Intro to Next Generation Sequencing preprocessing

McGill Initiative in Computational Medicine, Summer 2022

For these exercises we will be using an dataset sequenced with Illumina MiSeq technology. The organism in question is an enterohaemorrhagic E. coli (EHEC) of the serotype O157, a potentially fatal gastrointestinal pathogen. The sequenced bacterium was part of an outbreak investigation in the St. Louis area, USA in 2011.

Set Up

Let's create the directories to organize our data.

```
## This will be today's working directory
mkdir ngsintro_exercises && cd ngsintro_exercises

## Create directories to organize the data
mkdir Fastq QC Trimmed Mapped Reference
```

Part 1 - Quality control

1. Downloading sequencing data

There are several ways of downloading sequencing data from different databases.

The Ecoli raw data that we will use was deposited at the European Nucleotide Archive, under the accession number SRR957824. You could go to the ENA website and search for the run with the accession SRR957824 and download the dataset directly or with the url link for example.

```
## Do not run!
wget
ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR957/SRR957824/SRR957824_2.fastq.gz
```

But this are large datasets, containing around 3 million reads. They will take great time to download and to work with. So instead we will be using a subset of the data:

Download the data from this link

Or download directly from the terminal in the Fastq directory.

```
cd Fastq
curl -O -J -L https://osf.io/shqpv/download
curl -O -J -L https://osf.io/9m3ch/download
```

2. Exploring the fastq files

We want to avoid modifying the files by mistake so lets make them unwritable

```
cd Fastq
chmod u-w *.fastq*
```

Judging from the file names, what type of reads do we have?

Let's see a bit of the format

```
zmore *_R1.fastq.gz | head
```

Do you think this was short-read or long read?

Let's count how many reads we have per file

```
zmore *_R1.fastq.gz | grep -E "^@SR" | wc -l
zmore *_R2.fastq.gz | grep -E "^@SR" | wc -l
```

There are 500K reads on each file. Does that mean that we have 1 Million independent reads?

3. Assessing Quality Control with FastQC

We will use FastQC to assess the quality of our reads. FastQC will produce an html report for each of our files.

To explore FastQC options you can run

```
fastqc --help
```

Run fastQC in each of our fastq files

```
# To go back to our main wd
cd ..

fastqc --outdir QC/ Fastq/SRR957824_500K_R1.fastq.gz
fastqc --outdir QC/ Fastq/SRR957824_500K_R2.fastq.gz
```

Alternatively, you can also tell fastqc to take all of the fastq files as input

```
fastqc --outdir QC/ Fastq/*.fastq*
```

Let's inspect the QC reports generated by fastQC.

What do you see? Is there any property that could cause bias in the downstream analyses?

Part 2 - Improving the Quality of Reads

4. Trim and filter with CutAdapt

After inspecting and interpreting our quality plots, we know that there is still some adapters in our reads.

We will use cutadapt to trim and filter our reads.

You can inspect the options by running --help.

```
cutadapt --help
```

We want to remove the illumina adapters and trim low-quality bases from 5' ends of each paired read before adapter removal. So let's give those options to cutadapt.

> It is a convention to add to the names whatever process/filtering was done. In this case we will add *_trimmed* to indicate that we have removed adapters and trimmed ends off our fastq files.

```
cutadapt -q 25 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT --minimum-length 50 -o
Trimmed/SRR957824_R2_trimmed.fastq.gz -p
Trimmed/SRR957824_R1_trimmed.fastq.gz Fastq/SRR957824_500K_R1.fastq.gz
Fastq/SRR957824_500K_R2.fastq.gz > cutadapt.log
```

Why do you think we need to run both paired-end files together?

Lets run fastQC again

```
fastqc --outdir QC/ Trimmed/*trimmed.fastq*
```

5. Combining Reports in one with MultiQC

Imagine you have a lot of fastq files. Going through each of the fastqc reports individually before AND after filtering is time consuming and difficult to track.

MultiQC is a tool that can recopilate reports from different bioinformatics tools and from multiple files into a single report.

```
multiqc QC --outdir QC
```

Part 3 - Mapping to a Reference Genome

6. Getting the Reference Genome

We are going to map our reads to the E. coli O157:H7 str. Sakai Genome with accession number ASM886v2 in NCBI.

You can download the fasta from the previous link or directly using wget.

```
cd Reference
wget
https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/008/865/GCF_000008865.2_A
SM886v2/GCF_000008865.2_ASM886v2_genomic.fna.gz
```

Before aligning the reads against a reference, it is necessary to build an index of that reference.

Indexing the reference is a necessary pre-processing step that makes searching for patterns much much faster. Many popular aligners such as Bowtie and BWA use an algorithm called the Burrows–Wheeler transform to build the index.

Let's Index the reference genome.

```
bowtie2-build GCF_000008865.2_ASM886v2_genomic.fna.gz GCF_000008865
```

7. Mapping reads to the Reference using Bowtie2

Bowtie2 is a splice-unaware aligner used to map sequencing reads to long reference genomes.

```
cd ..
bowtie2 --help
```

```
bowtie2 -x Reference/GCF_000008865 -1
Trimmed/SRR957824_R1_trimmed.fastq.gz  -2
Trimmed/SRR957824_R2_trimmed.fastq.gz -S Mapped/SRR957824.sam
```

▶ Bowtie2 Output

```
  414756 reads; of these:
    414756 (100.00%) were paired; of these:
      46753 (11.27%) aligned concordantly 0 times
      340072 (81.99%) aligned concordantly exactly 1 time
      27931 (6.73%) aligned concordantly >1 times
      ----
```

```
        46753 pairs aligned concordantly 0 times; of these:
          36607 (78.30%) aligned discordantly 1 time
        ----
        10146 pairs aligned 0 times concordantly or discordantly; of these:
          20292 mates make up the pairs; of these:
            12707 (62.62%) aligned 0 times
            1141 (5.62%) aligned exactly 1 time
            6444 (31.76%) aligned >1 times
      98.47% overall alignment rate
```

Let's inspect our Sam file output!

```
cd Mapped
head SRR957824.sam
```

Can you identify the CIGAR string of the first mapped read?

7. Using Samtools: Sam to Bam, sorting and indexing

Doing a head/tail is not very informative, but we can use samtools to manage our sam/bam files.

Samtools is a software package for parsing and manipulating alignments in the SAM/BAM format.

The first step is to compress the sam file to a bam file.

> Generally, try not to keep alignments in sam format as they can get very *very* large and will consume a lot of precious storage space!

To convert a sam to a bam we use samtools view.

```
samtools view -hbo SRR957824.bam SRR957824.sam
```

8. Marking Duplicates

Duplicates are sets of reads pairs that have the same unclipped alignment start and unclipped alignment end. They're suspected to be non-independent measurements of a sequence.

We deal with duplicated reads by marking them using samtools or picard.

```
samtools fixmate -m SRR957824.bam SRR957824.fixmate.bam
```

We can now sort by position our bam file and mark the duplicated reads. Removing the duplicated reads is optional and can be performed by adding the *-r* option to samtools markdup.

```
samtools sort -o SRR957824.fixmate.sorted.bam SRR957824.fixmate.bam
samtools markdup -r SRR957824.fixmate.sorted.bam SRR957824.markdup.bam
```

To finish, we need to index our bam file.

Indexing a genome sorted BAM file allows one to quickly extract alignments overlapping particular genomic regions and is required for most visualizers like IGV.

```
samtools index SRR957824.markdup.bam
```

9. Alignment metrics

Since the BAM file contains records for both mapped and unmapped reads, just counting records doesn't provide information about the mapping rate of our alignment. The samtools flagstat tool provides a simple analysis of mapping rate based on the the SAM flag fields.

Let's see out alingment stats:

```
samtools flagstat SRR957824.markdup.bam
```

> Exercise: run samtools flagstat in the file we had before removing the duplicates.

▶ Flagstat Output

```
    - Before Marking and removing duplicates -
    828899 + 0 in total (QC-passed reads + QC-failed reads)
    828899 + 0 primary
    0 + 0 secondary
    0 + 0 supplementary
    0 + 0 duplicates
    0 + 0 primary duplicates
    816192 + 0 mapped (98.47% : N/A)
    816192 + 0 primary mapped (98.47% : N/A)
    828899 + 0 paired in sequencing
    414448 + 0 read1
    414451 + 0 read2
    735544 + 0 properly paired (88.74% : N/A)
    815048 + 0 with itself and mate mapped
    1144 + 0 singletons (0.14% : N/A)
    114 + 0 with mate mapped to a different chr
    14 + 0 with mate mapped to a different chr (mapQ>=5)
```

Can you infer how many duplicates we had in out data by comparing the stats?

Samtools idxstats report shows how many reads aligned to each contig in your reference.

```
samtools idxstats SRR957824.markdup.bam
```

▶ IdxStats Output

```
    NC_002695.2     5498578 799173  1190
    NC_002127.1     3306    0       0
    NC_002128.1     92721   17632   63
            *       0       0       11454
```

The output has four tab-delimited columns:

1. contig name
2. contig length
3. number of mapped reads
4. number of unmapped reads

If you're mapping to a non-genomic reference such as miRBase miRNAs or another set of genes (a transcriptome, samtools idxstats gives you a quick look at quantitative alignment results.

10. More filters with Samtools

Let's say you are only interested in a certain region, or need only reads with a certain flag. Samtools allows you to filter bam/sam files.

Extract only reads that mapped to the region from 18000bp-18050bp

```
samtools view SRR957824.markdup.bam "NC_002695.2:18000-18050"
```

Extract only the unmapped reads.

```
samtools view -f 4 SRR957824.markdup.bam
```

Samtools is a very extensive software package! Just in samtools you can:

- Filter
- Get mapping stats
- Call variants
- Split bam
- and much more!

I invite you to see the documentation to know more about the tools available.

Markdown and exercises prepared by Georgette Femerling.