Question 1

The FASTQ file that was given in the first practice which contains reads from a simulated whole-genome sequencing experiment, the Negative split generated per barcode (using cutadapt), returned very poor mapping results. Here, there is an attempt to remap that FASTQ in order to observe an improvement in the mapping statistics.

Note: All the following commands are executed either in the departmental Crystallography Hope Server using a remote SSH connection or in a local environment using ArchLinux and the respective software installed either using pacman or the respective source files.

An exploratory view of the file using less shows that:

```
thoth.cryst.bbk.ac.uk> head trimmed_Negative.fq
@AFPN02.1_merge-1076320
NNNNNAAGGCTCAATTCTTTACGCCGAGATTTTTATTCTCTCAACTAAACAGCGNNNN
+
#####BACCCGFGGGGGGGGGGGGGGGGGGGG<GGGGGGGGGGGGGGGGGGGGGG###

@AFPN02.1_merge-1076319
NNNNNCTTTATCCGGAAGCGTTTGTGTAAGCTGCTGCGATTAAACCGGTTGACCCAGNNNN
+
#####EBCCB=G>GGGGGG>GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG##

@AFPN02.1_merge-1076318
NNNNNCTTTGCGGATTTCGCTCGGCGCGGCAGCGGCGGTGATCATATTCGACACCCNNNN
+
#####EBCCB=G>GGGGGG>GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG##
```

It can be easily observed that there are poor quality bases (N) on the first 5 positions (at the 5’-end) and on the last 4 positions (at the 3’-end). Using the generated FASTQC Quality Report in Fig that can be seen quite clearly. Apart from this the reads seem to be of a very good quality.

To obtain a better alignment than the one in the Practical, a global alignment with the reference genome was performed using bowtie2 while trimming the respective N-bases from the 5’-end and 3’-end.
Note: For all the following code, the binary locations are added on the path for the session with the command:

```
PATH=$PATH:/s/software/anaconda/python3/bin/:/s/software/samtools/v1.9/bin/
```

```
bowtie2 -p 127 -5 5 -3 4 --end-to-end -x ${st_path}/course_materials/genomes/AFPN02.1/AFPN02.1_merge -q \\
${st_path}/course_materials/fastq/trimmed_Negative.fq -S \\
Negative.sam 2> Negative_bowtie_stats.txt
```

This obtained a 99.97% overall alignment rate! That is the same percentage as in the trimmed Positive sample from the Practical.

```
thoth.cryst.bbk.ac.uk> more Negative_bowtie_stats.txt
1076320 reads; of these:
1076320 (100.00%) were unpaired; of these:
308 (0.03%) aligned 0 times
1020259 (94.79%) aligned exactly 1 time
55753 (5.18%) aligned >1 times
99.97% overall alignment rate
```

The respective FASTQ graph shows as well the alignment coverage in Fig 2

Looking for information from different sources such as (bowtie2: Relaxed Parameters for Generous Alignments to Metagenomes, n.d.) and (Viljetic et al., 2017) and exploring different options using the manual of the software, the parameters for bowtie2 that gave the best result were the following:
thoth.cryst.bbk.ac.uk> bowtie2 -a -p 8 --np 0 \ 
--n-ceil L,0,0.2 --non-deterministic -x \ 
${st_path}/course_materials/genomes/AFPN02.1/AFPN02.1_merge -q \ 
${st_path}/course_materials/fastq/trimmed_Negative.fq -S \ 
Negative_with_L.sam 2> Negative_bowtie_stats_with_L.txt

thoth.cryst.bbk.ac.uk> cat !$
1076320 reads; of these:
  1076320 (100.00%) were unpaired; of these:
    5060 (0.47%) aligned 0 times
    1018111 (94.59%) aligned exactly 1 time
    53149 (4.94%) aligned >1 times
99.53% overall alignment rate

The meaning of each is as following:

- **non-deterministic**: Re-initialisation of pseudo-random generator for each read
- **-a**: The program searches for the most distinct, valid alignments for each read
- **-p 8**: Multi-threading run of the `bowtie2` alignment
- **--np 0**: Penalty for having an N in either the read or the reference
- **--n-ceil L,0,0.2**: It sets an upper limit on the number of positions that may contain ambiguous reference characters in a valid alignment. It sets the \( N(x) = 0 + 0.15 \times x \) as a default option. In our case a value of 0.2 multiplied by the number of reads was slightly larger than the N reads and definitely enough to ignore the ambiguous reads.
It is noticeable that the above code needs only 6 seconds to run in contrast with just a plain -L parameter which takes x6 times more! Other options like local and very sensitive alignment were explored but unfortunately without any success (they only returned 0% of matches).

To conclude, two different ways of using the bowtie2 (Langmead & Salzberg, 2012) were attempted with great success, in the first case similar with the Positive Trimmed sequence on the Practical (99.97% and 99.53% respectively). Nevertheless, the best practice is to trim before the alignment using software like trimmomatic or cutadapt. For Whole Genome Sequencing (WGS) or larger number of exomes, having the full sequencing information in the BAM, allows to get rid of the FASTQ which essentially cuts storage costs by one half. Thus one can have major budget implications. Additionally, read mappers like Burrows-Wheeler Aligner (BWA) will just soft clip the adapter(s).

In addition to detecting and handling adapters sequence contamination it is important that the QC information and results are fed back into the wet lab to ensure higher quality in the future.

In the Fig 3, Fig 4 and Fig 5 the samtools (Li et al., 2009) and multiqc (Ewels, Magnusson, Lundin, & Käller, 2016) reports can be seen. Those were generated as following, for both Negative.sam and Negative_with_L.sam

```
samtools sort Negative.sam > Negative.bam
samtools index Negative.bam
samtools stats Negative.bam > Negative_stats.txt
samtools flagstat Negative.bam > Negative_flagstat.txt
```

![Figure 3: Bowtie2 alignment scores](image)

---

4
As previously noted, there is an important difference between the (pre-)trimmed sequences: the error rate is only 0.11% while on the L (--n-ceiling) parameter there’s a much higher error rate observed: 15.34%.

There is also the Integrative Genomics Viewer plot of the two different BAM files shown in Fig 6

Question 2

This refers to the BQ fq alignment. The aim here is to achieve higher alignment rates as well as minimal alignment error rate.

In Fig 7 it’s shown the Quality Report for BQ. It can be easily seen that the scores are really low quality. Similarly with the previous sequence there are also five ambiguous bases on the 5’-prime and four on the 3’-prime. That may be because of different reasons like the sample quality or the starting material, e.tc.

As in the practical the adapters were removed as well.
cutadapt -g Positive=^GATACA -g \ 
Negative=^AGTAGT -g BQ=^CACACA -g \ 
Long=^AAACCC -o trimmed_'{name}'.fq \ 
final_merge_syntetic_reads.fq

cutadapt (Verma et al., 2018) is used to trim the ambiguous bases (N) from the 5’ and 3’ ends and to filter the reads according to sequence quality using the parameter --trim-n, which trims N’s on ends of reads.

cutadapt -g BQ=^CACACA --trim-n -j 127 \ 
--quality-cutoff 15,8 -o BBQ.fq final_merge_syntetic_reads.fq

The parameter -q is used by default, only the 3’ end of each read is quality-trimmed. To trim the 5’ end as well, the option is used with two comma-
separated cutoffs, as shown above. This parameter is used to trim low-quality ends from reads before adapter removal.

Different values were tried but this was found to have to most succesfull alignment.

The output summary of this is:

This is cutadapt 1.18 with Python 3.7.1
Command line parameters: -g BQ=^CACACA --trim-n -j 127 -q 15,8 -o BBQ.fq final_merge_synthetic_reads.fq
Processing reads on 127 cores in single-end mode ...
Finished in 2.82 s (1 us/read; 76.44 M reads/minute).

=== Summary ===
Total reads processed: 3,597,656
Reads with adapters: 1,086,246 (30.2%)
Reads written (passing filters): 3,597,656 (100.0%)

Total basepairs processed: 256,808,830 bp
Quality-trimmed: 8,690,673 bp (3.4%)
Total written (filtered): 236,169,451 bp (92.0%)

=== Adapter BQ ===
Sequence: CACACA; Type: anchored 5'; Length: 6; Trimmed: 1086246 times.
No. of allowed errors:
0-6 bp: 0

Overview of removed sequences
length count expect max.err error counts
6 1086246 878.3 0 1086246

Bringing this back into bowtie2 for alignment we get a 94.51% alignment rate!

thoth.cryst.bbk.ac.uk> cat BBQ_bowtie_stats.txt
3597656 reads; of these:
3597656 (100.00%) were unpaired; of these:
197548 (5.49%) aligned 0 times
3252609 (90.41%) aligned exactly 1 time
147499 (4.10%) aligned >1 times
94.51% overall alignment rate
Finally, the samtools and flagstat analysis are shown in Fig 8 and Fig 9.

As described in (Salari, Zare-Mirakabad, Sadeghi, & Rokni-Zadeh, 2018), to decrease the error rate “(we) reduce the search space for the reads which can be aligned against the genome with mismatches, insertions or deletions to decrease the probability of incorrect read mapping”. This involves a quite intrinsic pipeline workflow through which isn’t possible to achieve in this coursework. Despite that many different values were tested against the quality algorithm of the software in order to find the best outcome.

Question 3

Before processing the BAM file a report was generated in order to extract some useful information from it. For that, RSeQC (Wang, Wang, & Li, 2012) was locally installed from source and used like this:

    thoth.cryst.bbk.ac.uk> ~/.local/bin/bam_stat.py -i BBQ.bam
    Load BAM file ... Done

#==================================================
#All numbers are READ count
#==================================================

Total records: 3597656
QC failed: 0
Optical/PCR duplicate 0
Non primary hits 0
Unmapped reads: 197548
mapq < mapq_cut (non-unique): 2565319

mapq >= mapq_cut (unique): 834789
Read-1: 0
Read-2: 0
Reads map to '+': 417536
Reads map to '-': 417253
Non-splice reads: 834789
Splice reads: 0
Reads mapped in proper pairs: 0
Proper-paired reads map to different chrom: 0

In order to separate the BBQ.bam in two, with one part including all the mapped and the other all the unmapped reads, the following is run:

# samtools seperation
samtools view -h -F 4 BBQ.bam | samtools view -bS > BBQ_mapped.bam
samtools view -h -f 4 BBQ.bam | samtools view -bS > BBQ_unmapped.bam

To get the uniquely mapped reads the following command was run:

grep -v "XS:i" | grep "AS:i" BBQ.sam >| Uniquely_mapped.sam

samtools view -h BBQ.bam | grep -E -v "XS:i" \
| grep -E "@[AS:i]" | samtools view -b - >| \
Uniquely_mapped.bam

and for multi-mapped reads respectively:

samtools view BBQ.bam | grep -v NH:i:1 \
| perl -pe 's/AS.+(NH:i:\d+)/\$1/' | \ncut -f1,10,12 | perl -pe 's/NH:i://' \n| sort -u -k3,3nr > sorted-multi-mapped.txt

The output format of this file is: read_ID<tab>read<tab>number times mapped

Unfortunately, I couldn’t continue further with this practical as for some reason the hope server refused to accept my SSH connection.

[wizofe@beastie ]$ ssh vi001@hope-ext.cryst.bbk.ac.uk
ssh: connect to host hope-ext.cryst.bbk.ac.uk port 22: Connection refused
Note: For all the code examples and in order to be nicely displayed for the pdf the continuity shell character has been used ‘\’. Sometimes, this is not parsed correctly from some PDF viewers, so the copy paste of the code may not work as it is. Thus, one can remove the character and concatenate the command.
References


