Whole genome assembly workshop

In this exercise, we will do two projects: Illumina short reads assembly and PacBio long reads assembly. Both are bacterial genomes. First do assembly with paired-end Illumina data.

Part 1. Trim adapters from raw reads

1. Copy the data files to you working directory.

```
mkdir /workdir/$USER

cd /workdir/$USER

cp /shared_data/assembly_workshop_2019/*fastq.gz ./
```

2. Remove adapters from the raw reads.

The Illumina adapter sequences (either Tru-seq or Nextera) and low quality sequences should be removed from the reads before assembly. Trimmomatic is a software for this purpose.

```
java -jar /programs/trimmomatic/trimmomatic-0.39.jar PE -phred33
SRR1982238_1.fastq.gz SRR1982238_2.fastq.gz r1.fastq.gz u1.fastq.gz r2.fastq.gz
u2.fastq.gz ILLUMINACLIP:/programs/trimmomatic/adapters/TruSeq3-PE-2.fa:2:30:10
LEADING:10 TRAILING:10 SLIDINGWINDOW:4:15 MINLEN:150
```

There will be 4 new files created after this step, r1.fastq.gz, r2.fastq.gz, u1.fastq.gz, u2.fastq.gz. We will use the r1, r2 and u1 files for next step.

The r1 and r2 files contain paired-end reads after trimming. The u1 and u2 files contain single-end reads after trimming, due to one read of the pair too short after trimming. You would find u1 file much larger than the u2 file. These are reads from short DNA fragments (shorter then read length, so that the read pair completely overlap), Trimmomatic only keep one end of these sequences and put them in the u1 file.

Part 2. Do assembly with SPAdes

[DO IT AT HOME] It could take an hour to finish the run. Run the software in "screen".

SPAdes automatically select k-mer size for you based on reads length (up to k=127 for the installation on BioHPC computers). If you need to define custom k-mer values, you can use "-k" option, e.g. -k 85,95,105,115.

```
export OMP_NUM_THREADS=6
/programs/spades/bin/spades.py -t 6 --careful --s1 u1.fastq.gz --pe1-1
r1.fastq.gz  --pe1-2 r2.fastq.gz -o spades_run
```

- OMP_NUM_THREADS=6: OMP_NUM_THREADS is an environment variable to define maximum OpenMP threads. This number must be larger than the "-t" option in your SPAdes command. This is necessary as SPAdes uses OpenMP for parallelization.
- -t: Number of threads;
- --careful: Instruct SPAdes to polish the assembly by correcting errors.

Part 3. Assessment with QUAST

It would take an hour for the assembly to finish. While you are waiting for the run to finish, you can use the SPAdes results we have prepared for you to finish the rest of the workshop.

Copy the SPAdes results to your working directory.

```
cp -r /shared_data/assembly_workshop_2019/spades_results ./
cd spades_results
```

A few files of interest:

- contigs.fasta: Fasta file of assembled contigs
- scaffolds.fasta: Fasta file of assembled scaffolds
- spades.log

Let's examine the spades.log file in the output directory.

Check the steps in the SPAdes pipeline:

```
grep "=====" spades.log
```

What is the estimated genome size?

```
grep "genome size" spades.log
```

You would find several genome sizes estimated based on different kmer sizes. I would use the medium value.

Now we will generate some statistics of the assembly use the software Quast. We will run Quast on both the contig and scaffolds.

```
/programs/quast-5.0.2/quast.py contigs.fasta

/programs/quast-5.0.2/quast.py scaffolds.fasta
```

Many report files will be generated. The most important numbers are summarized in the last 10 lines of the report.txt file.

```
 tail quast_results/*/report.txt
```

The numbers we are interested are:

Largest contig, Total length, N50.

In this particular run, you would find N50 is the same for contigs and scaffolds. But the largest scaffold is much larger than the largest contig.

Part 4. Evaluate with BUSCO

BUSCO is a tool to evaluate the completeness of gene space in your assembly, by checking the presence/absence of common genes of the species you are working on.

First, we need to download the BUSCO dataset that is closest to your genome. Since the genome we assemble here is in the genus of *Listaria*, and BUSCO has a dataset for *Bacillales* (the "order" above *Listeria*) , we will download the file bacillales_odb9.tar.gz.

```
wget https://busco.ezlab.org/datasets/bacillales_odb9.tar.gz

tar xvfz bacillales_odb9.tar.gz
```

Now we run BUSCO. Make sure you are in the directory: /workdir/$USER/spades_results

```
cp /programs/busco-3.1.0/config/config.ini /workdir/$USER/
cp -r /programs/Augustus-3.3.2/config /workdir/$USER/

export AUGUSTUS_CONFIG_PATH=/workdir/$USER/config

export BUSCO_CONFIG_FILE=/workdir/$USER/config.ini


export PYTHONPATH=/programs/busco-3.1.0/lib/python3.6/site-packages
export PATH=/programs/busco-3.1.0/scripts:/programs/Augustus-
3.3.2/bin:/programs/Augustus-3.3.2/scripts:$PATH

run_BUSCO.py --in ./contigs.fasta --lineage_path ./bacillales_odb9 --mode genome
--out busco_bacillales --cpu 8
```

The results you would want to see is in the file:
run_busco_bacillales/short_summary_trinityBUSCO.txt.

```
cat run_busco_bacillales/short_summary_busco_bacillales.txt
```

We would like to see the "C" score close to 90% or better. "C" score is the the percentage of BUSCO genes in full length. In this file, you would a string:

C:97.0%[S:96.6%,D:0.4%],F:0.4%,M:2.6%,n:526

The result shows that among the 526 core-genes in bacillales, 510 (97%) are in full length in this assembly, and only 14 (2.6%) genes are missing. This is a really good BUSCO sore.

Part 5. Assemble PacBio data with Canu

In this second project, we will assemble PacBio reads of a E. coli genome.

The data file is downloaded from https://www.ebi.ac.uk/ena/data/view/SRR10405213. The expected genome size is 4.6MB.

[DO IT AT HOME] Do the following assembly step at home. It would take an hour to finish. Do it in "screen".

```
cd /workdir/$USER
cp /shared_data/assembly_workshop_2019/SRR10405213.fastq.gz ./
/programs/canu-1.8/Linux-amd64/bin/canu  useGrid=false maxThreads=8
maxMemory=24g -p ecoli -d /workdir/$USER/ecoli genomeSize=4.6m  -pacbio-raw
SRR10405213.fastq.gz
```

- Run "/programs/canu-1.8/Linux-amd64/bin/canu -options" to see all options.
- The option genomeSize is required. You can supply with the estimated genome size, and it does not need to be accurate.
- Set "maxThreads" "maxMemory" if you share the computer with other users. The default will be all available threads and memory.

- -p: output file name;
- -d output directory

After it is done, you would see two fasta files:

*.contigs.fasta: Final filtered assembly result. If there are two alleles for diploids genome, only one allele is included in this file.

*.unitigs.fasta: This file includes all alternative paths and not filtered, e.g. allelic haplotypes are represented as two sequences.

There is also a *.report file which gives you the statistics of the raw data and final assembly, including N50 for the raw reads, error corrected reads and assembly, as well as total length of the assembly.

We have pre-run the software canu. You can use copy over the pre-run results with this command.

```
cp -r /shared_data/assembly_workshop_2019/ecoli/ ./
```

Next, you will polish the assembly. PacBio has its own tool "arrow" for polishing genomes which are included in its GUI software package SMRT Link (https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=442#c) .

Here we will use Pilon to polish the genome. Pilon is primarily developed for polishing genomes with Illumina reads. Latest version of Pilon provides limited support for polishing with long reads. To do this, you will first run minimap2 to align the raw reads to the assembled genome:

```
 /programs/minimap2-2.17/minimap2 -a ecoli/ecoli.contigs.fasta
SRR10405213.fastq.gz | samtools view -b - > ecoli.bam

samtools sort -@ 6  -m 5G -T ./ -o ecoli.sorted.bam ecoli.bam

samtools index ecoli.sorted.bam
```

Now run pilon:

```
java -jar /programs/pilon-1.23/pilon-1.23.jar --genome ecoli/ecoli.contigs.fasta
--pacbio  ecoli.sorted.bam --output ecoli.pilon
```

A new polished fasta file "ecoli.pilon.fasta" will be created.