Bacterial genome assembly - Basic

This exercise assumes the following:
 * A native Linux environment / terminal
 * A functional micromamba installation
 * Ability to install environments from
.yaml files
 * Reads of a few bacterial genomes
 * All files are stored within the home directory for the
`gebt` user. Feel free to change locations according to own needs!

Authors

These exercises where authored and tested by Povilas Matusevicius and Kasper Thystrup Karstensen

Introduction

The goal of this practical is to help building a pipeline for handling bacterial data from raw reads to sequence type determination. The practical is build to hold you in the hand, while guiding you throughout the process.

Therefore it is recommended to attempt the Intermediate exercise and only consult this exercise to guide your progress through the intermediate exercises.

Prerequisites

For this exercise we will use the `BTG_QC`, `BTG_spades_4.0.0`, `BTG_alignment` environments. In addition, we will use the following **files** and **file paths**.

- Path for read mate 1: */home/gebt/BTG/SequenceData/Ec016.illumina_R1.fastq.gz*
- Path for read mate 2: */home/gebt/BTG/SequenceData/Ec016.illumina_R2.fastq.gz*
- Path to Output folder: */home/gebt/BTG/Day8_pipelines/*bacterial_asembly

Executing commands from different environments

A drawback of using conda/mamba environments, is that not all software can run in the same environment. When running bash scripts, the shell doesn't always know how to invoke the conda/mamba commands. With `micromamba` though, there is a hack that can be utilized: Micromamba allow for cherry picking commands from different environments using `micromamba run`. This way environments are not required to be loaded. So in order to run e.g. MLST from within any (or no) environment, the following command can be used in the bash script.

```
micromamba run -n BTG_alignment mlst
```

Bacterial pipeline

In the exercise, your task is to write a script called "bacterial_assembly.sh" that will provide a QC report, trim raw reads, generate an assembly for sample, and run mlst on "*Ec016*" it.

Setting up parameters

1. Navigate to the `Day8_pipelines` folder in `BTG` directory
2. Make a script file called `bacterial_assembly.sh` and open it.

> 💡 If you use `nano` to make the file, you don't have to open it afterwards!

3. In the very beginning of the file, paste the following code chunk:

```
#!/bin/bash

read1="/home/gebt/BTG/SequenceData/Ec016.illumina_R1.fastq.gz"
read2="/home/gebt/BTG/SequenceData/Ec016.illumina_R2.fastq.gz"
output_dir="/home/gebt/BTG/Day8_pipelines/bacterial_assembly"

# Determine sample name from read filename
read1_filename=$(basename $read1)
sample_name=${read1_filename%.illumina_R1*gz}

# Generating output folders
fastqc_out=$output_dir/$sample_name/fastqc
fastp_out=$output_dir/$sample_name/fastp
spades_out=$output_dir/$sample_name/spades
mlst_out=$output_dir/$sample_name/mlst
results_dir=$output_dir/Results

mkdir -p $fastqc_out
mkdir -p $fastp_out
mkdir -p $spades_out
mkdir -p $mlst_out
mkdir -p $results_dir
```

6. Add print statements to ensure that the script contains the correct information. add the following lines to the end of the script. (They are to be removed again shortly).

```
echo "This is the read pair, consisting of $read1 and $read2"
echo "This is the sample name: $sample_name"
echo "This is the output folder: $output_dir"
echo "FastQC output will be written to $fastqc_out and fastp output will be in $fastp_out"
echo "Spades output will be in $spades_out"
echo "The final results directory is $results_dir"
```

5. Save and exit the file.
4. Add execution permission using: `chmod u+x bacterial_assembly.sh`
7. Execute the script by running `./bacterial_assembly.sh`
9. Once it succeeds, reopen the file with `nano` and remove the `echo` statements.

QC and Read trimming

To ensure that we get a great overview of quality parametrics we utilise `FastQC` and impose semi-strict filtration criteria using `fastp` introduced during the Quality Assurance on Illumina Reads exercises.

> 💡 Use `cd` and `ls` to investigate outputs after every step.

1. Add the following lines of code to the script

```
micromamba run -n BTG_QC fastqc -o $fastqc_out --memory 2048 --threads 6 --quiet $read1 $read2
micromamba run -n BTG_QC fastp -i $read1 -o $fastp_out/"$sample_name"_trimmed_R1.fastq.gz -I $read2 -
```

🚨 Warning these are long commands, meaning that you likely cannot see full command in one line, make sure you copy full command!!!

2. Execute the pipeline to ensure everything is working so far. If you forgot how to - check step7 in the segment above.

🎓 **Pro advice:** There are many steps, and it is easy to make a typing errors (some of the commands are very long!). So make your script one step at the time, and check that it works, before moving on to the next step. This can most easily be achieved by having two terminal open simultaneously, both with the loaded environment. One terminal handles the coding, while the other handles execution.

Remember, you can easily disable commands in your script simply by adding a comment symbol (`#`) at the start of the line. Once you are ready to include the commands again, remove the comment symbol again.

▼ The script so far

```
#!/bin/bash

read1="/home/gebt/BTG/SequenceData/Ec016.illumina_R1.fastq.gz"
read2="/home/gebt/BTG/SequenceData/Ec016.illumina_R2.fastq.gz"
output_dir="/home/gebt/BTG/Day8_pipelines/bacterial_assembly"

# Determine sample name from read filename
read1_filename=$(basename $read1)
sample_name=${read1_filename%.illumina_R1*gz}

# Generating output folders
fastqc_out=$output_dir/$sample_name/fastqc
fastp_out=$output_dir/$sample_name/fastp
spades_out=$output_dir/$sample_name/spades
mlst_out=$output_dir/$sample_name/mlst
results_dir=$output_dir/Results

mkdir -p $fastqc_out
mkdir -p $fastp_out
mkdir -p $spades_out
mkdir -p $mlst_out
mkdir -p $results_dir

micromamba run -n BTG_QC fastqc -o $fastqc_out --memory 2048 --threads 6 --quiet $read1 $read2
micromamba run -n BTG_QC fastp -i $read1 -o $fastp_out/"$sample_name"_trimmed_R1.fastq.gz -I $read2 -O
```

Adding the assembler to the pipeline

- After you trimmed low quality reads and determined that the general quality of reads will suffice, you need to make an assembly. For this purpose you can use various assemblers, some of the popular ones are **spades**, **skesa**, **unicycler** and many others. For this task we will be using spades, below is a command that runs spades on our trimmed reads files:

```
micromamba run -n BTG_spades_4.0.0 spades.py --isolate -1 $fastp_out/"$sample_name"_trimmed_R1.fastq.gz -
```

Determine sequence type

- One of the most useful result after you have assembled the sequence is to determine the type of it. In this task we will be using mlst (multilocus sequence typing)

```
micromamba run -n BTG_alignment mlst "$spades_out"/contigs.fasta --quiet --label $sample_name > $mlst_out/
```

Results

Lets collect all relevant information in a Results folder, so they are easily accessible from the Results directory

- Generate summary of all relevant tools using `MulitQC` (Currently only `FastQC` works, MutliQC must be updated in order to work with `fastp`)
- Copy important results files to the Results directory

```
# Generate a report on output and collect relevant files
micromamba run -n BTG_QC multiqc -o $results_dir -qf $output_dir
cp $fastp_out/"$sample_name"_trimmed_*.fastq.gz $results_dir/.
cp $spades_out/contigs.fasta $results_dir/$sample_name.fasta
cp $mlst_out/$sample_name.tsv $results_dir/.
```

Congratulations on your very own Bacterial assembly pipeline!

▼ Solution

```
#!/bin/bash

read1="/home/gebt/BTG/SequenceData/Ec016.illumina_R1.fastq.gz"
read2="/home/gebt/BTG/SequenceData/Ec016.illumina_R2.fastq.gz"
output_dir="/home/gebt/BTG/Day8_pipelines/bacterial_assembly"

# Determine sample name from read filename
read1_filename=$(basename $read1)
sample_name=${read1_filename%.illumina_R1*gz}

# Generating output folders
fastqc_out=$output_dir/$sample_name/fastqc
fastp_out=$output_dir/$sample_name/fastp
spades_out=$output_dir/$sample_name/spades
mlst_out=$output_dir/$sample_name/mlst
results_dir=$output_dir/Results

mkdir -p $fastqc_out
mkdir -p $fastp_out
mkdir -p $spades_out
mkdir -p $mlst_out
mkdir -p $results_dir

micromamba run -n BTG_QC fastqc -o $fastqc_out --memory 2048 --threads 6 --quiet $read1 $read2
micromamba run -n BTG_QC fastp -i $read1 -o $fastp_out/"$sample_name"_trimmed_R1.fastq.gz -I $read2 -O
micromamba run -n BTG_spades_4.0.0 spades.py --isolate -1 $fastp_out/"$sample_name"_trimmed_R1.fastq.g
micromamba run -n BTG_alignment mlst "$spades_out"/contigs.fasta --quiet --label $sample_name > $mlst_o

# Generate a report on output and collect relevant files
micromamba run -n BTG_QC multiqc -o $results_dir -qf $output_dir
cp $fastp_out/"$sample_name"_trimmed_*.fastq.gz $results_dir/.
cp $spades_out/contigs.fasta $results_dir/$sample_name.fasta
cp $mlst_out/$sample_name.tsv $results_dir/.
```

Run the pipeline with different sample

- Try to run whole pipeline on different sample: ***SRR27240827.*** *You will need to change R1 and R2 reads path.*

```
read1="/home/gebt/BTG/SequenceData/Ec004.illumina_R1.fastq.gz"
read2="/home/gebt/BTG/SequenceData/Ec004.illumina_R2.fastq.gz"
```