PS2_Report

Emma Pearce

2025-09-05

Assignment Overview

In this assignment, I have processed RNA-seq reads from electric organ/skeletal muscle in preparation for future differential gene expression analysis. Throughout this process, I used a variety of established tools to assess read quality and perform trimming. I also compared those results to the quality assessments generated by my own code. I then aligned and counted the reads, gaining hands-on experience with each step of the pipeline.

Data

I was assigned two RNA-seq files from the PRJNA1005245 and PRJNA1005244 projects on NCBI.

SRR25630303 = Campylomormyrus rhynchophorus, young (6cm)

SRR25630398 = Campylomormyrus rhynchophorus, adult

Downloading data and creating environment:

To download both files from NCBI I ran these commands:

```
prefetch SRR25630303
prefetch SRR25630398

fasterq-dump --split-files SRR25630303/
fasterq-dump --split-files SRR25630398/
```

To set up the environment for this project, I ran these commands:

```
conda create --name QAA
conda activate QAA

conda install bioconda::fastqc
conda install bioconda::cutadapt=5.0
conda install bioconda::trimmomatic=0.39
```

Analysis:

I first ran the fastqs through fastqc and generated the html files for each, alsong with many plots

```
fastqc SRR25630398_1.fastq SRR25630398_2.fastq
fastqc SRR25630303_1.fastq SRR25630303_2.fastq
```

Fastqc plots:

For SRR25630303_1:

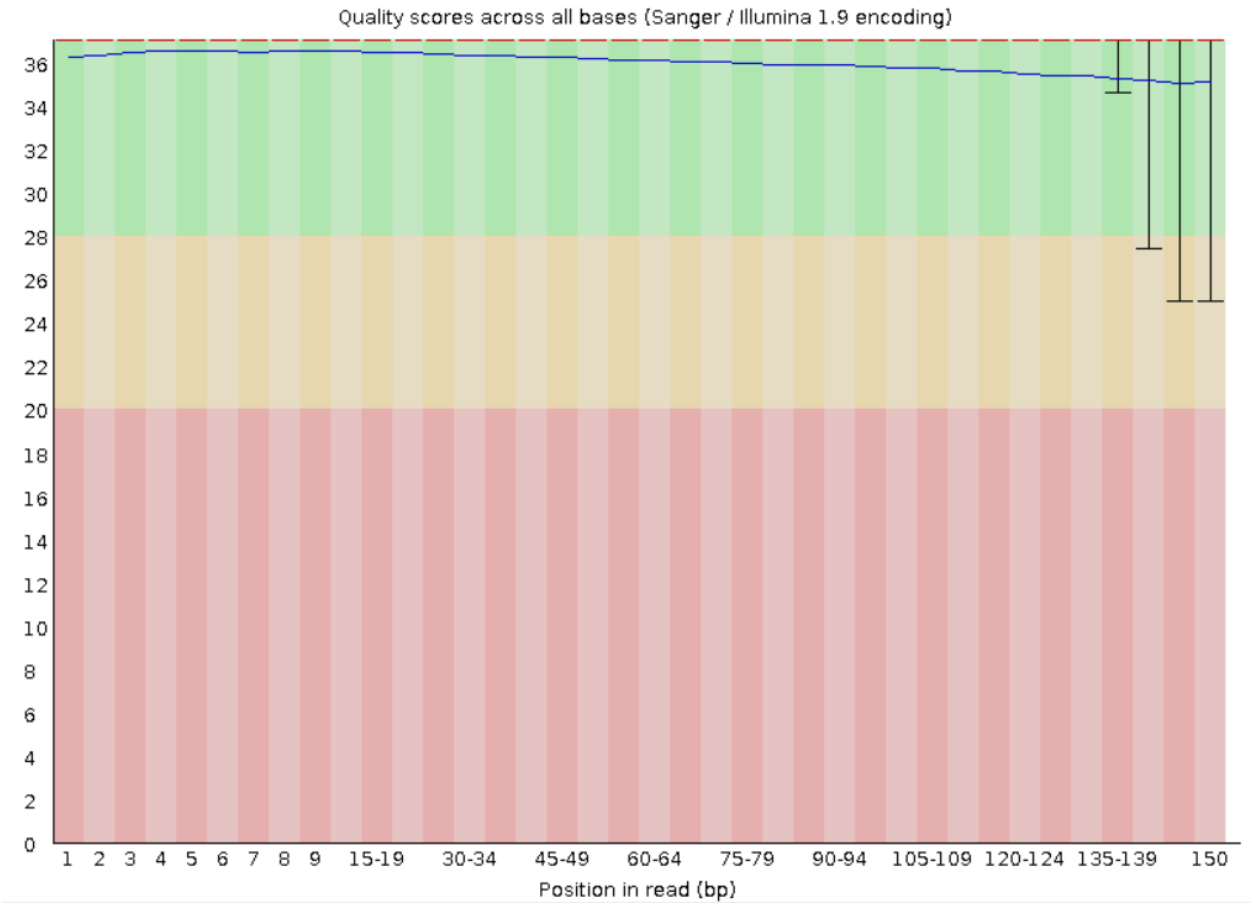

Figure 1: Quality scores across all bases using Sanger/Illumina 1.9 encoding from fastqc. The x axis is position in read, either single position or binned. The y axis is quality score. Green indicates good quality scores, yellow indicates okay quality scores, red indicates low quality scores. The blue line shows the average quality score for that base.

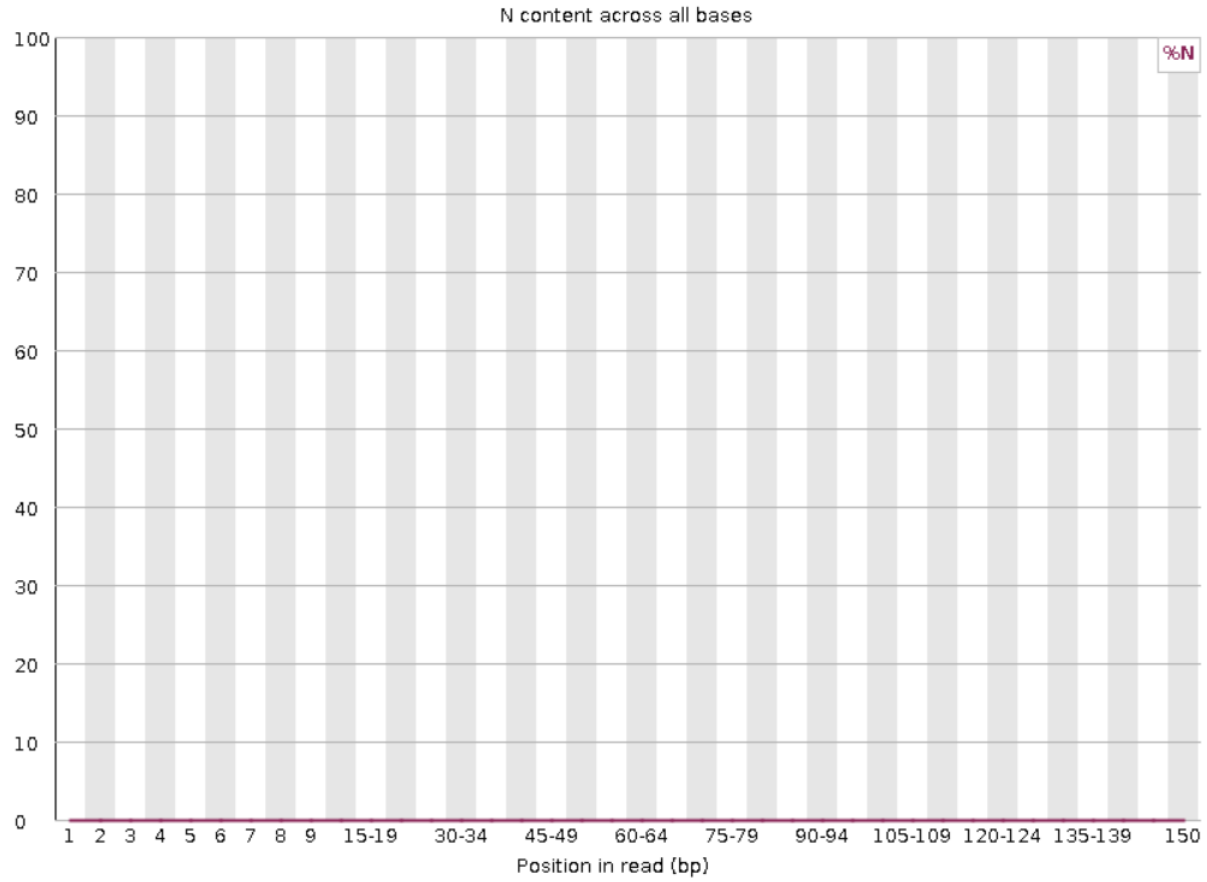

Figure 2: Percent of Ns at each position in read. X axis is the position in the read and y axis is the percent of Ns found there. The red line indicates what percent of bases were Ns at a given position.

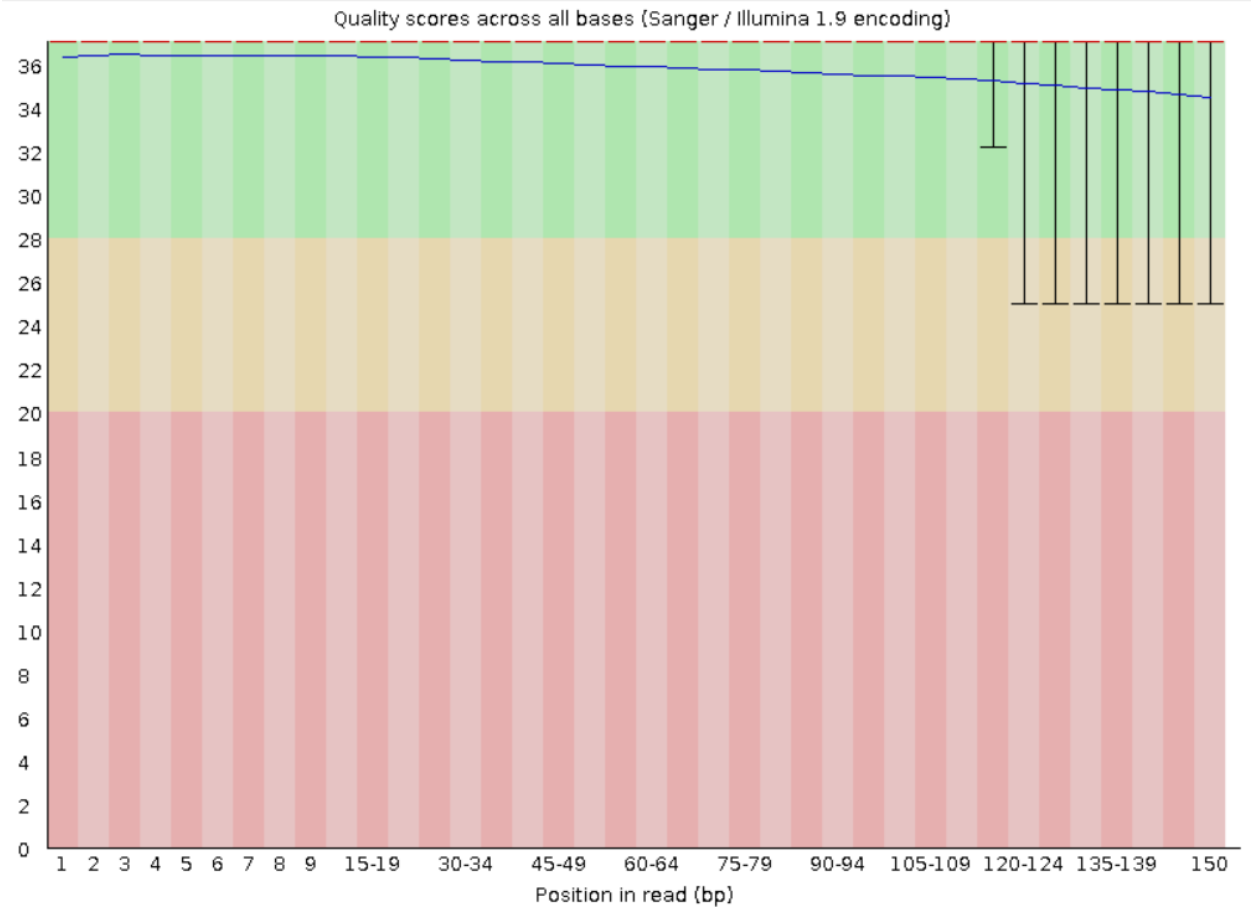

Figure 3: Quality scores across all bases using Sanger/Illumina 1.9 encoding from fastqc. The x axis is position in read, either single position or binned. The y axis is quality score. Green indicates good quality scores, yellow indicates okay quality scores, red indicates low quality scores. The blue line shows the average quality score for that base.

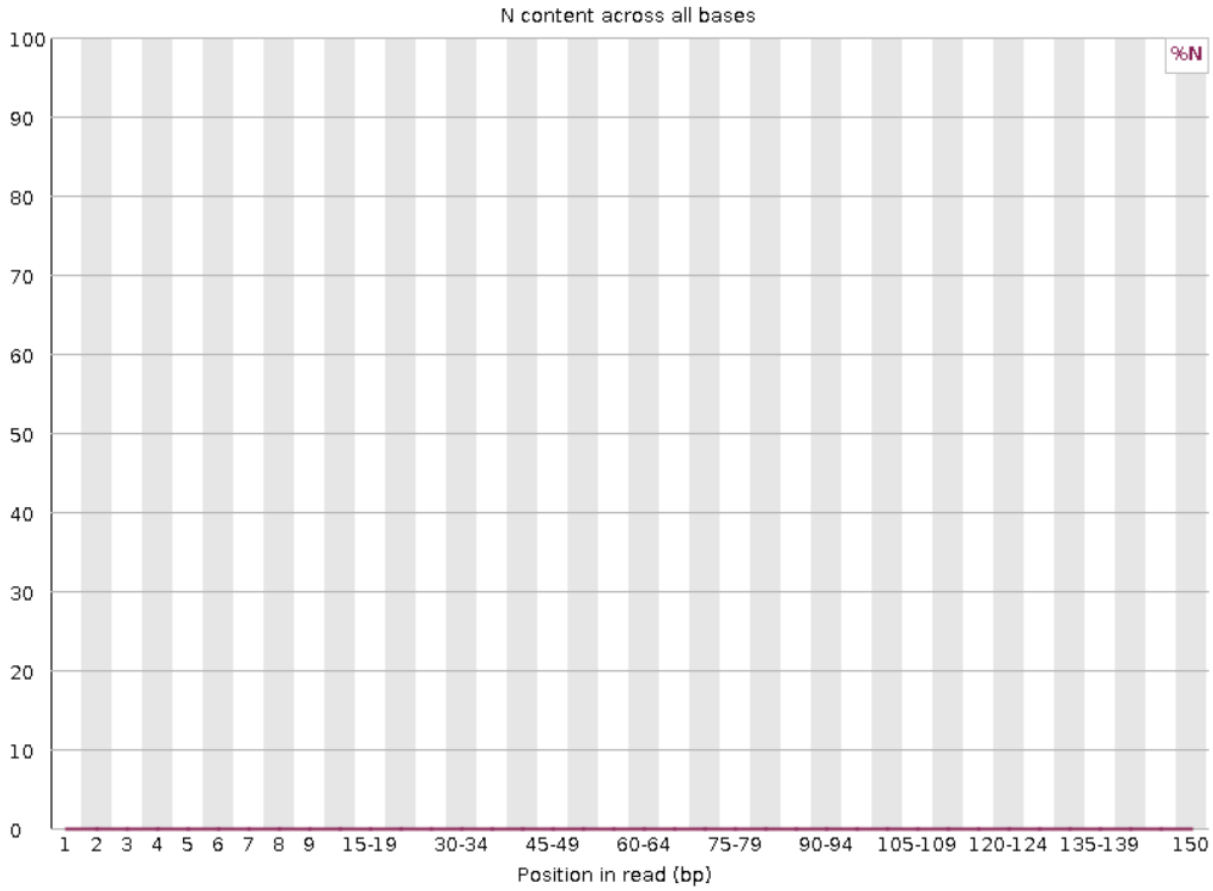

Figure 4: Percent of Ns at each position in read. X axis is the position in the read and y axis is the percent of Ns found there. The red line indicates what percent of bases were Ns at a given position.

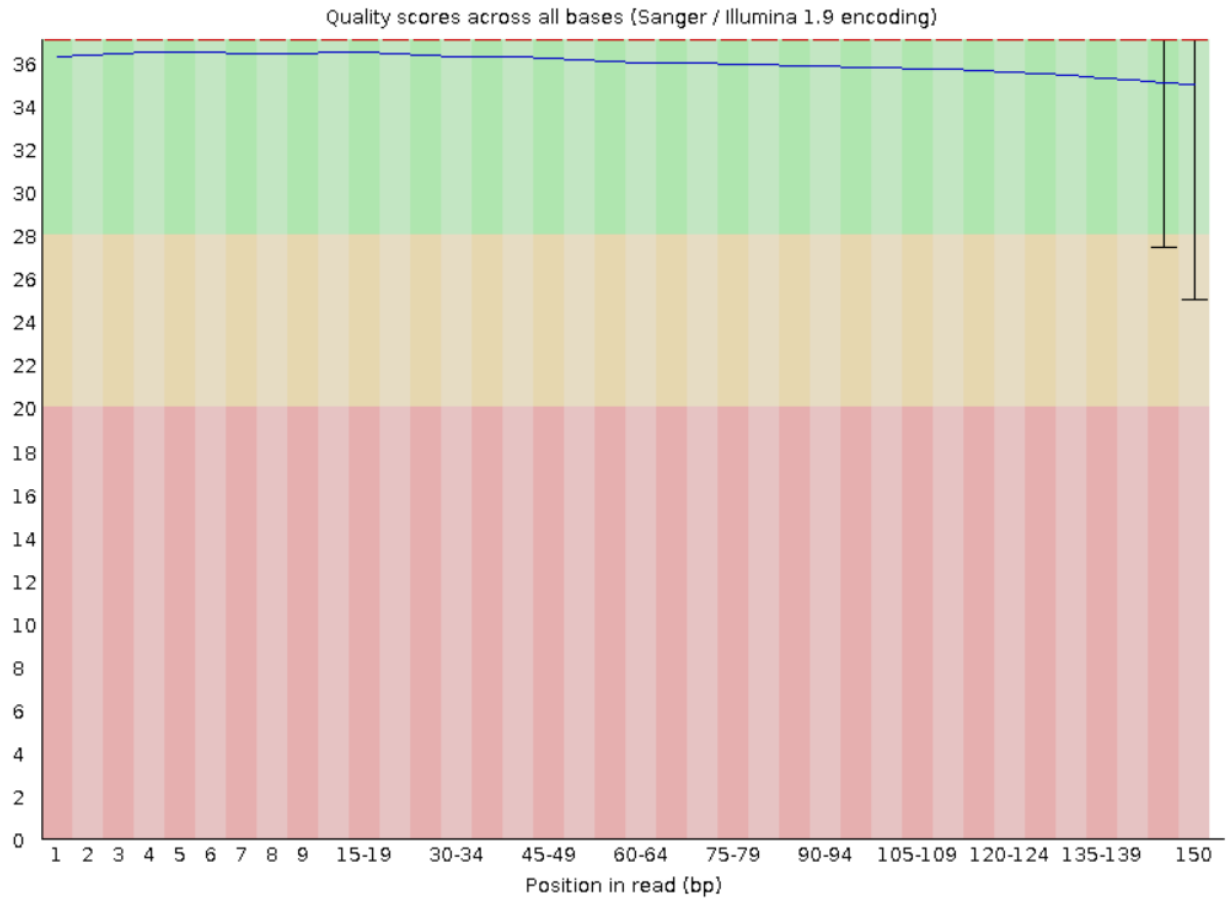

Figure 5: Quality scores across all bases using Sanger/Illumina 1.9 encoding from fastqc. The x axis is position in read, either single position or binned. The y axis is quality score. Green indicates good quality scores, yellow indicates okay quality scores, red indicates low quality scores. The blue line shows the average quality score for that base.

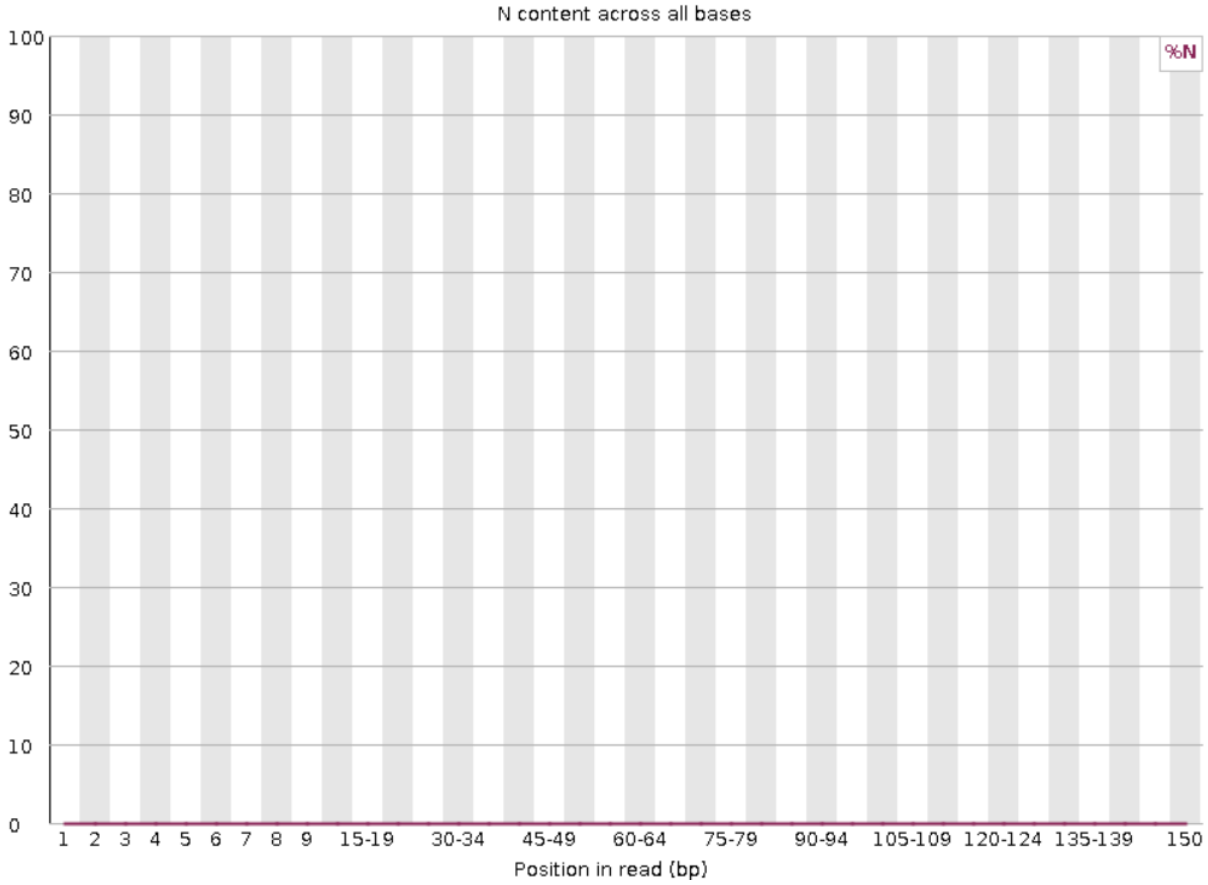

Figure 6: Percent of Ns at each position in read. X axis is the position in the read and y axis is the percent of Ns found there. The red line indicates what percent of bases were Ns at a given position.

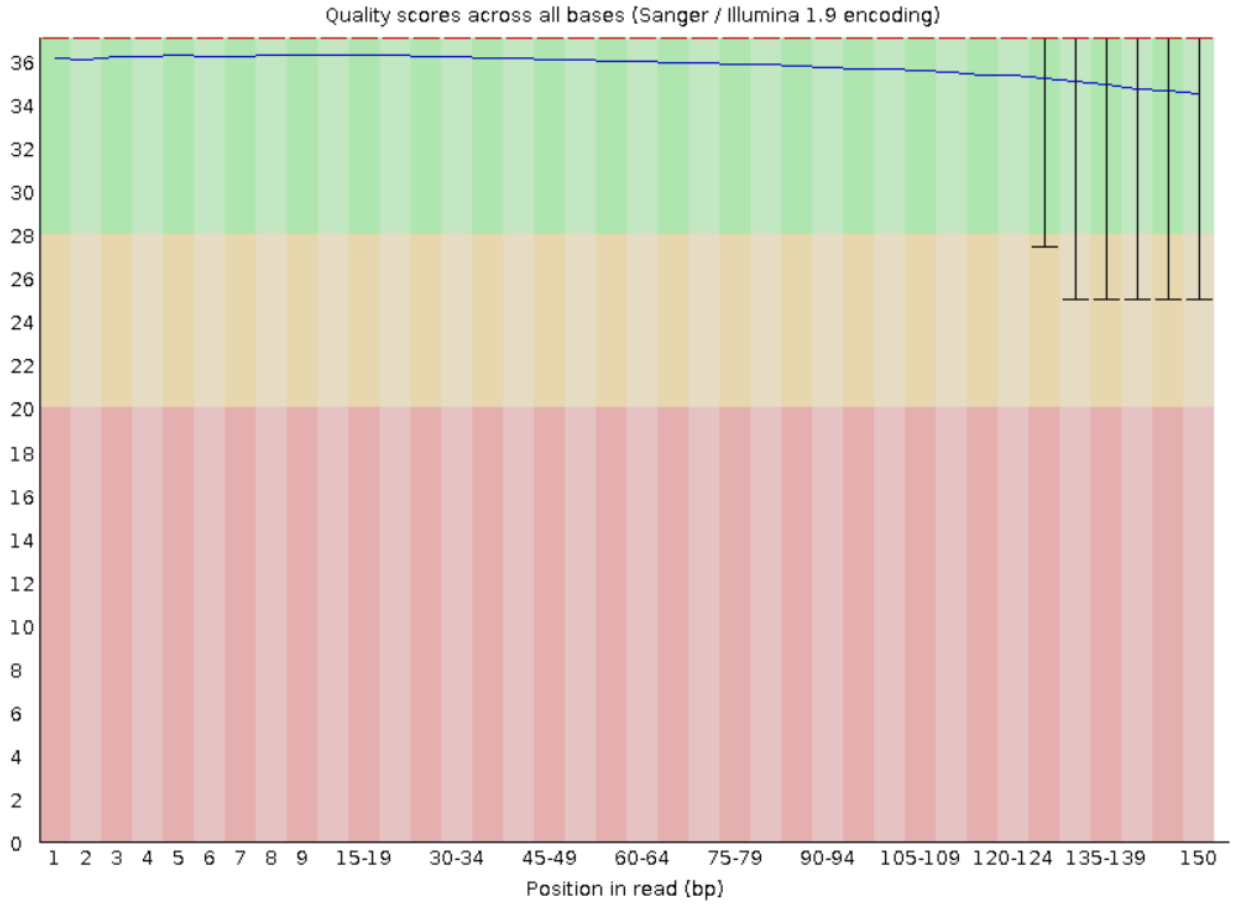

Figure 7: Quality scores across all bases using Sanger/Illumina 1.9 encoding from fastqc. The x axis is position in read, either single position or binned. The y axis is quality score. Green indicates good quality scores, yellow indicates okay quality scores, red indicates low quality scores. The blue line shows the average quality score for that base.

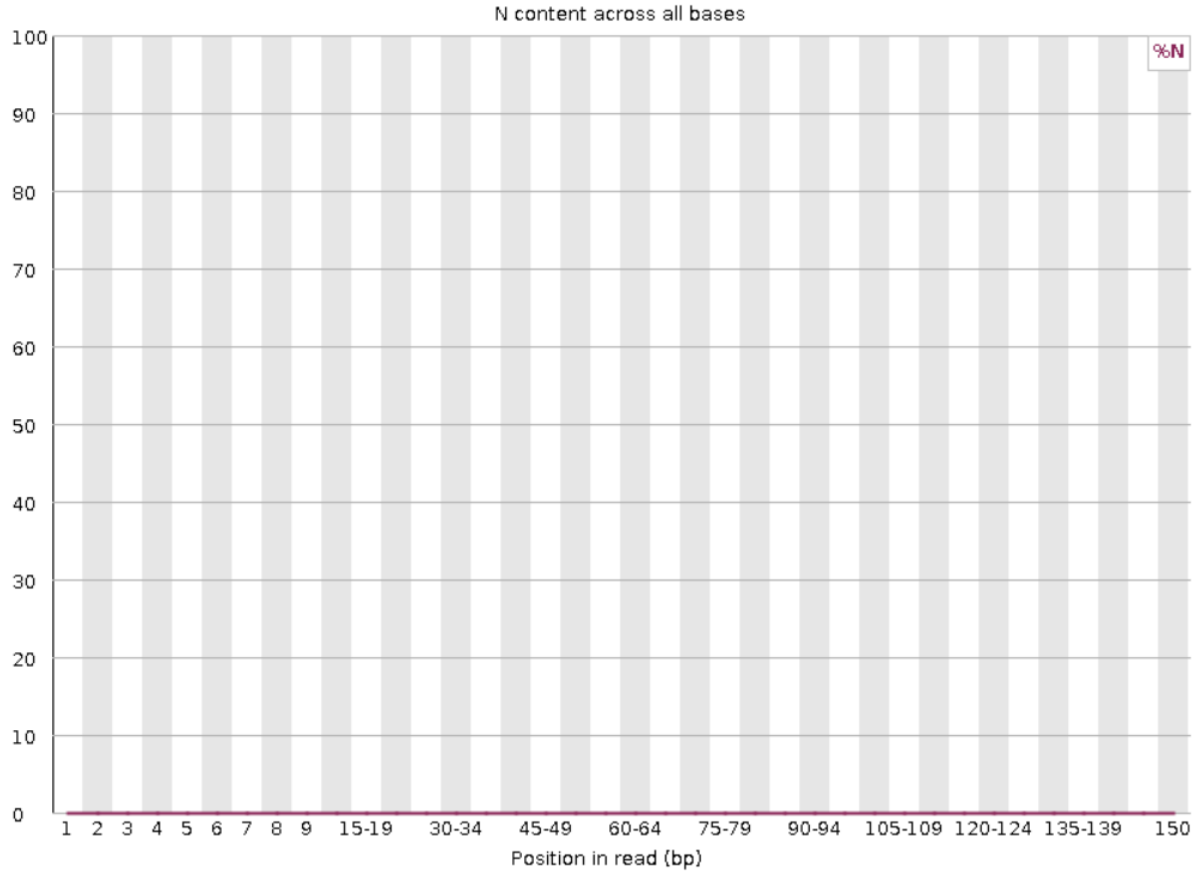

Figure 8: Percent of Ns at each position in read. X axis is the position in the read and y axis is the percent of Ns found there. The red line indicates what percent of bases were Ns at a given position.

For SRR25630303_2:

For SRR25630398_1:

SRR25630398_2:

Overall comments:

All four of these files have very similar quality scores at each position in the read. Interestingly in both the SRR25630303_2 and SRR25630398_2 files, the quality scores are lower at the end of the read than the similar position in the _1 file.

Histograms made with my code:

For SRR25630303_1:

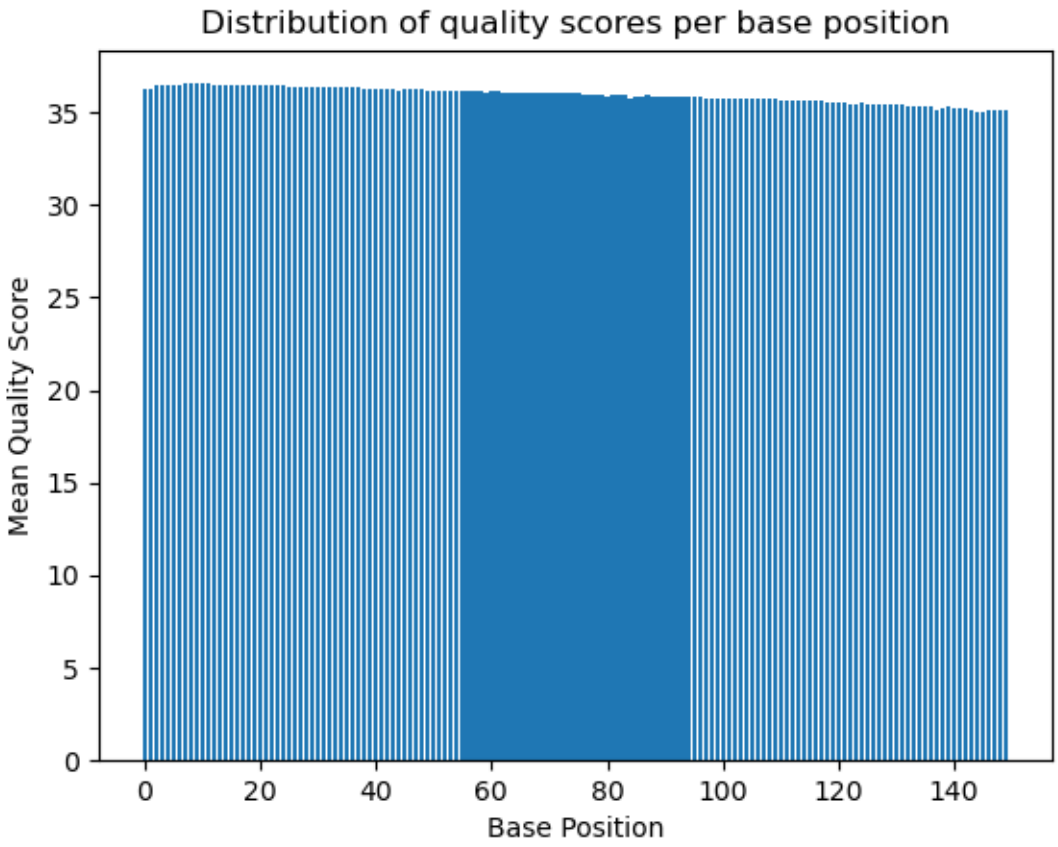

Figure 9: Distribution of quality scores at each base position. X axis is position in read and y axis is the mean quality score. The bars indicate the mean quality score at each position

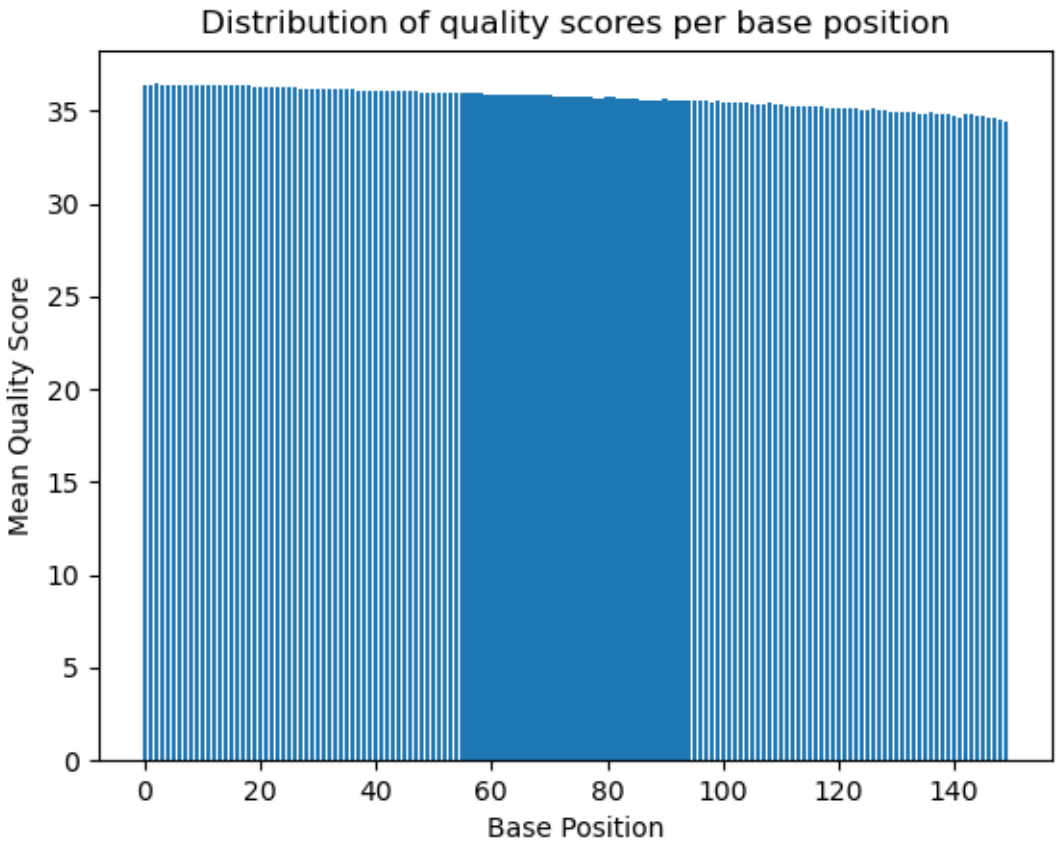

Figure 10: Distribution of quality scores at each base position. X axis is position in read and y axis is the mean quality score. The bars indicate the mean quality score at each position

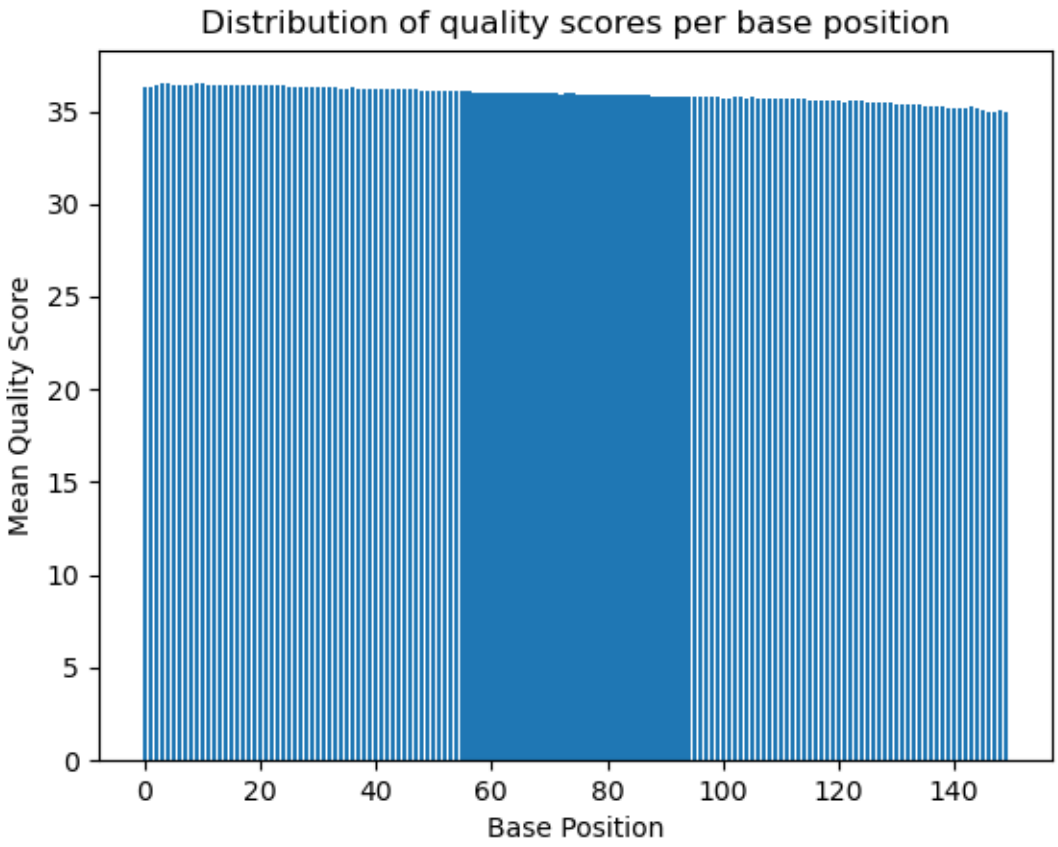

Figure 11: Distribution of quality scores at each base position. X axis is position in read and y axis is the mean quality score. The bars indicate the mean quality score at each position

For SRR25630303_2:

For SRR25630398_1:

For SRR25630398_2:

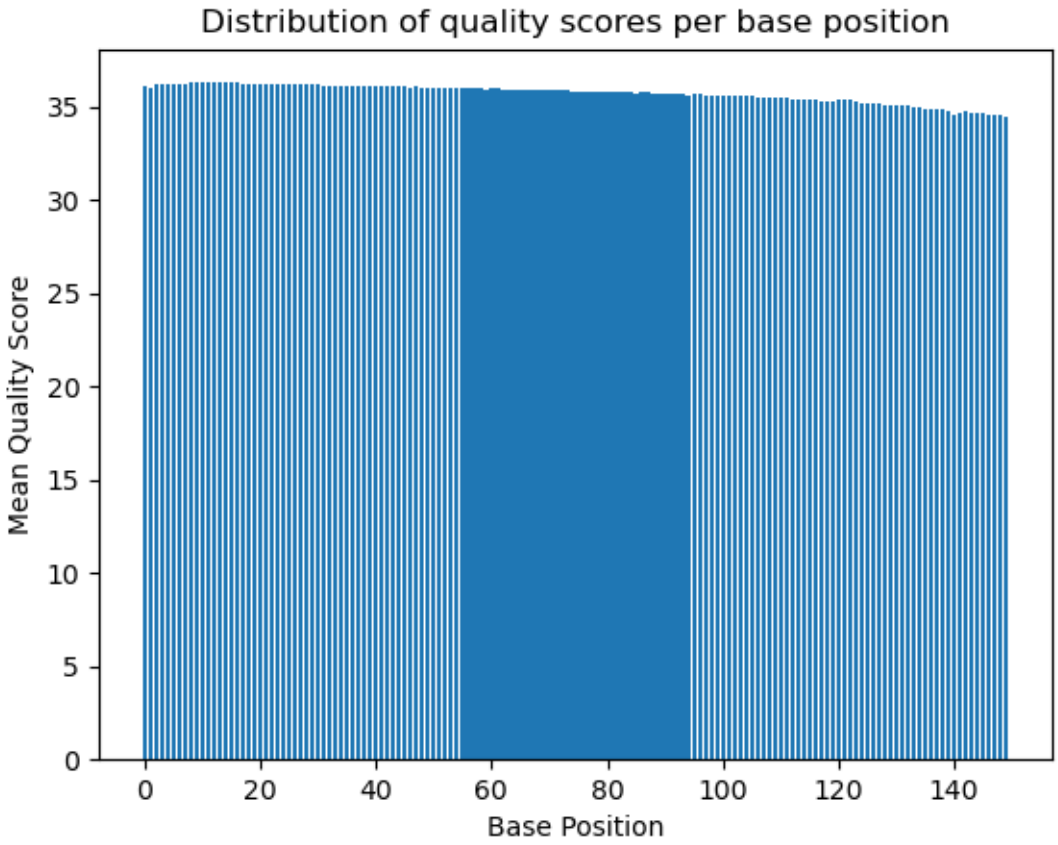

Figure 12: Distribution of quality scores at each base position. X axis is position in read and y axis is the mean quality score. The bars indicate the mean quality score at each position

Overall comments:

My histograms look fairly similar to the fastqc histograms of base quality, just without the fancier formatting. In mine we still see that the quality scores at the end of the reads in the second files of each sample dip slightly lower than the rest of the positions. I prefer the way my x axis is set up, with all the positions rather than some in bins and others not. However I prefer the y axis of the fastqc plots because it is easier to see what score it is. My histograms also took about 3x (5 minutes vs 15 minutes) as long to run as the fastqc, but took about the same amount of CPU.

Overall Quality:

The vast majority of all the reads in each file have a mean sequence quality near 36. This indicates a pretty high quality overall. The first ten (or so) positions in each read have unusual base content and are likely

barcodes and could be removed. SRR25630303 has very little adapter presence, but SRR25630398 has a noticeable amount of Illumina Universal Adapter near the ends of the reads. It would probably be useful to trip out the adapters in SRR25630398.

Adapter trimming:

SRR25630303:

```
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o trimmed_SRR256303
```

Timing: User time (seconds): 179.38

System time (seconds): 10.08

Percent of CPU this job got: 98%

Elapsed (wall clock) time (h:mm:ss or m:ss): 3:12.05

Proportions: Total read pairs processed: 41,934,422

Read 1 with adapter: 1,767,108 (4.2%)

Read 2 with adapter: 2,066,337 (4.9%)

Pairs written (passing filters): 41,934,422 (100.0%)

Total base pairs processed: 12,580,326,600 bp

Read 1: 6,290,163,300 bp

Read 2: 6,290,163,300 bp

Total written (filtered): 12,536,648,265 bp (99.7%)

Read 1: 6,268,302,186 bp

Read 2: 6,268,346,079 bp

SRR25630398:

```
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o trimmed_SRR256303
```

Timing: User time (seconds): 185.54

System time (seconds): 6.71

Percent of CPU this job got: 98%

Elapsed (wall clock) time (h:mm:ss or m:ss): 3:14.23

Proportions: Total read pairs processed: 34,878,180

Read 1 with adapter: 4,844,719 (13.9%)

Read 2 with adapter: 4,978,371 (14.3%)

Pairs written (passing filters): 34,878,180 (100.0%)

Total base pairs processed: 10,463,454,000 bp

Read 1: 5,231,727,000 bp

Read 2: 5,231,727,000 bp

Total written (filtered): 10,264,089,024 bp (98.1%)

Read 1: 5,128,839,415 bp

Read 2: 5,135,249,609 bp

Sanity check:

```
awk 'NR % 4 == 2' SRR25630303_1.fastq | grep -c 'AGATCGGAAGAGCACACGTCTGAACTCCAGTCA'
129695

wk 'NR % 4 == 2' SRR25630303_1.fastq | grep -c 'TGACTGGAGTTCAGACGTGTGCTCTTCCGATCT'
0

# Good! this makes sense that the r1 would have lots of the r1 adapter and very little of the reverse c
# This indicates it is the correct direction for this file
# This was repeated for each of the fastq files and has similar results
```

Trimmomatic:

Useing `Trimmomatic` to quality trim your reads. With the following specifed in this order: - LEADING: quality of 3 - TRAILING: quality of 3 - SLIDING WINDOW: window size of 5 and required quality of 15 - MINLENGTH: 35 bases

```
/usr/bin/time -v trimmomatic PE -threads 8 trimmed_SRR25630303_1.fastq trimmed_SRR25630303_2.fastq \
  qualtrimmed_paired_SRR25630303_1.fastq.gz qualtrimmed_unpaired_SRR25630303_1.fastq.gz \
  qualtrimmed_paired_SRR25630303_2.fastq.gz qualtrimmed_unpaired_SRR25630303_2.fastq.gz \
  LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35


/usr/bin/time -v trimmomatic PE -threads 8 trimmed_SRR25630398_1.fastq trimmed_SRR25630398_2.fastq \
  qualtrimmed_paired_SRR25630398_1.fastq.gz qualtrimmed_unpaired_SRR25630398_1.fastq.gz \
  qualtrimmed_paired_SRR25630398_2.fastq.gz qualtrimmed_unpaired_SRR25630398_2.fastq.gz \
  LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35
```

Plots:

SRR25630303:

SRR25630398:

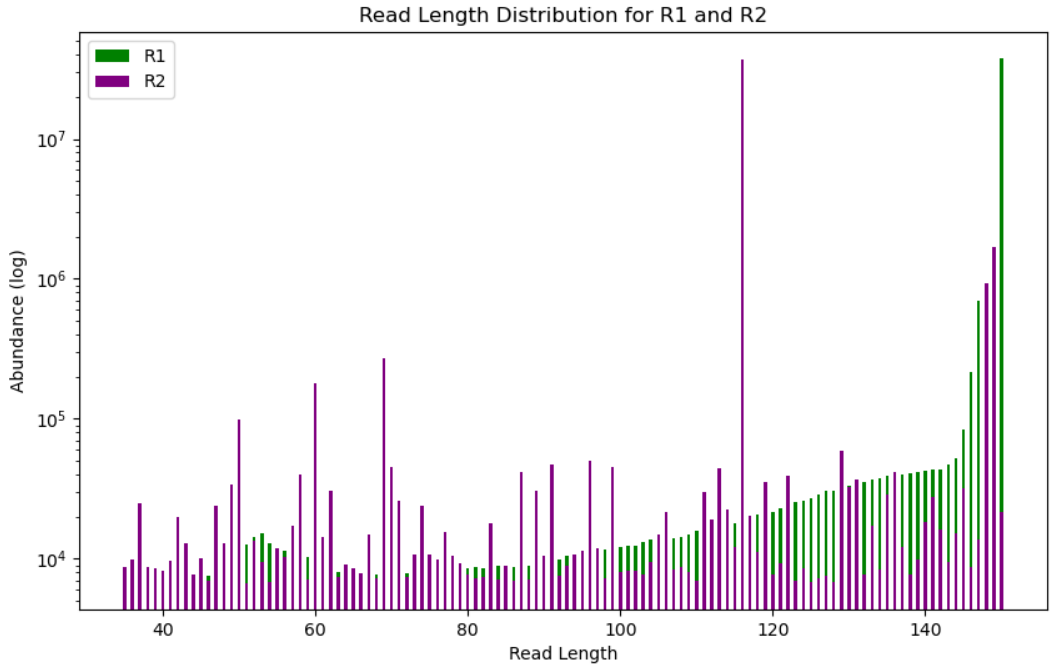

Figure 13: A histogram of read lengths and their abundances for both R1 (shown in green) and R2 (shown in purple). Read length is on the x axis and abundance is on the y axis.

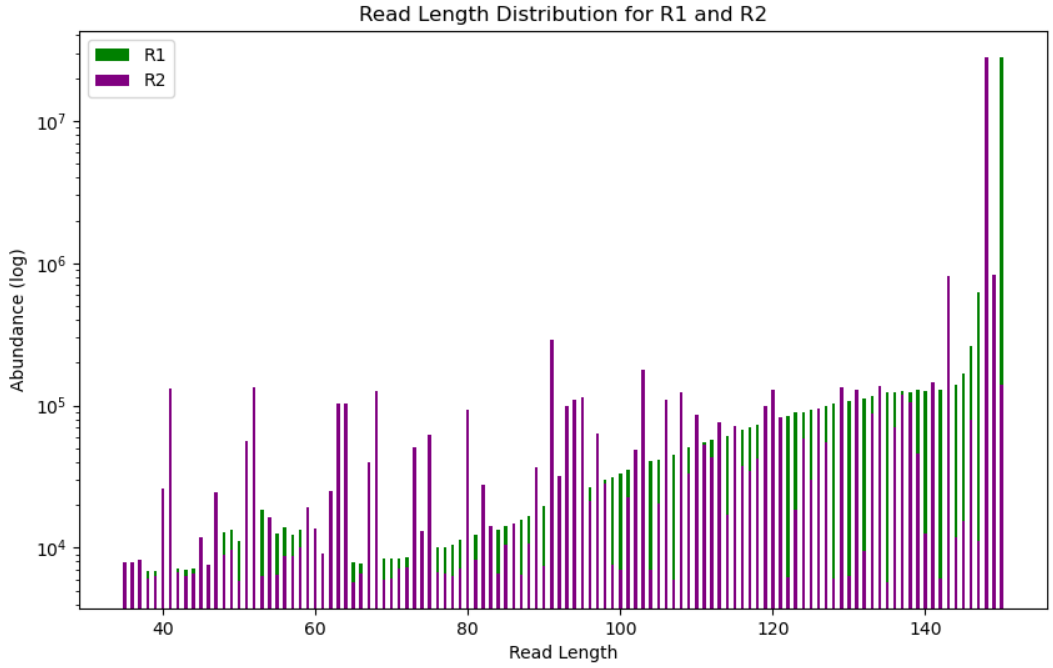

Figure 14: A histogram of read lengths and their abundances for both R1 (shown in green) and R2 (shown in purple). Read length is on the x axis and abundance is on the y axis.

Comments:

Here we see that the R2 reads for both samples is often more trimmed than the R1. R1s and R2s are often adapter-trimmed at different rates because R2s are more likely to read into adapter sequences when RNA fragments are shorter than the combined read length. Additionally, lower sequencing quality in R2s can lead to more frequent or aggressive trimming.

Aligning:

In this set I am aligning both SRR25630303 and SRR25630398 to the campylomormyrus genome on Dryad

I made the genome database using STAR genomeGenerate run mode. I then aligned both files to that database also using star

Example code: (repeated for SRR25630398)

```
/usr/bin/time -v STAR --runThreadN 8 --runMode alignReads \
 --outFilterMultimapNmax 3 \
 --outSAMunmapped Within KeepPairs \
 --alignIntronMax 1000000 --alignMatesGapMax 1000000 \
 --readFilesCommand zcat \
 --readFilesIn /projects/bgmp/epea/bioinfo/Bi623/PS/QAA/qualtrimmed_paired_SRR25630303_1.fastq.gz /proj
 --genomeDir /projects/bgmp/epea/bioinfo/Bi623/PS/QAA/align \
 --outFileNamePrefix aligned_SRR25630303_
```

I then converted to bams and sorted them to be used as input to picard (removing PCR duplications). And finally reconverted back to sam files to count the number of mapped and unmapped reads

```
samtools view -b aligned_SRR25630303_Aligned.out.sam > aligned_SRR25630303_Aligned.out.bam
samtools sort aligned_SRR25630303_Aligned.out.bam > aligned_SRR25630303_Aligned.sorted.out.bam

picard MarkDuplicates INPUT=aligned_SRR25630303_Aligned.sorted.out.bam OUTPUT=dedup_aligned_SRR25630303

samtools view -h dedup_aligned_SRR25630303_Aligned.sorted.out.bam > dedup_aligned_SRR25630303_Aligned.s

# Repeated for SRR25630398
```

Mapped and unmapped reads:

SRR25630303: Number of reads mapped: 18484340

Number of reads unmapped: 3651623

SRR25630398: Number of reads mapped: 32532195

Number of reads unmapped: 4777469

htseq-count:

Here I am using htseq-count to count the number of deduplicated reads that map to features. The first run will be with the –stranded=yes option and the second with –stranded=reverse to try and tell the strandedness of both files.

```
/usr/bin/time -v htseq-count --stranded=yes -i gene_id dedup_aligned_SRR25630398_Aligned.sorted.out.sam

/usr/bin/time -v htseq-count --stranded=reverse -i gene_id dedup_aligned_SRR25630398_Aligned.sorted.out

awk '$1 !~ /^__/ {sum += $2} END {print sum}' counts_stranded_SRR25630398
awk '$1 !~ /^__/ {sum += $2} END {print sum}' counts_reverse_SRR25630398
```

SRR25630398:

Stranded: 899068

Reverse: 19458807

SRR25630303:

Stranded: 486475

Reverse: 11196054

Comments:

In both samples, over 95% of reads map to the reverse strand of the gene annotation. This is a strong indication that the data is strand specific and in the reverse. This mean that the correct option to be using with htseq-count is –stranded=reverse.