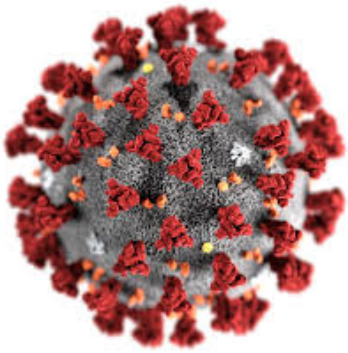
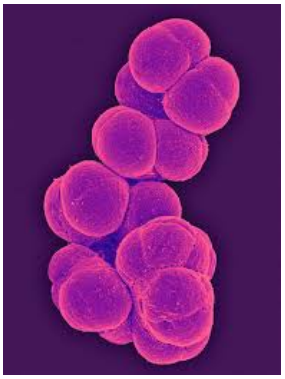
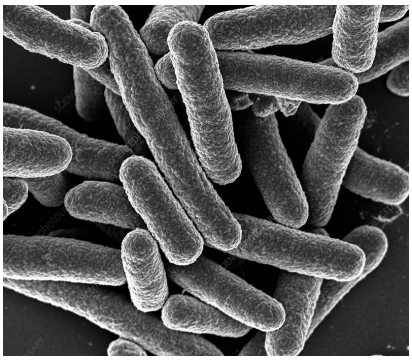
# Genome Biology and Genomics

Tom Williams, School of Biological Sciences
(tom.a.williams@bristol.ac.uk)

Session 2: Genome Assembly

**Overview:** Today's practical contains two main sections. In the first, you will be guided through assembly of a viral genome from a patient sample. In the second, you will apply the assembly skills you have learned to analyse a sample of bacterial DNA from a somewhat impure culture. Both exercises use Illumina MiSeq short reads (2x300bp).

**Learning objectives:** Get experience with applying the genome assembly workflow to realistic datasets. Get experience with some of the most widely used software for each step in the process (read quality assessment, trimming, assembly, assembly assessment, taxonomic classification, gene calling, assembly visualisation). Solidify and build on experience using command line tools. Use bioinformatics to answer biological questions about organisms in their environment.

## General notes

- These practicals are not assessed, but you may find it helpful to keep some notes on your progress in a format that makes sense to you. You could consider a Word processing document, a Jupyter notebook, or anything in between. The timed assessment at the end of the course directly tests the skills you are learning in these practicals, so you may find keeping detailed notes of use at that stage.
- Students on this course have a fairly broad range of background knowledge and existing experience with genomics, bioinformatics, working at the command line, and so on. To help to deal with this, the practical contains some activities labelled as "optional". If you

quickly complete the main tasks and are looking for something to do, you may find these more interesting/challenging/time consuming/useful than the main activities. These are truly optional, and you should not feel nervous if you do not complete them.

- Text in the `Consolas font` indicates a command you might run in a bash session or job script. The convention `myprogram <input file>` means that <input file> should be replaced with the name of the input file. For example, if your input file was called hello.fasta, the above example indicates that you should run `myprogram hello.fasta` on the command line. You should not use the characters "<>" on the command line unless you mean to. Feel free to use informative names for your program outputs and inputs, not just the default ones in the example commands below.
- Most bioinformatics tools will print something helpful if you run them with -h, --h, -help, or --help arguments.
- All of these analyses can either be run on your laptop/some other machine (with a UNIX environment), or on the BluePebble cluster. If the latter, please use job submission scripts and do not run any jobs on the head node. If you run jobs on a remote machine (such as BluePebble), you will need to use scp or an alternative to copy files back and forth between your machine and the server.
- Please ask for help if you are stuck, or if you have questions about the underlying theory or rationale of what you have been asked to do. It's also OK to ask your more knowledgeable friend, or to try to solve issues yourself by googling for the answer.
- When in doubt, searching chunks of sequence against the NCBI RefSeq or GenBank databases may be a good way to understand what sort of data you are dealing with. This can be done interactively using a web browser at https://blast.ncbi.nlm.nih.gov/Blast.cgi.

## Running jobs on BluePebble

If you plan to work through the practical on BluePebble, you should first log in in the usual way, then request an interactive job:

```
srun --nodes=1 --ntasks-per-node=1 --time=01:00:00 --pty bash -i
```

This will launch an interactive bash session on one of the compute nodes, so you can run commands directly. **Whatever you do, do not run jobs directly on the head (login) node!**

All of the needed tools are installed on BluePebble already. To use them, you need to load the appropriate module(s) using the `module add <insert name>` command. To see a list of available modules, type `module avail`

The modules needed for this practical are:
**FastQC**: module add apps/fastqc/0.11.9
**Trimmomatic**: apps/trimmomatic/0.39
Then run trimmomatic with:

java -jar /sw/apps/Trimmomatic-0.39/trimmomatic-0.39.jar

**MEGAHIT**: module add lang/python/anaconda/3.8-2021-TW
**QUAST**: module add lang/python/anaconda/3.8-2021-TW
**Prokka**: Loading prokka requires loading three modules in the correct order. (If you get things wrong, use module unload moduleName to remove the offending module and start again):

module add lang/python/anaconda/3.8-2021-TW
module add lang/perl/5.30.0-bioperl-TW
module add apps/blast/2.2.31+

## Installing software on your local machine

In general, this involves visiting the software website or github repository and following the installation instructions. As people following this route often know what they are doing, we have decided to give the precise commands for the tools installed on BluePebble and loaded via `module add` in the tutorial below; you will need to modify them slightly for a local install (e.g., you may need to change the path to the binary).

(NB: If you cannot find something, try Googling for it. The Github pages of software packages usually point to some documentation. Sources like Stack Overflow, Biostars can also be very useful).

FastQC: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
Trimmomatic: http://www.usadellab.org/cms/index.php?page=trimmomatic
MEGAHIT: https://github.com/voutcn/megahit
QUAST: http://quast.sourceforge.net/docs/manual.html
Kraken2: https://github.com/DerrickWood/kraken2/wiki/Manual
Prokka: https://github.com/tseemann/prokka

## Part 1. Clone the course repository from Github

All of the data needed to follow the course practicals can be found in the Github repository for the course (https://github.com/Tancata/GenomeBiologyandGenomics). Use git to clone (create a local copy of) the repository in your home directory (or somewhere else convenient, such as a directory you might create for this course):

```
module load tools/git

git clone https://github.com/Tancata/GenomeBiologyandGenomics.git
```

This will create a directory, “GenomeBiologyandGenomics”, containing the course materials. Later in the course, you may need to update the repository (as I update materials to fix errors that you might helpfully report to me, or post more materials for later practicals). You can do this by cd’ing into the GenomeBiologyandGenomics directory, and typing:

```
git pull
```

# Part 2. Viral genome assembly from Illumina MiSeq short read data

A colleague in Wuhan has sent you a dataset of 2x300bp MiSeq reads (`virus_all_R1.fastq`, `virus_all_R2.fastq`) from a patient suffering a new respiratory viral illness. Assemble the viral genome and determine the closest known relatives of this virus in the public sequence databases.

**1. Use FASTQC to evaluate the quality of the sequence data.**

```
fastqc read_file_1 read_file_2
```

The FastQC output files are written to a zip archive, which can be unzipped using the `unzip` command.

(a) Do the data contain adaptors, or other over-represented k-mers?
(b) Do the data contain low-quality bases? Errors in base calls interfere with de Bruijn-graph based assembly.

**2. Use trimmomatic to delete any low-quality (score < 3) bases from the read data.**

```
java -jar /sw/apps/Trimmomatic-0.39/trimmomatic-0.39.jar PE
virus_all_R1.fastq virus_all_R2.fastq output_1_paired.fastq
output_1_unpaired.fastq output_2_paired.fastq output_2_unpaired.fastq
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
```

(NB: If running on your own machine, modify the path `/sw/apps/Trimmomatic-0.39/trimmomaic-0.39.jar` to point to your installed version).

**3. Use FastQC to assess the quality of the trimmed reads. Have any issues been corrected?**

**4. Assemble the reads using MEGAHIT.** This is a memory-efficient short-read assembler that performs reasonably well in comparison to the more resource-intensive assemblers. You should be able to use it to run the assemblies on your laptop or personal computer, if needed.

Note: the input (forward and reverse) files for this step will be the output_1_paired and output_2_paired files from the trimming step.

```
/sw/lang/anaconda.3.8-2021-TW/bin/megahit -1 output_1_paired.fastq -2
output_2_paired.fastq -o assembly_all_contigs
```

**5. Compute basic assembly statistics (N50 and L50) from the assembly using QUAST.** Note that you may have to type the path to quast.py in order to execute this command correctly.

```
quast.py assembly_all_contigs/final.contigs.fa
```

What do the N50 and L50 suggest to you about the quality of the assembly? Did you assemble all of the viral genome into a single contig?

**6.** The output folder will contain a FASTA-format file, `final.contigs.fa`, that contains all of the contigs assembled from the input read data. **From what biological source(s) were these contigs derived?** A quick way to get some idea is to copy a large-ish chunk of sequence from one of the the contigs and search for it using NCBI BLASTN (https://blast.ncbi.nlm.nih.gov/Blast.cgi?LINK_LOC=blasthome&PAGE_TYPE=BlastSearch&PROGRAM=blastn).

A more systematic and high-throughput approach is to use `kraken2` to evaluate the taxonomic origin of each contig using a custom database set up for this practical.

```
Kraken2 --db GBG final.contigs.fa
```

Kraken2 uses a fast (faster than BLAST) similarity search based on comparison of k-mers. *However, running Kraken2 requires a bit more hard disk space and RAM than you may have access to.* I'll run this step live during the practical and show you the results on my system. The output file from the above analysis is included in the Git repo under "session2"; it's `kraken2_output.txt`.

**7. What was in the DNA sample? Are there any surprises?**

---

[OPTIONAL] Write a Python script that sorts the assembled contigs into one file for each organismal source. The result should be a set of one or more files, each containing contigs from just one taxonomic source.

---

**8. Use prokka to call genes** and do basic annotation on the viral contigs. Prokka is a very useful gene calling and genome annotation tool for viral and prokaryotic genomes. Since prokaryotic gene structures are very constrained, *ab initio* prediction just from sequence features is usually quite reliable. In next week's session, we'll look at some more in-depth

annotation tools, but prokka is a very good start. As you will see when you watch Prokka run, it is actually a wrapper script that uses a number of different analyses to identify different kinds of genomic features on the input contigs.

```
prokka --kingdom Viral <sequences.fa>
```

Confirm your taxonomic classification by inspecting the gene annotations produced by prokka, and using NCBI protein BLAST.

**9.** You may like to **visualise the gene models predicted by Prokka on the viral genome.** You can use a genome viewer software such as Artemis (Sanger; https://github.com/sanger-pathogens/Artemis) or Integrative Genomics Viewer (IGV; Broad; http://software.broadinstitute.org/software/igv/) to do this. ASCIIGenome (https://asciigenome.readthedocs.io/en/latest/) is a command line alternative, if you prefer to work in the terminal.

In IGV, you can load a FASTA genome (for example, a “final.contigs.fa” file) using Genomes => Load Genome from File. PROKKA annotations for such a file can then be loaded with “File => Load from File”, choosing the .gff file. How does the structure and coding features of the assembled genome compare to the published reference (https://www.nature.com/articles/s41586-020-2008-3)?

# Part 3. Bacterial genome assembly from Illumina MiSeq short read data

A collaborator has given you a set of DNA sequence reads (`env_all_1.fastq`; `env_all_2.fastq`) obtained from the culture of an environmental bacterium that has proven difficult to grow in the lab. After inoculating with soil from the site where the bacterium was originally isolated, they have finally managed to get the thing to grow to a high enough level to obtain DNA. Initial efforts to identify the lineage of bacteria using amplicon sequencing produced results that are difficult to interpret.

1. **Using the same workflow as above, assemble the genome.**
2. **How do the N50 and L50 scores for this assembly compare to those of the viral assembly? Does this assembly give a better, or worse, representation of the underlying organismal genome?**
3. **What do you notice about the taxonomic source of the contigs? Might this explain your collaborator’s difficulties in assigning taxonomy?** (NB: a second kraken2 output file is provided for you to parse).
4. **Why do you think these “contaminants” are present?**