QAA REPORT

Tam Ho

2023-09-14

Contents

PART 1: Read Quality Score Distribution **1**

PART 2: Adaptor Trimming Comparison **5**

PART 3: Alignment and Strand Specificity **6**

PART 1: Read Quality Score Distribution

29_4E_fox_S21

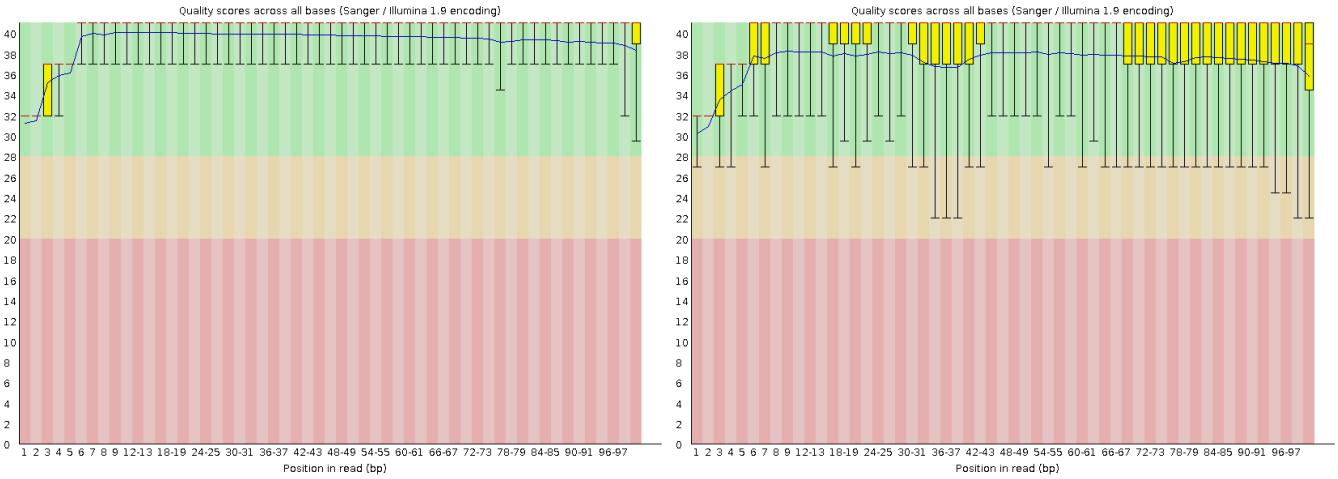

Figure 1: FASTQC- generated: Per Base Quality Score for 29-4E-fox-S21-L008 library. Read 1 (Left). Read 2 (Right). Blue line represents mean quality scores, red lines represented median quality scores, yellow box represents 25th and 75th percentiles, whiskers represents 10th and 90th percentiles

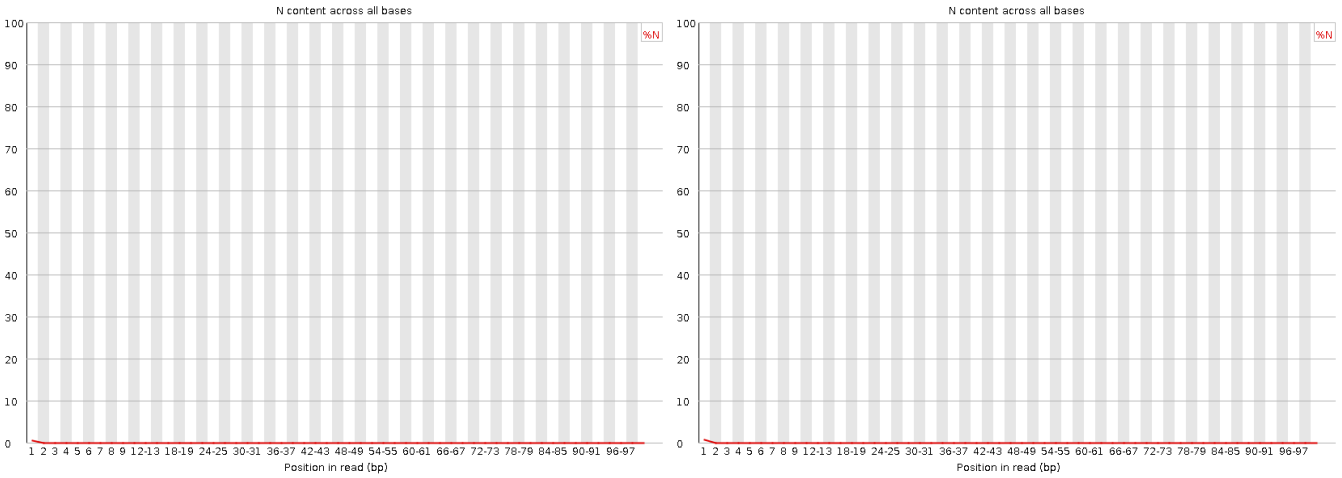

Figure 2: FASTQC- generated: Per Base N Content for 29-4E-fox-S21-L008 library. Read 1 (Left). Read 2 (Right). Red line indicates percent of N per base.

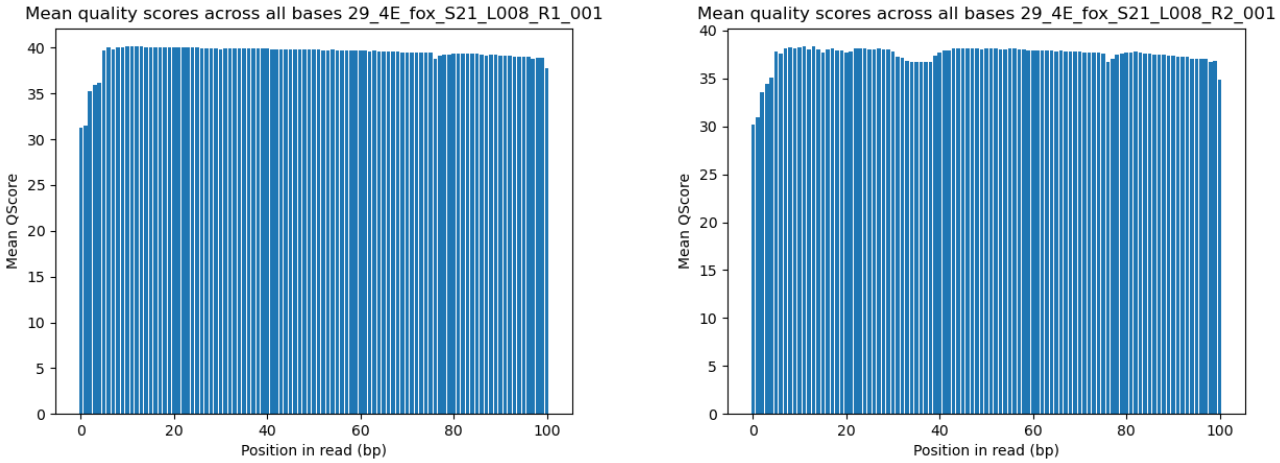

Figure 3: PYTHON- generated: Per Base Mean Quality Score for 29-4E-fox-S21-L008 library. Read 1 (Left). Read 2 (Right)

8__2F__fox__S7

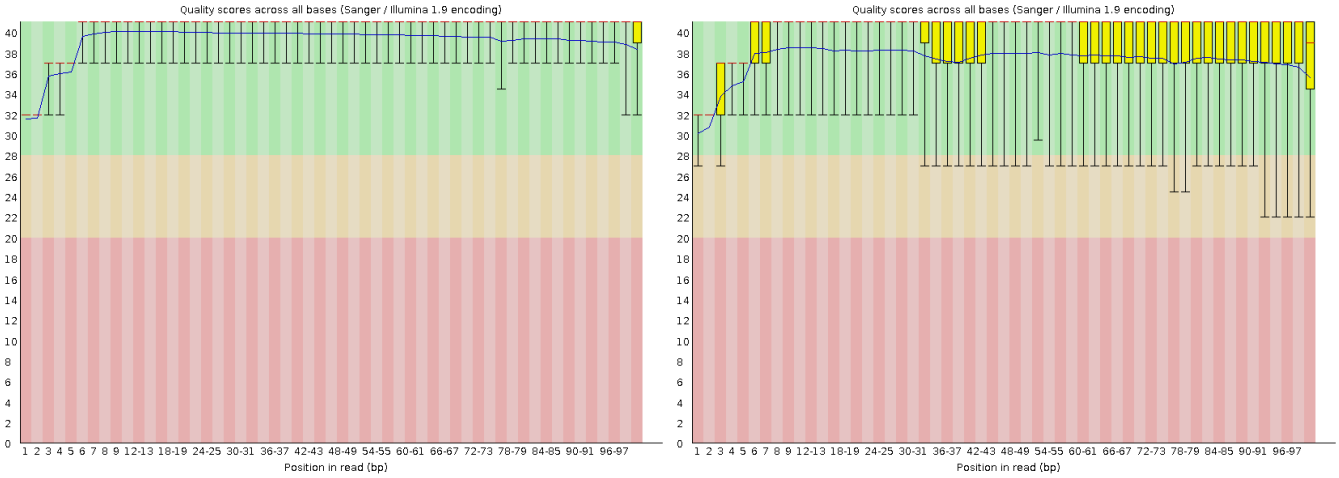

Figure 4: FASTQC- generated: Per Base Quality Score for 8-2F-fox-S7 library. Read 1 (Left). Read 2 (Right). Blue line represents mean quality scores, red lines represented median quality scores, yellow box represents 25th and 75th percentiles, whiskers represents 10th and 90th percentiles.

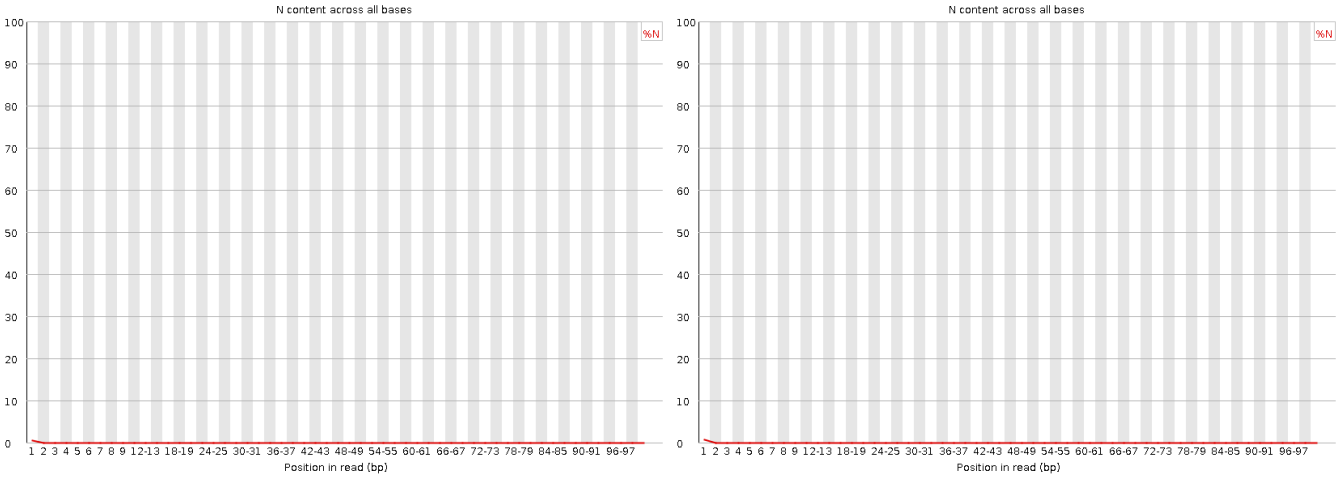

Figure 5: FASTQC- generated: Per Base N Content for 8-2F-fox-S7 library. Read 1 (Left). Read 2 (Right). Red line indicates percent of N per base.

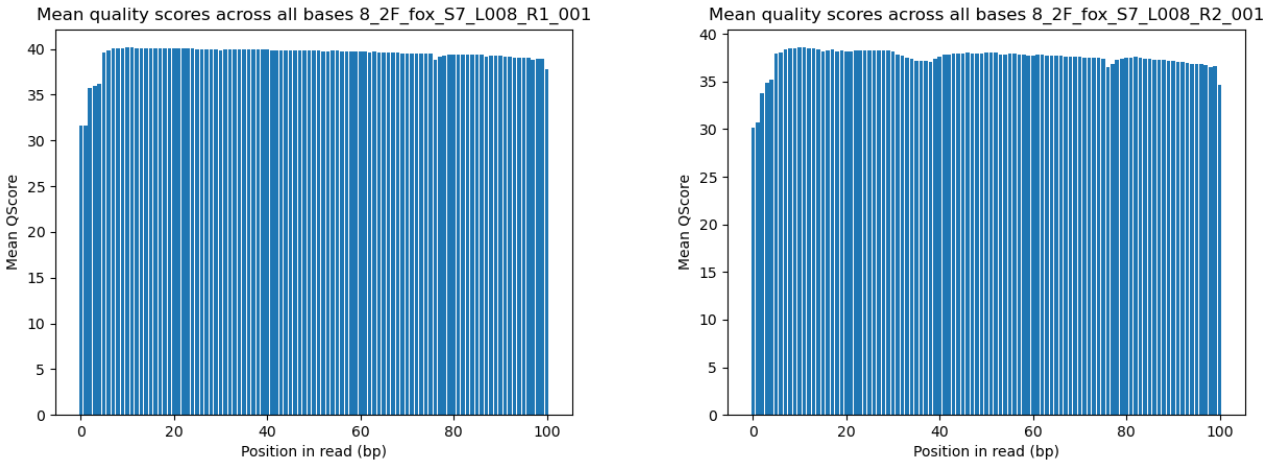

Figure 6: PYTHON- generated: Per Base Mean Quality Score for 8-2F-fox-S7 library. Read 1 (Left). Read 2 (Right)

ANSWERS (part 1)

1. Per base N content distribution is consistent with per base quality score plot. Given that there is a spike in %N in the the first base pairs and this is consistent with low mean quality score for the first few base pairs.

2. Both FASTQC and Python script (running sum strategy) produce consitent plots for mean quality score distribution per base position.

Run time summary (in seconds):

File	FASTQC	PYTHON
29_4E_fox_S21_L008_R1	25.88	90.89
29_4E_fox_S21_L008_R2	26.63	103.23
8_2F_fox_S7_L008_R1	172.88	691.86
8_2F_fox_S7_L008_R2	173.51	709.80

Run time for fastqc module is significantly faster than python script. My python script involves a list that is used to append values while iterating through every quality score line in the fastq file, whereas fastqc probably utilize some dynamic algorithm.

3. Overall report:

a. Basic Statistics
b. Per base sequence quality
c. Per tile sequence quality
d. Per sequence quality scores
e. Per base sequence content
f. Per sequence GC content
g. Per base N content
h. Sequence Length Distribution
i. Sequence Duplication Levels

j. Overrepresented sequences
k. Adapter Content
l. Kmer Content

File	a	b	c	d	e	f	g	h	i	j	k	l
29_R1	P	P	FAIL	P	FAIL	P	P	P	WARN	P	P	FAIL
29_R2	P	P	WARN	P	WARN	P	P	P	WARN	P	P	FAIL
8_R1	P	P	FAIL	P	FAIL	P	P	P	FAIL	P	P	FAIL
8_R2	P	P	WARN	P	WARN	P	P	P	FAIL	WARN	P	FAIL

From FASTQC report, it can be concluded that these libraries are good enough to be analyzed:

- Both libraries passed 8 parameters: Basic Statistics, Per base sequence quality, Per sequence quality score, Per sequence GC content, Per base N content, Sequence Length Distribution, Over-represented sequences, Adapter content.
- Problem with per tile sequence quality score may be transient since the problem seems to get better with Reads 2, so can be acceptable.
- We expect high sequence duplication levels since this is an RNA-seq experiment, so this error can be ignored.
- Mean quality score distribution does not fall below 20.

PART 2: Adaptor Trimming Comparison

Adapters sequences:

R1: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA

R2: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Confirmed by FASTQC report on adapter content where it is stated that Illumina Universal Adapters were used. By searching for "Illumina Universal Adapter Trim Sequence" we get the above sequences for cutadapt.

Using Unix commands to search for adapters (did this for both inputs and outputs of cutadapt):

- zcat $file_path_R1 | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"
- zcat $file_path_R2 | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

Result of cutadapt:

File	Percent R1 trimmed	Percent R2 trimmed
29_4E_fox_S21_L008	7.5	8.3
8_2F_fox_S7_L008	5.9	6.6

The output files from quality trimming (trimmomatic) have the following read length distribution:

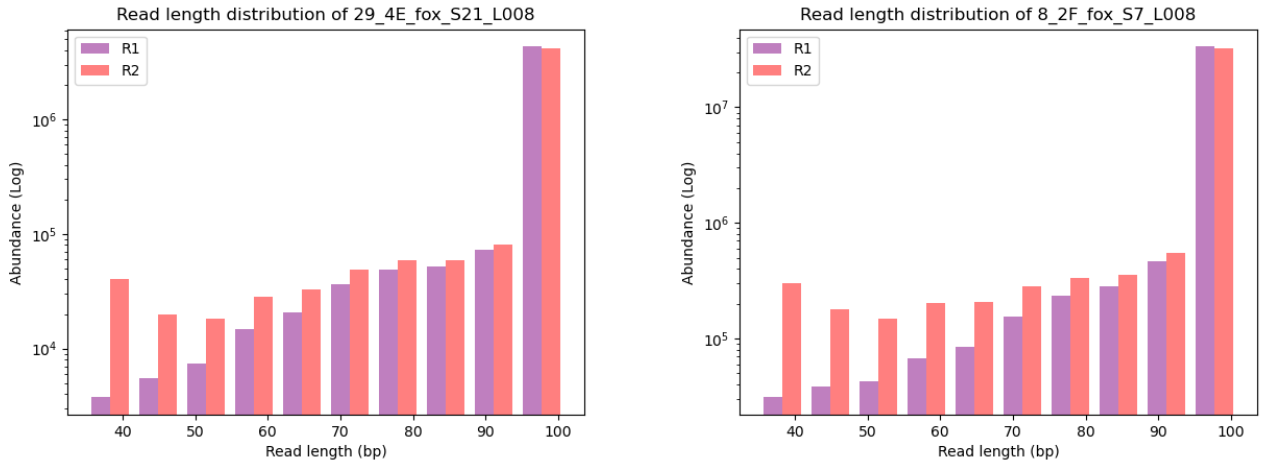

Figure 7: Read Length Distribution for 29-4E-fox-S21-L008 (left) and 8-2F-fox-S7 (right). Read 1 is shown in purple, Read 2 is shown in pink. Count of read length is displayed on a log scale.

Sequences are quality trimmed using the following parameters:

- Remove low quality bases from the beginning (first 3 bases)
- Remove low quality bases from the end (last 3 bases)
- Remove any 5 bases that has average quality score below 15
- Finally, removes any reads that falls below 35bp in length

With that in mind, it is consistent that Read 2 would be trimmed more than Read 1 since quality score distribution is lower for Read 2

PART 3: Alignment and Strand Specificity

Run STAR align

Output:

File	Number of mapped sequences	Number of unmapped sequences
Aligned 29_4E_fox_S21_L008.sam	8883008	260800
Aligned 8_2F_fox_S7_L008.sam	67070899	2511415

htseq-count was used using aligned output from STAR (SAM file) and Mouse genome gene model (GTF file) using both options: –stranded=yes and –stranded=reverse for each library

Outputs from htseq-count are tsv files with Gene ID and corresponding count:

Using unix commands: to get number of mapped reads in each file:

- cat | grep -v “^___” | awk ‘BEGIN{sumreads = 0}{sumreads+=$2}END{print sumreads}’

to get total number of reads in each file:

- cat | awk 'BEGIN{sumreads = 0}{sumreads+=$2}END{print sumreads}'

File	Number of mapped reads	Total reads	Percent mapped reads
stranded 29_4E_fox_S21_L008	185940	4571904	4.07
reverse 29_4E_fox_S21_L008	3859931	4571904	84.43
stranded 8_2F_fox_S7_L008	1260830	34791157	3.62
reverse 8_2F_fox_S7_L008	28039864	34791157	80.59

I propose that both of these data are strand-specific because 84.4% and 80.6% (in 29_4E_fox_S21_L008 and 8_2F_fox_S7_L008 respectively) of the reads are reverse, as opposed to an even (50-50) distribution of mapped reads