RNA-seq Quality Assessment Assignment

Ian VanGordon

2023-09-09

Objectives

The goal of this project is to determine whether or not this data is good enough to be used in down-stream transcriptomic analysis.

This task was broken down into 3 broad sections:

1. Assess quality score distributions
2. Trim and filter read data
3. Align transcript reads to reference genome and determine strandedness

Background

These libraries were assembled via RNA-seq by the 2017 BGMP cohort from mice. This data was sequenced by Illumina paired-end reads.

The files can be found on University of Oregon's HPC "Talapas."

File path: /projects/bgmp/shared/2017_sequencing/demultiplexed/

The following analysis was done on 15_3C_mbnl_S11_L008_R1, 15_3C_mbnl_S11_L008_R2, 17_3E_fox_S13_L008_R1, 17_3E_fox_S13_L008_R2.

Part 1

The first step was to run FastQC on the libraries and generate QScore and N content distributions per base of the files. Then, using my own python script, I calculated the average QScore per base.

15_3C_mbnl_S11_L008_R1

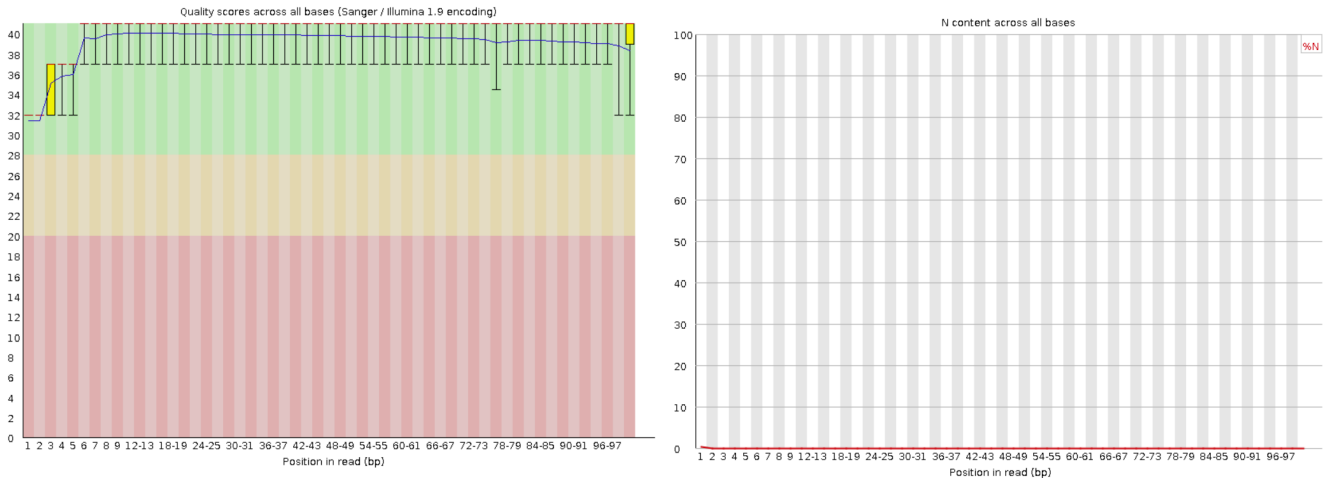

Figure 1: Plots generated by FastQC. (A) shows the average quality score per base in the read file. Error bars show 10th and 90th percentiles. Yellow boxes show INR where applicable. (B) shows the average 'N' content per base in the read file as percent.

15_3C_mbnl_S11_L008_R2

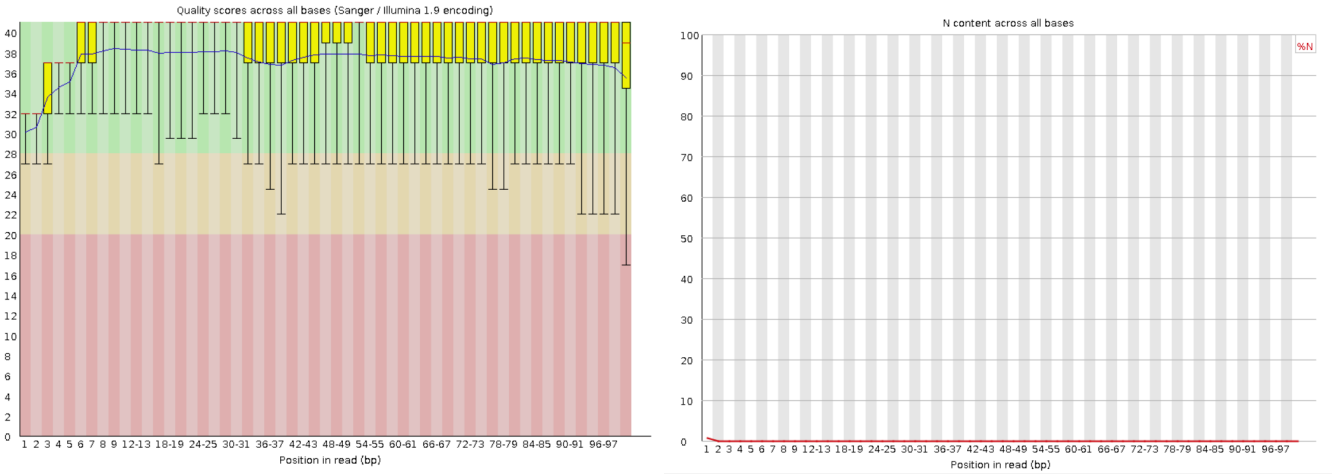

Figure 2: Plots generated by FastQC. (A) shows the average quality score per base in the read file. Error bars show 10th and 90th percentiles. Yellow boxes show INR where applicable. (B) shows the average 'N' content per base in the read file as percent.

17_3E_fox_S13_L008_R1

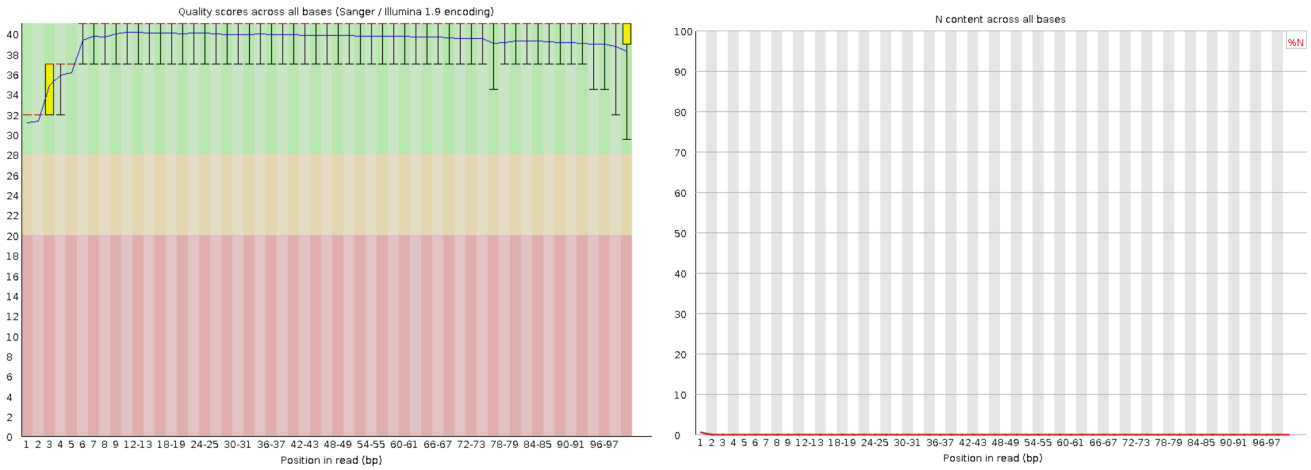

Figure 3: Plots generated by FastQC. (A) shows the average quality score per base in the read file. Error bars show 10th and 90th percentiles. Yellow boxes show INR where applicable. (B) shows the average 'N' content per base in the read file as percent.

17_3E_fox_S13_L008_R2

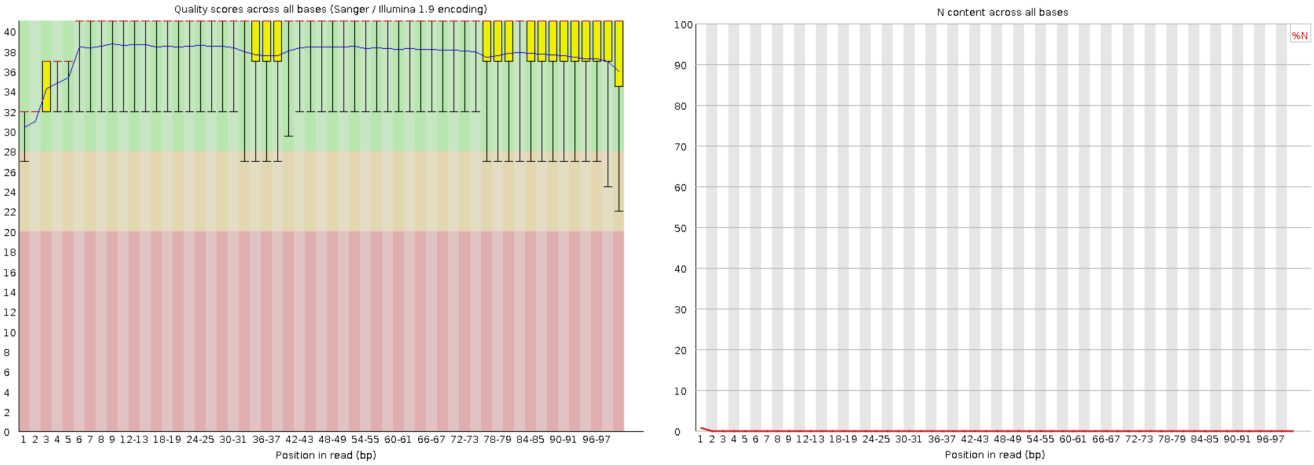

Figure 4: Plots generated by FastQC. (A) shows the average quality score per base in the read file. Error bars show 10th and 90th percentiles. Yellow boxes show INR where applicable. (B) shows the average 'N' content per base in the read file as percent.

15_3C_mbnl_S11_L008

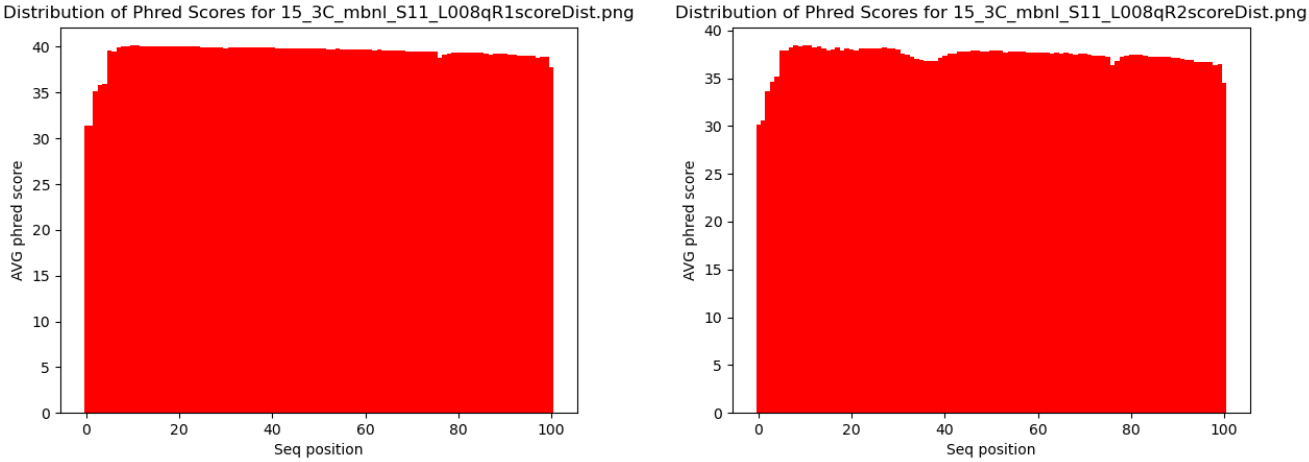

Figure 5: Plots generated with fastq_qscore_distribution.py. (A) shows the average quality score per base in the forward read file. (B) shows the average quality score per base in the reverse read file.

17_3E_fox_S13_L008

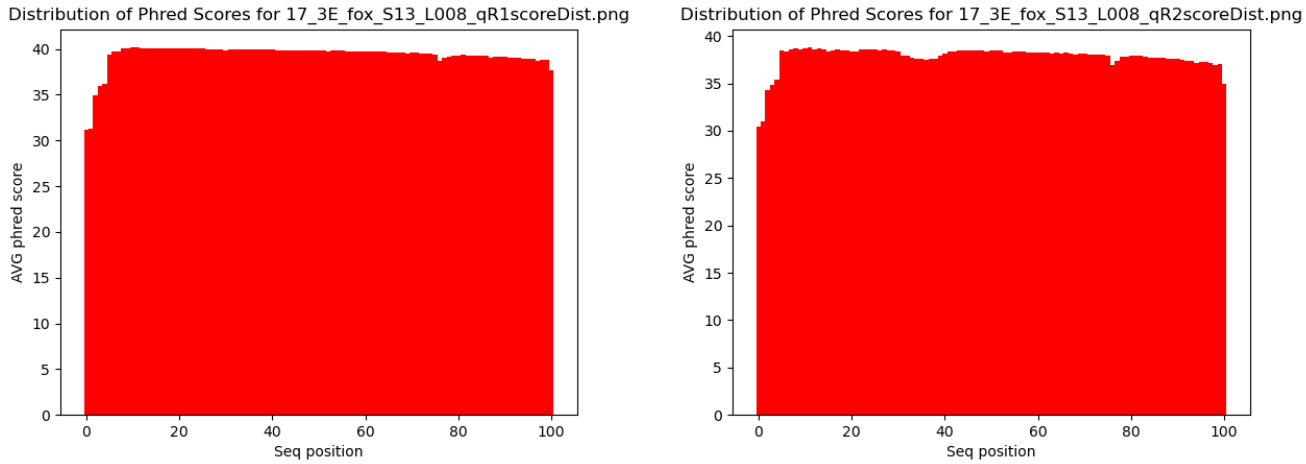

Figure 6: Plots generated with fastq_qscore_distribution.py. (A) shows the average quality score per base in the forward read file. (B) shows the average quality score per base in the reverse read file.

The quality of the beginning of all distributions are lower quality than the mean. As expected, the overall quality declines as bp number increases. Additionally, the reverse reads for each library tend to be more variable and lower quality than their forward read counter parts. Regardless, the overall quality of this data is good enough to be used in further analysis. Qscore cutoff is relatively arbitrary, and the downstream applications should be taken into consideration when making the final call.

Part 2

Next, forward and reverse reads for both libraries were trimmed and filtered using trimmomatic and cutadapt.

Using bash tools, the following proportions of reads were calculated:

```
bash commands:
wc -l <filename>
grep "^+" -B1 --no-group-separator <filename> | grep -v "^+" | awk '{if (length($0)<101){sum+=1}}END{pr
```

Table 1: The proportion of trimmed reads after removing adapters.

File	Trimmed.Reads	Total.Reads	Percent Reads Trimmed(%)
15R1	417810	7806403	5.352145
15R2	477359	7806403	6.114967
17R1	1024588	11784410	8.694436
17R2	1104503	11784410	9.372578

For both libraries the reverse reads contained more adapter sequences. This is expected as errors are more likely to occur in the second set of reads for paired-end sequencing.

Additionally, using bash, removal of the adapter sequences was confirmed from both forward and reverse reads in both libraries.

After removing adapter sequences, the files were quality trimmed.

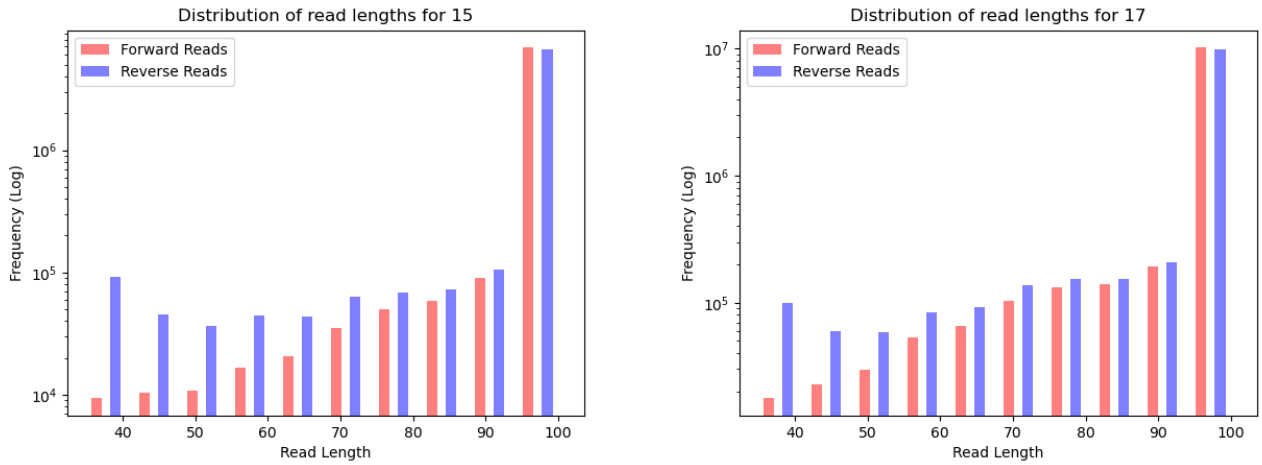

Figure 7: Plots generated using read_length_dis.py. (A) shows the distribution of reads lengths for library 15 after trimming and quality filtering. (B) shows the distribution of reads lengths for library 17 after trimming and quality filtering.

As shown by the two distribution plots, the reverse reads had many more reads with fewer lengths when compared to the forward reads.

Part 3

Next, the transcript reads were aligned to the publicly available mouse genome on ensemble (v110). This was done using star.

Running my own python script on the output sam files, I found the number of mapped and unmapped reads:

Table 2: The proportion of mapped and unmapped reads after aligning transcripts with star.

Sample	Mapped	Unmapped	Total	Percent Mapped
15	13972117	365225	14337342	97.45263
17	20950603	896279	21846882	95.89745

Table 3: Shows the count of mapped genes for different htseq-count options for primary mappings (SamF-Analysis.py).

Sample	Stranded.Option	Gene.Counts
15	yes	262956
15	reverse	5969733
17	yes	425542
17	reverse	8711216

I believe that these reads are stranded due to when htseq-count –stranded was set to reverse, orders of magnitude more of the reads mapped to the mouse genome. According to the htseq documentation:

"stranded=yes and single-end reads, the read has to be mapped to the same strand as the feature. For paired-end reads, the first read has to be on the same strand and the second read on the opposite strand. For stranded=reverse, these rules are reversed."

This means that the libraries were not prepped with Illumina kits. Kappa kits were used to assemble this sequence. –stranded-reverse option still means stranded but opposite of the Illumina-patented method.