# Practical: Assembling a bacterial genome from long reads

## Background

De novo long-read genome assembly can be done using only long reads but is typically done in combination with short reads, called hybrid assembly. Hybrid assemblies can be made in two ways; by polishing a long-read only assembly with short reads or by using long reads to bridge the contigs of a short-read assembly.

The quality of the assembly you make will reflect the quality of the reads you use as input so remember to quality control your input before starting the assembly.

We did, however, skip how to subset a large dataset so we will begin with that right after downloading the result files we will be working with.

**Polishing a long-read assembly with short reads**



**Short-read-first hybrid assembly**



In this practical you will make a long-read-only assembly using Flye and polish it with both long and short reads. And you will make a short-read-first hybrid assembly with Unicycler.

## Overview

**Step 1**: Download result files

**Step 2:** Filter input

**Step 3:** Long-read assembly with Flye

**Step 4:** Polish assembly with long reads using Medaka

**Step 5:** Polish assembly with short reads using Polypolish

**Step 6:** Hybrid assembly with Unicycler

**Extra 1**: Use rMLST to look for contamination

**Extra 2:** Visualize and compare assemblies

## Step 1: Download result files

Some of the programs you are going to run in this exercise will take a very long time to finish, therefore I have run the programs for you and put the result files on google drive for you to download.

1. First you need to make a folder for the result files.
mkdir [folder\_name]
2. Download the data.
3. Unzip the downloaded files into the folder you made in step 1.1.

## Step 2: Filter input

A sequencing depth above 100 is very unlikely to improve your assembly and is in fact likely to introduce errors in the assembly due to high representation of miscalled bases. Furthermore, a very high sequencing depth will increase the computational burden of the assembly unnecessarily and it is therefore recommended to filter the data if the sequencing depth is above 100.

1. Locate the Ec001\_super.q10.fastq.gz which is a raw version of the E. coli Ec001\_super.fastq.gz reads. This will be our input data but first we need to filter it.
2. Use cd to go the nanopore\_reads folder
3. Take a quick look at how many bases are in our raw file.
#Replace [raw\_reads.fastq.gz] with the name of the fastq.gz file with the raw reads.
zcat [raw\_reads.fastq.gz] | paste - - - - | cut -f 2 | tr -d ‘\n’ | wc -c
4. The E. coli we are working with has a genome size of approximately 5M. Filter the reads until the sequencing depth is 100.
For this we are going to use [Filtlong](https://github.com/rrwick/Filtlong), a tool for filtering long reads by length and quality.
Look at the help message for filtlong

filtlong -h

1. Make a new folder. This is where you will put your output from filtlong
2. Use cd to navigate to your new folder.
3. Run filtlong with the option –target\_bases set to 500000000 and remember to give the full path to the input reads.

#We want to filter our reads so we keep the best 500Mbps
#Filtlong outputs an unzipped file. To get a gzipped output we can *pipe* the output into gzip

filtlong --target\_bases 500000000 [full\_path/input\_reads.fastq.gz] | gzip > [name\_of\_output\_file.fastq.gz]

1. What information does filtlong give you?

## Step 3: Flye assembly

Now we are ready to assemble our E. coli genome and for this we will use [Flye](https://github.com/fenderglass/Flye), a de novo assembler for long sequencing reads from PacBio or Nanopore.

Look at the help message and consider which arguments to use. Specifically look at the options for nanopore input: --nano-raw --nano-corr --nano-hq

flye -h

1. Use cd to navigate to /home/student/BTG/Nanopore\_assembly
2. Start the flye assembly with the following command:

#Run flye. Remember to consider how many threads you can use (its 2) and what to call the output folder.
flye --nano-hq [path\_to\_filtered\_reads/input\_reads.fastq.gz] -t [threads] --deterministic -o [output\_dir]

1. Make sure the program runs properly. Flye takes 1+ hours to run so I already made the assembly for you. Stop the program on your computer with “Ctrl” + “c”.
2. Look at the output from Flye that you downloaded in the beginning of the exercise.
3. Look at the flye assembly file (assembly.fasta). How many contigs are in it?
grep “>” assembly.fasta

## Step 4: Polish with Medaka

[Medaka](https://github.com/nanoporetech/medaka) is a polishing tool that uses long reads to polish a long-read assembly. It takes a long-read assembly and long reads as input. It can be useful to polish the assembly several times until no further changes are made but for now we are just going to polish one time.

1. Look at the help message for medaka.
2. You are going to polish the flye assembly you just looked at with the long reads you also used to make the flye assembly.
To run Medaka you need to specify a model.

**Optional!** See which models are available:
medaka tools list\_models

1. Run Medaka. Consider what to name the output directory.

medaka\_consensus -i [path\_to\_filtered\_reads/input\_reads.fastq.gz] -d [path\_to/flye\_assembly.fasta] -m r941\_min\_sup\_g507 -o [output\_dir]

It should take approximately 5 min to run the program.

Use cd to navigate into the new folder medaka just created.
Look at the output from medaka. The only important file right now is consensus.fasta which is the polished assembly.

## Step 5: Polish with Polypolish

Polish the Medaka polished assembly with Illumina reads using [Polypolish](https://github.com/rrwick/Polypolish). Polypolish works by aligning short reads to the long-read assembly and correcting errors. It can be necessary to polish the assembly several times until no further changes are reported but for now we are just going to polish one time.

To align the short reads we will need bwa which we will clone from github.

1. Make a folder named ‘git\_repos’

mkdir git\_repos

1. Go to your new git\_repos folder

cd git\_repos

1. Clone bwa from github

git clone https://github.com/lh3/bwa.git

cd bwa; make

1. Index the medaka polished long-read assembly with bwa

#Change ‘Medaka\_folder’ with the actual name of your medaka folder
git\_repos/bwa/bwa index Medaka\_folder/flye\_medaka\_assembly.fasta

1. Find the Illumina reads that matches the long reads we are working with
2. Align the short reads to the long-read assembly with bwa
Remember to point to your medaka polished assembly.

#Change ‘Medaka\_folder’ with the actual name of your medaka folder
git\_repos/bwa/bwa mem -a Medaka\_folder /flye\_medaka\_assembly.fasta /[full\_path-to]/illumina\_R1.trimmed.fastq.gz] > [alignment\_1.sam]

#Change ‘Medaka\_folder’ with the actual name of your medaka folder
git\_repos/bwa/bwa mem -a Medaka\_folder /flye\_medaka\_assembly.fasta /[full\_path-to]/illumina\_R2.trimmed.fastq.gz] > [alignment\_2.sam]

1. With the short reads aligned you can now run Polypolish

#Change ‘Medaka\_folder’ with the actual name of your medaka folder
polypolish Medaka\_folder/flye\_medaka\_assembly.fasta [alignment\_1.sam] [alignment\_2.sam] > polypolish\_output.fasta

## Step 6: Assemble with Unicycler hybrid assembly method

Unicycler is an assembler that can make short-read assemblies, long-read assemblies and, if supplied with both short and long reads, short-read-first hybrid assemblies.

1. Start the short-read first hybrid assembly with the following command:

unicycler -1 [illumina\_R1.trimmed.fastq.gz] -2 [illumina\_R2.trimmed.fastq.gz] -l [input\_reads.fastq.gz] -o ./[unicycler\_hybrid\_output\_dir]

1. Make sure Unicycler runs properly. Unicycler would take a very long time to run so I have provided a finished assembly. Stop the program on your computer with “Ctrl” + “c”.
2. Look at the output from Unicycler in the folder you downloaded in step 1
grep “>” unicycler\_output/assembly.fasta
3. We do not like that the file is named ‘assembly.fasta’ -it is too unspecific. So we will rename it:
mv unicycler\_output/assembly.fasta unicycler\_output/unicycler\_hybrid\_assembly.fasta

## Extra 1: Look for contamination

After making an assembly you can use rMLST to check for contamination.

Use the Flye assembly to check for contamination in the Nanopore data.

## Extra 2: visualize and compare assemblies

To visualize our assemblies we will use [Bandage](https://rrwick.github.io/Bandage/)

1. Open Bandage
2. Under ‘File’ click ‘Load graph’



1. Find your Spades assembly gfa file and click ‘open’
2. Click ‘Draw graph’



1. Open new windows of Bandage (unfortunately you cannot have multiple tabs open so to compare assembly graphs you need a new window for each assembly).
Open a new window by right clicking the Bandage icon and clicking ‘open’
2. Open your flye long-read-only assembly and your short read first hybrid assembly and compare all three assemblies.
3. See the total length of the assemblies by marking all contigs:

