AQAA

Laura Paez

9/1/2021

Quality Assessment Assignment

Part 1 – Read quality score distributions

Using FastQC via the command line on Talapas, produce plots of quality score distributions for R1 and R2 reads. Also, produce plots of the per-base N content, and comment on whether or not they are consistent with the quality score plots.

![Per base Quality Score for File 1, Read 1

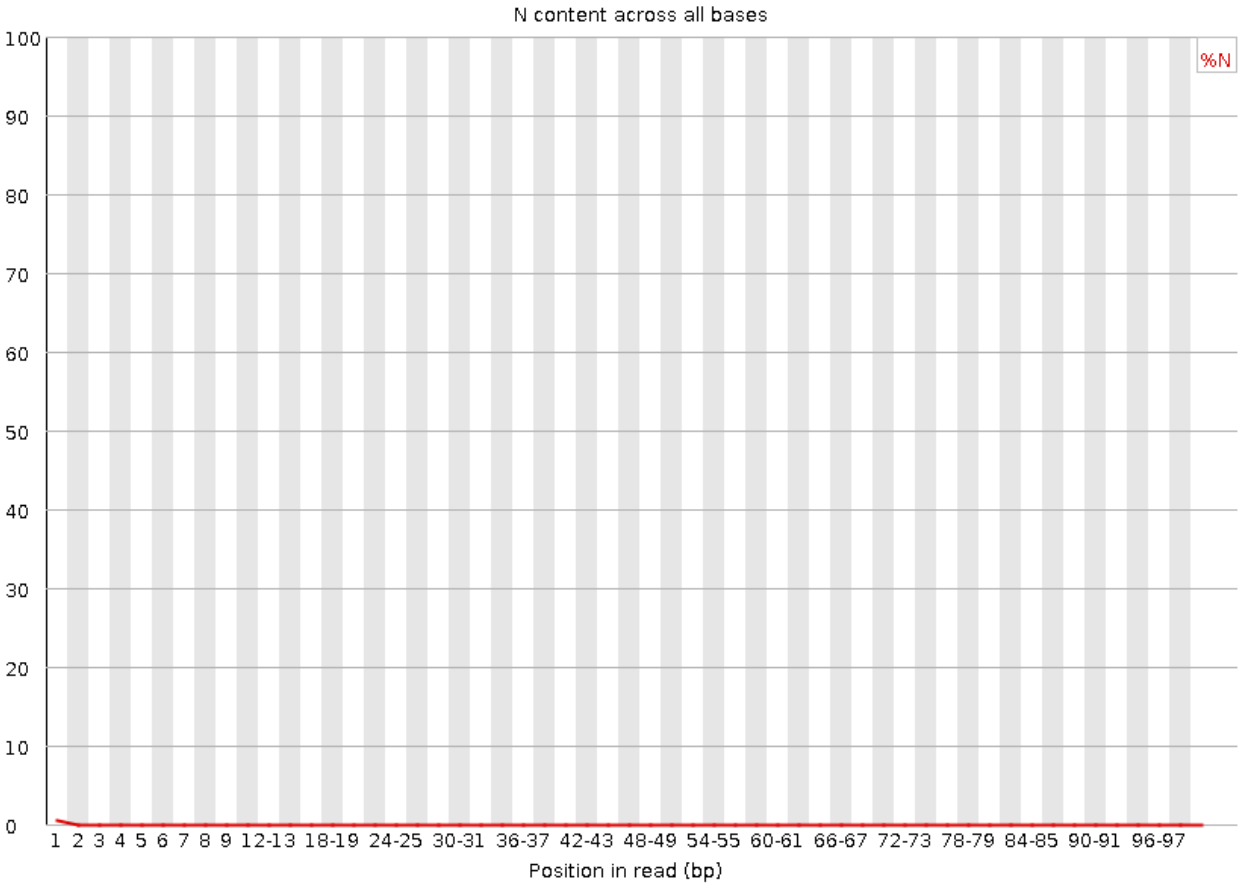
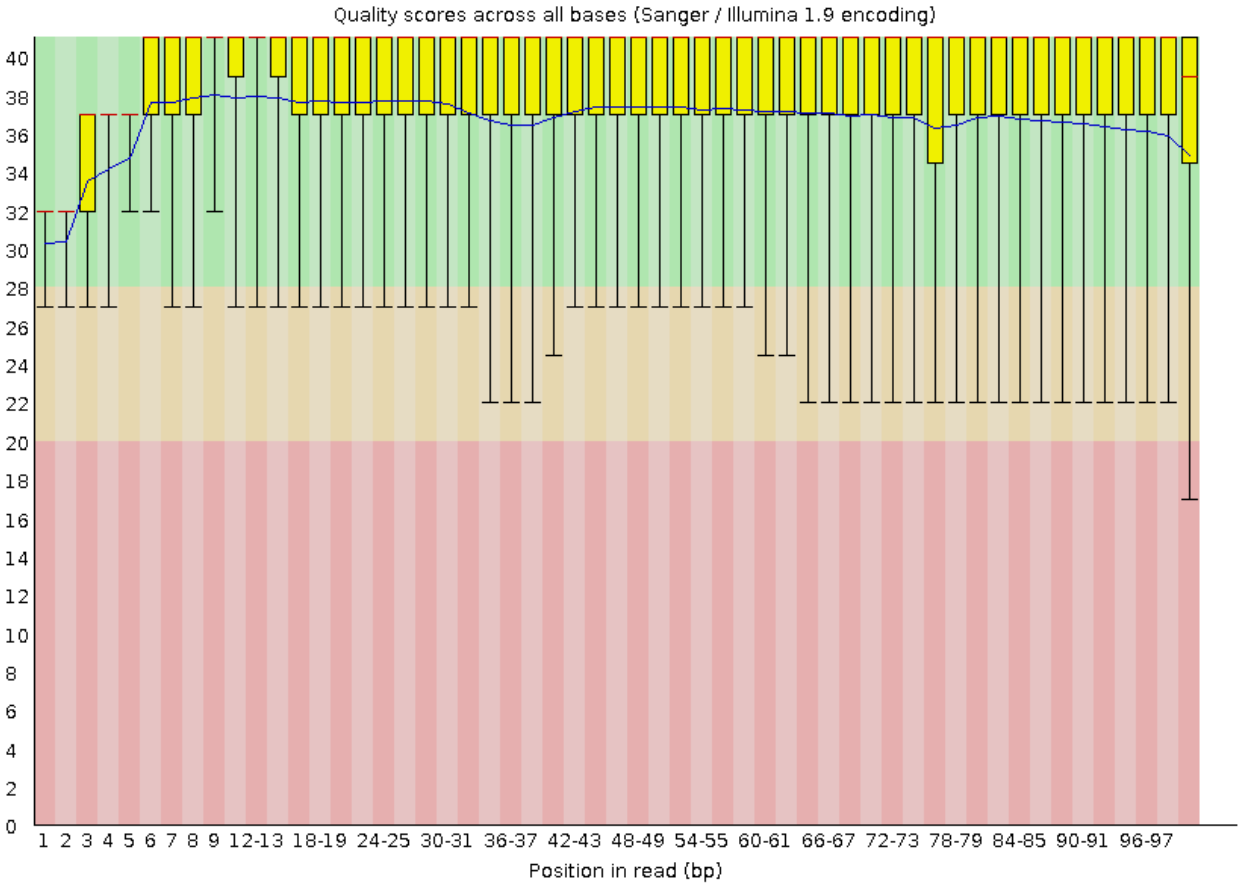
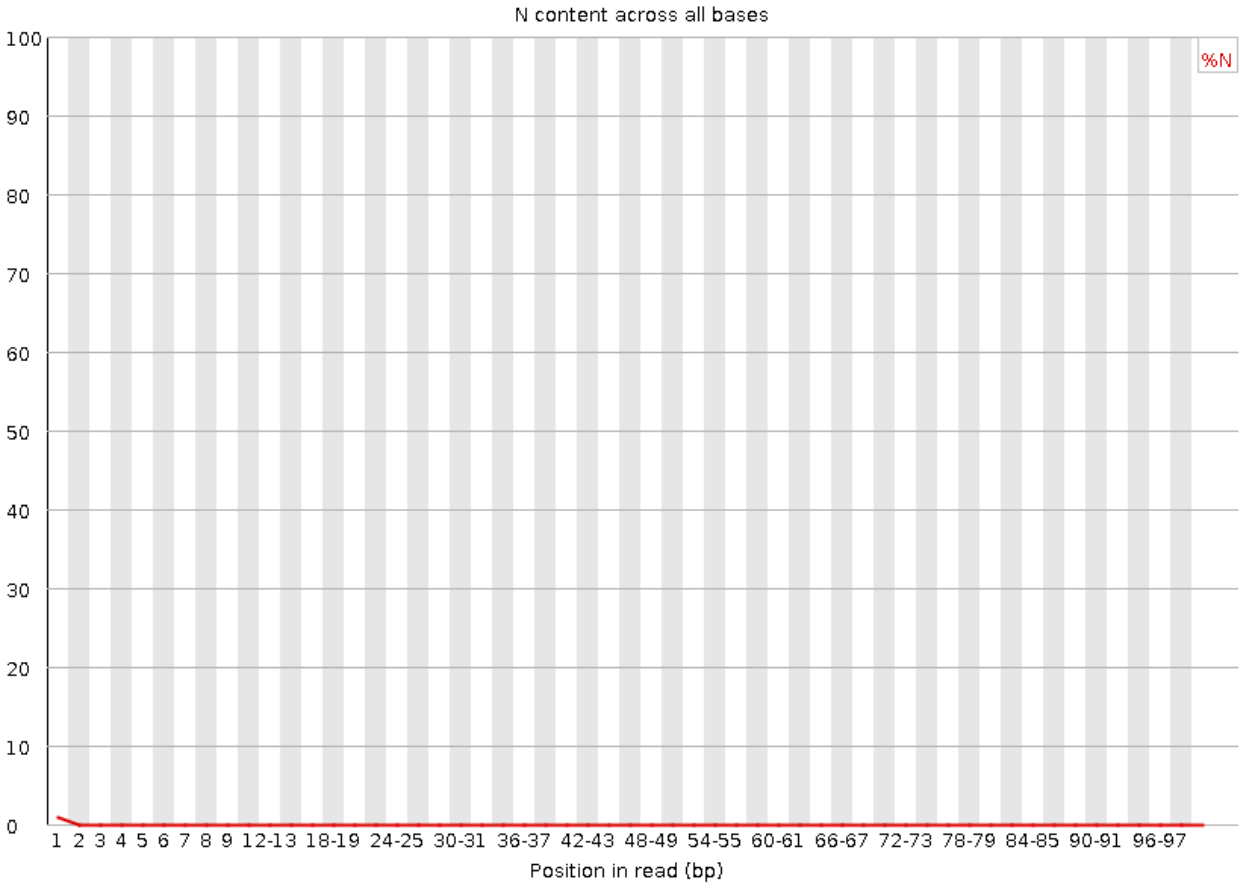


Figure 1: Per base N content for File 1, Read 1


Quality scores across all bases (Sanger / Illumina 1.9 encoding)
40
38
36
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
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)



The plots of quality score distributions are all consistent with the per-base N content plots because towards the beginning, there are lower quality scores paired with higher N counts, which is expected.

```
fastqc -t 2 -o run1 /projects/bgmp/shared/2017_sequencing/demultiplexed/24_4A_control_S18_L008_R1_001.fa

fastqc -t 2 -o run2 /projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R1_001.fas
```

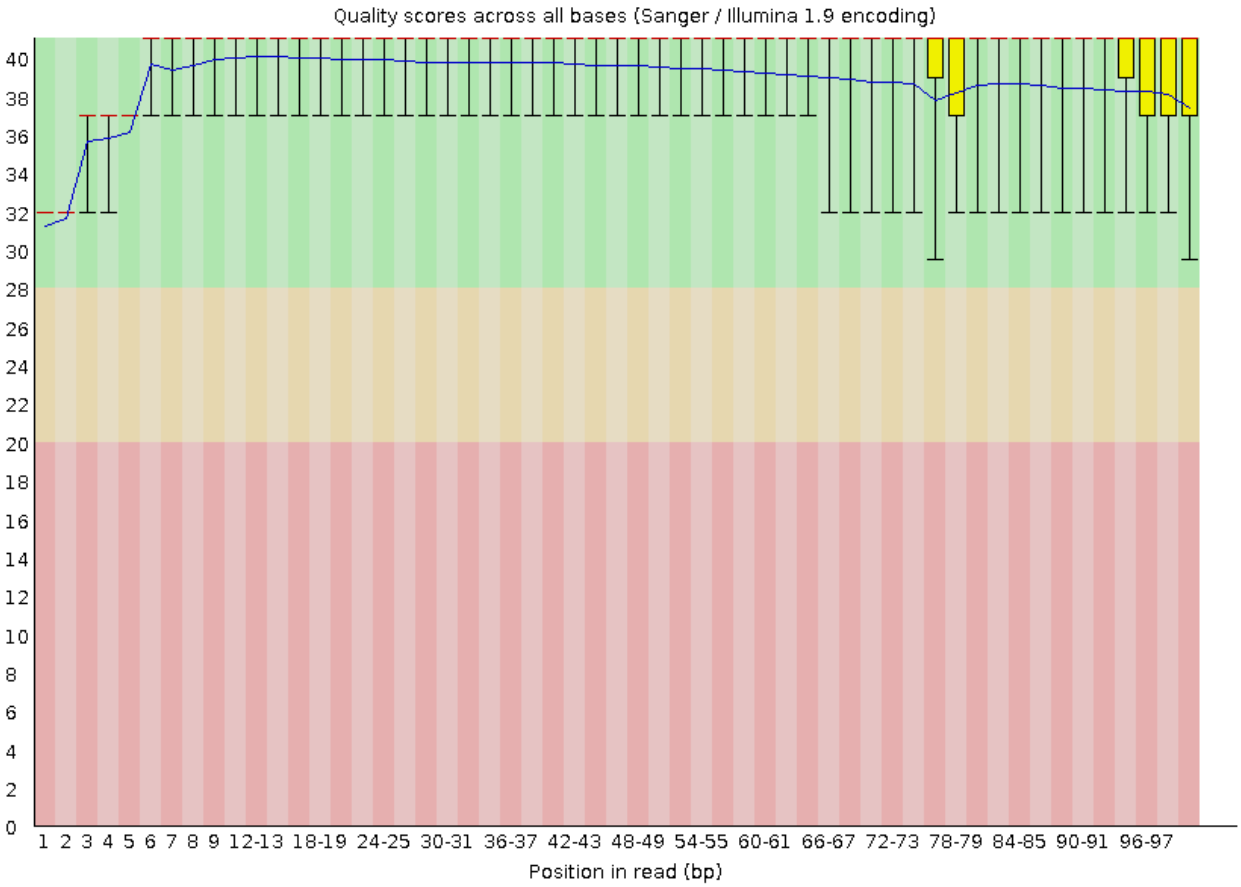


Figure 2: Per base Quality Score for File 24, Read 1

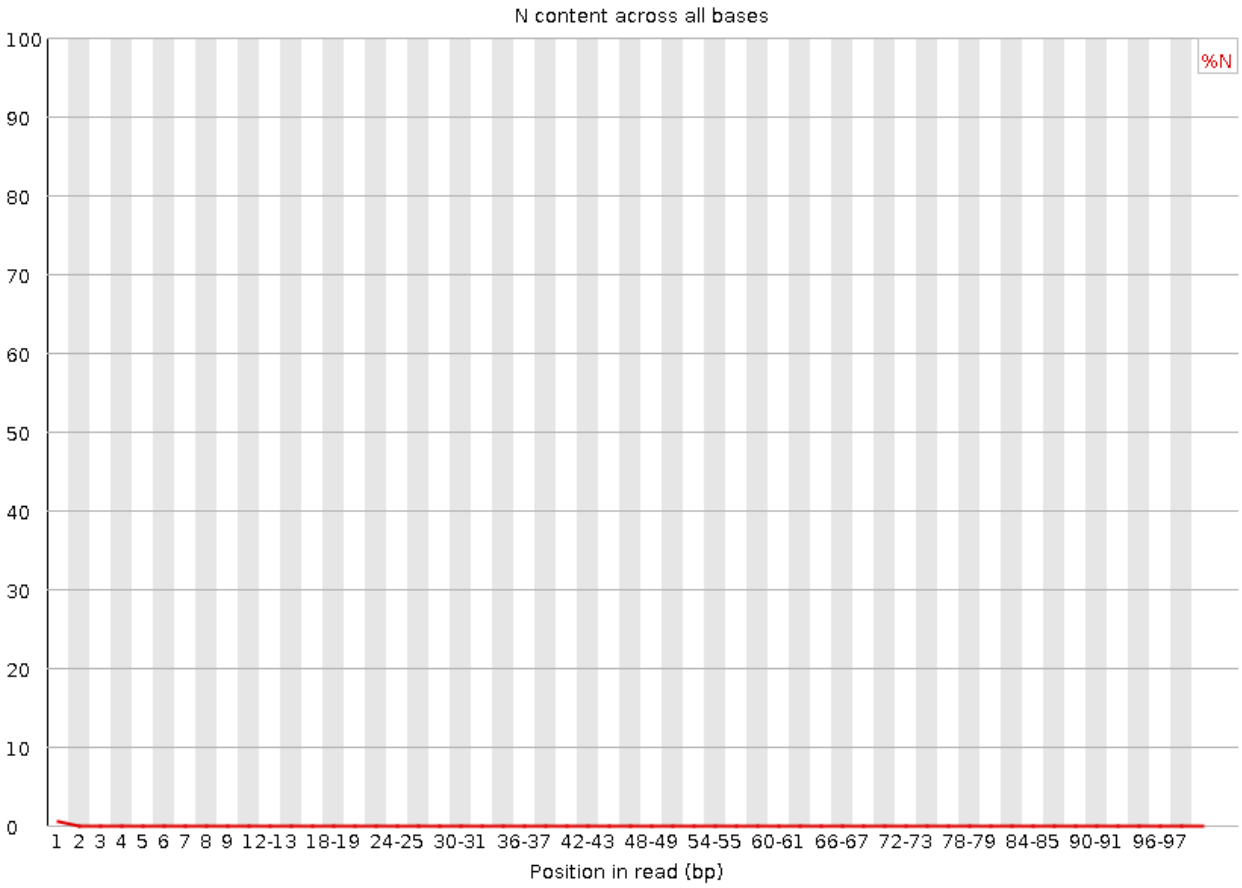


Figure 3: Per base N content for File 24, Read 1

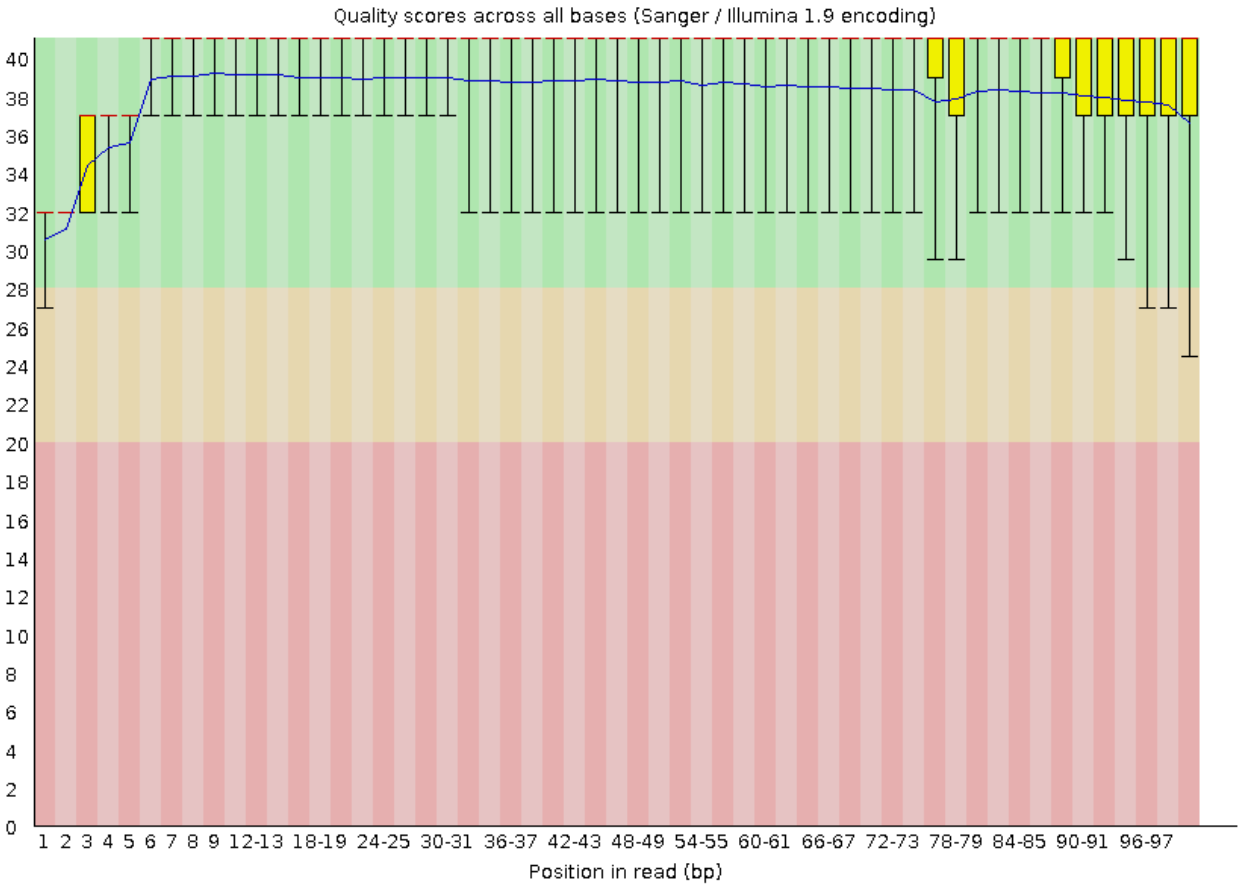


Figure 4: Per base Quality Score for File 24, Read 2

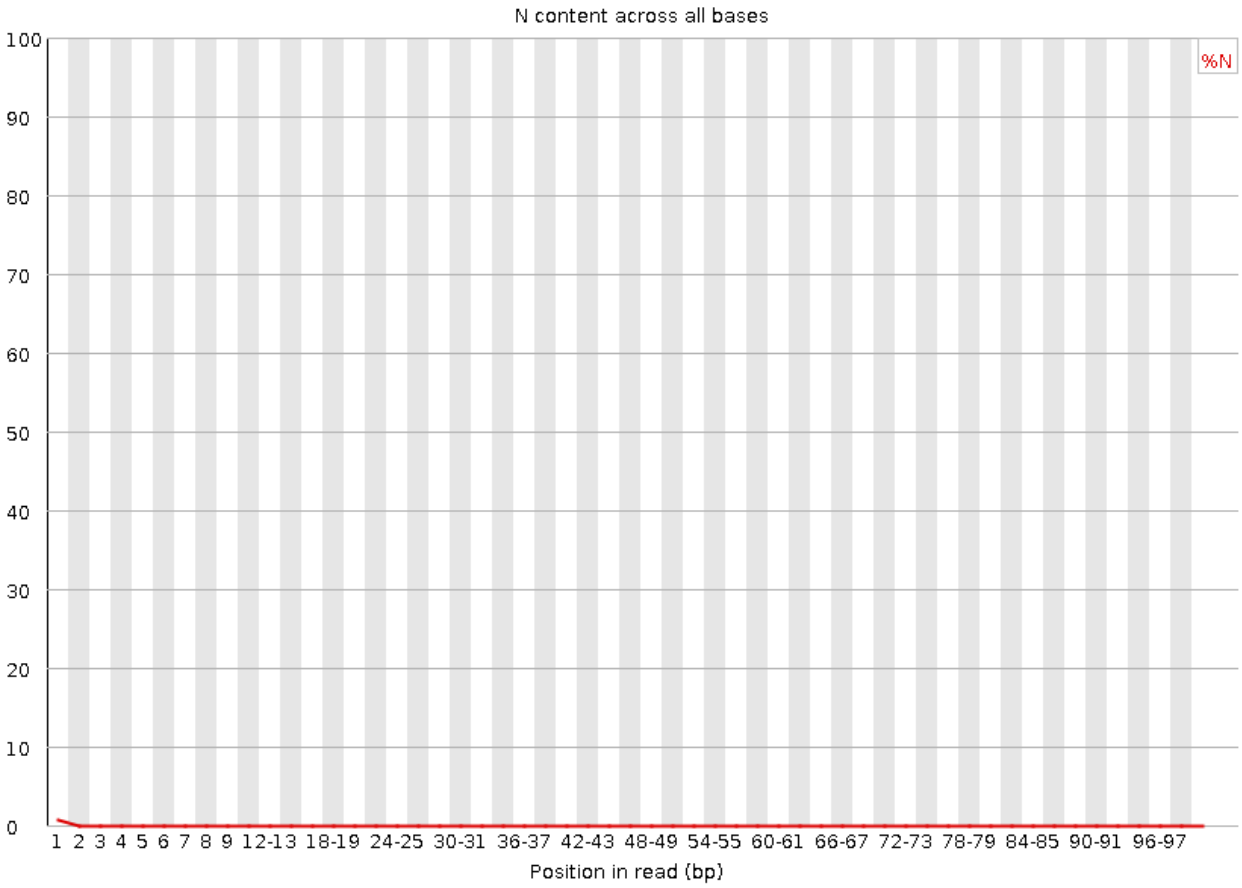


Figure 5: Per base N content for File 24, Read 2

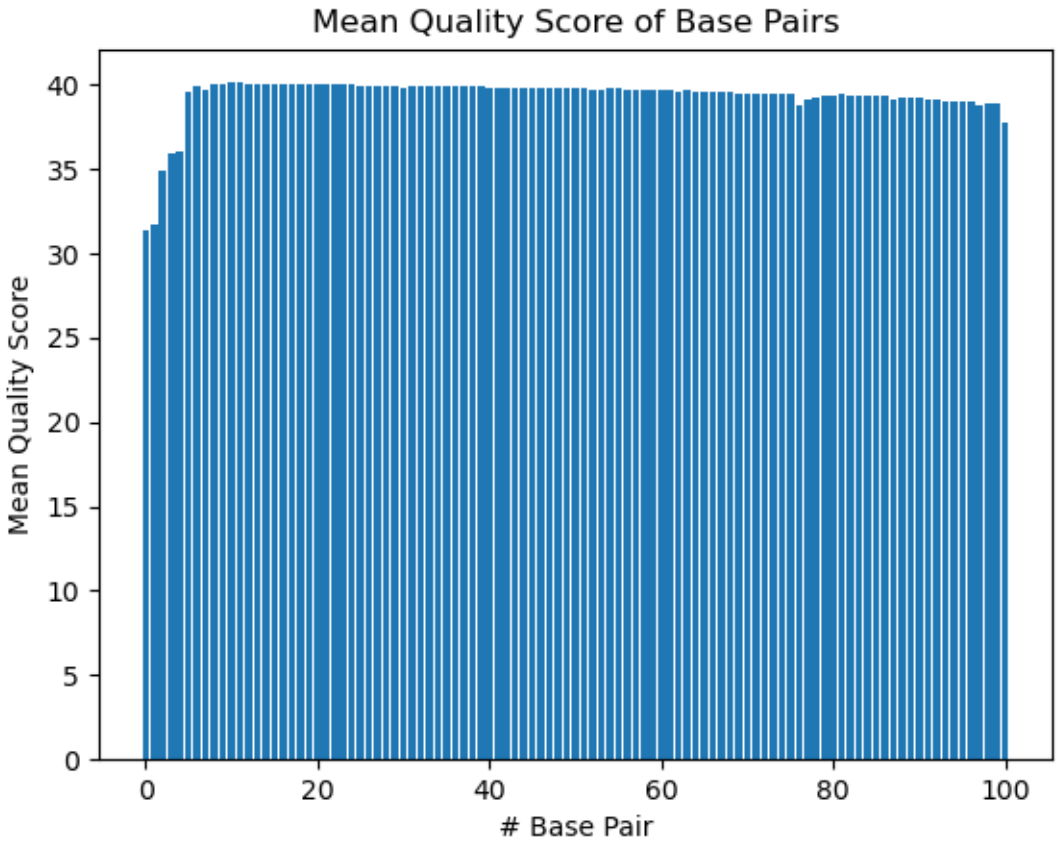


Figure 6: Per base N content for File 24, Read 2

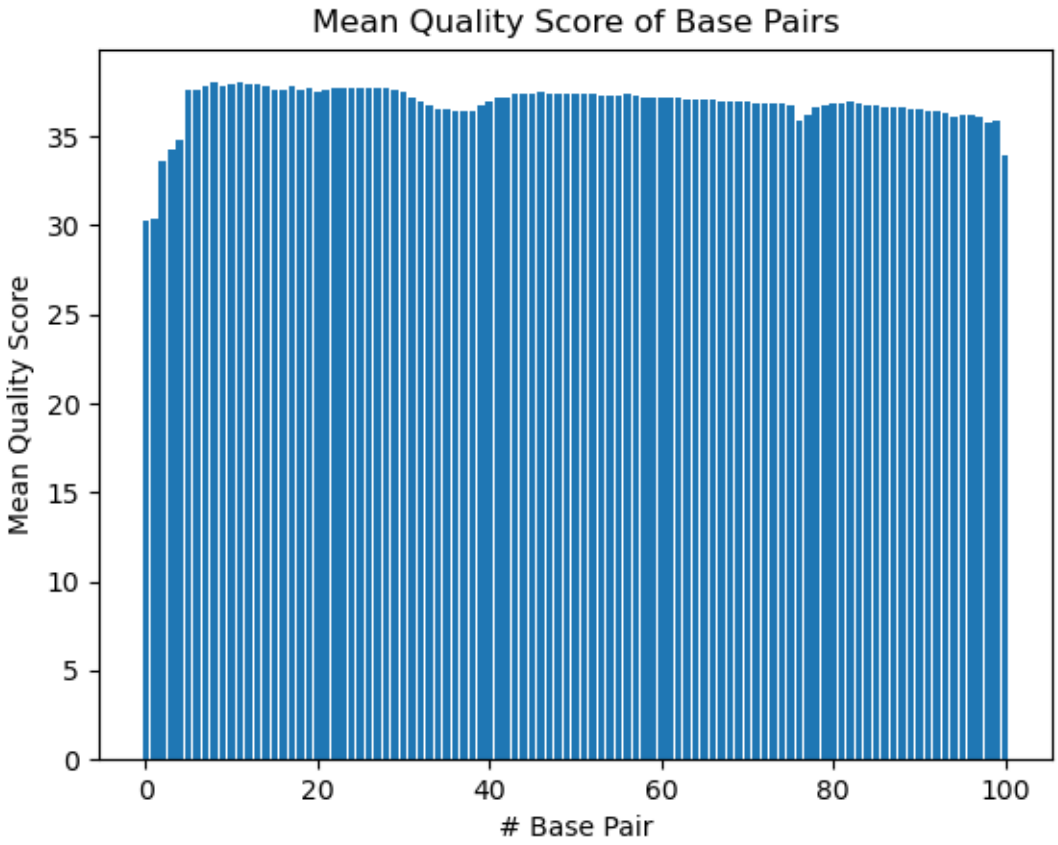


Figure 7: Per base N content for File 24, Read 2

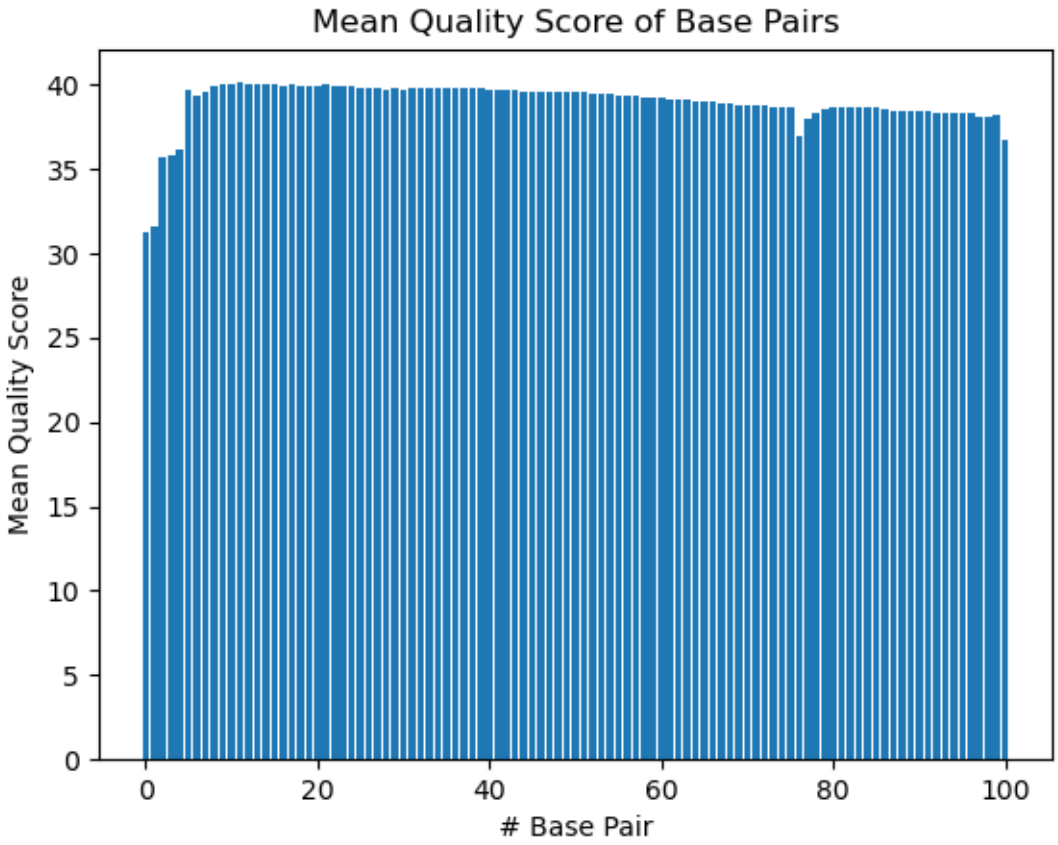
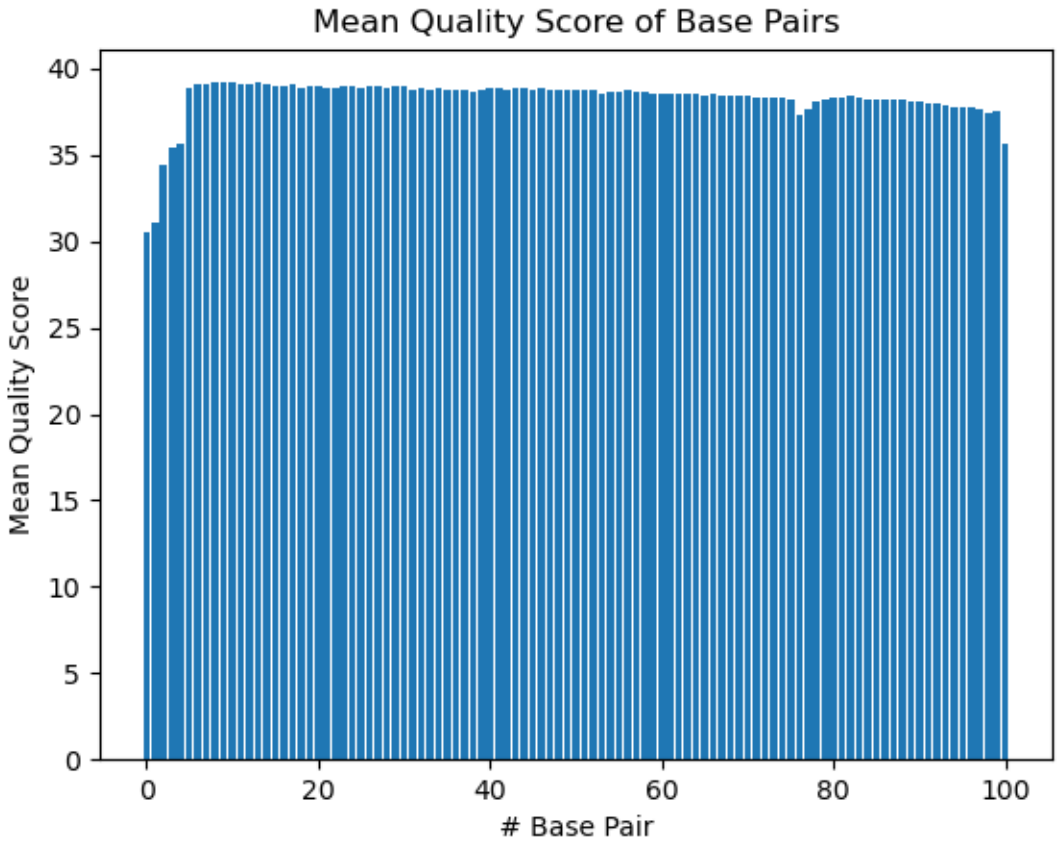


Figure 8: Per base N content for File 24, Read 2

Run your quality score plotting script from your Demultiplexing assignment from Bi622. 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?



While comparing my plots to the fastqc output plots, I found the same pattern of rises and dips across each pair I was comparing. Fastqc ran faster than the Demultiplexing assignment, not considering that fastqc has the option to run multiple files at once while the Demultiplexing assignment works on one file at a time. This is probably due to the level of programming and design that fastqc was written with, along with it being written in Java.

```
import argparse

def get_args():
    parser= argparse.ArgumentParser(description="Parses sam file")
    parser.add_argument("-f", help= "filename")
    parser.add_argument("-l", type=int, help="length")
    parser.add_argument("-o", help="output directory")
    return parser.parse_args()

args = get_args()

def init_list(lst: list, value: float=0.0) -> list:
    '''This function takes an empty list and will populate it with
    the value passed in "value". If no value is passed, initializes list
    with "l" values of 0.0.'''
```

```
    for i in range(args.l):
        lst.append(value)
    return lst
my_list: list = []
my_list = init_list(my_list)

import Bioinfo
import gzip

def populate_list(file):
    """Takes a fastq file and returns a list of sums of quality scores for each position in the DNA seq
    with gzip.open(file, 'rt') as fh:
        all_qual = init_list([])
        n = 0
        for line in fh:
    #         print(line)
            if n%4==3:
                stripped = line.strip()
    #             print(stripped)
                for i, qual in enumerate(stripped):
                    qscore = Bioinfo.convert_phred(qual)
                    all_qual[i]+=qscore
            n+=1
    return all_qual, n

my_list, num_lines = populate_list(args.f)

print("# Base Pair"+"\t"+"Mean Quality Score")
for i, score in enumerate(my_list):
    my_list[i] = score/(num_lines/4)
    print(str(i)+"\t"+str(my_list[i]))

import matplotlib.pyplot as plt

plt.bar(x=range(args.l), height=my_list)
plt.title('Mean Quality Score of Base Pairs')
plt.ylabel('Mean Quality Score')
plt.xlabel('# Base Pair')
plt.savefig(f'{args.o}.png')
```

Comment on the overall data quality of your two libraries.

Although there are some red marks on the fastqc files, upon closer inspection and taking into account the lecture notes, I found that my four plots were good quality. The red marks were usually on the section on per tile sequence quality, which never seemed to have an alarming amount of red entries on the graph.

Part 2 - Adaptor Trimming Comparison

Using cutadapt, properly trim adapter sequences from your assigned files. Be sure to read how to use cutadapt. Use default settings. What proportion of reads (both forward and reverse) were trimmed?

```
#!/usr/bin/bash
```

```
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=8
#SBATCH --time=10:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-o 1_r1_output.fastq -p 1_r2_output.fastq \
/projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R1_001.fastq.gz \
/projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R2_001.fastq.gz
```

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.

gunzip -c /projects/bgmp/shared/2017_sequencing/demultiplexed/24_4A_control_S18_L008_R1_001.fastq.gz | grep -c 'AGATCGGAAGAGCACACGTCTGAACTCCAGTCA' 8025

gunzip -c /projects/bgmp/shared/2017_sequencing/demultiplexed/24_4A_control_S18_L008_R1_001.fastq.gz | grep -c 'AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT' 0

gunzip -c /projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R1_001.fastq.gz | grep -c 'AGATCGGAAGAGCACACGTCTGAACTCCAGTCA' 31837

gunzip -c /projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R1_001.fastq.gz | grep -c 'AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT' 0

Use Trimmomatic to quality trim your reads.

```
#!/usr/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=8
#SBATCH --time=10:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v trimmomatic PE -phred33 -trimlog logoutput1.txt -summary sumoutput1.txt \
/projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R1_001.fastq.gz \
/projects/bgmp/shared/2017_sequencing/demultiplexed/1_2A_control_S1_L008_R2_001.fastq.gz \
-baseout trimoutput_1.fq.gz \
LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35
```

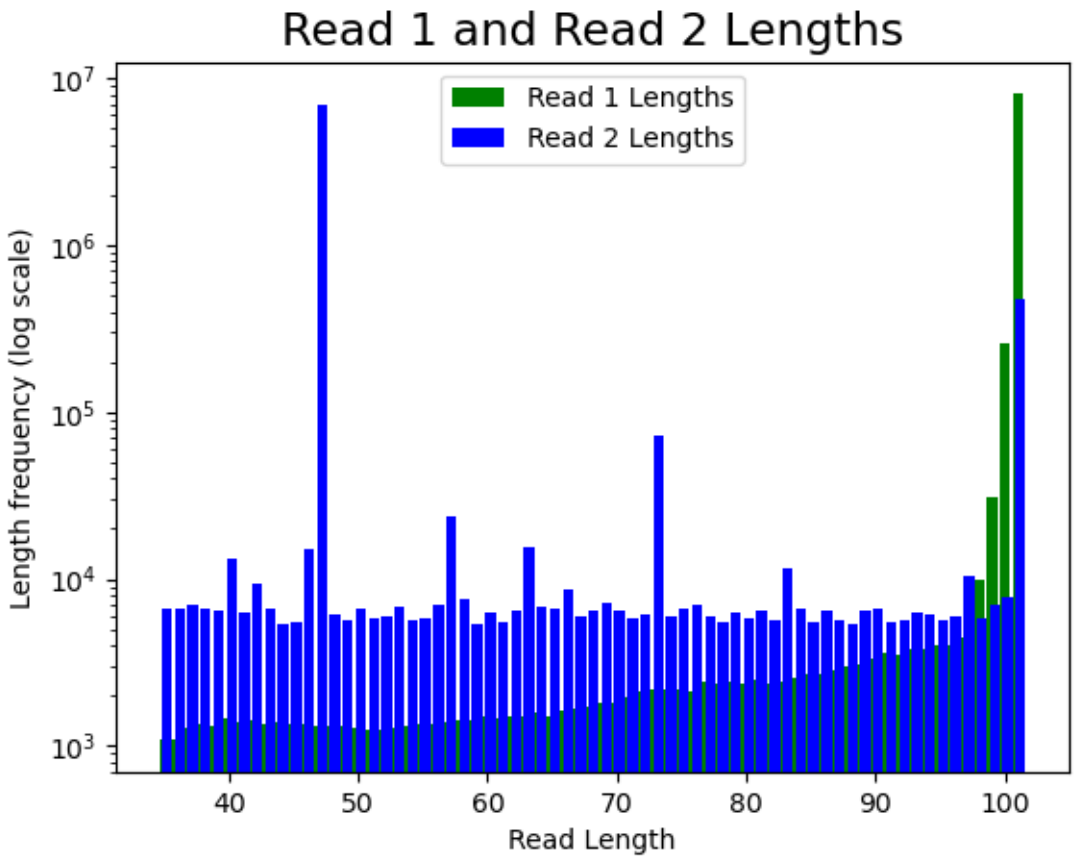
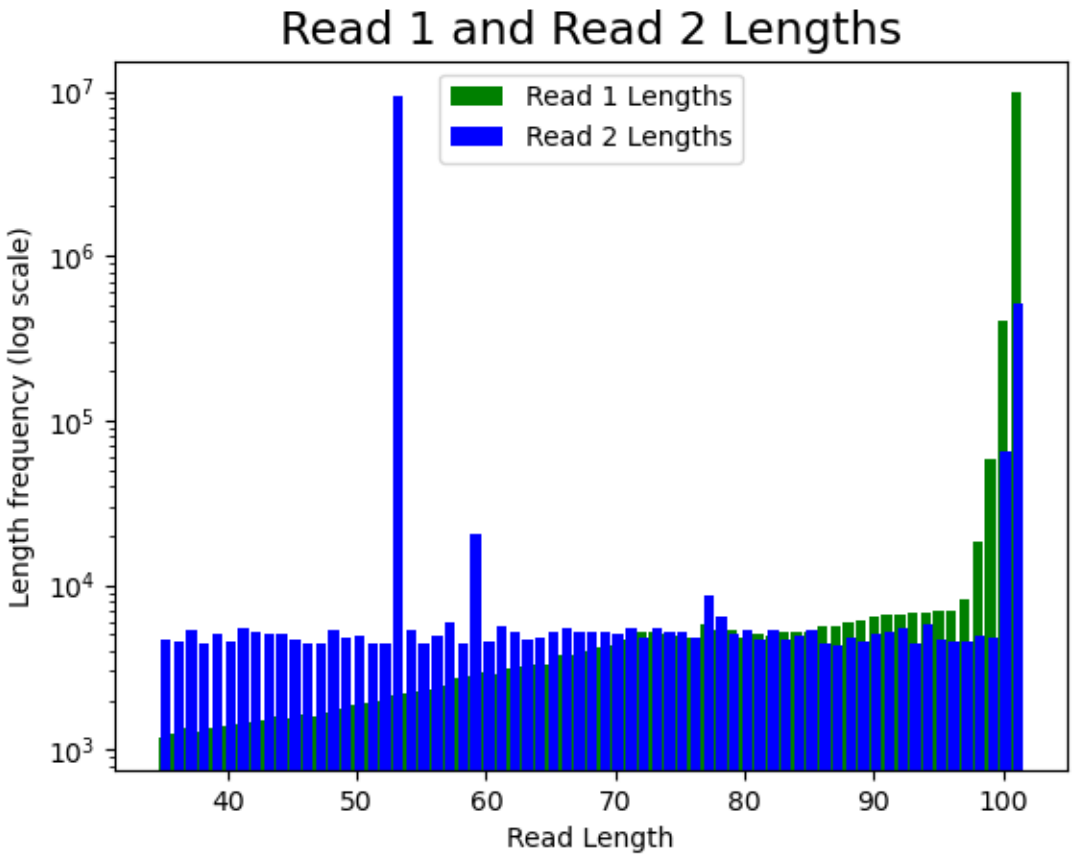


Figure 9: Read 1 and Read 2 Plot for File 1

Plot the trimmed read length distributions for both R1 and R2 reads (on the same plot). You can produce 2 different plots for your 2 different RNA-seq samples. There are a number of ways you could possibly do this. One useful thing your plot should show, for example, is whether R1s are trimmed more extensively than R2s, or vice versa. Comment on whether you expect R1s and R2s to be adapter-trimmed at different rates.



I expected R1 to have less adapter-trimming because it was more recently sequenced compared to R2, which was on the Illumina machine longer, leading to more degradation.

Part 3

Find publicly available mouse genome fasta files (Ensemble release 104) and generate an alignment database from them. Align the reads to your mouse genomic database using a splice-aware aligner. Use the settings specified in PS8 from Bi621.

```
#!/usr/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=8
#SBATCH --time=02:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v STAR --runThreadN 8 \
```

```
--runMode genomeGenerate \
--genomeDir  stardb \
--genomeFastaFiles /projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/mousedna/Mus_musculus.GRCm39.dna.prima
--sjdbGTFfile /projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/mousedna/Mus_musculus.GRCm39.104.gtf


#!/usr/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=16
#SBATCH --time=02:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v STAR --runThreadN 16 \
--runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 \
--alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn /projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/trimoutput_1_1P.fq.gz /projects/bgmp/lpaez/
--genomeDir /projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/stardb \
--outFileNamePrefix aligned_1_
```

Using your script from PS8 in Bi621, report the number of mapped and unmapped reads from each of your 2 sam files. Make sure that your script is looking at the bitwise flag to determine if reads are primary or secondary mapping (update your script if necessary).

```
#!/usr/bin/env python
import argparse

def get_args():
    parser= argparse.ArgumentParser(description="Parses sam file")
    parser.add_argument("-f", help= "filename")
    return parser.parse_args()

args = get_args()

with open(args.f) as fh:

    mapped = 0
    unmapped = 0

    for line in fh:
        line = line.strip()
        #getting rid of the header lines in sam file.
        if line.startswith('@'):
            continue
        #capturing bitwise flag in variable
        flag = line.split("\t")[1]

        #if secondary alignment condition is met, continue to next loop
```

```
        if((int(flag) & 256) == 256):
            continue
        else:
            if ((int(flag) & 4) != 4):
                mapped+=1
            else:
                unmapped+=1

print(mapped, unmapped)
```

Count reads that map to features using htseq-count. You should run htseq-count twice: once with –stranded=yes and again with –stranded=no. Use default parameters otherwise.

```
#!/usr/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=8
#SBATCH --time=10:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v htseq-count --stranded=yes -r name \
sort_aligned_1_Aligned.out.sam \
/projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/mousedna/Mus_musculus.GRCm39.104.gtf \
> stranded1.txt
```

```
#!/usr/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --cpus-per-task=8
#SBATCH --time=10:00:00
#SBATCH --mail-user='lpaez@uoregon.edu'
#SBATCH --mail-type=END,FAIL

/usr/bin/time -v htseq-count --stranded=no -r name \
sort_aligned_1_Aligned.out.sam \
/projects/bgmp/lpaez/bioinformatics/Bi623/AQAA/mousedna/Mus_musculus.GRCm39.104.gtf \
> unstranded1.txt
```

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 (e.g. "I propose that these data are/are not strand-specific, because X% of the reads are y, as opposed to z.").

```
Results format:
mapped
total
percent mapped

cat stranded1.txt | awk '{if ($0 ~ /'ENSMU'/) {sum1+=$2} else if ($2 > 0) {sum2+=$2}} END {print sum1;p
291114
7967362
0.0365383
```

```
cat stranded24.txt | awk '{if ($0 ~ /'ENSMU'/) {sum1+=$2} else if ($2 > 0) {sum2+=$2}} END {print sum1;p
340416
10246153
0.0332238

cat unstranded24.txt | awk '{if ($0 ~ /'ENSMU'/) {sum1+=$2} else if ($2 > 0) {sum2+=$2}} END {print sum
8094151
10246153
0.78997

cat unstranded1.txt | awk '{if ($0 ~ /'ENSMU'/) {sum1+=$2} else if ($2 > 0) {sum2+=$2}} END {print sum1
6558459
7967362
0.823166
```

I propose that these data are not strand-specific due to the significantly greater amount of percent of reads mapped in the unstranded results. Over 70% of reads were mapped in the unstranded files as opposed to the stranded files.