Quality Assessment

André Watson

January 3, 2024

Contents

1 Read Quality Score Distributions

1.1 Data

I was tasked with analyzing sequencing data containing two sets of paired-end reads. The first set of read pairs is contained in files

- `10_2G_both_S8_L008_R1_001.fastq.gz`
- `10_2G_both_S8_L008_R2_001.fastq.gz`

and the second set of read pairs is contained in

- `31_4F_fox_S22_L008_R1_001.fastq.gz`
- `31_4F_fox_S22_L008_R1_001.fastq.gz`.

1.2 FastQC

Using FastQC, I produced graphs of average quality score across reads by base position. The plots for the "10_2G_both" and "31_4F_fox" libraries follow.

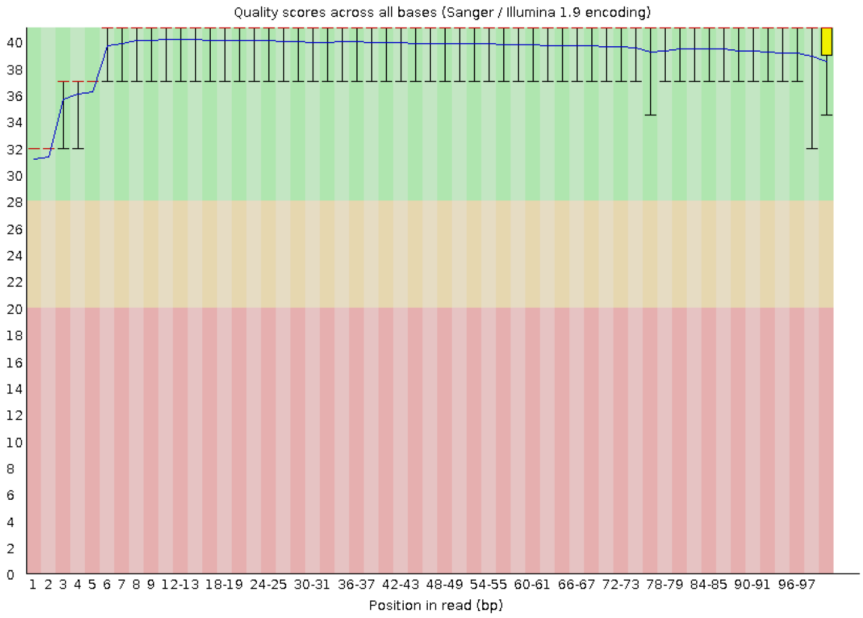


Figure 1: Quality scores per base position for read 1 of 10_2G_both. Solid line indicates mean quality score.

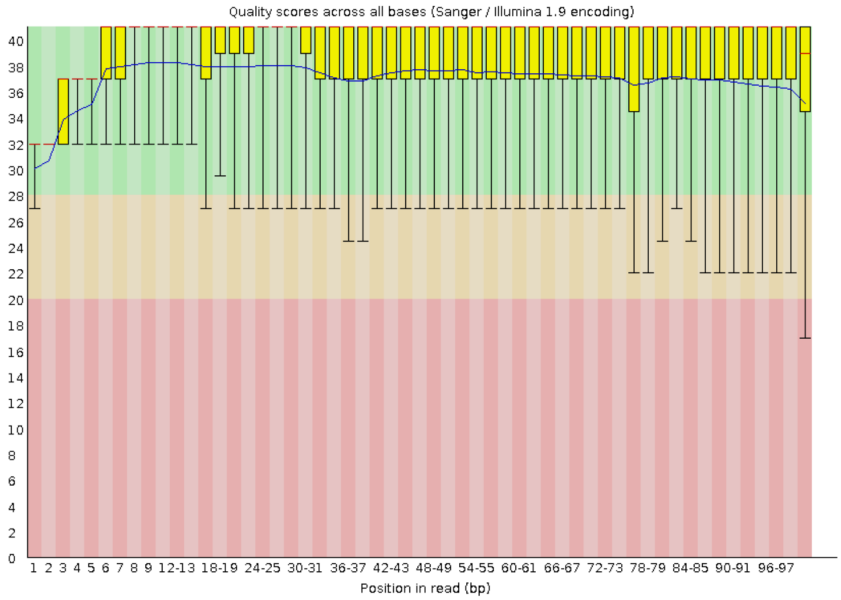


Figure 2: Quality scores per base position for read 2 of 10_2G_both. Solid line indicates mean quality score.

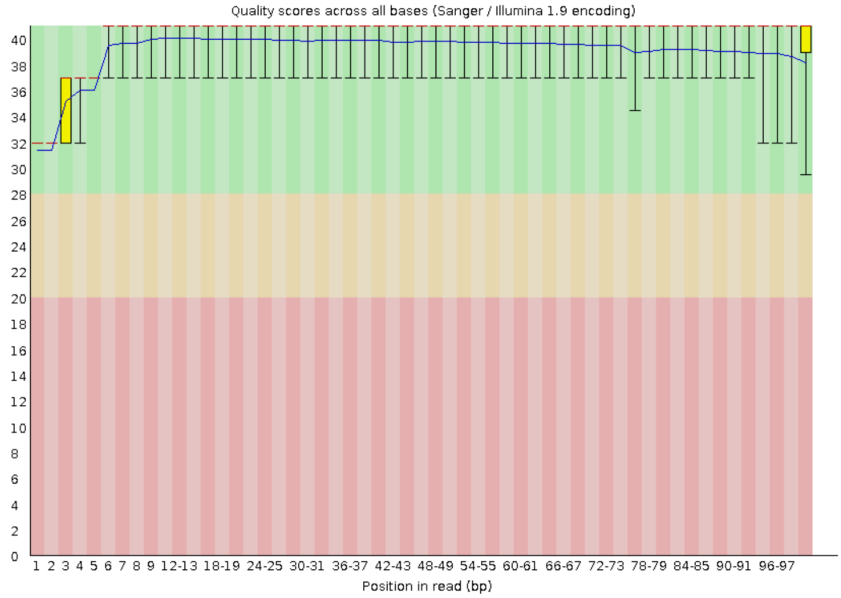


Figure 3: Quality scores per base position for read 1 of 31_4F_fox. Solid line indicates mean quality score.

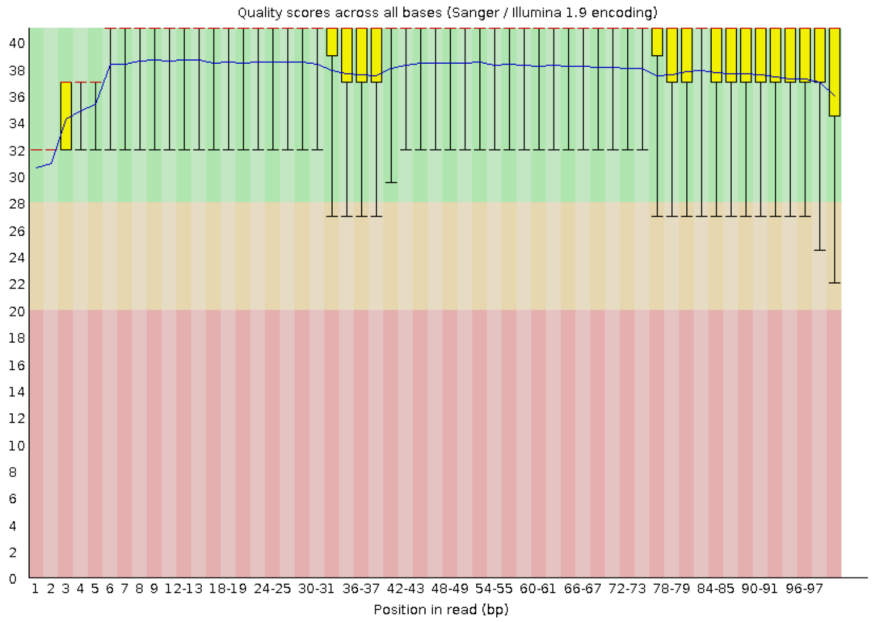


Figure 4: Quality scores per base position for read 2 of 31_4F_fox. Solid line indicates mean quality score.

Although mean quality scores remain high, we see a decrease in each library's R2 reads, with lower minimum quality scores at that base and more variable quality scores. This may simply be due to the R2 reads occurring later, and therefore allowing more time for the molecules being sequenced to degrade.

Low-quality read scores may be associated with base calls of "N", indicating an unknown nucleotide. FastQC also lets us analyze per-base N content.

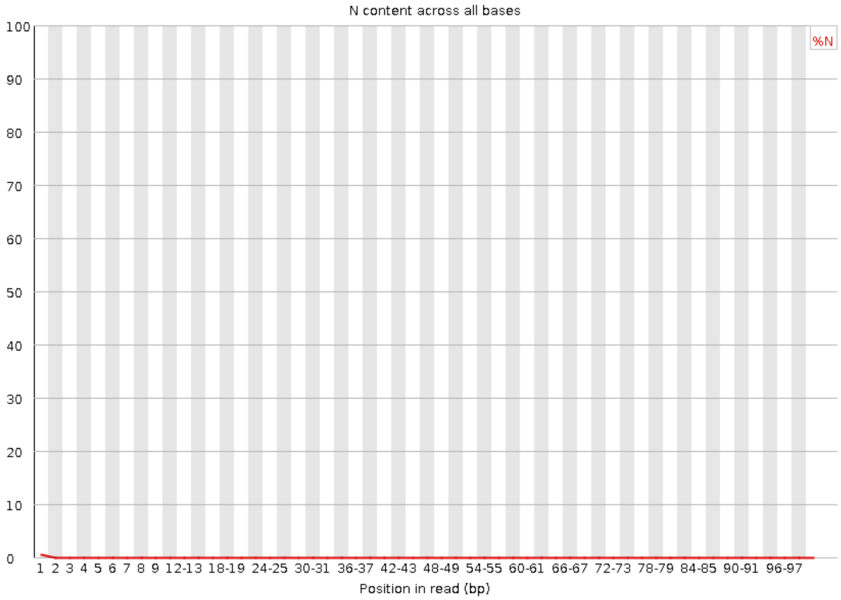


Figure 5: Per-base N content for read 1 of 10_2G_both.

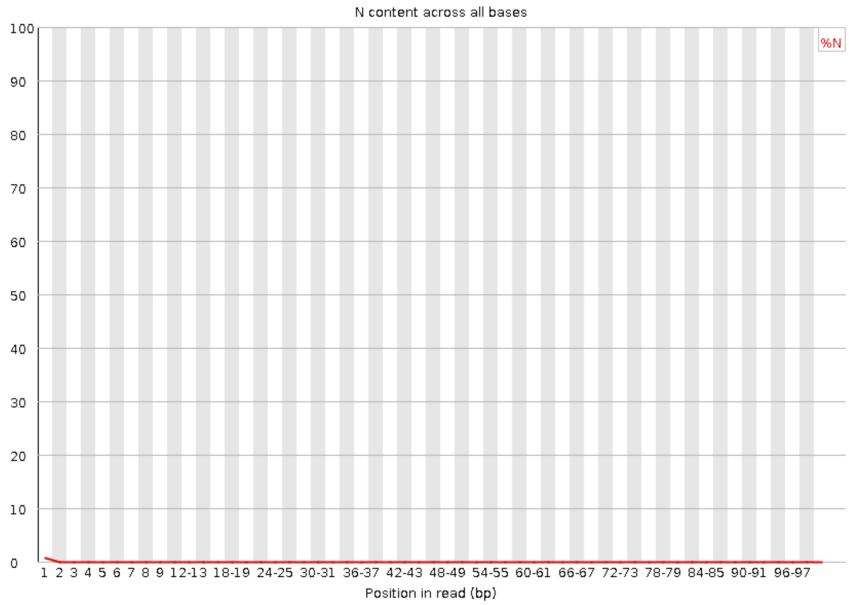


Figure 6: Per-base N content for read 2 of 10_2G_both.

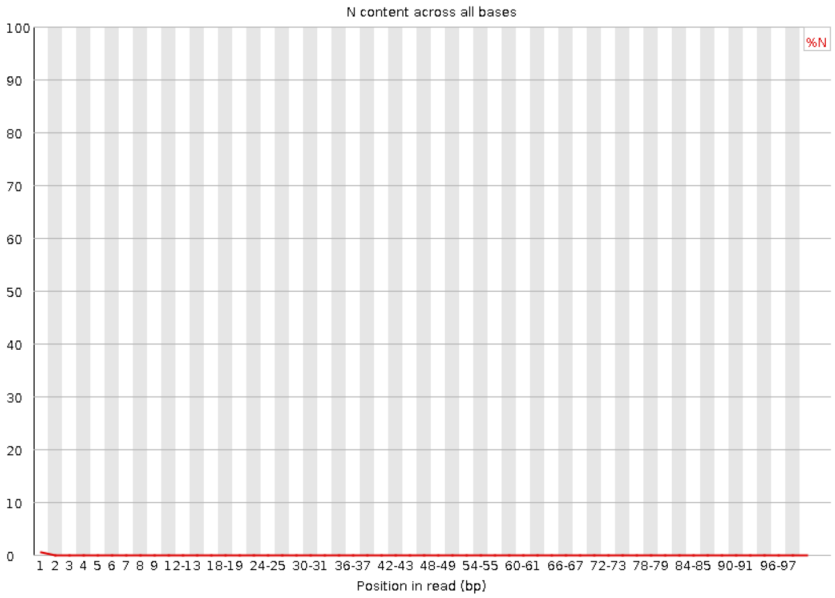


Figure 7: Per-base N content for read 1 of 31_4F_fox.

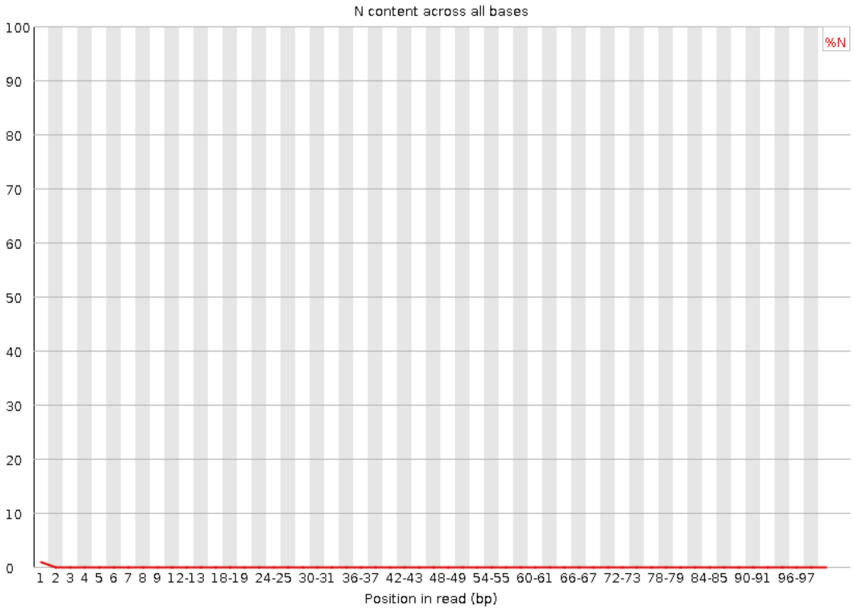


Figure 8: Per-base N content for read 2 of 31_4F_fox.

We see very few N calls, with slight upticks at the beginning of reads. This is consistent with the lower mean quality score early in each read across both reads for both our libraries.

1.3 Original Plots

I also generated per-base position distributions of quality scores using my own script from previous work.

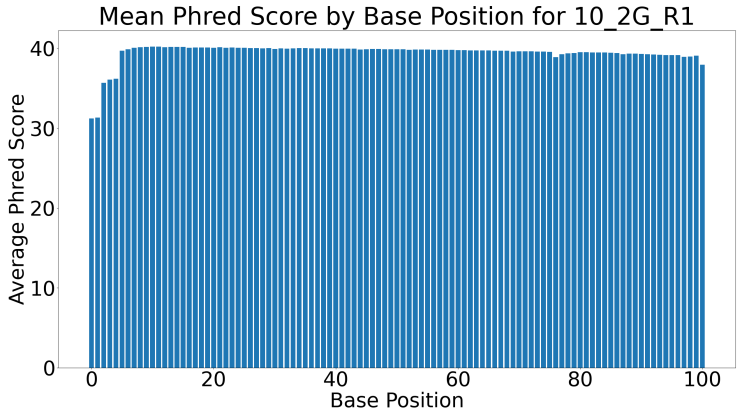


Figure 9: Quality scores per base position for read 1 of 10_2G_both

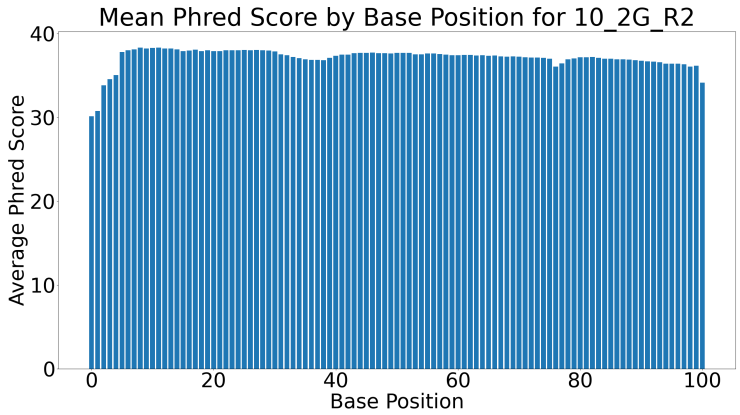


Figure 10: Quality scores per base position for read 2 of 10_2G_both.

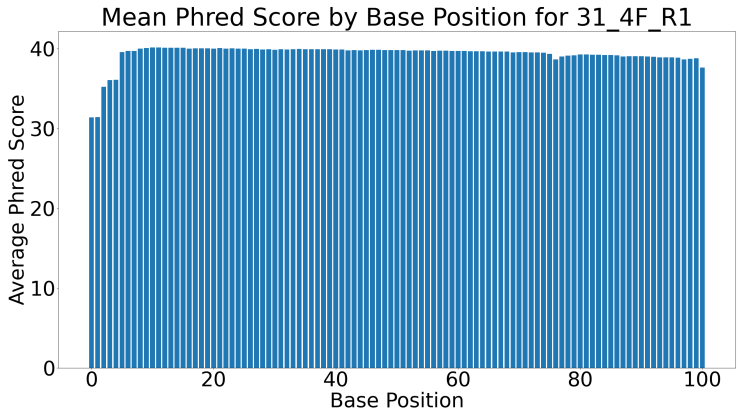


Figure 11: Quality scores per base position for read 1 31_4F_fox.

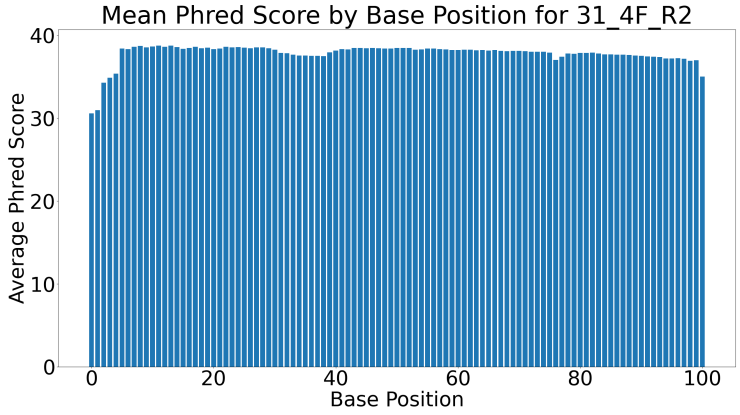


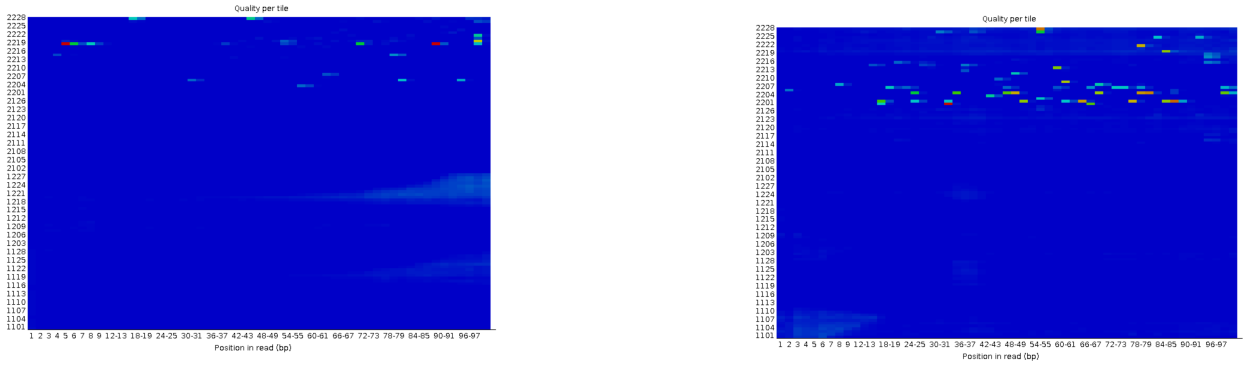
Figure 12: Quality scores per base position for read 1 31_4F_fox.

Similarly to FastQC's plots, we see a dip in average quality scores at the beginning of reads. Unlike FastQC, my plots don't include a visualization of the spread of quality scores, which hides the increased variability in quality scores for R2. However, the overall shape of the distribution and the high average quality scores are consistent with FastQC.

I also generated these plots more slowly—generation took just over an hour for all four plots, compared to FastQC, which took 15 minutes for all four analyses.

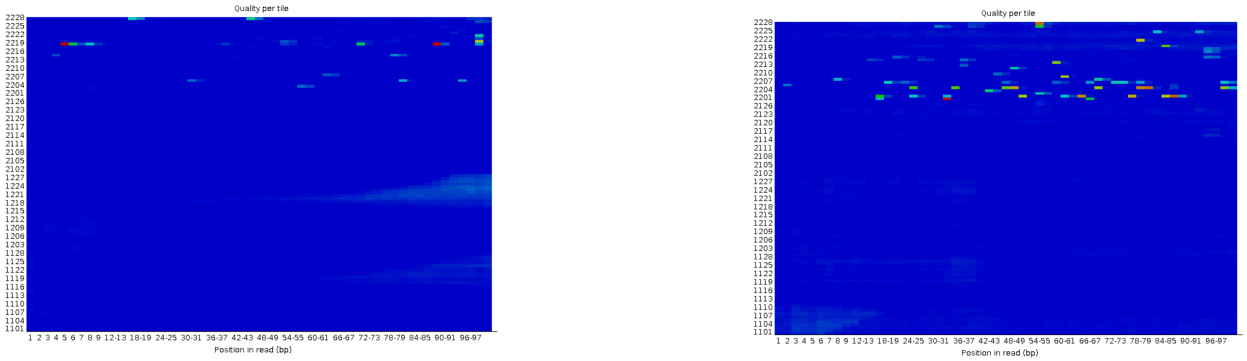
1.4 Data Quality

Quality scores look good if we only account for base position. We should also examine if there's a major issue with the instruments causing issues in certain areas on the flowcell. Below are the plots FastQC generates for average quality score for each base across flowcell tiles.



(a) (b)

Figure 13: Per-base average quality scores by tile for R1 (a) and R2 (b) of 10_2G_both



(a) (b)

Figure 14: Per-base average quality scores by tile for R1 (a) and R2 (b) of 31_4F_fox

Both cases look fine. There are no tiles where the entire row is low-quality, there are no large areas of low-quality data, and while there is a dip in quality near the end of both R1 reads for a few tiles, it's not extreme.

I would characterize these data as at least good enough to process further. We can let down-

stream quality filters catch any scattered low-quality reads, but there's no reason to toss this data entirely.

2 Adapter trimming comparison

2.1 Trimming

FastQC identified some adapter contamination in the 31_4F_fox library near the end of reads that it labeled as "Illumina universal adapter". I was able to find that FastQC considers Illumina universal adapter to be the sequence `AGATCGGAAGAG`. FastQC did not identify the same adapter in the 10_2G_both library.

To confirm the presence of the adapter in the libraries, I searched for it in the FASTQ files using the command `zcat <filename> | grep "AGATCGGAAGAG" --color='always'| head`. This extracts sequence lines containing the adapter, highlights the sequence, and displays the first few such lines. I found adapter sequences at the ends of reads in all four files.

The bases immediately following those 12 nucleotides were different between R1 and R2 files, implying that different adapters were used for each end of the molecules being sequenced. I used the full sequences to direct trimming.

- The adapter sequence trimmed from forward reads was `AGATCGGAAGAGCACACGTCTGAACTCCAGTCA`
- The adapter sequence trimmed from reverse reads was `AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT`

Adapter trimming was done using `cutadapt` with default settings. For each file, `cutadapt` reported:

- 10_2G_both, R1: 2.617% of reads were trimmed
- 10_2G_both, R2: 3.401% of reads were trimmed
- 31_4F_fox, R1: 12.04% of reads were trimmed
- 31_4F_fox, R2: 12.74% of reads were trimmed

It makes sense that adapter is detected and trimmed at similar rates between R1 and R2: adapter contamination comes from sequencing through our insert, and we expect reads in both directions to read through a short insert.

Reads were quality-trimmed using `trimmomatic`, specifying paired-end reads and with the following options:

- LEADING: quality of 3
- TRAILING: quality of 3
- SLIDINGWINDOW: window size of 5 and required quality of 15
- MINLEN: 35 bases

For the 10_2G_both library, `trimmomatic` reported keeping 95.14% of paired reads. For 31_4F_fox, 94.97% of paired reads survived.

2.2 Trimmed Read Length Distribution

I plotted the distribution of trimmed read lengths, which appear below.

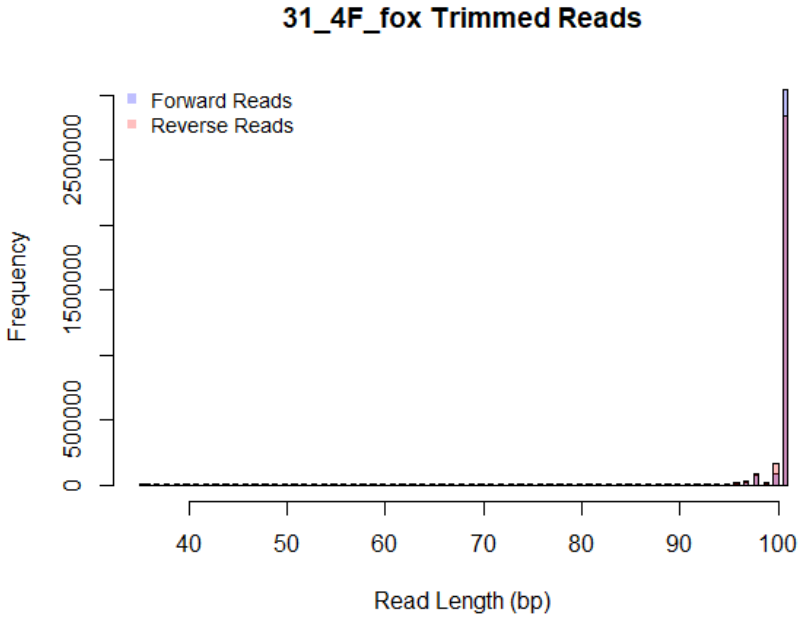


Figure 15: Read lengths for post-`trimmomatic` 10_2G_both data. Histograms are semi-transparent and overlaid to show overlap.

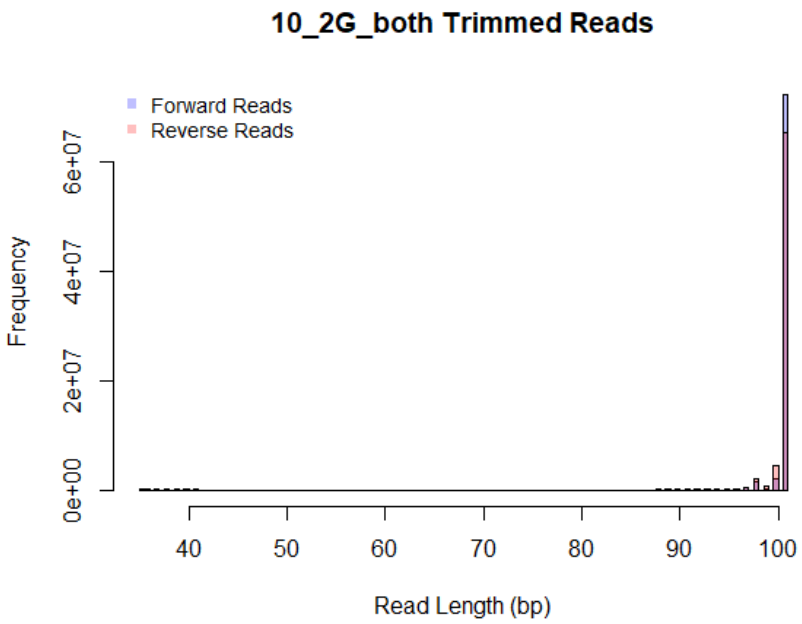


Figure 16: Read lengths for post-`trimmomatic` 31_4F_fox data. Histograms are semi-transparent and overlaid to show overlap.

Most quality-trimmed reads were reverse or R2 reads, so we see more forward reads at an untrimmed 101 bp and more reverse reads further left in the histogram. This is consistent with the slightly lower quality of R2 reads.

3 Alignment and strand-specificity

Reference genome and GTF file were downloaded from Ensembl (Ensembl 110, *Mus musculus*, GRCm39). Alignment to the genome was performed using STAR.

3.1 Alignment Mapping Results

Based on the SAM file's binary flag, we can count how many reads are mapped and how many are unmapped (ignoring secondary alignments) for each alignment.

- Alignment of 10_2G_both to the *Mus musculus* genome:
  - Mapped Reads: 152719219
  - Unmapped Reads: 2322587
- Alignment of 31_4F_fox to the *Mus musculus* genome:
  - Mapped Reads: 6969853
  - Unmapped Reads: 225963

3.2 Strandedness

`htseq-counts` outputs a file with predictable formatting. We can count the proportion of reads mapping to a feature (prefixed with `ENSMUS` in this case) out of total reads (mapped and unmapped) using the command `awk '$1 "ENSMUS" {acc_feat+=$2} {acc_tot+=$2} END {print(acc_feat/acc_tot)}' <GENECOUNTS FILE>`.

- 10_2G_both, `htseq-counts` option `stranded=yes`: 3.84% of reads mapped to a feature.
- 10_2G_both, `htseq-counts` option `stranded=reverse`: 86.88% of reads mapped to a feature.
- 31_4F_fox, `htseq-counts` option `stranded=yes`: 5.10% of reads mapped to a feature.
- 31_4F_fox, `htseq-counts` option `stranded=reverse`: 82.16% of reads mapped to a feature.

This library seems to have been strand-specific, with R2 corresponding to the sense strand. If our library had not been strand-specific, we would expect about the same percentage of reads in each direction to have mapped to features. Since so many more reads map to features when specifying `stranded=reverse`, we can say that the "reverse" reads (R2) are the direction corresponding to the sense strand.