QAA_Report

Thejes Nair

2025-09-08

Note: Commands located in .Rmd AND Lab notebook. QAA_Report includes figures, tables, and answers to questions.

RNA-seq Quality Assessment Assignment - Bi 623 (Summer 2025 Assignment/PS 2)

Part 1 – Read quality score distributions

1. FastQC plots: perbase quality score distribution and per base N content for R1 and R2 reads in both datasets (Cco, CcoxCrh).

FastQC plots: per base quality and per base n content

Fig1-2, fastqc output of per base quality in Cco R1 and R2. Different colored backgrounds represent good, intermediate, and low quality calls. Box and whisker plots show the upper and lower 10% and 90% points. The blue line represents the median.

Fig3-4, fastqc output of per base n content in Cco R1 and R2. Fastqc will plot the percentage of N per base, both R1 and R2 reads have a %N of close to or at 0.

Fig5-6, fastqc output of per base quality in CcoxCrh R1 and R2. Different colored backgrounds represent good, intermediate, and low quality calls. Box and whisker plots show the upper and lower 10% and 90% points. The blue line represents the median. Bases towards the end of the 150bp read lengths have a larger box whisker plot and expand into the poor quality.

Fig7-8, fastqc output of per base n content in Cco R1 and R2. Fastqc will plot the percentage of N per base, both R1 and R2 reads have a %N of close to or at 0.

The quality of the per base QS distribution is consistent in both plots R1 and R2 have a similar shape. Fastqc shows that the quality passes and is high across. QS starts at ~36-37 and tapers down to ~34-35, which is to be expected. The whiskers extend more towards the end but the mean reads still stay within the green range. The per base n content for both reads is at 0% even when the QS starts to decline towards the end of the read which supports the confidence of the results seen in the fastqc per base qs distribution plots.

2. Describe how the `FastQC` quality score distribution plots compare to your own. If different, propose an explanation. Also, does the run time differ? Mem/CPU usage? If so, why?

Running fastqc took <10 mins for each readpair (so ~20 mins total). My demux script took ~30 mins to run for all four readpairs. The CPU usage was the same between the two approaches, 97-99% but the memory needed for fastqc was much greater than the demux script (593452 kbytes vs ~ 64000-70000 kbytes). This makes sense because fastqc is generating much more informative plots and additional information than the single QS distribution plot my demux script does and does so with the same CPU %.

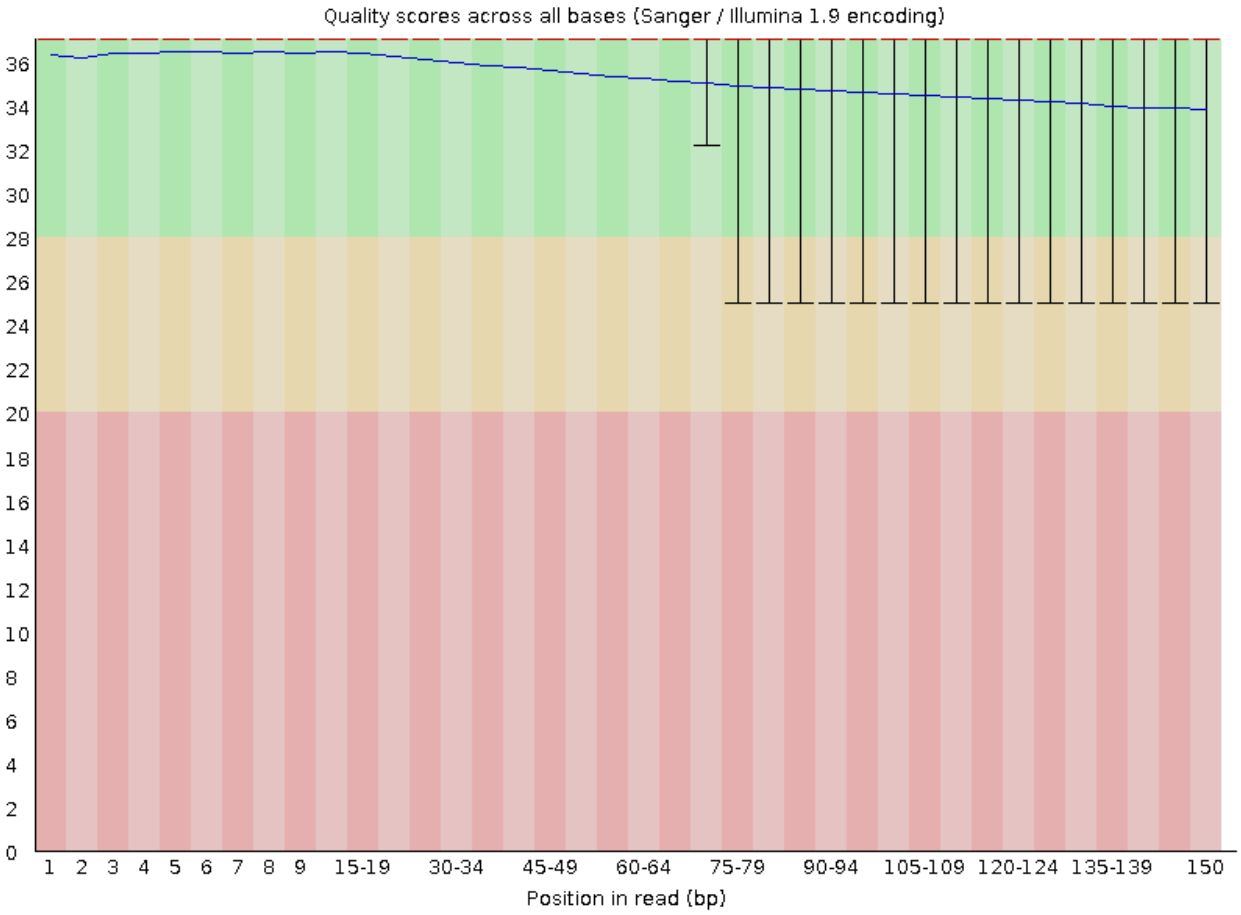


Figure 1: Cco per base quality R1

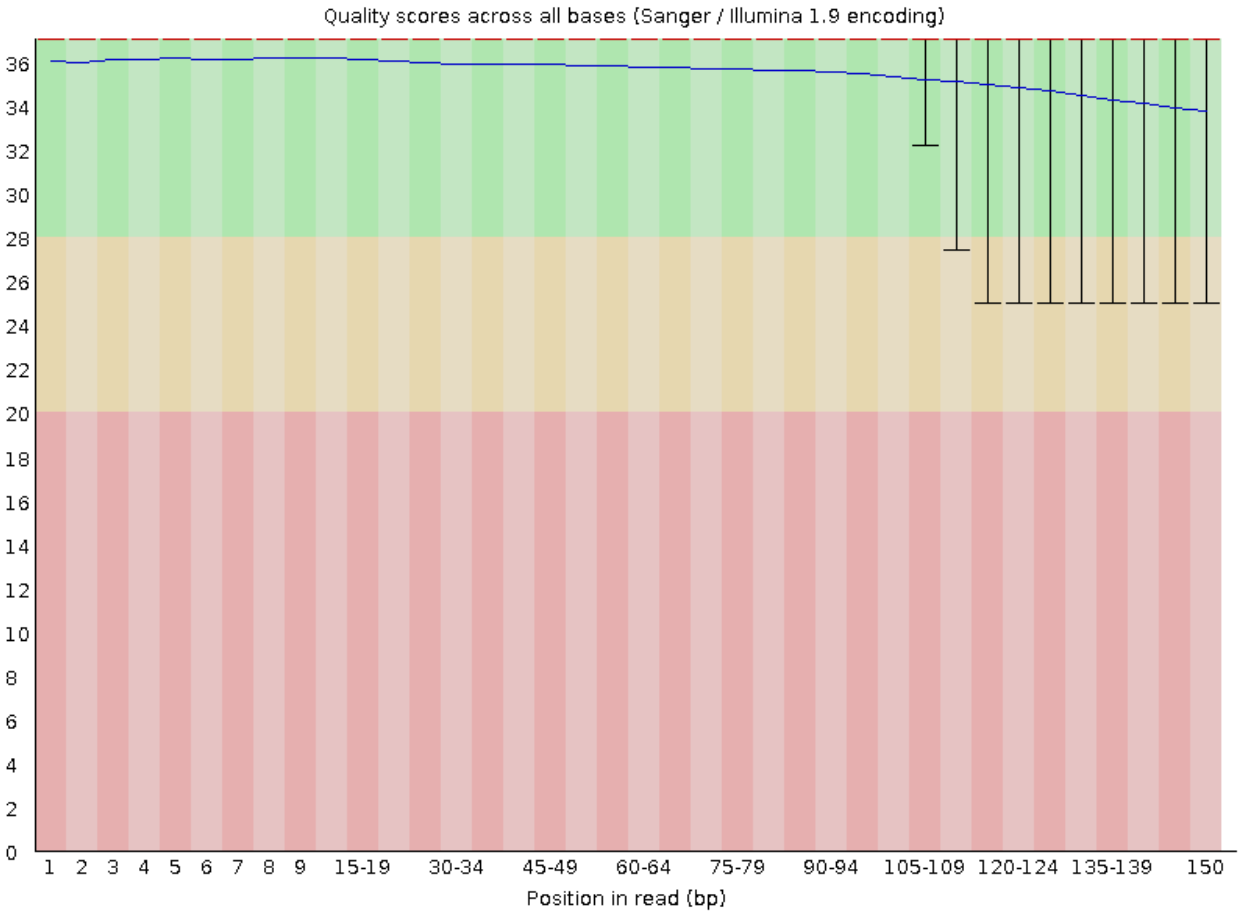


Figure 2: Cco per base quality R2

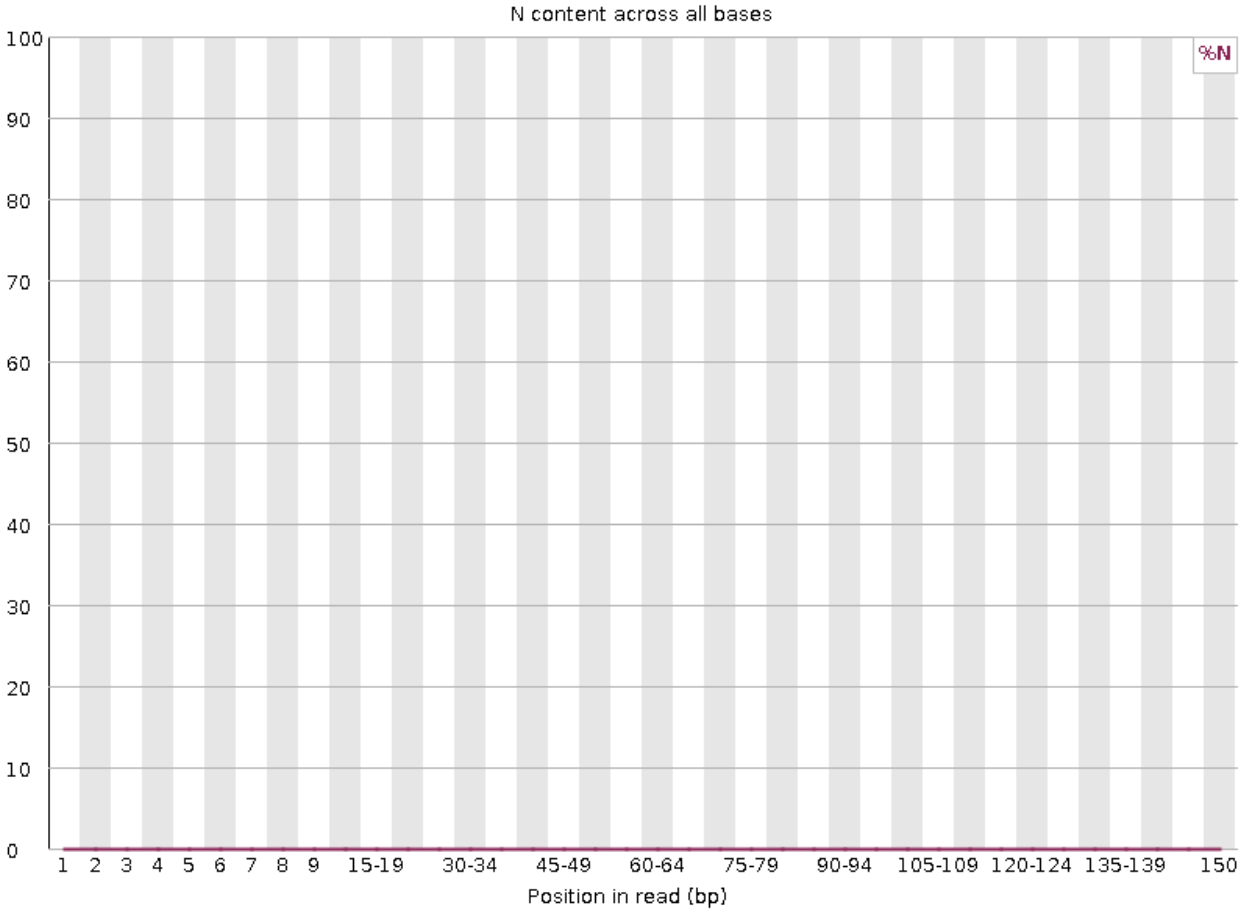


Figure 3: Cco per base n content R1

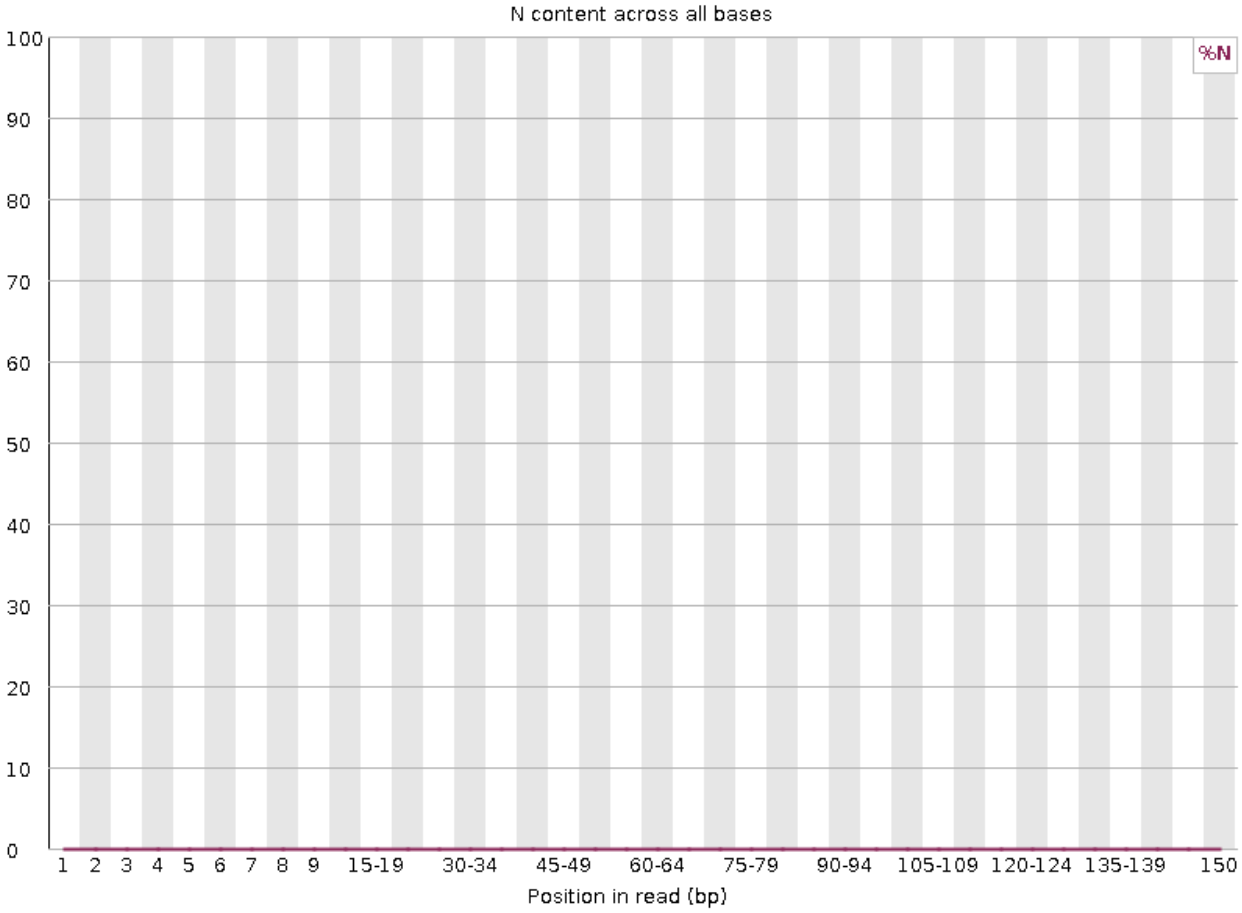


Figure 4: Cco per base n content R2

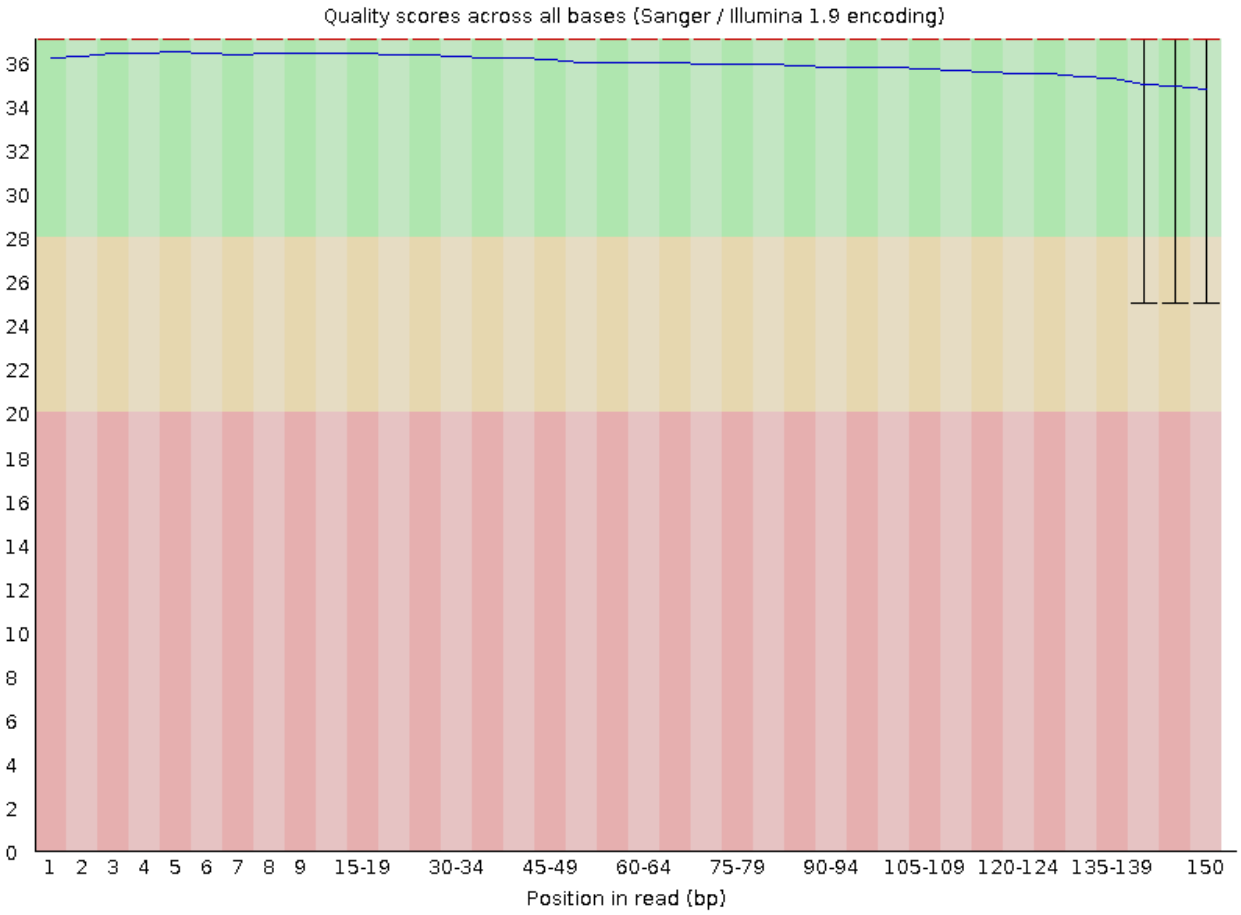


Figure 5: CcoxCrh per base quality R1

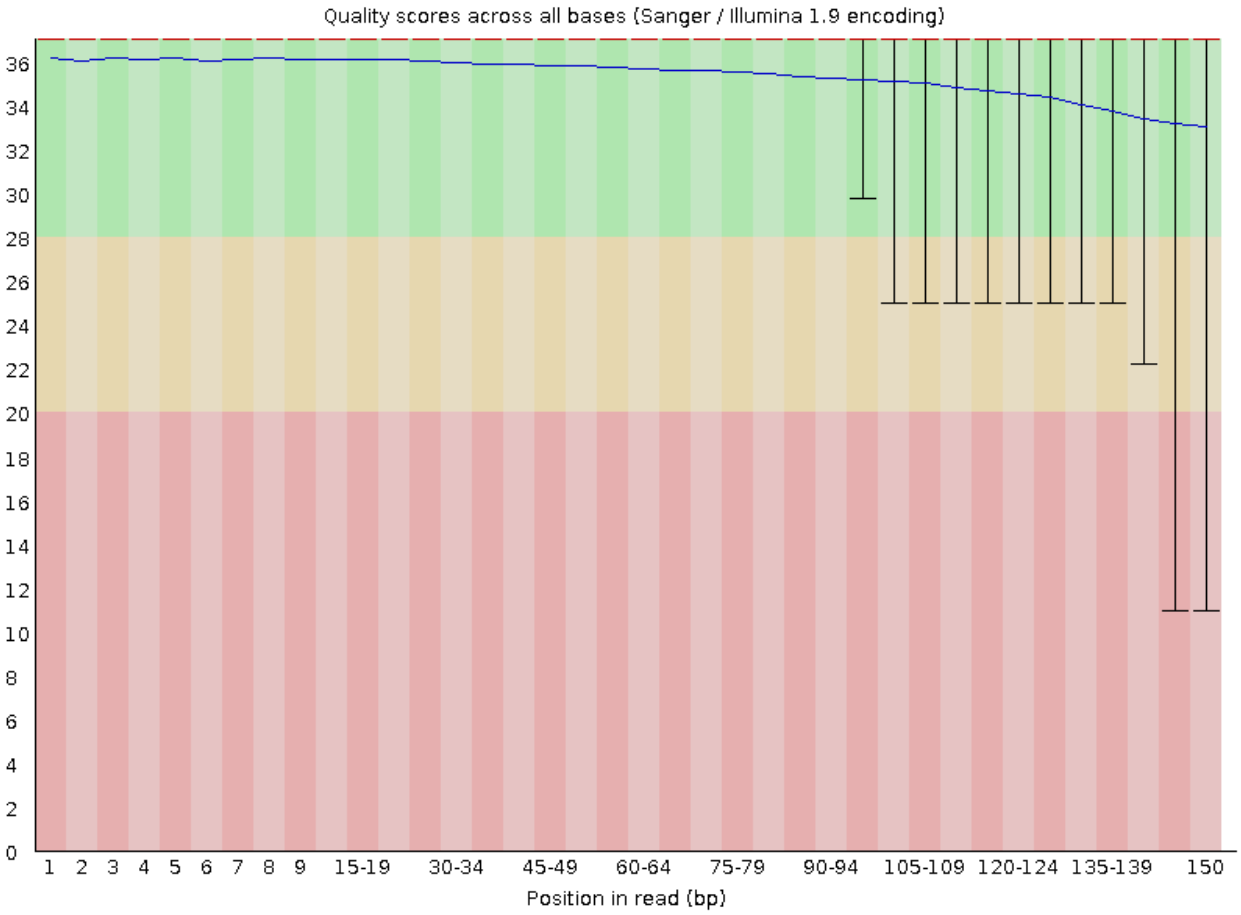


Figure 6: CcoxCrh per base quality R2

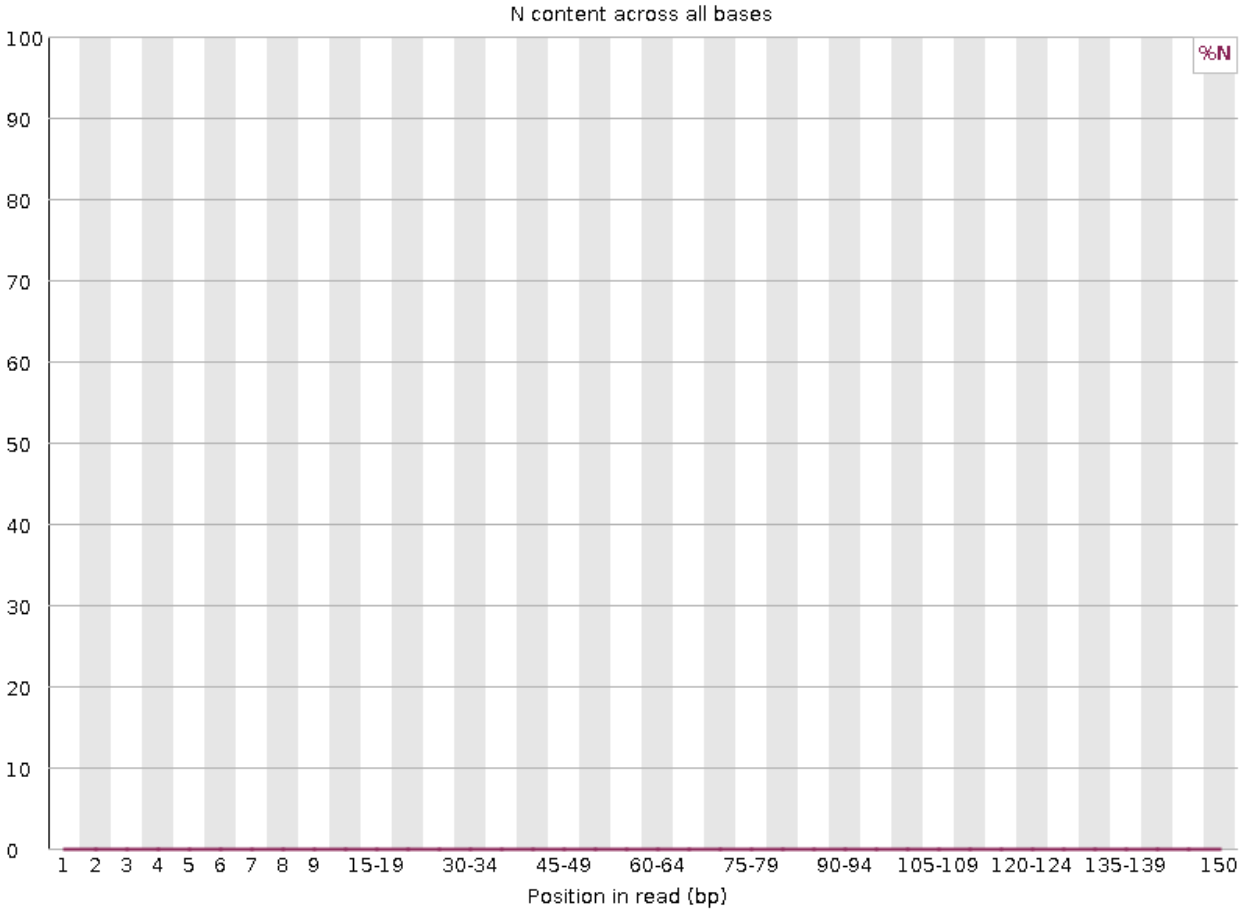


Figure 7: CcoxCrh per base n content R1

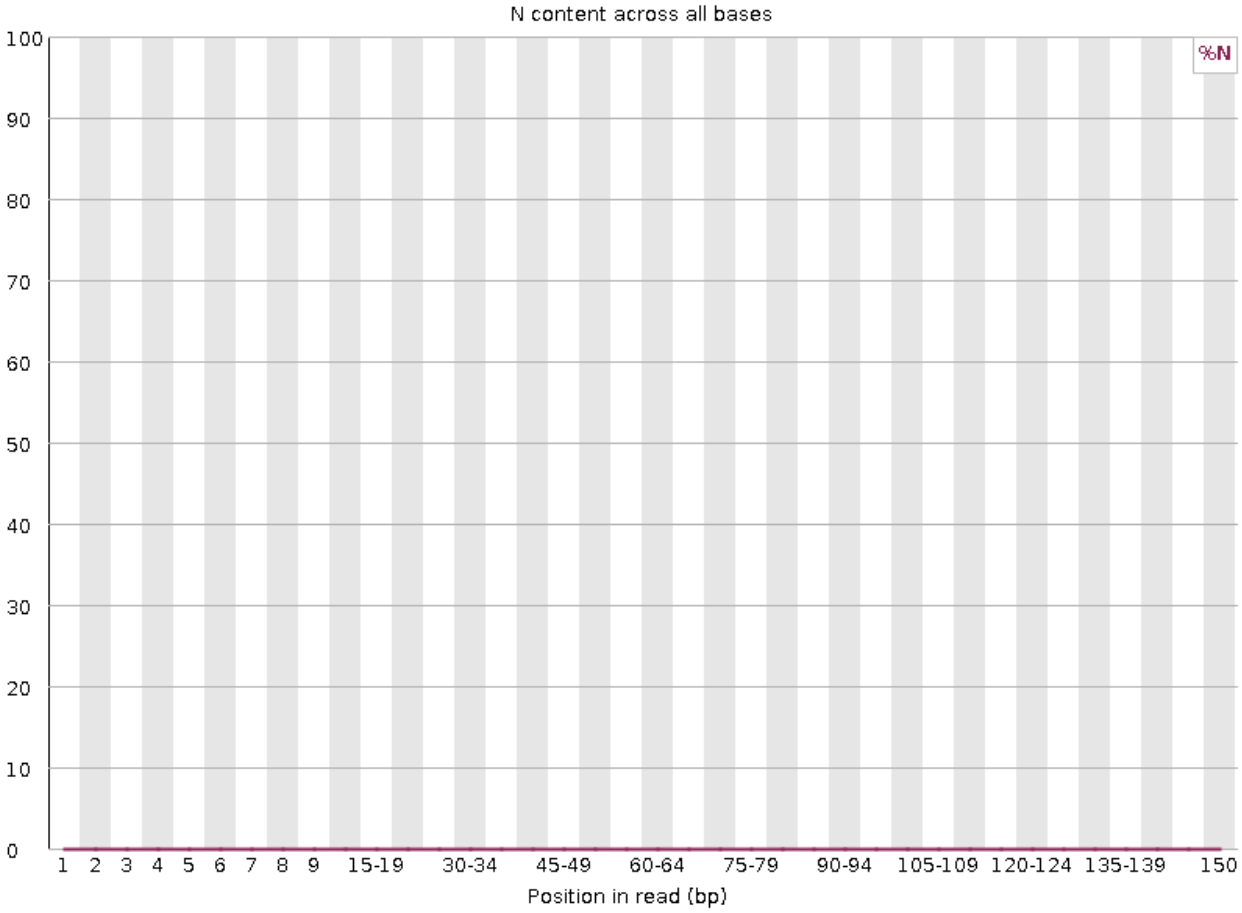


Figure 8: CcoxCrh per base n content R2

Both plots show similar trends with QS distribution being around 36-37 and then a slight decline to 34-35 by the bp 150. My script reflects the results seen in fastqc's plot. There are no differences in the trends only that the fastqc is more informative with its plot and tells you additional information like the box whisker plots for each position as well as the quality of calls (green, orange, or red).

Fig9-12, QS distribution utilizing demux script for Cco and CcoxCrh datasets The average quality score at each base position is taken and then plotted. The middle bp around ~50-90 look darker because these bps share similar heights so when graphed they appear as a block. The trends seen in theses distributions reflect the same in the plots generated by FastQC. Average quality score begins ~36-37 and slowly tapers down to ~34-35 towards the end of the 150bp read lengths.

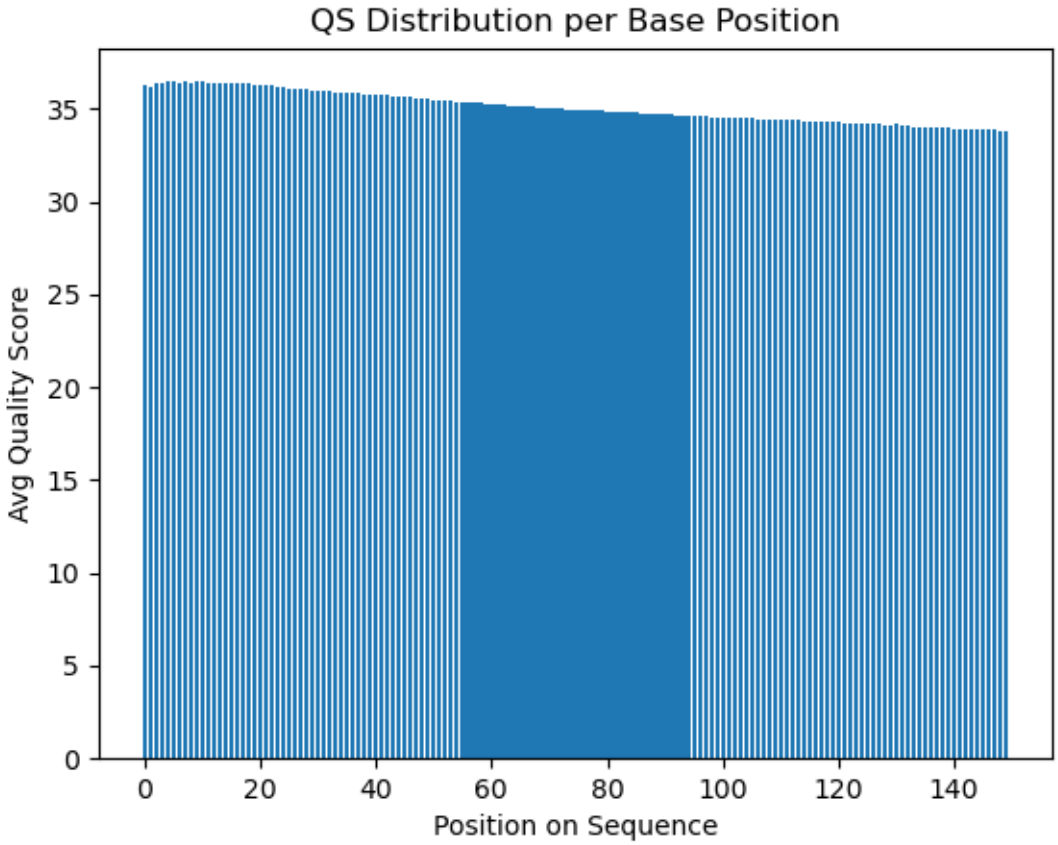


Figure 9: Cco demux script R1

[Include quality score distribution plot, a comparison to FastQC, comments on any differences between your quality score plotting and FastQC]

3. Comment on the overall data quality of your two libraries. Make and justify a recommendation on whether these data are of high enough quality to use for further analysis.

The overall quality of the reads are good and can be used for further downstream analysis. The per base sequence quality passed fastqc's parameters for both R1 and R2 for both SRR files run. The quality scores were consistently high and tapered slowly towards the end of the 150 bps, but this is to be expected with Illumina sequencing, the %N content passes too with ~0% N's being detected. This is consistent with the high qs observed. There are fail flags for things like adapater content, overrepresented sequences, or per

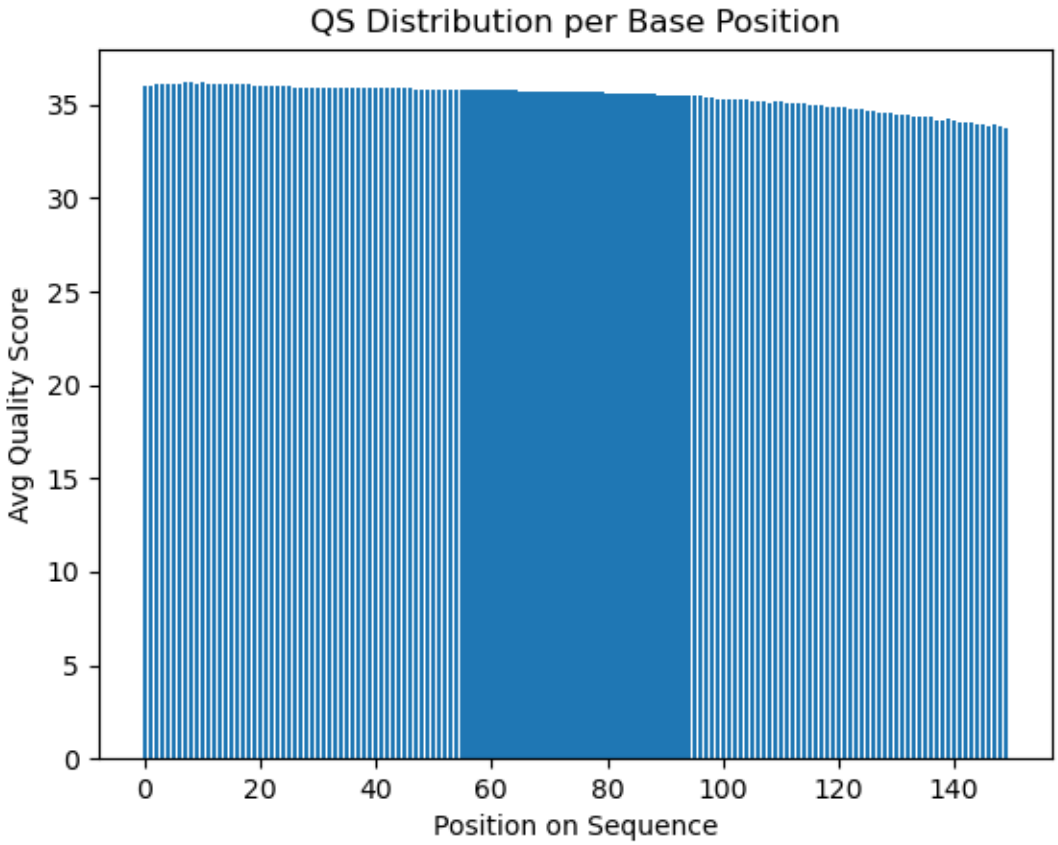


Figure 10: Cco demux script R2

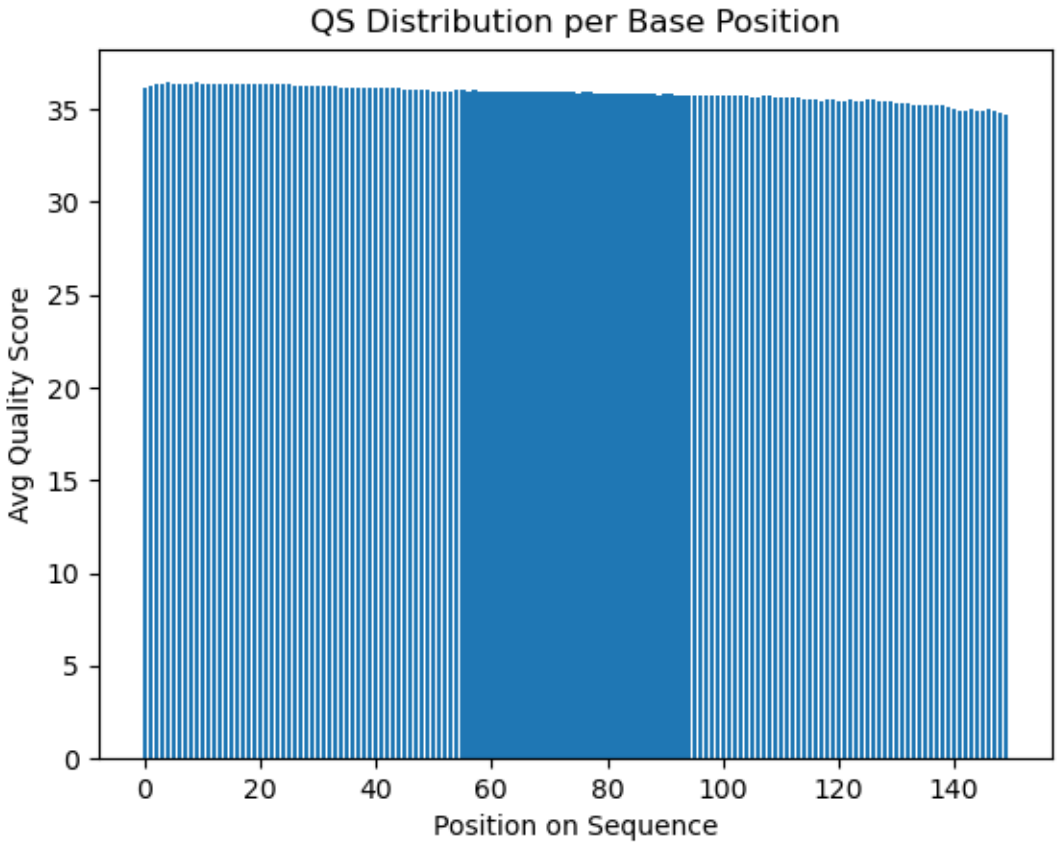


Figure 11: CcoxCrh demux script R1

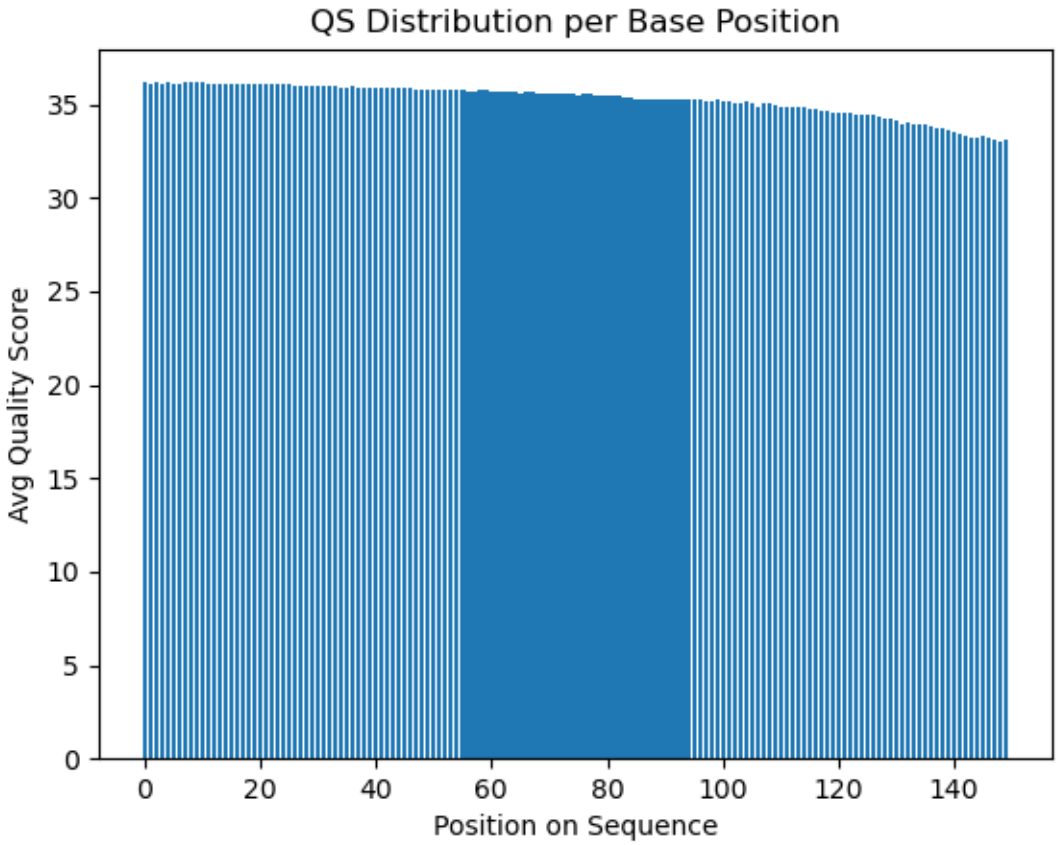


Figure 12: CcoxCrh demux script R2

sequence GC content. We expect there to be reptition because adapters present in the raw reads, high GC content can also be an indication of contamination of a highly over expressed gene.

The pass flags indicate our data is good, and the fail flags are expected especially when considering things like adapters and potential of overexpressed genes. By trimming the adapters out of our data and any low QS tails we can proceed with further analysis.

Part 2 – Adaptor trimming comparison

4. What proportion of reads (both R1 and R2) were trimmed? *Sanity check*: Use your Unix skills to search for the adapter sequences in your datasets and confirm the expected sequence orientations. Report how you confirmed the adapter sequences.

https://support-docs.illumina.com/SHARE/AdapterSequences/Content/SHARE/AdapterSeq/TruSeq/CDIndexes.htm

Illumina adapters:

R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Seed (13-nt) shared by both: AGATCGGAAGAGC

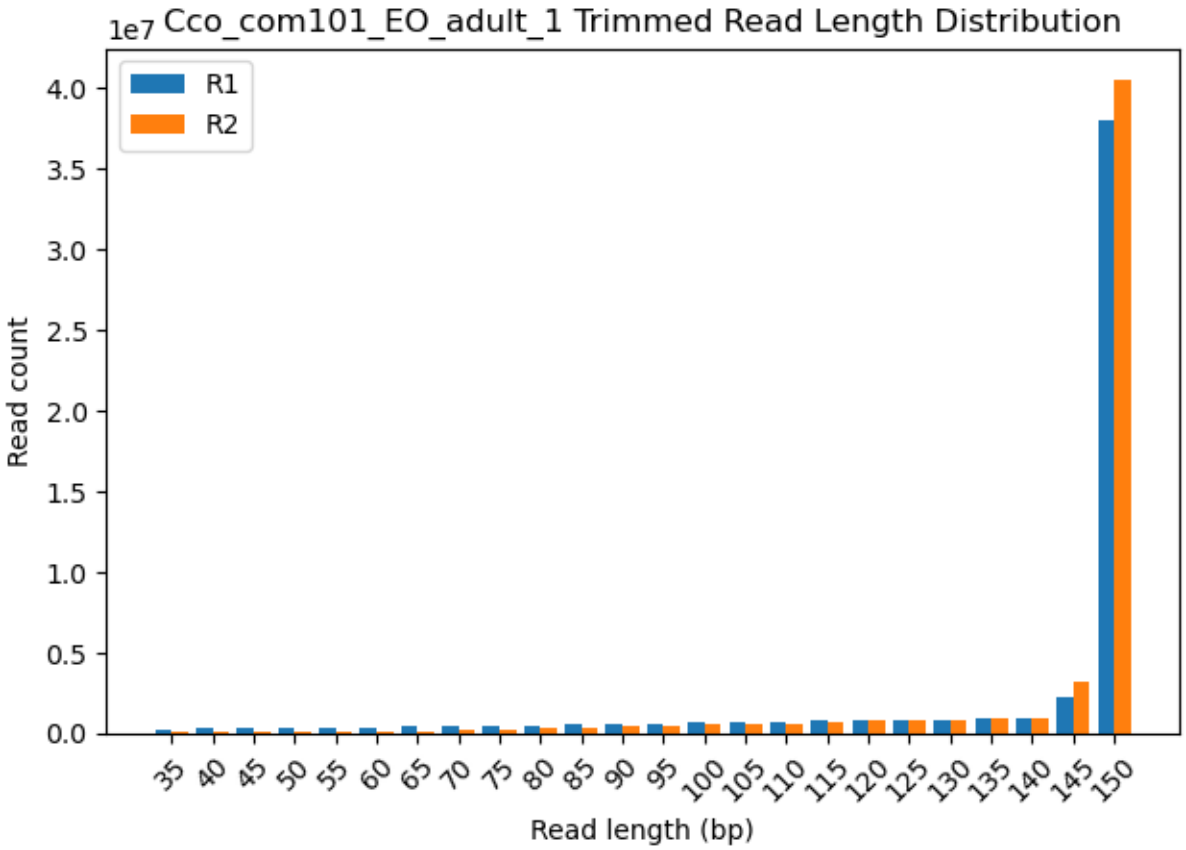
The first thirteen nucleotides of both R1 and R2 adapters are the same. We can use unix commands to search for this seed pattern in our reads towards the 3' end. Checking the last 30 nts shows that the adapter is specifically enriched within this window, filtering out random hits throughout the read. These identified adapters will later be trimmed with cutadapt. Taking into account the fastqc results regarding adapter content, we can confirm that the adapters are in the correct position as well as the expected sequence orientation.

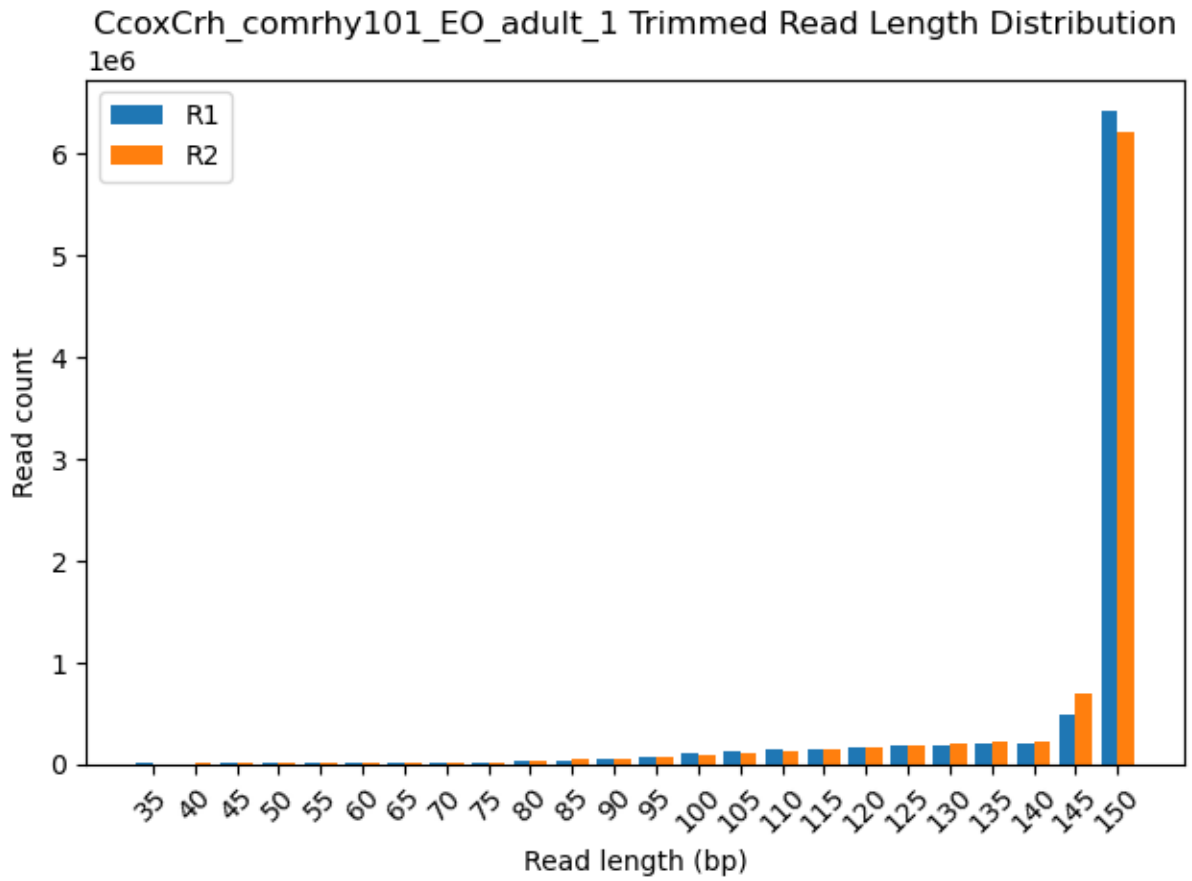
To further confirm this we can look for the adapter in the 5' start and the expected number should be much lower further indicating that the adapters are correctly placed. We can be confident about the sequences and orientation and officially confirm through the use of cutadapt.

5. Plot the trimmed read length distributions for both paired R1 and paired R2 reads. Comment on whether you expect R1s and R2s to be adapter-trimmed at different rates and why.

The plots are left skewed, the majority of the reads peak around 150 bps in both datasets. Adapter trimming is similar between R1 and R2, which we saw in the cudadapt metrics after trimming. I don't expect a big difference in adapter trimming between reads because of the insert size and read lengths, however it is common for R2 to be more trimmed because R2 has lower 3' quality on Illumina instruments so more bases were removed using Trimmomatic's quality filtering. The small differences we see are normal and expected from sequencing quality or detection between the two different reads. If libraries were prepped properly, read lengths, and inserts are around the same size we should have similar adapter trimming in both reads. Cco's R1 was trimmed more than R2, suggesting that there may be adapter accumulation or low quality bases more present in the forward read. Whereas in CcoxCrh, R2 was trimmed more than R1 which is expected.

Fig13-14, Trimmed read length distribution for Cco and CcoxCrh datasets after cutadapt and trimmomatic. Data is skewed heavily to the left indicated by the left trail on the graphs. Majority of the read lengths are distributed at the 150bp mark. In the Cco dataset, R1 appears to have had more extensive adapter and low quality reads trimming, while the opposite is present in the CcoxCrh data.


Cco_com101_EO_adult_1 Trimmed Read Length Distribution
1e7
R1
R2
Read count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
Read length (bp)



Part 3 – Alignment and strand-specificity

6. Report the number of mapped and unmapped reads from each of your 2 SAM files post deduplication with picard.

Sample	Mapped Reads	Unmapped Reads
Cco_com101_EO_adult_1	35,718,855	16,734,296
CcoxCrh_comrhy110_EO_adult_1	9,524,551	977,002

15. Demonstrate convincingly whether or not the data are from "strand-specific" RNA-Seq libraries **and** which `stranded=` parameter should you use for counting your reads for a future differential gene expression analyses. Briefly describe your evidence, using quantitative statements (e.g. "I propose that these data are/are not strand-specific, because X% of the reads are y, as opposed to z."). This kit was used during library preparation. This paper may provide helpful information.

By comparing assigned gene counts between the two different stranded parameters, we can see that for both datasets the reverse condition had far greater number of assigned reads and a lower number of reads that were designated as no feature. This indicates strand-specific reverse protocol, further supported by looking at the percentage of total reads that were assigned to reads. Cco (stranded yes) had only 2.07% reads assigned compared to 38.58% in the reverse stranded. Similarly with the CcoxCrh data, the percentages are 2.64% and 52.24%.

The no feature reads show the opposite pattern, where reverse strandedness has a lower number than the yes strandedness. This means that fewer reads fell into the no feature category, indicating that the reverse setting matches our library. Additionally, if we look at the ambiguous values the amibugous values are greater in the reverse conditions than in the yes strandedness, this simply implies that there are more reads overlapping with real features so this pattern is expected. Another thing to consider is that the revvity kit used to generate libraries preserves strand orientation with dUTP second strand synthesis, which means R1 maps to the antisense strand which is what we mark as reverse stranded in htseq-count. Our results confirm this since we had a far greater number of reads with reverse strandedness and a reduced number of reads that were no feature.

The libraries are strand specific and we should utilize reverse strandedness for downstream analysis like DGE.

HTSeq-count results: stranded=yes vs stranded=reverse

Sample	Strandedness	Assigned Reads	___no_feature	___ambiguous
Cco	yes	555,332	16,650,247	2,263
	reverse	10,328,955	6,747,807	131,080
CcoxCrh	yes	138,529	4,448,568	427
	reverse	2,745,246	1,808,597	33,681

Table 1, Summarized results from htseq-count for the Cco and CcoxCrh datasets. Strandedness is indicated as well as the assigned reads, amount of no feature, or ambiguous reads. These were the three criteria looked at to determine whether libraries were strand specific and which stranded parameter should be used for downstream analyses.

HTSeq-count results: stranded=yes vs stranded=reverse. % Assigned Reads

Sample	Strandedness	Assigned Reads	Total Reads	% Assigned
Cco	yes	555,332	26,773,576	2.07%
	reverse	10,328,955	26,773,576	38.58%
CcoxCrh	yes	138,529	5,255,293	2.64%
	reverse	2,745,246	5,255,293	52.24%

Table 2, Results summarizing the total reads and % assigned reads for both Cco and CcoxCrh data post htseq-count analysis. In both datasets the reverse parameter had a greater percentage of reads, 38.58 and 52.24, respectively.