QAA PDF Analysis Report

Kaden Coulter

2025-04-25

RNA-seq Quality Assessment Report

PLEASE NOTE: Tables in this .pdf report have been inverted so that they render as pdf properly.

Data set:

Here, we process electric organ and/or skeletal muscle RNA-seq reads for a future differential gene expression analysis. We will be working with 2 RNA-seq files from two different electric fish studies (PRJNA1005245 and PRJNA1005244). The methods for the PRJNA1005244 data set are published and the methods for the PRJNA1005245 data set are written in the third chapter of a thesis.

SRR25630309 -> Cco_com123_EO_6cm_1 and SRR25630393 -> Crh_rhy116_EO_adult_3 are the selected files.

Part 1 – Read quality score distributions

Before beginning, I created a new conda environment called QAA and with FASTQC, cutadapt, and Trimmomatic. Using the tool FASTQC, I produced plots of the per-base quality score distributions and per-base N content for R1 and R2 reads. The full FASTQC HTML files can be found in the figures_html directory within this repo. These plots can be seen below, along with interpretation for each of the file outputs.

Cco_com123_EO_6cm_1_1_FASTQC.html:

For the Cco_com123_EO_6cm_1_1 per-base quality score distribution, we see consistent quality scores across all bases. There is some slight improvement from 1-3 bp and then an extremely gradual decline through 150, with a quality score greater than or equal to 34. The per-base N content is also seemingly fantastic. Through all positions, 0% N content is maintained. That is, through all bases we have no N content. We can see clear consistency with the quality score plots.

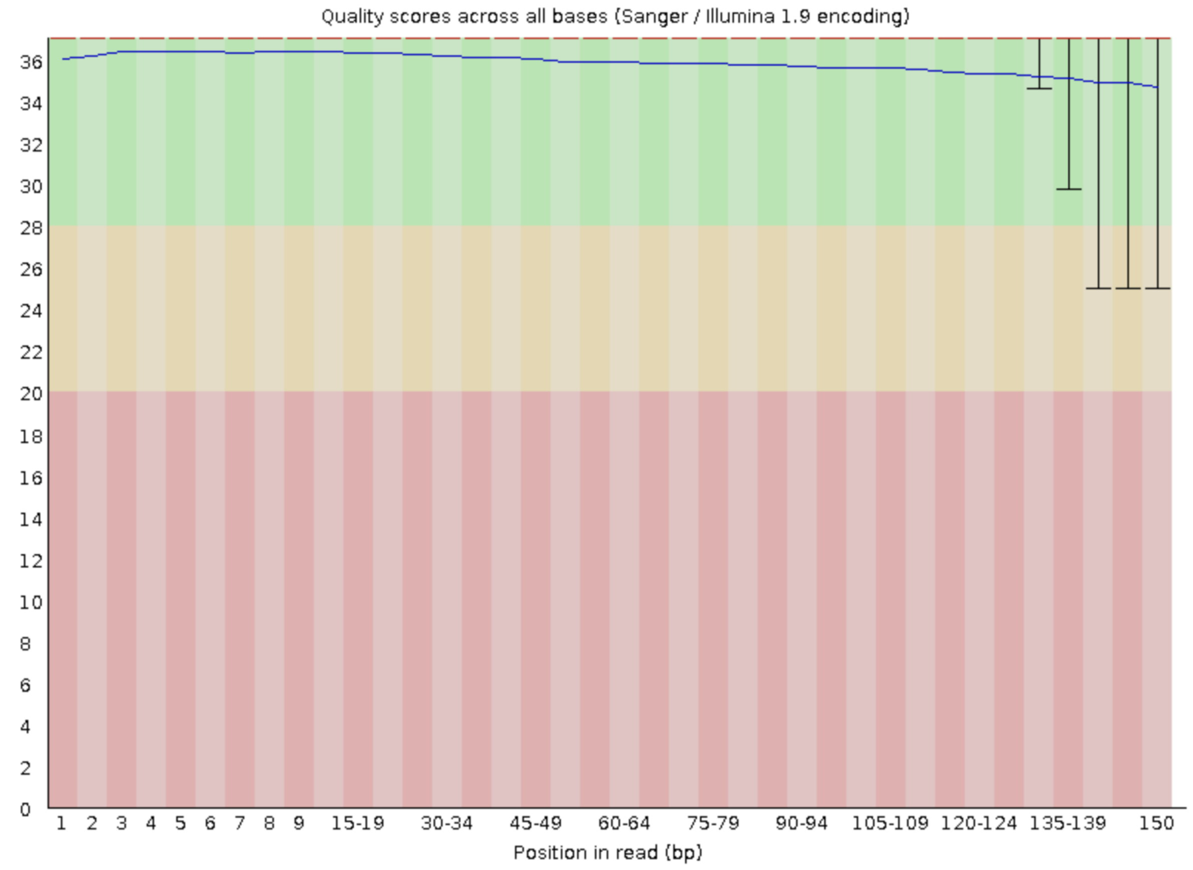


Figure 1: Mean Quality score (0-37) across all base positions in the read with error bars and color coded regions to denote high, potentially problematic, and problematic quality scores in Cco_com123_EO_6cm_1_1.

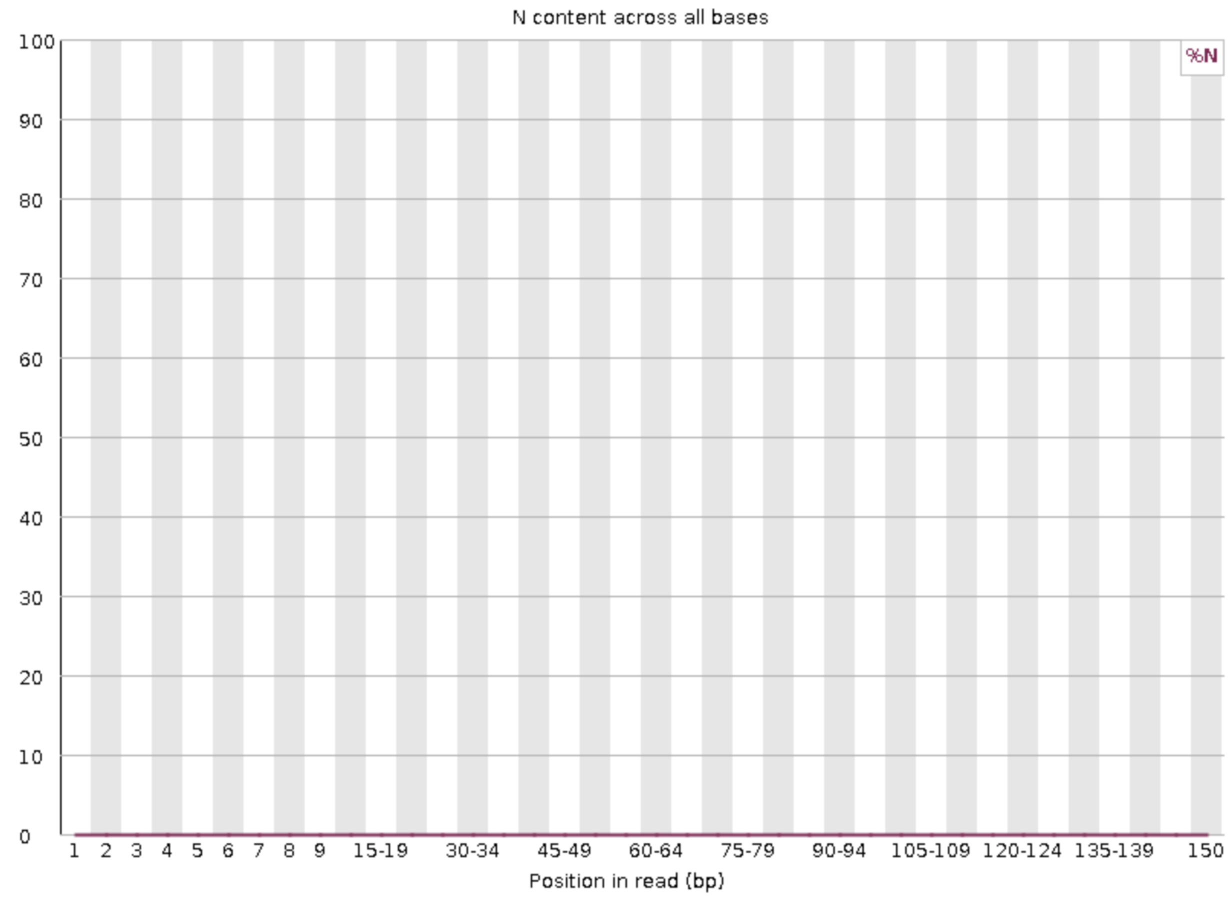


Figure 2: Percent of Unknown Reads (N) by base position in read of Cco_com123_EO_6cm_1_1.

Cco_com123_EO_6cm_1_2_FASTQC.html:

For the Cco_com123_EO_6cm_1_2 per-base quality score distribution, we see less consistent, but still great quality scores. We see a gradual decline from bp 1 through bp 150, dropping from 36 to what appears to be a Quality score of 33. In the per-base N content plot, we can, again, see 0% N content through all bases. We can see clear consistency with the quality score plots.

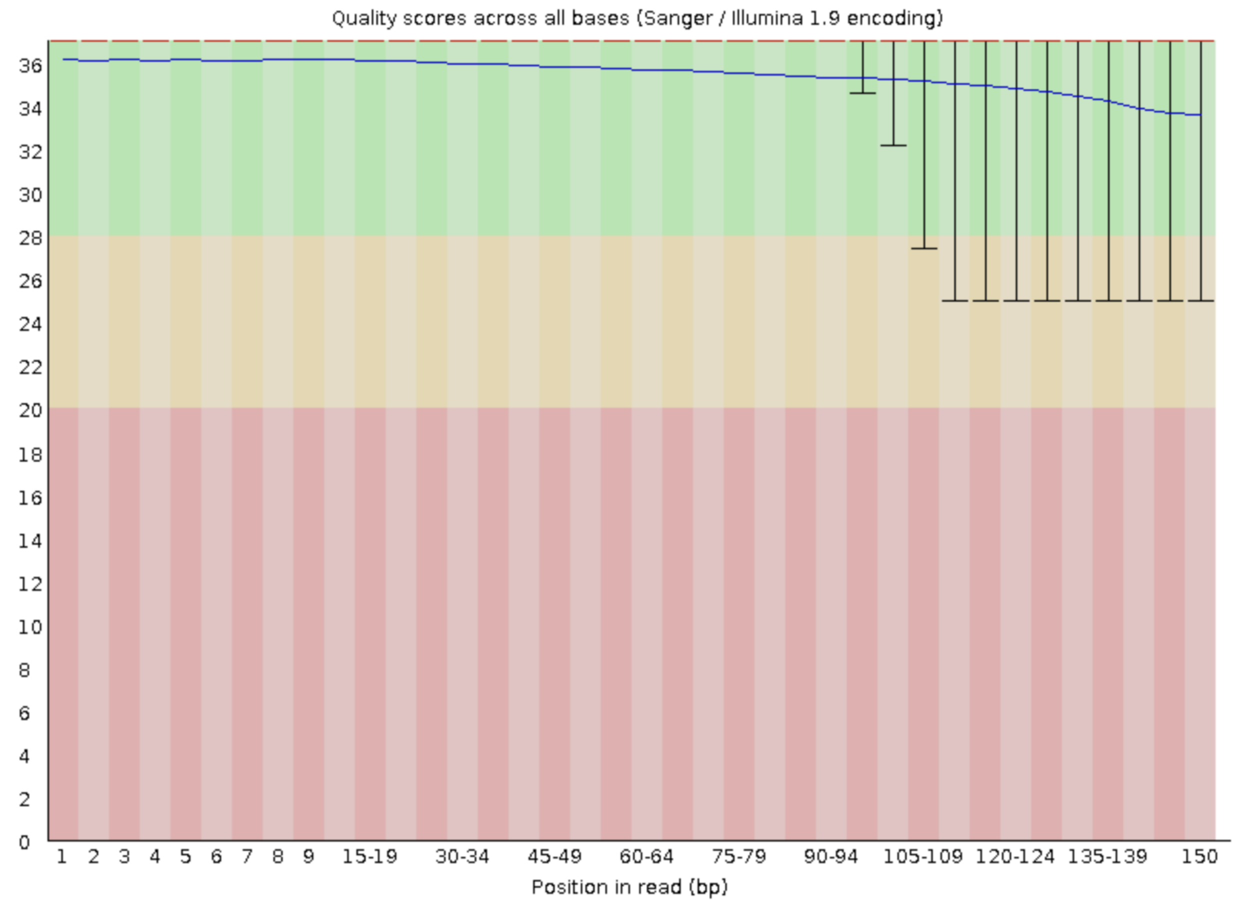


Figure 3: Mean Quality score (0-37) across all base positions in the read with error bars and color coded regions to denote high, potentially problematic, and problematic quality scores in Cco_com123_EO_6cm_1_2.

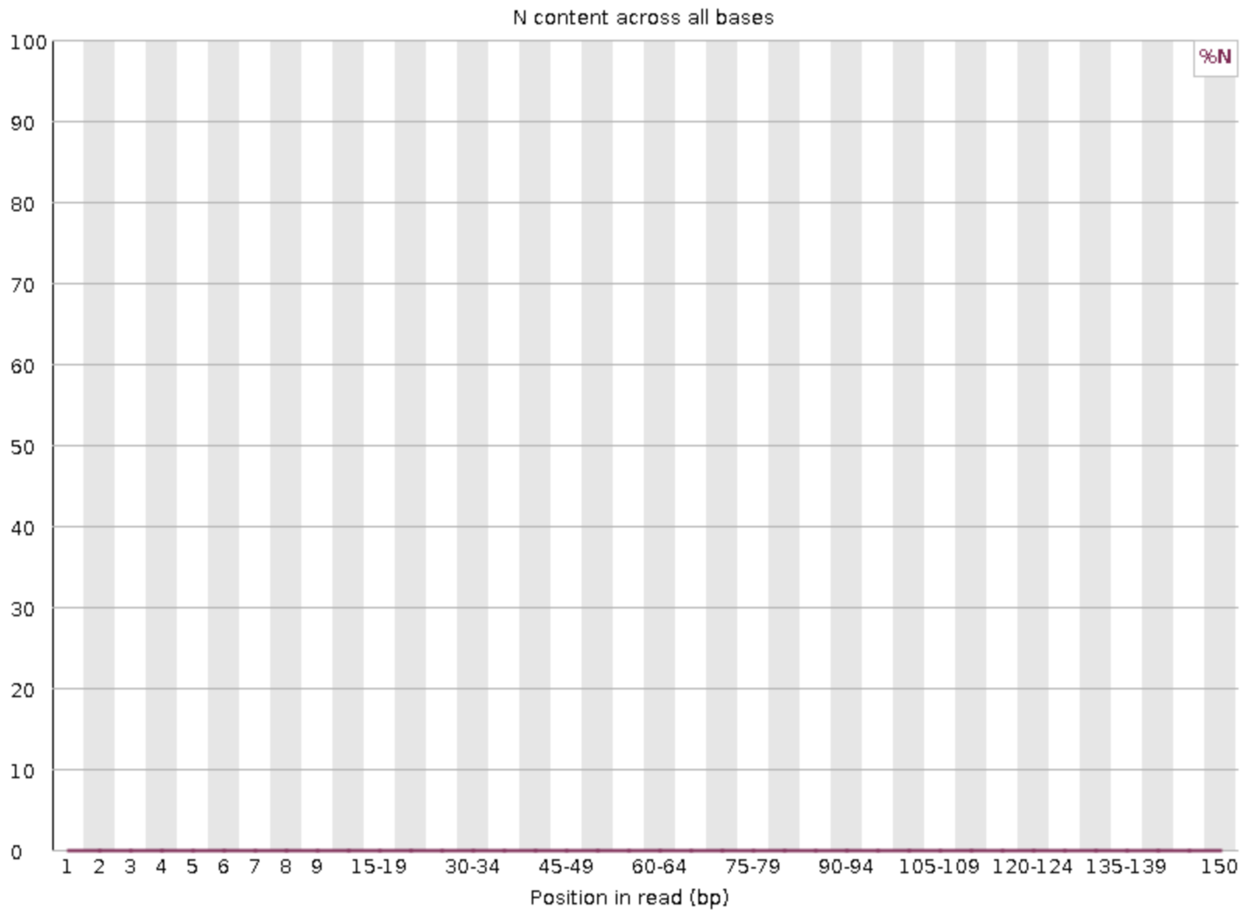


Figure 4: Percent of Unknown Reads (N) by base position in read of Cco_com123_EO_6cm_1_2.

Crh__rhy116__EO__adult__3__1__FASTQC.html:

For the Crh_rhy116_EO_adult_3_1 per-base quality score distribution, we, once again, see consistent quality scores across all bases.There is some slight improvement from 1-5 bp followed by an extremely gradual decline through bp 150, with quality scores in the 34-36 range. When examining the per-base N content, we, again, see 0% N content through all bases and maintain consistency with the quality score plots.

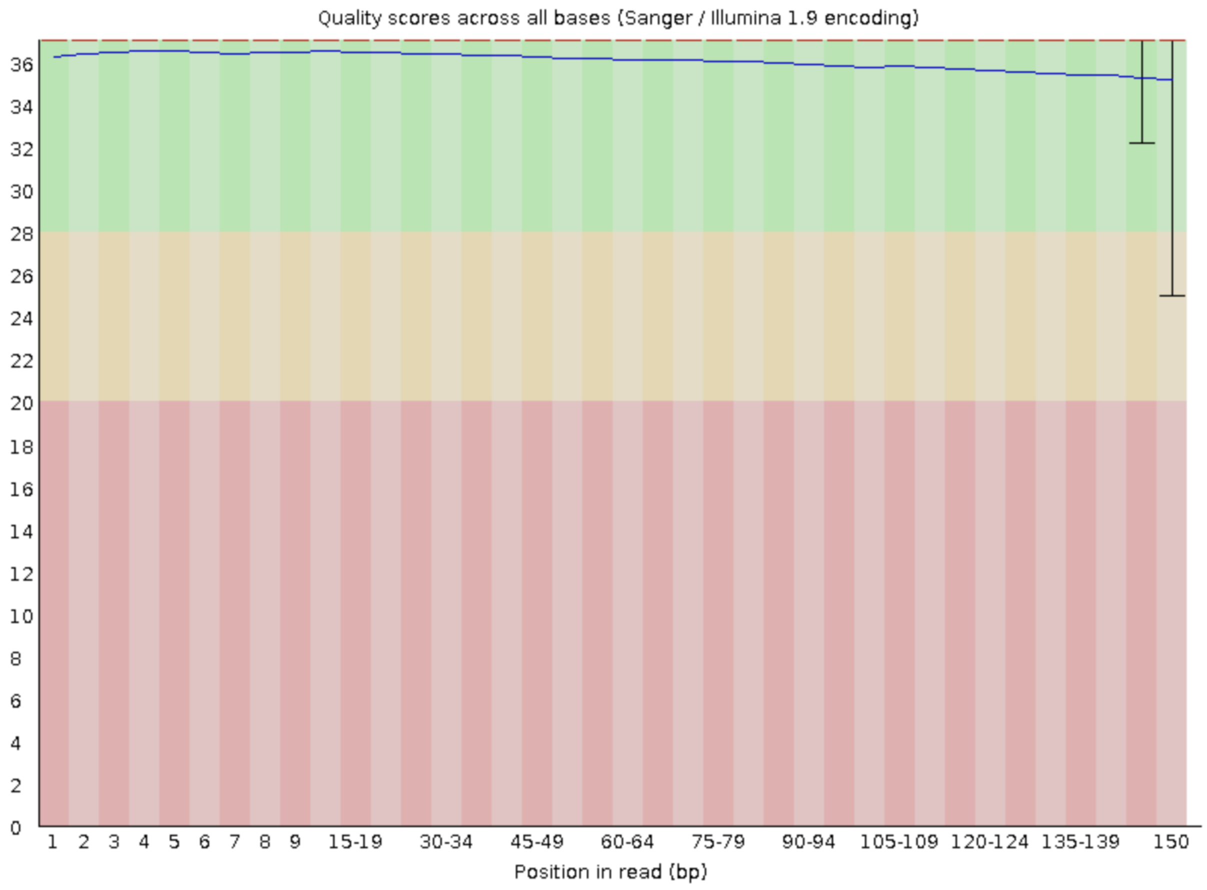


Figure 5: Mean Quality score (0-37) across all base positions in the read with error bars and color coded regions to denote high, potentially problematic, and problematic quality scores in Crh_rhy116_EO_adult_3_1_bq.

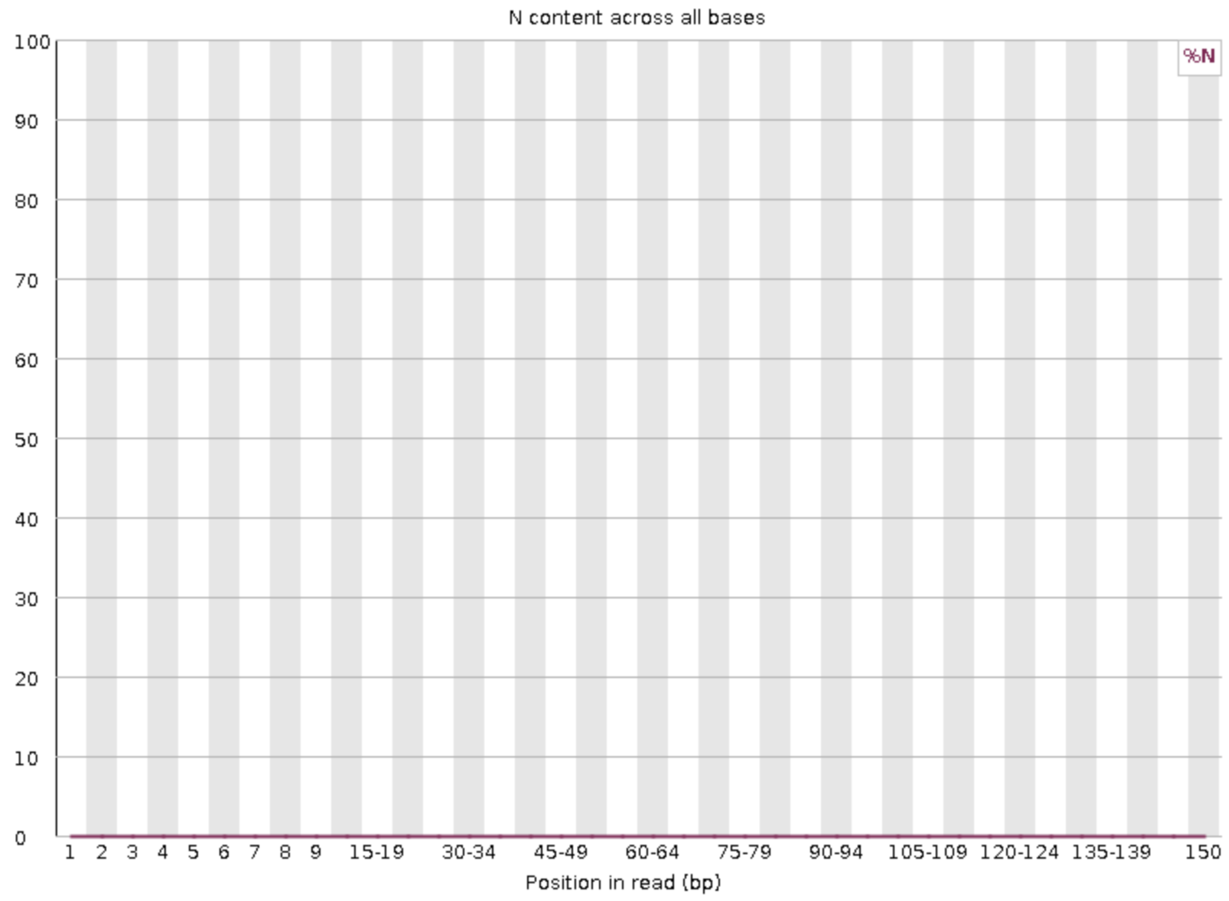


Figure 6: Percent of Unknown Reads (N) by base position in read of Crh_rhy116_EO_adult_3_1n.png.

Crh_rhy116_EO_adult_3_2_FASTQC.html:

For the Crh_rhy116_EO_adult_3_2_FASTQC, we, once again, see a consistent quality score of 34-36 across all bp in the per-base quality score distribution. Once again, is a very gradual decline over bp in the quality score. Again, we see a per-base N content consistent with our quality scores. However, we see a small spike at bp1, seemingly demonstrating a 1% N Content at bp1. Even so, this is consistent with the other plot.

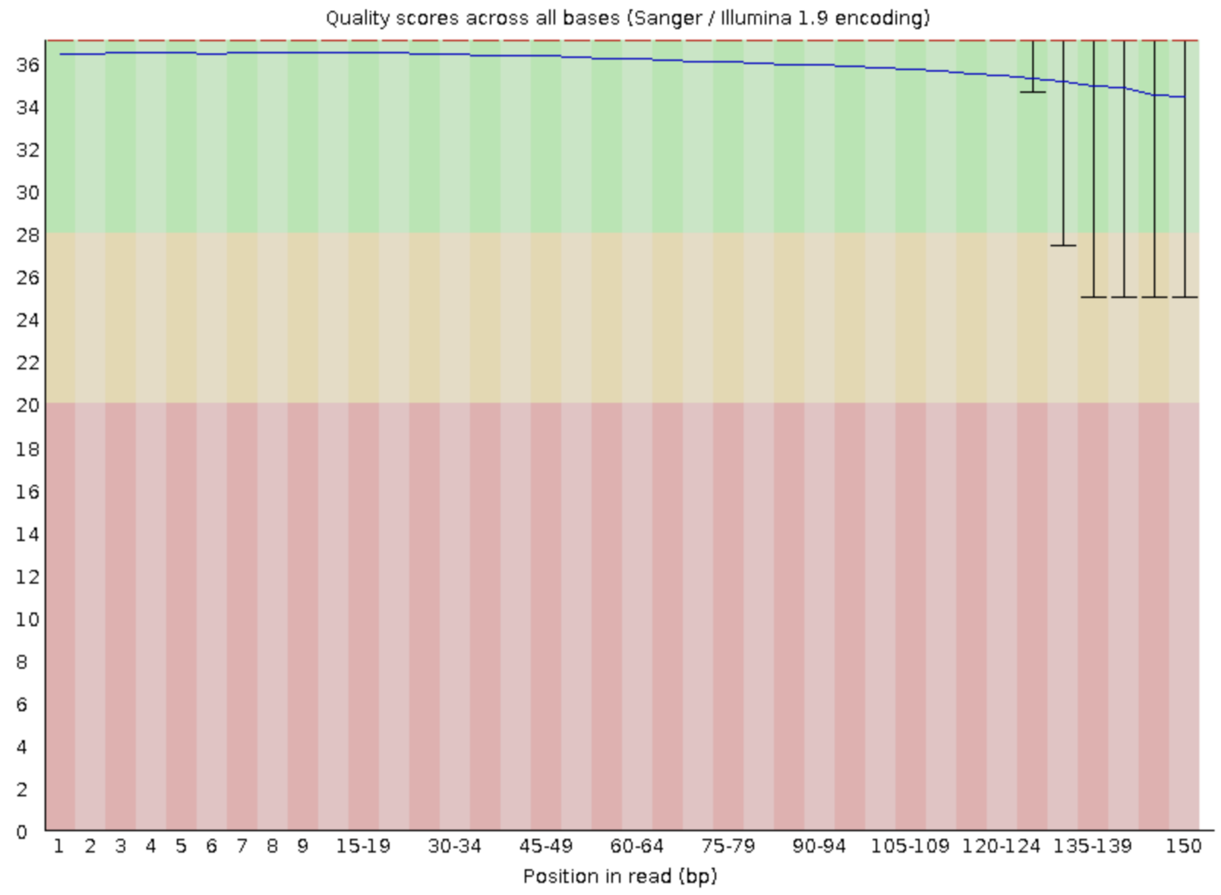


Figure 7: Mean Quality score (0-37) across all base positions in the read with error bars and color coded regions to denote high, potentially problematic, and problematic quality scores in Crh_rhy116_EO_adult_3_2_bq.

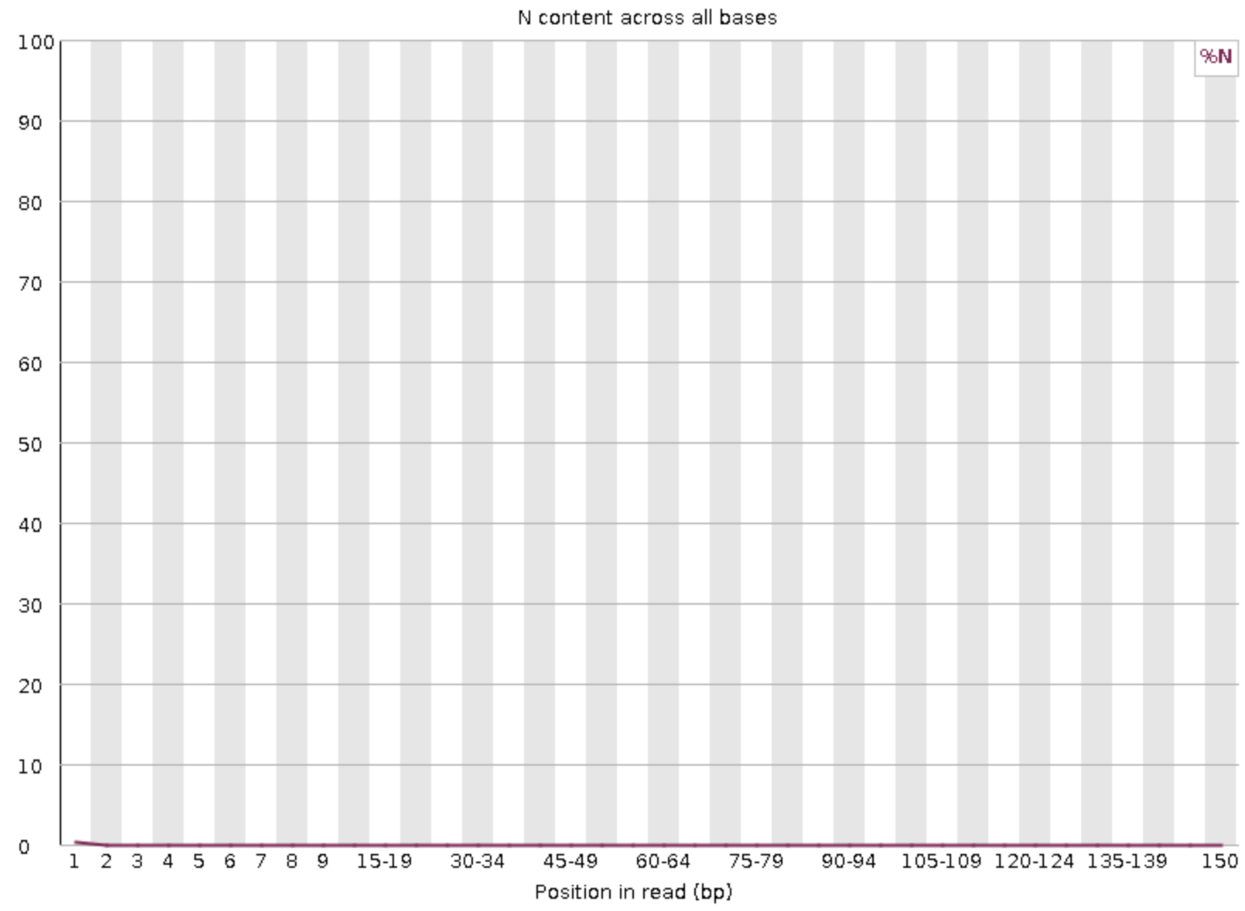


Figure 8: Percent of Unknown Reads (N) by base position in read of Crh_rhy116_EO_adult_3_2n.png.

Non-FASTQC Plots and Interpretation

After this, I copied and ran an altered quality score plotting script from Demultiplexing called qual_by_nuc_slurm.sh, which implements the running sum strategy, to generate a figure of the quality score by base position figure. The plot I created and the quality score plots from FASTQC look similar (as far as values), but my process was faster than the entire FASTQC output. However, at the beginning of the reads, my plot shows a slightly higher quality score. This is because my script does not take into account any adapters added to the beginning of the reads. So, I am artificially inflating these mean values and offsetting my reads. Still, my plot is comparable to FASTQC. It is worth noting that ff this data wasn't high quality, this could become an enormously important oversight.

While my run time was shorter (3:12.14 minutes to approximately 15 minutes), my process may not actually be as fast or resource efficient as FASTQC. My script is written in Python, which is much slower than the languages FASTQC is likely built in, such as Java or C. In addition, my script prioritizes logic and readability, while FASTQC is clearly optimized for speed and resource efficiency. I'm also only generating a single plot, not multiple figures, formatting them into an HTML report, or producing detailed user messages and warnings like FASTQC does. While other factors also play a role in the higher memory usage, CPU load, and run time of my script, these are the main reasons it doesn't appear faster than FASTQC.

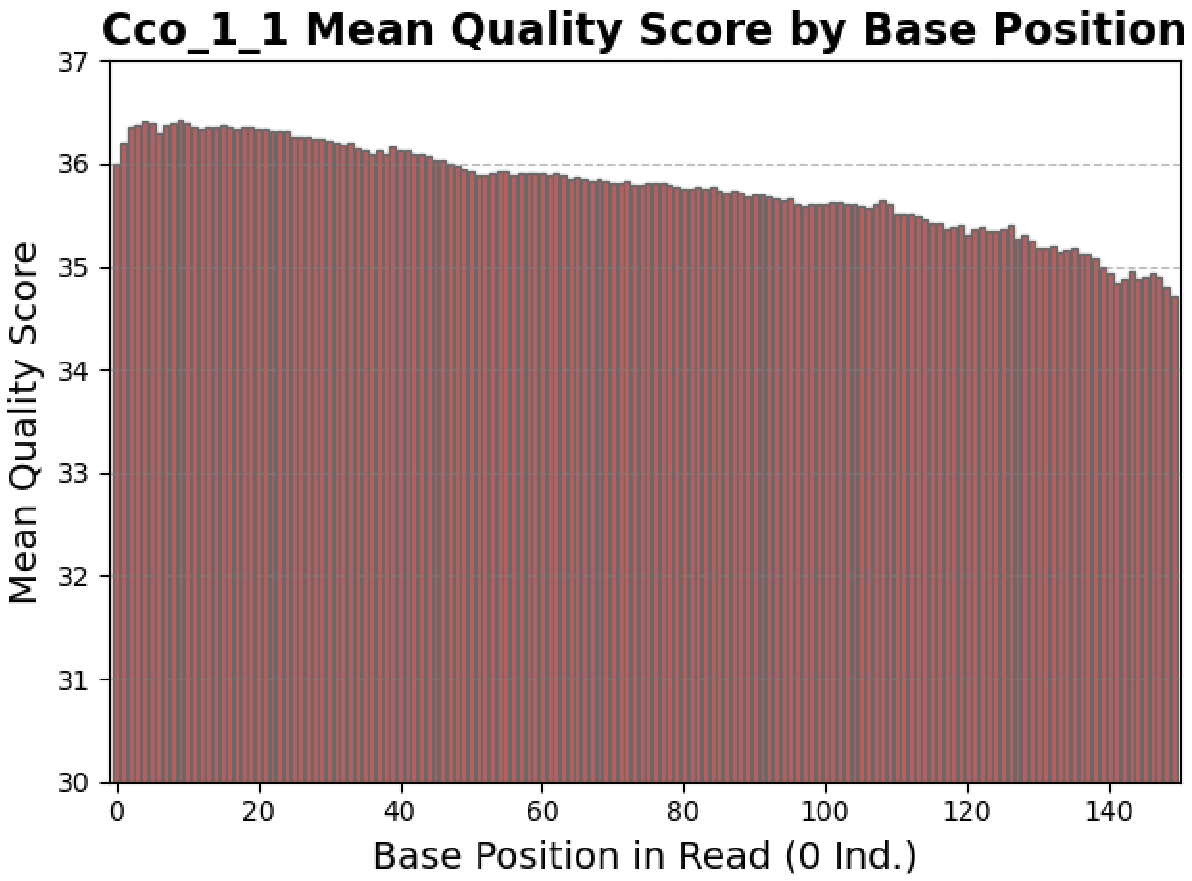


Figure 9: Mean quality score (30 - 37) by base position in read, 0 indexed. Each bin is representative of a single base within the read.

Overall data quality of libraries:

Using the `FASTQC` documentation results from FASATQC were interpreted as follows:

- Per Tile Sequence Quality: !

  In Cco_com123_EO_6cm_1_1, we see a bit of decline in quality per tile as we reach the end of our reads. However, this is very minor, and displays no effect on the read quality in any of the other measures. In our sequence quality plots, at these base positions, we see more variance in the quality score, but a more than acceptable score. So, there is nothing to worry about in this warning.

- Per Base Sequence Content: X

  In all files, we see Per base sequence content flagged as failure. However, this is just our bar codes at the beginning of the reads. This warning can be dismissed.

- Per Sequence GC content: !

  In Cco_com123_EO_6cm_1_1, GC content is sightly left skewed from the normal distribution that is expected. However, we still create a normal distribution that is only *slightly* offset from what is expected. We see a bit of noise in the peak, but, again, this is not something to be overly concerned with and this warning can be dismissed.

- Sequence Duplication Levels: X

  In every file we get Sequence Duplication Levels flagged as failure. However, this is expected in RNA-seq, which we have here. Due to highly expressed transcripts, paired end reads in our sequencing process, and fragment size bias, this can be expected. We can dismiss this warning.

- Adapter Content: X

  In every file, we get an adapter content warning flagged as failure. Again, this is an artifact or leftover of RNA-seq and can be dismissed. We have short fragments, causing us to sequence the adapter at the end of our reads. We have a bit of contamination at the end of our reads, suggesting our adapters are a bit long. This is expected in illumina RNA-seq and can be dismissed.

Can We Use This For Further Analysis? Based on the data quality and the dismissal of all raised warnings, we can continue to analyze this data. Everything looks as expected for the type of sequencing we used to generate this data, the data is high quality, and nothing seems questionable or suspicious. Continue with analysis!

Part 2 – Adapter trimming comparison

Using `Cutadapt` to properly trim adapter (Illumina TruSeq) sequences from files:

We obtain the following proportion of reads trimmed from our files.

File	Read	Trimmed. . . .
Cco_com123_EO_6cm_1_1.fastq.gz	R1	15.8%
Cco_com123_EO_6cm_1_2.fastq.gz	R2	16.3%
Crh_rhy116_EO_adult_3_1.fastq.gz	R1	13.1%
Crh_rhy116_EO_adult_3_2.fastq.gz	R2	13.7%

Following this, I conducted a sanity check confirming the expected sequence orientations.

When we switch adapter and file pairings (R1 adapter with R2 file) we find 0 matches in the file. However, a search maintaining what we now know is the correct pairing yields many matches, indicating that this pairing is correct. Still, it should be noted that is is possible to get 0 with a match, assuming every adapter was partially cut off. However, this incredibly unlikely. Even more so when compared to the number of what would be non-adapter matches in the read of the other files. More so considering that adapters are intentionally selected, as they do not occur naturally. Basically, there is some chance that this method could be wrong, but it is incredibly unlikely. Keeping this in mind, we pass our sanity check.

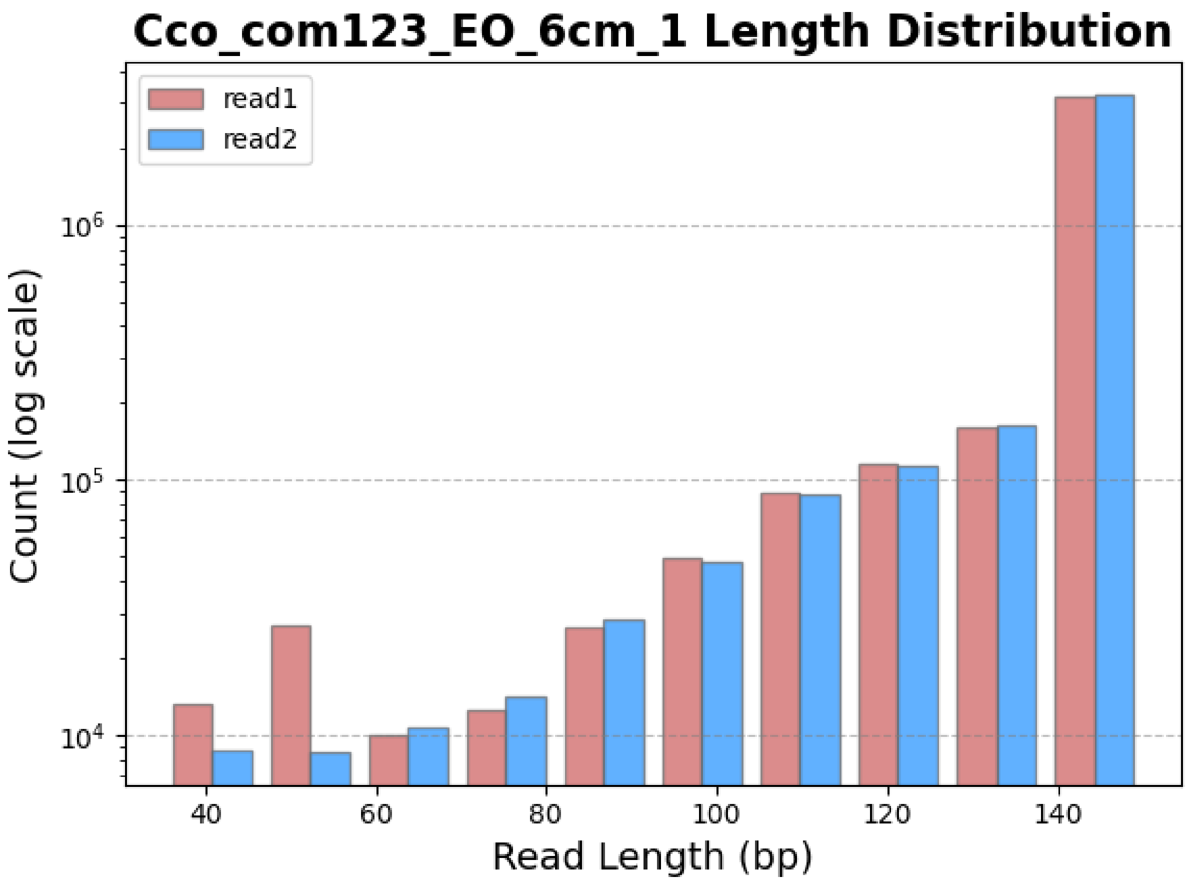
Also, note the difference in proportions between these results and cutadapt is because here, we are searching for the *entire* adapter, which can often be cutoff and/or lost in sequencing.

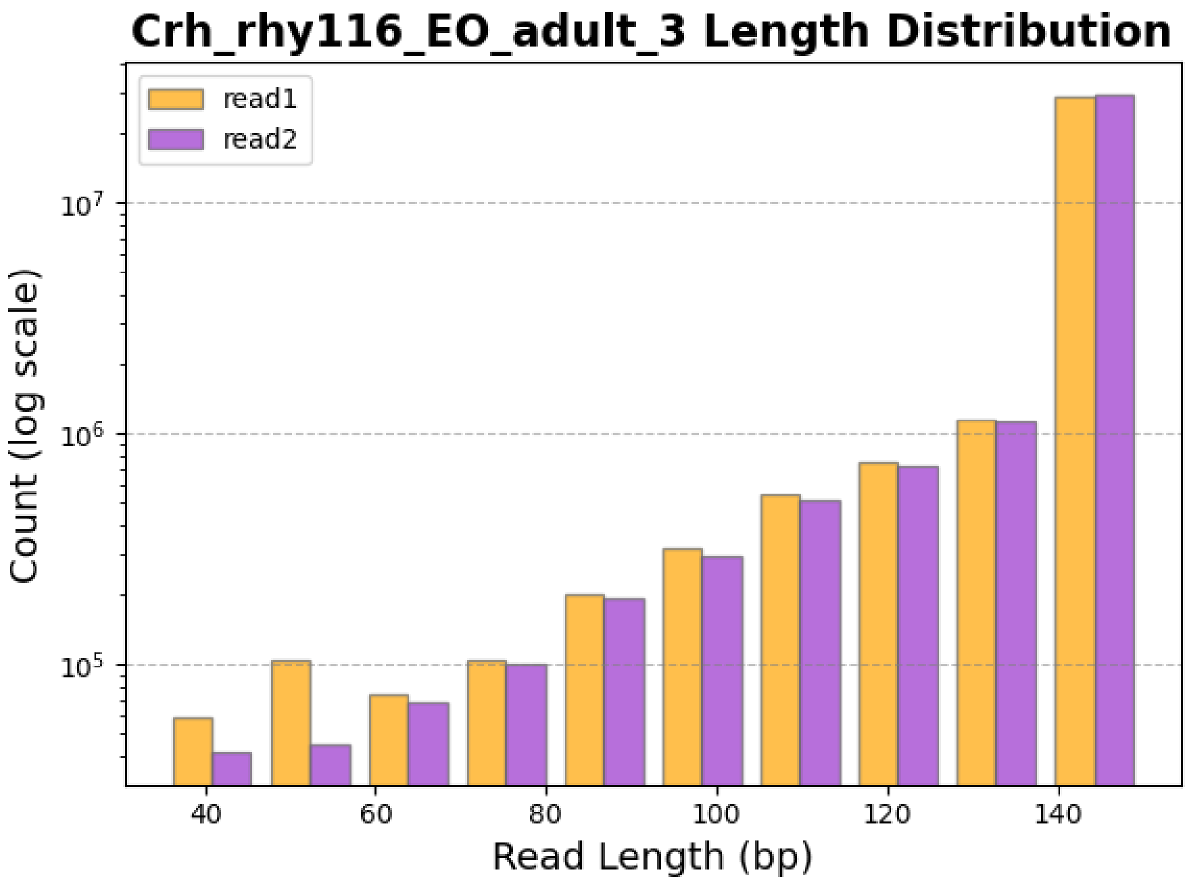
Using Trimmomatic to quality trim reads.

Following this, I used trimmomatic to quality trim reads, with a leading quality of 3, a trailing quality of 3, a sliding window size of 5 with required quality of 15 and a minimum length of 35 bases. This yielded the values shown in the table below. Following this, I plotted the trimmed read length distributions for both paired R1 and paired R2 reads using the slurm script read_len_dist_slurm.sh which uses the script read_len_dist.py.

Metric	Cco_com123_EO_6cm_1	Crh_rhy116_EO_adult_3
Input Read Pairs	3735980	32594028
Both Surviving	3695585 (98.92%)	32425688 (99.48%)
Forward Only Surviving	29316 (0.78%)	111529 (0.34%)
Reverse Only Surviving	9603 (0.26%)	51935 (0.16%)
Dropped	1476 (0.04%)	4876 (0.01%)

!!! Cco_com123_EO_6cm_1_readlen.png Crh_rhy116_EO_adult_3_readlen.png !!!


Cco_com123_EO_6cm_1 Length Distribution
read1
read2
Count (log scale)
10^6
10^5
10^4
40
60
80
100
120
140
Read Length (bp)



In the plots, we can see that read 2 is generally shorter than read 1 after trimming. This is likely because we cutoff more low quality reads at the 3' end, cutting more bases and adapters for R2, making the reads shorter than R1. R2 is expected to be adapter trimmed at a higher rate than R1.

Part 3 – Alignment and strand-specificity

Genome Download

At this point, I Downloaded the publicly available *Campylomormyrus compressirostris* genome fasta and gff, available on Dryad, from `/projects/bgmp/shared/Bi623/PS2/campylomormyrus.fasta`, `/projects/bgmp/shared/Bi623/PS2/campylomormyrus.gff`. I then Created a database from using the slurm script: database_builder.sh and aligned the reads to the *C. compressirostris* genome using STAR via the slurm script: starliner.sh.

Afterwards, I removed PCR duplicates using Picard. To do this, I needed to sort my reads with samtools before running picard. The current version of picard requires alternate setup. So, I activated a new environment called stupidpicard (naturally) with Picard2.18 and Python3.60 to safely use the tools. When running picard, I removed duplicates and set validation stringency to be lenient.

Metric	Cco_com123	Crh_rhy116
Library	Unknown Library	Unknown Library
UNPAIRED_READS_EXAMINED	7738	18939
READ_PAIRS_EXAMINED	3412953	31081660

Metric	Cco_com123	Crh_rhy116
SECONDARY_OR_SUPPLEMENTARY_RDS	1105531	3946526
UNMAPPED_READS	558103	2670143
UNPAIRED_READ_DUPLICATES	4710	15466
READ_PAIR_DUPLICATES	1280227	15159849
READ_PAIR_OPTICAL_DUPLICATES	0	0
PERCENT_DUPLICATION	0.375373	0.487843
ESTIMATED_LIBRARY_SIZE	3321328	20327822

After this, I calculated the number of mapped and unmapped reads from each of the 2 SAM files post deduplication with picard. The script sam_reporter.py was copied from BI621PS8, altered to accept multiple file inputs simultaneously, and checked to ensure validation of primary or secondary mapping using the bit wise flag. Afterwards, I determined that we have:

File	Mapped	Unmapped
Campy_Cco_com123_sort_marked.sam	4268480	557526
Campy_Crh_rhy116_sort_marked.sam	31847095	2669117

Following this, I counted deduplicated reads that map to features using htseq-count. I ran htseq-count twice, once set to stranded and once set to reverse stranded. Beyond this, I used default parameters.

After this, I had to determine whether or not the data are from strand-specific RNA-Seq libraries **and** which stranded parameter in htseq is necessary for later differential gene expression analyses.

To do this, I created strand_tester.py, which accepts stranded and reverse files and provides some insight on which parameter and output are more appropriate. It is worth noting that this script does not consider the possibility of not strand specific libraries. Using this script, I determined that 928660 stranded reads and 19659636 reverse stranded reads were assigned for Crh_rhy116_EO_adult_3 and that 97989 stranded reads and 2026304 reverse stranded reads were assigned for Cco_com123_EO_6cm_1.

Because –stranded=reverse assigned 63.2% of Crh_rhy116_EO_adult_3 and 59.4 % of Cco_com123_EO_6cm_1 reads to features, compared to –stranded=yes 3.0% of Crh_rhy116_EO_adult_3 and 2.9 % of Cco_com123_EO_6cm_1 reads, I conclude that this data is string-specific with the correct parameter for future analysis being –stranded=reverse. If more reads are aligned with –stranded=reverse, we know that our strands must be reverse stranded, meaning that the reads map to the opposite strand as the gene.

Note, this –stranded=reverse conclusion only applied to this data. Other contexts, this conclusion may very well be false.

REPO ORGANIZATION

Within this Repo:

- Lab Notebook
- scripts
- figures
- htmls
- htseq_files
- QAA_report.pdf *Also on Canvas*