




PRACTICAL: Quality control, filtering, assembly & validation

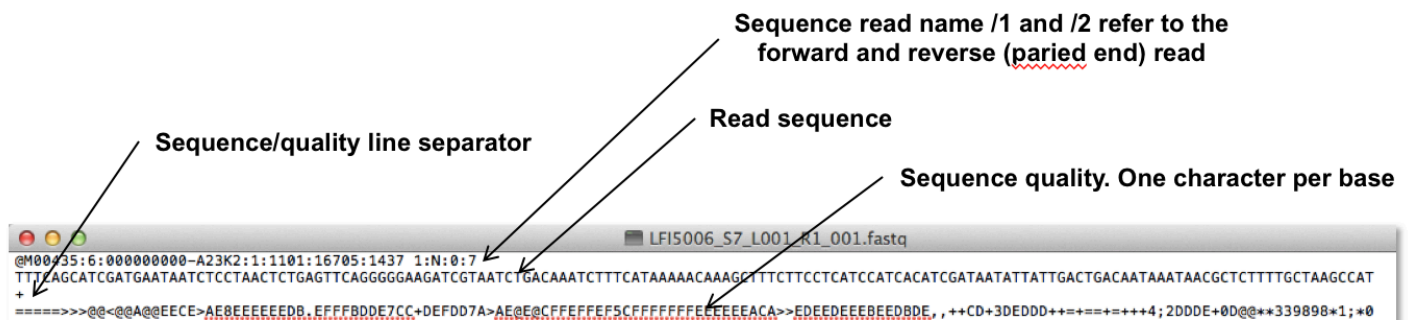
In this assignment we will be working with a metagenomic paired end sample from the marine environment, more specific the faeces of the Atlantic salmon. The main objectives of the practical are: To pre-process and assemble next-generation sequence data from this metagenomic sample, and finally validate the assembly. You will be working with Illumina random shotgun sequencing data.

This exercise consists of seven parts:

1. [Make sure all your data is there and get to know the FASTQ file format](#)
2. [Quality control - FastQC](#)
3. [Remove contamination Fastq Screen](#)
4. [Pre-processing next-generation sequence data - Trimmomatic](#)
5. [Assembly - Megahit](#)
6. [Quality assessment - MetaQUAST](#)
7. [Generate report - MultiQC](#)

 **Note:** The results you generate in this exercise will be used for input in later exercises (e.g. metagenomic binning)

1. Make sure all your data is there and get to know the FASTQ file format



I] Move to the directory where the sequence data is and view what is in the directory:

```
cd practical/2/rawdata
```

II] Open the `md5_data.txt` file and look at the content in a text editor. It should have one checksum per file.

III] Check that the sum is correct for your files by typing in the **terminal**:

```
md5sum -c md5_data.txt
```

If the files are complete, the test should return "OK" for all files

Optional: You can try to generate MD5 checksum on a file and output it to a file:

```
md5sum filename > md5_test.txt
```

The FASTQ files containing the sequence reads are normally compressed with **gzip**. The command `file` let you test the type of file.

IV] In the directory where the compressed FASTQ files are located, type:

```
file sample_R1.fastq.gz
```

? What type of file is this?

► **Solution** - Click to expand

To look at the content you must either uncompress the files, or you can use **zcat** to preview the content. **head** is a preview tool. `|` pipes the output from one command into the next command.

V] Try to run the following command:

```
zcat sample_R1.fastq.gz | head
```

? Each sequence read has four lines. What information does these lines contain?

► **Solution** - Click to expand

VI] You can preview more of the file, eg. the 100 first lines by:

```
zcat sample_R1.fastq.gz | head -n 100
```

It may be useful to know how many sequences a FASTQ file contain. Also it is useful to convert FASTQ files to FASTA format.

VII] Count numbers of sequence reads in the compressed file:

```
zcat sample_R1.fastq.gz | echo $((`wc -l`/4))
```

? Are there equal number of sequence reads in both the paired FASTQ files?

► **Solution** - Click to expand

💡 **Seqkit** is a nice and extremely fast tool for processing sequences in the FASTA or FASTQ format. You can use the **Seqkit** to convert the FASTQ file to a FASTA file in addition to other manipulations such as extracting subsequences from FASTA/Q file with reads ID (a name list file). Instructions for usage can be found [here](#).

VIII] Convert fastq.gz to FASTA:

```
seqkit fq2fa sample_R1.fastq.gz -o sample_R1.fasta
```

HINT: here is a unix one-liner to do the same. You can try it optionally:

```
gunzip -c sample_R1.fastq.gz | awk '{if(NR%4==1) {printf(">%s\n",substr($0,2));} e1
```

IX] Have a look at the FASTA file:

```
head sample_R1.fasta
```

? What is the main difference between a FASTQ and a FASTA file?

► **Solution** - Click to expand

X] Convert the reverse FASTQ to FASTA:

```
seqkit fq2fa sample_R2.fastq.gz -o sample_R2.fasta
```

XI] Concatenate the two FASTA files (head to tail) using **cat** and write out the new file:

```
cat sample_R1.fasta sample_R2.fasta > sample_cat.fasta
```

It is possible to use **Seqkit** in order to subsample sequences by number or proportion.

XII] Subsample 100 read pairs from the FASTQ files using **Seqkit**:

```
seqkit sample -n 100 -s 0 sample_R1.fastq.gz -o sub_sample_R1.fastq.gz  
seqkit sample -n 100 -s 0 sample_R2.fastq.gz -o sub_sample_R2.fastq.gz
```

? How many sequence reads are there in the reduced file:

► **Solution** - Click to expand

XIII] Subsample 10% of the reads from the R1 FASTQ file using **Seqkit**:

```
seqkit sample -p 0.1 sample_R1.fastq.gz -o porp_sample_R1.fastq.gz
```

XIV] Count the number of sequence in one of the reduced files.

? How many sequence reads were there?

► **Solution** - Click to expand

XV] Make a quick simple statistics for all the FASTQ files including the subsampled files:

```
seqkit stats -a *.fastq.gz
```

? What is the average length of the sequence reads in sample_R1.fastq.gz?

► **Solution** - Click to expand

Progress tracker

Part 1 finished

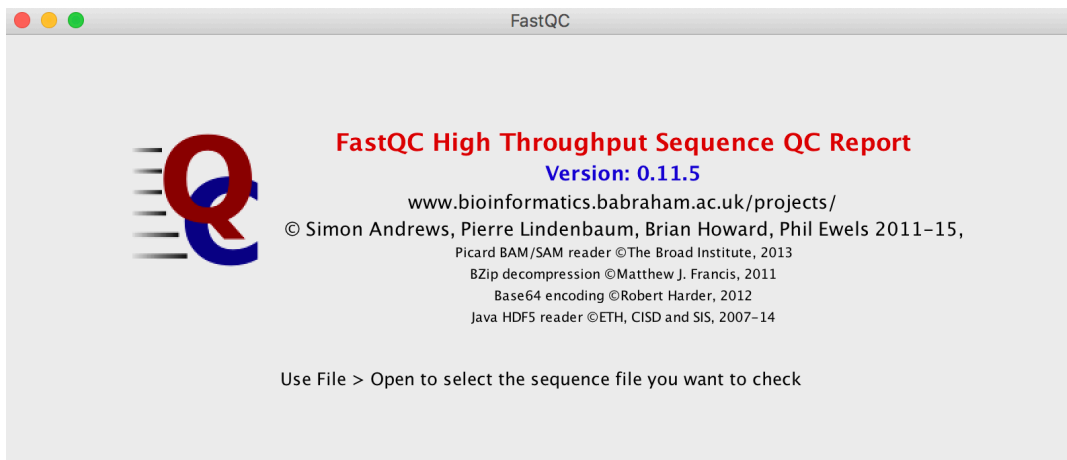
2. Quality control - FastQC



💡 **FastQC** is a quality control tool for high throughput sequence data. **FastQC** aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

We will perform the quality control of the raw data with **FastQC**, a program that has a graphical interface. Instructions and examples of the **FastQC** usage can be found [here](#) and [here](#).

I] Start **FastQC** by typing `fastqc` in a terminal. After a few second the graphical **FastQC** interface will appear.



II] In the **FastQC** GUI, click on "**File**" and select the two compressed FASTQ files `sample_R1.fastq.gz` and `sample_R2.fastq.gz` in the `/practical/2/rawdata` directory

FastQC operates a queuing system where only one file is opened at a time, and new files will wait until existing files have been processed

The left hand side of the main interactive display show a summary of the modules which were run and an evaluation of the results.

? What does green tick, the orange triangle and the red cross indicate?

► **Solution** - Click to expand

? How many sequence reads does each of your dataset consist of, and what is the sequence length range??

► **Solution** - Click to expand

? What is the average GC content of the sequences?

► **Solution** - Click to expand

? What does a quality score of 30 indicate about the accuracy, and which of the sequence files would you say have the best base calling quality?

► **Solution** - Click to expand


III] The "**Per Base Sequence Quality**" shows an overview of the range of quality values across all bases at each position in the FASTQ file. The y-axis on the graph shows the quality scores. The higher the score the better the base call. 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 are the major differences between the quality of the reads in R1 and in R2?

► **Solution** - Click to expand

IV] View the "**Per Base Sequence Content plots**". In a random library you would expect that there would be little to no difference between the different bases of a sequence run, so the lines in this plot should run parallel with each other. Does your sequence reads have any biases?

► **Solution** - Click to expand

 **Note:** Illumina believes this effect is caused by the "not so random" nature of the random priming process used in the protocol. This may explain why there is a slight overall G/C bias in the starting positions of each read. The first 12 bases probably represent the sites that were being primed by the hexamers used in the random priming process. Look at the K-mer content plot and see which K-mers are overrepresented and in which position they are.

V] Have a look at the other modules, especially the one marked with a red cross. Can you say something about the quality of these sequences, e.g duplication level, etc?

? How many reads in `sample_R2.fastq.gz` consists of only N's?

► **Solution** - Click to expand

V] Finally, save the two reports in `practical/2/rawdata`

Progress tracker

Part 2 finished

3. Remove contamination Fastq Screen

FastQ Screen

💡 Metagenomic samples isolated from a host may often contain host DNA in the final sequence library, and will therefore be sequenced together with the microbial DNA that we are interested in. This contamination may cause problems in the the downstream analysis and should be removed as early in the analysis as possible.

The tool **FastQ Screen** allows you to set up a standard set of libraries (eg. human) against which all of your sequences can be searched. You can also set up search libraries containing the genome of the particular host organism you isolated your metagenomic sample from. The DNA sequenced just have to be pre-indexed using **Bowtie2** and tell **FastQ Screen** to search against this organism. You can read more how to make you own search libraries [here](#).

The sample you are working with is isolated from salmon. We have pre-indexed the salmon genome and set up **FastQ Screen** to search against this.

I] Remove reads that are likely to be host contamination from your samples using **FastQ Screen**:

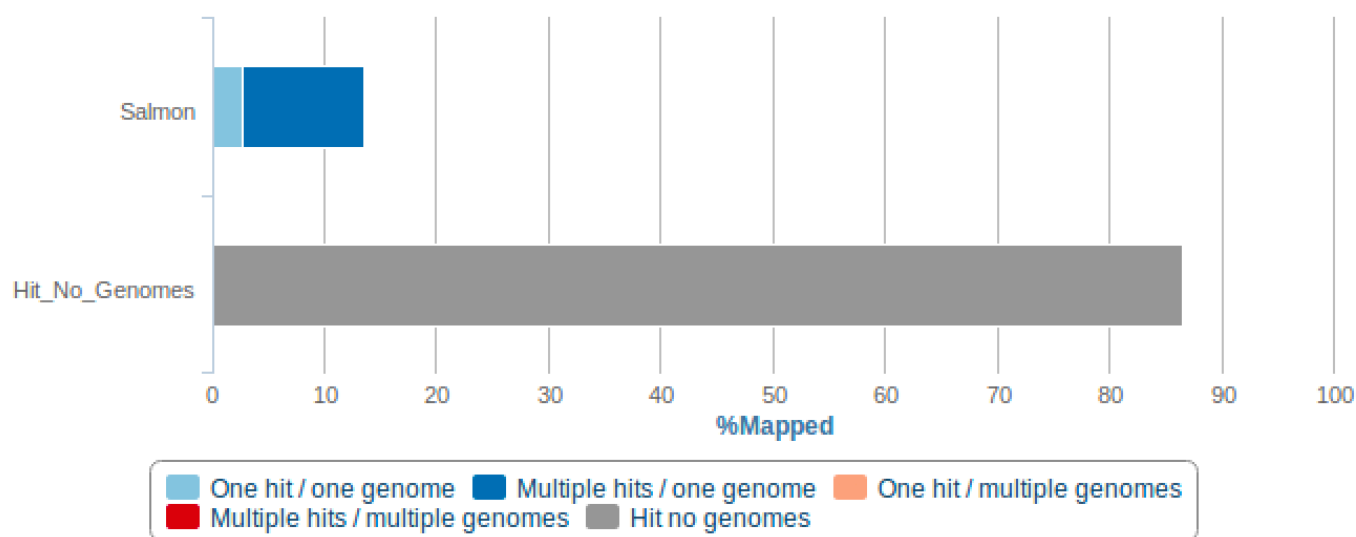
```
fastq_screen --nohits sample_R1.fastq.gz sample_R2.fastq.gz  
  
# --nohits tells the tool to write a new FASTQ file containing all reads that does
```

Screening the files for contamination will take a while - 6 minutes on a 8 GB RAM, 2 CPU machine, so it is possible to stretch your legs now:).

II] Once **FastQ Screen** is done, open the two HTML files you prodeded to look at the screening results. You can open the files either by finding them in the file browser and double clicking on the files. A **Firefox** wiondow should open displaying the reuslts, or you can start **Firefox** from the **terminal**:

```
firefox
```

Mapping Results



File	Reads processed	Unmapped	One hit / one genome	Multiple hits / one genome	One hit / multiple genomes	Multiple hits / multiple genomes
Salmon	1,000,000	864,335	27,072	108,593	0	0

? Approximately how many percent of the total reads were from the host?

► **Solution** - Click to expand

? How many reads were kept in R1 and how many in R2?

► **Solution** - Click to expand

💡 From the number of reads in the R1 and R2 files you can tell that the pairs are no longer synchronized. Many downstream tools will not accept PE files that are out of sync, therefore it can be crucial to perform a paired-end synchronization step. The tool **repair.sh** in the **BBsuite** takes reads in two potentially unsynchronized FASTQ files and writes out two synchronized FASTQ files and a file containing unpaired reads.

III] Synchronize `sample_R1.tagged_filter.fastq.gz` and `sample_R2.tagged_filter.fastq.gz` with the **repair.sh**:


```
repair.sh in=sample_R1.tagged_filter.fastq.gz in2=sample_R2.tagged_filter.fastq.gz
```

? How many unpaired reads were there?

► **Solution** - Click to expand

For metagenomes, merging overlapping read pairs can improve the final assembly.

IV] Use **BBmerge** of the **BBsuite** to merge overlapping reads:

```
bbmerge.sh in1=sample_R1_pair.fastq.gz in2=sample_R2_pair.fastq.gz out=sample_merge
```

💡 **Note:** If you want to check whether the job you started is running, you can open another terminal window and type:

```
top
```

top is the Unix version of the **Activity/Resource monitor** on Mac or Windows. If your job is running, it should be listed in top (in this case COMMAND should read repair.sh)

In order to exit **top**, type:

```
q
```

? How many read paired were joined?

► **Solution** - Click to expand

V] Now you have generated two PE files that are in sync, but also two files with unpaired reads (`sample_unpair.fastq.gz` and `sample_merged.fastq.gz`). To keep this a bit tidy, concatenate the two unpaired reads files:

```
zcat sample_merged.fastq.gz sample_unpair.fastq.gz | gzip -c > singletons.fastq.gz
```

Progress tracker


Part 3 finished

4. Pre-processing next-generation sequence data - Trimmomatic

Trimmomatic: A flexible read trimming tool for Illumina NGS data

Citations

Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.

 **Trimmomatic** is a filtering/trimming tool written in java. **Trimmomatic** is a fast, multithreaded command line tool that can be used to trim and crop Illumina (FASTQ) data as well as to remove adapters. There are two major modes of the program: Paired end mode and Single end mode, therefore you will need to run the tool twice: once for the paired data, and once for the file with the merge reads (single end). For the full explanation of the parameters see manual [here](#).

I] In the terminal window type:

```
TrimmomaticPE -h
```

This will display a list of options and parameter settings you can apply on your sequence data

The quality check of the reads using **FastQC** showed that there were an enrichment of certain K-mers towards the end of the reads. The quality of the bases also dropped toward the end.

II] Use **Trimmomatic** to trim the sequence reads. You must specify both the input (-fastq and -fastq2). In addition, filter all reads with average quality below phred score 20 (AVGQUAL). Trim the reads in both ends: the first 5 nt in the 5' (HEADCROP), and all reads below threshold quality 3 in both ends the 3' (LEADING and TRAILING). Finally, filter the reads below 75 nt (MINLEN).

We also want you to write out a log file from the run. **Trimmomatic** writes the log to stderr (and not stdout). Here is a small hack that concatenates stdout + stderr:

```
your command 2> file_with_stdout_and_stderr
```

```
TrimmomaticPE -phred33 sample_unmerged_R1.fastq.gz sample_unmerged_R2.fastq.gz samp
```

? Can you explain the arguments you have used to run **Trimmomatic**?

► **Solution** - Click to expand

III] You also need to trim the file containing the merged read pairs:

```
TrimmomaticSE -phred33 singletons.fastq.gz singletons_trimmomatic.fastq.gz AVGQUAL:
```

IV] Perform a Fast Quality Control of the trimmed sequence data, and compare the preprocessed data with the raw data from the first **FastQC** run.

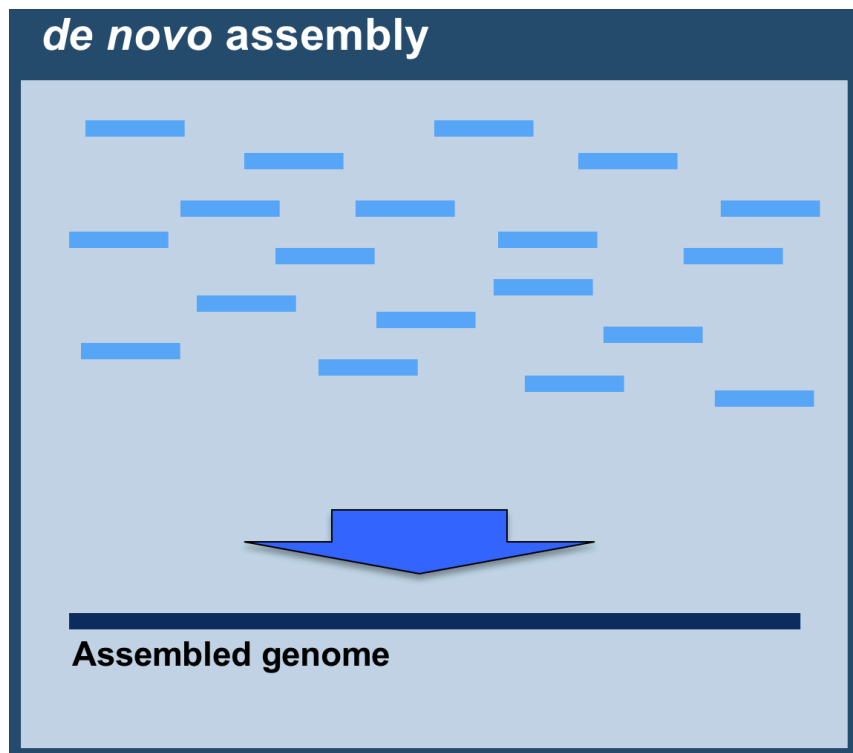
? What are the main differences before and after trimming the reads?

► **Solution** - Click to expand

Progress tracker

Part 4 finished

5. Assembly - Megahit



💡 We will use the assembler **Megahit** to assemble the trimmed data. **Megahit** is an ultra fast assembly tool written for the assembly of metagenomic data. You can read more about the assembler [here](#).

Assembling metagenomic data can be very resource demanding. This data set was tested on a virtual machine with 32 GB RAM and 8 cores and took 22 minutes to assemble!! You can choose to start the assembly now, and continue on the last part of this exercise in the practical.

I] **ALTERNATIVELY** you can go to the directory `/practical/2/sample_megahit_trimmed/` and view the

prerun results. In this directory we provide you the result from the **Megahit** assembly so you don't have to wait for the assembly to finish. **Megahit** was run with these parameter settings:

```
megahit -1 sample_trimmomatic_R1.fastq.gz -2 sample_trimmomatic_R2.fastq.gz -r sing
```

II] Depending on if you chose to run the assembly or not, go to the directory where the assembled contigs are:

```
cd /sample_megahit_trimmed  
or  
cd /sample_megahit_trim
```

III] Since you will be working with several assemblies, rename the FASTA file with the contigs to `sample_megahit_trim.fasta`:

```
mv final_contigs.fa sample_megahit_trim.fasta
```

IV] Count the number of contigs in the FASTA file:

```
grep -c "^>" sample_megahit_trim.fasta
```

? How many contigs did you find?

► **Solution** - Click to expand

? What is the size of the largest contig? HINT: Sort the contigs by size using **Seqkit**

► **Solution** - Click to expand

Progress tracker

Part 5 finished

6. Quality assessment - MetaQUAST

QUAST

Quality Assessment Tool for Genome Assemblies by [CAB](#)

💡 **MetaQUAST** evaluates and compares metagenome assemblies based on alignments to close

references. It is based on **QUAST** genome quality assessment tool, but addresses features specific for metagenome datasets. The tool will divide all contigs into groups aligned to each reference genome they align to. You will find more information about **MetaQUAST** [here](#).

We have pre-run the assembly of the same data you have been working on, but using a different assembler called **MetaSPades**. In addition, we run assembly directly on the raw data and on the data after removing host contamination. The contig files are located in directory `practical/2/validation`

I] First, move the assembly you made with the trimmed data to the directory named `/validation`. If you are in the directory containing the assembly `sample_megahit_trim.fasta`:

```
mv sample_megahit_trim.fasta ../validation .
```

II] Change directory to the `/validation` and check that all four assemblies (FASTA files) are there:

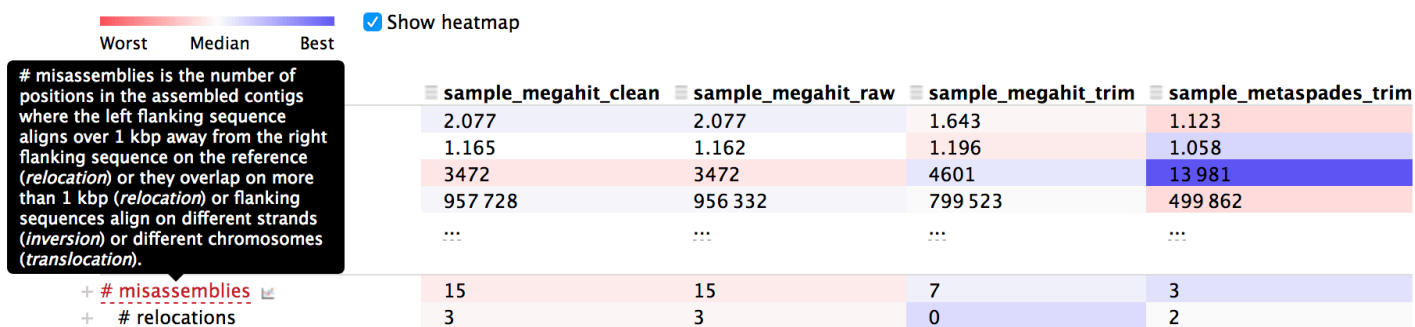
```
cd ../path_to/validation
ll
```

In this directory, there is also a file named `reference_list.txt`. This text file contain a pre-made list of reference genomes we know are in the sample. To save compute time, you can provide this list of reference genomes together with the input files. Alternatively, you let **MetaQUAST** identify which species are present by searching for 16S rRNA sequences in the contigs and produce a list of references from this. The reference genomes will be downloaded from NCBI and the contigs will be aligned against these genomes. This will make the run time longer.

III] Run **MetaQUAST** comparing the four assemblies:

```
metaquast.py *.fasta --references-list reference_list.txt -R references -t 4
```

IV] Open the **MetaQUAST** report in a web browser. There is a lot of information about your assemblies here. **Tip:** if you mouse over any of the text descriptions in the table (or plot) more detailed information about the particular descriptor. Use the report to evaluate which performs best/worst.



? Which of the assemblies is the largest and the smallest (in bp)?

► **Solution** - Click to expand

? Which of the assemblies produces the longest contig?

► **Solution** - Click to expand

? Which of the assemblies produces most unaligned contigs?

► **Solution** - Click to expand

? Which of the assemblies produces the longest sum of partially unaligned contigs and what species does this assembly have less presence of relative to the other assemblies?


► **Solution** - Click to expand

? Which of the assemblies produces most mismatches relative to the reference genomes?

► **Solution** - Click to expand

? Which of the assemblies has most aligned contigs (largest fraction) aligned to the reference genomes and how many percent of the reference is covered?

► **Solution** - Click to expand

 Show heatmap

	sample_megahit_clean	sample_megahit_raw	sample_megahit_trim	sample_metaspades_trim
Genome statistics				
+ Genome fraction (%)	2.077	2.077	1.643	1.123
+ Duplication ratio	1.165	1.162	1.196	1.058
+ Largest alignment	3472	3472	4601	13 981
+ Total aligned length	957 728	956 332	799 523	499 862
+ NGA50
Misassemblies				
+ # misassemblies	15	15	7	3
+ Misassembled contigs length	9252	9252	5078	26 868
Mismatches				
+ # mismatches per 100 kbp	2565.19	2561.55	2749.56	2534.32
+ # indels per 100 kbp	34.97	34.8	44.76	46.64
+ # N's per 100 kbp	0	0	0	0
Statistics without reference				
+ # contigs	18 924	19 438	17 067	7829
+ Largest contig	19 011	19 008	47 543	45 250
+ Total length	16 315 720	16 630 264	16 569 070	11 339 349
+ Total length (>= 1000 bp)	6 561 823	6 598 879	8 407 330	7 995 557
+ Total length (>= 10000 bp)	37 399	37 396	129 802	1 321 296
+ Total length (>= 50000 bp)	0	0	0	0

Have a look at the contig versus reference plot.

? To which specie does the largest number of contigs align to and what is the genome size of the and number of contigs of this specie?

► **Solution** - Click to expand

? How many contigs from the assembly made with **MetaSPAdes** align to this specie?

► **Solution** - Click to expand

V] Select the *Aliivibrio_logei* genome from the list of reference genomes at the bottom of the webpage. A new page will open with only mapping information against this specie.

? Which assembler produces the largest contig that align to this reference genome and how long is it?

► **Solution** - Click to expand

? How large fraction of this reference genome is covered by the **MetaSPAdes** assembly?

► **Solution** - Click to expand

? Change the view of the plot from "**Plots**" to "**Total aligned len**". What is the total length of the contigs from `sample_metaspades_trim` aligning to *Aliivibrio_logei*?

► **Solution** - Click to expand

Optional exercise:

Icarus generates contig size viewer and one or more contig alignment viewers. Contig size viewer contigs ordered from longest to shortest. Click on the "**View in Icarus contig browser**" and go to the contig size viewer.

[Try to fade contigs shorter than 10000 bp](#)

? Why do you think so many of the largest contigs are tagged as unaligned?

► **Solution** - Click to expand

? How many bp is the longest correctly aligned contig, which assembly produced this contig, which specie does it align to, and what is the percent sequence identity?

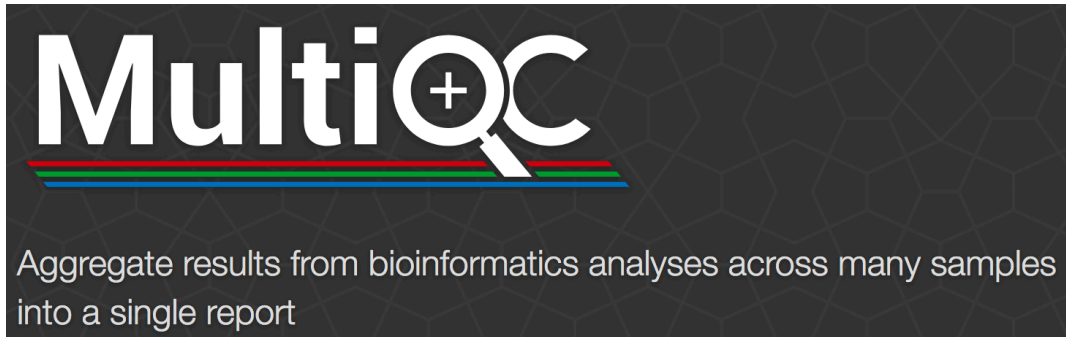
► **Solution** - Click to expand

? Use the search function to identify the contig named `NODE_6_length_25184_cov_30.2503`. **MetaSPAdes** produces longer contigs than **Megahit** on these data. The end of this contig from **MetaSPAdes** is flagged as "misassembled". Is it always better with obtaining the longest contigs?

► **Solution** - Click to expand

Progress tracker

7. Generate report - MultiQC



MultiQC is a reporting tool that parses summary statistics from results and log files generated by other bioinformatics tools. **MultiQC** doesn't run other tools for you - it's designed to be placed at the end of analysis pipelines or to be run manually when you've finished running your tools.

When you launch **MultiQC**, it recursively searches through any provided file paths and finds files that it recognises. It parses relevant information from these and generates a single stand-alone HTML report file. It also saves a directory of data files with all parsed data for further downstream use.

You can get an overview of **MultiQC** supported tools [here](#).

In order to run **MultiQC** you need to use a **Conda** environment with a different python installation than the default python that comes with the operative system.

I] To activate this environment, type:

```
conda activate multiqc
```

When activated, you will see that the prompt has changed to `(multiqc) kursXX@kurslabXX:`

II] Go to your analysis directory `/practical/2` and run **MultiQC**, followed by a list of directories to search. At it's simplest, this can just be `.` (the current working directory):

```
multiqc .
```

? The results from which tools did **MultiQC** include in the final report?

► **Solution** - Click to expand

? How large fraction of the "singleton" reads were removed by **Trimmomatic**?

► **Solution** - Click to expand

? From the "**General Stats**" part of the report what is the average GC content of the reads in `sample_R1` ?

► **Solution** - Click to expand

? From the **FastQC** part of the report "**Per Sequence GC Content**", how many reads in `sample_R1` have GC content of 40%?

► **Solution** - Click to expand

III] Deactivate the **Conda** environment:

```
conda deactivate
```

Progress tracker

Complete

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