembly .” in the folder that contains the Dockerfile.

---

## 1.2 MGnify Services

In the presentation we will cover:

- Overview of MGnify annotation pipeline
- Taxonomic assignment
- Functional characterisation
- Pathways/systems
- Other MGnify services
- API Overview

### 1.2.1 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.

#### 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 list, filter for sample “ERS1215575”, select it 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 25-50m?


### 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 assemblies are included in this study?


 Now, search for the analysis **MGYA00510849**. You can use the text search to find this accession. 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?

 What proportion of the bacterial predictions are proteobacteria?

 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 figure 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 annotations they contain.

### MGnify sequence search

Now we will have a look at the database of proteins identified by MGnify.



Click on “Sequence search” from the tabs at the top of the page.

This will open a HMMER search page specific for MGnify. (For more information about the HMMER suite of tools see the HMMER website <https://www.ebi.ac.uk/Tools/hmmer/>)



Copy and paste the protein sequence below into the sequence search box at the top of the page, and click “submit”.

```
GEFWHWTNLLHFILVGLAGGMAFLTALLHLKGHPARRYTLWALGLIALDLFVLWAESPARFRFTHV
WLFLSFHPTSPIWWSWGLALSVSAGLLYLKGKPSKPLAWGLLAFSLVALAYPGMALAVNLNRPLWN  AL-
LAGLFPLTALVLGLGVAVLMKSSWALYPLRILLGASLFLAFLYPFTLTLEARGHLWEEGGVLYGL  FLALGL-
GAFGKESLAPWAAFLAAAGLRALLVAVGQWQG
```



How many query results are significant? (i.e. above the red cut-off line)



Click on the “Customise” button at the top right of the results table, and select to make “Run and sample IDs” column visible and click “Update”. Have a look at the sample data for some of the runs listed in the results (for example the top match result).



We know that the protein belongs to IPR032796 - Polysulfide reductase, we looked for this before hand using InterProScan (<https://www.ebi.ac.uk/interpro/search/sequence/>). Looking at the samples included in the significant results, does it make sense that the example sequence was from this protein family?.

### API

An **API** (“Application programming interface”) is how your scripts (e.g. Python or R) can talk to the MGnify database. For this section of the practical session we will use a Jupyter Notebook.

### Opening the Jupyter notebook



Open a new Terminal. There should be a Terminal application icon in the lefthand menubar.



Type these commands to enter the repository:

```
cd ~/mgnify-ebi-2021
conda activate mgnify-ebi-2021
```



That should have worked, but if you got an error you’ll need to create the environment:

```
conda create --n mgnify-ebi-2021 python=3.9
conda activate mgnify-ebi-2021
pip install -r requirements.txt
```



To open the notebooks, run:

```
jupyter-lab course.jupyterlab-workspace
```

Today's Notebook is "Day 2".

## 1.3 Assembling data

- What constitutes a good assembly?
- How to estimate assembly quality
- Co-assembly

### 1.3.1 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, 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 contiguous 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.



Let's first change to the working directory where we will be running the analyses:

```
cd /home/training/Data/Assembly/files
```



To run metaspades you would execute the following commands (but don't run it now!):

```
mkdir assembly
metaspades.py -t 4 --only-assembler -m 10 -1 reads/oral_human_example_1_splitaa_
→kneaddata_paired_1.fastq -2 reads/oral_human_example_1_splitaa_kneaddata_paired_2.
→fastq -o assembly
```



Since the assembly process would take ~1h we are just going to analyse the output present in assembly.bak. Let's look at the contigs.fasta file.

For this, 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 the assembly perform the following:

```
head -41 assembly.bak/contigs.fasta
```

Descriptions

Graphic Summary

Alignments

Taxonomy

Sequences producing significant alignments

Download

Manage Columns

Show 100

select all

19 sequences selected

	Description	GenBank	Graphics	Distance tree of results			
		Max Score	Total Score	Query Cover	E value	Per. Ident	Accession
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium matruchotii strain NCTC10206 genome assembly, chromosome_1</a>	4183	4240	100%	0.0	98.13%	LR134504.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium matruchotii strain ATCC 14206 chromosome</a>	4170	4227	100%	0.0	98.04%	CP050134.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium atypicum strain R2070, complete genome</a>	333	333	32%	3e-86	74.84%	CP008944.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium sp. NML98-0116 genome</a>	298	298	25%	1e-75	75.69%	CP017939.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium testudinoris strain DSM 44614, complete genome</a>	296	296	30%	4e-75	74.22%	CP011545.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium efficiens Y9-314 DNA, complete genome</a>	283	283	36%	3e-71	72.86%	BA000035.2
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium endometrit strain LMM-1653 chromosome, complete genome</a>	255	255	32%	6e-63	72.77%	CP039247.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium callunae DSM 20147, complete genome</a>	156	156	36%	7e-33	70.37%	CP004354.1
<input checked="" type="checkbox"/>	<a href="#">Corynebacterium variabile DSM 44702, complete genome</a>	117	117	19%	3e-21	72.01%	CP002917.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus sp. DMU1 plasmid unnamed</a>	89.8	89.8	4%	7e-13	81.03%	CP050953.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus rhodochrous strain ATCC BAA870 chromosome, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP032675.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus pyridinivorans strain YF3 chromosome, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP040719.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus bisphenylivorans strain TG9 chromosome, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP022208.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus pyridinivorans strain GF3, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP022915.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus sp. ZQ, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP018063.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus sp. c52, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP016819.1
<input checked="" type="checkbox"/>	<a href="#">Rhodococcus pyridinivorans SB3094, complete genome</a>	62.1	62.1	4%	2e-04	76.72%	CP006996.1
<input checked="" type="checkbox"/>	<a href="#">Salacthaeum sp. JOR-1 chromosome, complete genome</a>	54.7	54.7	1%	0.025	100.00%	CP042241.1
<input checked="" type="checkbox"/>	<a href="#">Rubrobacter xylanophilus DSM 9941, complete genome</a>	54.7	54.7	1%	0.025	100.00%	CP000386.1

**Q** 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?

**A** Now let us consider some statistics about the entire assembly

```
assembly_stats assembly.bak/scaffolds.fasta
```

**i** 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.

**Q** What is the length of longest and shortest contigs?

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

**i** 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.

**A** 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. 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.



At the terminal, type

Bandage

In the the Bandage GUI perform the following

Select File -> Load graph

Navigate to Home -> training -> Data -> Assembly -> files -> assembly.bak and open the file 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: Home -> training -> Data -> Assembly -> files -> genome and select GCA\_000164695.fasta

Step 4 - Modify the BLAST filters to 95% identity

Step 5 - Run BLAST

Step 6 - Close this window

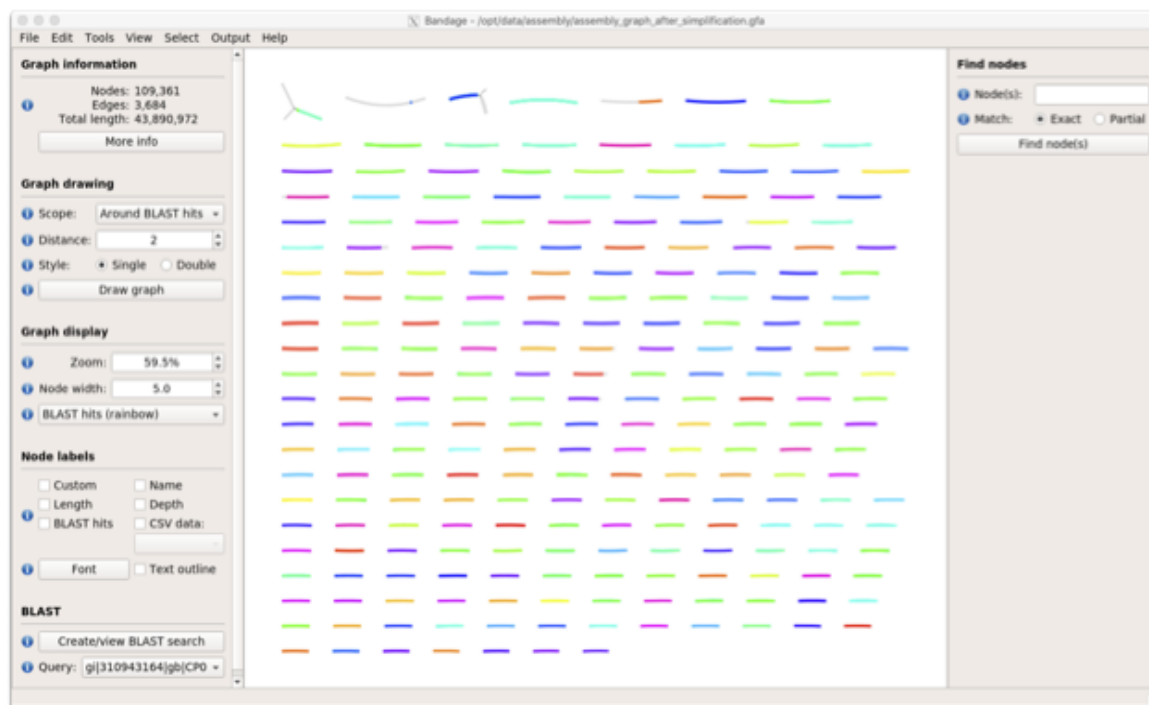
To visualise just these hits, go back to “Graph drawing” panel.

Set Scope to ‘Around BLAST hits’

Set Distance 2

The click on ‘Draw graph’

You should then see something like this:



**i** In the following steps of this exercise, we will look at performing co-assembly of multiple datasets. Each should take about 15-20 min. In case you do not manage to finish these on time, the directory **coassembly.bak** contains all the expected results.

**v** First, we need to make sure the output directories we are going to create do not already exist (MEGAHIT cannot overwrite existing directories). Run:

```
rm -rf coassembly/assembly*
```

**v** Then, perform the coassemblies with MEGAHIT, as follows:

```
megahit -1 reads/oral_human_example_1_splitac_kneaddata_paired_1.fastq -2 reads/oral_
human_example_1_splitac_kneaddata_paired_2.fastq -o coassembly/assembly1 -t 4 --k-
list 23,51,77
```

```
megahit -1 reads/oral_human_example_1_splitac_kneaddata_paired_1.fastq,reads/oral_human_
example_1_splitab_kneaddata_paired_1.fastq -2 reads/oral_human_example_1_splitac_
kneaddata_paired_2.fastq,reads/oral_human_example_1_splitab_kneaddata_paired_2.fastq -
o coassembly/assembly2 -t 4 --k-list 23,51,77
```

```
megahit -1 reads/oral_human_example_1_splitab_kneaddata_paired_1.fastq,reads/oral_human_
example_1_splitac_kneaddata_paired_1.fastq,reads/oral_human_example_1_splitaa_
kneaddata_paired_1.fastq -2 reads/oral_human_example_1_splitab_kneaddata_paired_2.
fastq,reads/oral_human_example_1_splitac_kneaddata_paired_2.fastq,reads/oral_human_
example_1_splitaa_kneaddata_paired_2.fastq -o coassembly/assembly3 -t 4 --k-list 23,51,
77
```

**v** You should now have three different assemblies, let us compare the results.



```
assembly_stats coassembly/assembly1/final.contigs.fa
assembly_stats coassembly/assembly2/final.contigs.fa
assembly_stats coassembly/assembly3/final.contigs.fa
```



How do these assemblies differ to the one generated previously with metaSPAdes? Which one do you think is best?

## 1.4 MAG generation

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

### 1.4.1 Prerequisites

For this tutorial you will need to first start the docker container by running:

```
sudo docker run --rm -it -v /home/training/Data/Binning:/opt/data microbiomeinformatics/
↪mgnify-ebi-2020-binning
```

**Note:** It's possible that the docker image is not available in dockerhub. In that case you can build the container using the [Dockerfile](#)

To build the container, download the Dockerfile and run “docker build -t microbiomeinformatics/mgnify-ebi-2020-binning .” in the folder that contains the Dockerfile.

password: training

### 1.4.2 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.

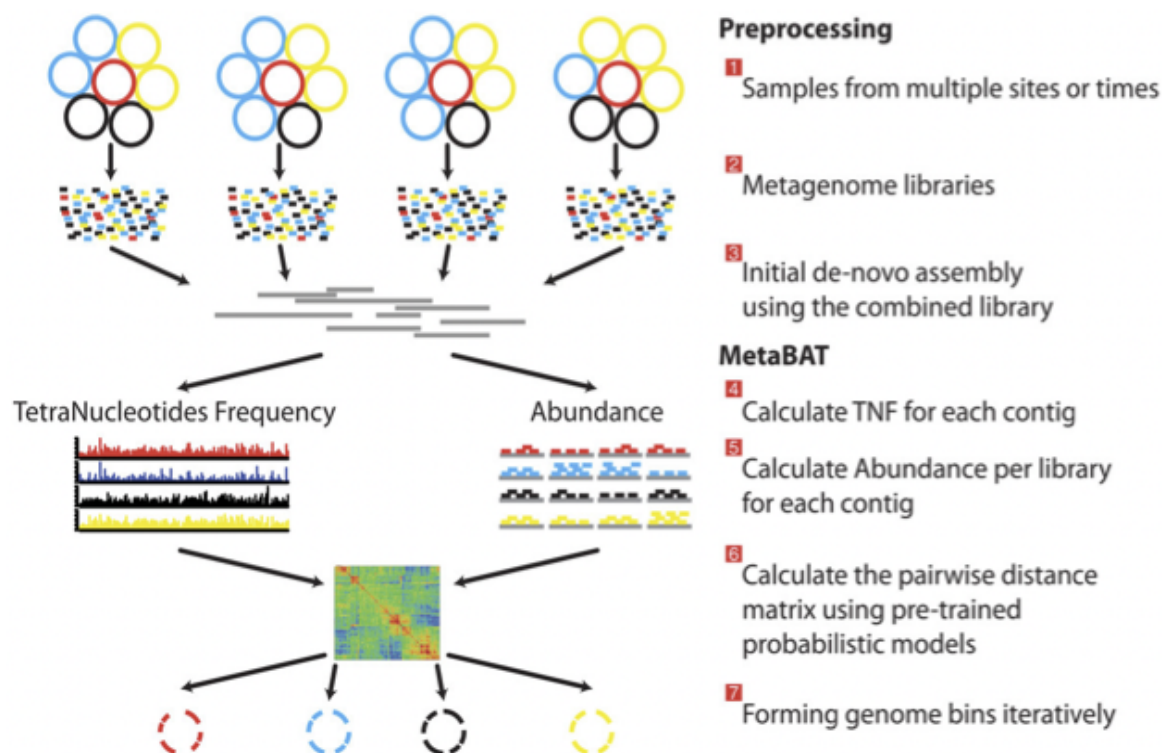


Figure 1. MetaBAT workflow (Kang, et al. *PeerJ* 2015).

**i** 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:
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

**i** Create a subdirectory where files will be output:

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

In this case, the directory might already be part of your VM, so do not worry if you get an error saying the directory

already exists. You can move on to the next step.



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

```
jgi_summarize_bam_contig_depths --outputDepth contigs.fasta.depth.txt input.fastq.sam.bam
```



Now you can run MetaBAT as:

```
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**.



Inspect the output of the binning process.

```
ls contigs.fasta.metabat-bins2000/bin*
```



How many bins did the process produce?



How many sequences are in each bin?

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
tar -xf checkm_data_2015_01_16.tar.gz -C /opt/data/checkm_data
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.

Now run CheckM with the following command:

```
cd /opt/data/assemblies
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.

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%**.

You can inspect the CheckM output with:

```
cat MAGs_checkm.tab
```

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](http://cegg.unige.ch/newick_utils)).

To do this, run the following command:

```
nw_rename checkm_answers/concatenated.tre assemblies/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/>).

In order to open **FigTree** open a new terminal window (without docker) and type **figtree**



Open the **renamed.tree** file with **FigTree** (**File -> Open**) and then select from the toolbar **File -> Export Trees**. In the **Tree file format** select **Newick** and export the file as **renamed.nwk** (or 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.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** present in `/home/training/Data/Binning/iTOL_Files/` 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?

## 1.5 MGnify Genomes Catalogues

In the presentation we will cover:

- Recap of *MAG generation*
- Overview of MGnify MAG Catalogues
- Annotations available for catalogues and genomes
- Metadata available for catalogues and genomes
- Download files available for catalogues and genomes, via web, FTP, and the API
- Search mechanisms available against catalogues

### 1.5.1 MAG Catalogues hands-on exercises

These exercises will again use the [API](#) (“Application programming interface”), showing how your scripts (e.g. Python or R) can talk to the MGnify database.

This builds on the exercises you looked at in the [API](#) practical.

The practical will use a Jupyter Notebook.

#### Do you have MAGs?



For the last few exercises, you need some MAGs. If you didn’t get as far as making your own MAGs in the *MAG generation* exercises, you can copy some we made earlier with this command in a Terminal:

```
cp -r /media/penelopeprime/Metagenomics-Nov21/Day4/day-4-example-mag-bins/* ~/mags
```

If you DID make your own, copy them to ~/mags.

#### Opening the Jupyter notebook



Open a new Terminal. There should be a Terminal application icon in the lefthand menubar.



Type these commands to enter the repository:

```
cd ~/mgnify-ebi-2021
conda activate mgnify-ebi-2021
```



That should have worked, but if you got an error you'll need to create the environment:

```
conda create --n mgnify-ebi-2021 python=3.9
conda activate mgnify-ebi-2021
pip install -r requirements.txt
```



To open the notebooks, run:

```
jupyterlab course.jupyterlab-workspace
```

Today's Notebook is "Day 4".

## 1.6 License

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

## 1.7 Viral detection and classification

### 1.7.1 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 -xzf 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.

## 1.7.2 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** directory. 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
```



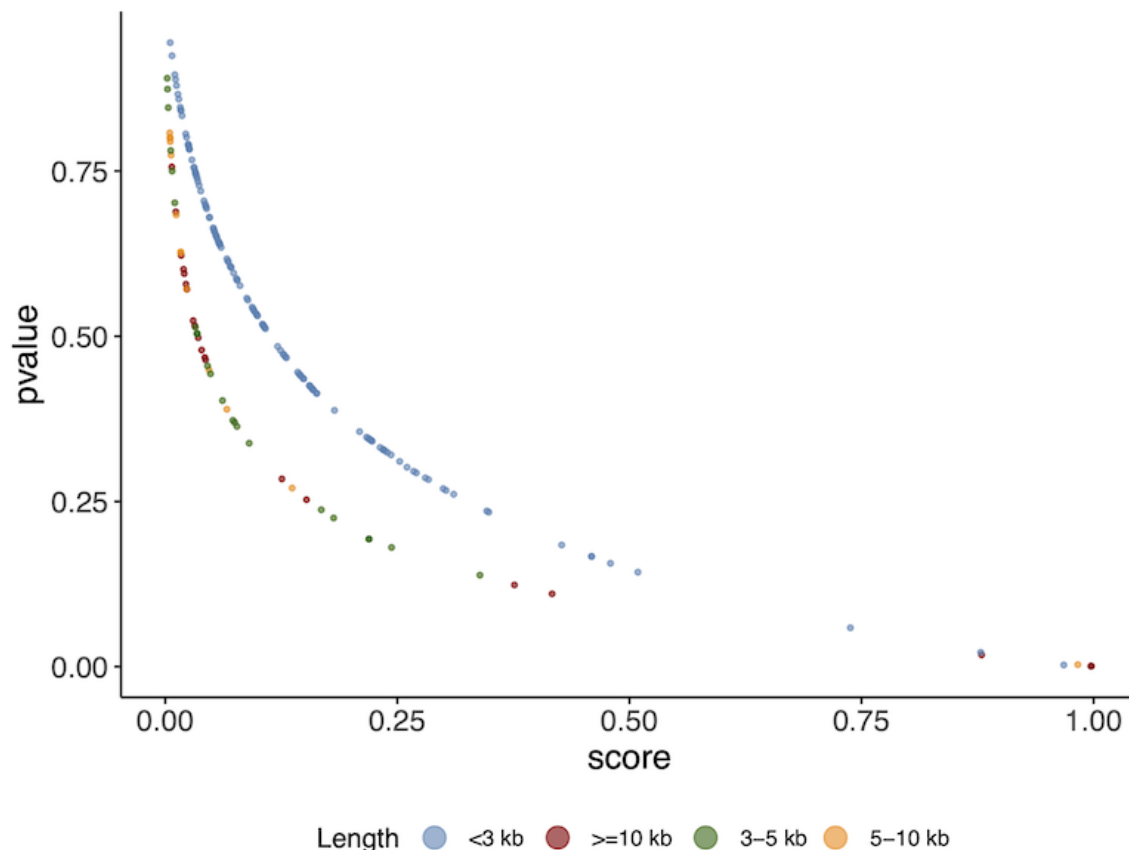
**VirSorter classifies its predictions into different confidence categories:**

- **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 **/vir-**

`ify_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**.

### 1.7.3 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 **hmmsearch** 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
```

### 1.7.4 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.



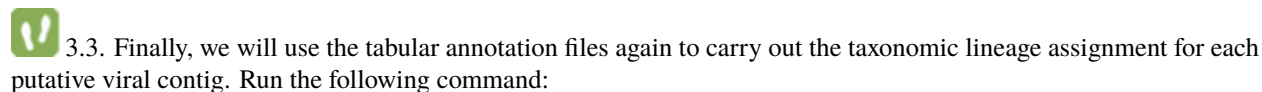
3.1. First, we are going to generate a tabular file that lists the taxonomic annotation results obtained for each protein from the putative viral contigs. We will generate this file for the putative viral sequences in each prediction category. Run the following:

```
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
```



```
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.

