QAA Assignment

Jeremy Jacobson

9/6/2021

Talapas Modules and Setup

Modules installed in conda environment 'QAA' in the following order using "conda install _______"

fastqc/0.11.5
easybuild
STAR 2.7.9a
numpy 1.13.1
pysam 0.13.0-intel-2017b-Python-3.6.3
matplotlib 2.0.1-Python-3.6.1
HTSeq 0.9.1-Python-3.6.1 - *installed from pip*

All work on talapas was done in an interactive node using:
srun –account=bgmp –partition=bgmp –nodes=1 –ntasks-per-node=1 –time=1-0:00:00 –cpus-per-task=1 –pty bash
*All scripts/commands were timed using "**usr/bin/time -v**"*
All scripts were batched in talapas using:
sbatch –account=bgmp –partition=bgmp –time=1-0:00:00 –cpus-per-task=1 ./_______
In the case of trimmomatic and STAR, 8 CPUs were used for multithreading functionality.

Read Quality Score Distributions

15_3C_mbnl_S11_L008 Read 1

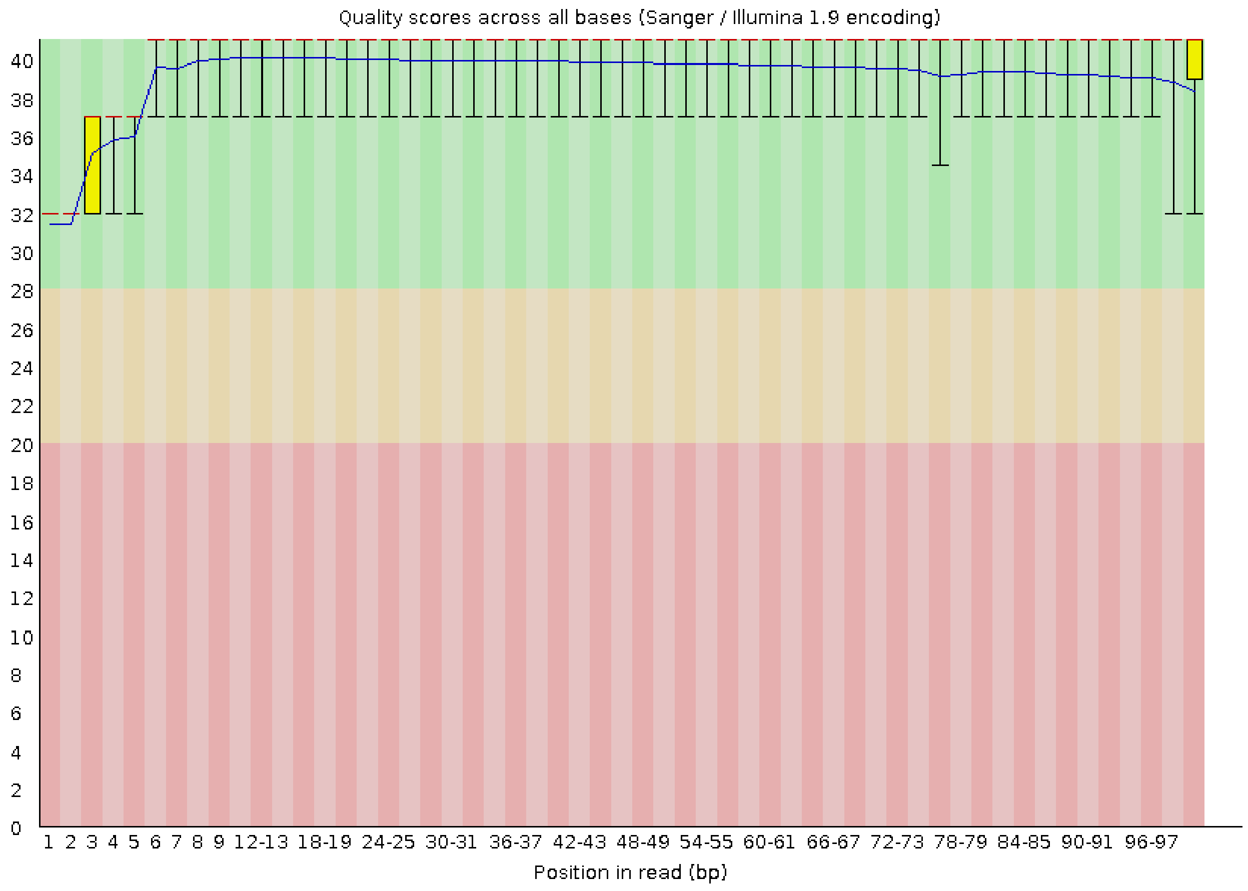


Figure 1: **Per Base Quality Score Distribution**

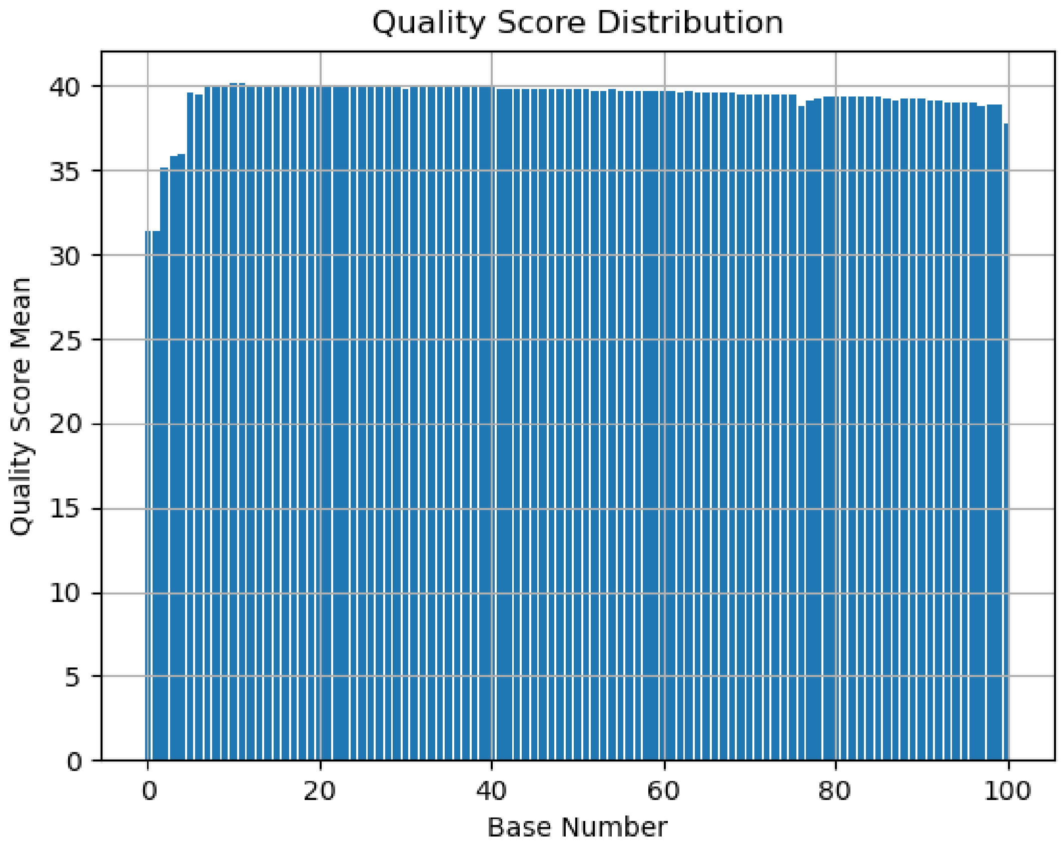
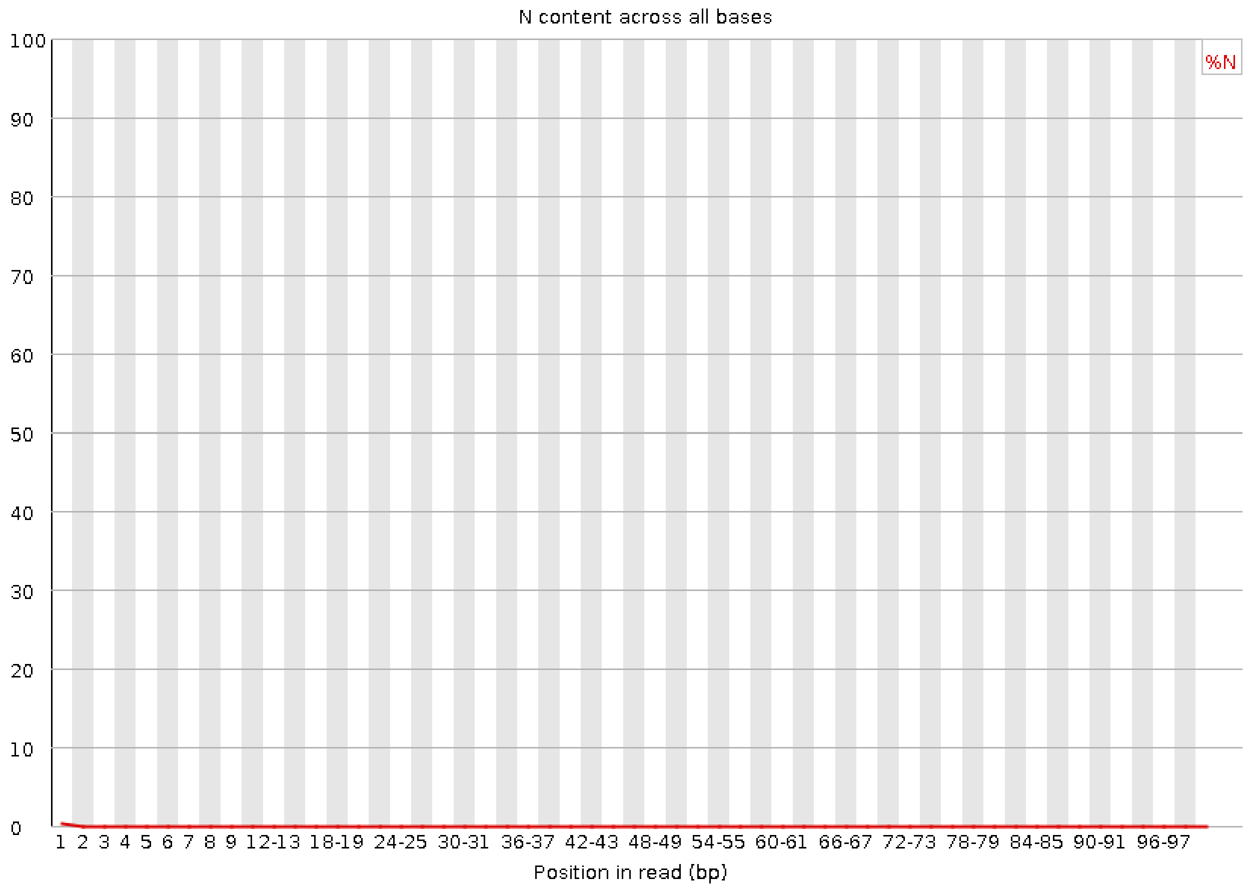


Figure 2: **Per Base Quality Score**


N content across all bases
100
90
80
70
60
50
40
30
20
10
0
%N
1 2 3 4 5 6 7 8 9 12-13 18-19 24-25 30-31 36-37 42-43 48-49 54-55 60-61 66-67 72-73 78-79 84-85 90-91 96-97
Position in read (bp)

15_3C_mbnl_S11_L008 Read 2

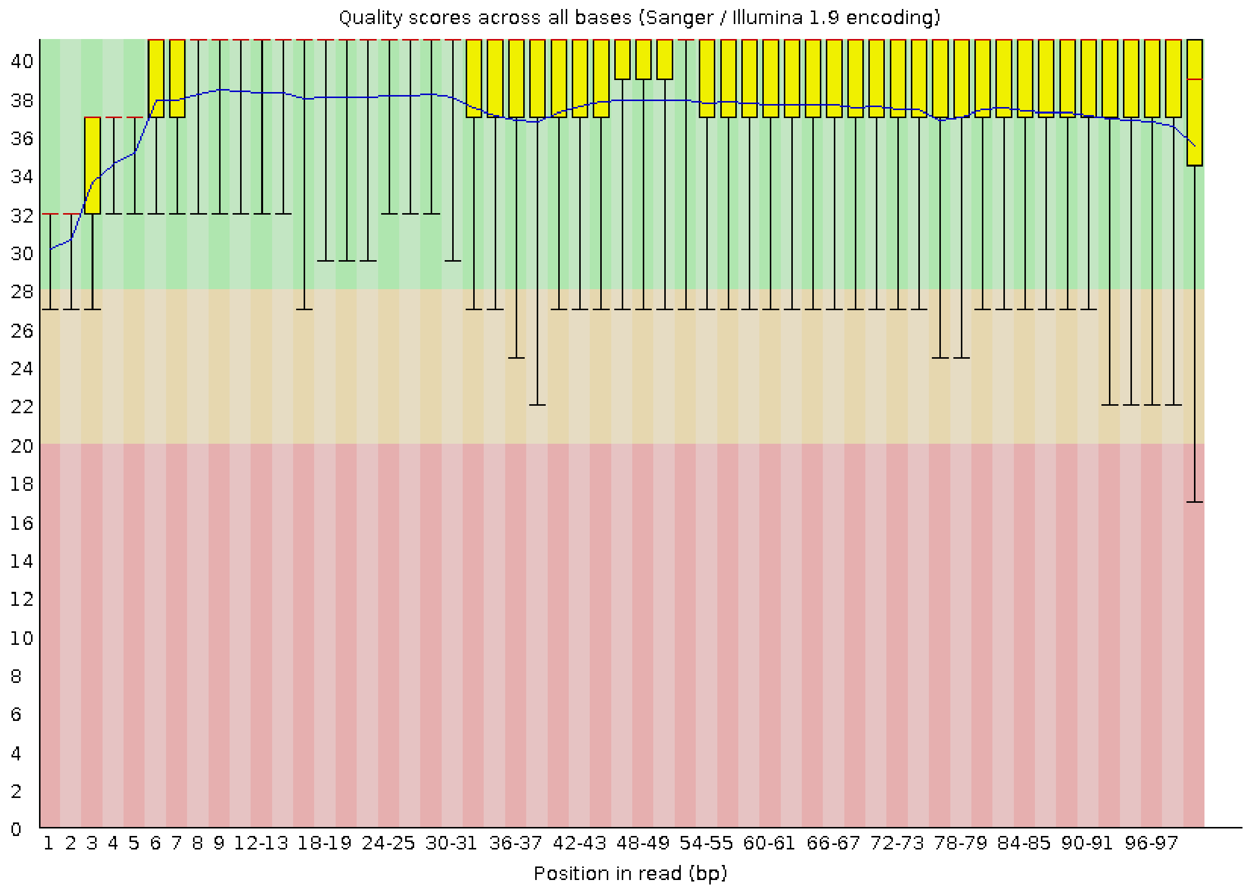


Figure 3: **Per Base Quality Score Distribution**

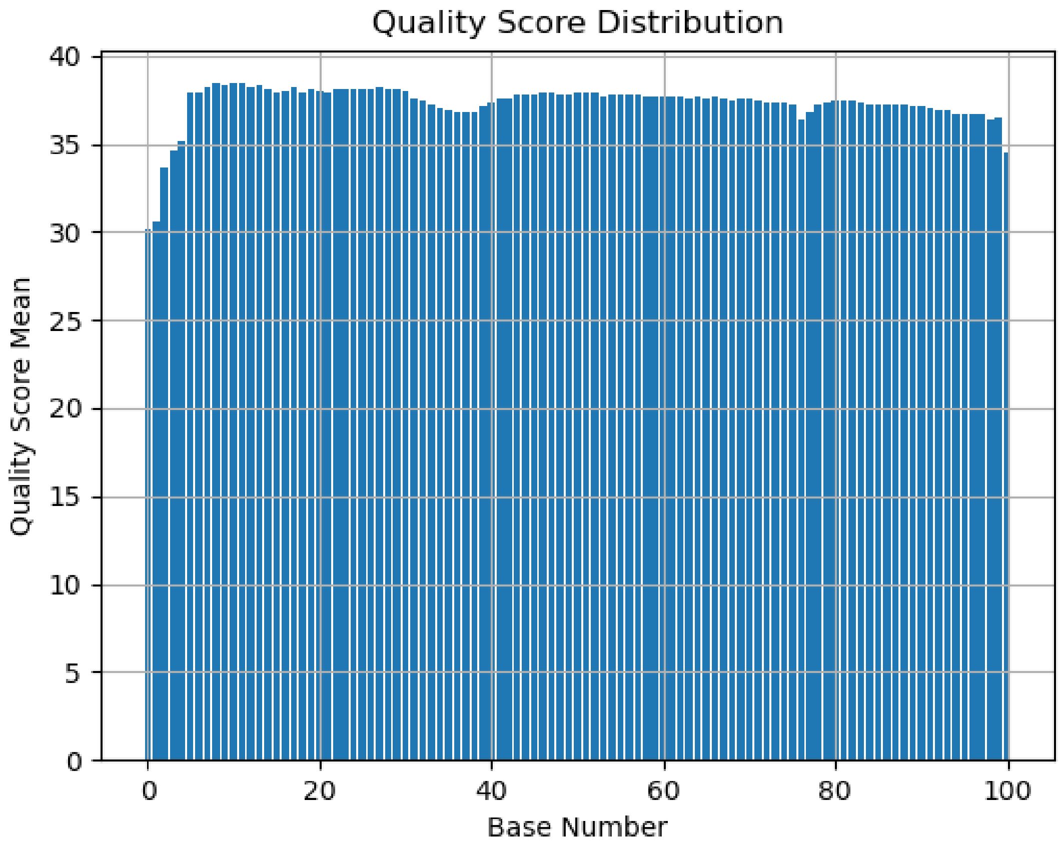
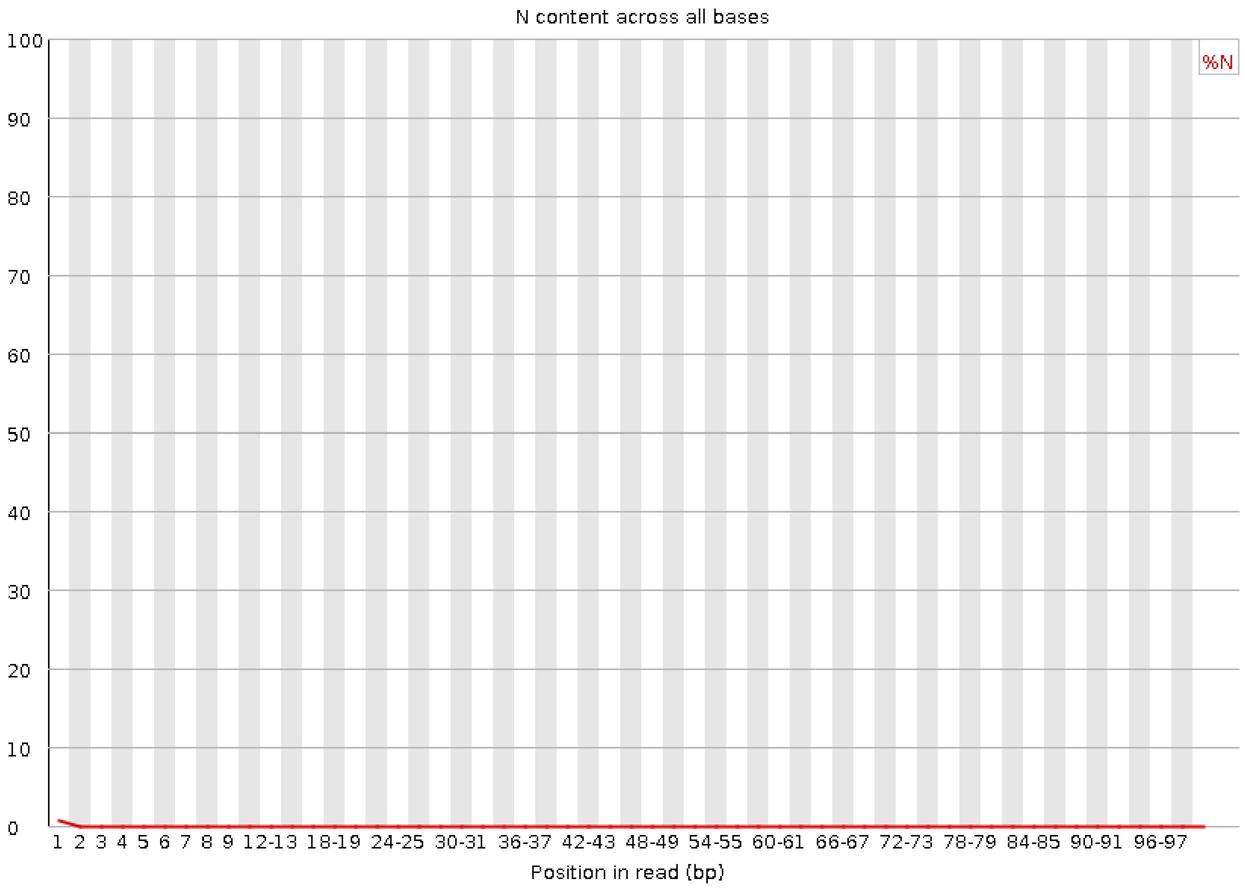


Figure 4: **Per Base Quality Score**


N content across all bases
100
90
80
70
60
50
40
30
20
10
0
%N
1 2 3 4 5 6 7 8 9 12-13 18-19 24-25 30-31 36-37 42-43 48-49 54-55 60-61 66-67 72-73 78-79 84-85 90-91 96-97
Position in read (bp)

24_4A_control_S18_L008 Read 1

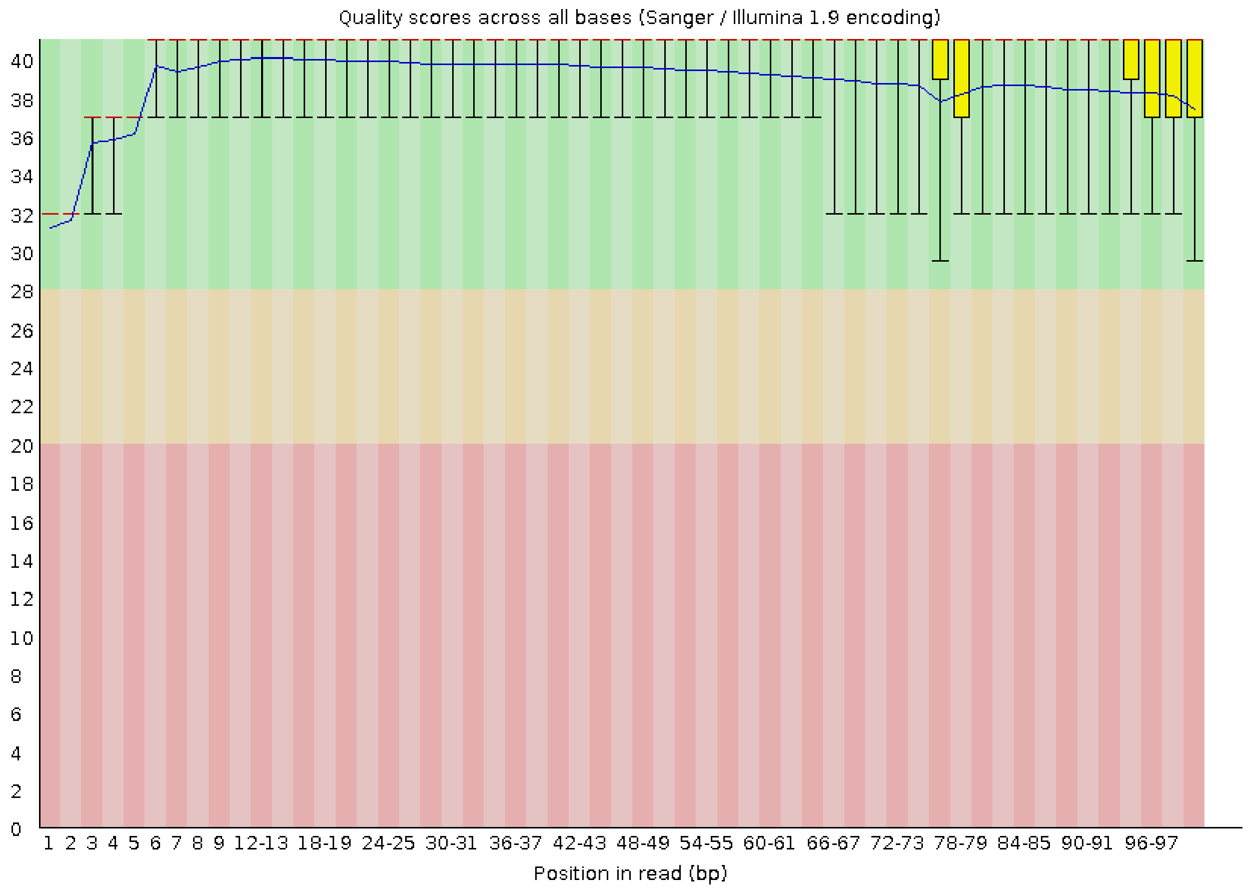


Figure 5: **Per Base Quality Score Distribution**

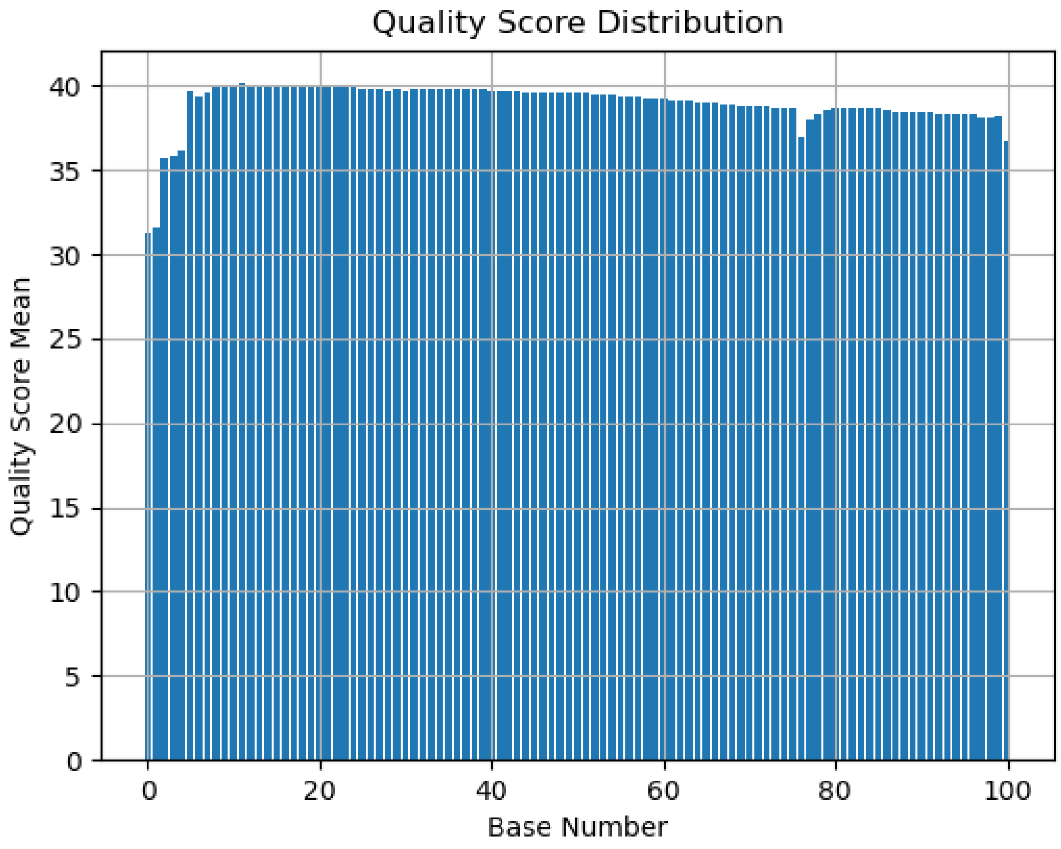
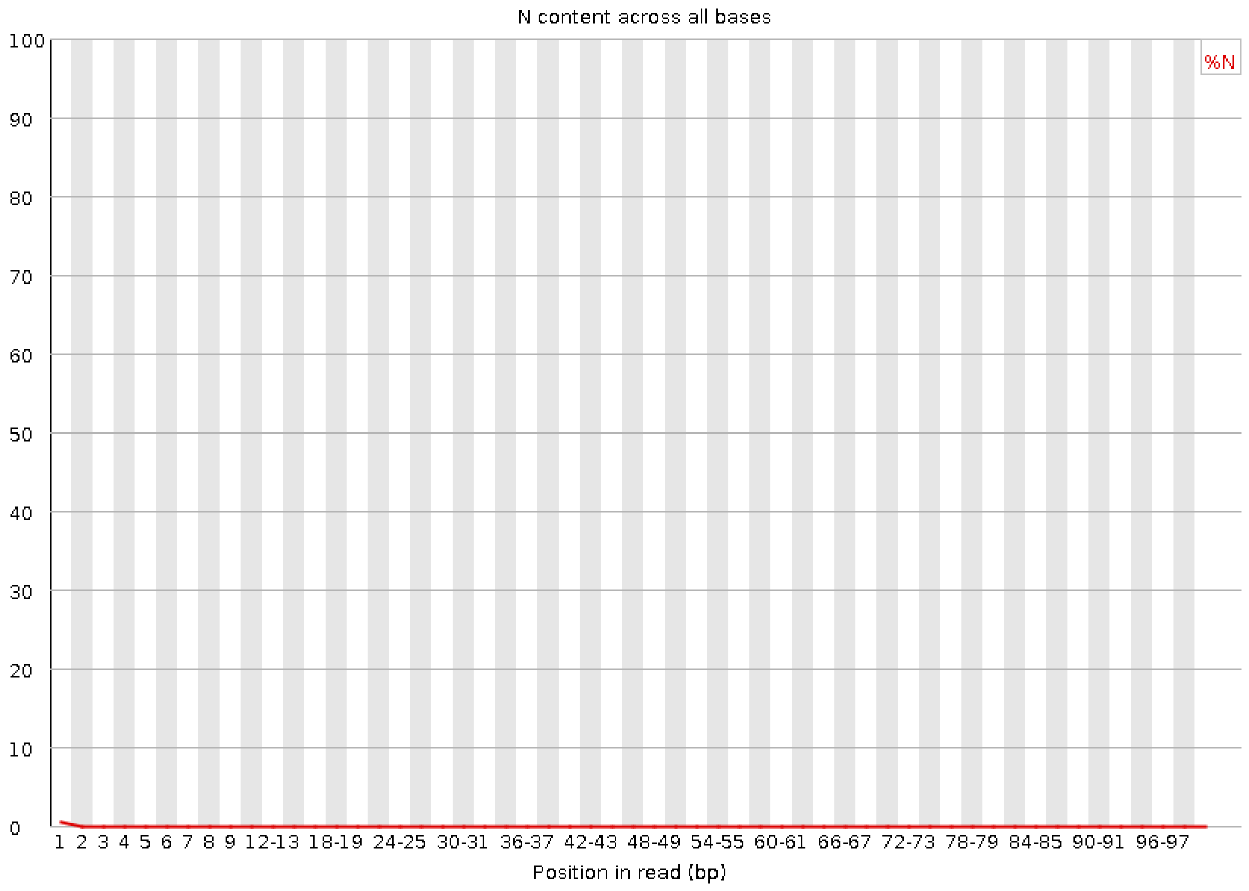


Figure 6: **Per Base Quality Score**


N content across all bases
%N
Position in read (bp)
0
10
20
30
40
50
60
70
80
90
100
1 2 3 4 5 6 7 8 9 12-13 18-19 24-25 30-31 36-37 42-43 48-49 54-55 60-61 66-67 72-73 78-79 84-85 90-91 96-97

24_4A_control_S18_L008 Read 2

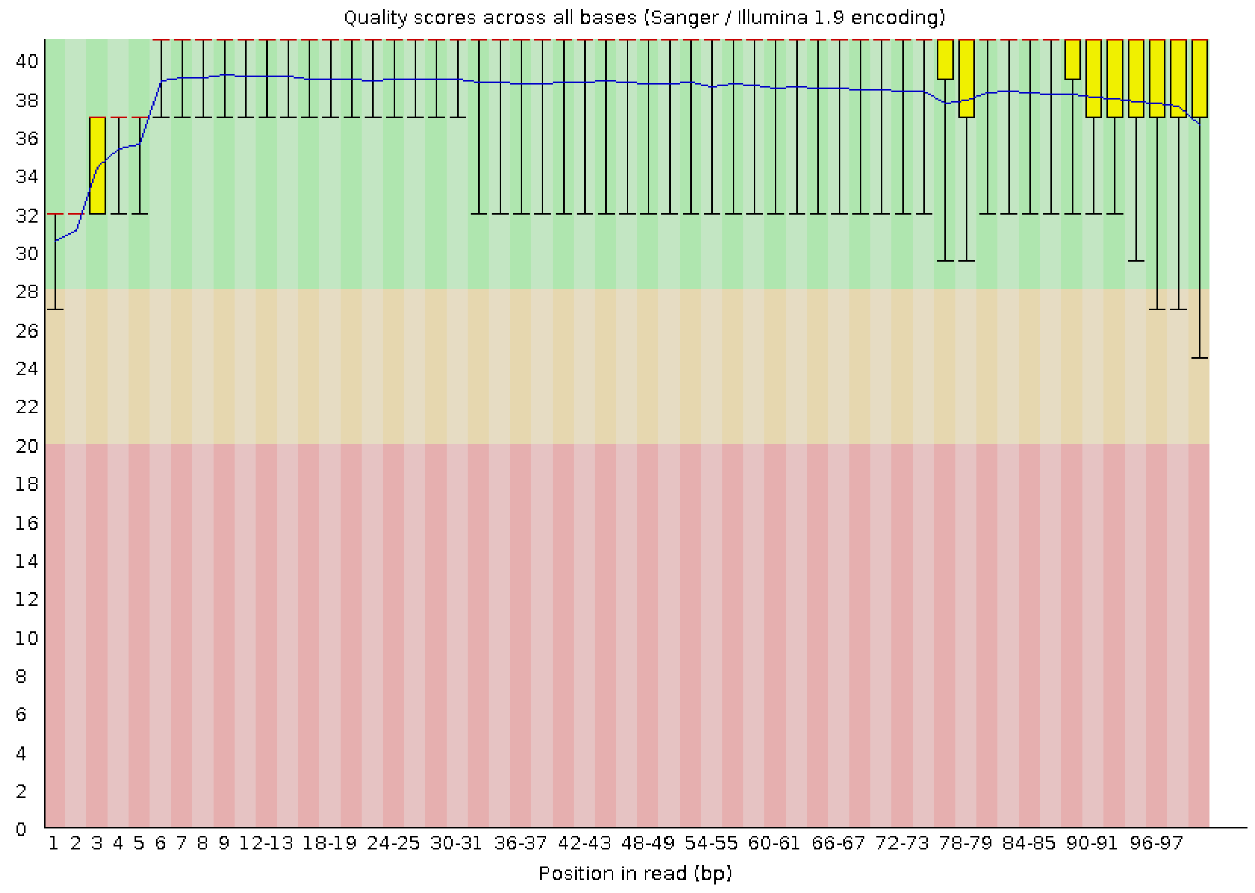


Figure 7: **Per Base Quality Score Distribution**

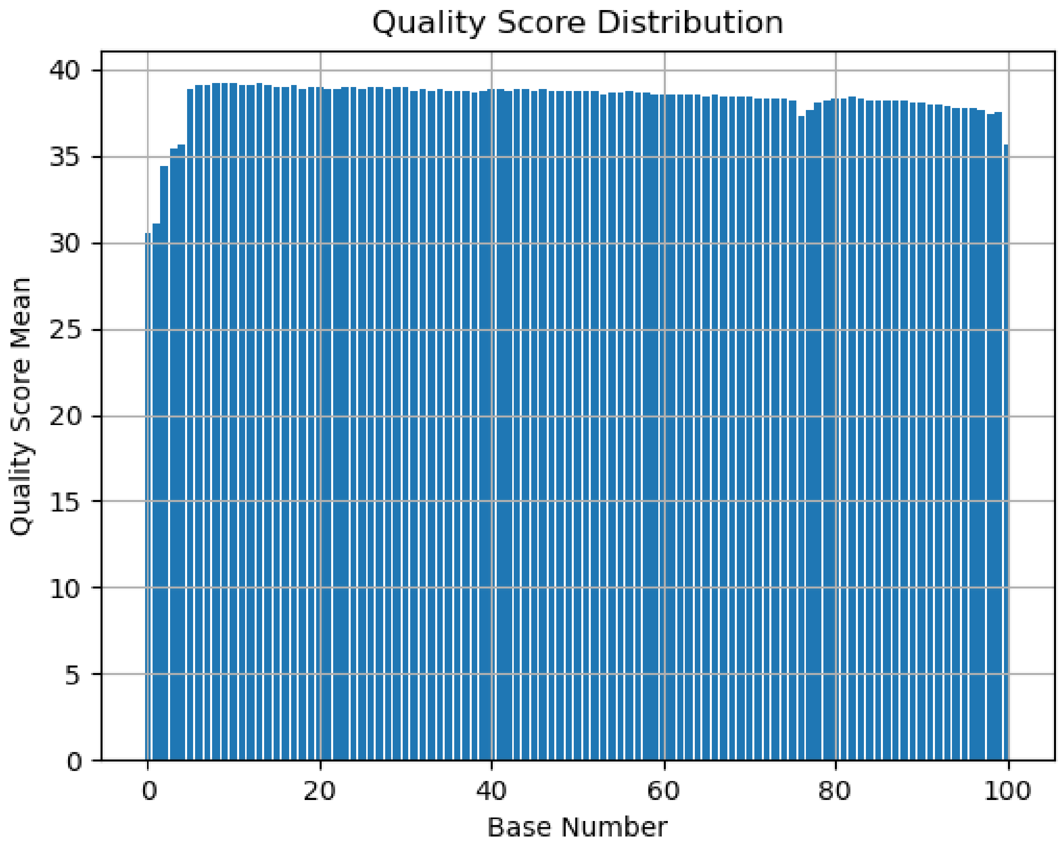


Figure 8: **Per Base Quality Score**

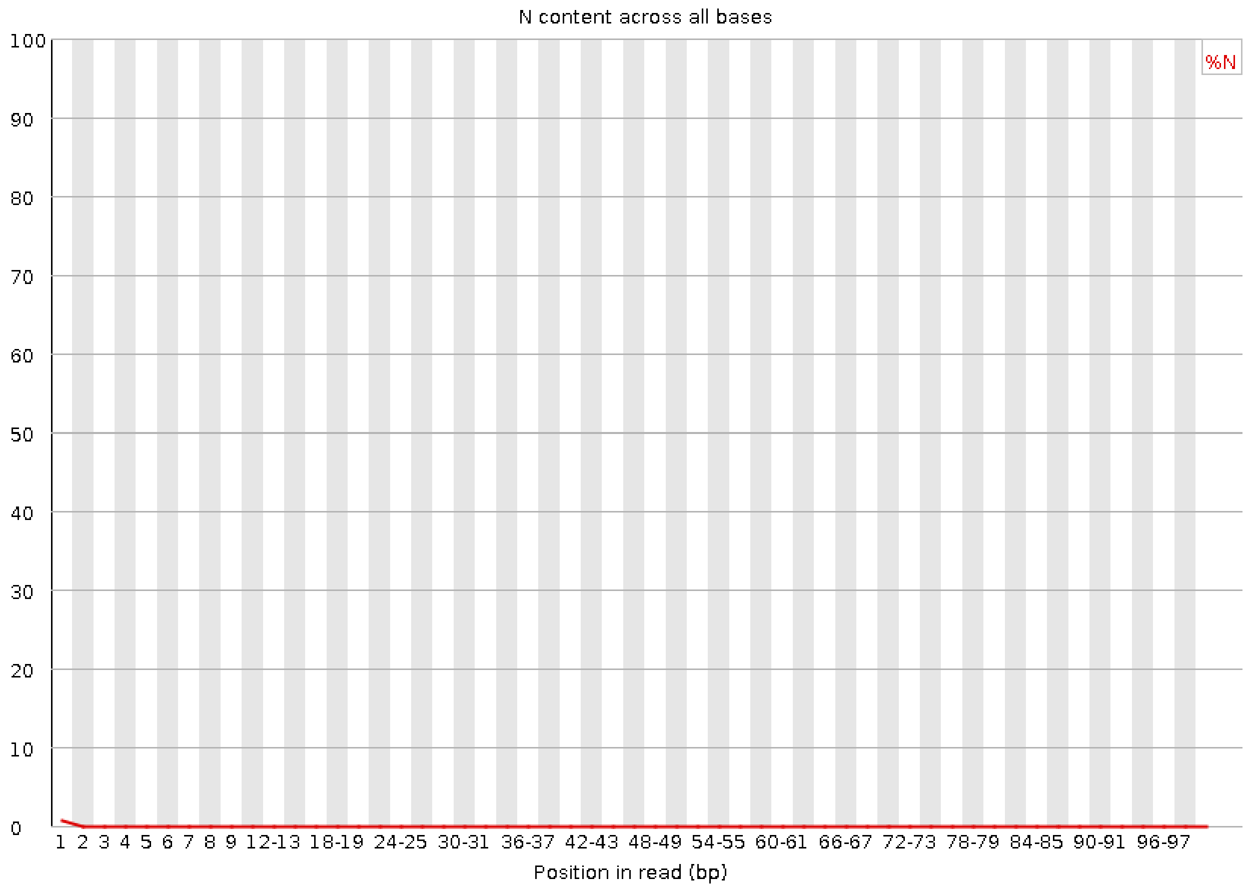


Figure 9: **Per Base N Content**

Question 1

All paired graphs are consistent with each other. Increased N content is correlated with a decreased quality score. This is most visually distinct in base position 1.
Example code:
fastqc /projects/bgmp/shared/2017_sequencing/demultiplexed/15_3C_mbnl_S11_L008_R1_001.fas
-o .

Python 'Per Base N Content' graphs were generated using Demultiplexing_Quality_score_plotter.py
"-R_file" argument was used to choose file. Output file was manually changed.
Python Code:

```
#!/usr/bin/env python import Bioinfo
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import argparse

#argparse
parser = argparse.ArgumentParser()
parser.add_argument("-R_file", default=1)

args = parser.parse_args()
file = args.R_file
```

```
#Reads with 101 characters
phred_list =[]
fred_list =Bioinfo.populate_list(file)
phred_list = fred_list[0]
#sums_list = phred_list[:]
line_count = fred_list[1]

print(“phred_list output:”, phred_list)
print(“line_count output:”, line_count)

count = 0
for value in phred_list:
phred_list[count] = value/(line_count/4)
count += 1

plt.bar(range(101), phred_list)
plt.title(‘Quality Score Distribution’)
plt.xlabel(‘Base Number’)
plt.ylabel(‘Quality Score Mean’)
plt.grid(True)
plt.savefig(“/home/jjacobso/bgmp/bioinformatics/Bi623/Assignments/QAA/24_R2.png”)
plt.show()
```

Question 2

The graphs generated using part of the demultiplexing script display nearly the same information as those produced through fastqc. The main difference is that fastqc provides a range of scores and a trendline. The python graphs allign almost perfectly to the trendline.
Runtime:
- Fastqc: 1minute 30seconds.
- Python: 2minutes 45seconds.

Fastqc is written in java which is a compiled language and generally faster than python. It was able to generate numerous figures in the time it took python to produce one.

Question3

Both libraries follow the same pattern. The quality scores near the earliest base positions are the lowest, hovering at 32 for R1 and 30 for R2. The rest of the qscores are around 39/40 for R1 and 37/38 for R2. The last nucleotide of each library and read also has lower quality. R1 is higher quality than R2 for both libraries because it was subjected to less chemicals and had less time to degrade before sequencing.

Adapter Trimming Comparison

Pt. 1: Cutadapt

Example code:

```
cutadapt -b AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -o ./15_R1.fastq
/projects/bgmp/shared/2017_sequencing/demultiplexed/15_3C_mbnl_S11_L008_R1_001.fastq
```

Average Runtime: 1minute 30seconds Results:
15_3C_mbnl_S11_L008_R1_001:
Adapter:AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -Trimmed 554754x (7.1%)
15_3C_mbnl_S11_L008_R2_001:

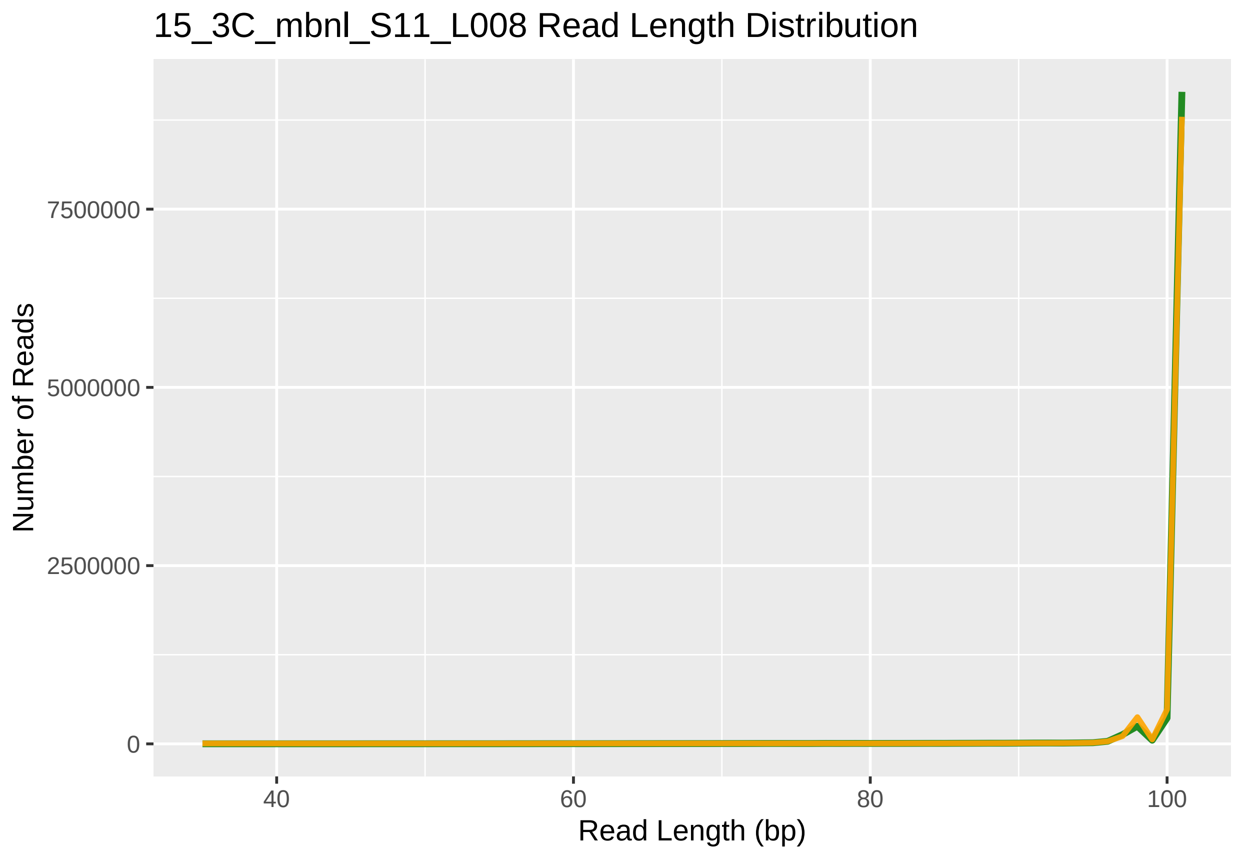
Adapter:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -Trimmed 582968x (7.5%)
24_4A_control_S18_L008_R1_001:
Adapter:AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -Trimmed 489040x (4.7%)
24_4A_control_S18_L008_R2_001:
Adapter:AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -Trimmed 598531x (5.7%)

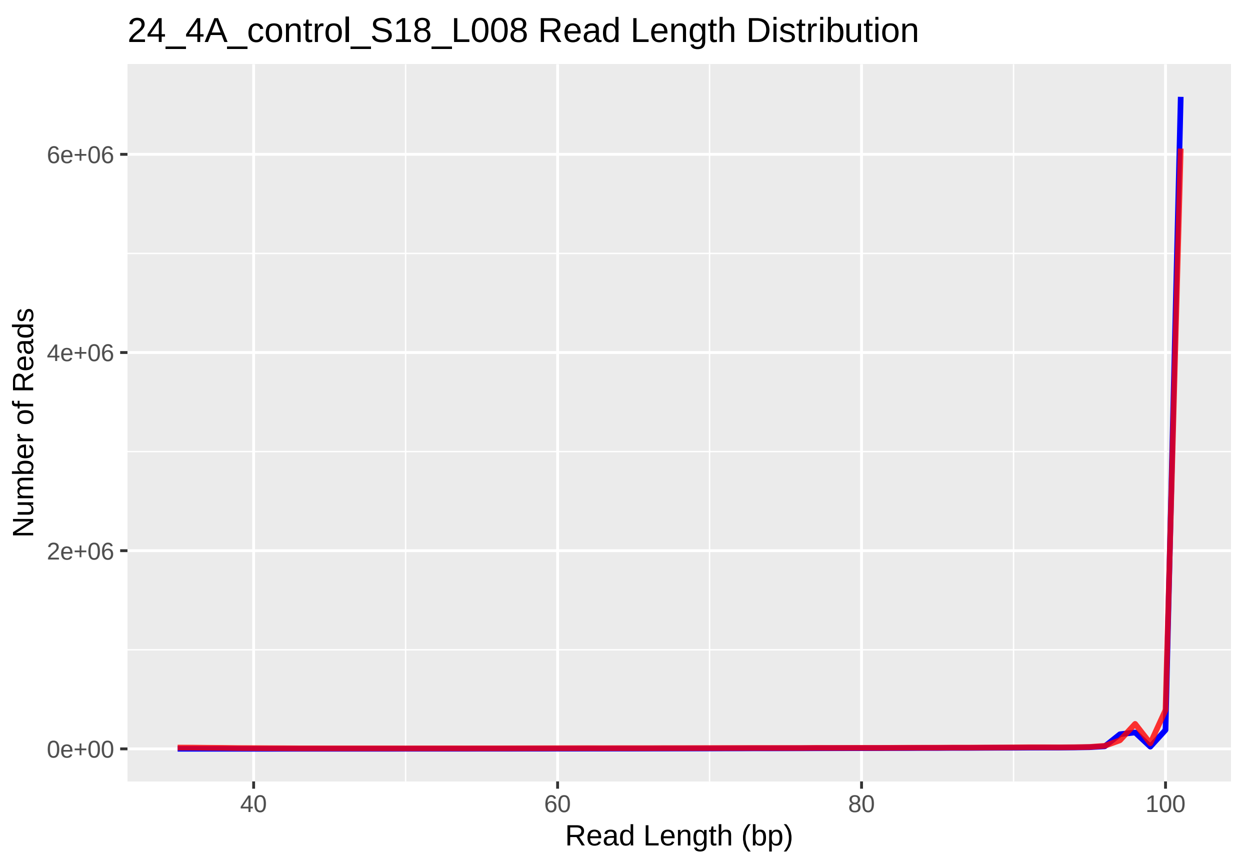
Pt. 2: Trimmomatic

Example code:
trimmomatic PE -threads 8 /home/jjacobso/bgmp/bioinformatics/Bi623/Assignments/QAA/15_R1.fas
/home/jjacobso/bgmp/bioinformatics/Bi623/Assignments/QAA/15_R2.fastq
15_R1_001_trimmo.fastq.gz 15_R1_001_untrimmo.fastq.gz 15_R2_001_trimmo.fastq.gz
15_R2_001_untrimmo.fastq.gz LEADING:3 TRAILING:3 SLIDINGWIN-DOW:5:15 MINLEN:35
Average Runtime: 4minutes 37seconds

Trimmed Reads Results Library: 15_3C_mbnl_S11_L008_R1/R2_001_fastq
Input Read Pairs: 7806403 Both Surviving: 7418603 (95.03%) Forward Only Surviving: 377796 (4.84%) Reverse Only Surviving: 5705 (0.07%) Dropped: 4299 (0.06%)
Library: 24_4A_control_S18_L008_R1/R2_001_fastq
Input Read Pairs: 10515874 Both Surviving: 10245586 (97.43%) Forward Only Surviving: 255904 (2.43%) Reverse Only Surviving: 10860 (0.10%) Dropped: 3524 (0.03%)

Trimmed Read Distribution Plot





Legend: Green: 15_3C_mbnl_S11_L008_R1 | Red: 15_3C_mbnl_S11_L008_R2 Blue: 24_4A_control_S18_L008_R1 | Orange: 24_4A_control_S18_L008_R2

In each paired library, the second read (red and yellow) had a decreased proportion of full length (101bp) sequences. Read 2 was likely trimmed more frequently due to lower quality scores resulting from increased exponsure time / chemical degradation. Read 2 likely had more adapter trimming as well. These plots also demonstrate that library 24_4A_control_S18_L008 has fewer overall reads than 15_3C_mbnl_S11_L008.

Alignment and Strand-Specificity

STAR Assembly:

Example Code:
STAR –runThreadN 8 –runMode genomeGenerate –genomeDir /projects/bgmp/jjacobso/bioinformatics
–genomeFastaFiles /projects/bgmp/jjacobso/bioinformatics/Bi623/Assignments/QAA/Alignment_stu
–sjdbGTFfile /projects/bgmp/jjacobso/bioinformatics/Bi623/Assignments/QAA/Alignment_stuff/Mu
Runtime: 19minutes 17 seconds

STAR Alignment:

Example code:
STAR –runThreadN 8 –runMode alignReads –outFilterMultimapNmax 3 –outSAMunmapped Within KeepPairs –alignIntronMax 1000000 –alignMatesGapMax 1000000 –readFilesCommand zcat –readFilesIn /home/jjacobso/bgmp/bioinformatics/Bi623/Assignmen /home/jjacobso/bgmp/bioinformatics/Bi623/Assignments/QAA/15_R2_001_trimmo.fastq.gz –genomeDir /projects/bgmp/jjacobso/bioinformatics/Bi623/Assignments/QAA/Alignment_stuff/Mus –outFileNamePrefix Mus_musculus_15_R2_001
Runtime: 1minute 16seconds

Mus_musculus_15_001_Aligned.out.sam

Mapped Reads: 14432097 (97.3%)
Unmapped Reads: 405109 (2.7%)

Mus_musculus_24_001_Aligned.out.sam

Mapped Reads: 19778684 (96.6%)
Unmapped Reads: 712488 (3.4%)

Mapped and Unmapped reads were found using Map_reader.py

```
#! /usr/bin/python3.6
alignment_tracker = {}
mapped = 0
unmapped = 0
with open ("/projects/bgmp/jjacobso/bioinformatics/Bi623/Assignments/QAA/Alignment_stuff/Mus_musculus_24_001A
"r") as fh:
for line in fh:
if not line.startswith("@"):
line = line.split("")
flag = (int(line[1]))
if((flag & 4) != 4) and ((flag & 256) != 256):
mapped +=1
else:
if ((flag & 256) != 256):
unmapped +=1
print ("Mapped read count:", mapped)
print("Unmapped read count:", unmapped)
```

HTSeq Results

HTSeq:

Example code:
htseq-count stranded=yes Mus_musculus_24_001Aligned.out.sam Mus_musculus.GRCm39.104.gtf > Mus_musculus_24_001_stranded_Aligned.out.sam

Proportion of mapped reads:

Example code:
grep -v "\t0$" Mus_musculus_15_001_stranded_Aligned.out.sam | awk '{if ($1 ~ "ENS.*") sum+=$2; else sum_two+=$2} END {print (sum/(sum_two+sum))}'

HtSeq Results

Mus_musculus_15_001_stranded_Aligned.out.sam
__no_feature 6612743
__ambiguous 6184
__too_low_aQual 13778
__not_aligned 195207
__alignment_not_unique 325109
Proportion of mapped reads = **0.0357995**
Mus_musculus_15_001_unstranded_Aligned.out.sam
__no_feature 559675
__ambiguous 341555
__too_low_aQual 13778
__not_aligned 195207
__alignment_not_unique 325109
Proportion of mapped reads = **0.806524**
Mus_musculus_24_001_stranded_Aligned.out.sam
__no_feature 9052611
__ambiguous 7141
__too_low_aQual 11069
__not_aligned 350360
__alignment_not_unique 481260
Proportion of mapped reads = **0.033492**
Mus_musculus_24_001_unstranded_Aligned.out.sam
__no_feature 829369
__ambiguous 446219
__too_low_aQual 11069
__not_aligned 350360
__alignment_not_unique 481260
Proportion of mapped reads = **0.79325**

Final Conclusion: I propose that these data are strand specific due to the above results. In libraries, 24_4A_control_S18_L008 and 15_3C_mbnl_S11_L008, the unstranded reads that mapped to genes were 80.6% and 79.3% respectively, likewise, only 3.6% and 3.3% of the stranded reads mapped to genes. Unstranded reads should theoretically map to 50% while stranded reads should map to either 100% or 0%. In this case, the stranded reads would likely be mapped at a high percentage to the reverse strand.