QAA_report

Rachel Ferina

2022-09-06

Summary

Data files: 22_3H_both_S16_L008, 23_4A_control_S17_L008; referred to as 22 and 23 in this report.

Python Version: Python 3.10.5

R Version: R 4.2.1

Environment: QAA

Part 1

FastQC Plots

22_3H read 1: There is only a small amount of N content for the first base, and the average quality score is lower at the first base, so these plots are consistent. It is also expected to have lower quality for the first base.

22_3H read 2: There is only a small amount of N content for the first base, and the average quality score is again lower at the first base as expected, so these graphs are consistent. The quality also declines at the end a bit, which is expected at the end of sequencing.

23_4A read 1: These plots are consistent with the low N content and low quality at the first base. Quality also declines towards the end, which is expected at the end of sequencing.

23_4A read 2: Again, these plots have low N content and low quality at the first base. Quality declines towards the end, which is expected at the end of sequencing.

FastQC VS Demultiplexing Algorithm

As similar to the FastQC graphs, there is a lower quality score for the first few bases. However, a decline in quality at the end isn't as noticeable as it is in the FastQC plots.

The runtime differed for the quality score distributions for FastQC and my demultiplexing algorithm. My demultiplexing program took 1 minute and 10 seconds for the 22_3H files, and 13 minutes for the 23_4A files, which makes sense as the 23_4A files are longer. The FASTQC program took 5 minutes to process all 4 files. The FASTQC program is likely more optimized than my code, so the runtime is faster.

My demultiplex program graphs show that the 22_3H read 1 and read 2 data all has average quality scores above 30. The 23_4A read 1 data also has all average quality scores above 30, but the read 2 data had the first base pair's average below 30, but the rest above 30.

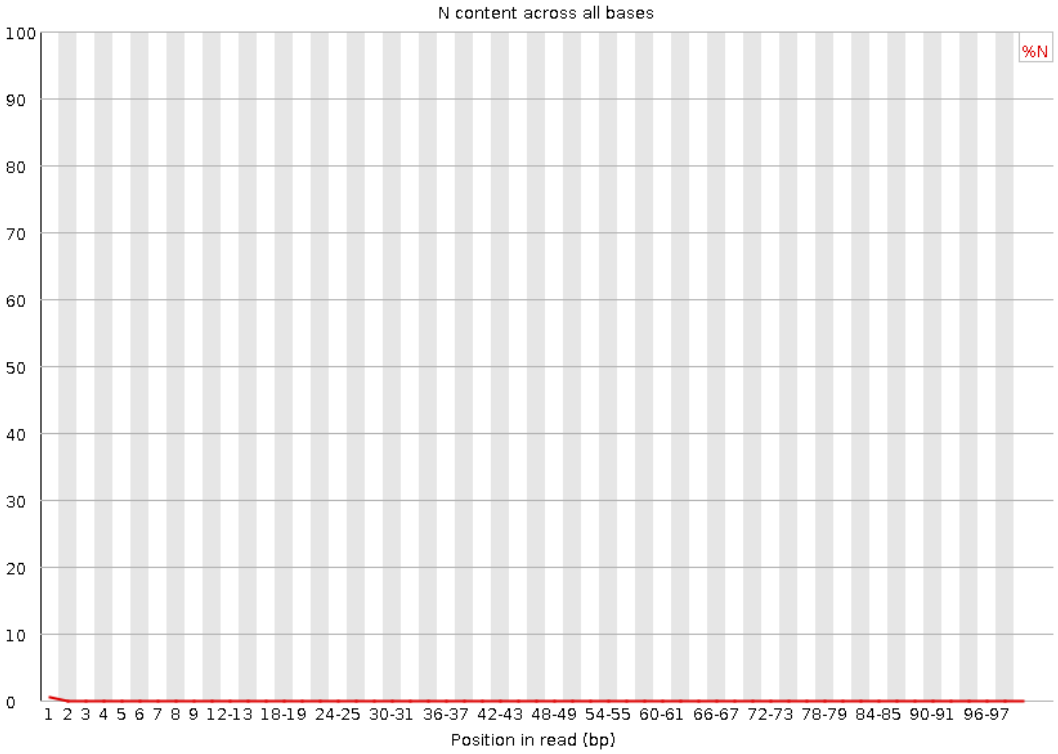

Figure 1: 22 Read 1 N Content Per Base shows small N content at the first base.

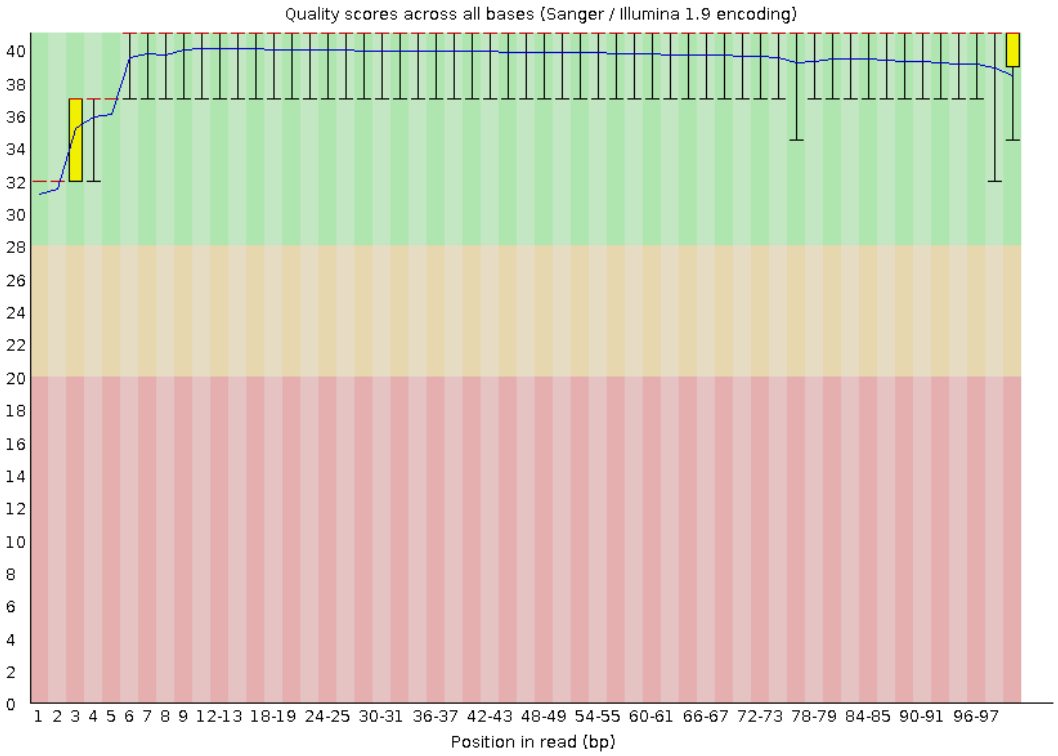

Figure 2: 22 Read 1 Quality Score Per Base shows lower quality at the first base.

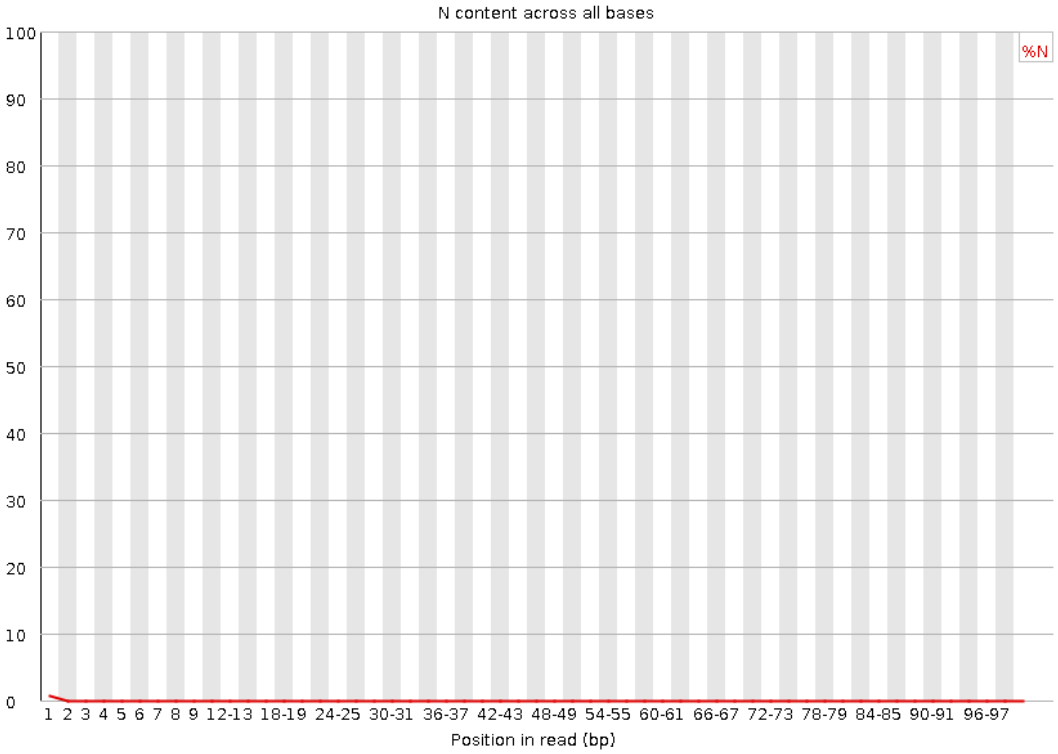

Figure 3: 22 Read 2 N Content Per Base shows N content at the first base.

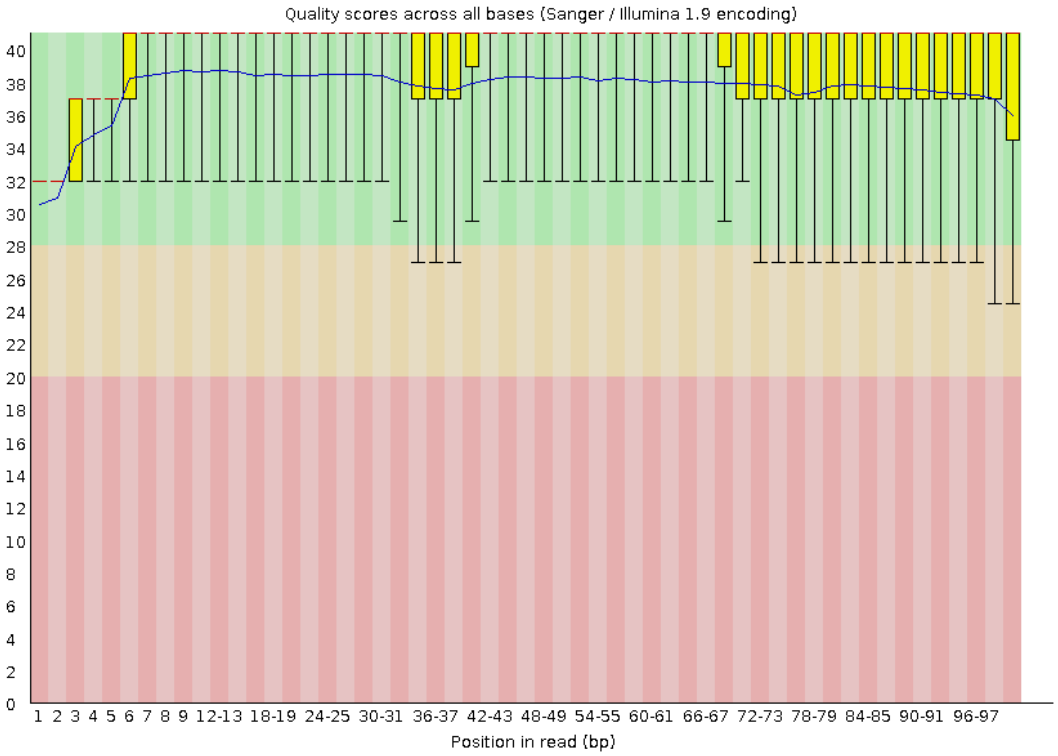

Figure 4: 22 Read 2 Quality Score Per Base shows lower quality at the first base.

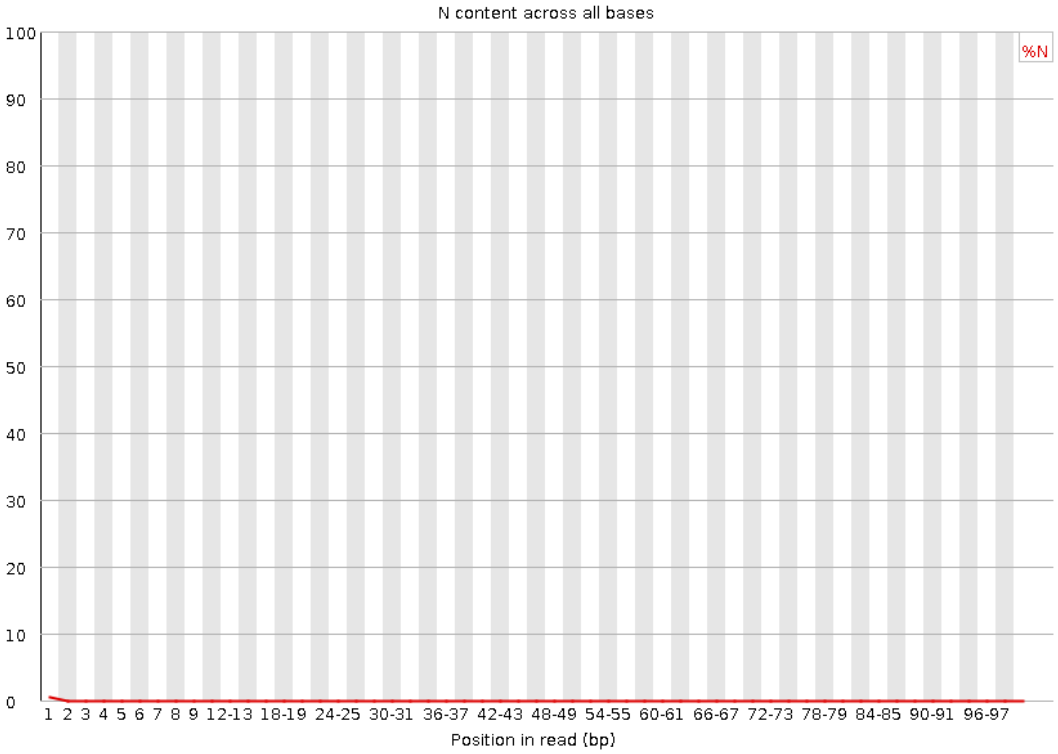

Figure 5: 23 Read 1 N Content Per Base shows N content at the first base.

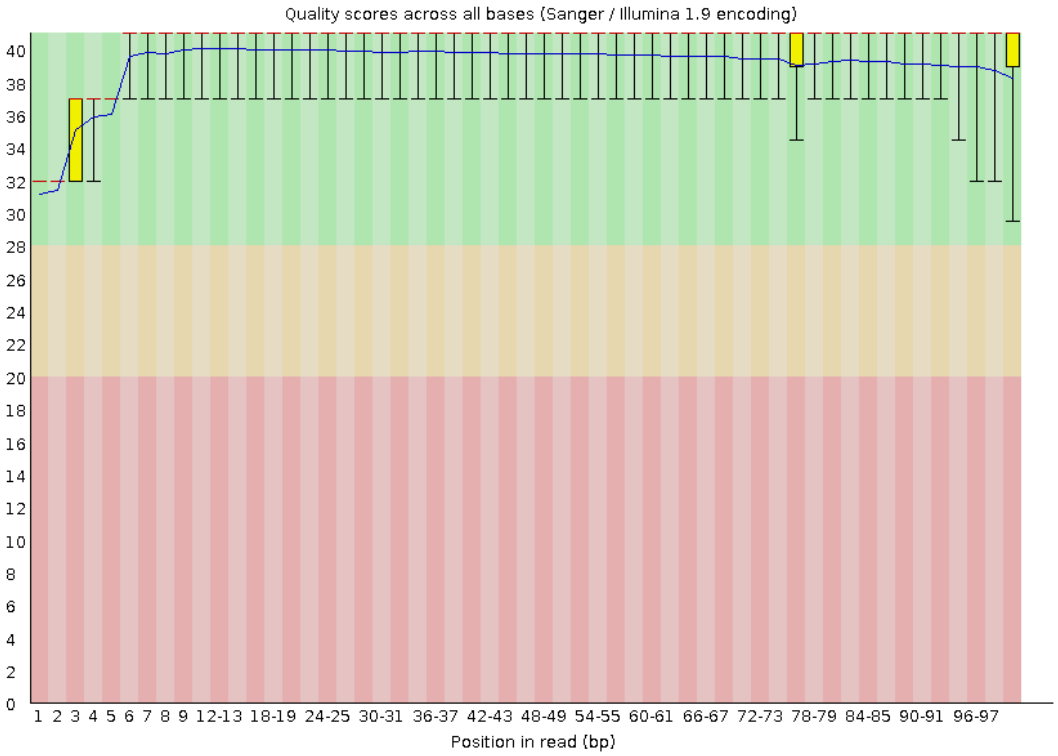

Figure 6: Read 1 Quality Score Per Base shows lower quality at the first base.

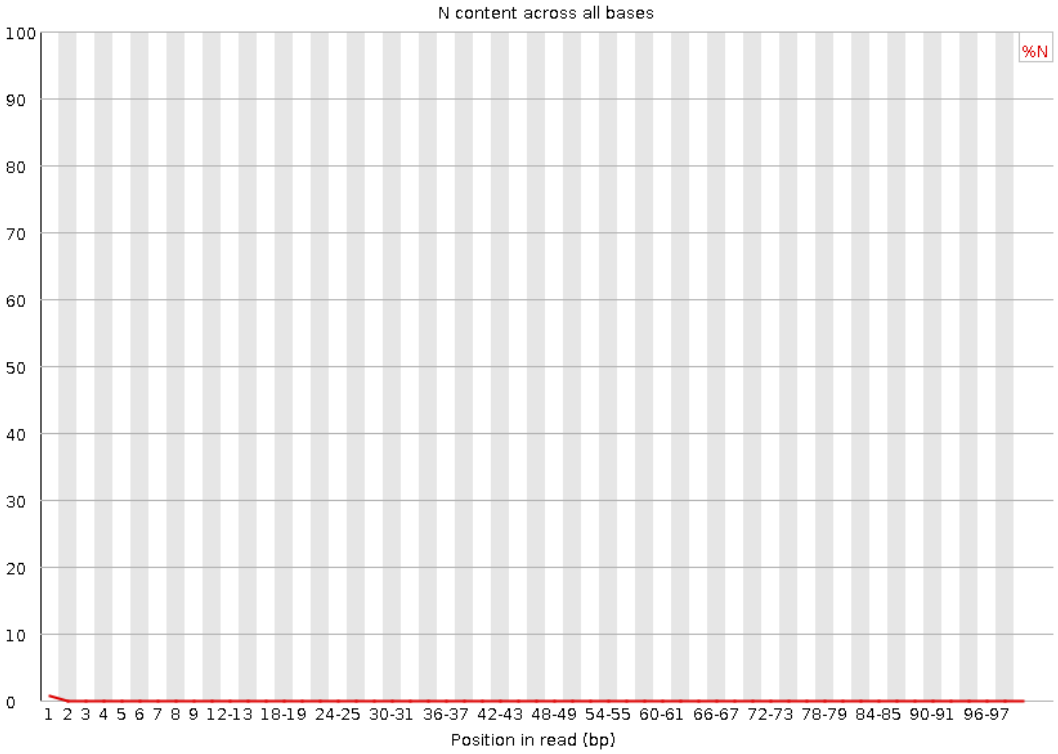

Figure 7: Read 2 N Content Per Base shows N content at the first base.

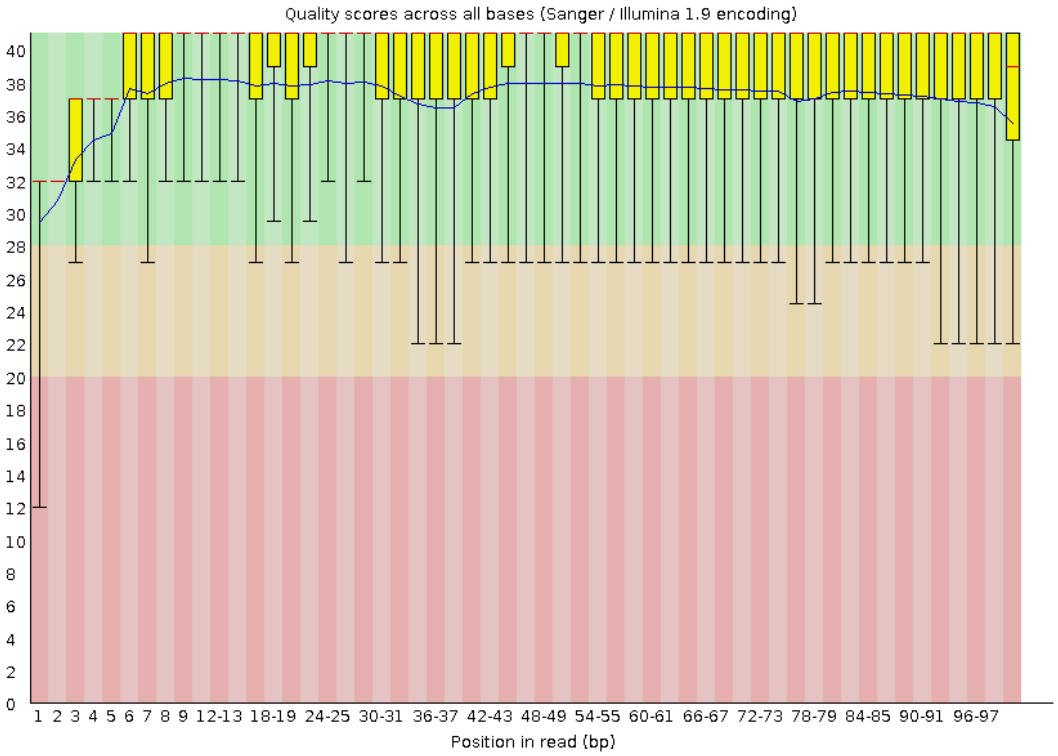

Figure 8: Read 2 Quality Score Per Base shows lower quality at the first base.

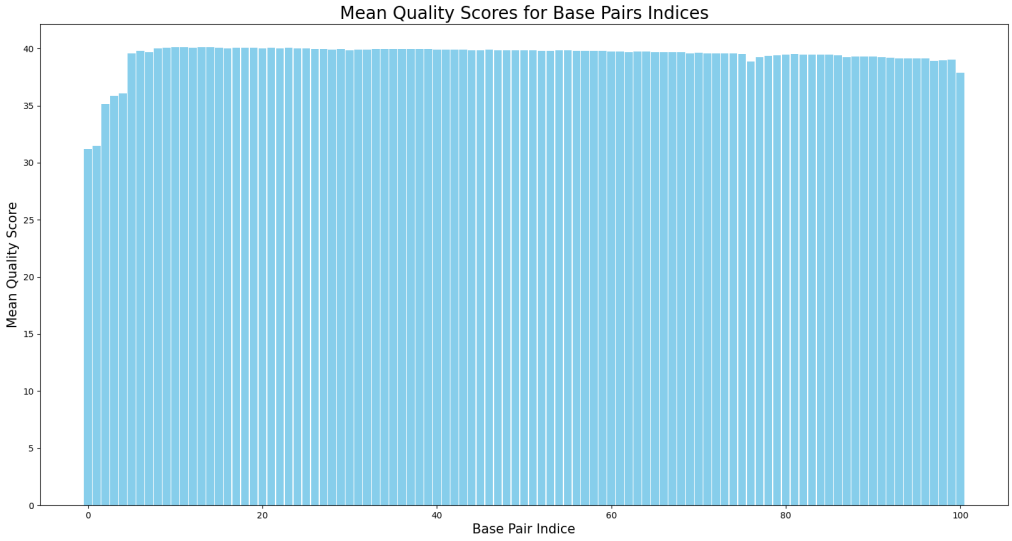

Figure 9: 22 read 1 distribution shows low quality at the beginning as expected.

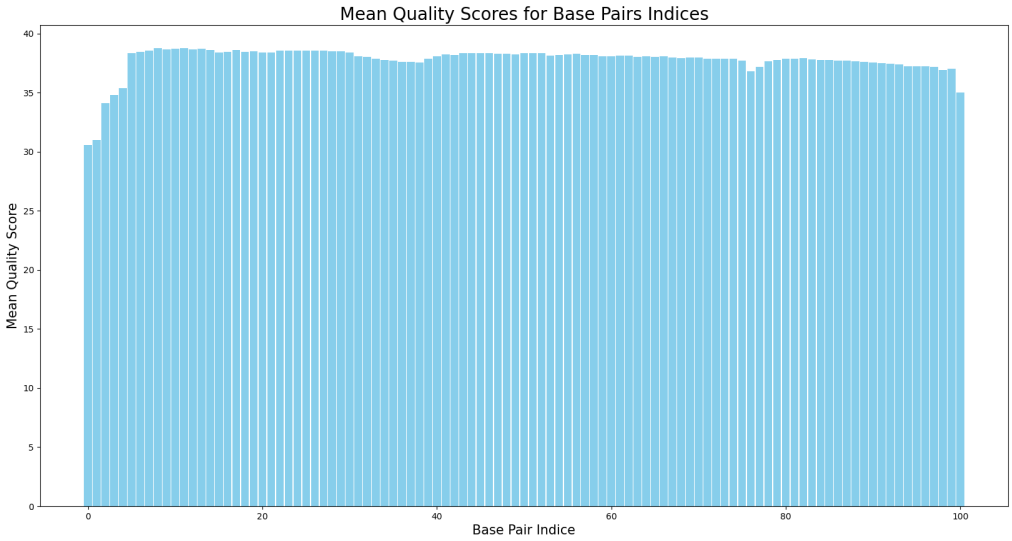

Figure 10: 22 read 2 distribution shows lower quality at the earlier bases as expected.

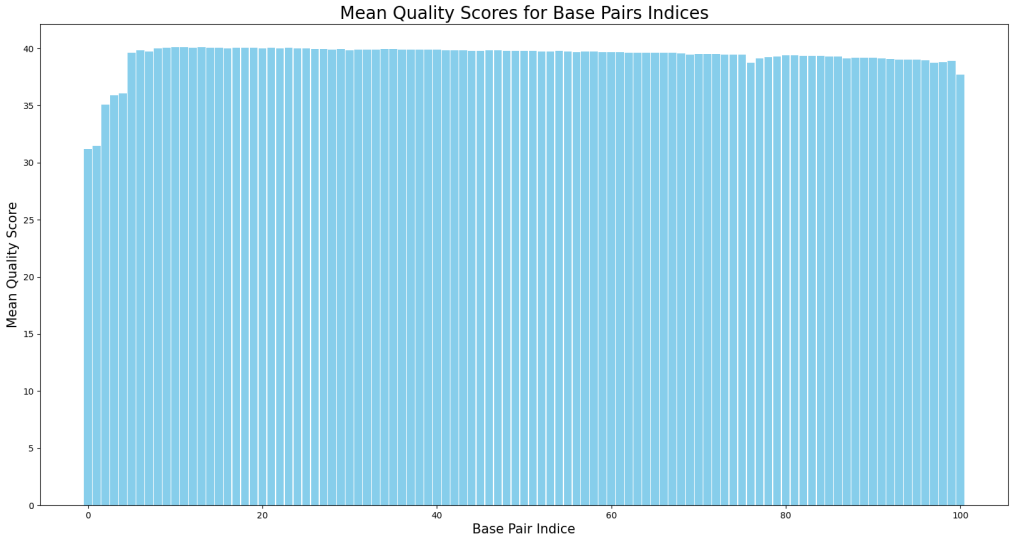

Figure 11: 23 read 1 distribution shows lower quality at the beginning of the bases, as anticipated.

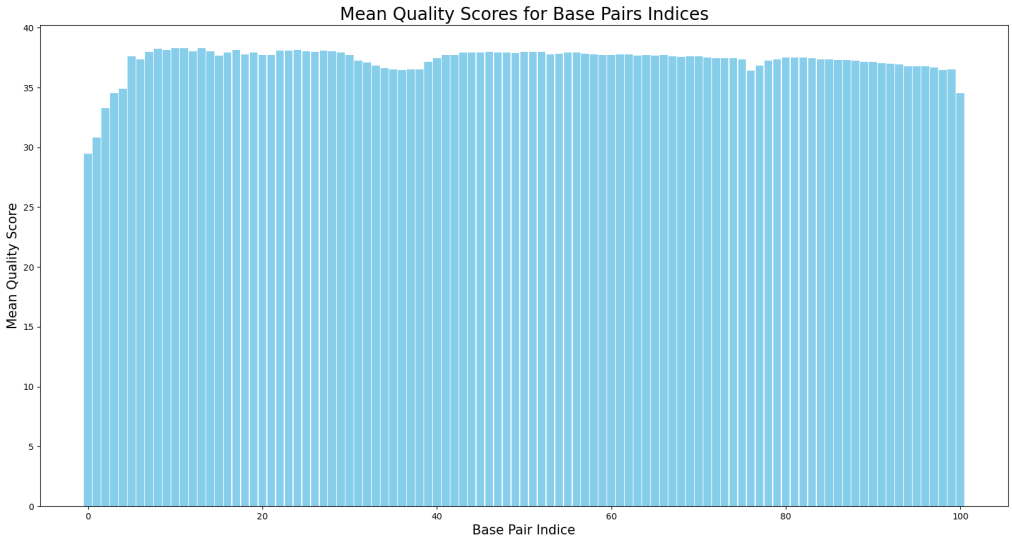

Figure 12: 23 read 2 distribution shows low quality at the earlier bases as expected.

Part 2

Proportion of Trimmed Reads

22

R1: 153089 / (954639 / 4) * 100 = 64.15% R2: 186534 / (1086026 / 4) * 100 = 68.70%

23

R1: 1359563 / (10058232 / 4) * 100 = 54.07% R2: 1657134 / (10058232 / 4) * 100 = 65.90%

Adapter Content

I examined the adapter content graphs, and observed presence of the Illumina Universal adapter. I also checked overrepresented sequences in the fastqc_data files, however they didn't match adapter sequences. I confirmed the Illumina Universal adapter sequences by zgrepping for them in the original data files. The read 1 files had the read 1 adapter as expected, and the read 2 files had the read 2 adapter in them as expected.

22 R1:

Read 1 adapter:

```
zgrep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" 22_3H_both_S16_L008_R1_001.fastq.gz
```

7563

Read 2 adapter:

```
zgrep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" 22_3H_both_S16_L008_R1_001.fastq.gz
```

0

22 R2:

Read 1:

```
zgrep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" 22_3H_both_S16_L008_R2_001.fastq.gz
```

0

Read 2:

```
zgrep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" 22_3H_both_S16_L008_R2_001.fastq.gz
```

7848

23 R1:

Read 1:

```
zgrep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" 23_4A_control_S17_L008_R1_001.fastq.gz
```

43466

Read 2:

```
zgrep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" 23_4A_control_S17_L008_R1_001.fastq.gz
```

0

23 R2

Read 1:

```
zgrep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" 23_4A_control_S17_L008_R2_001.fastq.gz
```

0

Read 2:

```
zgrep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" 23_4A_control_S17_L008_R2_001.fastq.gz
```

46541

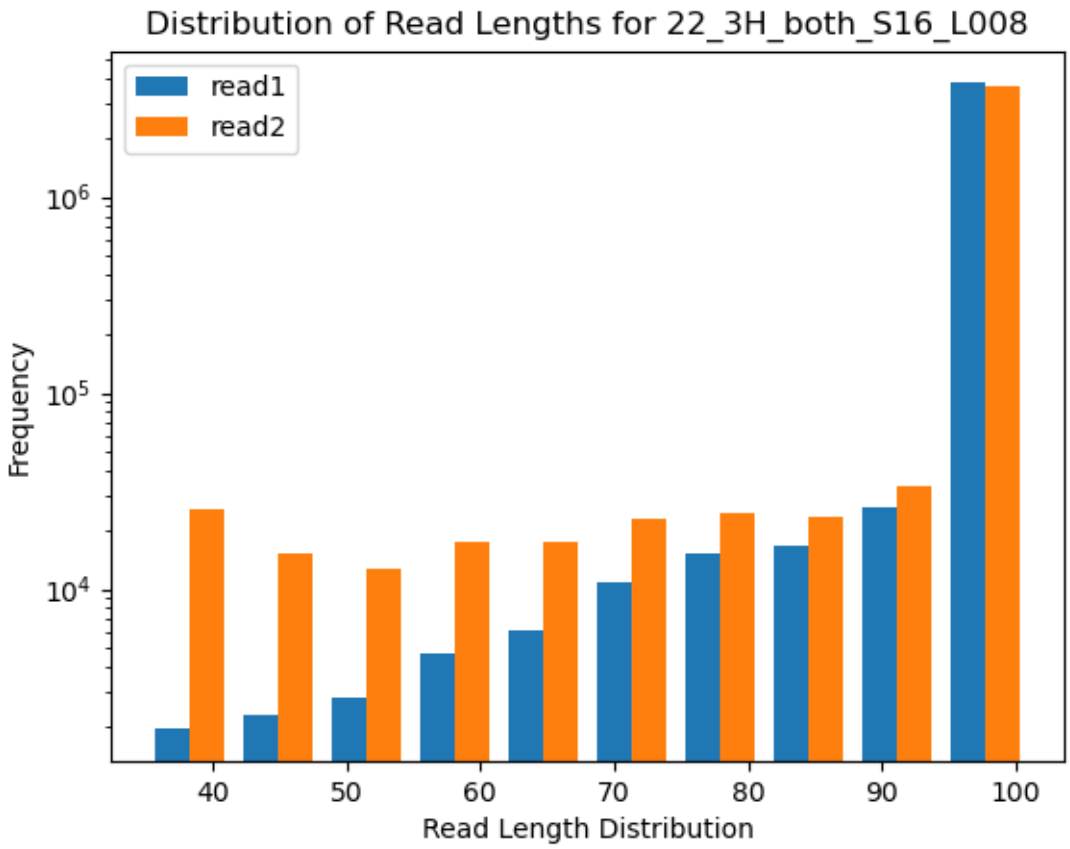

Figure 13: 22 read length distribution shows read 1 has a fewer short reads compared to read 2. Both are left skewed distributions.

I would expect read 1 and read 2 to have different adapter-trimming rates because read 2 is performed second, and there's more time for it to degrade. This may result in less adapter trimming if more ends of the read 2's are degraded. However, my percentages indicated that read 2 had more adapter trimming for both files. This could be because read 2 inserts could be shorter due to degradation, resulting in more adapters being present in the sequence that needed to be trimmed. Another possible explanation is an excess of adapters would have more time to bind to read 2, so more adapters would need to be trimmed.

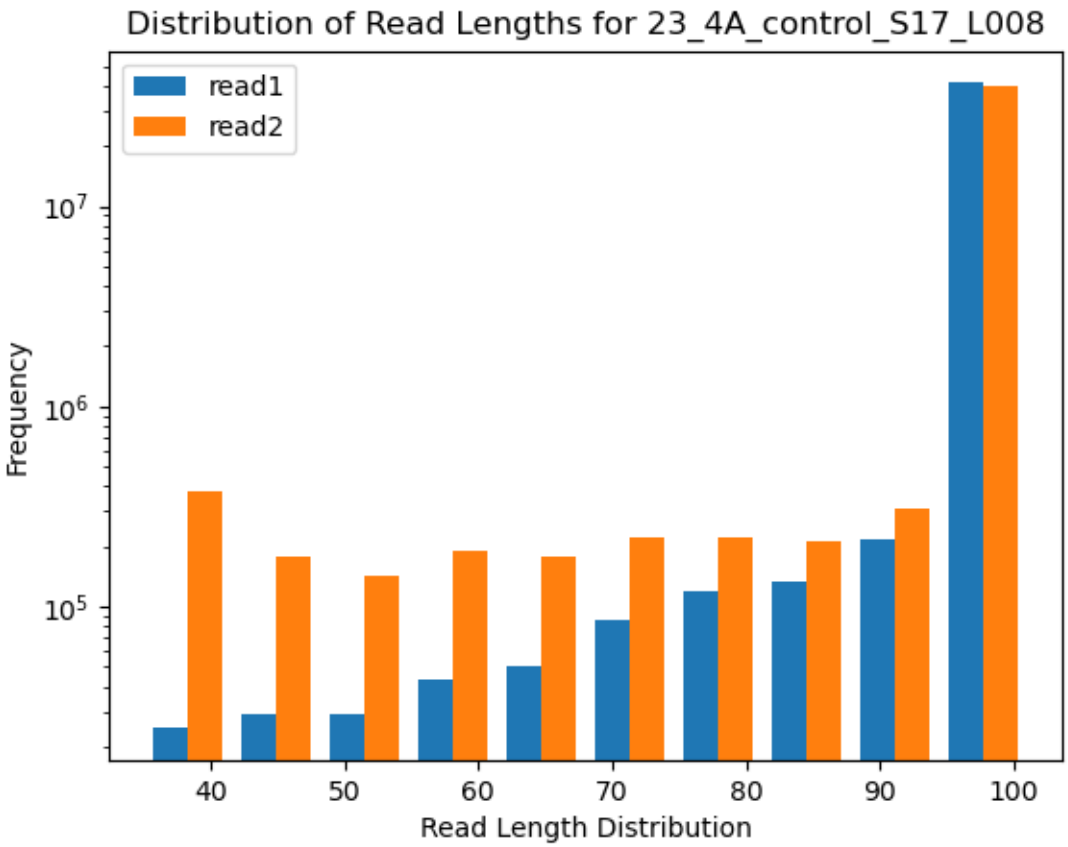

Figure 14: 22 read length distribution shows read 1 has a fewer short reads opposed to read 2. Both are left skewed distributions.

Part 3

Sam File: 22align_to_refAligned.out.sam

Mapped: 7677920 Unmapped: 124334

Sam File: 23align_to_refAligned.out.sam

Mapped: 79473028 Unmapped: 4640124

Determining Strand Specificity

Number of Mapped Reads:

```
cat 22stranded_count.tsv | grep -v "^__" | awk '{sum+=$2} END {print sum}'
```

148785

```
cat 22stranded_reverse_count.tsv | grep -v "^__" | awk '{sum+=$2} END {print sum}'
```

3394239

```
cat 23stranded_count.tsv | grep -v "^__" | awk '{sum+=$2} END {print sum}'
```

1575358

```
cat 23stranded_reverse_count.tsv | grep -v "^__" | awk '{sum+=$2} END {print sum}'
```

32951139

Total Reads:

```
cat 22stranded_count.tsv | awk '{sum+=$2} END {print sum}'
```

3901127

```
cat 22stranded_reverse_count.tsv | awk '{sum+=$2} END {print sum}'
```

3901127

```
cat 23stranded_count.tsv | awk '{sum+=$2} END {print sum}'
```

42056576

```
cat 23stranded_reverse_count.tsv | awk '{sum+=$2} END {print sum}'
```

42056576

Percentages of Mapped Reads

22 stranded percentage: 3.81% 22 reverse percentage: 87.01%

23 stranded percentage: 3.75% 23 reverse percentage: 78.35%

There is a large difference in the percentages of mapped reads between the stranded and reverse htseq-count for both the 22_3H_both_S16_L008 and 23_4A_control_S17_L008 files. This suggests that the library prep was stranded. If the library prep was unstranded, the percentages for htseq-count stranded and reverse would be about the same, because there would be a 50-50 chance of a gene aligning to either strand. The

difference in percentages indicates strandedness, because it matters which strand the gene aligns to–in this case, significantly more reads are mapped to the reverse strand. In summary, for 22_3H_both_S16_L008, I propose that the library prep was stranded because only 3.81% mapped with stranded htseq-count, while 87.01% mapped with reverse htseq-count, and this difference in percentage indicates that which strand a gene maps to matters. For 23_4A_control_S17_L008, I propose that the library prep was stranded because of the difference in percentages–3.75% mapped with stranded htseq-count, while 78.35% mapped with reverse htseq-count.

Summary Tables

Sam File Results

File	Mapped Read Counts	Unmapped Read Counts
22_3H_both_S16_L008	7677920	124334
23_4A_control_S17_L008	79473028	4640124

Htseq-count Results

File	Total Reads	Reads Mapped to Feature Counts	Percentages of Mapped Feature Reads (%)
22_3H_both_S16_L008 stranded	3901127	148785	3.81
22_3H_both_S16_L008 reverse	3901127	3394239	87.01
23_4A_control_S17_L008 stranded	42056576	1575358	3.75
23_4A_control_S17_L008 reverse	42056576	32951139	78.35

For more detailed information on running the programs, see the lab notebook.