QAA_report

2025-09-05

QAA Report

I was assigned two samples from NCBI SRA, SRR25630307 from Campylomormyrus compressirostris and SRR25630395 from Campylomormyrus rhynchophorus. I used SRA_tools to download the data and transform them into fastq files.

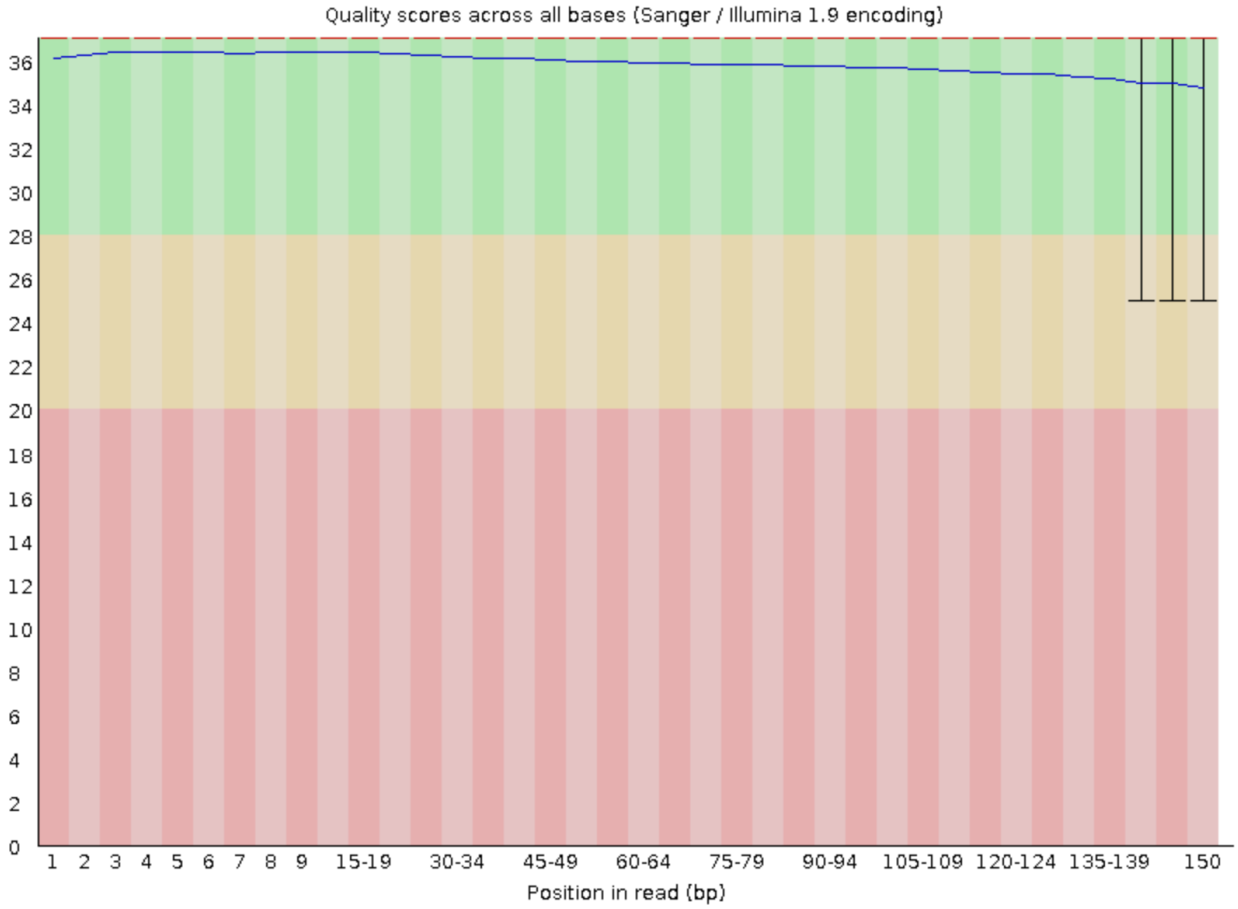
```
prefetch SRR25630307
fasterq-dump SRR25630307 --split-files -O ./fastq

prefetch SRR25630395
fasterq-dump SRR25630395 --split-files -O ./fastq
```

FastQC and Read Quality Score Distributions

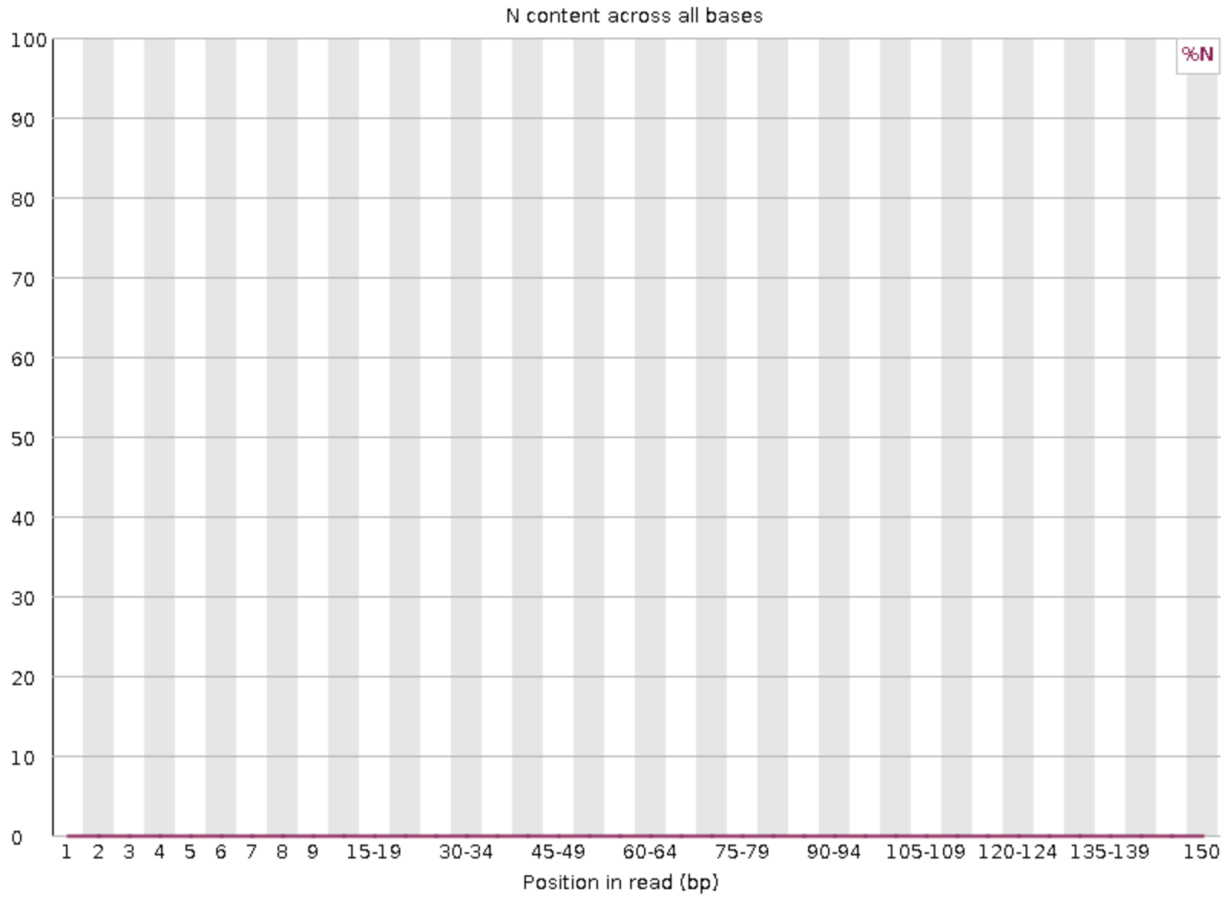
I created a new environment using conda that I called QAA, and installed trimmomatic, fastqc and cutadapt. Then I used FastQC to generate data about my fastq file, specifically per base quality score and the number of "N"s found per base position. Next, I compared this output to my per base quality score python script I had previously created.

```
fastqc fastq/SRR25630307_1.fastq.gz
fastqc fastq/SRR25630307_2.fastq.gz
fastqc fastq/SRR25630395_1.fastq.gz
fastqc fastq/SRR25630395_2.fastq.gz
```

cco__com124__EO__6cm__1__R1.fastq.gz

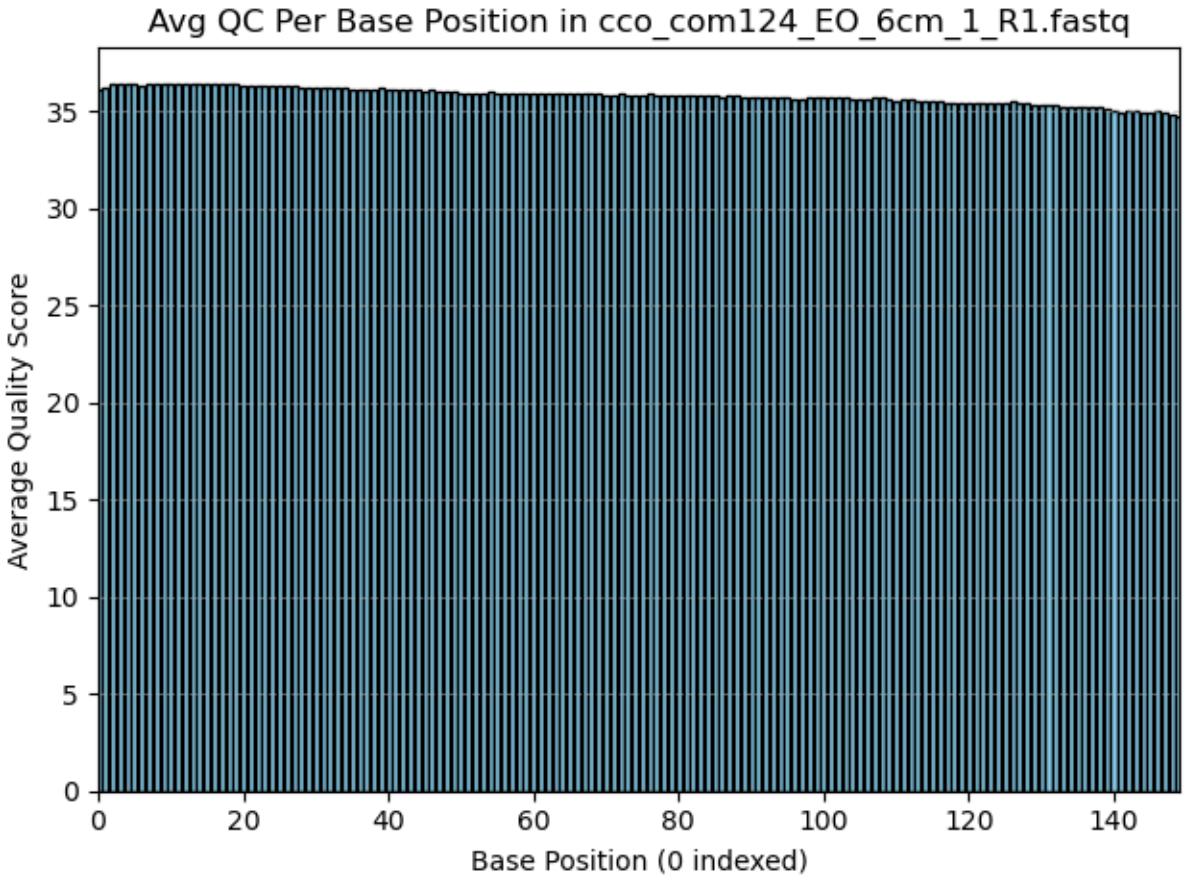


Caption: Per Base Quality Score Distribution of R1 C. compresssirostris fastq file generated from FastQC



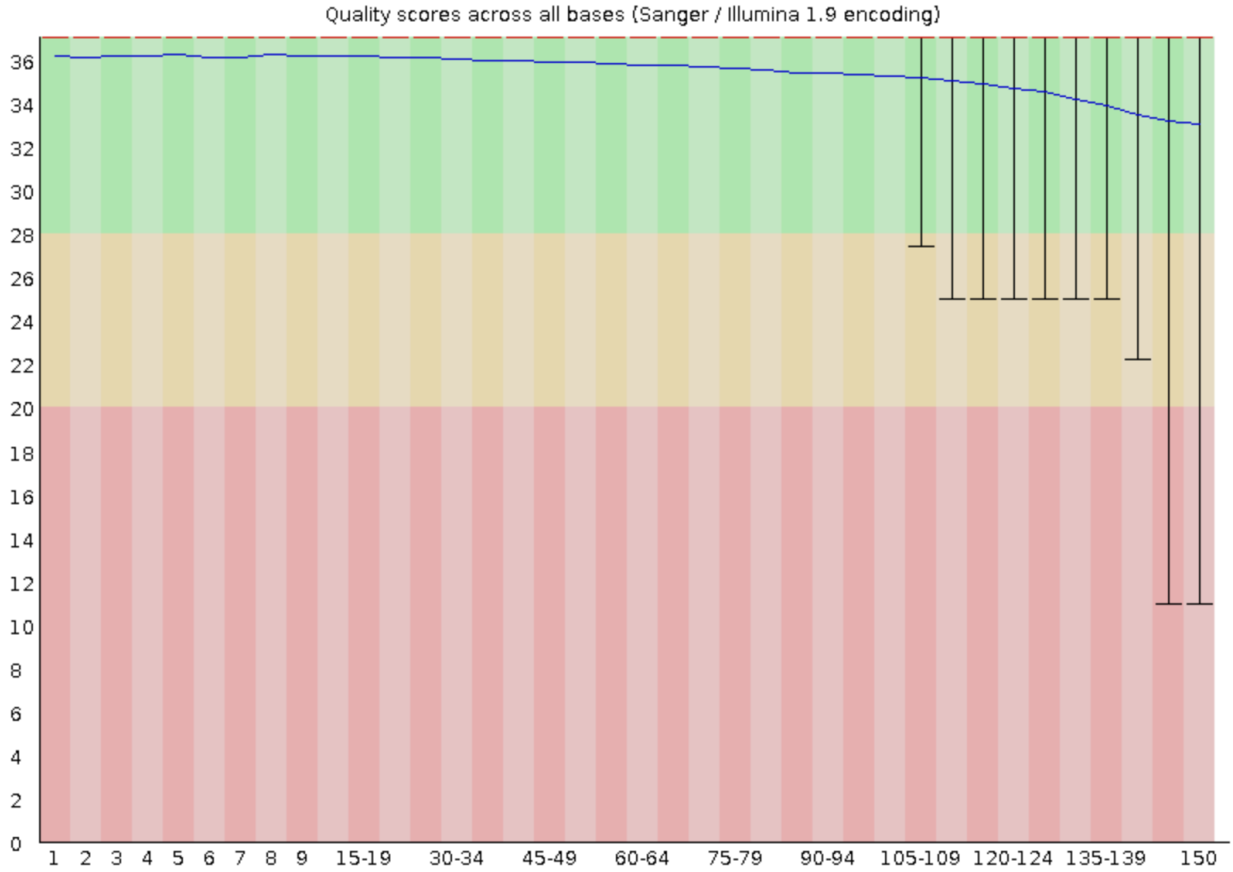
Caption: Distribution of "N"s found per base in the R1 C. compresssirostris fastq file generated from FastQC

The average quality score per base per read was roughly 36. No sequences were flagged as being low quality, and the most common sequence length was 150 base pairs. There were nearly zero bases that were marked with an N, which is consistent with the fact that the average quality score was 36.

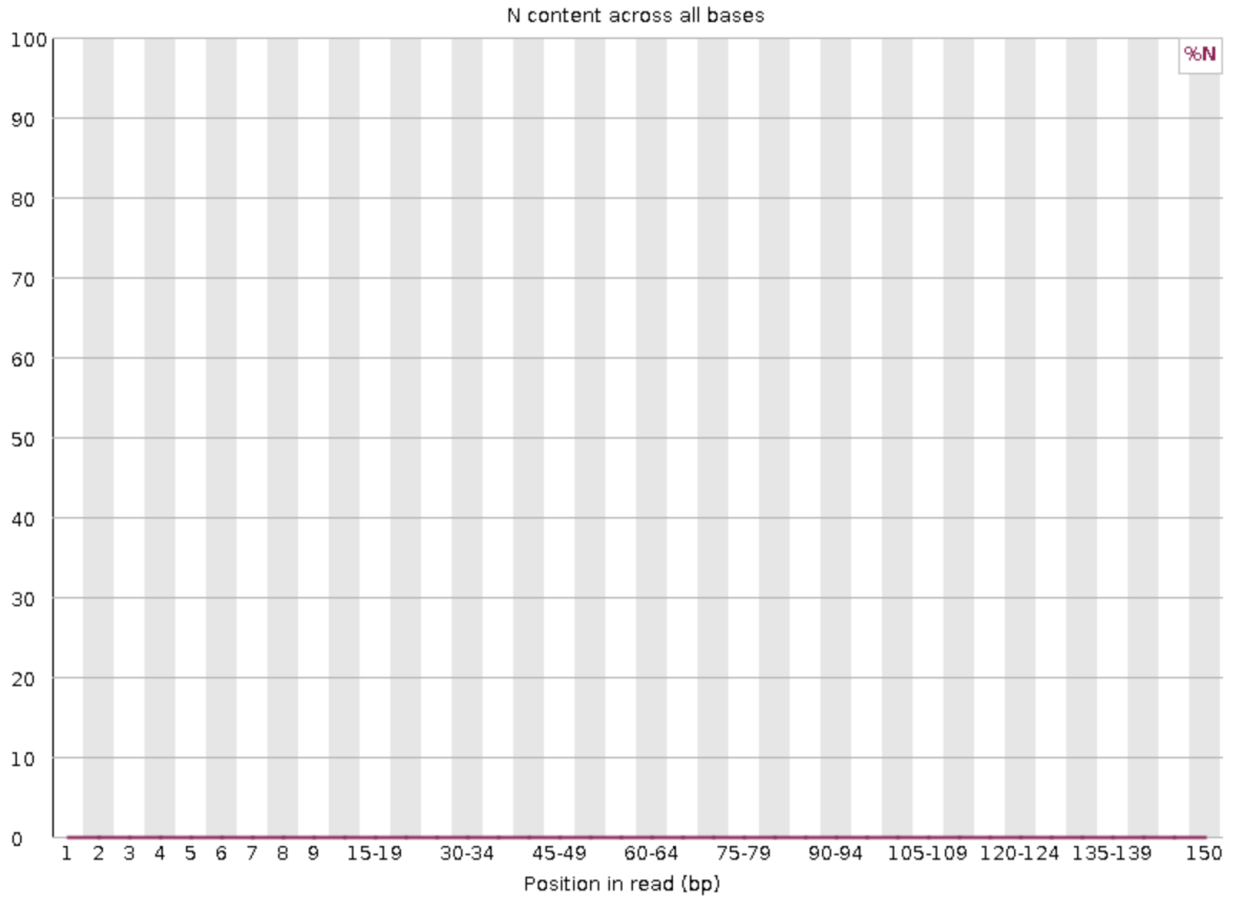


Caption: Per Base Quality Score Distribution of R1 C. compresssirostris fastq file generated from my python script

cco__com124__EO__6cm__1__R2.fastq.gz

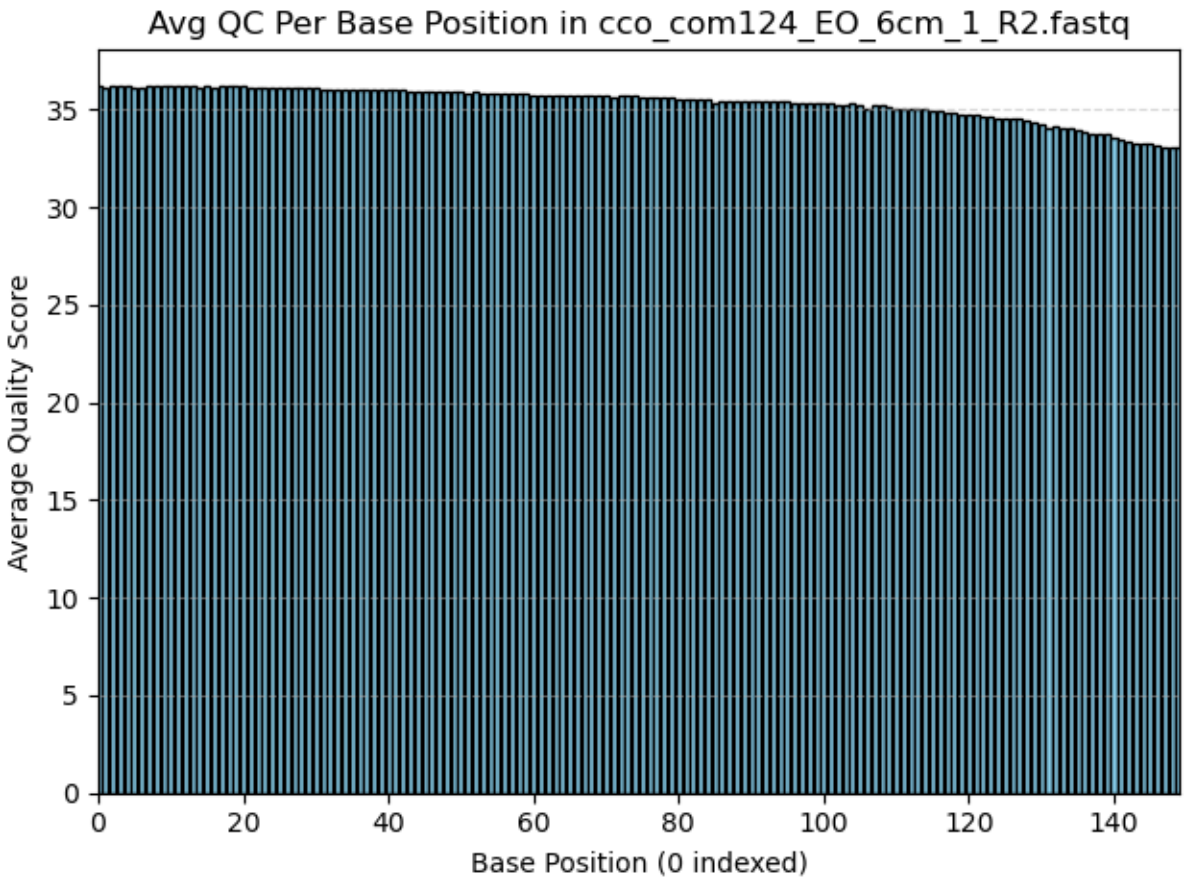


Caption: Per Base Quality Score Distribution of R2 C. compresssirostris fastq file generated from FastQC



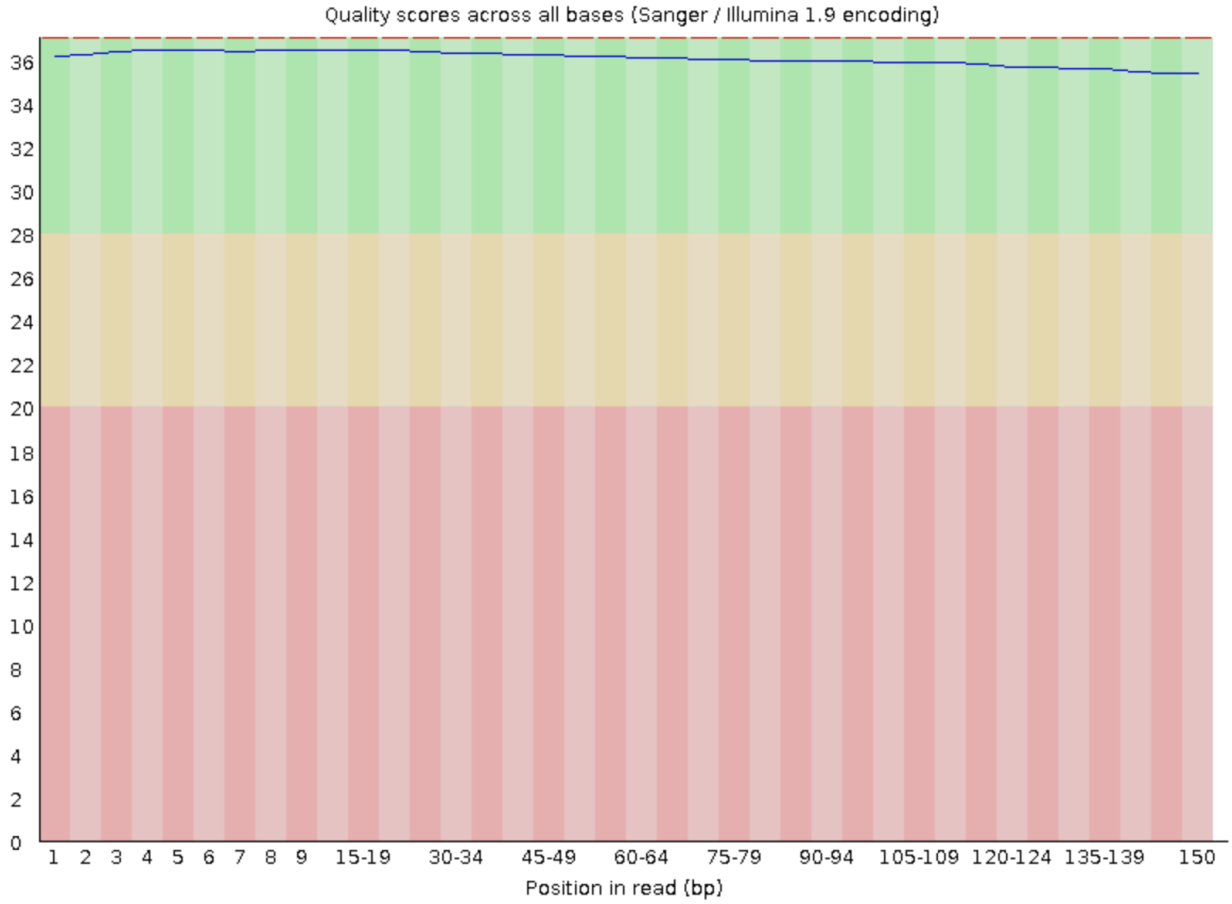
Caption: Distribution of "N"s found per base in the R2 C. compresssirostris fastq file generated from FastQC

The average quality score per base per read was roughly 36. There were no sequences that were flagged as low quality, and the sequence length was 150 base pairs. The average quality score began to dip at the end, which is typically occurs as errors begin to accumulate.

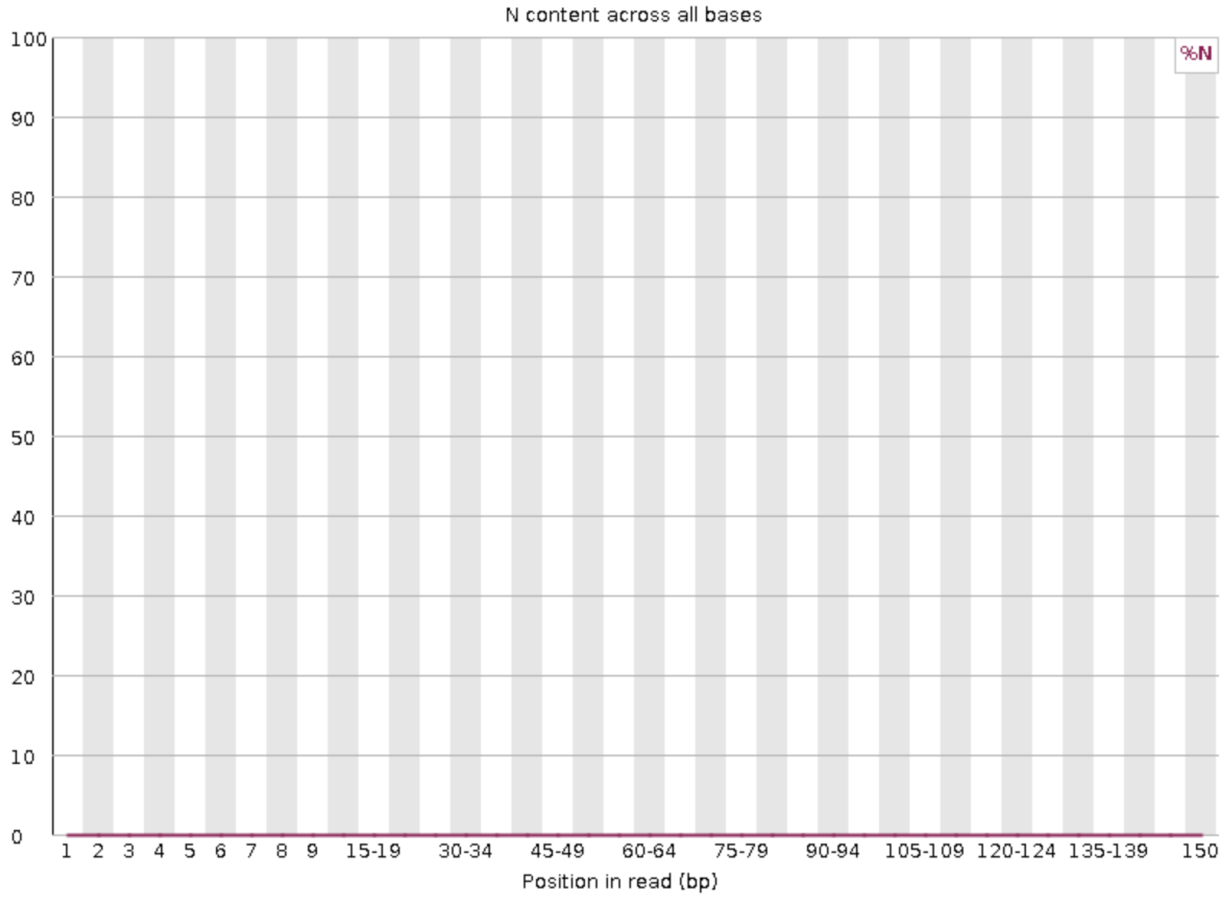


Caption: Per Base Quality Score Distribution of R2 C. compresssirostris fastq file generated from my python script

crh__rhy115__EO__adult__2__R1.fastq.gz

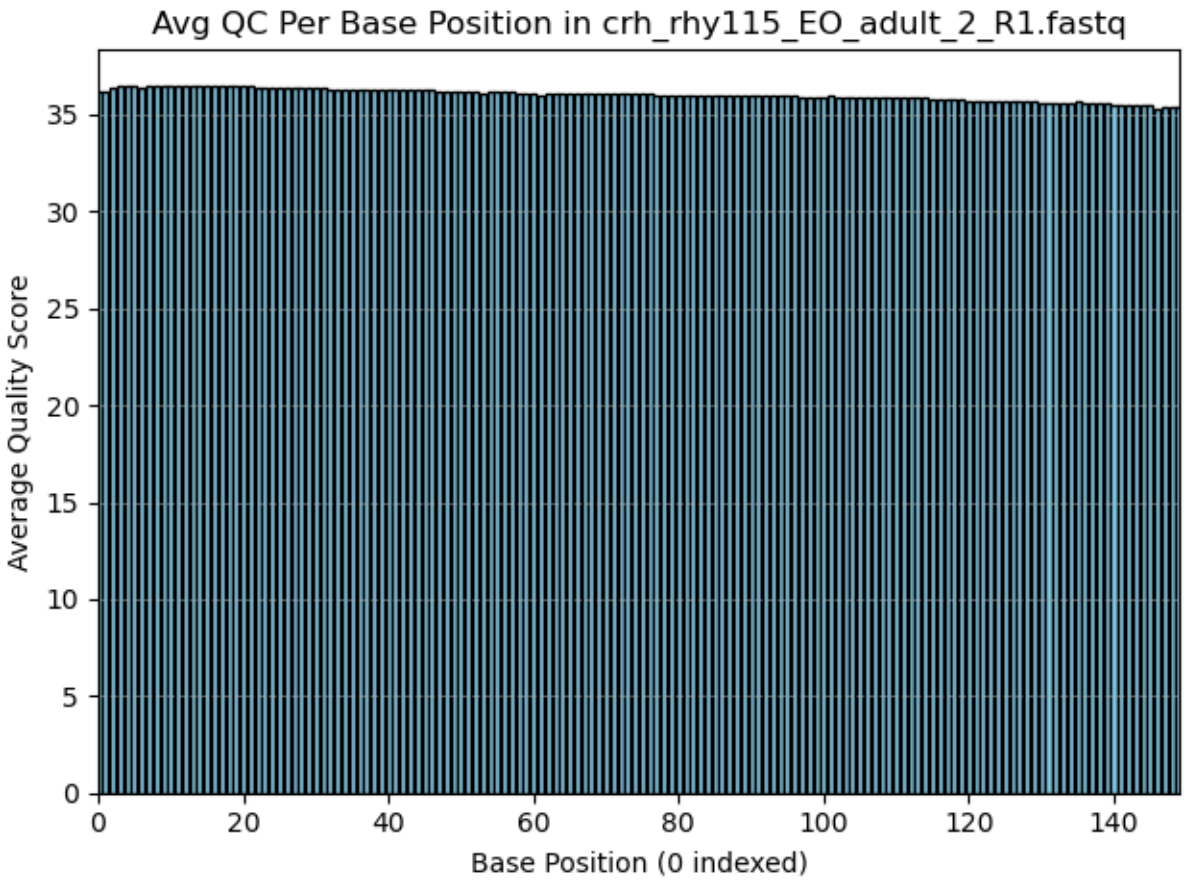


Caption: Per Base Quality Score Distribution of R1 C. rhynchophorus fastq file generated from FastQC



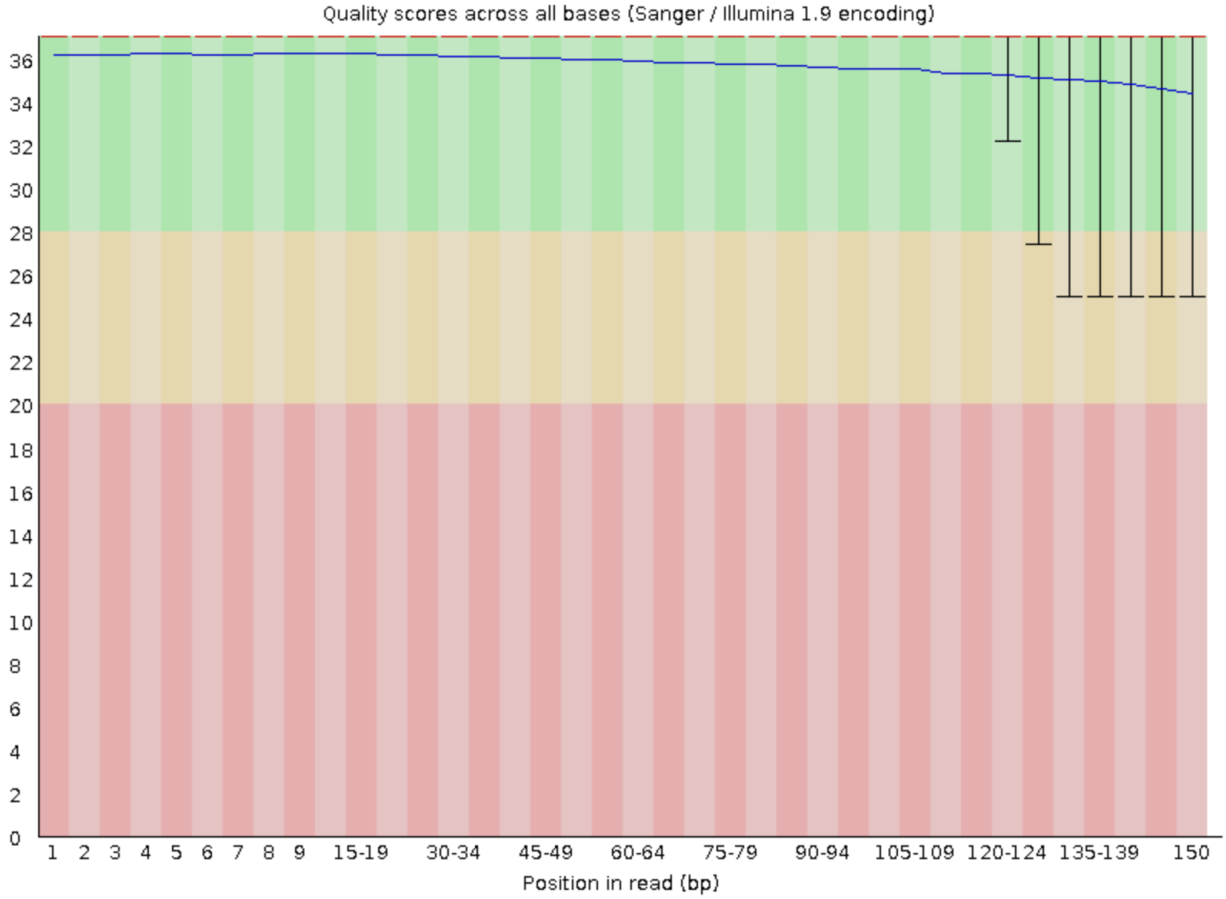
Caption: Number of "N"s per base in R1 C. rhynchophorus fastq file generated from my python script

The average quality score per base per read was roughly 36 and no sequences were flagged as low quality. There was no "N" nucleotides found, which is makes sense with the high average quality score per base.

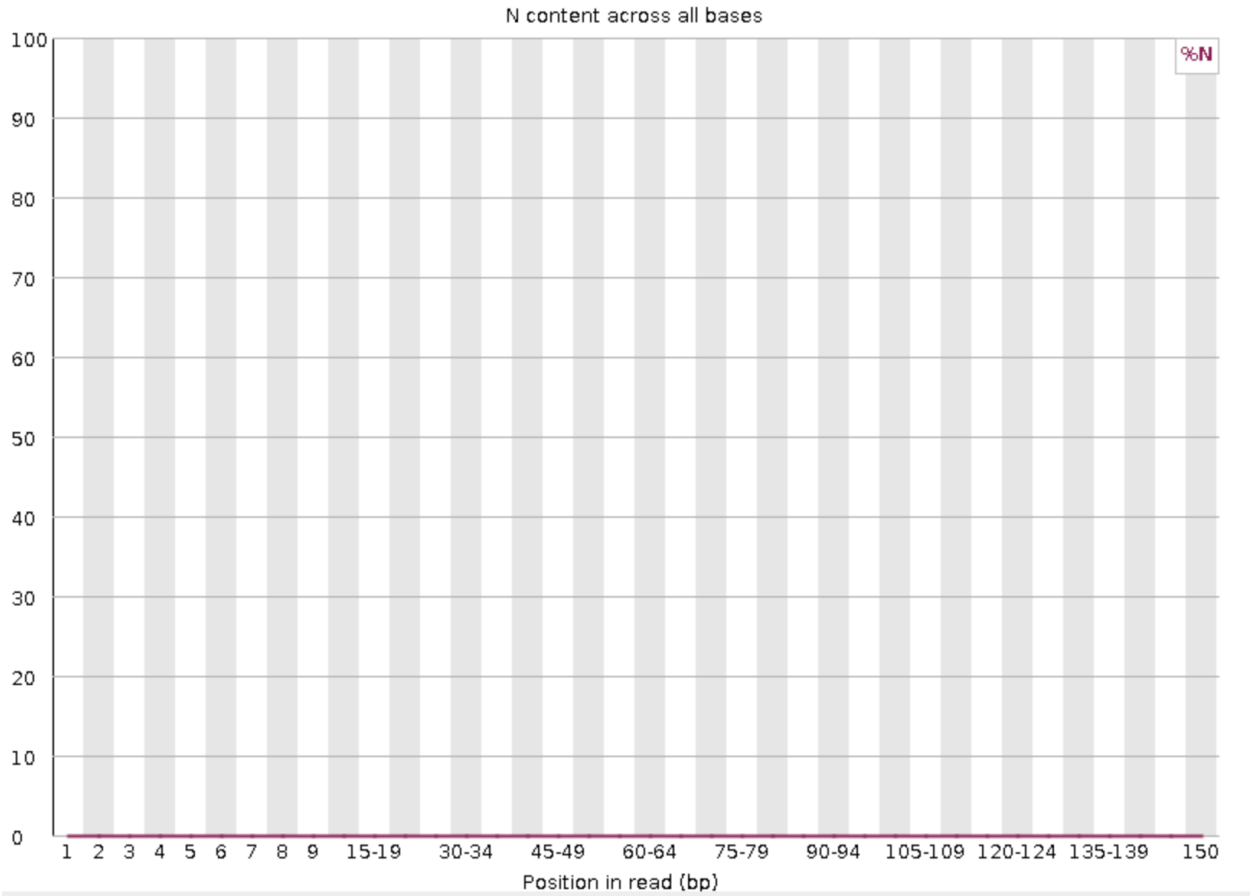


Caption: Per Base Quality Score Distribution of R1 C. rhynchophorus fastq file generated from my python script

crh__rhy115__EO__adult__2__R2.fastq.gz

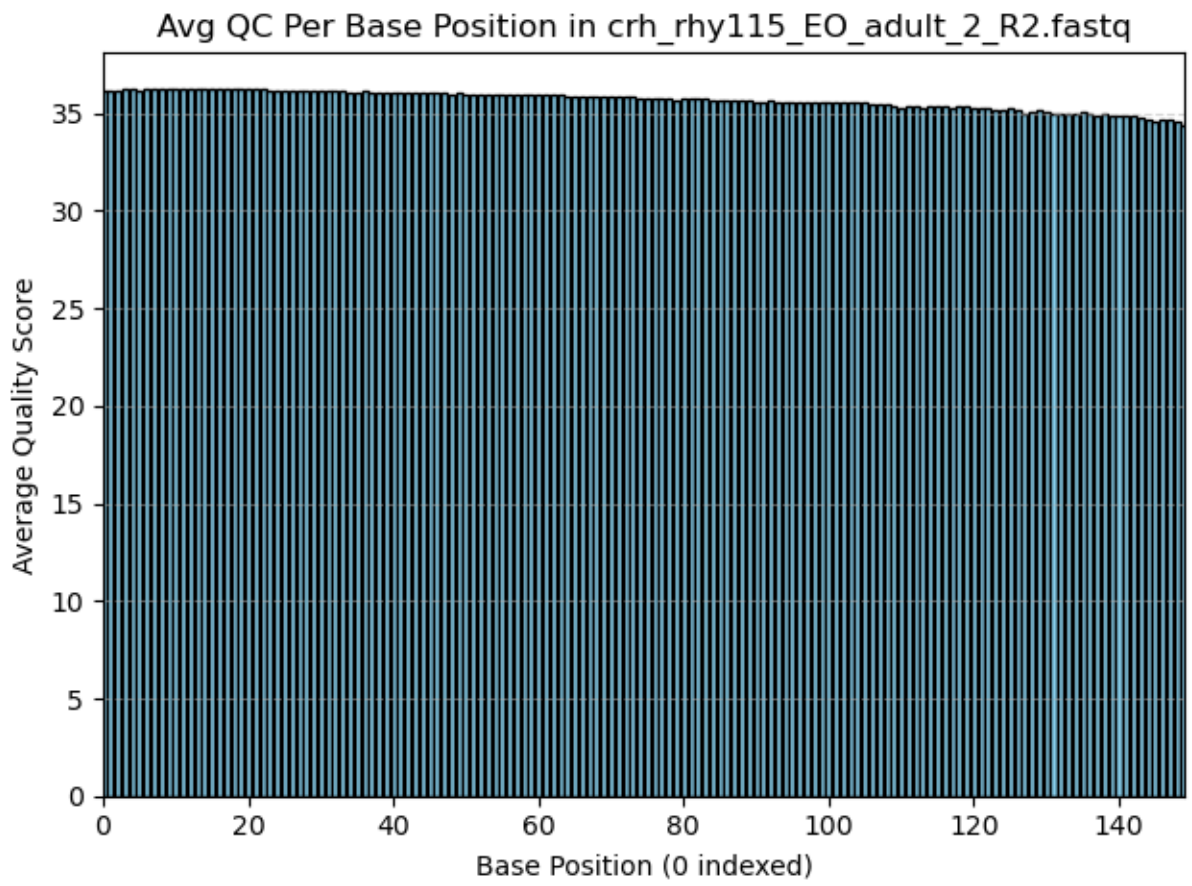


Caption: Per Base Quality Score Distribution of R2 C. rhynchophorus fastq file generated from FastQC



Caption: Number of "N"s per base in R1 C. rhynchophorus fastq file generated from my python script

The average quality score per base per read was roughly 36 and there were not any sequences flagged as low quality. Also, there were no "N" nucleotides detected, which is consistent with the high average quality score per base.



Caption: Per Base Quality Score Distribution of R2 C. rhynchophorus fastq file generated from my python script

My script vs FastQC

My plots looked the same as the FastQC plots, besides the fact they the FastQC plots included the standard deviation and the mean, while mine just included the mean; however, the runtime that it took to generate the plots from c. compressirostris was roughly 8 minutes, and the max resident set size was 68,004 KB. For C. rhynchophorus, it took roughly 25 minutes to generate the plots and the maximum resident set size was 65436 KB. My runtime and memory usage were far higher than FastQC.

Adaptor Trimming Comparison

Next I used cutadapt to remove adaptors from our reads. I searched for the adaptor sequences using zcat FASTQFILE | awk 'NR % 4 == 2 { print; print "—" }' to view only the sequences in the file. I knew from the FastQC plots that the barcodes were on the 3' end, so I tried to find common sequences near the 3' end. This was difficult. Eventually, I determined that the R1 adapator was AGATCGGAAGAGCACACGTCT-GAACTCCAGTCA and the R2 adapator was AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT. To verify the adaptor sequences, I used the bash commands shown in the code block below on all fastq files to confirm that the adaptor sequence for the R1 files wasAGATCGGAAGAGCACACGTCTGAACTCCAGTCA and AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT was the adaptor sequence for the R2 files, with both being located on the 3' end of the reads.

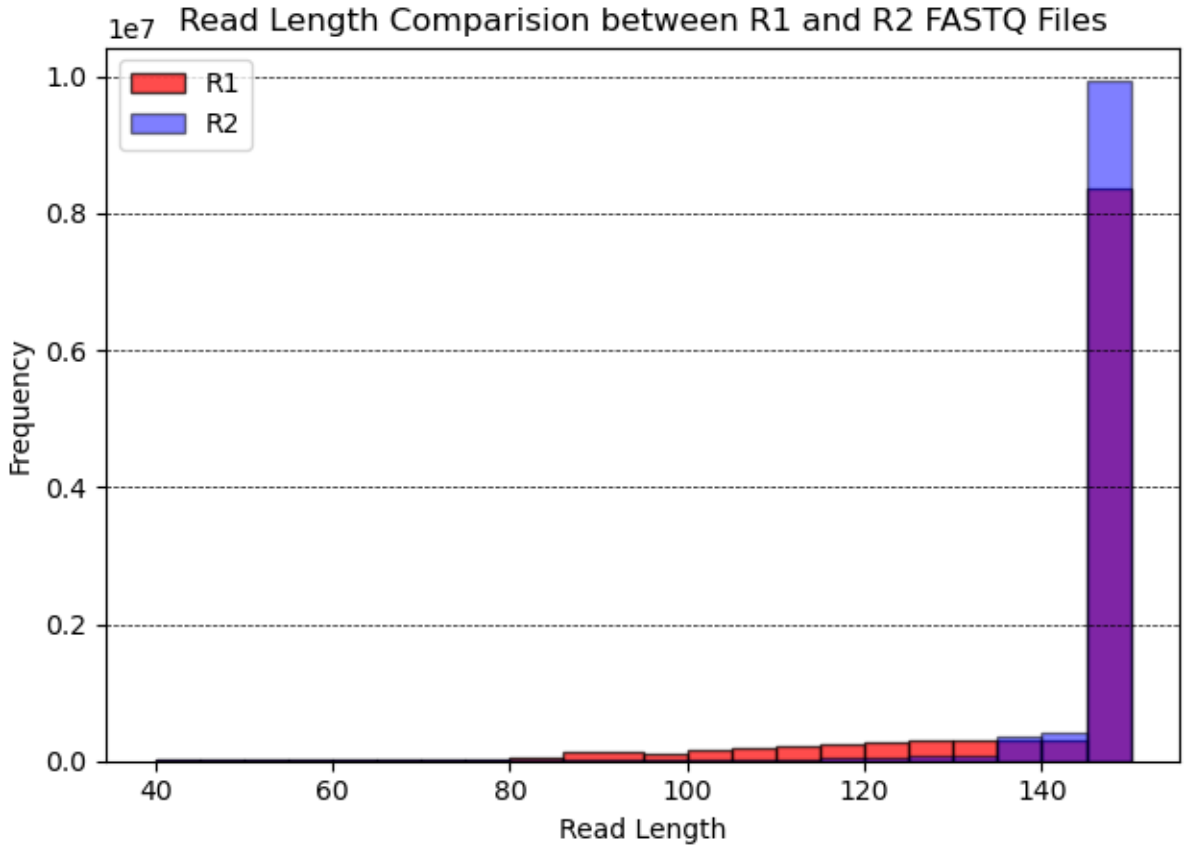
```
zcat FASTQFILE | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" --color=always
zcat FASTQFILE | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" --color=always
```

```
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -o trimmed_cco_com124_EO_6cm_1_R1.fastq.gz fastq/cco_com1
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -o trimmed_crh_rhy115_EO_adult_2_R1.fastq.gz fastq/crh_rhy
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -o trimmed_cco_com124_EO_6cm_1_R2.fastq.gz fastq/cco_com1
cutadapt -a AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o trimmed_crh_rhy115_EO_adult_2_R2.fastq.gz fastq/crh_rhy
```

File	Percentage of Reads with Adaptors	Reads Trimmed
cco_com124_EO_6cm_1_R1.fastq.gz	25.3%	2856436
cco_com124_EO_6cm_1_R2.fastq.gz	9.1%	1031026
crh_rhy115_EO_adult_2_R1.fastq.gz	7.7%	2785605
crh_rhy115_EO_adult_2_R2.fastq.gz	8.4%	3017409

```
trimmomatic PE cutadapt_cco_com124_EO_6cm_1_R1.fastq.gz cutadapt_cco_com124_EO_6cm_1_R2.fastq.gz paired

trimmomatic PE cutadapt_crh_rhy115_EO_adult_2_R1.fastq.gz cutadapt_crh_rhy115_EO_adult_2_R2.fastq.gz pa
```

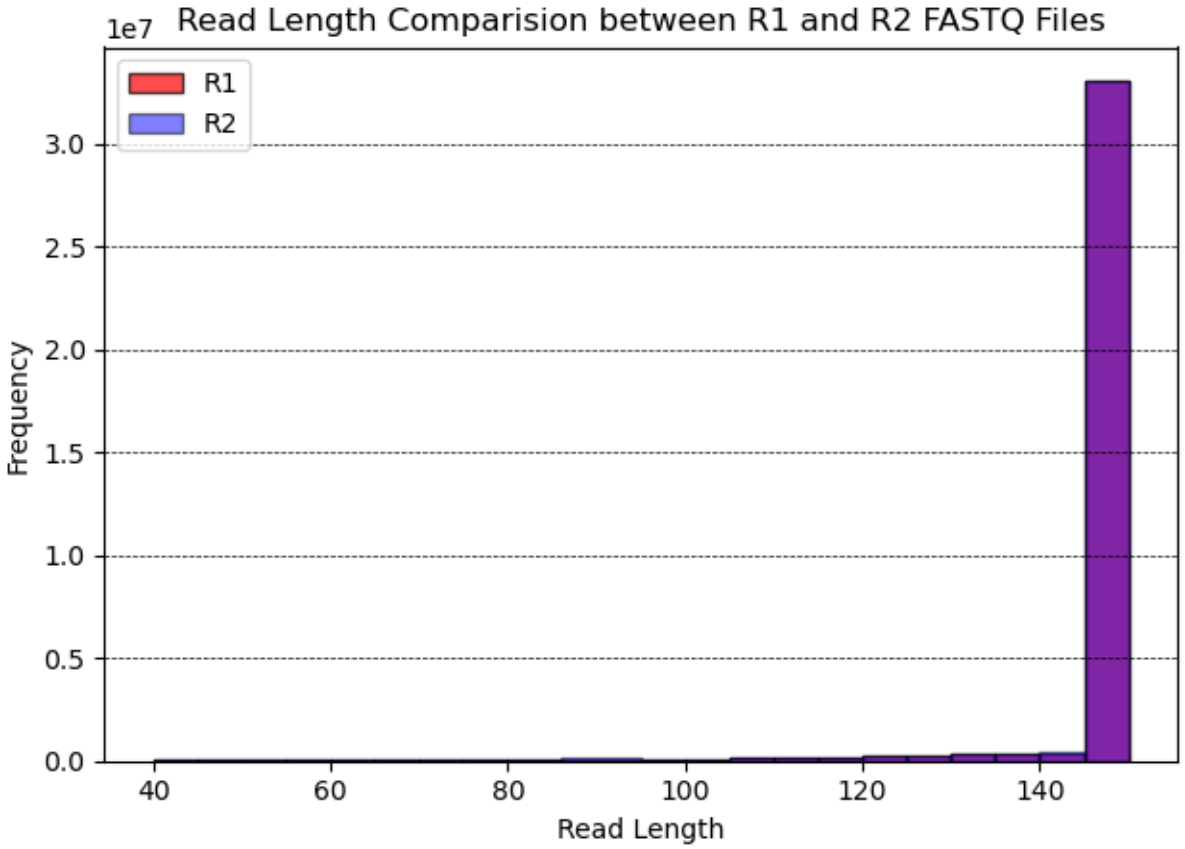
C. compressirostris



Caption: Histogram of read lengths after adaptor and trimmomatic trimming of C. compressirostris fastq files. Red represents the R1 reads, blue represents the R2 reads and purple is the overlap.

Looking at this plot, the R1 reads were trimmed at a much higher rate than the R2 reads, which were barely trimmed at all. This makes sense, because according to Trimmomatic, 98.99% of total read pairs survived. This means that Trimmomatic did not have a large effect on the read sizes, so we must look at the cutadapt output. The cutadapt output stated that the R1 had roughly 25% reads with adaptors removed, while R2 only had 8%.

C. rhynchophorus



Caption: Histogram of read lengths after adaptor and trimmomatic trimming of C. rhynchophorus fastq files. Red represents the R1 reads, blue represents the R2 reads and purple is the overlap.

When looking at this plot, you can see that the R1 and R2 files were trimmed at roughly the same rate. Looking at the Trimmomatic ouput, it says that 99.04% of reads survived. This means that we have to look at cutadapt, the R1 file had 9.1% of reads with adaptors and the R2 file had 8.4% of reads with adaptors trimmed. This data supports what the graph as output as both the R1 and R2 lengths were roughly the same.

Alignment and strand-specificity

To align the reads and determine strand specificity, I needed to download Star, Picard, Samtools, gffread and Htseq. Star was used to create the C. compressirostris reference database using a reference genome and a gtf file that was converted from a gff file using gffread. Next, we created SAM files using Star, that were sorted using Samtools, with duplicates removed by Picard. Htseq was used to count the number of occurances of each gene in the SRA files that I was given.

```
conda install bioconda::star
conda install bioconda::picard=2.18
conda install bioconda::samtools
conda install bioconda::htseq
# Already had matplotlib and Numpy installed
```

```
scp campylomormyrus.fasta joshkram@login.talapas.uoregon.edu:/projects/bgmp/joshkram/bioinfo/Bi623/PS/QA
scp campylomormyrus.gff joshkram@login.talapas.uoregon.edu:/projects/bgmp/joshkram/bioinfo/Bi623/PS/QAA
conda install bioconda::gffread
gffread campylomormyrus.gff -T -o campylomormyrus.gtf
sbatch slurm-scripts/star-make-database.sh
sbatch slurm-scripts/star-align-reads.sh
```

```
samtools sort cco_mapped/cco_mappedAligned.out.sam -o cco_mapped/sort_cco_mappedAligned.out.sam
samtools sort crh_mapped/crh_mappedAligned.out.sam -o crh_mapped/sort_crh_mappedAligned.out.sam
picard MarkDuplicates INPUT=cco_mapped/sort_cco_mappedAligned.out.sam OUTPUT=cco_mapped/no_dups_cco_mapp
picard MarkDuplicates INPUT=crh_mapped/sort_crh_mappedAligned.out.sam OUTPUT=crh_mapped/no_dups_crh_mapp
```

Counted the number of mapped versus number of unmapped reads after sorting with Samtools and removing duplicates with Picard.

SAM File	Mapped Reads	Unmapped Reads
cco_com124_EO_6cm_1	11368479	2447441
crh_rhy115_EO_adult_2	33644471	2996185

```
sbatch slurm-scripts/htseq-count.sh
```

```
grep -v "^_" forward_cco.txt | awk '{sum += $2} END {print sum}'
# 272487

grep -v "^_" rev_cco.txt | awk '{sum += $2} END {print sum}'
# 5870632

grep -v "^_" forward_crh.txt | awk '{sum += $2} END {print sum}'
# 943076

grep -v "^_" rev_crh.txt | awk '{sum += $2} END {print sum}'
# 20039325
```

Strand Specificity

Organism	strand=yes Hits	strand=reverse Hits
C. compressirostris	272487	5870632
C. rhynchophorus	943076	20039325.

After analysis, our reads are strand specific, and the stranded parameter that should be used on this data when using htseq-count is –strand=reverse. RNA is single-stranded, and is created from the coding strand. This means that it should map to the non-coding strand, and the non-coding strand only. When I looked

at the forward and reverse stranded output, for Campylomormyrus compressirostris, the forward strand had 272487 occurances while the reverse strand had 5870632 occurances. This pattern continued with Campylomormyrus rhynchophorus, where the forward strand had 943076 occurances while the reverse strand had 20039325. Biologically, it makes sense that there would be more hits for the reverse strand than the forward strand. As previously stated, RNAs are the reverse complement of the coding strand, which is also known as the forward strand, so it would be the same as the reverse strand. This was supported by the data, which had more hits when using –strand=reverse for both organisms.