QAA

2022-09-07

Part 1: Read quality score distributions

#1 produce plots of the per-base N content, and comment on whether or not they are consistent with the quality score plots

There is almost no N content in any of these four reads. This is consistent with the quality score plots of 10_2G_both_S8_R1 and 31_4F_fox_S22_R1, both of which demonstrate relatively high quality scores. The highest quality scores with smallest error bars are in 10_2G_both_S8_R1. 10_2G_both_S8_R2 and 31_4F_fox_S22_R2 both had wider error bars and lower average scores, so you would expect a higher N content.

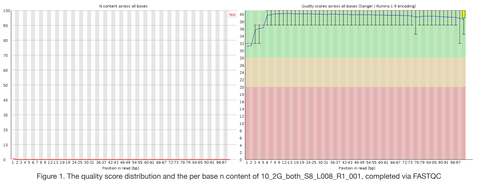

Figure 1. The quality score distribution and the per base n content of 10_2G_both_S8_L008_R1_001, completed via FASTQC

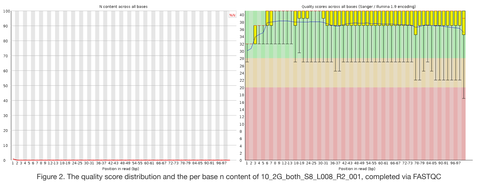

Figure 2. The quality score distribution and the per base n content of 10_2G_both_S8_L008_R2_001, completed via FASTQC

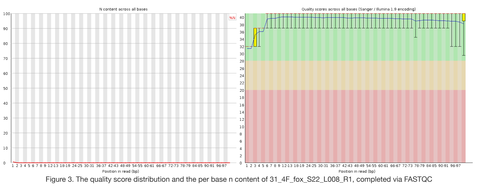

Figure 3. The quality score distribution and the per base n content of 31_4F_fox_S22_L008_R1, completed via FASTQC

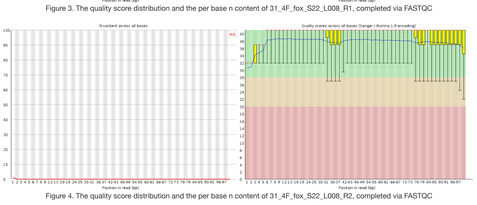

Figure 4. The quality score distribution and the per base n content of 31_4F_fox_S22_L008_R2, completed via FASTQC

#2 Describe how the `FastQC` quality score distribution plots compare to your own. If different, propose an explanation. Also, does the runtime differ? If so, why?

All of the FASTQC plots are very similar if not the exact same to my own. The run time was dramatically different, with my own script to complete all four graphs taking almost 1.5 hours while the fastqc graphs took very little. I suspect the difference is almost entirely due to the expertise of the creators of FASTQC and their extremely efficient programming.

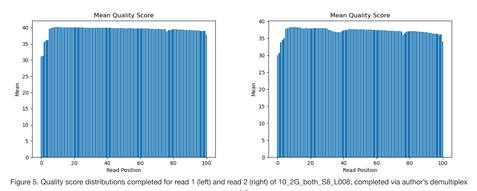

Figure 5. Quality score distributions completed for read 1 (left) and read 2 (right) of 10_2G_both_S8_L008, completed via author's demultiplex script

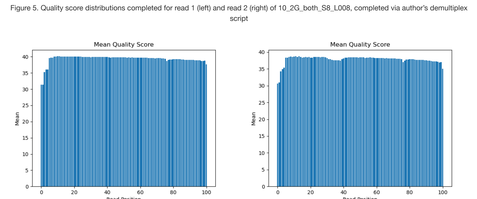

Figure 6. Quality score distributions completed for read 1 (left) and read 2 (right) of 31_4F_fox_S22_L008, completed via author's demultiplex script

#3 Comment on the overall data quality of your two libraries.

The 10_2G_both_S8_R1 library appears very high quality with minimal error. Unfortunately the 10_2G_both_S8_R2 library has high quality score distributions with a large amount of possible error, so the library may actually have much lower quality score distributions. The 31_4F_fox_S22_R1 library has high quality scores with minimal error while (similar to above) the 31_4F_fox_S22_R2 library has high quality scores with a large amount of error. So we're uncertain about the actual quality of the R2 library. Overall I would say that the two libraries have relatively good quality scores, even accounting for error.

Part 2 – Adaptor trimming comparison

#5 What proportion of reads (both R1 and R2) were trimmed?

read 1 = 2.6% read 2 = 3.4%

#*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.

By greping for the adaptor sequences in the original files, we confirm that the sequences are correct and in the expected orientation. The R1 adaptor sequence was only found in the R1 files while the R2 adaptor sequence was only found in the R2 files.

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/10_2G_both_S8_L008_R1_001.fastq.gz | grep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"

count = 23922

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/10_2G_both_S8_L008_R1_001.fastq.gz | grep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

count = 0

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/31_4F_fox_S22_L008_R1_001.fastq.gz | grep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"

count = 99875

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/31_4F_fox_S22_L008_R1_001.fastq.gz | grep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

count = 0

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/10_2G_both_S8_L008_R2_001.fastq.gz | grep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

count = 31424

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/10_2G_both_S8_L008_R2_001.fastq.gz | grep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"

count = 0

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/31_4F_fox_S22_L008_R2_001.fastq.gz | grep -c "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT"

count = 100174

zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/31_4F_fox_S22_L008_R2_001.fastq.gz | grep -c "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA"

count = 0

#7 Plot the trimmed read length distributions for both R1 and R2 reads (on the same plot). Comment on whether you expect R1s and R2s to be adapter-trimmed at different rates.

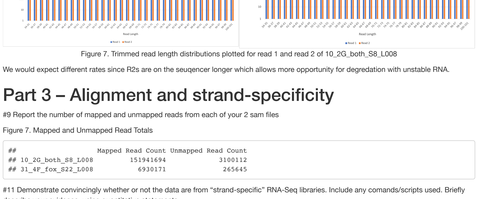

Figure 7. Trimmed read length distributions plotted for read 1 and read 2 of 10_2G_both_S8_L008

We would expect different rates since R2s are on the sequencer longer which allows more opportunity for degredation with unstable RNA.

Part 3 – Alignment and strand-specificity

#9 Report the number of mapped and unmapped reads from each of your 2 sam files

Figure 7. Mapped and Unmapped Read Totals

```
##                    Mapped Read Count Unmapped Read Count
## 10_2G_both_S8_L008           151941694             3190112
## 31_4F_fox_S22_L008            6930171              265445
```

#11 Demonstrate convincingly whether or not the data are from "strand-specific" RNA-Seq libraries. Include any comands/scripts used. Briefly describe your evidence, using quantitative statements

I propose that the library 10_2G_both_S8_L008 is strand-specific, because 3.78% of the reads are stranded (first read on the same strand as the feature and the second read on the opposite strand), as opposed to reverse (second read on the same strand as the feature and the first read on the opposite strand)(85.03%). If it was not stranded you would expect to see about the same percent of reads from stranded and reverse.

I also propose that the library 31_4F_fox_S22_L008 is strand specific, because 4.99% of the reads are stranded(first read on the same strand as the feature and the second read on the opposite strand), as opposed to reverse (second read on the same strand as the feature and the first read on the opposite strand)(80.4%). If it was not stranded you would expect to see about the same percent from stranded and reverse.

cat ./htseq/htseq_31_4F_stranded.txt | awk '$1~"ENSM" {sum+=$2} END {print sum}'

Result=179775

cat ./htseq/htseq_31_4F_stranded.txt | awk '{sum+=$2} END {print sum}'

Result=3597908

%31_4F stranded= 4.99%

cat ./htseq/htseq_31_4F_reverse.txt | awk '$1~"ENSM" {sum+=$2} END {print sum}'

Result=2893204

cat ./htseq/htseq_31_4F_reverse.txt | awk '{sum+=$2} END {print sum}'

Result=3597908

%31_4F reverse= 80.4%

cat ./htseq/htseq_10_2G_stranded.txt | awk '$1~"ENSM" {sum+=$2} END {print sum}'

Result=2928715

cat ./htseq/htseq_10_2G_stranded.txt | awk '{sum+=$2} END {print sum}'

Result=77520903

%10_2G stranded=3.78%

cat ./htseq/htseq_10_2G_reverse.txt | awk '$1~"ENSM" {sum+=$2} END {print sum}'

Result=65918552

cat ./htseq/htseq_10_2G_reverse.txt | awk '{sum+=$2} END {print sum}'

Result=77520903

%10_2G reverse=85.03%