Genome assembly

In this exercise, you will do two projects: Illumina short-read assembly and PacBio long-read assembly. Both are small bacterial genomes. When you work with large eukaryotic genomes that require more memory and CPU cores to assemble, you will need to use more powerful machines on BioHPC.

Some steps could take very long time to run (marked as **[DO IT AT HOME]** in the instructions). Please skip these steps for now. Use the result files we provide you to continue with. You can finish the "DO IT AT HOME" steps later during the week.

1. Prepare the working directory

1.1 Copy the data files to you working directory.

```
mkdir -p /workdir/$USER/project1

cd /workdir/$USER/project1

cp /shared_data/genomic_2020/project1/* ./

ls -l
```

You should see 5 files.

SRR1982238_1.fastq.gz & SRR1982238_2.fastq.gz: Illumina short-read data;

SRR10405213.fastq.gz: PacBio (CLR) long-read data;

spades_results.tar.gz: assembly results from short-read data;

canu_results.tar.gz: assembly results from long-read data.

1.2 Getting familiar with "screen"

If you do not know about Linux "screen" or "tmux" commands, now it is time to get familiar with it.

Most of the tools you will be using in this exercise take long time to finish. You will need to use the "screen" persistent sessions to run the software, so that you can safely detach from the session, and the job will keep running in the background on the server.

```
#Start screen. You would notice that "[screen 0 ...]" shows up at the header of
the terminal window.
screen

#Now you are in the "screen" persistent session, and you can run any software in
the session now.
ls -l

#To detach from the "screen", press "ctrl-a" "d" ("d" for "detach").
#After this, you will notice that "[screen 0 ...]" dispears from the header, you
are back to regular session.
#The screen session is still alive in the background.
```

```
#To re-attach back to the screen session
screen -r

#You can create many independent sessions within one "screen" by pressing "ctrl-
a" "c" ("c" for "create").
#After the new session is created, "[screen 1 ...]" shows up at the header of the
terminal window.
#You can create as many independent sessions as you want, they will be named
"screen 0","screen 1", "screen 2", et al.

#You can switch between these sessions by pressing "ctrl-a" "n" ("n" for next).

#To kill a screen session
#press "ctrl-d" when you are inside screen session.

#To detach from the screen
#press "ctrl-a" "d" to detach from "screen".

#Please be aware of the difference bwtween "detach" and "kill". With "detach",
you can re-attach back to the session. With "kill", the session is permanently
removed, together with any jobs running in the session.
```

2. Short reads assembly with SPAdes

2.1 Trim adapters and low quality sequences 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
```

When you work with your own data, you need to adjust at least two of the parameters:

- "MINLEN:150" needs to be adjusted based on your original read length. The data used here is 250x2 bp. "MINLEN:150" would only keep reads >150bp after trimming. If your input data is in the most common format "2x150 bp", you can use "MINLEN:100".
- "ILLUMINACLIP:/programs/trimmomatic/adapters/TruSeq3-PE-2.fa:2:30:10". The file "TruSeq3-PE-2.fa" is a fasta file with adapter sequences. Your sequencing service provider should be able to tell you what adapters were used. In the directory "/programs/trimmomatic/adapters/", you can find other adapter sequences. If you are not able to find the adapter information, one option is to use other trimming tools like "bbduk" (https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/), which keep all adapter sequences in a single file.

Trimmomatic produces 4 new files: r1.fastq.gz, r2.fastq.gz, u1.fastq.gz, u2.fastq.gz. We will use the r1, r2 and u1 files for genome assembly.

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

2.2 Run SPAdes

[DO IT AT HOME] It could take an hour to finish this step. Skip it for now.

When you do this step at home, make sure to run it in "screen" session.

SPAdes automatically try multiple k-mer sizes based on the input reads length (up to k=127 for the installation on BioHPC computers). You can also define k-mer values by yourself using the "-k" option, e.g. -k 85. It would be useful if you need to assemble many genomes, a single fixed "-k" would make it a lot faster, but the quality might be slightly compromised.

```
#set OMP_NUM_THREADS
export OMP_NUM_THREADS=6

cd /workdir/$USER/project1

#run spades command
/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 greater or equal to the "-t" option (threads) in your SPAdes command. This is necessary as SPAdes uses OpenMP for parallelization.
- -t: Number of threads. When you are working on a real data analysis, you should increase the "-t" and OMP_NUM_THREADS settings, up to the total number of CPU of the machine;
- --careful: SPAdes would polish the assembly by correcting errors.

The output files are in the directory "spades_run".

2.3 Evaluate genome contiguity with QUAST

As you skipped step 2.2, we have pre-run SPAdes results for you to finish the rest of the workshop.

De-compress the "spades_results.tar.gz" file in your working directory.

```
cd /workdir/$USER/project1/
tar xvfz spades_results.tar.gz
cd spades_results
ls -l
```

A few files of interest in the directory "spades_results":

- 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
```

You would see that "SPAdes" actually assembled the genome with different k-mer sizes from K21 to K127, and give you the results with the best k-mer.

What is the estimated genome size?

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

You would find genome sizes estimated based on different k-mers. I would use the medium value to represent the genome size.

Now we will generate some statistics of the assembly using the software Quast. You 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 under the new directory "quast_results". 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.

2.4 Install BUSCO with Conda

BUSCO is the software to evaluate the genome completeness. You will install BUSCO version 4.1.4 using Conda.

Note (5/17/2017): Installing BUSCO with Conda could be very slow, and sometimes not working. You can skip this step and move one to 2.5

1. Install miniconda3 in your home directory if it has not been installed.

   ```
   cd /workdir/$USER
   wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
   chmod u+x Miniconda3-latest-Linux-x86_64.sh
   ./Miniconda3-latest-Linux-x86_64.sh
   ```

- The “chmod u+x” command makes the file executable.

- During the installation, you will be asked for multiple questions:

  1) “Please, press ENTER to continue”: press “ENTER” key;

  2) “More”: keep pressing “SPACE” key until you reach next question;

  3) “Do you accept the license terms?”: enter “yes”

  4) “Miniconda3 will now be installed into this location … ”: press “ENTER” key and accept the default “/home/xxxxx/miniconda3”.

  5) **“Do you wish the installer to initialize Miniconda3”** : Press ENTER to accept the default **“no”;**

2. Install BUSCO with Conda

```
source ~/miniconda3/bin/activate
conda create -y -n busco -c bioconda -c conda-forge busco=4.1.4
```

2.5 Evaluate genome completeness with BUSCO

When working with real data, BUSCO could take a long time to finish, it is a good idea to run BUSCO in "screen".

Detailed instructions how to run BUSCO can be found at https://busco.ezlab.org/busco_userguide.html#running-busco

```
#activate the busco environment only if you did not skip 2.4
source ~/miniconda3/bin/activate busco
```

Note (5/17/2017): If you skipped step 2.4. You can activate Busco pre-installed on BioHPC .

```
#activate the busco environment if you did not skip 2.4
source /programs/miniconda3/bin/activate busco-5.1.2
```

Retrieve available lineages and save to a file named "lineages_list". Examine the file content.

```
cd /workdir/$USER/spades_results

busco --list-datasets >lineages_list

less lineages_list
```

Press "q" to exit "less" command

Run busco command, set the lineage bacillales_odb10, and let BUSCO to determine the "augustus_species" for you.

```
#check the core gene name for the clade "bacillales"
grep bacillales lineages_list

busco -i scaffolds.fasta -l bacillales_odb10 -o busco_scaffold -m genome --cpu 6
```

The results are saved in the directory "busco_scaffold":

```
cd busco_scaffold

#check result file
cat short_summary.specific.bacillales_odb10.busco_scaffold.txt

#(Optional) The following command will generate a plot. it will create a image
file "busco_figure.png".
#You can use "Filezilla" to download the image file to your laptop to check.
generate_plot.py -wd ./
```

We would like to see the "C" score 90% or better for a good genome assembly. "C" score is the the percentage of BUSCO genes in full length.

The result shows that among the 450 core-genes in bacillales, 434 (96.4%) are in full length in this assembly, and only 15 (3.4%) genes are missing. This is a really good BUSCO sore.

After you are done with BUSCO, you would want to kill the "screen" session where BUSCO was running by pressing "ctr-d".

3. 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.

3.1 Run CANU

[DO IT AT HOME] It would take an hour to finish. Skip this step for now.

Run canu in "screen".

```
cd /workdir/$USER/project1

ls -l SRR10405213.fastq.gz

/programs/canu-2.1/bin/canu  useGrid=false maxThreads=8 maxMemory=24g -p ecoli -d
/workdir/$USER/project1/ecoli_run genomeSize=4.6m  -pacbio-raw
SRR10405213.fastq.gz >& canu.log &
```

- Run "/programs/canu-2.1/bin/canu -options" to see all parameters.
- With "-pacbio-raw" parameter, canu would do both read error correction and assembly.
- 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.

The assembled contigs will be in the directory ecoli_run.

3.2 Polish the genome assembly.

We have pre-run the software canu. The results are in the file canu_results.tar.gz.

```
cd /workdir/$USER/project1/
tar xvfz canu_results.tar.gz
cd canu_results
ls -l
```

You would see up to three 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, e.g. allelic haplotypes are represented as two sequences.

*.unassembled.fasta: A fasta file of contigs with poor support and not included in *.contigs.fasta.

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.

Here you will run Racon to polish the genome. First, you will run minimap2 to align the raw reads to the assembled genome:

```
cd /workdir/$USER/project1/

/programs/minimap2-2.17/minimap2 -a canu_results/ecoli.contigs.fasta
SRR10405213.fastq.gz > ecoli.sam
```

Now run Racon ("-t 4": run on 4 threads):

```
/programs/racon-1.4.13/bin/racon -t 4 SRR10405213.fastq.gz ecoli.sam
canu_results/ecoli.contigs.fasta > racon_corrected.fasta &
```

The corrected fasta file is written into a new file "racon_corrected.fasta".