QAA

Emily Bratlett

2023-09-15

QAA Write Up

INTRO:

Supplied with two libraries of files (R1 and R2):

- 2_2B_control_S2_L008 (CONTROL)
  - 2_2B_control_S2_L008_R1_001.fastq.gz
  - 2_2B_control_S2_L008_R2_001.fastq.gz
- 19_3F_fox_S14_L008 (FOX)
  - 19_3F_fox_S14_L008_R1_001.fastq.gz
  - 19_3F_fox_S14_L008_R2_001.fastq.gz

Part 1: Read Length and Quality Score Distributions

Quality Score and N Content Distribution

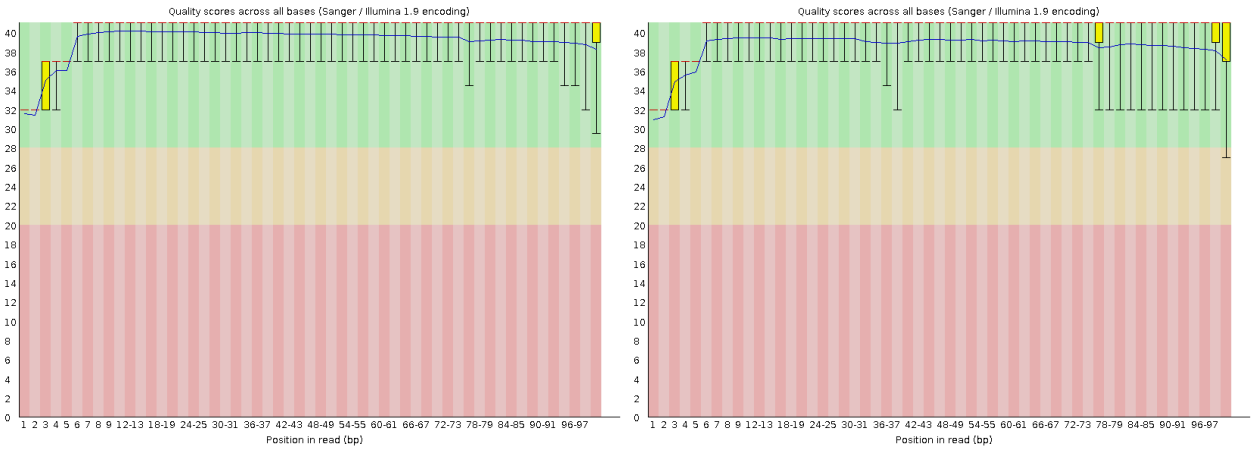

Figure 1: FastQC Per Base Quality Scores for CONTROL R1 (left) and R2 (right)

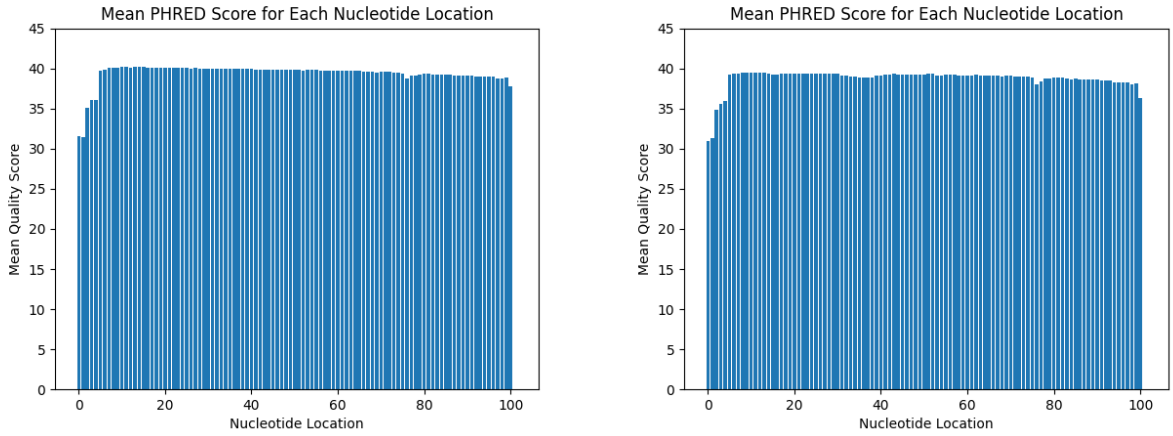

Figure 2: GGPlot Per Base Quality Scores for CONTROL R1 (left) and R2 (right)

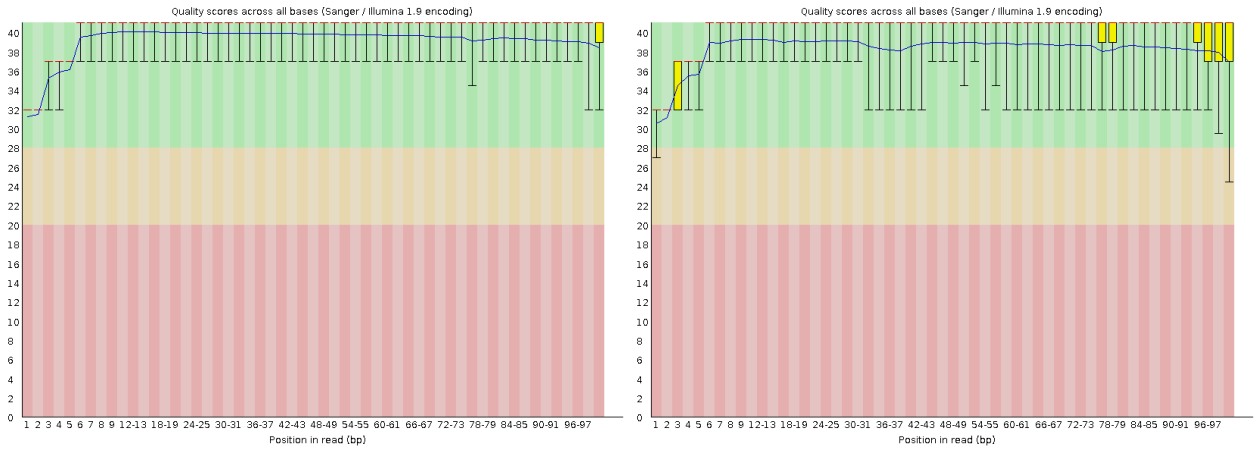

Figure 3: FastQC Per Base Quality Scores for FOX R1 (left) and R2 (right)

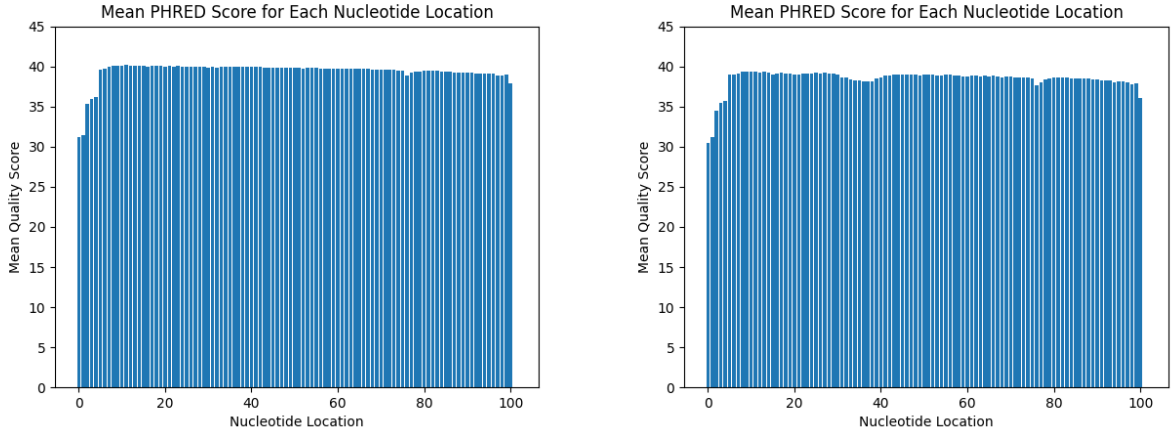

Figure 4: GGPlot Per Base Quality Scores for FOX R1 (left) and R2 (right)

Note on plots made by FastQC vs GGPlot using a python script:
The per base quality score plots generated by FastQC are similar to the plots generated by GGPlot. Both plots show poorer quality scores in the early basepairs of each read along with the tail of the basepairs. There is also evidence of lower quality in the R2 files. This is expected because of the increased chance of both reagent and sequence degredation by the time R2 is being processed.

Does Runtime Differ? The runtimes to generate plots by fastQC are faster compared to a python script that generates plots by ggplot. I would expect this as my python scipt hasnt gone through any optimization adjustments while FastQC is a publicly avail;able software specifically for this.

- FastQC: ss
  - CONTROL R1: 45.12
  - CONTROL R2: 15.23
  - FOX R1: 1:11.23
  - FOX R2: 1:41.07
- Python Script and GGPlot: m:ss
  - CONTROL R1: 1:31.62
  - CONTROL R2: 1:33.31
  - FOX R1: 3:53.99
  - FOX R2: 3:55.13

CONTROL and FOX Per Base N Content Distribution

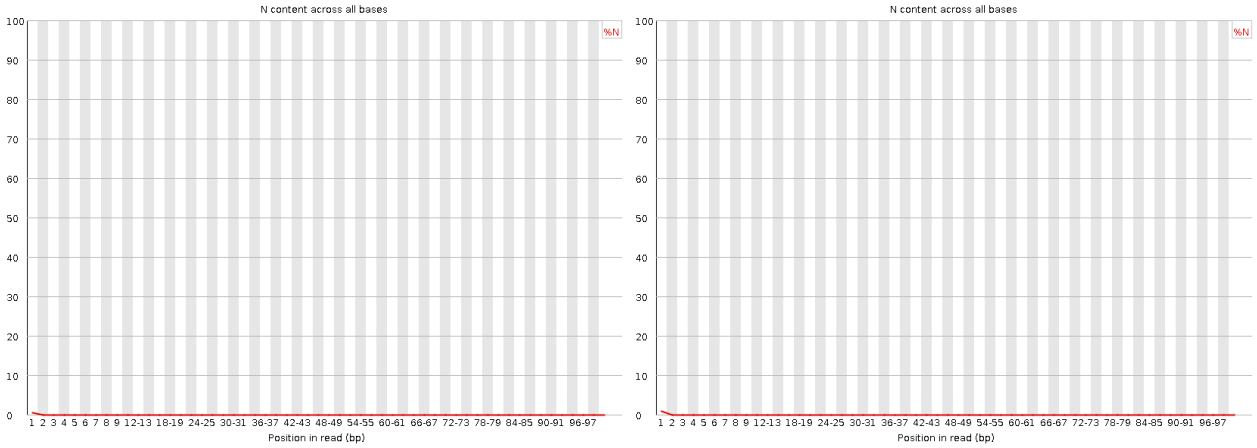

Figure 5: FastQC Per Base N Content for CONTROL R1 (left) and R2 (right)

Note on the N Content Distribution and Quality Score Distribution: The per Base N content appears to be very low for both CONTROL reads (Figure 3) with only a slight elevation in the earlier nucleotides. The slight elevation in N content is reflected in the dip in per base quality scores seen in Figure 1. A low N content supports these being a high quality sequencing read(s). ## **Overall Data Quality of Libraries:** The polts generated by FastQC and quality score python scripts suggest quality data for both the CONTROL and FOX libraries. Per base quality scores run along a trend line that hovers around 40, with the expected dip in the beginnings and ends. N content is minimal in the graphs as well. GC content based on FastQC data appears to follow expected distribution for both libraries as well, sequencers can have a hard time determining sequences with high GC content, so when base calling calls a GC content

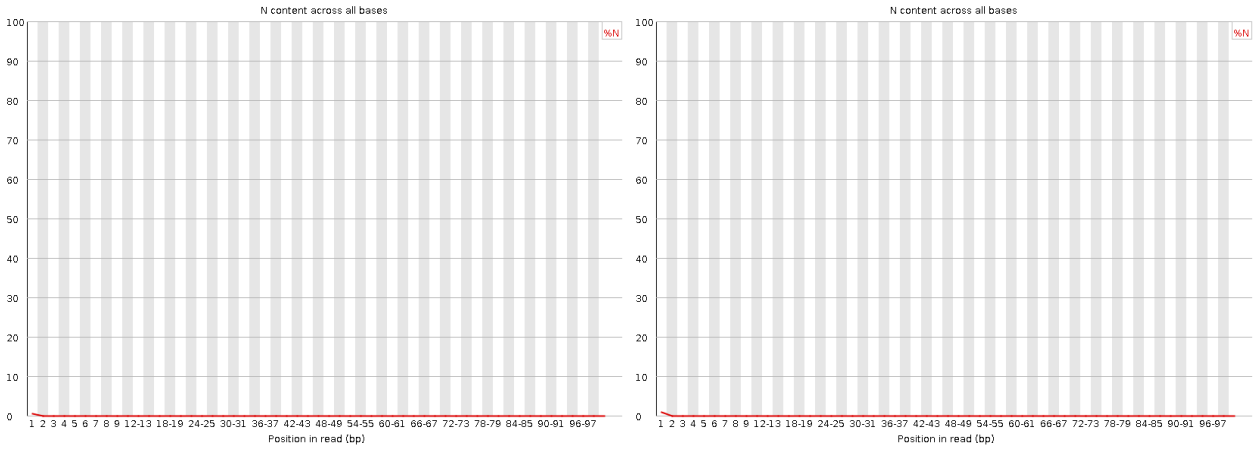

Figure 6: FastQC Per Base N Content for CONTROL R1 (left) and R2 (right)

comparable to what is expected, it is a sign of quality data. I think these libraries are both quality enough to proceed to adapter trimming, sequence quality trimming, and alignment.

Part 2: Adaptor trimming comparison
cutadapt To trim a 3' adapter

- Adapters:
  - R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
  - R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Adapter Sequence Confirmation Using the bash command grep, I searched for both the R1 and R2 adapters in both the R1 and R2 reads for each library. Unsurprisingly the R1 adapter was only found in the R1 file.

Library	# Times R1 Adapter Found	# Times R2 Adapter Found
CONTROL R1	31917	0
CONTROL R2	0	31965
FOX R1	13819	0
FOX R2	0	14775

Bash Commands

```
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/19_3F_fox_S14_L008_R1_001.fastq.gz
| grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/19_3F_fox_S14_L008_R1_001.fastq.gz
| grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/19_3F_fox_S14_L008_R2_001.fastq.gz
| grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/19_3F_fox_S14_L008_R2_001.fastq.gz
| grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R1_001.fastq.gz
| grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
```

```
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R1_001.fastq.gz
| grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R2_001.fastq.gz
| grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" |wc -l
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R2_001.fastq.gz
| grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" |wc -l
```

Proportion of reads trimmed The following table shows how many base pairs each read of the libraries was (pre and post filtering), and what number of base pairs were removed from each read, along with the percentages.

Library	# Basepairs Processed	# Basepairs Written	# R1 Basepairs/% Trimmed	# R2 Basepairs/% Trimmed
CONTROL	1177794330	1160435631	423,128 (7.3%)	473,368 (8.1%)
FOX	3302347510	3294067204	546,623 (3.3%)	676,564 (4.1%)

Trimmomatic to quality trim your reads

- Trimmomatic Run with the following soecifications
  - LEADING: quality of 3
  - TRAILING: quality of 3
  - SLIDING WINDOW: window size of 5 and required quality of 15
  - MINLENGTH: 35 bases

Rate of R1 vs R2 Trimming Rates I would expect the rates of quality trimming by trimmomatic to be higher in read 2 than for read 1 (for both libraries) because base call quality usually diminishes as a run goes on. Reagents start to break down, starting genomic material has now gone through multiple rounds of amplification so degradation is bound to occur. See trimmomatic output results in the table bellow. WHen you look at figure 7 and 8 you can see there are a higher number of read lengths for R2 after trimmomatic. Adapter trimming rates should be the same for both R1 and R2 since the whole goal of the software is to remove adapters if the adapters are accidentally sequenced in a read. This would occur if the read length happens to be longer than the insert length, which should be evenly distributed across reads.

Library	# and % Read Pairs for CONTROL	# and % Read Pairs for FOX
Input Read Pairs	5830665	16348255
Both Surviving	5652538 (96.94%)	15899268 (97.25%)
Forward Only Surviving	133553 (2.29%)	428450 (2.62%)
Reverse Only Surviving	4598 (0.08%)	14868 (0.09%)
Dropped	39976 (0.69%)	5669 (0.03%)

Read Length Distribution Graphs

Part 3 - Alignment and Strand Specificity

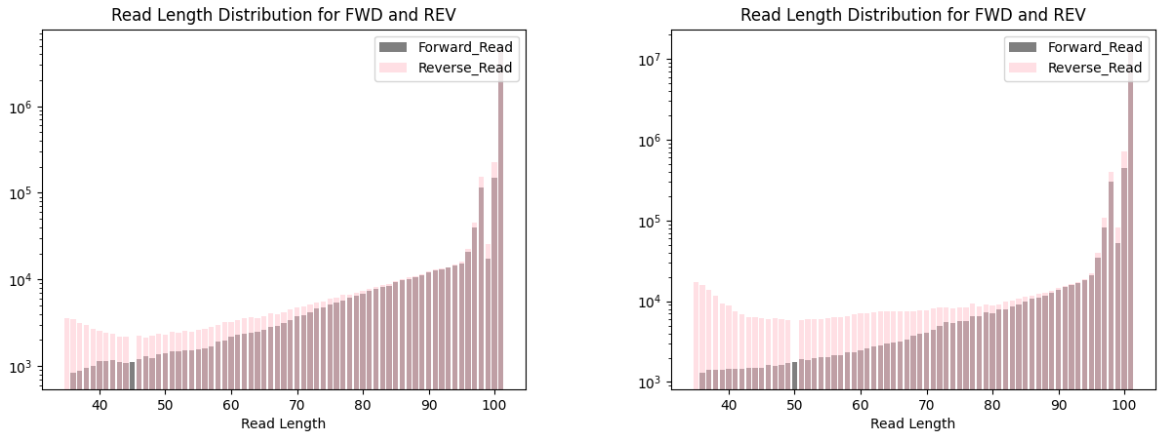

Figure 7: Read Length Distribution for CONTROL (left) and FOX (right)

STAR - Alignment of reads to mouse genomic database
`Script Location: /projects/bgmp/ebart/bioinfo/Bi623/QAA/star.sh` ##### fasta and gtf files from ensembl:

STAR Alignment

- Script Location: /projects/bgmp/ebart/bioinfo/Bi623/QAA/star_align.sh

Mapped and unmapped reads in SAM files: * Script used:/projects/bgmp/ebart/bioinfo/Bi623/QAA/mapped_unmapp

Library	# Mapped Read	# Unmapped Reads
CONTROL	11078806	226270
FOX	30512174	1286362

HTSeq

`HTSeq Script File Path`
`/projects/bgmp/ebart/bioinfo/Bi623/QAA/htseq/htseq.sh`

Post HTSeq How Many Reads Mapped

`Bash Command:`
`* Reads that mapped to a feature Command: cat <file> | grep -v "^__" | awk '{sum+=$2} END {print sum}'`
`* Total number of reads Command: cat <file> | awk '{sum+=$2} END {print sum}'`

Library	# of Reads Mapped	Total # Read
CONTROL (Stranded=Yes)	220,195	5,652,538
CONTROL (Stranded=Rev)	4,805,681	5,652,538

Library	# of Reads Mapped	Total # Read
FOX (Stranded=Yes)	545,746	15,899,268
FOX (Stranded=Rev)	12,935,826	15,899,268

Is data from a "strand-specific" library Both the FOX library and the CONTROL library are strand-specific. For the FOX library, when the HTSeq parameter was set to "stranded=yes" only 3.4% of reads mapped to features when compared to "stranded=reverse" where 81.4% of reads mapped to features. For the CONTROL library, when HTSeq was set to "stranded=yes" only 3.9% of reads mapped to features when compared to "stranded=reverse" where 85% of reads mapped to features. For the CONTROL library I also ran the parameter as "stranded=no", when I did this, 83.2% of reads mapped to features. If your libraries were stranded, you would expect to see a 50/50 split of reads mapped to total reads when you compare "stranded=yes" vs "stranded=no", which isn't what we see. When "stranded=yes" has a low percentage, it is because the libraries are strand specific for the reverse strand. Knowing these are kappa libraries also supports this. SInce illumina has trademarked stranding the forward strand, other library prep kits such as kappa use reverse strand specific library preps.