PS2_template

2025-04-25

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

Overall assignment:

In this assignment, you will process electric organ and/or skeletal muscle RNA-seq reads for a future differential gene expression analysis. We will be completing the differential gene expression analysis in our last bioinformatics assignment of this class. You will learn how to use existing tools for quality assessment and read trimming, compare quality assessments to those created by your own software, and how to align and count reads. Additionally, you will learn how to summarize important information in a high-level report. You should create a cohesive, well written report for your "PI" about what you've learned about/from your data.

This template is provided as reference for instructions. Files with specific naming conventions are requested to be turned in at the end of this problem set. You can use this template to gather notes while completing this assignment. Be sure to upload all relevant materials by the deadline and **double check** to be sure that your offline repository is up-to-date with your online repository. Answers to questions should be included in your final, high-level, report as a `pdf`. This pdf should be generated using Rmarkdown and submitted to Canvas as well as GitHub. Be sure to keep a well-organized, detailed lab notebook!

Dataset:

Each of you will be working with 2 RNA-seq files from two different electric fish studies (PRJNA1005245 and PRJNA1005244). The methods for the PRJNA1005244 dataset are published and the methods for the PRJNA1005245 dataset are written in the third chapter of a thesis. For all steps below, process the two libraries separately. SRR assignments are here: `/projects/bgmp/shared/Bi623/PS2/QAA_data_Assignments.txt`. If you have time, consider claiming and processing additional RNA-seq raw sequencing files via this google doc. Although this is not extra credit, it will make our downstream RNA-seq analysis more interesting and your classmates will appreciate your efforts.

You are responsible for downloading this data from NCBI SRA, dumping into FASTQ files, and zipping those files (check ICA1 for a refresher). We are processing this data for use in a future assignment, so please keep your files well organized. Finally, rename the files to the convention Species_sample_tissue_age/size_sample#_readnumber.fastq.gz.

Reminder: This template file IS not your final product; however, it gives you a space to record all of the necessary information for your final report.

```
## Download your data

#commands used to create sra conda environment in ICA1
srun -A bgmp -p bgmp --time=1:00:00 --pty bash
```

```
conda create -n sra
conda activate sra
conda install bioconda::sra-tools

#download data
prefetch SRR25630408
prefetch SRR25630384
```

Part 1 – Read quality score distributions

1. Create a new conda environment called `QAA` and install `FastQC`, `cutadapt`, and `Trimmomatic`. Google around if you need a refresher on how to create conda environments. Recommend doing this in an interactive session, not the login node! Record details of how you created this environment in your lab notebook! Make sure you check your installation with:

   - `fastqc --version` (should be 0.12.1)

[Record details on how you made the conda environment]

```
srun -A bgmp -p bgmp --time=1:00:00 --pty bash
conda create -n QAA -c conda-forge -c bioconda python=3.10 fastqc=0.12.1 trimmomatic=0.39 cutadapt=5.0

fastqc --version
trimmomatic -version
cutadapt --version
```

FastQC: 0.12.1 Trimmomatic: 0.40 Cutadapt: 5.0

2. Using `FastQC` via the command line on Talapas, produce plots of the per-base quality score distributions for R1 and R2 reads. Also, produce plots of the per-base N content, and comment on whether or not they are consistent with the quality score plots.

```
conda activate QAA
fastqc SRR25630384_1.fastq.gz SRR25630384_2.fastq.gz
fastqc SRR25630408_1.fastq.gz SRR25630408_2.fastq.gz

#pull out images for per-base qs and per base N content
unzip -p SRR25630384_1_fastqc.zip "*/Images/per_base_quality.png" > CcoxCrh_fastqc_per_base_quality_1.p
unzip -p SRR25630384_1_fastqc.zip "*/Images/per_base_n_content.png" > CcoxCrh_fastqc_per_base_n_content
```

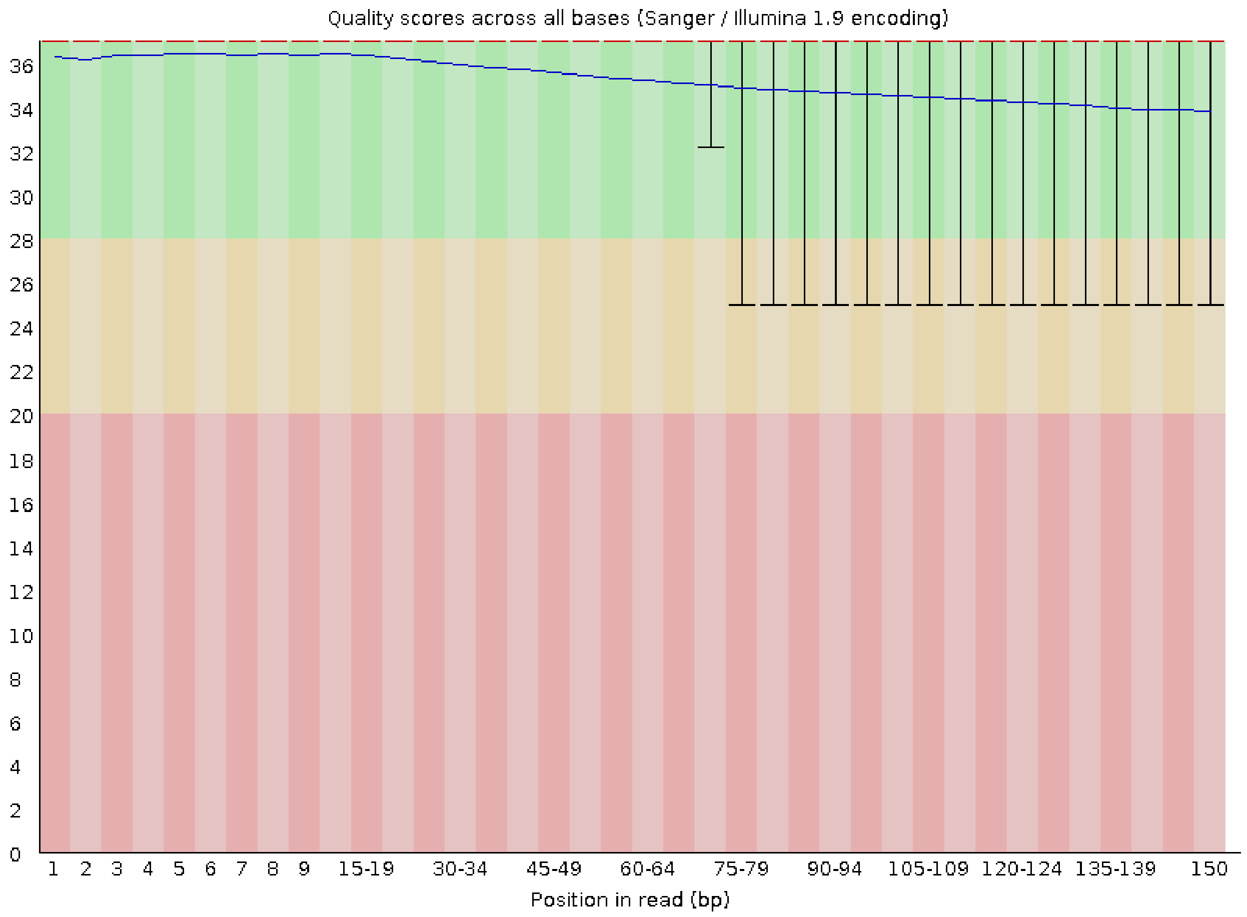


Figure 1: Cco per base quality R1

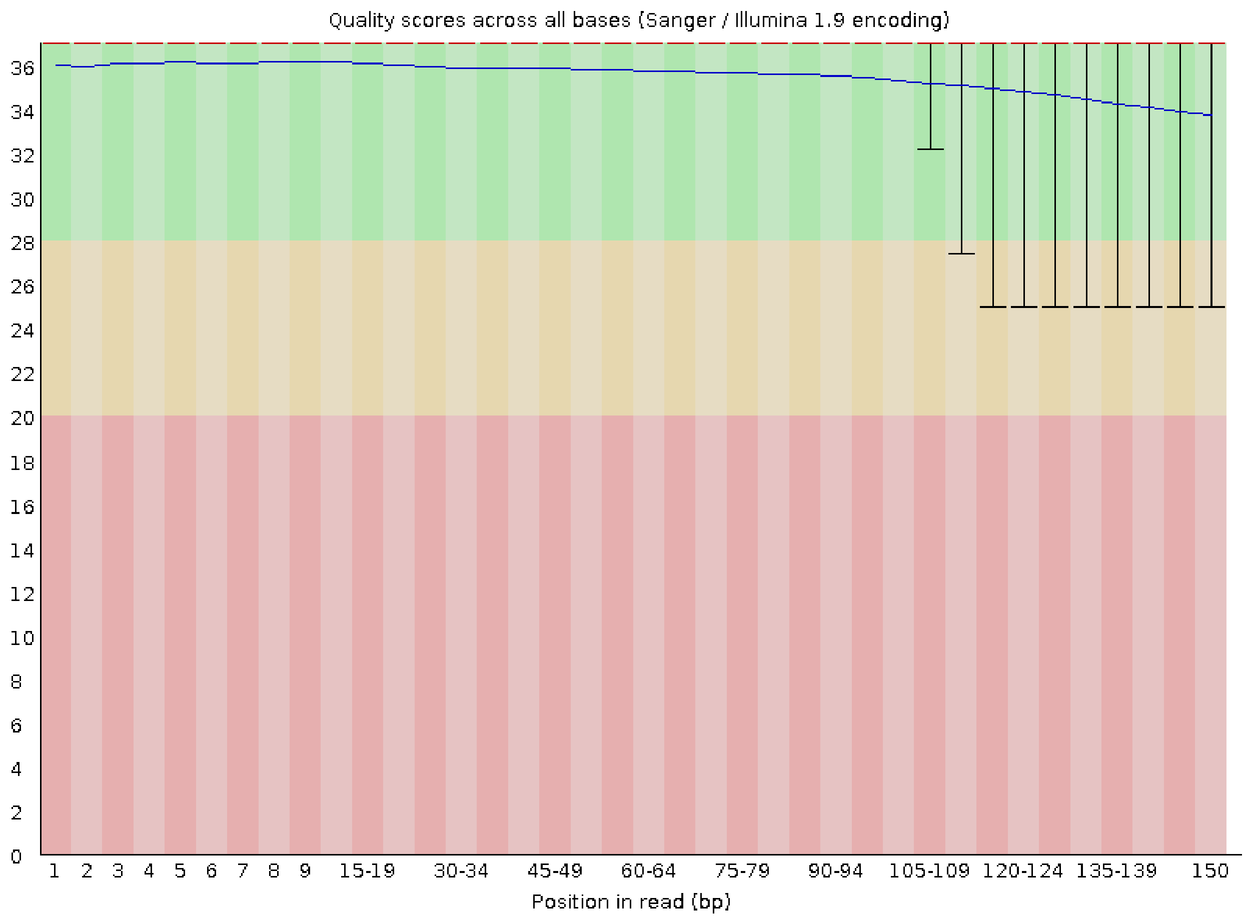


Figure 2: Cco per base quality R2

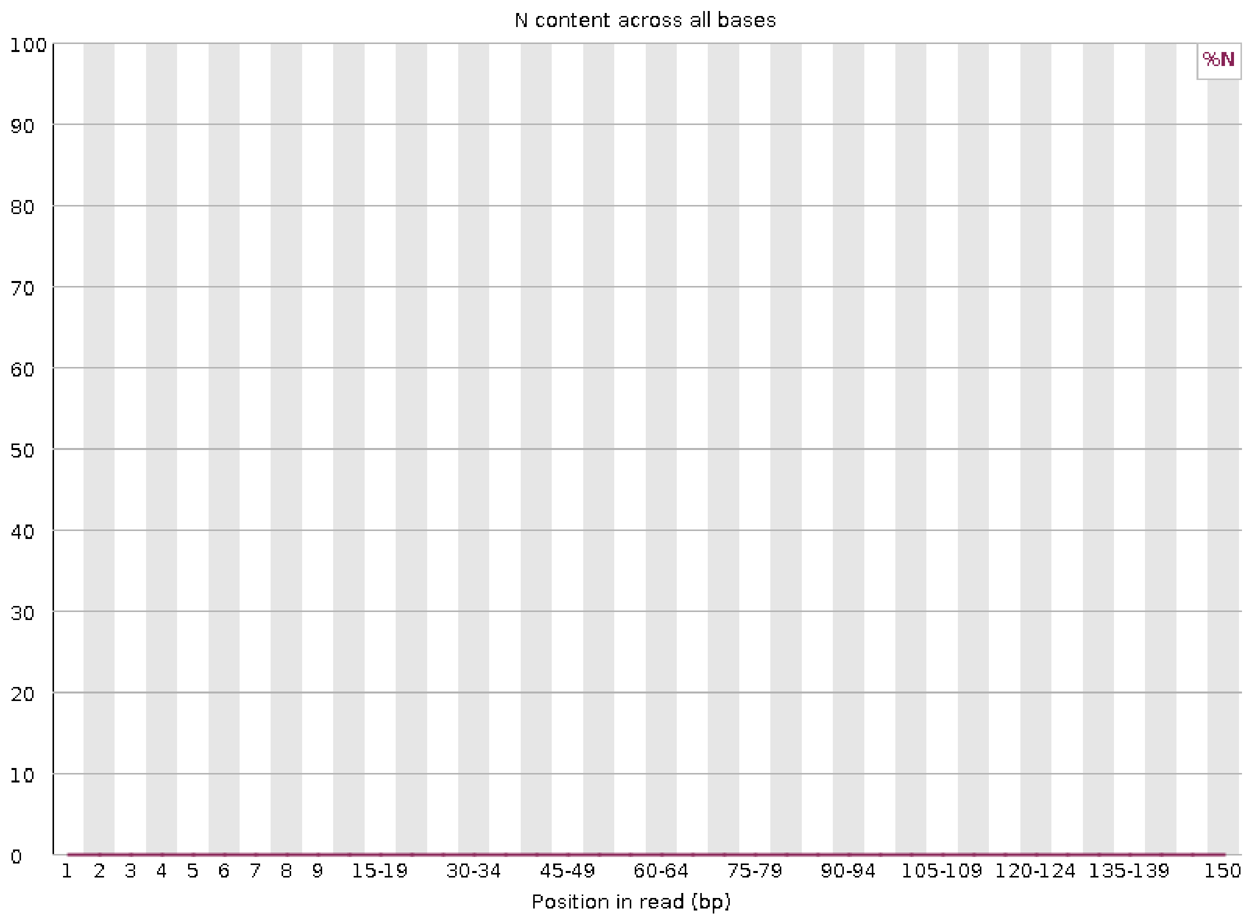


Figure 3: Cco per base n content R1

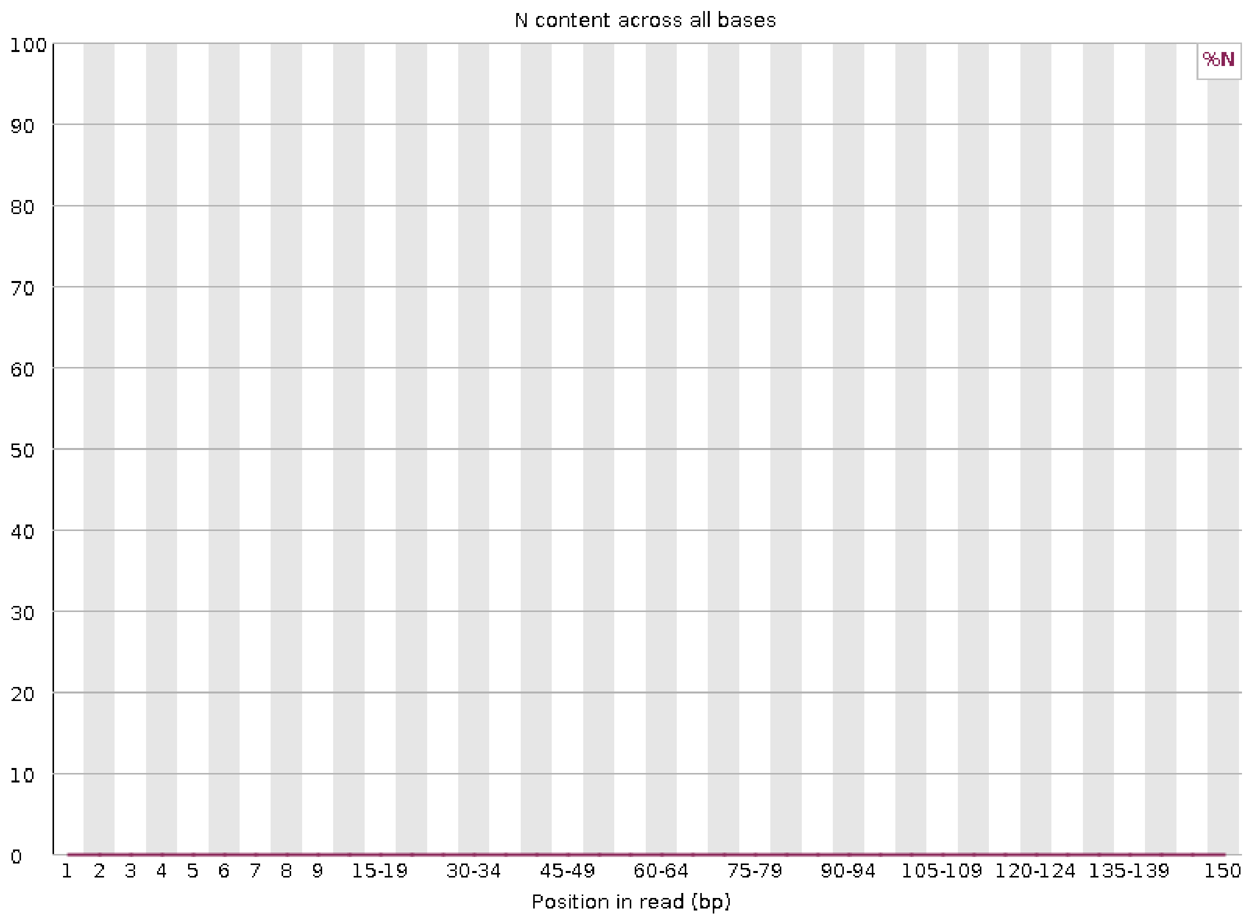


Figure 4: Cco per base n content R2

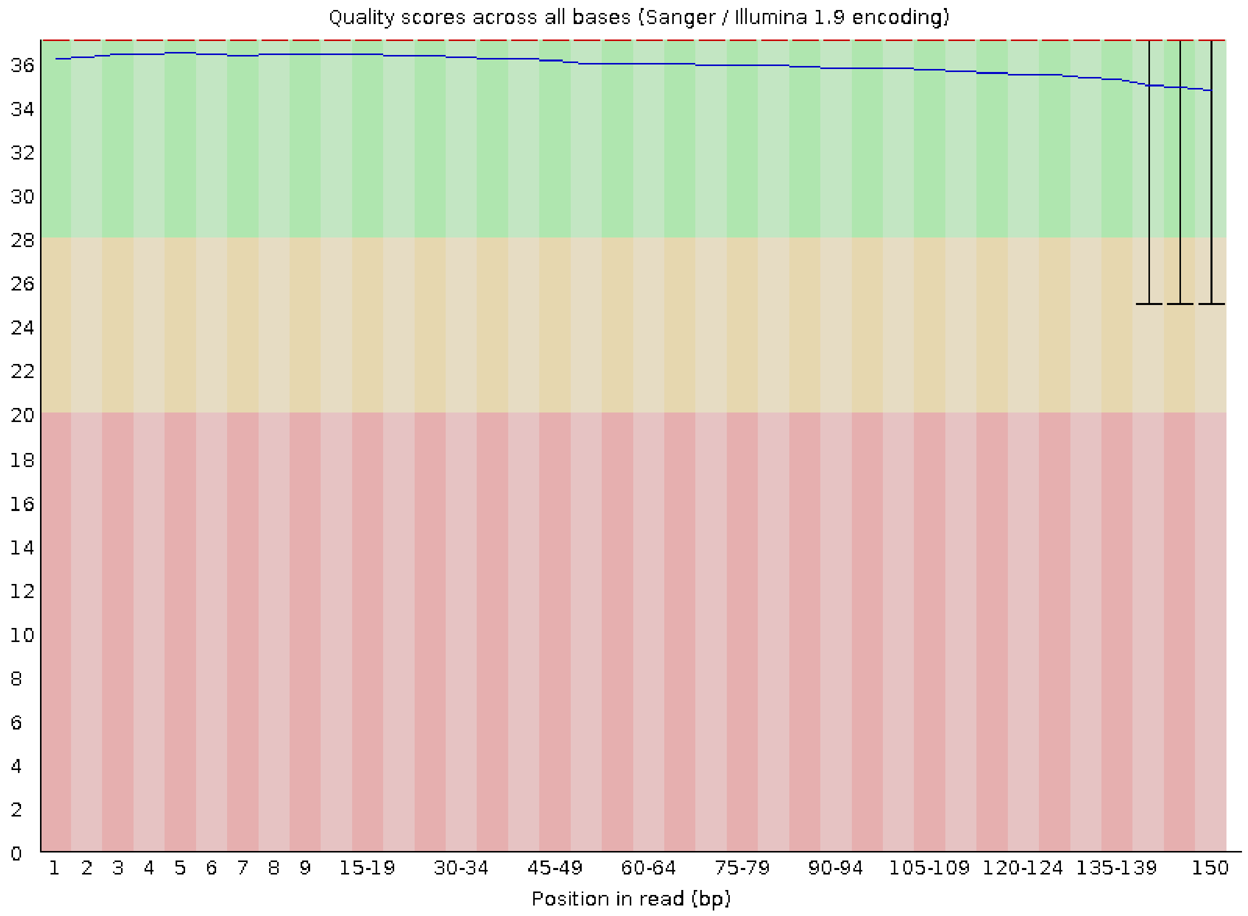


Figure 5: CcoxCrh per base quality R1

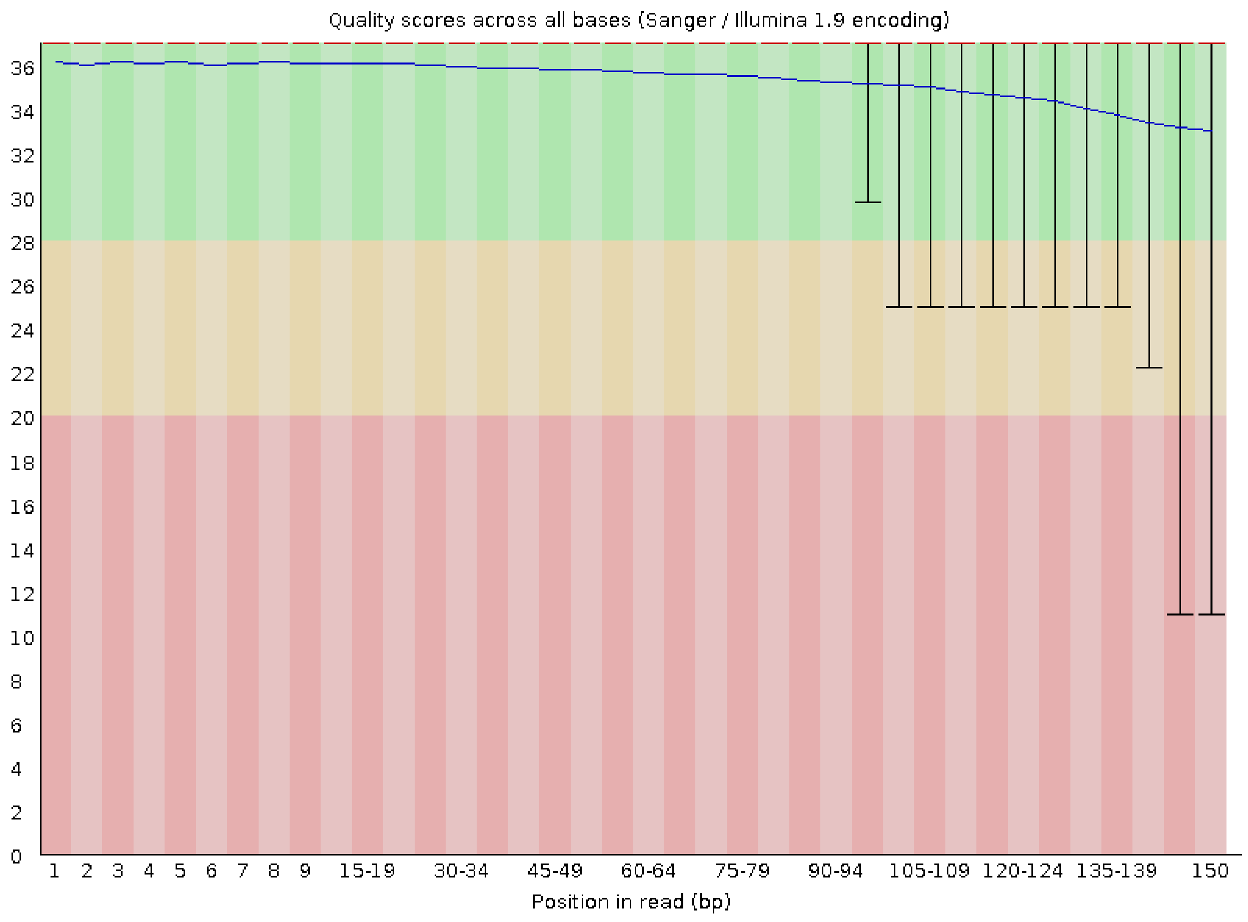


Figure 6: CcoxCrh per base quality R2

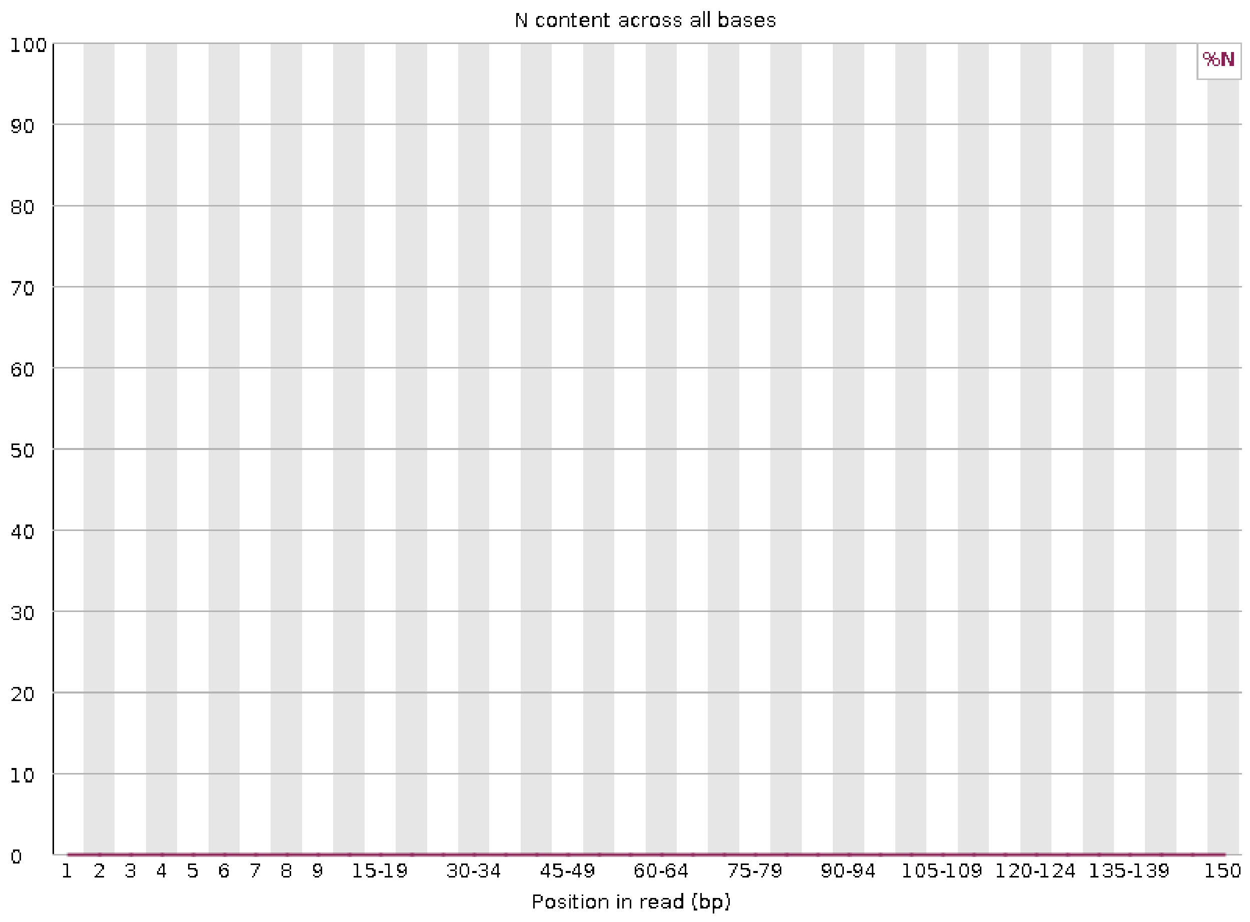
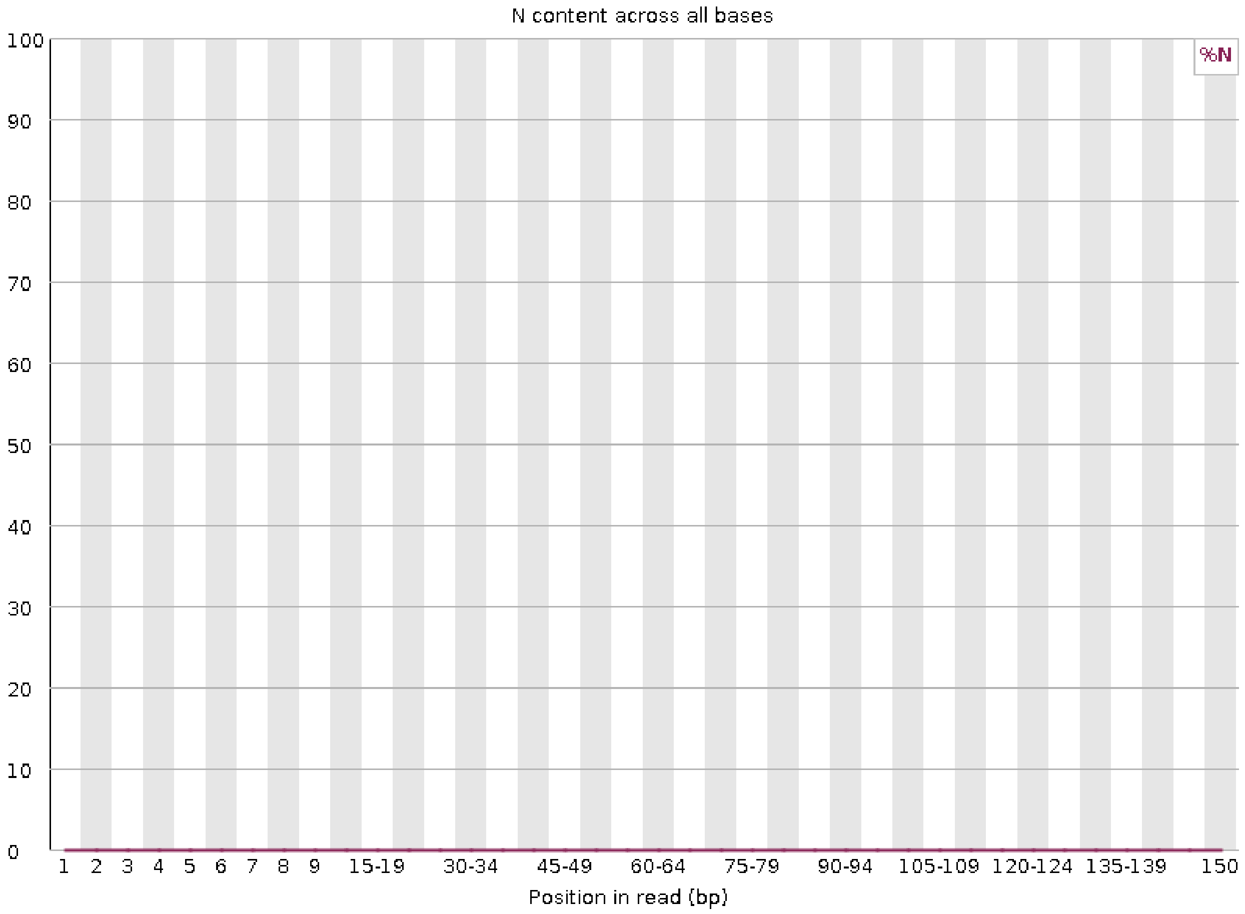


Figure 7: CcoxCrh per base n content R1



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.

[Include FastQC commands, plots of per-base N content, comments on consistency with quality score plots]

3. Run your quality score plotting script from your Demultiplexing assignment in Bi622. (Make sure you're using the "running sum" strategy!!) 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?

I was unable to run an sbatch using fastqc, because of maintenance on Talapas. It took <10 mins for each readpair (so ~20 mins total). I tested and confirmed by rerunning an sbatch on one readpair. 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 %.

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).

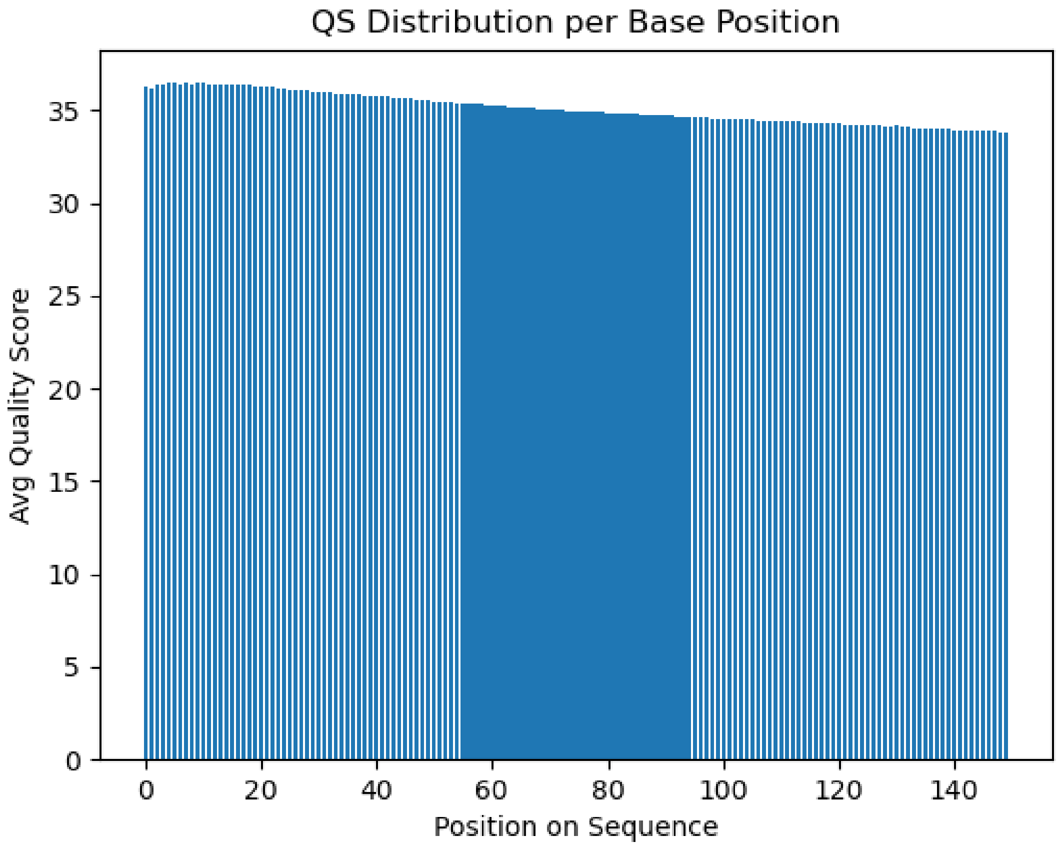


Figure 8: Cco demux script R1

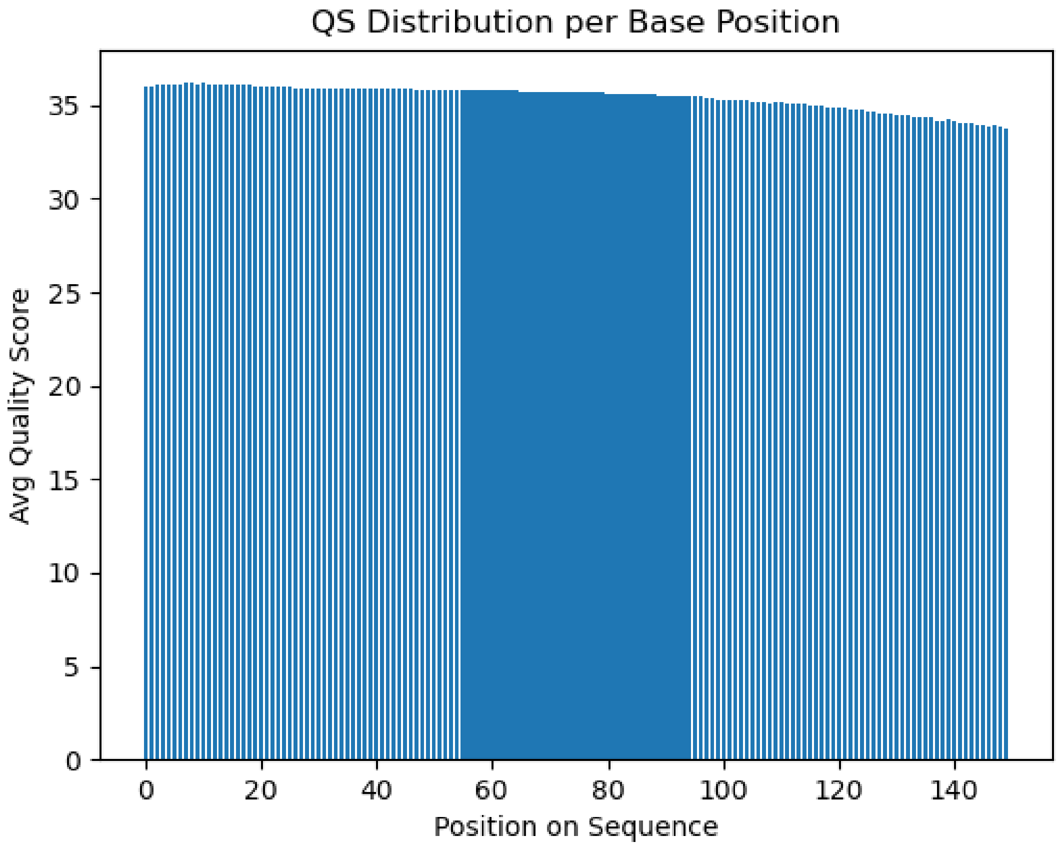


Figure 9: Cco demux script R2

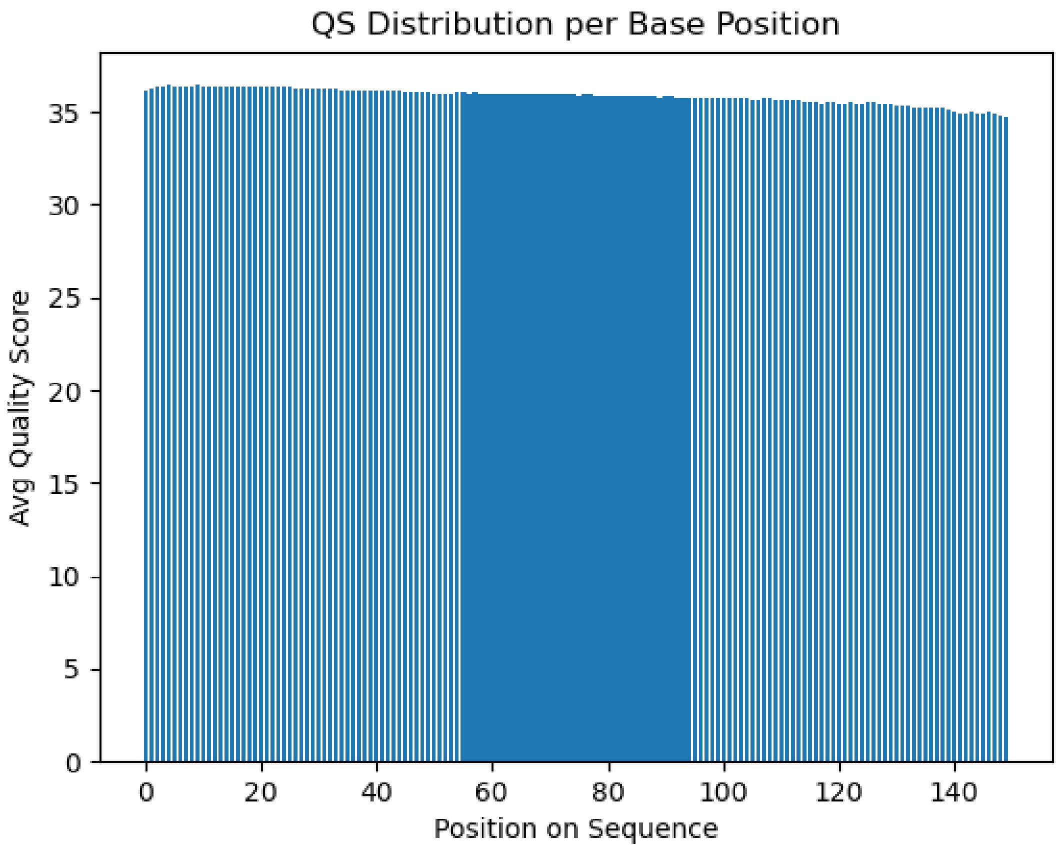


Figure 10: CcoxCrh demux script R1

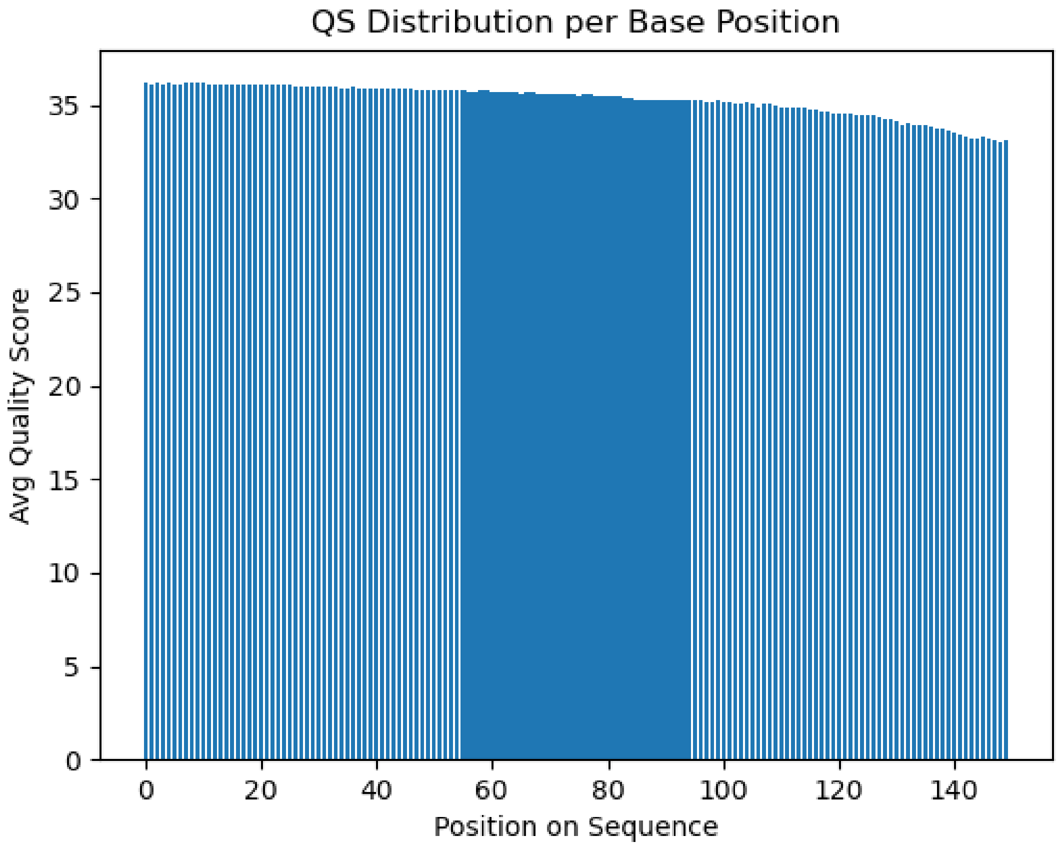


Figure 11: CcoxCrh demux script R2

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

4. Comment on the overall data quality of your two libraries. Go beyond per-base qscore distributions. Examine the `FastQC` documentation for guidance on interpreting results and planning next steps. Make and justify a recommendation on whether these data are of high enough quality to use for further analysis.

[Include comments on data quality and recommendation on whether this can be used 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 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

5. If you haven't already in your QAA environment, install `Cutadapt` and `Trimmomatic`. Check your installations with:
   - `cutadapt --version` (should be 5.0)
   - `trimmomatic -version` (should be 0.39)

```
conda activate
fastqc --version
trimmomatic -version
cutadapt --version
```

FastQC: 0.12.1 Trimmomatic: 0.40 Cutadapt: 5.0

[Record details on install (if happened here) and/or version checking]

6. Using `Cutadapt`, properly trim adapter sequences from your assigned files. Be sure to read how to use `Cutadapt`. Use default settings. What proportion of reads (both R1 and R2) were trimmed?

   Try to determine what the adapters are on your own. If you cannot (or if you do, and want to confirm), click here to see the actual adapter sequences used.

   R1: `AGATCGGAAGAGCACACGTCTGAACTCCAGTCA`

   R2: `AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT`

   - *Sanity check*: Use your Unix skills to search for the adapter sequences in your datasets and confirm the expected sequence orientations. Report the commands you used, the reasoning behind them, and how you confirmed the adapter sequences.

[Include commands and report out the proportion of reads trimmed]

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: AGATCGGAA-GAGC 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.

```
#check 3' end for adapter
zcat CcoxCrh_comrhy110_EO_adult_1_1.fastq.gz | sed -n '2~4p' | awk '{print substr($0,length($0)-29)}' |

#check 5' end for adapter, should yield zero
zcat Cco_com101_EO_adult_1_1.fastq.gz | sed -n '2~4p' | awk '{print substr($0,1,30)}' | grep -Ec '^AGAT
```

Proportion of reads trimmed

```
Cco_com101_EO_adult_1
  R1: 17.4% reads trimmed
  R2: 17.0% reads trimmed

CcoxCrh_comrhy110_EO_adult_1
  R1: 21.8% reads trimmed
  R2: 21.6% reads trimmed
```

7. Use `Trimmomatic` to quality trim your reads. Specify the following, **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

   Be sure to output compressed files and clear out all intermediate files.

Ran command in slurm script

```
#command in slurm
/usr/bin/time -v trimmomatic PE -threads 8 -phred33 $Cco_1_trimmed $Cco_2_trimmed \
    $Cco_1_qualtrim_paired $Cco_1_qualtrim_UNpaired \
    $Cco_2_qualtrim_paired $Cco_2_qualtrim_UNpaired \
    LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35

#command in terminal
rm -f *_UNpaired.fastq.gz
```

8. Plot the trimmed read length distributions for both paired R1 and paired R2 reads (on the same plot - yes, you will have to use Python or R to plot this. See ICA4 from Bi621). You can produce 2 different plots for your 2 different RNA-seq samples. There are a number of ways you could possibly do this. One useful thing your plot should show, for example, is whether R1s are trimmed more extensively than R2s, or vice versa. 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.

```
![Cco read len distr](/projects/bgmp/tnair/bioinfo/Bi623/PS2/QAA/plots/Cco_com101_EO_adult_1_readdistr_
![CcoxCrh read len distr](/projects/bgmp/tnair/bioinfo/Bi623/PS2/QAA/plots/CcoxCrh_comrhy110_EO_adult_1_
```

[Include your plot and comment on R1/R2 adapter trimming]

```
#ran python script, not R
```

9. Bonus - Run `FastQC` on your trimmed data. Comment on differences you observe between the trimmed and untrimmed data. Include any figures needed to support your conclusions.

[Include command, comments on differences, and plot/s]

```
```

Part 3 – Alignment and strand-specificity

10. Install additional software for alignment and counting of RNA-seq reads. In your QAA environment, use conda to install:
    - Star
    - Picard
    - Samtools
    - NumPy
    - Matplotlib
    - HTSeq

[Record details on how you installed these packages]

```
conda activate QAA

conda install bioconda::star
conda install bioconda::picard
conda install bioconda::samtools
conda install bioconda::htseq

conda install -c conda-forge numpy matplotlib
```

11. Download the publicly available *Campylomormyrus compressirostris* genome fasta and gff file from Dryad and generate an alignment database from it. If the download fails, the files are available `/projects/bgmp/shared/Bi623/PS2/campylomormyrus.fasta`, `/projects/bgmp/shared/Bi623/PS2/campylomormyr` Align the reads to your *C. compressirostris* database using a splice-aware aligner. Use the settings specified in PS8 from Bi621.

> [!IMPORTANT] You will need to use gene models to perform splice-aware alignment, see PS8 from Bi621. You may need to convert the gff file into a gtf file for this to work successfully.

[Record details on how you downloaded the genome, prepared the dataset for alignment, and commands for generating the alignment database and aligning reads]

```
#unable to download fr Dryad, cp from Talapas location

cp /projects/bgmp/shared/Bi623/PS2/campylomormyrus.fasta .
cp /projects/bgmp/shared/Bi623/PS2/campylomormyrus.gff .

#need gffread to convert gff to gtf
conda install bioconda::gffread
gffread campylomormyrus.gff -T -o campylomormyrus.gtf

#slurm script to generate database
/usr/bin/time -v STAR --runThreadN 8 \
--runMode genomeGenerate \
--genomeDir /projects/bgmp/tnair/bioinfo/Bi623/PS2/part3_alignment/campylomormyrus.STAR_2.7.11b \
--genomeFastaFiles /projects/bgmp/tnair/bioinfo/Bi623/PS2/part3_alignment/campylomormyrus.fasta \
--sjdbGTFfile /projects/bgmp/tnair/bioinfo/Bi623/PS2/part3_alignment/campylomormyrus.gtf \
--genomeSAindexNbases 13

#slurm script to align reads
/usr/bin/time -v STAR --runThreadN 8 --runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 --alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn  $Cco_R1 $Cco_R2 \
--genomeDir $genome_dir \
--outFileNamePrefix $Cco_alignment
```

12. Remove PCR duplicates using Picard MarkDuplicates. You may need to sort your reads with `samtools` before running Picard.

- Use the following for running picard: picard MarkDuplicates INPUT=[FILE] OUTPUT=[FILE] METRICS_FILE=[FILENAME].metrics REMOVE_DUPLICATES=TRUE VALIDATION_STRINGENCY=LENIENT

```
#convert to BAM and sort
/usr/bin/time -v samtools view -b $CcoxCrh_sam | samtools sort -o $CcoxCrh_sorted_bam

# Add readgroups
/usr/bin/time -v picard AddOrReplaceReadGroups \
    I=$Cco_sorted_bam \
    O=$Cco_sorted_RG \
    RGID=1 \
    RGLB=lib1 \
    RGPL=illumina \
    RGPU=unit1 \
    RGSM=Cco \
    VALIDATION_STRINGENCY=LENIENT
```

```
#deduplication with picard
/usr/bin/time -v picard MarkDuplicates INPUT=$CcoxCrh_sorted_bam OUTPUT=$CcoxCrh_dedup_bam METRICS_FILE=
REMOVE_DUPLICATES=TRUE VALIDATION_STRINGENCY=LENIENT
```

13. Using your script from PS8 in Bi621, report the number of mapped and unmapped reads from each of your 2 SAM files post deduplication with picard. Make sure that your script is looking at the bitwise flag to determine if reads are primary or secondary mapping (update/fix your script if necessary).

[Include the number of mapped and unmapped reads from both same files]

```
#convert to SAM
samtools view -h -o Cco_dedup.sam CcoxCrh_dedup.bam

#Count reads
./is_mapped.py -f /projects/bgmp/tnair/bioinfo/Bi623/PS2/part3_alignment/Cco_dedup.sam
```

Cco Mapped reads: 35718855 Unmapped reads: 16734296

CcoxCrh Mapped reads: 9524551 Unmapped reads: 977002

14. Count deduplicated reads that map to features using `htseq-count`. You should run htseq-count twice: once with `--stranded=yes` and again with `--stranded=reverse`. Use default parameters otherwise. You may need to use the `-i` parameter for this run.

```
#check gene id col for -idattr (-i) col
head campylomormyrus.gtf

#check what the gene id col looks like for --idattr (-i) parameter
head campylomormyrus.gtf

#Cco
htseq-count --format=bam --order=pos --stranded=yes --type=exon --idattr=gene_id $Cco_bam $campie_gtf >
htseq-count --format=bam --order=pos --stranded=reverse --type=exon --idattr=gene_id $Cco_bam $campie_g
```

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. Include any commands/scripts used. 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.

    [!TIP] Recall ICA4 from Bi621.

[Describe whether your reads are "string-specific", why you think they are, any evidence, and which stranded parameter is appropriate and why]

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.

```
#determine no of assigned reads
awk '$1 !~ /^__/ {s+=$2} END{print s+0}' Cco_counts_stranded_yes.txt

#determine no feature, ambiguous reads
tail Cco_counts_stranded_yes.txt
```

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

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%

16. BONUS - Turn your commands from part 1 and 2 into a script with a loop going through your two SRA files

Bonus (optional!)

Review the publication from PRJNA1005244 or the third chapter of the thesis for the PRJNA1005245 dataset. See if this information leads to any additional insight of your analysis.

[Add insights to the dataset]

Upload your:

- ☒ lab notebook

- ☒ Talapas batch script/code
- ☒ FastQC plots
- ☒ counts files generated from htseq-count (in a folder would be nice; **only include the counts files that would be used in a future differential RNA-seq analysis: use the format Species__sample__tissue__age/size__sample#*readnumber__htseqcounts*[revORyes]stranded.txt**)
- ☒ pdf report (see below; turn into both Github AND Canvas)
- ☒ and any additional plots, code, or code output

to GitHub.

Pdf report details

You should create a pdf file (using Rmarkdown) with a high-level report including:

- ☒ all requested plots
- ☒ answers to questions
- ☒ mapped/unmapped read counts from PS8 script (in a nicely formatted table)
- ☒ It should be named `QAA_report.pdf`
- ☒ Include at the top level of your repo
- ☒ ALSO, submit it to Canvas.

> [!TIP] You may need to install LaTeX to knit your rmarkdown into a pdf file. Run `tinytex::install_tinytex()` to install it on R.

The three parts of the assignment should be clearly labeled. Be sure to title and write a descriptive figure caption for each image/graph/table you present.

> [!TIP] Think about figure captions you've read and discussed in Journal Club. Find some good examples to model your captions on.