RNA-seq Quality Assessment / Assignment 2 Report

Kenneth Lai [BI623]

2024-09-06

Introduction

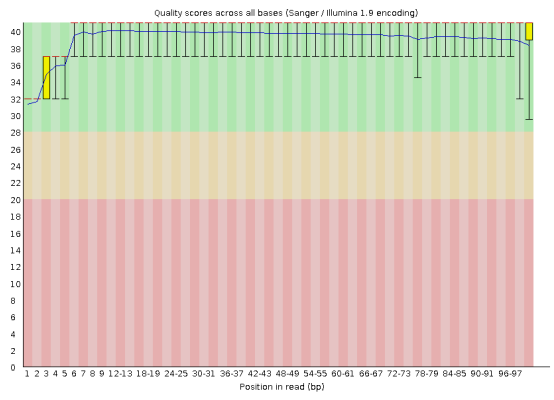
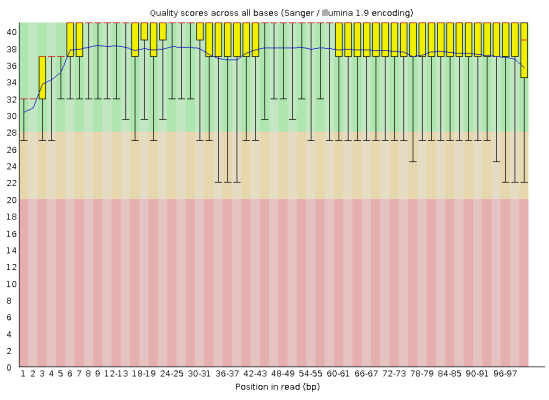
This document reports on the quality assessment of RNA-seq libraries *6_2D_mbnl_S5_L008* and *1_2A_control_S1_L008*. The analysis began with an investigation of read quality score distributions, including a comparison between customized scripts and FASTQC results. Sequences were subsequently trimmed for adapters and quality. Finally, the filtered transcripts were aligned to a mouse reference genome (from Ensemble), and a basic summary of the output data was provided.

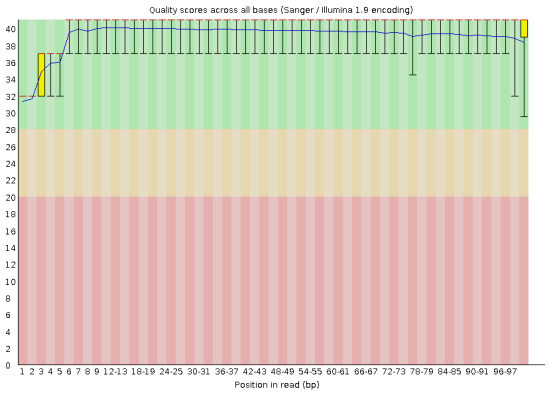
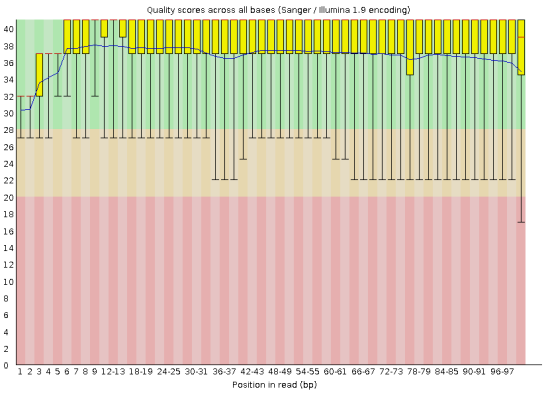
1. Read Quality Score Distributions

Commenting on the Consistency of FASTQC Plots

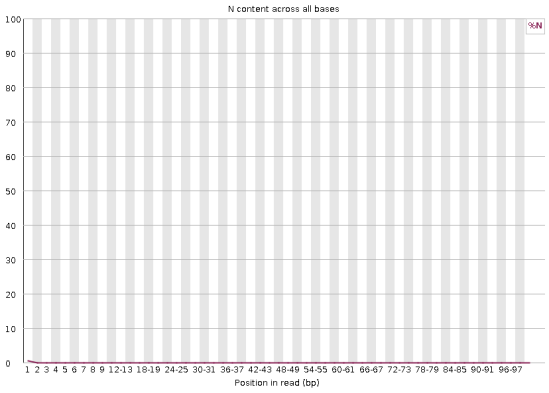
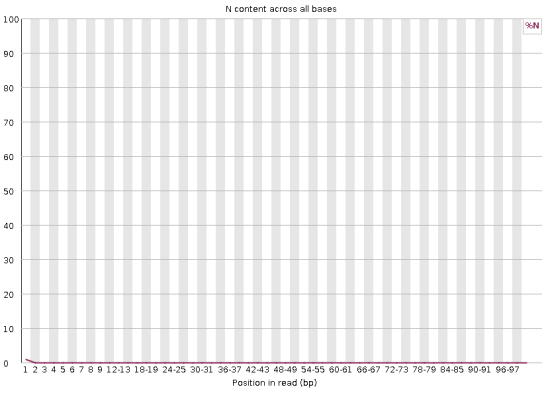
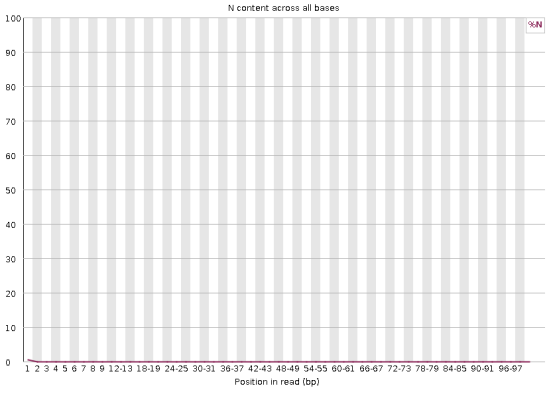
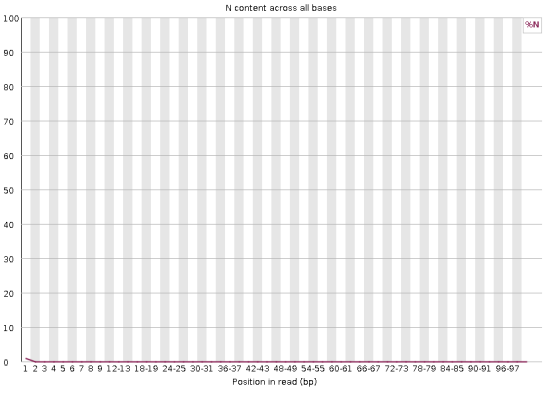
Demultiplexed R1 & R2 reads (per library) were processed through FASTQC. For our analyses, plots (produced by FASTQC) regarding per-base quality score [Fig.1] and per-base N content were compared [Fig.2].

[Figure 1] *Per base quality score plots generated by FASTQC for R1 vs. R2 of each library.* X axis represents each position in the sequence. Y-axis represents the quality score.

Library	R1	R2
6_2D		

Library	R1	R2
1_2A		

[Figure 2] ***Per base N content plots generated by FASTQC for R1 vs. R2 of each library.*** **X axis represents each position in the sequence. Y-axis represents the count of Ns in sequence for the given position.**

Library	R1	R2
6_2D		
1_2A		

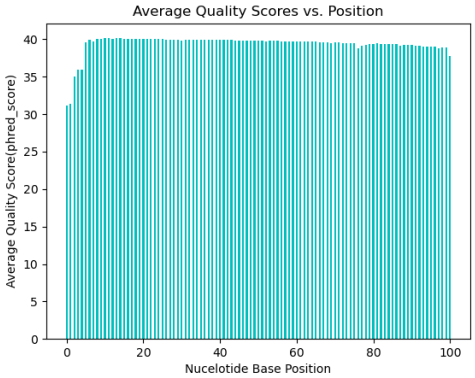
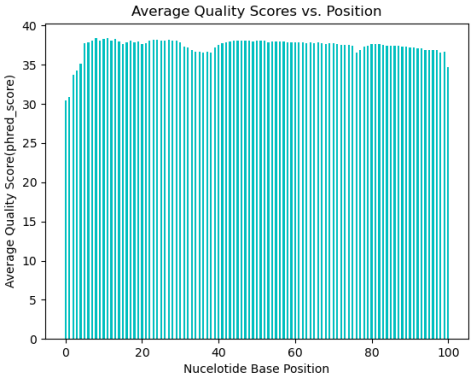
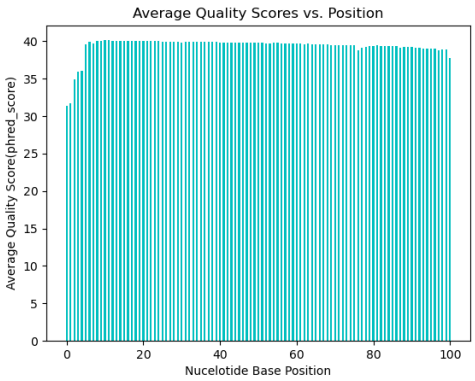
In the per base quality score plots of all four reads [Fig. 1] , scores generally exceeded 28 (green zone). As expected, R2 plots showed slightly lower quality scores than R1 (since R2 was last to be sequenced, they were exposed to degradation for longer periods of time). For the per base N content plots [Fig. 2], values were mostly 0, with only a minimal detection at the first position. This is typical, as higher N content often

correlates with lower quality scores. Thus, the N content and quality score plots are consistent for both R1 and R2 in both libraries.

Comparing FASTQC to Customized Scripts

To assess the efficiency of FASTQC, scripts which perform similar functions as FASTQC were created, specifically regarding generation of quality score distribution plots. Below are the plots generated from these scripts (from BI622 Demultiplexing Assignment) [Fig.3]

[Figure 3] ***Per base quality score plots (generated by the lab's own code) for R1 vs. R2 of each library.*** **X axis represents each position in the sequence. Y-axis represents the quality score.**

Library	R1	R2
6_2D		
1_2A		

When comparing the quality score (QS) distribution plots from FastQC [Fig.1] to our own plots [Fig.3], it became evident that both methods captured the overall trend and data points effectively. However, FastQC better excelled in several aspects. The high-resolution formatting of FastQC's visualizations enhanced the clarity of features critical for quality control. Specifically, the bins and intervals on both the x-axis (position number) and y-axis (quality scores) were smaller and more precise, allowing for accurate pinpointing of data points. Additionally, FastQC used color-coded zones to indicate quality cutoffs (green for good quality, yellow for acceptable, and red for poor quality) making it easier to identify areas of concern. In contrast, our plots were bar charts, whereas FastQC provided a scatter plot with an emphasis on error bars and trendlines.

When comparing the run time and resource usage of FastQC to our scripts, FastQC was significantly faster, completing the analysis approximately 74 seconds sooner, and it used slightly less CPU. This comparison

was based on the 1_2A R1 read. Details on how we generated our own per base quality score plots are available in Table 1.

Given these advantages, FastQC's plots were not only visually clearer but also more informative. While we could have enhanced our scripts to include similar details, the efficiency and high-quality output of FastQC, coupled with its faster processing time and lower resource usage, made it a preferable tool for quality assessment, especially when time was critical.

[Table 1] ***Runtime & CPU Usage Comparison for FASTQC vs. Our Own Scripts*** **To compare methods used to create per base quality score plots (FASTQC vs. the lab's own code), certain parameters were measured when generating our own plots. Recorded items include run time, CPU usage, and exit status. Regarding the comparative experiment, focus on rows 3 (our own script) and 5(FASTQC).**

Library	Read length	JobID	Run Time(mm:ss)	CPU Usage (%)	Exit Status	Method Used
6_2D_R1	101	15853756	2:35.65	95	0	our own script
6_2D_R2	101	15853762	2:31.28	100	0	our own script
1_2A_R1	101	15853763	1:58.37	96	0	our own script
1_2A_R2	101	15853765	1:56.37	99	0	our own script
1_2A_R1	101	15883238	0:41.95	94	0	FASTQC

Overall Assessment on Quality of Data

The overall quality of the two libraries was quite good. The per-base quality score distributions showed that most sequences fell within the higher end of the quality range. R1 generally exhibited higher per-base quality scores compared to R2, likely due to R2 being sequenced later in experiments. The total sequence counts for R1 and R2 matched across both pairs, and while there were slight inconsistencies in per base sequence content among the first 9-11 positions, these stabilized thereafter. Such variations were not unusual and could have resulted from adapter contamination or primer-dimer formation.

Sequence duplication levels were around 10-13% across all samples, possibly indicating some amplification bias. The GC content was approximately 50% for both libraries, aligning with the expected range for mouse genome sequencing and suggesting no significant contamination. All libraries had minimal N content, and adapter content was negligible, with only a small amount detected near the sequence ends in the 1_2A libraries, suggesting minimal need for adapter trimming post-FASTQC processing.

The exception was the 6_2D_R2 library, which had two over represented sequences with no known sources, making up about 0.17% and 0.16% of the genome. This low level of over representation was unlikely to indicate contamination. Overall, the data quality was high, with only minor concerns such as the over represented sequences in one library, which might require further investigation. The results were deemed reliable for further analysis.

2. Adapter & Quality Trimming Comparison

Adapter Trimming

Adapters were then removed from sequences using cutadapt (adapter sequences shown in Table2).

To confirm that cutadapt successfully removed the adapter sequences, a script (cutadapt.sh) was created to count (1) the occurrences of the expected adapter in the fastq file (if R1, we should expect R1 adapter), (2) the occurrences of the undesired adapter in the fastq file (if R1, we should expect 0 occurances of R2) to confirm the expected sequence orientations, and (3) re-count occurances of the expected adapter *after* cutadapt run. This script used the syntax of the following command to search for adapters of interest within our file: `unwanted_count=zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/6_2D_mbnl_S5_L008_R1_001.fastq.gz | awk 'NR%4==2' | grep -c 'AGATCGGAAGAGCACACGTCTGAACTCCAGTCA'`. The file name and adapter sequence would change depending on the library this command was run on. The end of the script included the following to print the resulting counts which were stored in variables `echo "Undesired Adaptor count: $unwanted_count", echo "Adaptor count BEFORE cutadapt: $pre_count", echo "Adaptor count AFTER cutadapt: $post_count"`. The resulting counts for each library are included in Table 3.

Below we reported the commands used to run the adapt.sh script on each library. For specific commands used to search for adapter sequences, please refer to cutadapt.sh(https://github.com/klai22/QAA/blob/master/cutadapt.sh)

6_2D_R1:

sbatch cutadapt.sh "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" "/projects/bgmp/shared/2017_sequencing/de
"/home/kenlai/bgmp/bioinfo/Bi623/Assignments/Assignment2/QAA/6_2D_R1_cut.fastq.gz" "AGATCG-
GAAGAGCGTCGTGTAGGGAAAGAGTGT"

6_2D_R2:

sbatch cutadapt.sh "GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" "/projects/bgmp/shared/2017_sequencing/dem
"/home/kenlai/bgmp/bioinfo/Bi623/Assignments/Assignment2/QAA/6_2D_R2_cut.fastq.gz" "AGATCG-
GAAGAGCACACGTCTGAACTCCAGTCA"

1_2A_R1:

sbatch cutadapt.sh "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" "/projects/bgmp/shared/2017_sequencing/de
"/home/kenlai/bgmp/bioinfo/Bi623/Assignments/Assignment2/QAA/1_2A_R1_cut.fastq.gz" "AGATCG-
GAAGAGCGTCGTGTAGGGAAAGAGTGT"

1_2A_R2:

sbatch cutadapt.sh "GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" "/projects/bgmp/shared/2017_sequencing/dem
"/home/kenlai/bgmp/bioinfo/Bi623/Assignments/Assignment2/QAA/1_2A_R2_cut.fastq.gz" "AGATCG-
GAAGAGCACACGTCTGAACTCCAGTCA"

[Table 2] *Adapter Sequences* that were referenced when trimming adapters with cutadapt.

Library Read	Adapter Sequence
R1	AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
R2	AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

[Table 3] *Adapter Count Data* Tracking output of the cutadapt.sh script which was used to verify libraries' orientations and the success of cutadapt for each run. Sample is synonyomous with library name in this table. If R1 was being examined, R2 could be considered the "undesired" adapter, helping us identify whether or not our read was of the expected orientation.

Sample	JobID	Run Time(mm:ss)	CPU Usage (%)	Exit Status	Undesired Adapter count	Correct Orient.	Adapter count BEFORE cutadapt	Adapter count AFTER cutadapt	Adapters Removed Successfully	Reads Trimmed
6_2D_R1	15884094	1:18.50	98	0	0	T	12870	0	T	416,045 (3.8%)
6_2D_R2	15884128	1:22.11	99	0	0	T	15377	0	T	361,205 (3.3%)
1_2A_R1	15884130	1:01.32	99	0	0	T	31837	0	T	468,835 (5.5%)
1_2A_R2	15884131	1:06.32	97	0	0	T	32536	0	T	409,400 (4.8%)

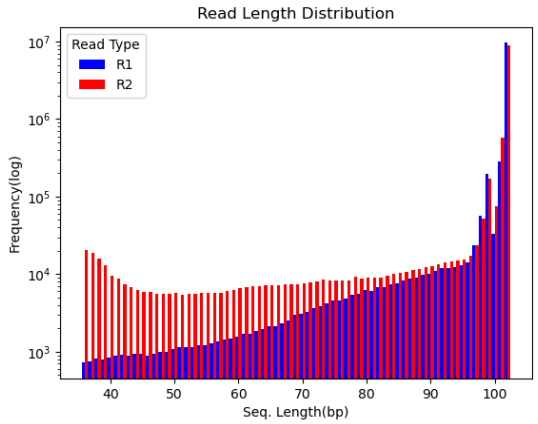
Quality Trimming

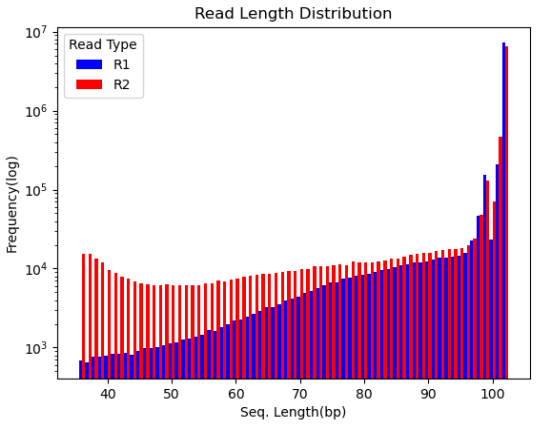
Trimmomatic was then applied to trim cutadapt-processed reads based on quality. The following parameters were used: LEADING: quality of 3, TRAILING: quality of 3, SLIDING WINDOW: window size of 5 and required quality of 15, MINLENGTH: 35 bases. Reads that were still considered *paired* post-quality trimming were then examined for read length distribution [Fig. 4] (and kept for downstream analysis).

For both libraries, THE same patterns were observed regarding their read length distributions post-quality-trimming via trimmomatic. For the longest read lengths, R1 showed a higher frequency compared to R2. Conversely, for shorter lengths, R2 tended to exceed R1's frequency. This pattern aligned with our hypothesized expectations that R2 experiences higher trimming rates due to its inherent lower quality (a result of increased exposure to degradation during sequencing). Higher trimming would reduce the number of longer reads and potentially increase the frequency of shorter reads in R2 compared to R1.

[**Figure 4**] ***Read Length Distributions (per library) after trimmomatic was applied for quality trimming.*** **X-axis represents the length of each detected read. Y-axis represents the log2() frequency of said length within the given library. Including only paired sequences. In other words, if pairs could no longer be identified for a given read after quality-trimming, they were no longer included in downstream analysis. R2 (in red) reads were more frequently observed in shorter lengths, whereas R1 (in blue) was more frequently observed in longer lengths. As expected, for these two lirbaries, R2 experienced higher trimming rates than R1. This was likeley due to the fact that R2 is sequenced last, yielding lower quality reads.**

Lib.6_2D	Lib.1_2A





3. Alignment & Strand-Specificity

Paired quality-trimmed R1&R2 transcript read files were then aligned to the mouse reference genome from Ensemble (release 112) [Mus_musculus.GRCm39.dna.primary_assembly.fa.gz] using STAR (a splice-aware aligner).

Mapped vs. Unmapped Read Counts

STAR outputs (SAM files) were examined to determine the mapped vs. unmapped read count for each library. Results can be found in Table4.

[Table 4] *Mapped vs. Unmapped Read Count Results* [Calculated using PS8 script]

Library	Mapped Reads Count	Unmapped Reads Count
6_2D	20127382	795232
1_2A	15530231	402461

Strand-specificity

Gene count tables were produced per library using `htseq --count` and subsequently used to determine the strand-specificity of our libraries.

When running htseq, the `--stranded` argument accepts the parameters `no`, `yes`, or `reverse`. To assess whether each RNA-seq library's data was strand-specific and, if so, whether it followed the reverse or standard orientation, htseq was run three times per library, each time using a different `--stranded` parameter.

We then calculated the percentage of reads which mapped to a feature for each of the three gene count tables per library. The parameter that yielded the highest percentage of mapped features would indicate the strand-specificity of the library. We created a script (mapped_features_calc.sh) to perform this analysis. The results of this method are included in Table 5.

[Table 5] ***Strand-Specificity Data*** **'Sample' refers to the library ID and the parameter used for said htseq run. '% of mapped features' indicates the amount of reads which "matched" to a gene-id in the mouse genome. The following ranking of –stranded parameters (in terms of % of mapped features) remained true for both libraries: yes < no < reversed.**

Sample	Strand-Specificity (parameter used for htseq-count to gen. input table)	JobID	Output	% of mapped features
6_2D_yes	yes	15903612	% of mapped features for 6_2D_yes: 3	3
6_2D_rev	reversed	15903624	% of mapped features for 6_2D_rev: 82	82
6_2D_no	none	15903625	% of mapped features for 6_2D_no: 79	79
1_2A_yes	yes	15903626	% of mapped features for 1_2A_yes: 3	3
1_2A_rev	reversed	15903627	% of mapped features for 1_2A_rev: 85	85
1_2A_no	none	15903629	% of mapped features for 1_2A_no: 83	83

The RNA-seq libraries of this report likely represent strand-specific libraries (6_2D & 1_2A). In both cases, applying the '–stranded=reverse' parameter to HTseq-count yielded the highest percentage of mapped features compared to other values ('–stranded=yes' & especially'–stranded-no'). Thus, the strandedness of this data is of the 'reverse' orientation, inferring that reads aligned to the strand which complemented the original RNA strand.

For 6_2D, 82% of reads mapped to features when htseq-count's –stranded parameter was = reverse.

For 1_2A, 85% of reads mapped to features when htseq-count's –stranded parameter was = reverse.

Other percentages of mapped reads yielded from different parameters can be found in Table5.

Software/Tool Versions:

Software/Tool	Version
FASTQC	0.12.1
cutadapt	4.9
trimmomatic	0.39
STAR	2.7.11b
Numpy	1.26.4
Matplotlib	3.9.2
Htseq	2.0.5

- For details on all the scripts created & used please refer to (https://github.com/klai22/QAA/blob/master/Table_of_Contents.md)