Next Generation Sequencing Notes

- Each NGS platform are happening simultaneously for distinct fragements, and they are massively parallel.
  - PCR amplification is used to turn weak bioluminescent signals into the strong signal of a cluster of 1000 identical fragments.
  - Sequencing by synthesis is composed of 4 steps: sample prep, cluster generation, sequencing, and data analysis[1,2].
    - ∗ Sample prep: Adaptors are added to the ends of the DNA. Through reduced cycle amplification, additional motifs are introduced, such as sequencing binding site, indices and regions complementary to the flow cell oligos.
    - ∗ Cluster generation: Clustering is a process where each fragment molecule is isothermally amplified. The flow cell is a glass slide with lanes, each a channel coated with a lawn composed of two types of oligos. Hybridization is enabled by the first of the two types of oligos on the surface. This oligo is complimentary to the adapter region on one of the fragment strands. A polymerase creates a complement of the hybridized fragment. The double-stranded molecule is denatured and the original sample is washed away. The strands are clonally amplified through bridge amplification. In this process, the strand folds over and the adapter region hybridizes to the second type of oligo on the flow cell. Polymerases generate the complimentary strand forming a double stranded bridge. This bridge is denatured, resulting in 2 single stranded copies of the molecule that are tethered to the flow cell. This process is then repeated many times and occurs simultaneously for millions of clusters resulting in clonal amplification of all the fragments, followed by cleaving and washing off the reverse strands. The 3' ends are blocked to prevent unwanted priming.
    - ∗ Sequencing: Sequencing begins with the extension of the first sequencing primer to produce the first read. With each cycle, fluorescently tagged nucleotides compete for addition to the growing chain. Only one is incorporated based on the sequence of the template. After the addition of each nucleotide, clusters are excited by a light source and a light source, and the characteristic fluorescent

[1] https://www.youtube.com/watch?v=fCd6B5HRaZ8&t=10s.

[2] https://www.well.ox.ac.uk/bioinformatics/training/MSc_GM_2022/tutorials/CM4-5_bioinformatics_pipelines_for_NGS_data/introduction_to_next_generation_sequencing_data_analysis/Short_read_theory/

signal is emitted. The number of the cycles determines the length of the read. The emission wavelength, along with signal intensity, determines the base call. For a given cluster, all identical strands are read simultaneously. Hundreds of millions of clusters are sequenced in a massively parallel process. After the completion of the first read, the read product is washed away. In this step, the index 1 read primer is introduced and hybridized to the template. The read is generated, similar to the first read. After completion of the index read, the read product is washed off, and the 3' ends of the template are deprotected, which then bends over and attaches to the second oligo on the flow cell. Index 2 is read in the same manner as index 1. Polymerases extend the second flow cell oligo forming a double-stranded bridge, which is then linearized and the 3' ends are blocked. The original forward strand is cleaved off and washed away, leaving only the reverse strand. Read 2 begins with the introduction of the read 2 sequencing primer. As with read 1, the sequencing steps are repeated until the desired read length is achieved. The read 2 product is then washed away.

  * Data analysis: This process generates millions of read, representing all the fragments. Sequences from pooled sample libraries are separated based on the unique indices introduced during the sample preparation. For each sample, reads with similar stretches of base calls are locally clustered. Forward and reverse reads are paired, creating contiguous sequences, which are then aligned back to the reference genome for variant identification. The paired end information is used to resolve ambiguous alignments.

- Base calling via bioluminescence has several pitfalls:
  - Signals from clusters in close proximity interfere with on another.
  - Synchronicity between strands within a cluster is gradually lost with each cycle.
  - The intensity of a signal naturally varies.
  - A signal can be ambiguous where bases repeat.
- Base call accuracy for NGS platforms can be increased by increasing read depth, that is, aligning more strands together.
- A `fastq` file contains millions of read, each written in a row where:
  - The first line is a header line that identifies the read.

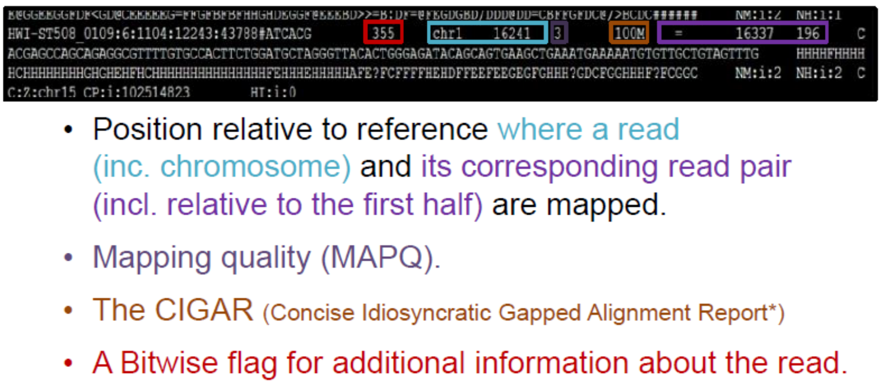
  - The second line is the base pair read itself.
  - The third line is a `+`.
  - The fourth line contains base qualities in PHRED encoding, denoted by Q.
    * Let P be the estimated probability that each base call is incorrect, then

$$Q = -10\log_1 P$$

      which is subsequently converted to ASCII+ (current standard is PHRED+33)[3]. PHRED scores are also used to quantify mapping uncertainty, which is applied to a single read rather than individual bases.

- Fast aligners largely fall into two categories based on the underlying data structure used to store and compare reads and reference:
  - Hash table is used by `Novoalign`, `BLAST`, and `MAQ`. It stores the reference genome as subsets of size k (k-mers) in a hash table. Subsets of reads are matched against the reference.
  - Suffix tries/arrays (FM-index) are structures based on storing all possible suffixes of a sequence. Suffix tries are used by `BWA` and `Bowtie2`.

Most popular fast-aligners take `FASTQ` as input and output `SAM/BAM` files. They adds alignment information to `FASTQ` read data (i.e. position relative to reference, mapping quality, presence of insertions/deletions, etc.)[4]. They contain:



```
B@GGEEGGFDF<GD@CEEEEEG=FFGFBFBFHHGHDEGGF@EEEBD>>=B:DF=@FEGDGBD/DDD@DD=CBFFGFDC@/>BCDC######     NM:i:2  NH:i:1
HWI-ST508_0109:6:1104:12243:43788#ATCACG      355    chr1    16241   3      100M   =       16337   196      C
ACGAGCCAGCAGAGGCGTTTTGTGCCACTTCTGGATGCTAGGGTTACACTGGGAGATACAGCAGTGAAGCTGAAATGAAAAATGTGTTGCTGTAGTTTG       HHHHFHHHH
HCHHHHHHHHGHGHEHFHCHHHHHHHHHHHHHHHHFEHHHEHHHHHAFE?FCFFFFHEHDFFEEFEEGEGFGHHH?GDCFGGHHHF?FCGGC      NM:i:2  NH:i:2  C
C:Z:chr15 CP:i:102514823         HI:i:0
```

- Position relative to reference where a read (inc. chromosome) and its corresponding read pair (incl. relative to the first half) are mapped.
- Mapping quality (MAPQ).
- The CIGAR (Concise Idiosyncratic Gapped Alignment Report*)
- A Bitwise flag for additional information about the read.

Figure 1 SAM/BAM file information

[3]0 is encoded by ASCII symbol 33 or `!`

[4]More about CIGAR scores: https://wiki.bits.vib.be/index.php/CIGAR

Nevertheless, not all reads match a region of the reference. There can be SNVs and indels (insertions and deletions).

- Once SAM files have been generated, a number of steps can be reduce file size and prepare the file for variant calling:
  - SAM files can be compressed to BAM files (no longer human readable).
  - BAM files can then be sorted and indexed to further reduce size and speed up subsequent variant calling.
  - Duplicates are removed.
  - If using GATK, one final step is needed: Base Quality Score Recalibration (BQSR).
- Variant caller needs to account for bas ad mapping quality, read depth, and so on.
- The variant call format (VCF) is designed to be human-readable.

Starting from this page, we describe a general procedure of analysing the data.

Step 1: Have a look at the FASTQ files:

- Decompress the file and look at the header:

```
zcat malaria/QG0033-C_Illumina-HiSeq_read1.fastq.gz | head -n 1
zcat malaria/QG0033-C_Illumina-HiSeq_read2.fastq.gz | head -n 1
```

  This couple of files differ only at the direction of the reads. A bit more information about the header:

  * The read header:

```
@ERR377582.7615542 HS23_10792:2:2307:6524:31920#1
```

    This tells us the sample ID (`ERR377582`), the read identifier (`7615542`), information about the instrument that generated the reads (`HS23_10792`), the flowcell lane and tile number (`2:2307`), the coordinates of the cluster within the tile (`6524`, `31920`), a number identifying the index of the sample within a multiplexed set of samples (`#15`), and its read (`/1` or `/2`)[5].

- Count the number of reads:

```
zcat malaria/QG0033-C_Illumina-HiSeq_read1.fastq.gz | wc -l
```

- Count the read length of a single read:

```
zcat malaria/QG0033-C_Illumina-HiSeq_read1.fastq.gz | head -n 2
    | tail -n 1 | wc -c
```

Step 2: Perform QC on the sequence reads. This step includes looking for poor quality data, looking for read duplication, assessing GC content, looking for over-represented sequences and the presence of sequence adapters in reads.

- First make a directory for the output and run `fastqc`:

```
mkdir fastqc_output
fastqc -o fastqc_output malaria/*.fastq.gz
```

[5]The format of this information is not standard across platforms, and it changes depending on your data provider.

It creates some `.html` and `.zip` files. The `.html` files can be viewed in a web browser.

- Interpreting `fastqc` output:
  * It is common to see high levels of duplication. PCR-based library preparation typically needs to still higher duplication rates.
  * Over-representation can suggest contamination or chemistry issues. Adapter sequences can appear in the reads if the fragments are too short so that the sequencer ’reads through’ into the adapters[6].

Step 3: Aligning reads:

- A basic pipeline is:
  * Align reads to the reference genome assembly.
  * Sort them so they are in genome position order.
  * ’Mark’ reads that look like duplicates (since these are likely to be artifacts.).
  * Gather some statistics about the alignments. For some dataset, we might need an extra step of read trimming at first. Read trimming refers to the process of removing trailing parts of the reads before analysis for various reasons including:
    · The presence of adapter sequences in the reads (trim everything after the adapter).
    · The presence of low-quality bases toward the end of the read (trim everything after the low-quality bases).
    · The presence of other contaminant or artifactual sequences (for example poly-G sequences).
- Now, move to the correct folder and run the pipeline:

```
cd ~/sequence_data_analysis/malaria

# Make a temp dir to hold intermediate files
mkdir -p tmp

```

[6]https://www.well.ox.ac.uk/bioinformatics/training/MSc_GM_2022/tutorials/CM4-5_bioinformatics_pipelines_for_NGS_data/introduction_to_next_generation_sequencing_data_analysis/Short_read_theory/.

```
# Create a FM index for the reference
bwa index Pf3D7_v3.fa

# align the reads (using 2 threads - don't use more please!)
bwa mem -t 2 -o tmp/QG0033-C-aligned.sam Pf3D7_v3.fa QG0033-
    C_Illumina-HiSeq_read1.fastq.gz QG0033-C_Illumina-
    HiSeq_read2.fastq.gz

# convert to BAM
samtools view -b -o tmp/QG0033-C-aligned.bam tmp/QG0033-C-
    aligned.sam

# fix mate-pair information and convert to BAM
samtools fixmate -m tmp/QG0033-C-aligned.bam tmp/QG0033-C-
    fixmate.bam

# sort reads by genomic position
samtools sort -T tmp -o tmp/QG0033-C-sorted.bam tmp/QG0033-C-
    fixmate.bam

# Identify probable duplicate reads
samtools markdup -s tmp/QG0033-C-sorted.bam tmp/QG0033-C-
    markdup.bam

# Rename to this folder, and index
mv tmp/QG0033-C-markdup.bam QG0033-C.bam
samtools index QG0033-C.bam
```

These commands generate a bunch of files that we are to have a look below.

– After some metadata, an output line from the SAM file is like this:

```
ERR377582.7615542 99 Pf3D7_09_v3 37181 21 100M = 37244 163
    AAAAATCCA... B@DECEFEE... NM:i:1 MD:Z:58T41 MC:Z:100M AS:i
    :95 XS:i:91
```

The alignment columns rows consist of the read ID (`ERR377582.7615542`), flag[7] (`99`), reference genome (`Pf3D7_09_v3`), position (`37181`), mapping quality (`21`), CIGAR string[8] (`100M`), the chromosome and position of the other read in the pair if it's aligned (`=` , ` ` if the other read is not aligned), template length (`37244`), etc[9].

- Count the alignments:

```
samtools view tmp/QG0033-C-aligned.sam | wc -l
```

  Note that this can also be done in the BAM files. It turns out there are 4 million alignments (but only 2 million reads). This is because the presence of supplementary alignments.

```
grep 'ERR377582.20226793' tmp/QG0033-C-aligned.sam
```

  The second alignment (`flag=2113`) represents the first read in the pair and is flagged as a supplementary alignment, which occurs when a read aligns in more than one part. Together with the first alignment, we observe zero mapping quality, meaning that confidence in these alignments is very low. What's really going on is that these reads originate from the telomeres. Telomeres in malaria, as in humans and other organisms, are highly repetitive and are very hard to analyse using short-read data.

- Count reads according to the `flags` column:

```
samtools flagstat tmp/QG0033-C-aligned.sam
```

  A much more detailed view of the reads can be generated by running `samtools stats` [10].

- About the command `samtools markdup` , we can run the following commands to see the difference before and after that function:

```
samtools view tmp/QG0033-C-sorted.bam | grep 'ERR377582
    .19356815'
samtools view QG0033-C.bam | grep 'ERR377582.19356815'
```

[7]For a complete table of flags, see https://broadinstitute.github.io/picard/explain-flags.html.

[8]Note that for a CIGAR score a mutation is also considered aligned. For example, if AAAATCCCC is aligned to AAAAGCCCC, it is also considered as a 'matched.'

[9]For details, see https://en.wikipedia.org/wiki/SAM_(file_format).

[10]See more information at http://www.htslib.org/doc/samtools-stats.html.

We observe the `flagstat` reduces by `1024`. Hence, `samtools markdup` identifies 5000 or so duplicates.

Step 4: Inspect read pileups and looking for variation. So far, we have a file called `QG0033-C.bam` containing aligned, duplicate-marked reads.

- We can use `samtools tview` to view alignments.:

```
samtools tview -p Pf3D7_07_v3:403818 QG0033-C.bam
```

  We can observe that the reads as they align to the reference sequence (upper-case for forward strand and lower-case for reverse strand).

  * Use arrow keys to move around (hold `shift` to move faster).
  * Use `g` to jump to another genome location.
  * Use `m` to color reads by mapping quality.
  * Use `b` to color reads by base quality.
  * Use `n` to color reads by nucleotide.
  * Use `q` to quit the program.

  However, this is not very helpful. Instead, run

```
samtools tview -p Pf3D7_07_v3:403818 QG0033-C.bam Pf3D7_v3.fa
```

  We can see punctures meaning 'reads with the same base as the reference' (dots for forward strand and commas for reverse strand). Anything else means a mismatching base.

Extra: Some extra steps and useful commands are summarised here:

- Pull out all unaligned reads:

```
samtools view -f 4 -o QG0033-C-unaligned.bam QG0033-C.bam
```

  Or

```
samtools view -F 4 -o QG0033-C-onlyaligned.bam QG0033-C.bam
```

  where `-f` and `-F` mean exclude or include reads with these flags.

- To convert BAM to SAM:

```
samtools view -h QG0033-C-unaligned.bam > QG0033-C-unaligned.
    sam
```

where the `-h` command is to include header in the SAM output.

- To count this file, one have to use `samtools`:

```
samtools view QG0033-C-unaligned.sam | wc -l
```

  Simply using `wc -l` won't give the correct answer (the command won't work if the header is not included in the SAM file).

- Note that we observe some unaligned reads contaminating the GC-content. We can use `seqtk` to convert that unaligned reads file to a FASTA file and then search on `BLAST`[11]:

```
samtools bam2fq input.bam | seqtk seq -A > output.fa
```

  Upload the `output.fa` to `BLAST` and it should work.

Appendix: *k*-mer counting refers to counting the number of occurrences of short *k*-mers in the reads and use that to figure out properties of the sequencing, which is usually quite informative, giving us:

- An estimate of sequencing error rate.
- Information about genome size.
- Information about the heterozygosity and the amount of repetitive sequence.

We use `jellyfish` to count *k*-mers:

```
jellyfish count -C -m 31 -s 10M -o NA12878_Illumina-Novaseq_6000.jf
     <(zcat NA12878_ERR3239334-Illumina_Novaseq_6000-chr19_region-
    read2.fastq.gz) <(zcat NA12878_ERR3239334-Illumina_Novaseq_6000
    -chr19_region-read1.fastq.gz)
```

Since `jellyfish` does not have in-built function of decompressing, we use `zcat` and `<()` here. Other commands involve:

- `-m 31` counts 31-bp *k*-mers.

[11]https://blast.ncbi.nlm.nih.gov/Blast.cgi.

- `-C` counts *k*-mers and their reverse complements together.
- `-s 10M` tells `jellyfish` how big a hash table to use. Note that for an analysis of a full genome, we need a much bigger (say 10 billion/ 10G) hash table. This is to allow for the roughly 3 billion true *k*-mers, plus some large number of false (error) *k*-mers in the reads.

Now, we use `R` to analyse the new `.jf` file:

```
# In an R session
X = read.table( "NA12878_Illumina-Novaseq_6000.jf.histogram",
    header = FALSE, as.is = TRUE )
colnames(X) = c( "observed_count", "number_of_kmers" )
plot( X$observed_count, X$number_of_kmers, type = 'l' )
```

We see a bump at $x = 30$ as the data is about 30-fold coverage of the human genome. *k*-mers that arise due to sequencing errors tend to only be seen once or a handful of times, thus lying near $x = 0$.

To work out the error rate, we assume coverage $<=5$ implies an error:

```
number_of_error_kmers = sum( X[1:5,1] * X[1:5,2] )
total_kmers = sum( X[,1] * X[,2] )
kmer_error_rate = number_of_error_kmers / total_kmers

cat( sprintf( "The estimated 31bp Kmer error rate is: %.2f%%!\n",
    kmer_error_rate * 100 ))
```

It turns out to be 8%, which is huge. We can also have a look at the most repeated sequence using `tail(X)` in `R` and find it by `jellyfish`:

```
jellyfish dump -L 4420 NA12878_Illumina-Novaseq_6000.jf
```

The result can be searched in `Dfam`[12]: it's an Alu element, a type of short interspersed nuclear element (SINE).

- An Alu element is a short stretch of DNA whose sole known function is self reproduction. However, they are likely to play a role in evolution and have been used as genetic markers.

[12]https://dfam.org/home.

– Short interspersed nuclear elements (SINEs) are non-autonomous, non-coding transposable elements. They are a class of retrotransposons, DNA elements that amplify themselves throughout eukaryotic genomes, often through RNA intermediates.