QAA Report

Kaetlyn Gibson

2022-09-06

Introduction

Objectives

- Compare own software with existing tools for:
  - Quality assessment
  - Adapter trimming
- Summarize important information about the given RNA-Seq data set.

Data Set(s):

Assigned 2 demultiplexed file pairs:

- 34_4H_both_S24_L008
- 21_3G_both_S15_L008

These two libraries will be processed separately for all parts below.

Part 1 – Read quality score distributions

FastQC Quality Score Distribution & Per-Base N Content Plots

The data for the Per-Base N Content (PBNC) plots appear to be consistent with the quality score plots. On the far left of the PBNC plots, the values appear slightly above zero, which matches up with the far left of the quality score plots where the quality scores are the lowest in the read. This is consistent for both reads in both libraries.

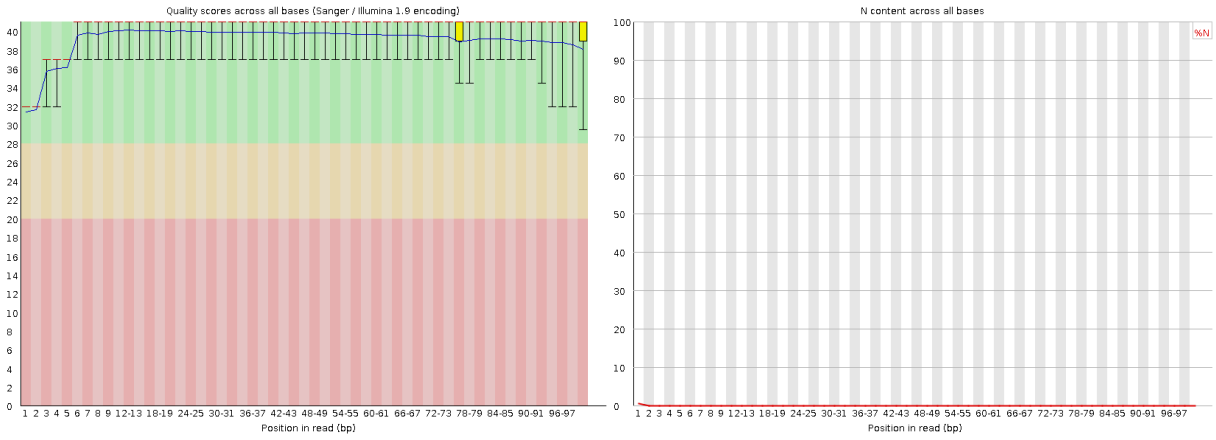


FIGURE: 21 R1 FastQC Quality Score Distribution (L) and Per-Base N Content (R) Plots

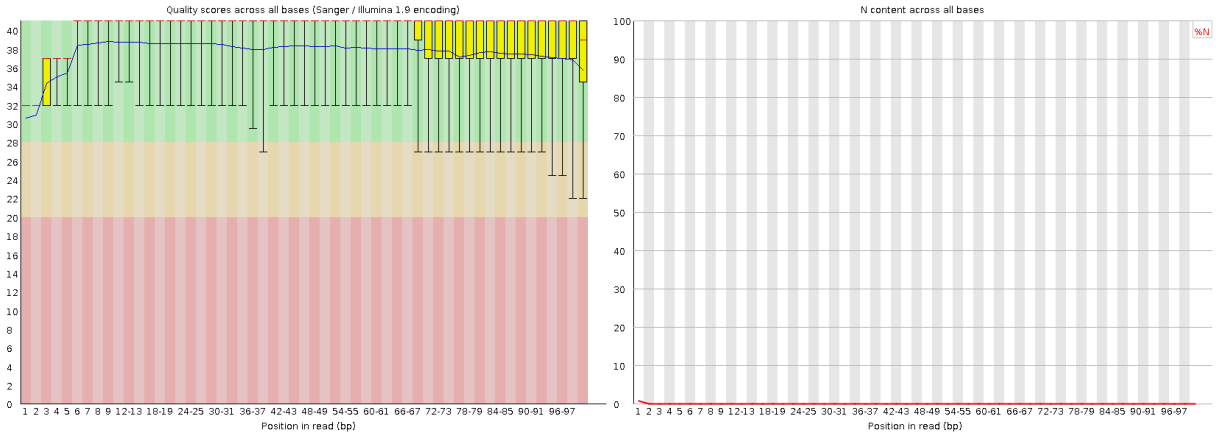


FIGURE: 21 R2 FastQC Quality Score Distribution (L) and Per-Base N Content (R) Plots

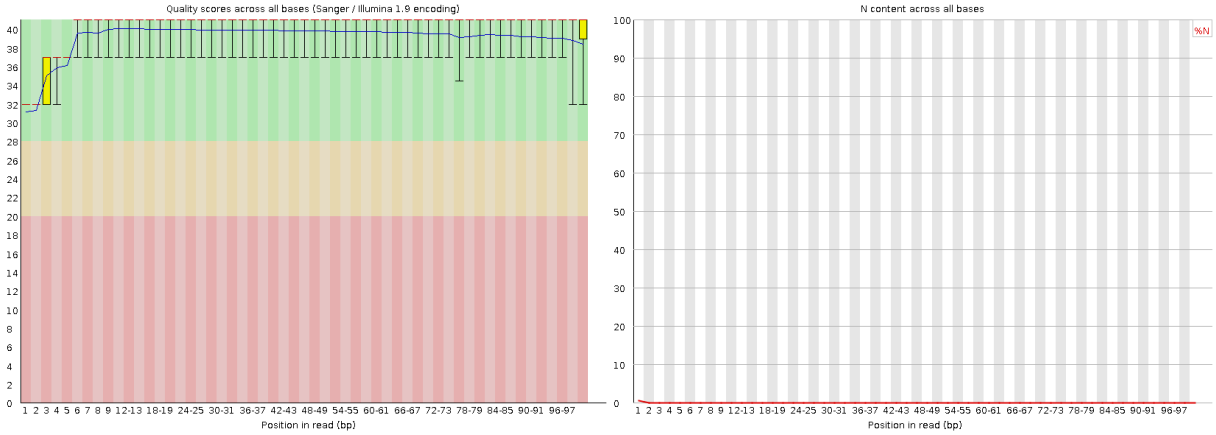


FIGURE: 34 R1 FastQC Quality Score Distribution (L) and Per-Base N Content (R) Plots

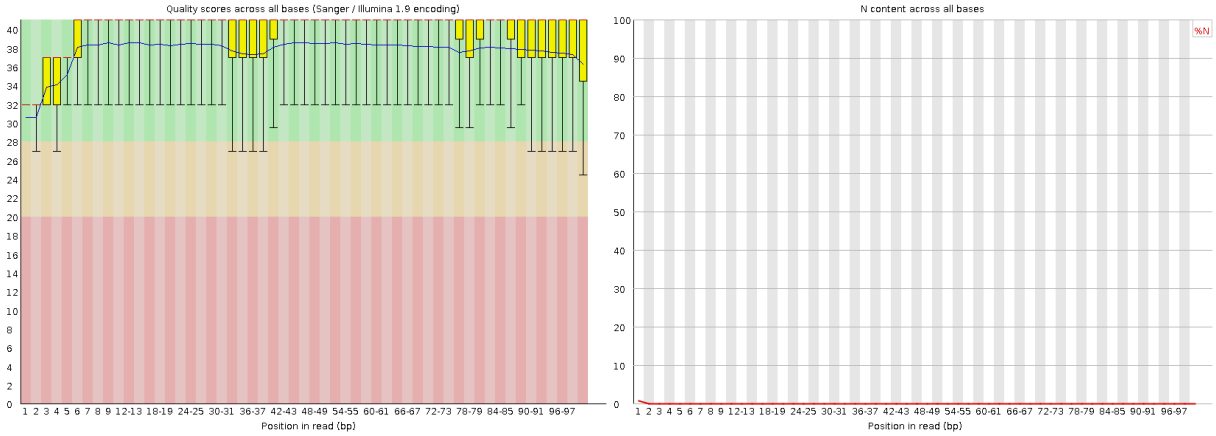


FIGURE: 34 R2 FastQC Quality Score Distribution (L) and Per-Base N Content (R) Plots

FastQC Quality Score Distribution & My Quality Score Distribution Plots

Aside from the error bars and aesthetic differences present in the FastQC plots (left), the quality score distribution plots are nearly identical. The runtime difference, however, was much larger. For FastQC, it took approximately 1:17.80 (h:mm:ss) for all 4 files. For my quality score distribution plots, it took approximately 5:20.97, 5:31.16, 5:23.32, and 5:17.10 (h:mm:ss) for each of the 4 files, respectively. A few reasons for the difference in runtime is likely due to the optimization of code for FastQC by a team of people (compared to just me), and the usage of Java as the programming language instead of Python. For the latter reason, here (https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/python3-java.html?_gl=1*zojrya*_ga*Njc5OTgzOTI2LjE2NjI1MDY4Nzk.*_gid*NzA0ODQ0Njk1LjE2NjI1MDY4Nzk.*_fplc*OXpDaFU4Mkp5R3g2RVJXaVBuZiUyQmFIVmZNaERJMlhqV are some benchmarks just to show how much faster Java is than Python 3 (hint: much faster).

Overall quality scores of the libraries are quite good, with most of the poorer quality scores occurring at the beginning of the reads. All libraries also had minor dips in quality at the end of the reads. For all libraries, there is also a slight dip in quality in the 70-80 bp position. For R2 in both libraries, there is a dip in quality occurring between the 30-40 bp position.

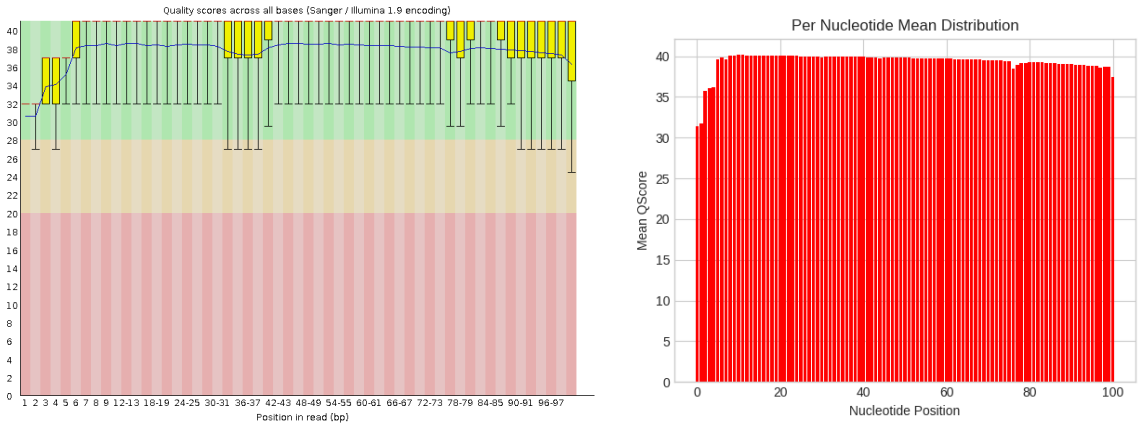


FIGURE: 21 R1 FastQC Quality Score Distribution (L) and My Quality Score Distribution (R) Plots

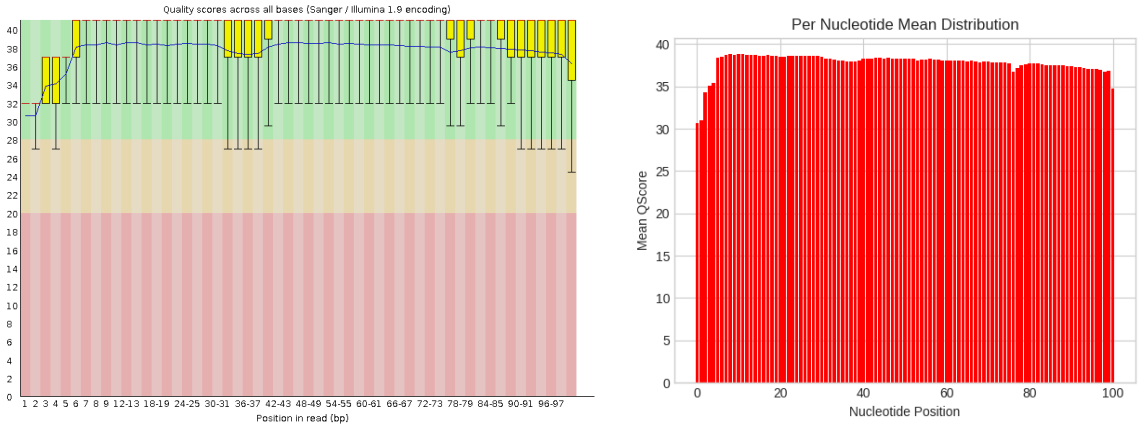


FIGURE: 21 R2 FastQC Quality Score Distribution (L) and My Quality Score Distribution (R) Plots

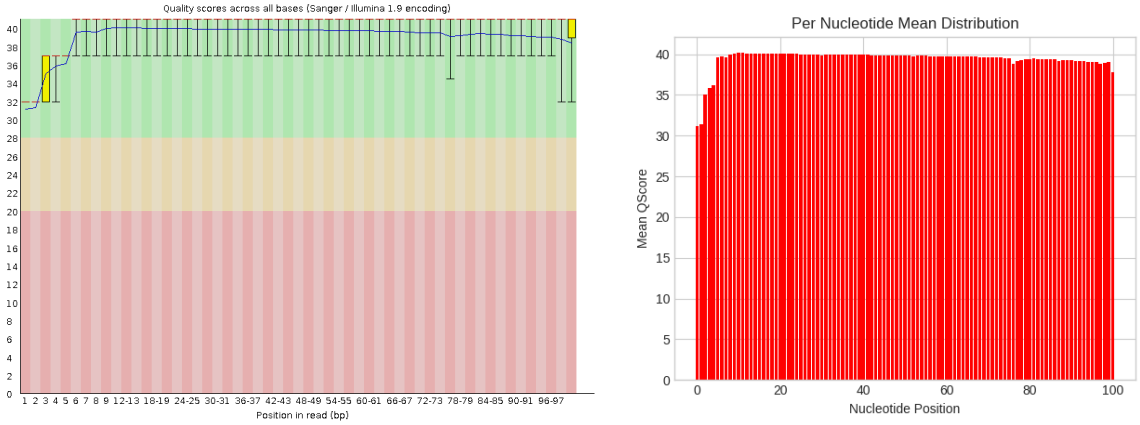


FIGURE: 34 R1 FastQC Quality Score Distribution (L) and My Quality Score Distribution (R) Plots

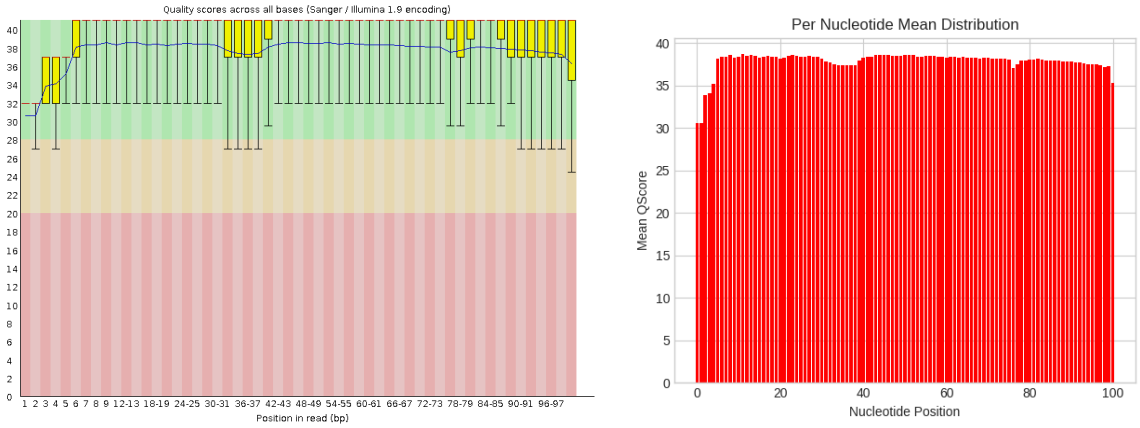


FIGURE: 34 R2 FastQC Quality Score Distribution (L) and My Quality Score Distribution (R) Plots

Part 2 – Adaptor trimming comparison

Cutadapt for Trimming Adapter Sequences

These were the adapter sequences used:

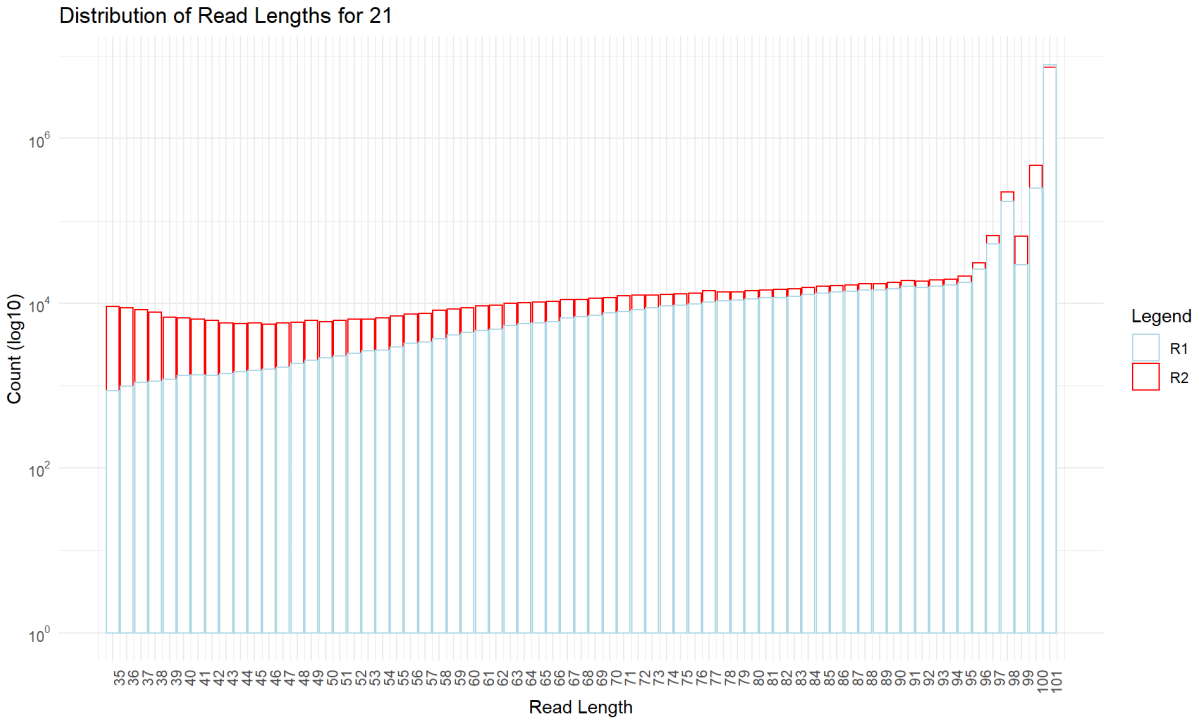
- R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
- R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

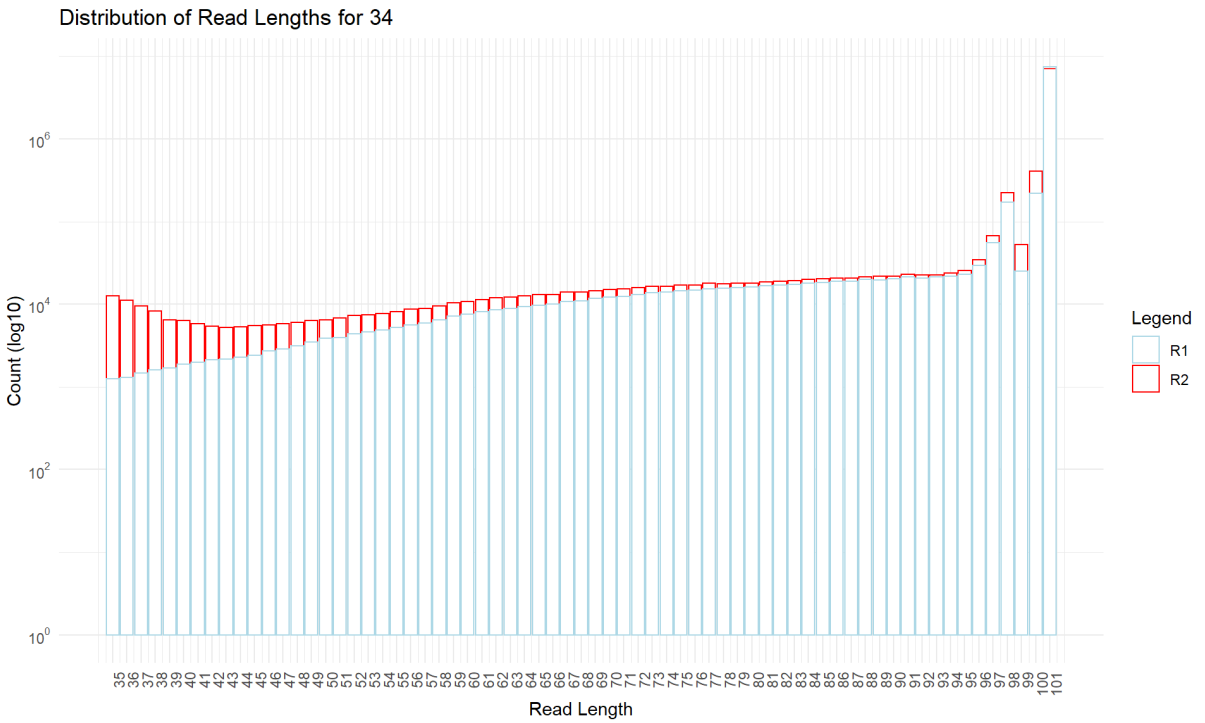
After running cutadapt, these were the results from the trimmed files:

- 21_R1
  - Total reads processed: 9,237,299
  - Reads with adapters: 613,874 (6.6%)
  - Reads written (passing filters): 9,237,299 (100.0%)
- 21_R2
  - Total reads processed: 9,237,299
  - Reads with adapters: 679,275 (7.4%)
  - Reads written (passing filters): 9,237,299 (100.0%)
- 34_R1
  - Total reads processed: 9,040,597
  - Reads with adapters: 819,166 (9.1%)
  - Reads written (passing filters): 9,040,597 (100.0%)
- 34_R2
  - Total reads processed: 9,040,597
  - Reads with adapters: 886,595 (9.8%)
  - Reads written (passing filters): 9,040,597 (100.0%)

Trimmomatic for Quality Trimming

I would expect R1s to be trimmed at a different rate than R2s. That is, for both of the plots there are more R2s present at smaller read lengths, suggesting that R2s are trimmed at a higher rate than R1s.





Part 3 – Alignment and strand-specificity

Number of Mapped and Unmapped Reads

Library	Mapped	Unmapped
21	17061173	645451
34	16822704	483578

TABLE: Mapped and Unmapped Reads of Libraries 21 and 34

Is the Data From the RNA-Seq Libraries “Strand-Specific”?

After running htseq-count on the each of the sam files twice using the parameters `--stranded=yes` and `--stranded=reverse`, two gene count files were produced for each of the sam files (four total) based on the parameters.

We can look at each of the four files and run these commands:

```
# Sum the number of reads that mapped to a feature
# 56748 is the number file lines prior to the no feature/ambiguous/etc. information (this was same for all files)
cat <file> | head -n 56748 | awk '{sum+=$2} END {print sum}'
# Calculate the total number of reads.
cat <file> | awk '{sum+=$2} END {print sum}'
```

Then you can determine the percentage of reads mapped by dividing the number of mapping reads by the total number of reads:
`(# mapping reads/total # of reads)*100`

After running these commands and calculating the percentage of mapped reads, these are the results:

- 21 `--stranded=yes`
  - (7181307/8853312)*100 = 81.11%
- 21 `--stranded=reverse`
  - (340380/8853312)*100 = 3.84%
- 34 `--stranded=yes`
  - (7214830/8653141)*100 = 83.38%
- 34 `--stranded=reverse`
  - (482525/8653141)*100 = 5.58%

Using this data, I propose that the data from both libraries are strand-specific. For library 21, this is because 81.11% of reads are forward stranded compared to 3.84% reverse stranded. For library 34, this is because 83.38% of reads are forward stranded compared to 5.58% reverse stranded. Because the difference between the forward and reverse strands is so large, this suggests that the libraries are strand-specific. If it was unstranded, the percentages would be about 50-50% between `--stranded=yes` and `--stranded=reverse`.