QAA Report

Jacob Jensen

2022-09-07

My libraries were 3_2B_control_S3_L008_R*_001 and 17_3E_fox_S13_L008_R*_001. I will refer to them as 3* and 17* for convenience.

Part 1 - Read quality score distributions

Quality score distributions:

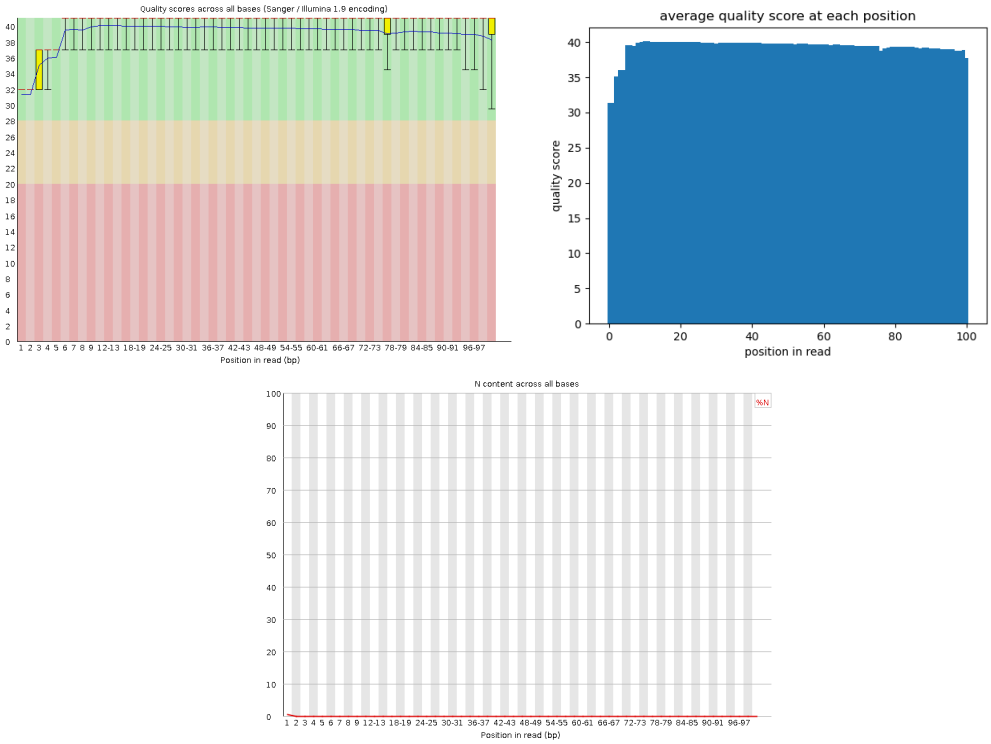


Figure 1: Per base quality plots for 3R1

These plots show the same trends. The plots produced by fastqc have information about the distribution of quality scores in each bin of base pairs, whereas mine only shows the mean quality score at each base pair. The runtime differs greatly. Fastqc took 196.62 seconds to process all four files, while the fastest my script processed a single file was 142.54 seconds. This runtime difference is because fastqc was built by a group of experts in a fast language, while my script was written by me in a slow language.

The average quality of both libraries appears to be good. Both read 1 files have reads with higher quality scores than their read 2 counterparts, but the average quality score at all base pairs is still good for the

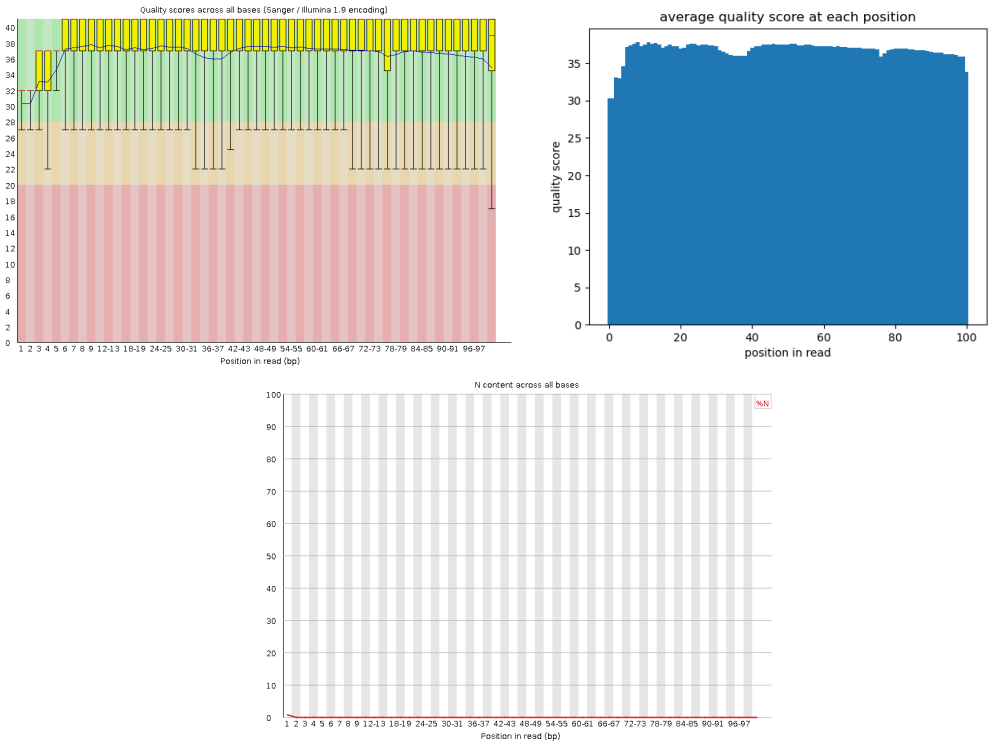


Figure 2: Per base quality plots for 3R2

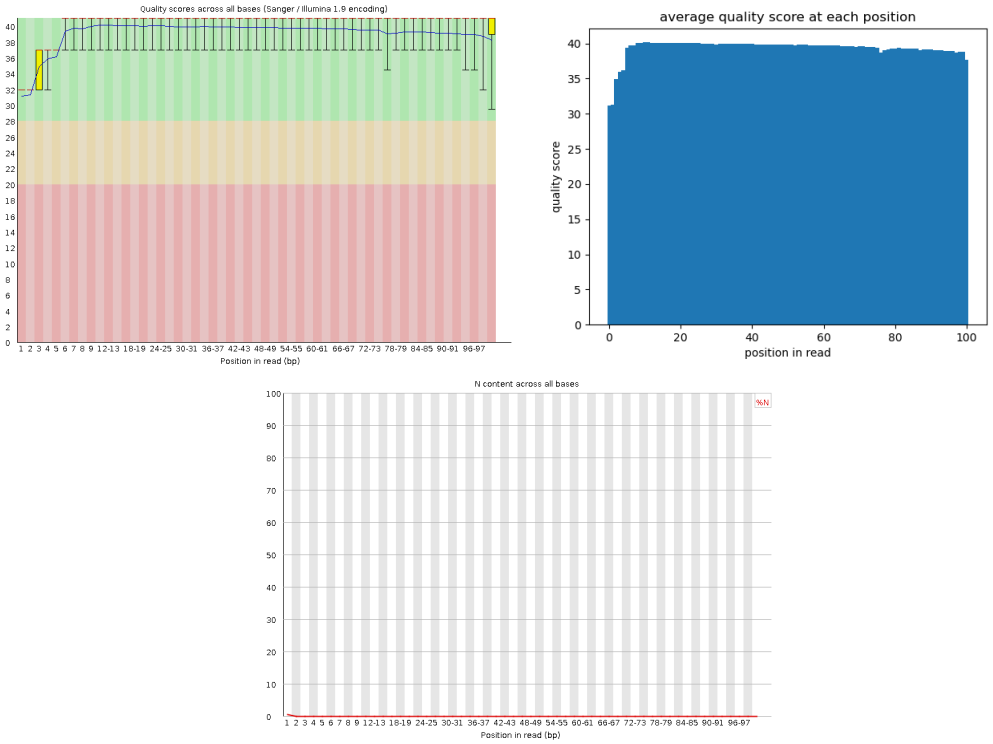


Figure 3: Per base quality plots for 17R1

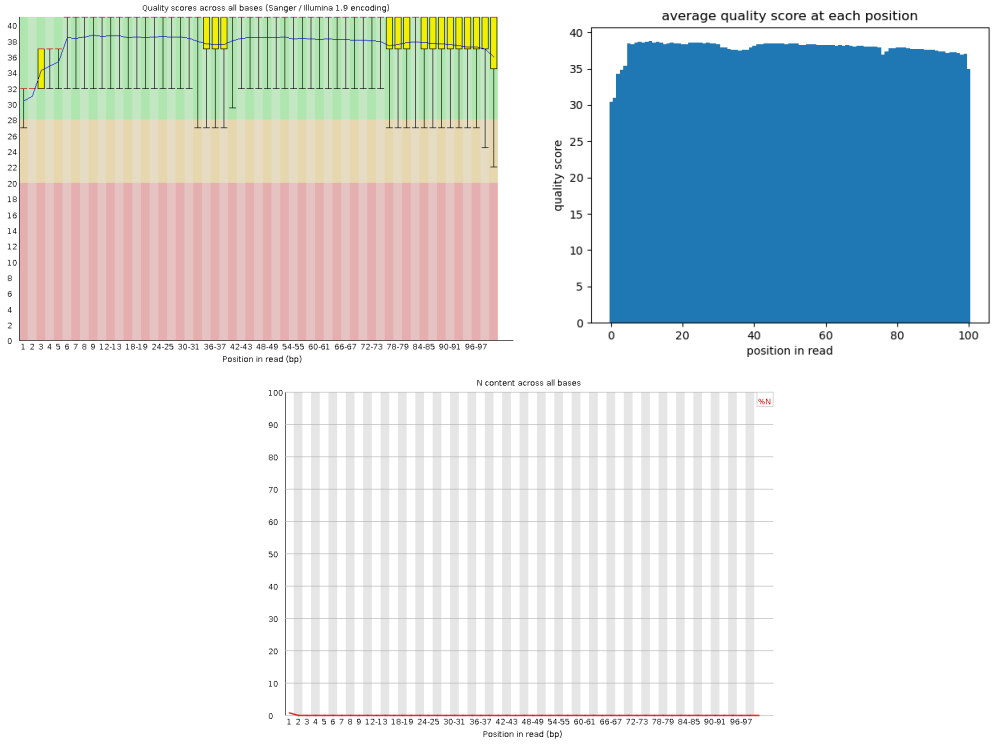


Figure 4: Per base quality plots for 17R2

second reads. The per base n content does not show any major issues, with only a few Ns being present almost exclusively at the beginning of reads, where the quality is suspect. Other output from fastqc showed no major issues with the libraries. There were some overrepresented sequences that blasted against the 18S rRNA in the R2 files. There were some TruSeq adapters present in the 17*R1 file, and an Illumina primer in the 17*R2 file.

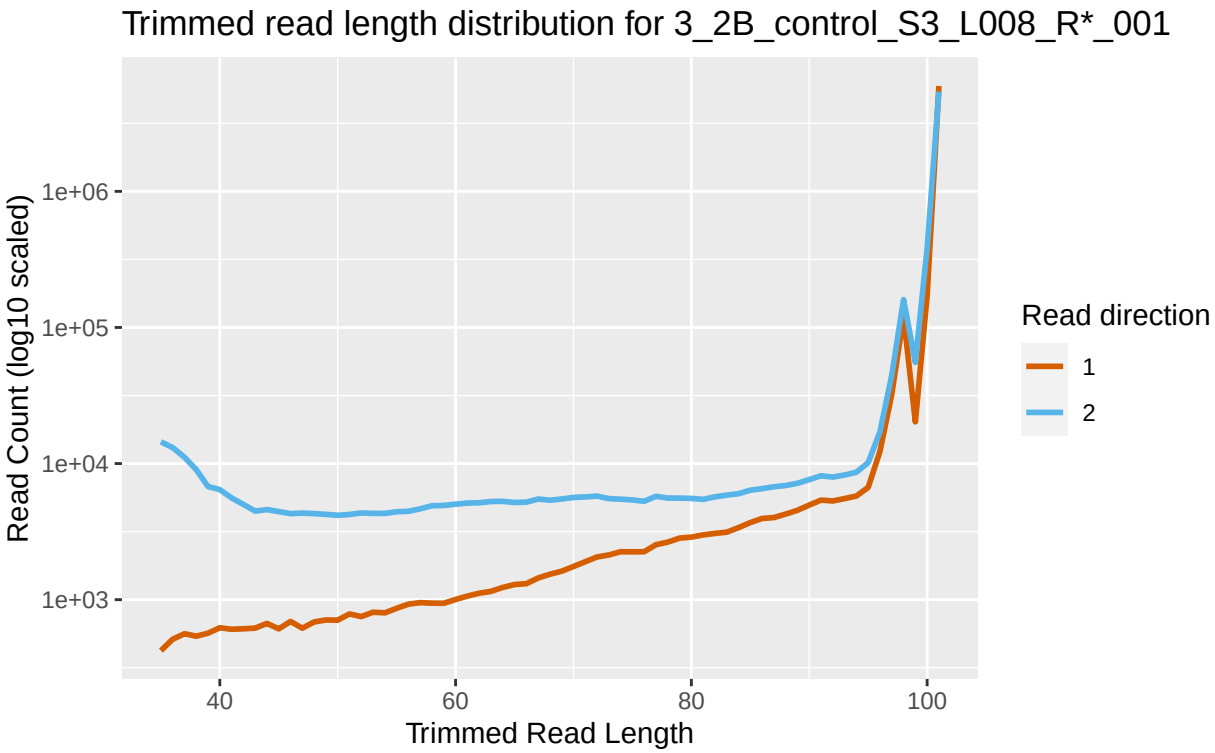
Part 2 - Adapter trimming comparison

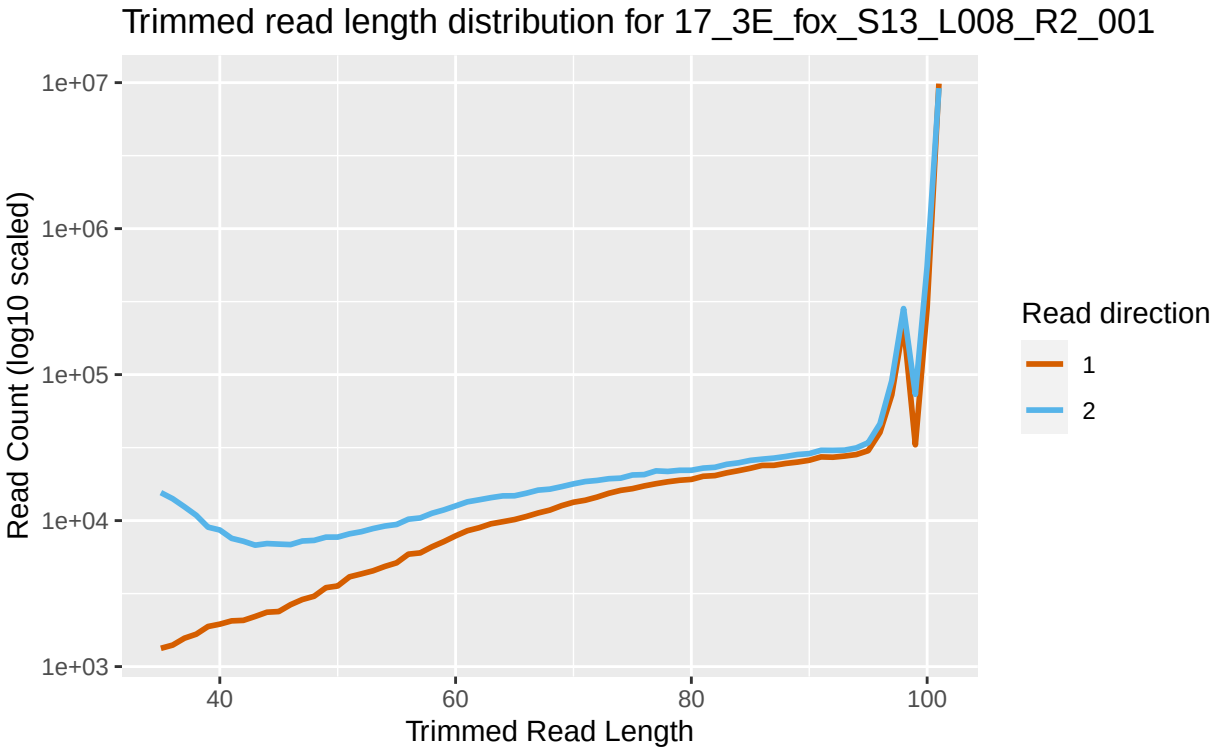
Grepping for the adapter and primer sequences listed by fastqc showed results in the beginning and middle of the sequences, so I ignored them and looked at the cutadapt documentation for TruSeq adapter sequences. It listed AGATCGGAAGAGCACACGTCTGAACTCCAGTCA and AGATCGGAAGAGCGTCGTG-TAGGGAAAGAGTGT as the sequences for R1 and R2, respectively. Grepping for these sequences found hits towards the end of sequences, which is to be expected for adapters. These sequences matched the sequences in the hint, so I proceeded.

cutadapt trimmed 3.2% of reads for 3*R1, 3.9% of reads for 3*R2, 8.7% of reads for 17*R1, and 9.4% of reads for R2.

trimmomatic was run with these parameters: `LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35`

Trimming produced paired reads with the following length distributions:





As the line for R2 is higher than the line for R1 for shorter read lengths, we know that R2 reads were trimmed more than R1 reads. Based on the cutadapt output, we know that the R2s had more adapter trimming. Based on the per base quality, we can assume that trimmomatic likely removed more bases from the R2s as well.

Part 3 - Alignment and strand-specificity

I downloaded the most recent mouse genome and gtf from ensembl. I used STAR to align the processed reads to the mouse genome.

After aligning, my script counted the total number of mapped and unmapped reads. I used htseq-count to check for mapping patterns between the two strandedness options, and the mapped reads were counted with `grep -v "^_" $htseq-count_output | cut -f 2 | awk '{s+=$1} END {print s}'` producing these results:

Table 1: Summarized read counts

	total mapped	unmapped	forward stranded	reverse stranded
3*	12359958	496080	245058	5260739
17*	21532824	948708	443307	8952669

As the htseq-count results are not close to each other within each library, we know that the libraries are stranded. As the counts are significantly higher with `stranded=reverse` (5015681 higher for 3* and 8509362 higher for 17*), we know that the R2 reads have the sequence of the coding strand.