QAA Report

Lena Allen

PART ONE: Read Quality Score Distributions

FastQC-Generated Per-Base Quality Score Plots:

Figure 1: Per-base sequence quality score plots generated using FastQC. Blue line represents mean, whiskers represent 10th and 90th percentiles, and yellow boxes represent interquartile range.

FastQC-Generated Per-Base N Content Plots:

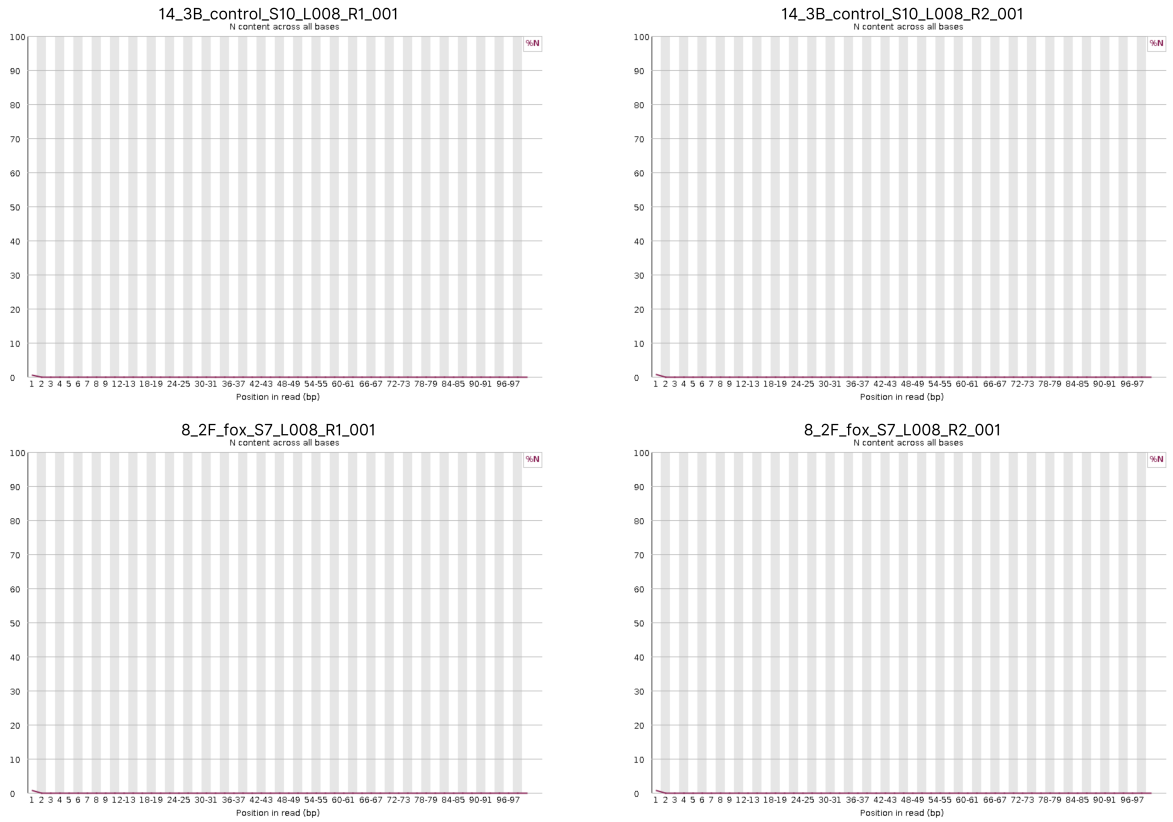

Figure 2: Per base N content plots generated using FastQC. There is very little N content present in these reads, and any present N content is restricted to the first few base positions.

For all four per-base N content plots produced by FastQC, the only detected N content was present in the first base position. This is consistent with the per-base quality score plots; for all files except `14_3B_control_S10_L008_R1` (in which position two has the lowest average quality score), base position one has the lowest average quality score.

Python Script-Generated Per-Base Quality Score Plots:

The below plots were all generated using my python script from demultiplexing, `qs_dist.py`:

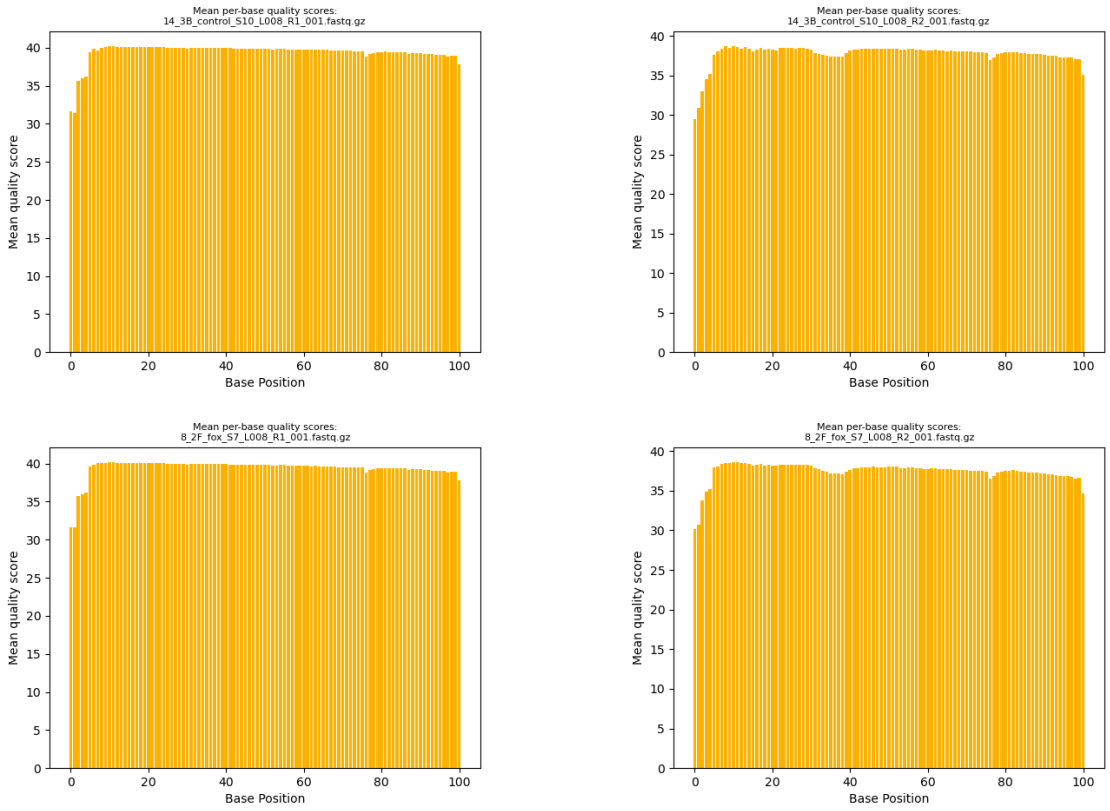

Figure 3: Mean per-base quality score distribution generated using my python script qs_dist.py

Comparison of quality score distribution plots- FastQC vs. My Python Script:

14_3B_Control_S10_L008_R1

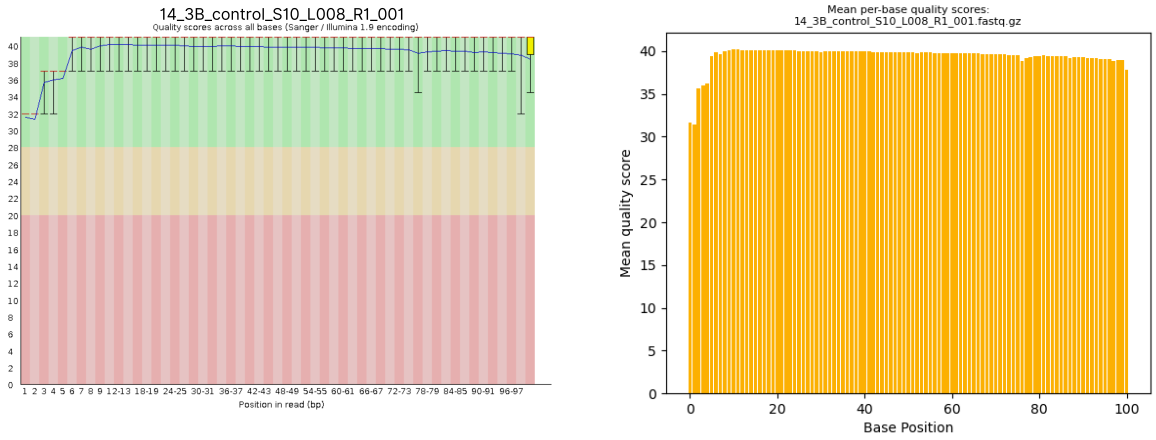

Figure 4: Comparison of per-base mean quality score plots for 14_3B_control_S10_L008_R1.FastQC (left) vs. qs_dist.py (right).

For read 1 of the 14_3B_control sample, the FASTQC plot is very similar to the plot generated by my python script. In both plots, the mean quality score of the second base position is lower than that of the first base position, and the mean quality score rises between base position three and seven. Additionally, there is a small dip in mean quality score observed in both plots around base position 76-77 and the final base position.

14_3B_control_S10_L008_R2

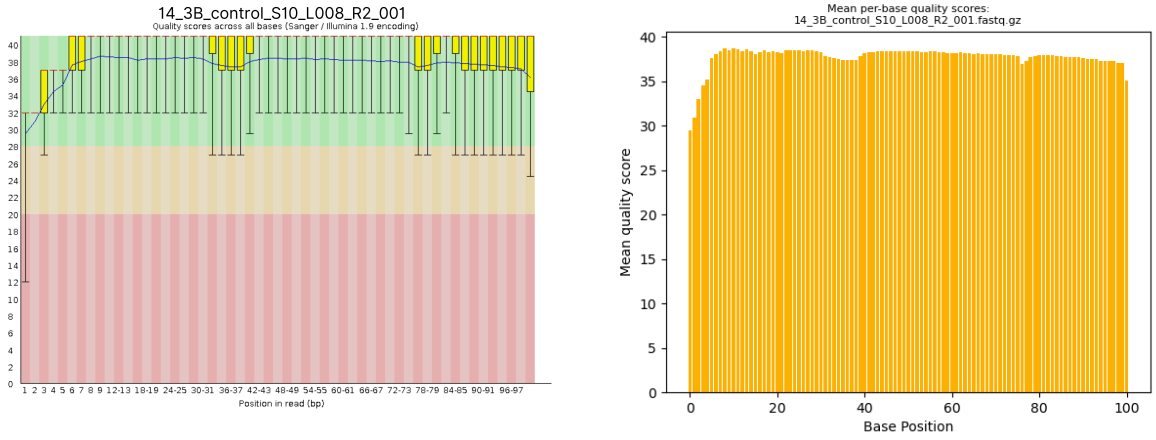

Figure 5: Comparison of per-base mean quality score plots fo 14_3B_control_S10_L008_R2.FastQC (left) vs. qs_dist.py (right).

Again, for read 2 of the 14_3B_control sample, the FASTQC plot is very similar to the plot generated by my python script. In both plots, mean quality score increases sharply between base positions one and six, then decreases between base positions 32-40 and 76-79. As observed in plots for read 1 of the

14_3B_control sample, mean quality score also decreases in both plots in the last two base positions. However, the FASTQC plot for this sample is much more descriptive, and you can see that the base positions towards the end of the read and those with low mean quality scores typically have interquartile ranges that are not equal to zero (meaning that the median is not equal to the lower and upper quartiles).

8_2F_fox_S7_L008_R1

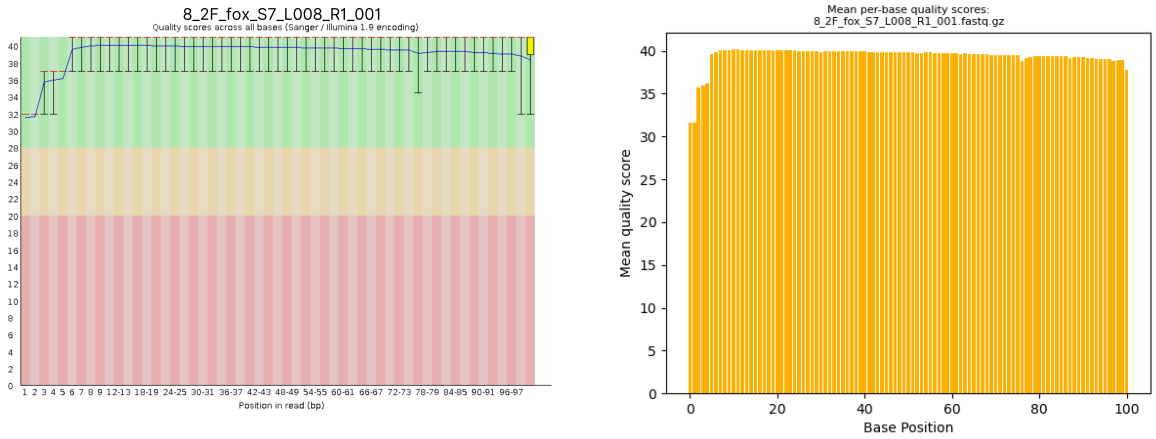

Figure 6: Comparison of per-base mean quality score plots for 8_2F_fox_S7_L008_R1.FastQC (left) vs. qs_dist.py (right).

For read one of the 8_2F_fox_S7_L008 sample, the per-base mean quality score plots generated by FASTQC and my python script are very similar. In both plots, mean quality score increases from base position one to base position seven. The mean quality scores then stay relatively consistent until base positions 76-77, where there is a small dip in mean quality score. Both plots also display a lower mean quality score for the last base position, which is the only position for which the median is not equal to both the upper and lower quartiles.

8_2F_fox_S7_L008_R2

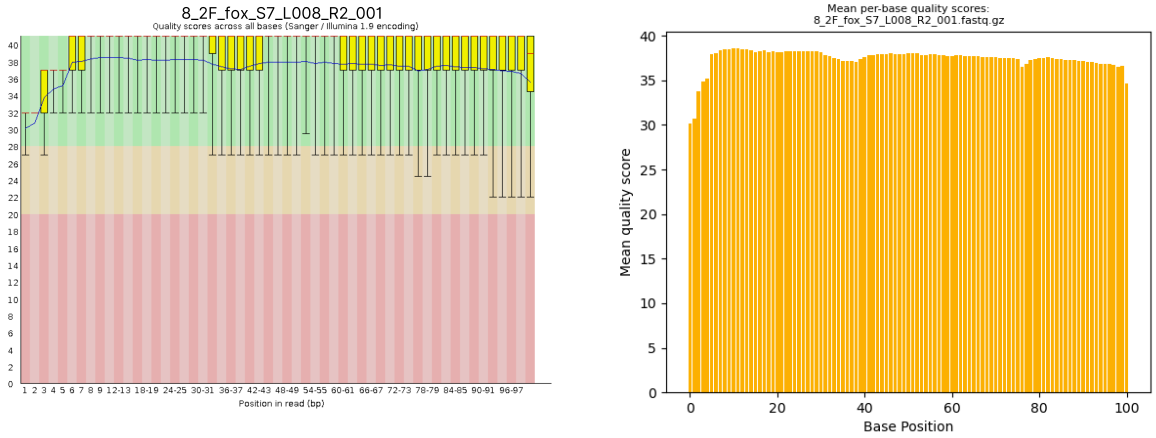

Figure 7: Comparison of per-base mean quality score plots for 8_2F_fox_S7_L008_R2.FastQC (left) vs. qs_dist.py (right).

Once again, both of the plots generated for read two of the 8_2F_fox_S7_L008 sample are very similar. Like the plot for read two of 14_3B_control_S10_L008, there is an increase in mean quality score between base positions one and seven, and decreases in quality score between base positions 32 and 42 and 76-79. Both plots also reflect that the last base position has a lower mean quality score. The FASTQC plot, which again contains information beyond mean quality score, shows that the interquartile range is not equal to zero for all base positions greater than 60, meaning there is more variation in quality score.

Resource usage comparisons:

Table 1: Comparison of resource usage between FastQC and my python script (qs_dist.py) across all files

File	Method	Runtime (s)	Memory usage (KB)	CPU usage
14_3B_control_S10 R1	FastQC	22.32	188560	92%
14_3B_control_S10 R1	qs_dist.py	80.66	61284	89%
14_3B_control_S10 R2	FastQC	23.05	169048	95%
14_3B_control_S10 R2	qs_dist.py	80.37	61152	99%
8_2F_fox_S7 R1	FastQC	163.11	191912	98%
8_2F_fox_S7 R1	qs_dist.py	638.09	61496	99%
8_2F_fox_S7 R2	FastQC	170.41	191452	99%
8_2F_fox_S7 R2	qs_dist.py	645.98	63296	99%

As shown in the above table, FastQC runs much faster than `qs_dist.py`; for each file, FastQC's runtime was around a quarter of `qs_dist.py`'s runtime. This is likely due in part to the fact that FastQC uses a significantly greater amount of memory compared to `qs_dist.py`. Across all files, the memory used by FastQC is around triple that of the memory used by `qs_dist.py`. It makes sense that FastQC would require more memory than `qs_dist.py` as FastQC computes many other statistics beyond the per-base mean quality score reported by `qs_dist.py`.

Overall Quality:

In my opinion, the FastQC reports for all four samples indicate an overall data quality that is sufficient for use in downstream analysis. As discussed above, per-base sequence quality is satisfactory for all files, with no base position in any file having a mean quality score of less than 30 (Figure 1). Though R2 files have poorer per-base sequence quality than R1 files, this is expected due to the fact that R2 is the last read to be sequenced, and thus there is the potential for cDNA and/or reagent degradation. Per-base N content, sequence length distribution, and adapter content plots all look as expected (Figure 2). Per-sequence mean quality score distributions, which are extremely right-skewed and unimodal for all files, are similarly unproblematic.

Read 1 files and Read 2 files have errors and warnings, respectively, associated with their per-tile sequence quality (Figure 8). However, it appears that tiles 2219-2222 and 2201-2204 have particularly low average sequence qualities, potentially indicating an error with the flow cell itself in these areas. Per the FastQC documentation, it is acceptable to proceed with data such as this, where only a few tiles are affected.

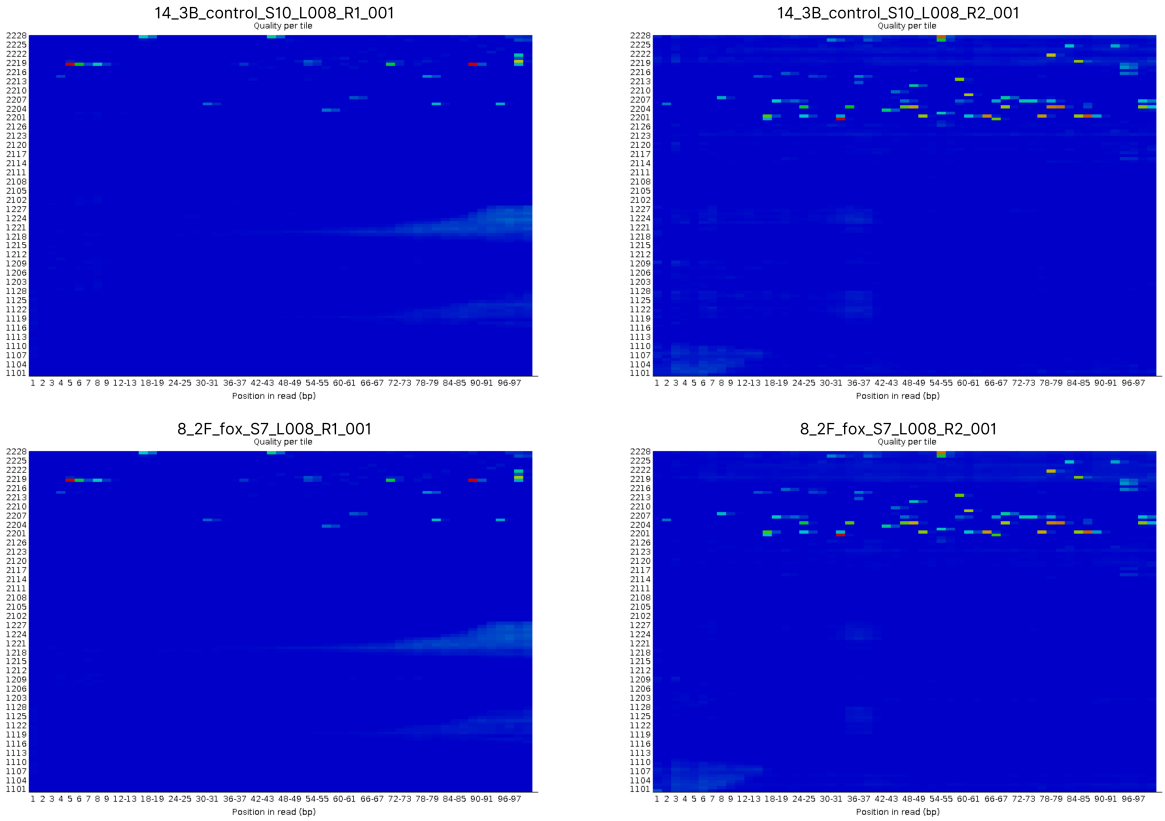

Figure 8: Per-tile sequence quality plots generated using FastQC

Though per-base sequence content is poor and erratic at the beginning of each read, this is not worrisome (Figure 9). This phenomenon is likely due to overall lower sequence quality at the beginning of all reads and priming bias, which is an artifact of library preparation. Since this is RNA sequencing data, and we expect to see many copies of some transcripts, the moderate to high sequence duplication levels observed in three of these files is expected and acceptable.

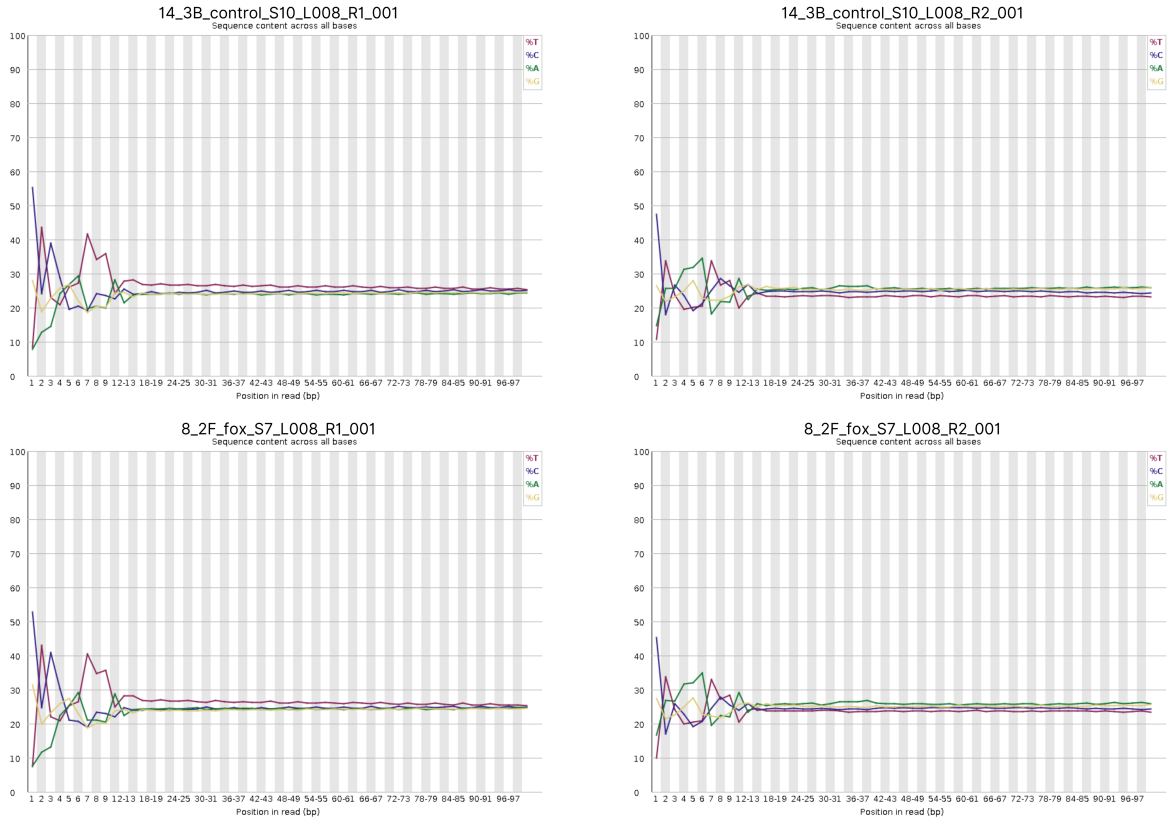

Figure 9: Per-base sequence content plots generated using FastQC. Across also samples, the plots indicate low quality at the beginning of reads.

While per-sequence GC content is flagged for Read 1 of `14_3B_control_S10_L008`, this is likely not cause for concern as there are no jagged peaks indicative of contamination (Figure 10). For these reasons, I feel that it is safe to continue with the analysis in the following sections of this report.

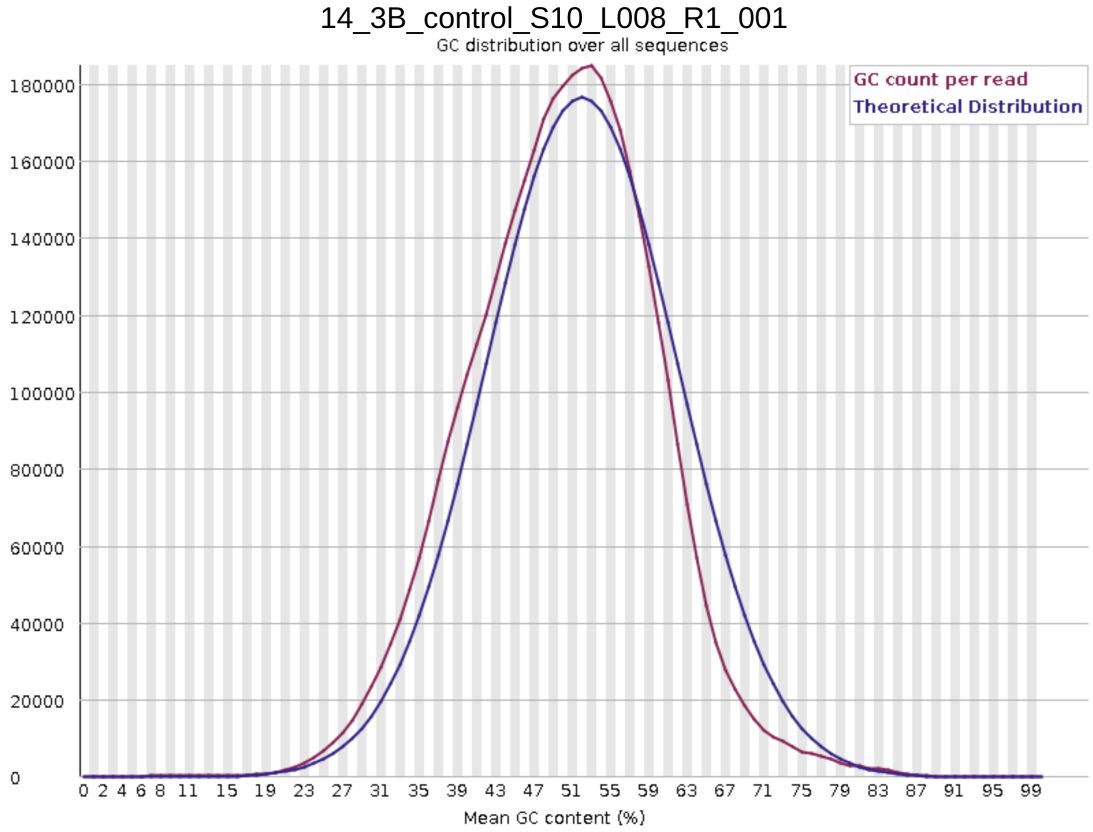

Figure 10: GC distribution across all reads in the R1 file of 14_3B_control_S10_L008. GC count per reads does not exactly fit the theoretical distribution.

PART TWO: Adapter trimming comparison

Cutadapt

Adapter sequences:

As shown in the below plots of adapter content, the only adapter sequence present in the reads is derived from Illumina Universal Adapters. Based on the FastQC github, this adapter content is measured using only a 12 base pair region of the Illumina Universal Adapter: `AGATCGGAAGAG`. However, according to the Illumina website, the full-length adapters are as follows:

R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA

R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

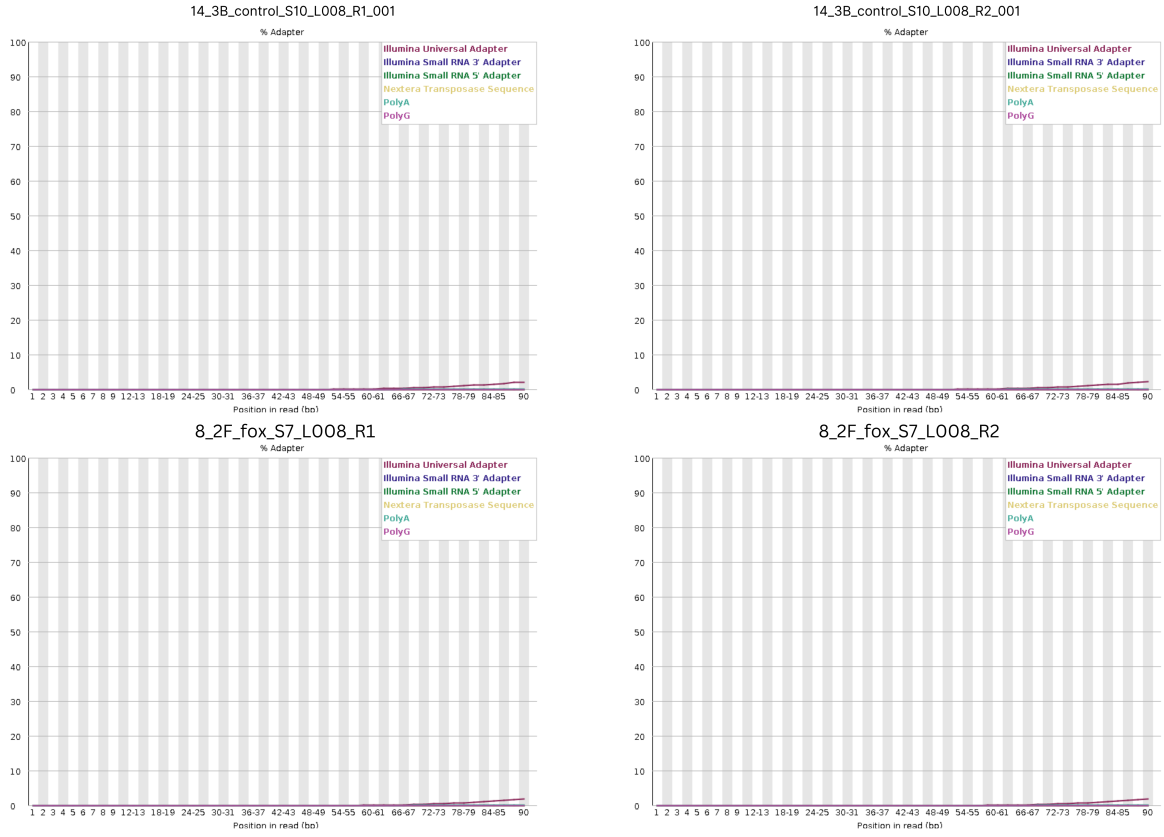

Figure 11: Adapter content plots generated using FastQC shows minimal Illumina Universal Adapter content towards the end of reads for all files.

Adapter content prior to trimming:

The following bash commands were used to directly evaluate where adapter sequences were found within the reads:

```
#Read 1 files
zcat 14_3B_control_S10_L008_R1_001.fastq.gz | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"

zcat 8_2F_fox_S7_L008_R1_001.fastq.gz | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"


#Read 2 files
zcat 14_3B_control_S10_L008_R2_001.fastq.gz | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

zcat 8_2F_fox_S7_L008_R2_001.fastq.gz | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"
```

For all files, adapter sequences were present at or towards the end of many reads. This is consistent with adapter content graphs generated by FastQC (Figure 11). However, using `grep` to search all reads for the **entire adapter sequence** only gives an idea of how many reads contained the **entire adapter**. We can assume that partial adapter sequences are present at the end of many more reads beyond those identified with these commands.

Each file was additonally searched for the reverse complements of both entire adapters, using similar grep commands. No reverse complements of either entire adapter were identified in any read.

Adapter trimming:

Table 2: Percent of reads (for each file) from which the Illumina Universal Adapter sequences were trimmed by Cutadapt.

	Read 1	Read 2
14_3B_control_S10_L008	6.0% reads trimmed	6.7% reads trimmed
8_2F_fox_S7_L008	5.9% reads trimmed	6.6% reads trimmed

Following adapter trimming, the above bash commands were sucessfully used to confirm that no adapter sequences remained in any reads across all files.

Trimmomatic

Post-Trimmomatic Read Length Distributions:

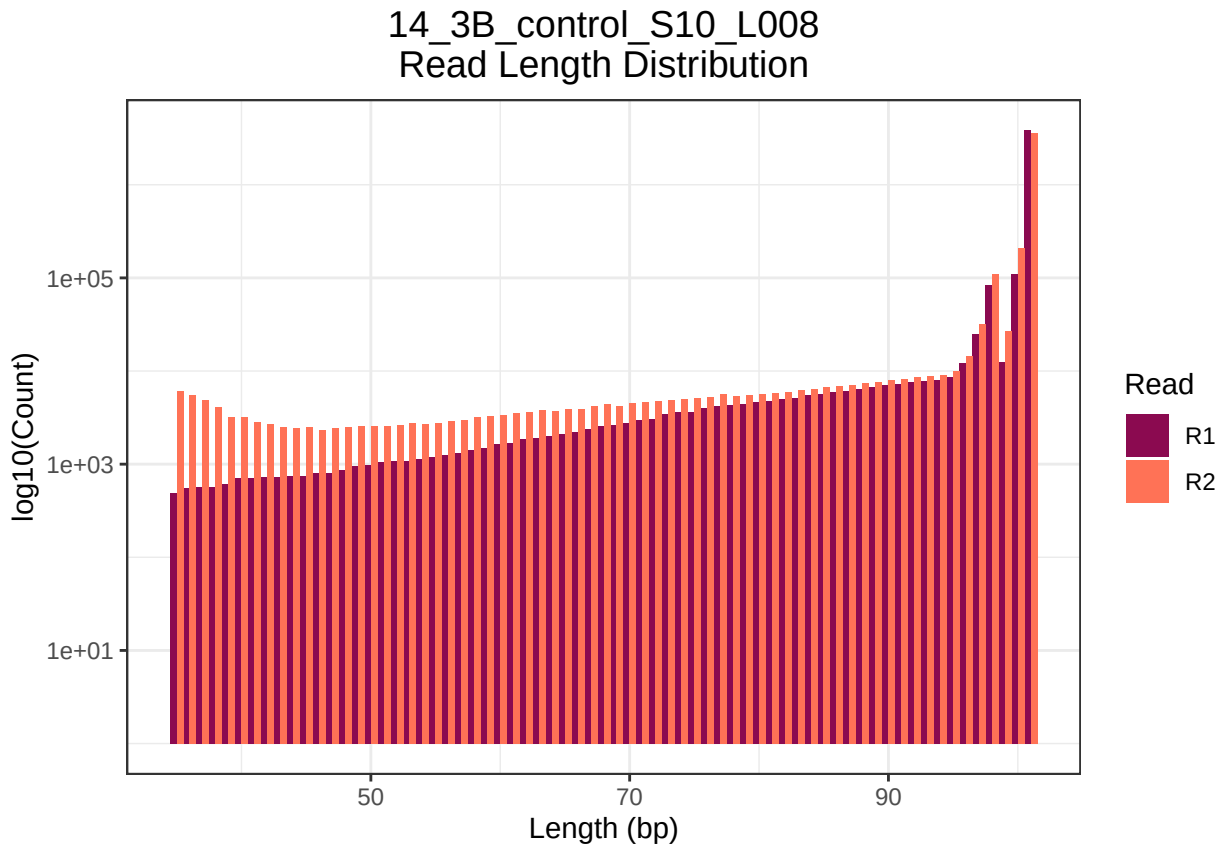

Figure 12: Distribution of read lengths in R1 and R2 files of sample 14_3B_control_S10_L008 after running Trimmomatic.

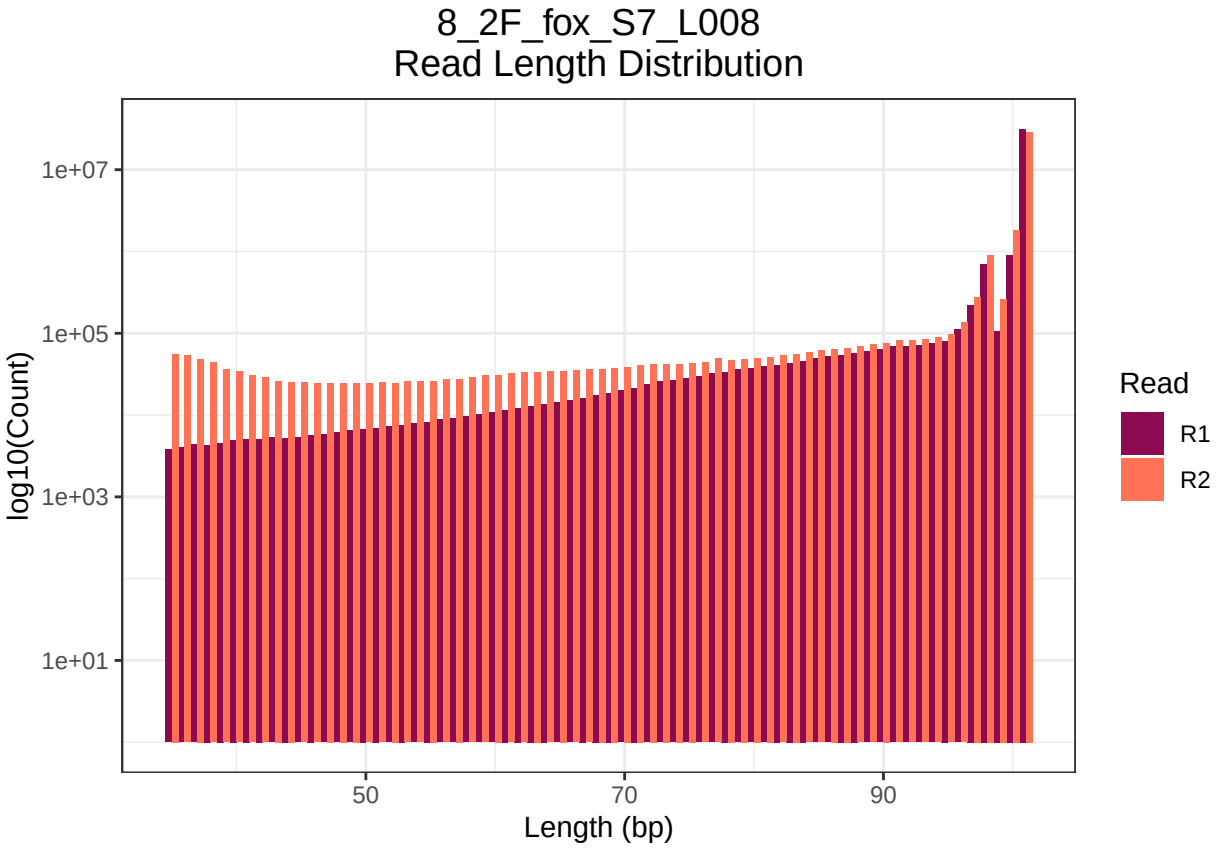

Figure 13: Distribution of read lengths in R1 and R2 files of sample 8_2F_fox_S7_L008 after running Trimmomatic.

Based on Figures 12 and 13, it is clear that `R2` reads are trimmed more extensively than `R1` reads for both samples. This is expected, as `R2` reads are the last to be sequenced and thus are typically of a lower quality than `R1` reads. This trend of reduced quality in Read 2 is reflected in Figures 12 and 13, where shorter/trimmed `R2` reads are more abundant than shorter/trimmed `R1` reads.

PART THREE: Alignment and strand-specificity

PS8 script - mapped vs. unmapped reads:

Table 3: Mapped vs. unmapped reads for each sample calculated using my python script from PS8 `mappedreads.py`

	mapped	unmapped	total	percentage mapped
14_3B_control_S10_L008	8312390	180914	8,493,304	97.87 %
8_2F_fox_S7_L008	67070894	2511420	69,582,314	96.39 %

`htseq-count` - mapped vs. unmapped reads:

Table 4: Mapped vs. unmapped reads identified for each sample using htseq-count and either the`--stranded=reverse` or `stranded=yes` options.

	`--stranded`	mapped reads	total reads	% mapped reads
8_2F_fox_S7_L008	reverse	28,037,914	34,791,157	80.5
8_2F_fox_S7_L008	yes	1,205,384	34,791,157	3.46
14_3B_control_S10_L008	reverse	3,666,607	4,246,652	86.34
14_3B_control_S10_L008	yes	161,502	4,246,652	3.8

Strand-specificity:

Based on the results outlined in Table 4, I propose that the library preparation for these samples *was* strand-specific. For the 14_3B_control_S10_L008 sample, 86.34% of reads mapped to a feature when run with the `--stranded = reverse` option, and only 3.8% of reads mapped to a feature when run with the `--stranded = yes` option. Similarly, for the 8_2F_fox_S7_L008 sample, 80.5% of reads mapped to a feature when run with the `--stranded = reverse` option, but only 3.46% of reads mapped to a feature when run with the `--stranded = yes` option. Because the percentage of reads mapped to features is significantly higher when run with `--stranded = reverse,` we can infer that these reads are reverse stranded, and that Read 2 aligns to the genomic DNA non-template/coding strand.