RNA-seq quality assessment (QAA)

Ruben Lancaster

2023-09-12

RNA-seq quality assessment for samples 27 and 28

Sample 27 and 28 from the 2017 RNA-seq BGMP dataset (available on Talapas at the below paths) were assessed for quality.

```
/projects/bgmp/shared/2017_sequencing/demultiplexed/
```

Filenames:

```
27_4C_mbnl_S19_L008_R1_001.fastq.gz
27_4C_mbnl_S19_L008_R2_001.fastq.gz
28_4D_mbnl_S20_L008_R1_001.fastq.gz
28_4D_mbnl_S20_L008_R2_001.fastq.gz
```

Data description These libraries are from mouse embryonic fibroblasts treated with MBNL and prepared using the KAPA HiFi stranded mRNA kit. Libraries were run on an Illumina HiSeq 4000, and data are 101 base pair paired-end reads.

Report overview In this report, I first assessed libraries for quality using FastQC and a custom script called qscore_dist_2.py. Then, I trimmed adapters with cutadapt and quality trimmed reads with trimmomatic. Finally, I aligned reads against the mouse genome (Mus Ensembl release 110) using a splice-aware aligner, STAR, and counted up reads that mapped to features using a custom script and htseq-count.

Overall, I recommend that these data are of sufficient quality to proceed with further analyses.

Part 1: Read Quality Score Distributions

FastQC per-base quality score distributions

Per-base quality scores were assessed using FastQC version 0.11.5. Overall per-base read quality scores are good (above 30). Even the lower mean quality scores at the beginning of the read for both read 1 and read 2 of each sample are acceptable ($mean \approx 30$).

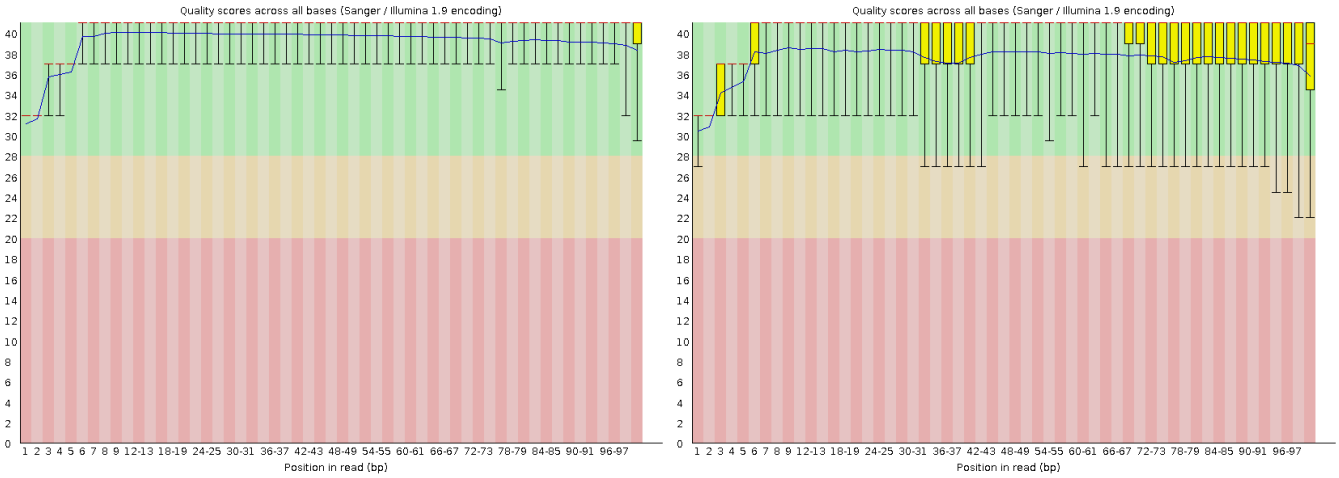

Figure 1: Sample 27 per-base quality scores, read 1 (left), read 2 (right). Mean is shown in blue. Box plots for each nucleotide bin are shown; box plots are structured by lowest 10 percent, lower quartile, median (red line), upper quartile, and upper 90 percent. Box plots are filled with yellow for visibility.

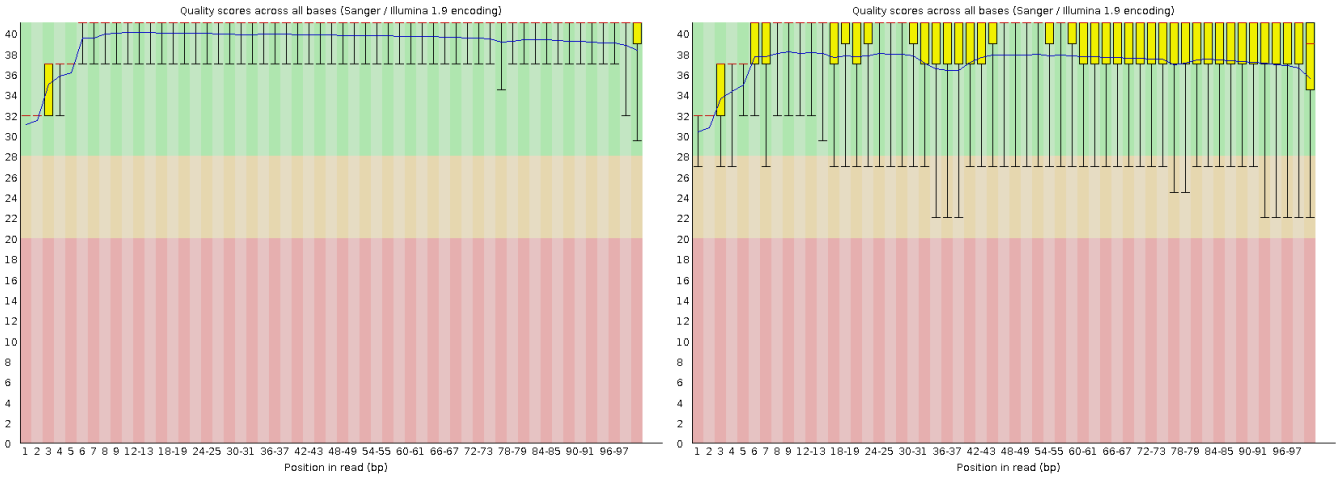

Figure 2: Sample 28 per-base quality scores, read 1 (left), read 2 (right). FastQC graph displayed as described in Figure 1.

FastQC per-base N content

Per-base N content was assessed using FastQC. The figures show that the N content and the quality score plots are consistent with each other; both show a drop in quality at the beginning of the read.

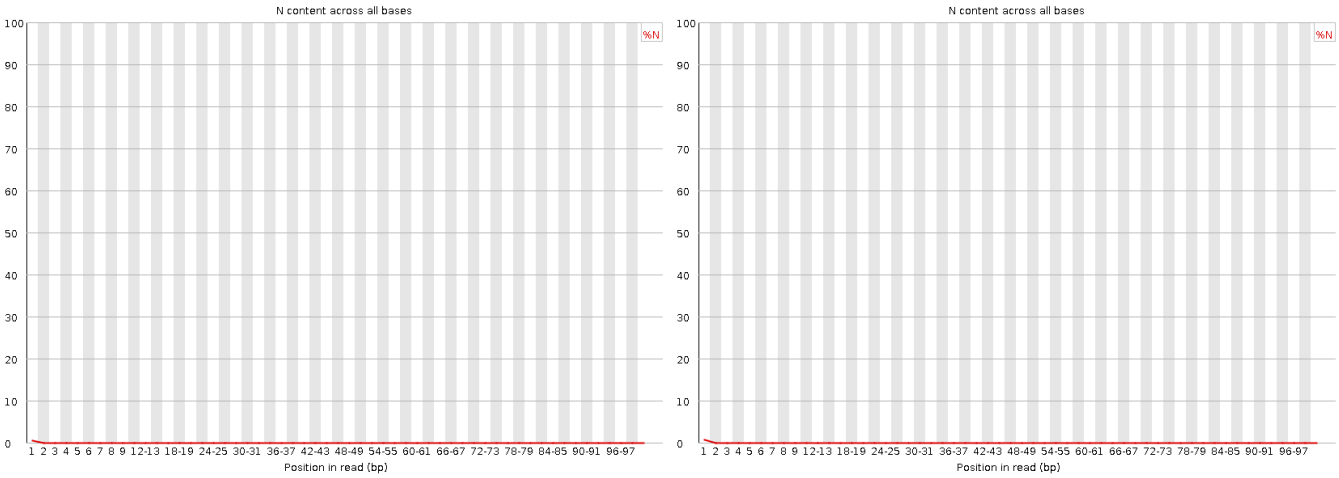

Figure 3: Sample 27 per-base N content, read 1 (left), read 2 (right). N content denoted as a percent of all bases at that position.

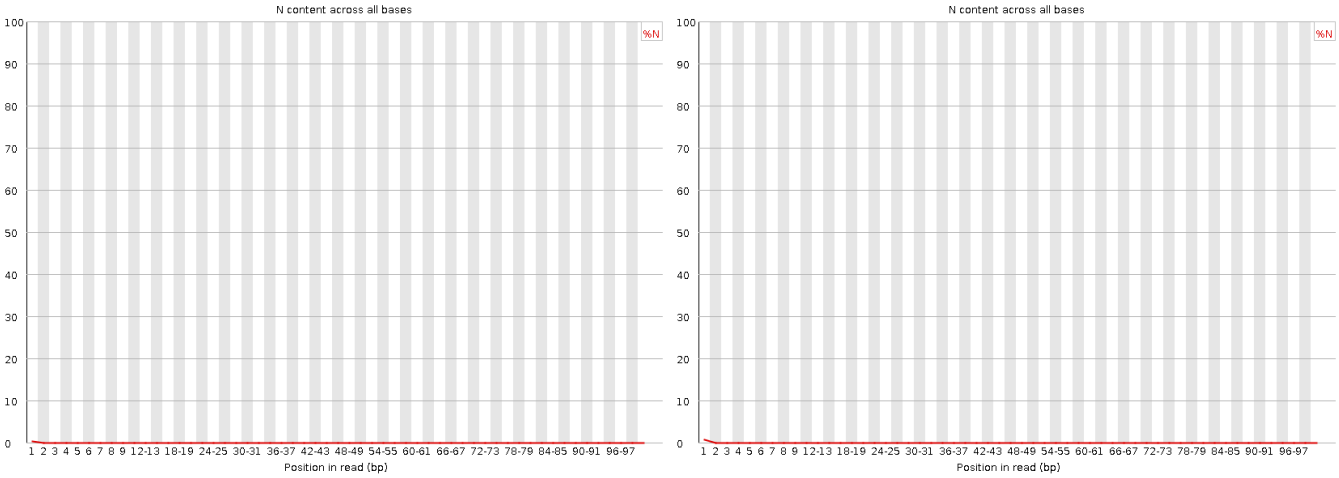

Figure 4: Sample 28 per-base N content, read 1 (left), read 2 (right). N content denoted as a percent of all bases at that position.

Custom script per-base quality score distributions

I confirmed the output of FastQC using my custom quality score plotting script qscore_dist_2.py. My plots and the FastQC plots show very similar results. My quality score distribution plots lack the standard deviation and median reported by FastQC plots.

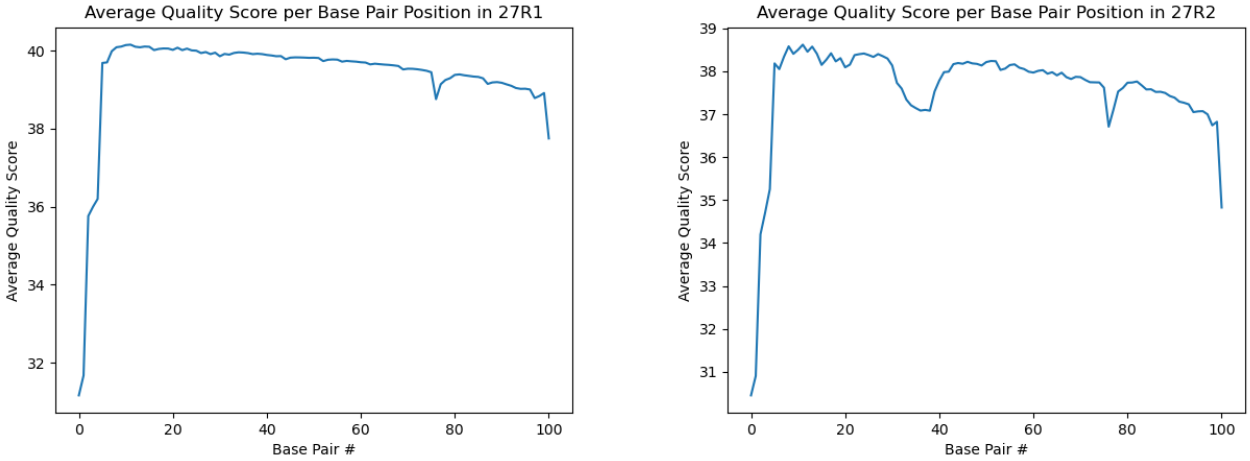

Figure 5: Sample 27 per-base quality score, read 1 (left), read 2 (right). Mean per-base quality score: blue.

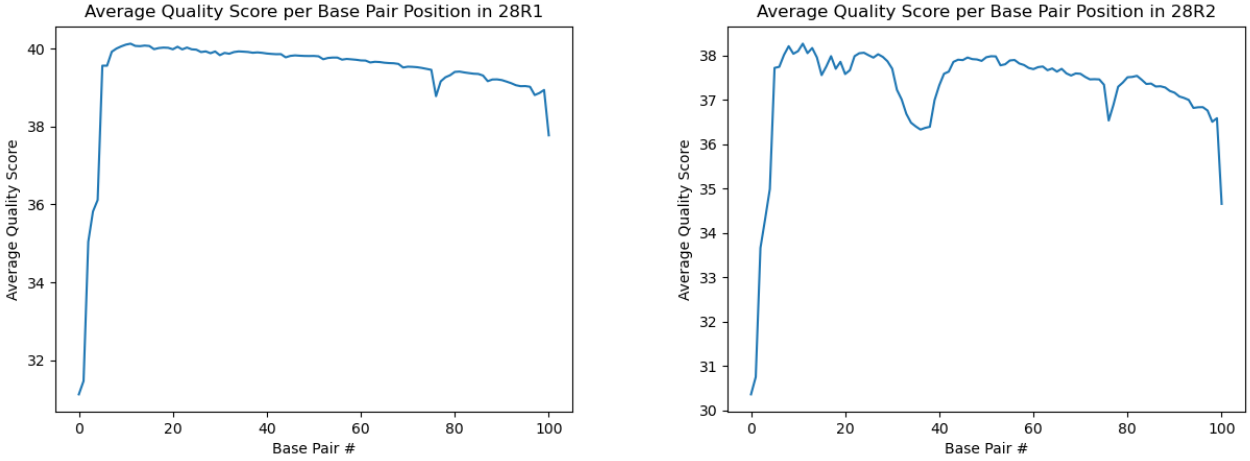

Figure 6: Sample 28 per-base quality score, read 1 (left), read 2 (right). Mean per-base quality score: blue.

FastQC and qscore_dist_2.py comparison In terms of runtime, FastQC was much faster than my custom Python script.

Runtime for my Python script:

Table 1: Runtime comparisons (m:ss)

	qscore_dist_2.py	FastQC
Sample 27 R1	2:27.21	0:38.53
Sample 27 R2	2:25.26	0:38.18
Sample 28 R1	4:09.99	1:03.22
Sample 28 R2	4:03.80	1:04.16

FastQC is much faster than my own code; it is optimized in ways that my code is not.

Overall, the libraries for samples 27 and 28 are both acceptable for further analysis, with satisfactory results from the FastQC report for all expected quality metrics.

On a sidenote, all runs failed "per base sequence content" scoring, which is expected of RNA-seq data and is not worrisome. Because this is RNA-seq, there we see warnings for duplicated sequences from FastQC, which is expected if we have multiple copies of some transcripts. (These quality data are not shown).

Although minor, one thing to note is that per tile sequence quality plots showed that there was a decrease in quality for read 1 of both sample 27 and 28 in tiles 1221 through 1224 after about the 70th nucleotide. This decrease in quality is slight, and likely due to dust or a bubble on the flow cell. It should be noted in case there are issues in downstream analysis for these reads.

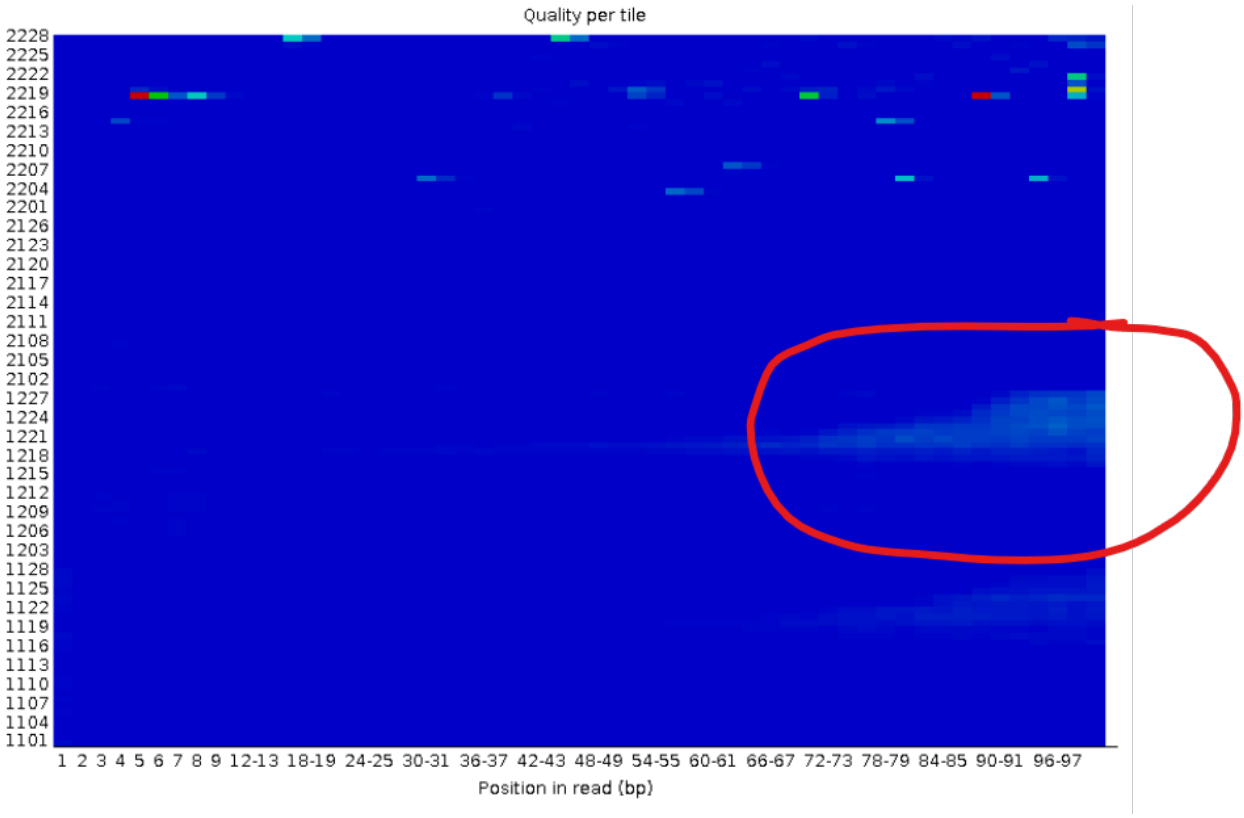

Figure 7: Sample 27 read 1 tile quality. Quality is shown on a gradient scale with blue as the highest quality and red as the lowest quality.

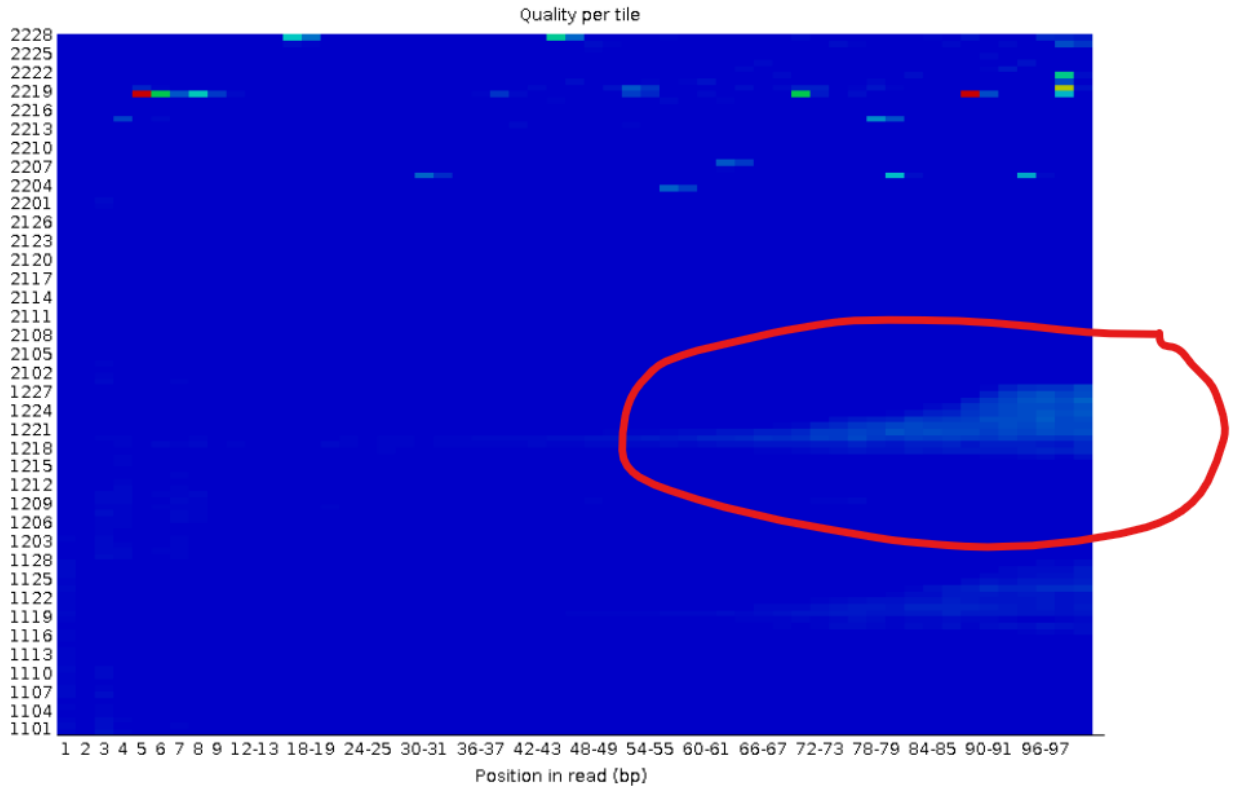

Figure 8: Sample 28 read 1 tile quality. Quality is shown on a gradient scale with blue as the highest quality and red as the lowest quality.

Part 2: Adapter trimming comparison

Adapters were trimmed using cutadapt version 4.4 and trimmomatic version 0.39. The following adapter sequences were trimmed with cutadapt:

R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Documentation for adapters is available at Illumina Knowledgebase (link).

Before trimming, it was confirmed that the R1 adapter was only found in read 1 for both samples, and the R2 adapter was only found in read 2 for both samples. The following Unix commands were used for positive confirmation:

```
zcat <fastq file read 1> | grep -c <read 1 adapter>
zcat <fastq file read 2> | grep -c <read 2 adapter>
```

The following Unix commands were used for negative confirmation (expect 0 results):

```
zcat <fastq file read 1> | grep -c <read 2 adapter>
zcat <fastq file read 2> | grep -c <read 1 adapter>
```

For sample 27, 10.4% of reads from read 1 were trimmed, and 11.1% of reads from read 2 were trimmed. For sample 28, 6.0% of reads from read 1 were trimmed, and 6.8% of reads from read 2 were trimmed.

Trimmomatic was run with the following parameters:
LEADING: quality of 3
TRAILING: quality of 3
SLIDING WINDOW: window size of 5 and required quality of 15
MINLENGTH: 35 bases

Trimmomatic produced the following:

Table 2: Trimmomatic Results

	Sample 27	Sample 28
Input Read Pairs	7226430	12428766
Both Surviving	6891402 (95.36%)	11736976 (94.43%)
Forward Only Surviving	326732 (4.52%)	677966 (5.45%)
Reverse Only Surviving	5500 (0.08%)	8735 (0.07%)
Dropped	2796 (0.04%)	5089 (0.04%)

Read length distributions

Between both samples, read 2 has higher frequencies of shorter read lengths. This means that adapter sequences appeared earlier in the read for R2, resulting in a greater length trimmed. R2 likely has more adapter trimming because the reads have been on the sequencing instrument for a longer time than R1, which could result in some fragmentation or degradation that results in shorter insert sequences.

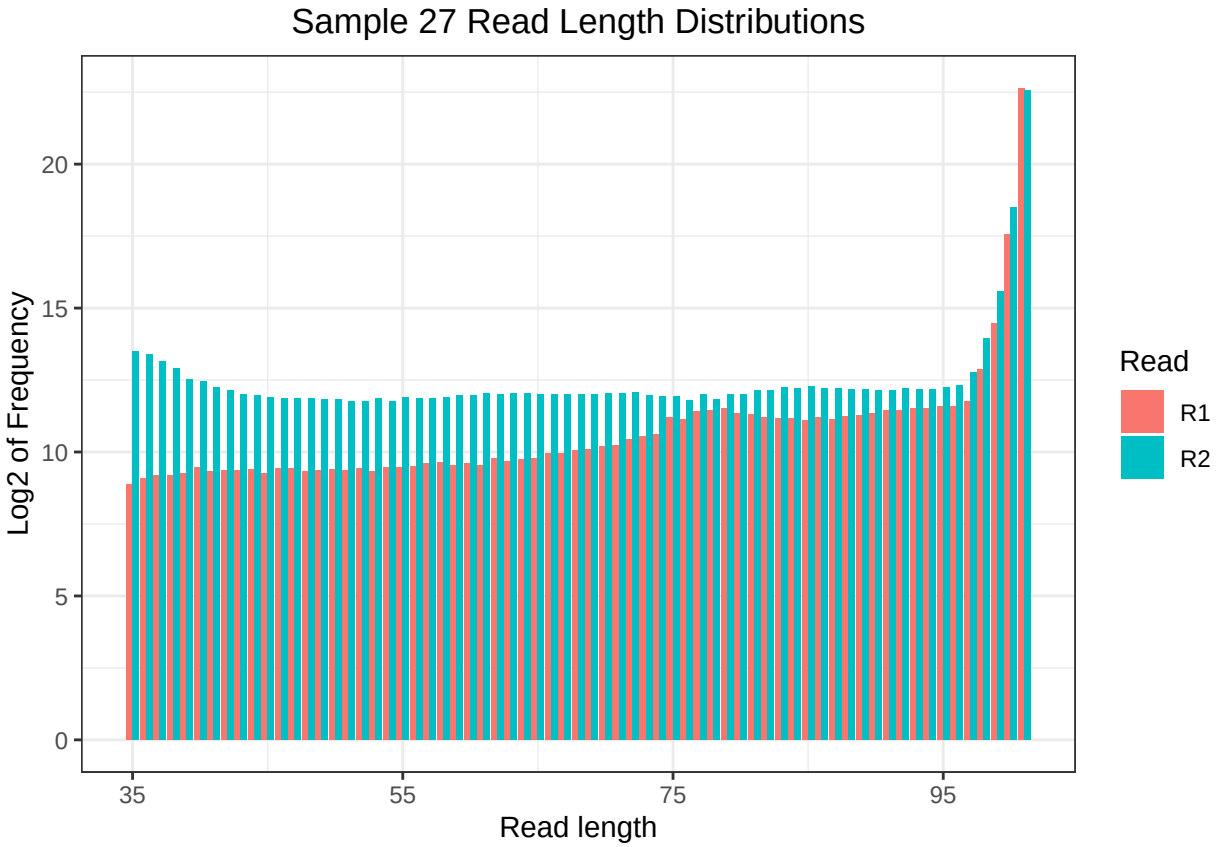

Figure 9: Sample 27 Read Length Distributions. Log2 of frequency is shown for readability.

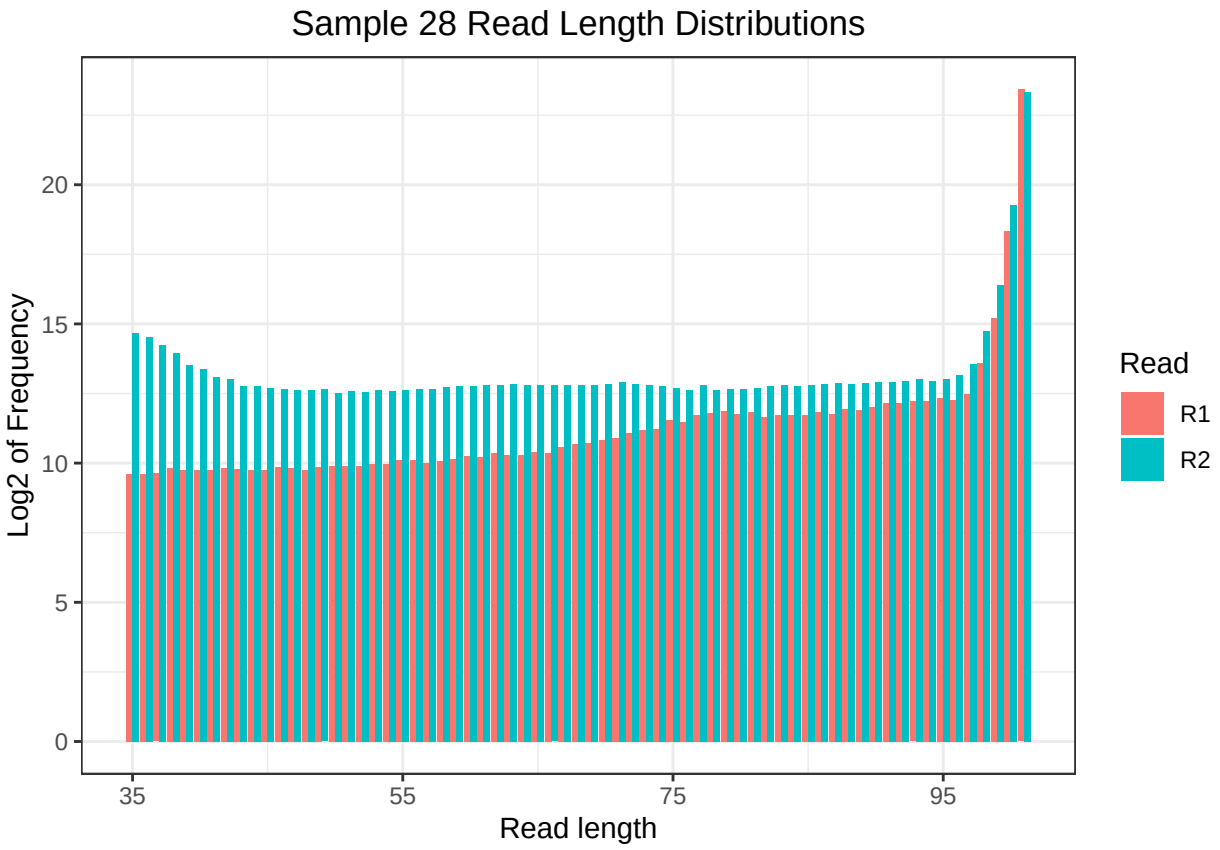

Figure 10: Sample 28 Read Length Distributions. Log2 of frequency is shown for readability.

Part 3: Alignment and strand specificity

I aligned samples 27 and 28 against a mouse genome database created with STAR version 2.7.10b (using Mus Ensembl release 110). I used a custom script, samparse_2.py, to count the number of reads that mapped to the reference genome.

Table 3: Reads mapped to the reference genome

	Mapped	Unmapped	Percent Mapped
Sample 27	12720381	1062423	92.3
Sample 28	22250477	1223475	94.8

I then counted reads that mapped to features using htseq-count version 2.0.3 and passed it both forward and reverse read parameters.

Table 4: Reads mapped to features

	Total reads	Reads mapped to feature	Percent mapped to feature
Sample 27 Forward	6891402	252094	3.66
Sample 27 Reverse	6891402	5453279	79.13
Sample 27 Forward	11736976	417513	3.56
Sample 27 Reverse	11736976	9537088	81.26

For both samples, only about 3% of reads map on the forward reads, and about 80% of reads mapped on the reverse reads. This means it likely was a stranded library. For an unstranded library, we would expect somewhere near a 50/50 split of mapped reads between forward and reverse reads. This was confirmed by looking at the methods; libraries were prepared with the KAPA Stranded mRNA-Seq kit.

Scripts used in this assignment

Talapas scripts:
| cutadapt.srun | htseq.srun | runfastq.srun |
| runstar.srun | staralign.srun | trim.srun |

Python scripts:
| qscore_dist_2.py | samparse_2.py |

Python modules:
| bioinfo.py version 0.5.2 |