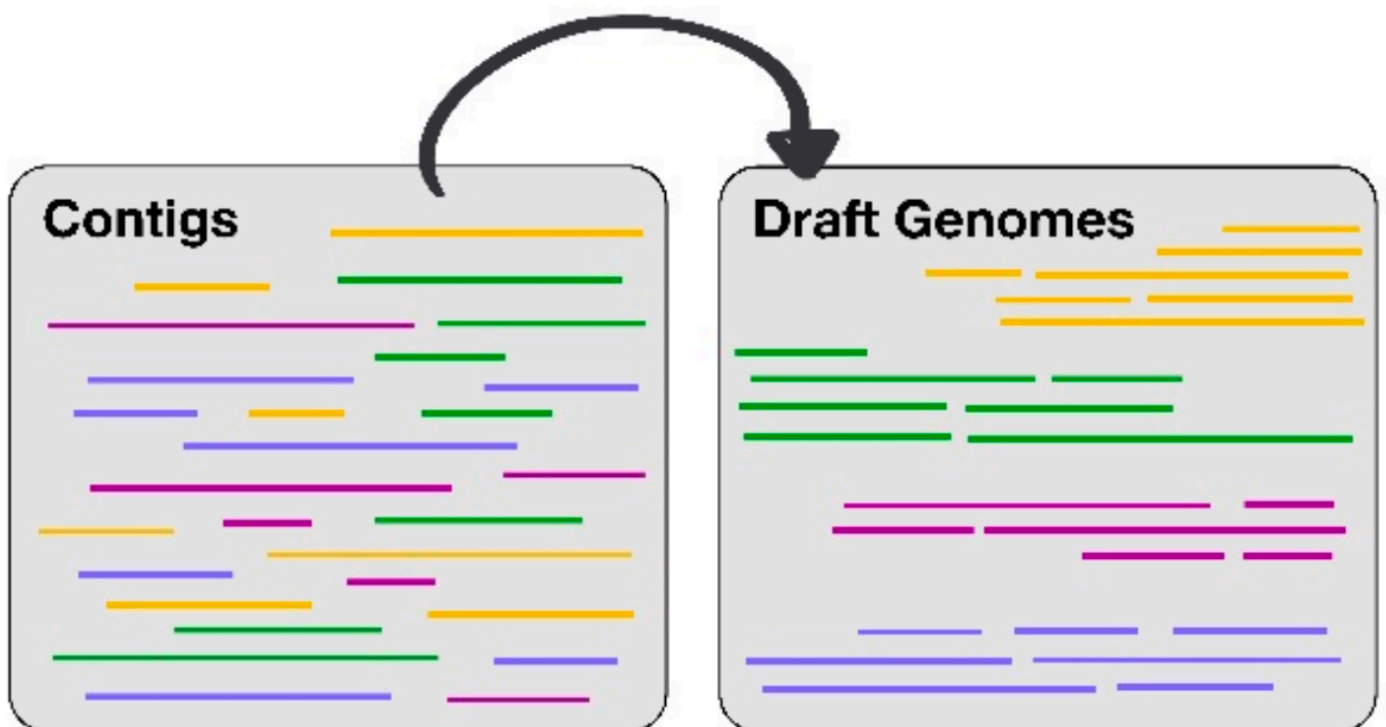




Hands on - Taxonomy independent binning

In this exercise you will perform taxonomy independent binning of both assembled contigs and directly on sequence reads. You will also check the quality of the metagenomic bins.




This exercise consists of six parts:

1. [Binning of assembled contigs](#)
2. [Assess the quality of metagenomic bins](#)
3. [Taxonomic classification of binned contigs](#)
4. [Taxonomic classification of binned contigs - Sketch](#)

5. [Binning of sequence reads](#)
6. [Taxonomic classification of binned reads](#)

1. Binning of assembled contigs

MaxBin

 **MaxBin** automates binning of assembled metagenomic contigs by first creating bins from initial identification of marker genes in the assembled sequences, then organize metagenomic sequences into individual bins based on similarities and differences in coverage and sequence composition. You can read more about **MaxBin** [here](#).

Maxbin needs both assembled metagenome and the sequence reads (alternatively reads coverage information) as input, and will report genome-related statistics, including estimated completeness, GC content and genome size in the binning summary page.

I] Go to `/practical/5/maxbin/` and create a symbolic link to the rawdata FASTQ files AND to the assembly produced by **MetaSPAdes**:

```
ln -s ~/practical/2/rawdata/sample_R* .
ln -s ~/practical/2/validation/sample_metaspades_trim.fasta .
```

II] Run **MaxBin** and create bins from the **MetaSPAdes** assembled contigs. Use the 40 marker gene sets, plot the marker genes and name the output `sample_maxbin`:

```
run_MaxBin.pl -contig sample_metaspades_trim.fasta -reads sample_R1.fastq.gz -reads
```

? Why do you think marker set 40 is better suited for this data?

► **Solution** - Click to expand

Depending on your system, it could take some minutes for **Maxbin** to finish.

Below is a description of the **MaxBin** output. Use this description to answer:

===MaxBin Output=== Assume your output file header is (out). **MaxBin** will generate information using this file header as follows.

```
(out).0XX.fasta -- the XX bin. XX are numbers, e.g. out.001.fasta
(out).summary -- a summary file describing which contigs are being classified into
(out).log -- a log file recording the core steps of MaxBin algorithm
(out).marker -- marker gene presence numbers for each bin. This table is ready to b
(out).marker.pdf -- visualization of the marker gene presence numbers using R. Will
(out).noclass -- this file stores all sequences that pass the minimum length thresh
(out).tooshort -- this file stores all sequences that do not meet the minimum lengt
(out).marker_of_each_gene.tar.gz -- this tarball file stores all markers predicted

(if -reassembly is given)
(out)_reassem/(out).reads.0xx -- the collected reads for the 0xx bin.
(out)_reassem/(out).reads.noclass - reads that cannot be assigned to any bin.
```

? How many bins did **MaxBin** generate for the assembly using the raw FASTQ files?

► **Solution** - Click to expand

? Open the file `sample_maxbin.marker` in a text editor. Why are some marker genes present in more than one copy in one bin?

► **Solution** - Click to expand

III] The file `sample_trim_maxbin.marker.pdf` contain the same information as the previous file you looked at. Open the pdf file and look at the plot.

? Open the file `sample_maxbin.summary` in a text editor. What is the estimated completeness of the most complete genome bin?

► **Solution** - Click to expand

Progress tracker

Part 1 finished

2. Assess the quality of metagenomic bins

💡 [CheckM](#) provides a set of tools for assessing the quality of genomes. It provides robust estimates of genome completeness and contamination by using collocated sets of genes that are ubiquitous and single-copy within a phylogenetic lineage. **CheckM** also provides tools for identifying genome bins that are likely candidates for merging based on marker set compatibility, similarity in genomic characteristics, and proximity within a reference genome tree.

Read more about **CheckM** [here](#)



The recommended workflow for assessing the completeness and contamination of genome bins is to use lineage-specific marker sets (Lineage-specific Workflow). This workflow consists of 4 mandatory (M) steps and 1 recommended (R) step:

```
(M) > checkm tree <bin folder> <output folder>
(R) > checkm tree_qa <output folder>
(M) > checkm lineage_set <output folder> <marker file>
(M) > checkm analyze <marker file> <bin folder> <output folder>
(M) > checkm qa <marker file> <output folder>
```

I] **CheckM** is installed in a separate **Conda** environment. Activate this environment:

```
conda activate Checkm
```

II] First you must configure the checkm database folder:

```
checkm data setRoot /opt/SfB-course/checkm_databases/
```

III] Go to the directory `~/practical/5/checkm` and make a directory here named `maxbin` and enter this directory.

IV] Make symbolic links to the 8 bins produced by **MaxBin** in the previous exercise here:

```
ln -s ../maxbin/sample_maxbin.001.fasta .
...
ln -s ../maxbin/sample_maxbin.008.fasta .
```

V] Unfortunately, **CheckM** requires more memory than you have available on the VM. We have therefore prerun **CheckM** with the following command. The output (named checkm) is located

```
~/practical/5/checkm/checkm :
```

```
checkm lineage_wf -t 32 -x fasta --tab_table -f bin_table maxbin checkm

# -t, --threads
# -x, --extension (e.g fasta)
# --tab_table -f bin_table (tab separated table named bin_table that can be directly
# ../maxbin is the folder with the FASTA files of the bins from MaxBin
# checkm is the output directory
```

CheckM can produce a number of plots for assessing the quality of genome bins.

VI] Generate a "**bin_qa_plot**" which is a visual representation of the completeness, contamination, and strain heterogeneity within each genome bin:

```
checkm bin_qa_plot -x fasta checkm maxbin plots

# plots is the output directory
```

? Does the **CheckM** completeness estimates correlate with the estimates from **MaxBin**?

► **Solution** - Click to expand

? Are any of the bins contaminated?

► **Solution** - Click to expand

III] Generate a "**gc_plot**" which is a visual plot for assessing the GC distribution of sequences within each genome bin:

```
checkm gc_plot -x fasta maxbin plots 95
```

💡 The first pane is a histogram of the number of non-overlapping 5 kbp windows with a given percent GC. A typical genome will produce a unimodal distribution. The second pane plots each sequence in the genome bin as a function of its deviation from the average GC of the entire genome (x-axis) and sequence length (y-axis). The dashed red lines indicate the expected deviation from the mean GC as a function of length. This expected deviation is pre-calculated from a set of trusted reference genomes and the percentile plotted is provided as an argument to this command. A good default value to use for this distribution parameter is 95.

IV] Look at the GC plot produced for bin001.

? Does the GC content in any of the contigs in this bin deviate significantly from the average GC content of the total contigs in the bin?

► **Solution** - Click to expand

? Would you regard these contigs as contaminants in the bin?

► **Solution** - Click to expand

Generate a "**tetra_plot**" which is a visual plot for assessing the tetra nucleotide distribution of sequences within each genome bin, similar to the GC plot above.

V] First copy the file `sample_metaspades_trim.fasta` from `~/practical/2/validation` to this directory (`~/practical/5/checkm`)

VI] Produce tetra nucleotide signatures for all sequences within a FASTA file. Tetranucleotide signatures are required for a number of the plots produced by **CheckM**:

```
checkm tetra sample_metaspades_trim.fasta sample_metaspades_trim.tsv
```


VII] Generate the "**tetra_plot**":

```
checkm tetra_plot -x fasta checkm maxbin plots sample_metaspades_trim.tsv 95
```

VIII] Look at the tetra nucleotide plot produced for bin001.

? Does the tetra nucleotide signature in any of the contigs in this bin deviate significantly from the average tetra nucleotide signature of the total contigs in the bin?

► **Solution** - Click to expand

 **Note:** It is possible to refine the bins by either merging bins or removing contigs from a bin. **CheckM** can identify genome bins with complementary sets of marker genes and perform merging. Also, you can identify which contigs have multiple marker genes and remove them.

However, it is not advised to do this unless you have strong evidence that supports this refinement. For example for contamination:


"The contamination is an estimate based on the marker genes, however it is possible (or likely) that contigs that originate from another organism may not contain marker genes. Simply removing contigs with the duplicated marker genes will improve the stats but may not mean that your genome is decontaminated and may give a false impression of quality."

IX] Before continuing on the next exercise, make sure to deactivate the **Conda** environment:

```
conda deactivate
```

Progress tracker

3. Taxonomic classification of binned contigs

 **Note:** The tools used for taxonomic classification of sequence reads are also possible to use for taxonomic classification of assembled contigs, but as mentioned yesterday, the results should be treated with caution, since they are not directly quantitative.

I] Go back to the directory `/practical/5/maxbin` Select one of the largest bins from the **MaxBin** output from the run with the raw reads (for example `sample_maxbin.008.fasta`).

? How many contigs are there in the bin?

► **Solution** - Click to expand

II] Classify the contigs in the file with large contigs using **Kraken** and name the output

`sample_kraken_maxbin8.out` :

```
kraken --db /net/software/databases/minikraken_20171013_4GB/ sample_maxbin.008.fast
```

III] Generate a taxonomic report of the **Kraken** output:

```
kraken-report --db /net/software/databases/minikraken_20171013_4GB/ sample_kraken_m
```

? Is it possible to decide from which specie this genomic bin belongs to?

► **Solution** - Click to expand

? At which taxonomic level can you possibly decide where this genomic bin belongs to?

► **Solution** - Click to expand

IV] Repeat the above steps (I - III) for the genomic bin `sample_maxbin.007.fasta` and generate a taxonomic report of the **Kraken** output.

? At which taxonomic level can you possibly decide where this genomic bin belongs to?

► **Solution** - Click to expand

As in the previous exercise, we have pre-run **Kaiju**. You can find the results from the taxonomic classification of the genomic bins from **MaxBin** here: `/practical/5/prerun/kaiju` .

V] Like yesterday, use **kaijuReport** to convert the corresponding **Kaiju** outputs as with **Kraken** (maxbin7

and maxbin8) into separate summary report files. Start with the output `sample_maxbin.008.out`, and create one report for each of the following taxonomic ranks: species, genus, family, and order. Name the output files containing the reports `sample_kaiju_maxbin8_species.summary`, `sample_kaiju_maxbin8_genus.summary`, `sample_kaiju_maxbin8_family.summary`, `sample_kaiju_maxbin8_order.summary`:

```
kaijuReport -t /net/software/databases/kaijudb/nodes.dmp -n /net/software/databases
kaijuReport -t /net/software/databases/kaijudb/nodes.dmp -n /net/software/databases
kaijuReport -t /net/software/databases/kaijudb/nodes.dmp -n /net/software/databases
kaijuReport -t /net/software/databases/kaijudb/nodes.dmp -n /net/software/databases
```

? At which taxonomic level can you possibly decide where this genomic bin belongs to? You need to open the report files...

► **Solution** - Click to expand

? At which taxonomic level did this bin belong to according to the **Kraken** report?

► **Solution** - Click to expand

VI] Make a **Kaiju** report from the `sample_maxbin.007.out` file at genus level.

? Which genus does this genomic bin belong to?

► **Solution** - Click to expand

? Which genus does this genomic bin belong to according to the **Kraken** report?

► **Solution** - Click to expand

Progress tracker

Part 3 finished

4. Taxonomic classification of binned contigs - Sketch

💡 Another very fast method for the taxonomic classification of assembled contigs is to use **SendSketch**, which is part of **BBtools**. It uses **MinHash Sketch** which is a method of rapidly comparing large strings or sets. The result is similar to **BLAST** a list of hits from a query sequence to various reference sequences, sorted by similarity but the mechanisms are very different. You can read more [here](#).

SendSketch first makes a sketch of your assembly (hash table). Then it opens an HTTP connection to

JGI's taxonomy server, and sends the sketch. The server will compare it to sketches of eg. RefSeq, and return the top hits. This is kind of convenient because the sketches sit around in memory all the time rather than needing to be loaded. You can specify other servers with sketches of eg. SILVA.

Sketch creation works like this:

1. Gather all length-K kmers in a sequence.
2. Apply a hash function to them.
3. Keep the N smallest hash codes and call this set a "sketch".

A hash code is a number generated from a sequence using a hash function. For example, consider the genome ACGTTA. This could be broken into 3 overlapping tetramers: ACGT, CGTT, GTTA. A hash function might transform ACGT -> 27, CGTT -> 97, and GTTA -> 42. In this case, if you were generating a sketch of fixed size 2, the two lowest (27 and 42) would be kept, and the other discarded. In practice, a kmer length of 31 is typically used, genomes tend to contain millions of kmers, and sketches use a fixed size of typically 10,000.

Compares query sketches to reference sketches hosted on a remote server via the Internet. The input can be sketches made by **sketch.sh**, or FASTA/FASTQ files from which **SendSketch** will generate sketches. Only sketches will sent, not sequences. It is also possible to set up at database locally and use **BBtools Sketch** to run a similar classification.

I] Perform taxonomic classifications using **SendSketch** of the genomic bins from **MaxBin** that you already have work with (maxbin7 and maxbin8). Start with `sample_maxbin.008.fasta`. Save the output to a file, eg. `sample_maxbin.008.bb.out`. In addition, write the sketch to a file using the `outskech` flag:

```
sendsketch.sh in=sample_maxbin.008.fasta out=sample_maxbin.008.bb.out
```

? Which specie is it likely that this genomic bin represents?

► **Solution** - Click to expand

By default, sketches are generated with autosize enabled. This yields variable sketch sizes (in kmers) based on genome size - around 120 for ribosomal sketches, 10000 for bacteria, and 40000 for vertebrates

II] Count the number of lines in the sketch file.

? How many k-mers does your sketch consist of?

► **Solution** - Click to expand

III] Repeat the above and perform taxonomic classifications for `sample_maxbin.007.fasta`

? Which specie is it likely that this genomic bin represents?

► **Solution** - Click to expand

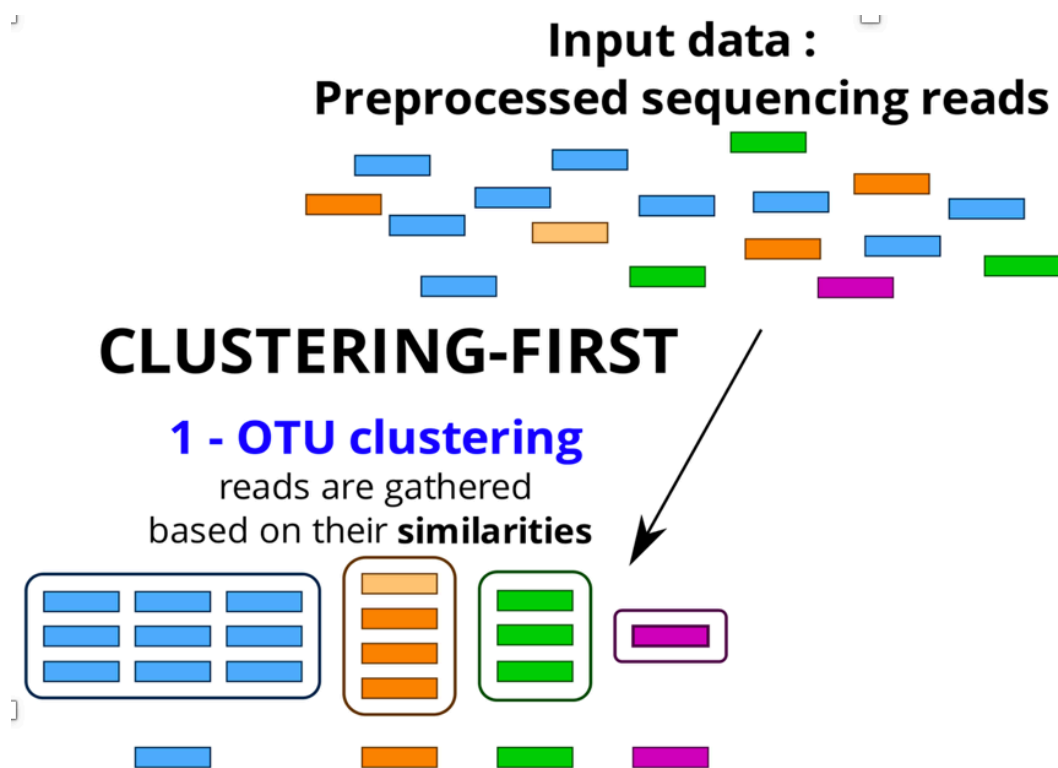
? How many k-mers does your sketch consist of?

► **Solution** - Click to expand

Progress tracker

Part 4 finished

5. Binning of sequence reads



💡 **MetaProb** is an assembly-assisted tool for reference-free binning of metagenomic sequence reads. The binning process is performed in two phases. Phase 1 groups overlapping reads into groups. Phase 2 builds the probabilistic sequence signatures of independent reads and merges the groups into clusters. **MetaProb** can run on single-end and on paired-end sequences, and only accepts decompressed FASTQ as input.

MetaProb is a quite memory intensive tool. It took 16 minutes to perform the binning on a machine with 256 GB RAM but ran out of memory on a "normal machine" with 8 GB RAM and 2 CPUs... Therefore, we have run the binning analysis with **MetaProb** beforehand using the rawdata as input. The following command was used to produce the results:

```
MetaProb -pi sample_R1.fastq sample_R2.fastq -numSp 8
```

? What does the parameters pi and numSp mean? Hint: type `MetaProb` in the terminal window.

► **Solution** - Click to expand

The result from **MetaProb** is by default written to a new directory named *output* (you must rename, move or delete this directory if you want to do the binning again). Here you will find the binning log file `binning.info`, and for each input file one `clusters.csv` and one `groups.csv` file is produced. The `sample_R1.fastq.clusters.csv` and `sample_R2.fastq.clusters.csv` will contain a list of all the reads that was clustered into bins.

I] Go to the directory `/practical/5/prerun/output_metaprobe/` and make symbolic links to the rawdata files here:

```
ln -s ~/practical/2/rawdata/sample_R* .
```

II] Open the log file `binning.info`.

? Of the 2 mill sequence reads, how many reads were clustered into bins?

► **Solution** - Click to expand

? What is the bin containing most sequence reads, and how many reads does it contain?

► **Solution** - Click to expand

? What is the smallest bin, and how many reads does it contain?

► **Solution** - Click to expand

III] Open the `sample_R1.fastq.clusters.csv` file by double clicking on the file in the file browser. The default tool to view comma-separated files (.csv) in Ubuntu is **LibreOffice**. It is a big file, so it will take a moment for it to open. In column A you will see the unique names of each read in your raw data FASTQ file. In column B you will see a number between 0 and 7 which represent each of the eight clusters was defined when performing the binning (with the `-numSp 8` flag).

You can use the script **metaprobsplit.py** written by Espen Robertsen to sort all the original reads into separate FASTQ files based on which bin it belongs to. Hence, in this case the script should produce eight separate FASTQ files for R1 and eight for R2.

The script has both a PE and a SE mode. However we recommend that you run the script in SE mode, thereby producing separate R1 and R2 reads

Important: the script writes the results to a new directory named `/output`, so remember to rename this directory when you run the script twice from the same directory.

IV] Split the sequence reads for R1 into separate FASTQ files for each bin using the script `metaprob-split.py`:

```
python metaprob_split.py --a ~/practical/2/rawdata/sample_R1.fastq.gz --aa sample_R
```

V] Rename `/output` to `/output_R1`

```
mv output output_R1
```

VI] Split the sequence reads for R2 into separate FASTQ files for each bin:

```
python metaprob_split.py --a ~/practical/2/rawdata/sample_R2.fastq.gz --aa sample_R
```

VII] Rename `/output` to `/output_R2`

```
mv output output_R2
```

VIII] Go into `/output_R1`.

? How many FASTQ files are produced?

► **Solution** - Click to expand

In order to separate the R1 from R2 files (they are named similarly) you need to rename all the 8 FASTQ files by adding R1 to the file names of all R1 FASTQ files. Also, remove the "." in the start of the file names

IX] In the `/output_R1` directory, type:

```
find . -name "*.fastq" -print | awk '{f=$0;sub(".fastq","_R1.fastq");print "mv "f" "$
```

X] Repeat the above IX in the `/output_R2` directory, but change the R1 with R2

Progress tracker

Part 5 finished

6. Taxonomic classification of binned reads

I] In the `~/practical/5/prerun/output_metaprobe/` perform a taxonomic classification using **Kraken** on

reads in bin 0:

```
kraken --db /net/software/databases/minikraken_20171013_4GB/ --paired output_R1/0_R
cut -f2,3 sample_kraken_cluster0.out > sample_kraken_cluster0.krona.in
ktImportTaxonomy sample_kraken_cluster0.krona.in -o sample_kraken_cluster0.krona.ht
kraken-report --db /net/software/databases/minikraken_20171013_4GB/ sample_kraken_c
```

? Which taxon is it likely that this genomic bin represents?

► **Solution** - Click to expand

II] Perform a taxonomic classification using **Kraken** on reads in bin 6.

? Which taxon is it likely that this genomic bin represents?

► **Solution** - Click to expand

Finally, perform taxonomic classification of binned reads after they have been assembled into contigs.

We have prerun an assembly of R1 and R2 from bin 0 using **Megahit**. The results are found here

`/practical/5/prerun/output_metaprobe/megahit_cluster0/`. The assembly was performed with the following command:

```
megahit -1 output_R1/0_R1.fastq -2 output_R2/0_R2.fastq -t 12 -o megahit_cluster0
```

III] Go to the directory containing the assembly:

? How many contigs does this bin contain, and how long is the assembly (IGV?)?

► **Solution** - Click to expand

IV] Perform a taxonomic classification of the contigs using **SendSketch** and name the output

`sample_megahit_cluster0.bb.out`:

```
sendsketch.sh in=final.contigs.fa out=sample_megahit_cluster0.bb.out
```

? Which specie is it likely that this genomic bin represents?

► **Solution** - Click to expand

We have prerun an assembly of R1 and R2 from bin 6 using **Megahit**. The results are found here

`/practical/5/prerun/output_metaprobe/megahit_cluster6/`. The assembly was performed with the following command:

```
megahit -1 output_R1/6_R1.fastq -2 output_R2/6_R2.fastq -t 12 -o megahit_cluster6
```

V] Perform a taxonomic classification of the contigs using **SendSketch** and name the output `sample_megahit_cluster6.bb.out`

► **Hint** - Click to expand

? Which specie is it likely that this genomic bin represents?

► **Solution** - Click to expand

? What does these results tell you about the binning method for this particular sample?

► **Solution** - Click to expand

Progress tracker

Complete

That was the end of the this practical - Good job 👍
