QAA_Report

Nora Kearns

9/1/2021

PART 1 – READ QUALITY SCORE DISTRIBUTIONS

QUESTION 1:
The quality score and per-base N distributions are consistent. The per base N-distributions indicate that the content across all bases is very low (effectively 0). This is reflected in the quality score distributions as well, as the mean quality scores for the S1 and S13 reads are between 36 and 38. There is a slight upward curve at the beginning of each N-content graph where the N-content is above 0. This is because the N content at the beginning of a read is generally higher.

The quality scores distributions for read 2 for both S1 and S13 are slightly lower and have wider distributions, which is expected given that by the second read of a sequencing run the DNA has undergone repeated chemical modifications and may be damaged.

QUESTION 2:
The distributions generated by FASTQC are very similar to the distributions generated by my own code. However, the time it took for FASTQC to generate the plot is substantially lower than the time my own program took. This is because FASTQC can be multithreaded, so that sets of instructions can be run independently and the work can be divided, which makes the program run much faster. My own program was not written to be multithreaded, so while FASTQC took 10-16 seconds to run, my own took over 6 minutes per distribution graph.

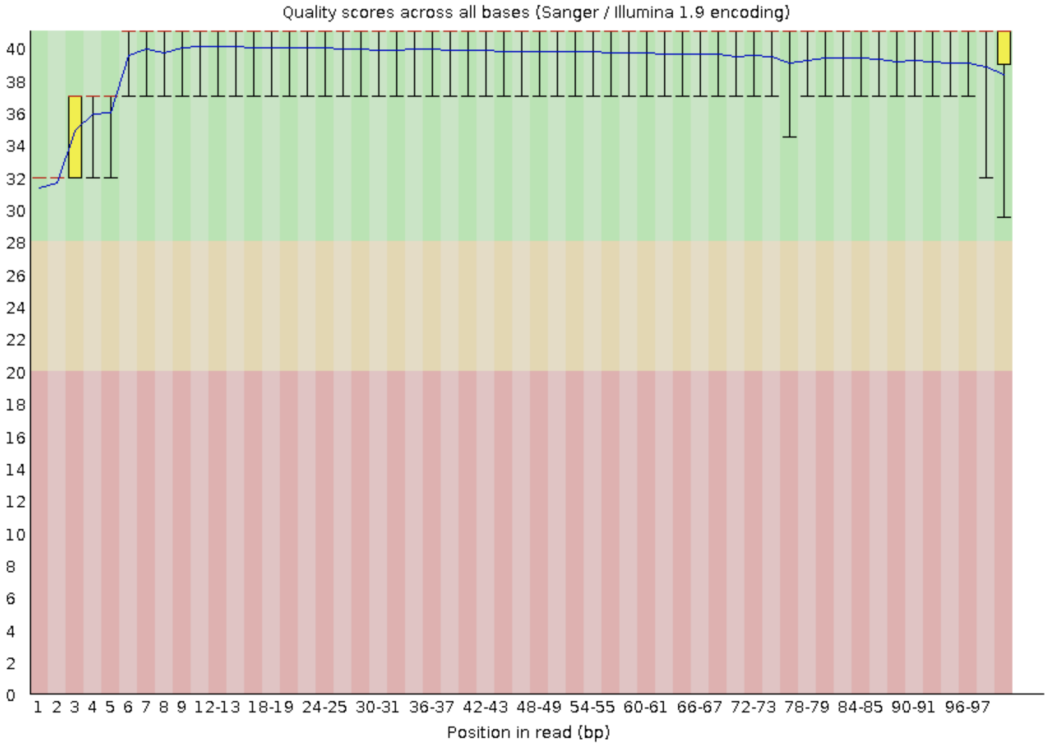
QUESTION 3:
The data quality of both of my libraries seems quite high, with the mean quality score distributions between 36 and 38. The quality score of read 2 is more variable, however, that is expected given that read 2 has undergone more chemical modifications.

Slurm script to run FASTQC

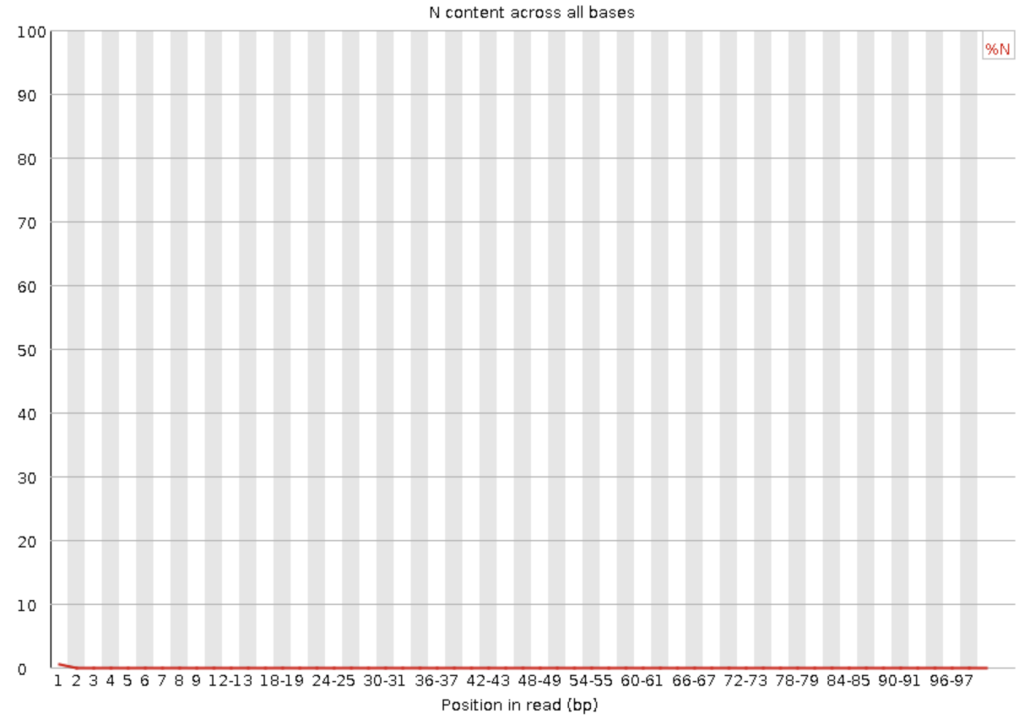
```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=QAA_P1
#SBATCH --output=QAA_P1_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --time=1:00:00
#SBATCH --cpus-per-task=1

module load fastqc/0.11.5
/usr/bin/time -v zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
1_2A_control_S1_L008_R1_001.fastq.gz | fastqc stdin -o S1_R1

/usr/bin/time -v zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
1_2A_control_S1_L008_R2_001.fastq.gz | fastqc stdin -o S1_R2

/usr/bin/time -v zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
17_3E_fox_S13_L008_R1_001.fastq.gz | fastqc stdin -o S13_R1

/usr/bin/time -v zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
17_3E_fox_S13_L008_R2_001.fastq.gz | fastqc stdin -o S13_R2
```

FASTQC: S1_Read1 Quality Score Distribution

User time (seconds): 11.52

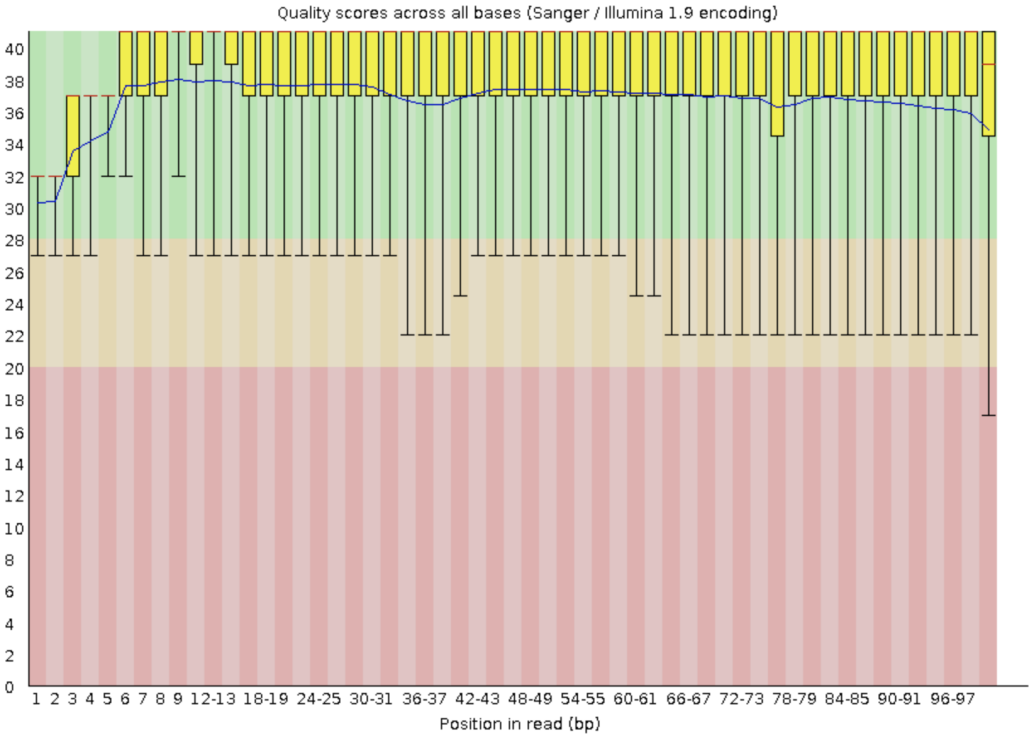


FASTQC: S1_Read1 Per-base N content

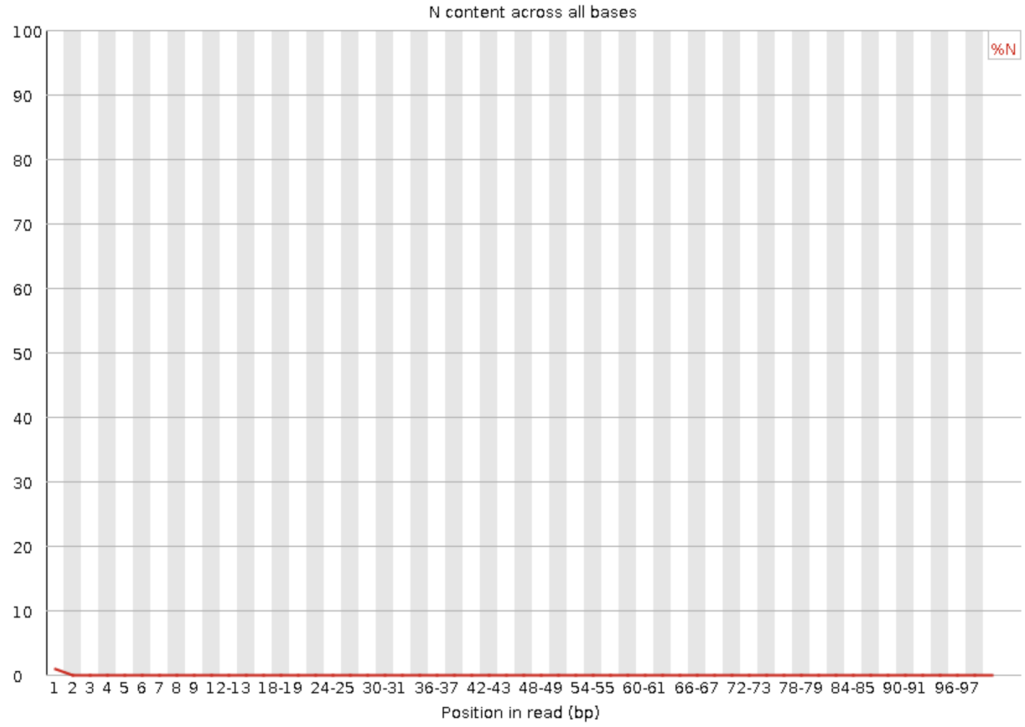


FASTQC: S1_Read2 Quality Score Distribution

User time (seconds): 12.19

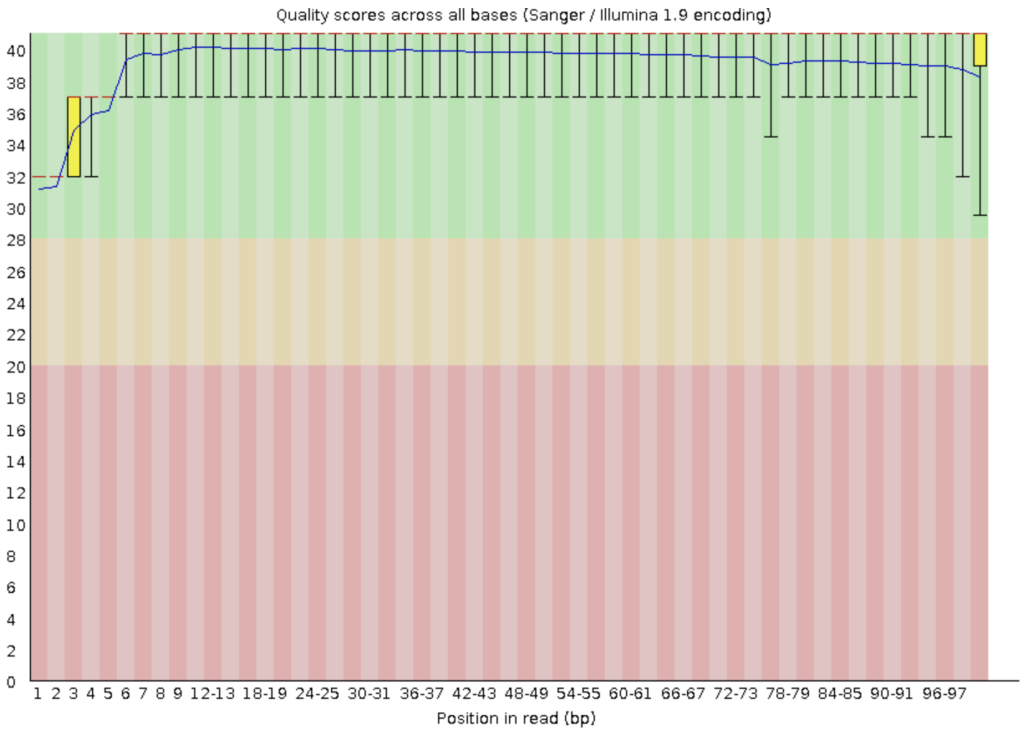


FASTQC: S1_Read2 Per-base N content

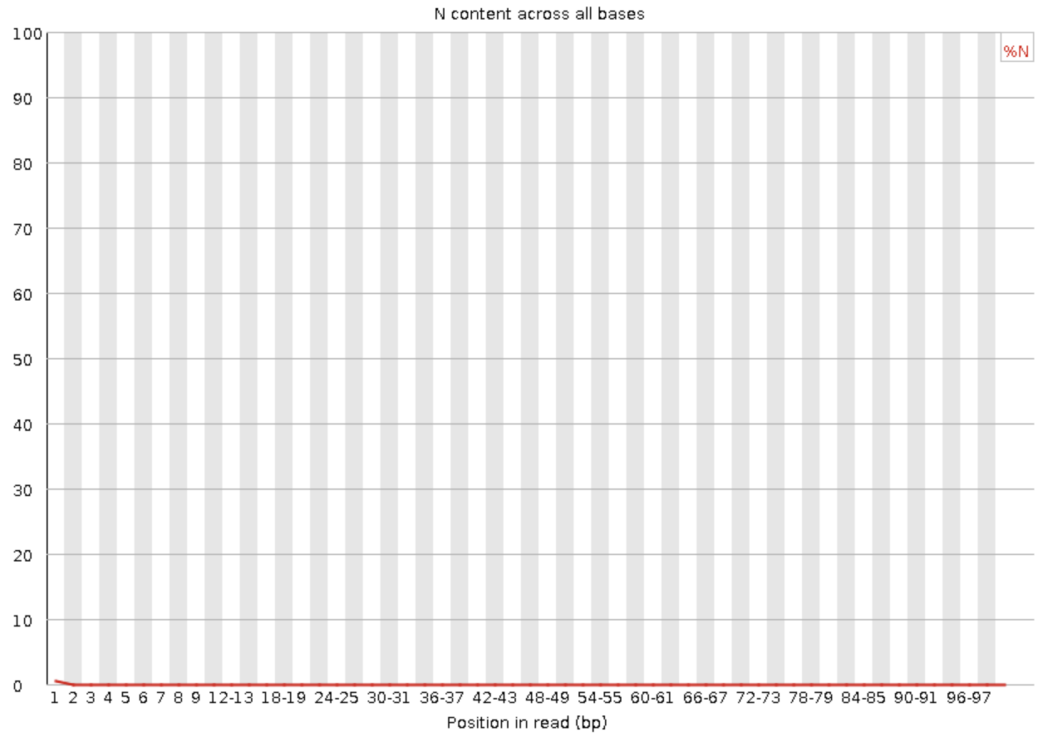


FASTQC: S13_Read1 Quality Score Distribution

User time (seconds): 16.21

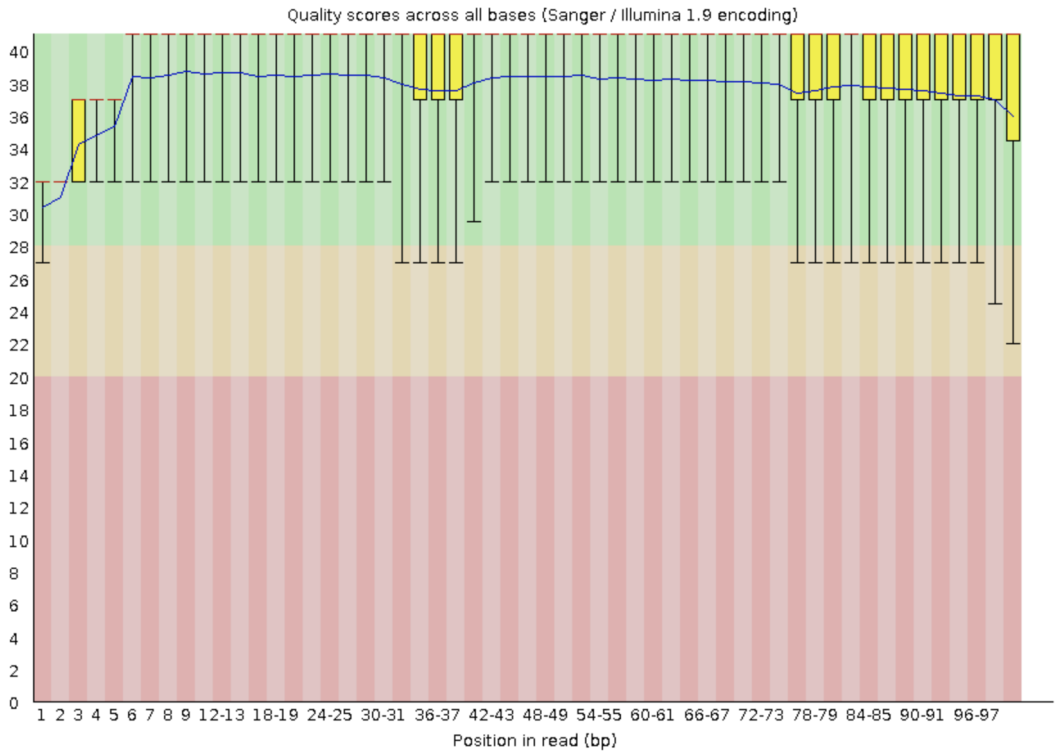


FASTQC: S13_Read1 Per-base N content

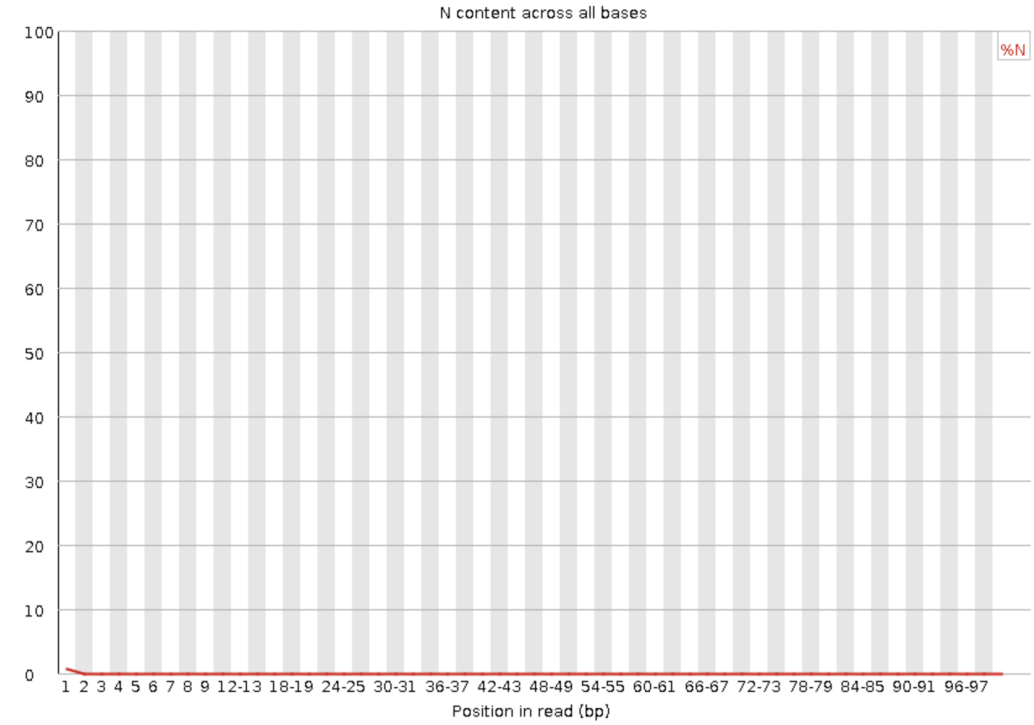


FASTQC: S13_Read2 Quality Score Distribution

User time (seconds): 15.68



FASTQC: S13_Read2 Per-base N content



Demultiplexing Script: Quality Score Distributions

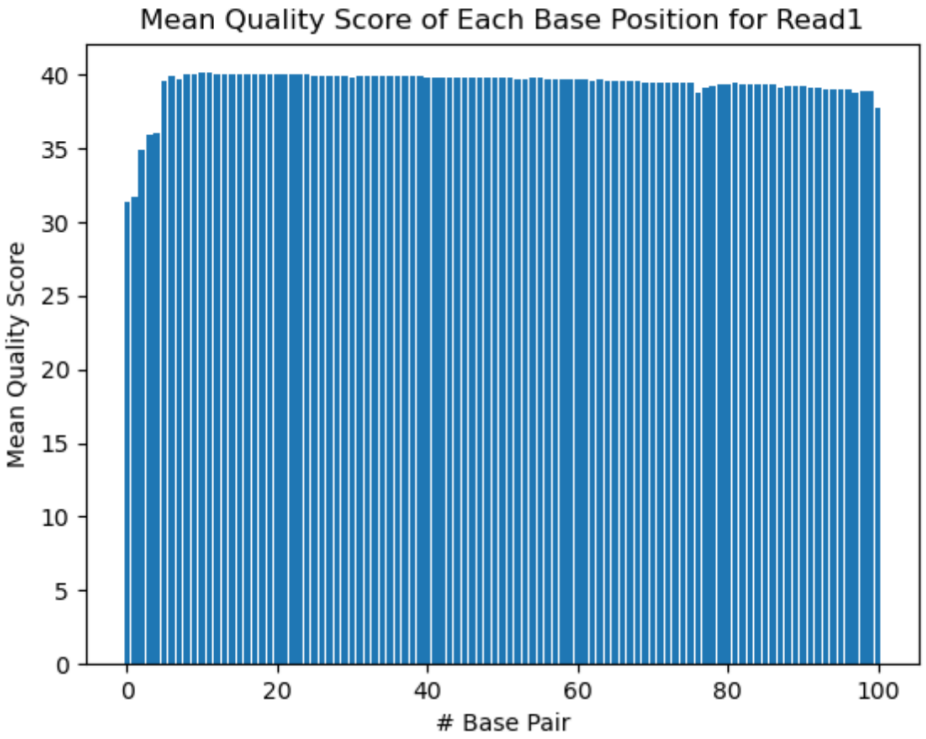
Slurm script to run Quality Score Distributions Script from Demux

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=P1_Dist
#SBATCH --output=P1_Dist_%j.out
```

```
#SBATCH --time=8:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=1

/usr/bin/time -v ./P1_Dist.py -f /projects/bgmp/shared/2017_sequencing/demultiplexed/
1_2A_control_S1_L008_R1_001.fastq.gz -p 101 -o S1_R1_hist.png -r Read1

/usr/bin/time -v ./P1_Dist.py -f /projects/bgmp/shared/2017_sequencing/demultiplexed/
1_2A_control_S1_L008_R2_001.fastq.gz -p 101 -o S1_R2_hist.png -r Read1

/usr/bin/time -v ./P1_Dist.py -f /projects/bgmp/shared/2017_sequencing/demultiplexed/
17_3E_fox_S13_L008_R1_001.fastq.gz -p 101 -o S13_R1_hist.png -r Read1

/usr/bin/time -v ./P1_Dist.py -f /projects/bgmp/shared/2017_sequencing/demultiplexed/
17_3E_fox_S13_L008_R2_001.fastq.gz -p 101 -o S13_R2_hist.png -r Read2
```

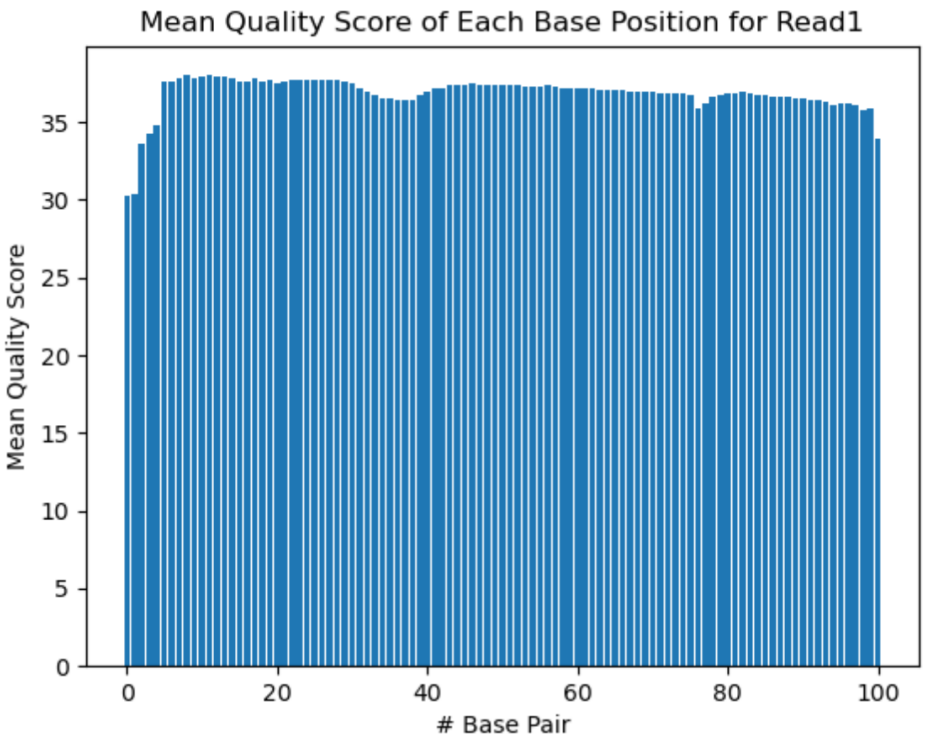
Quality Score Distribution for S1_Read

User time (seconds): 382.03



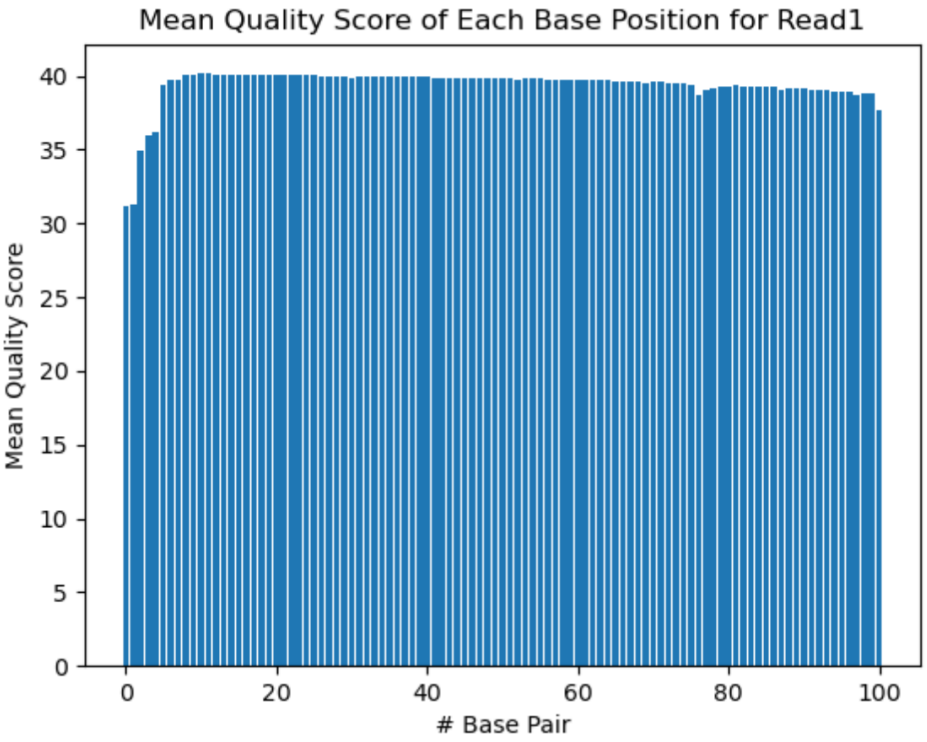
Quality Score Distribution for S1_Read2

User time (seconds): 392.06



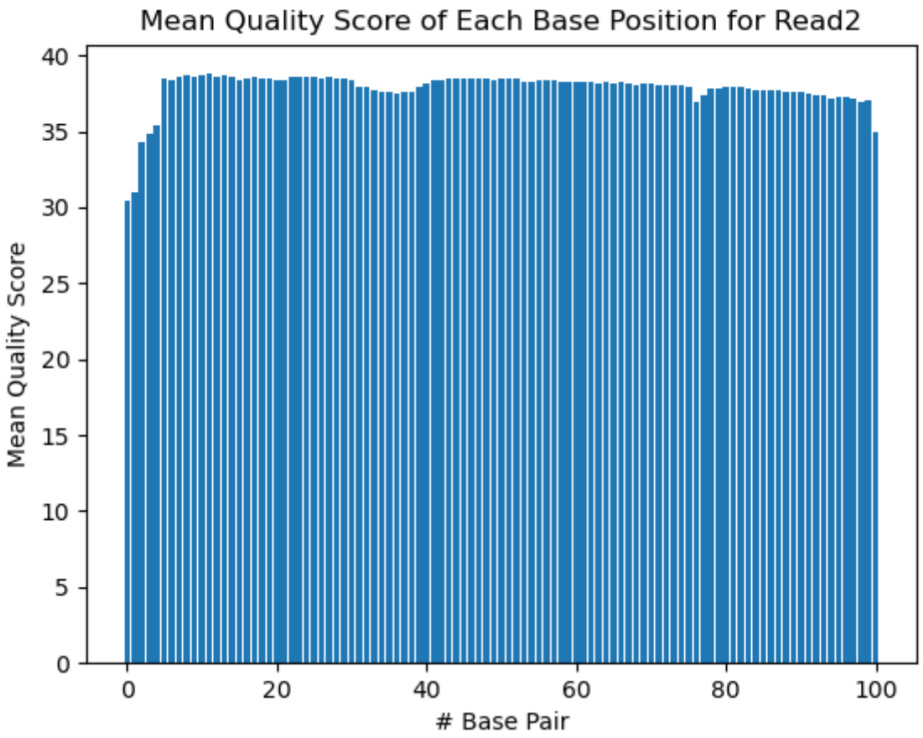
Quality Score Distribution for S13_Read1

User time (seconds): 548.93



Quality Score Distribution for S13_Read2

User time (seconds): 542.89



PART 2: ADAPTER TRIMMING COMPARISON

Slurm Script to run CUTADAPT

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=cutadapt
#SBATCH --output=cutadapt_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --time=8:00:00
#SBATCH --cpus-per-task=1

/usr/bin/time -v cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-j 0 -o S1_R1.fastq -p S1_R2.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
/usr/bin/time -v cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-j 0 -o S13_R1.fastq -p S13_R2.fastq /projects/bgmp/shared/2017_sequencing/demultiplexed/
17_3E_fox_S13_L008_R1_001.fastq.gz /projects/bgmp/shared/2017_sequencing/
```

Slurm Script to run TRIMMOMATIC

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=trimmomatic
#SBATCH --output=trimmomatic_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --time=1:00:00
#SBATCH --cpus-per-task=1
```

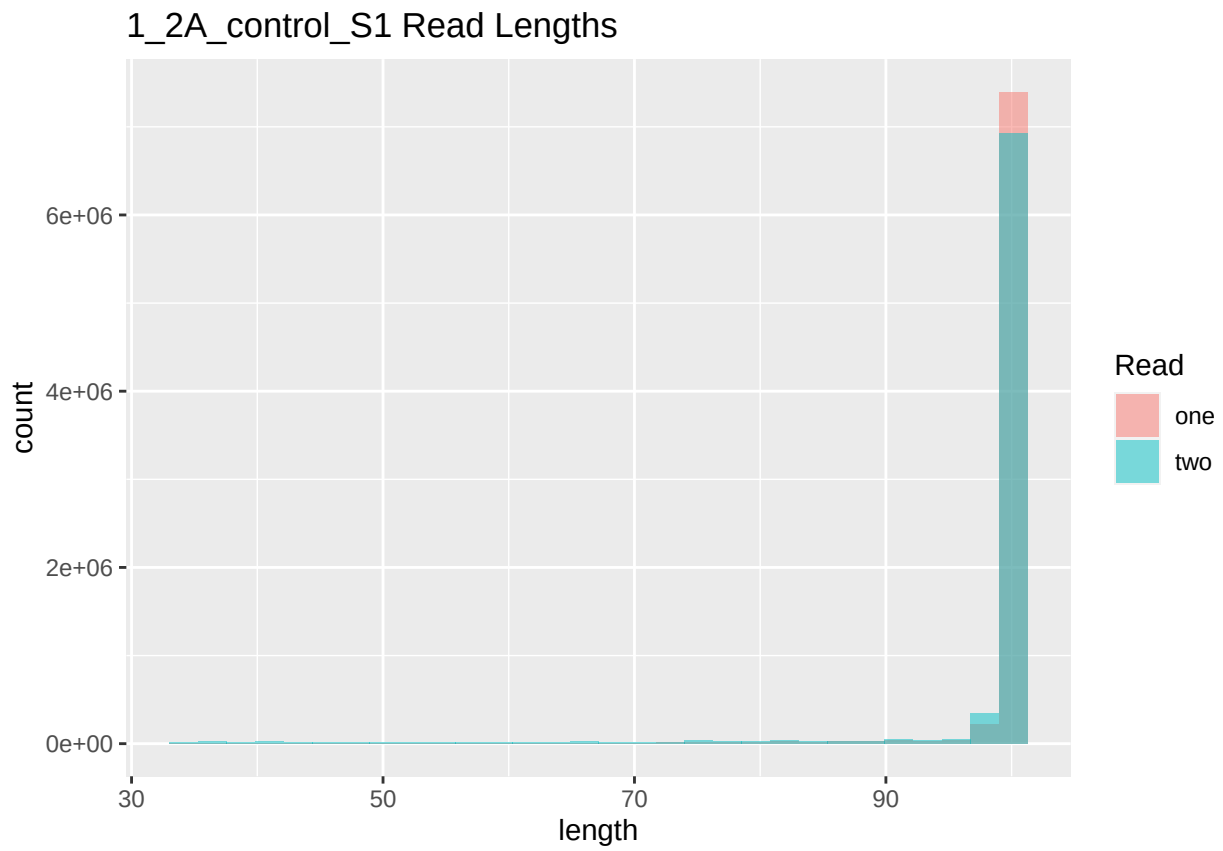
```
/usr/bin/time -v trimmomatic PE -threads 4 S1_R1.fastq S1_R2.fastq S1_R1.trimmed.fastq \
S1_R1un.trimmed.fastq S1_R2.trimmed.fastq S1_R2un.trimmed.fastq LEADING:3 TRAILING:3 \
SLIDINGWINDOW:5:15 MINLEN:35

/usr/bin/time -v trimmomatic PE -threads 4 S13_R1.fastq S13_R2.fastq S13_R1.trimmed.fastq \
S13_R1un.trimmed.fastq S13_R2.trimmed.fastq S13_R2un.trimmed.fastq LEADING:3 TRAILING:3 \
SLIDINGWINDOW:5:15 MINLEN:35
```

Trimmed Read Length Distributions

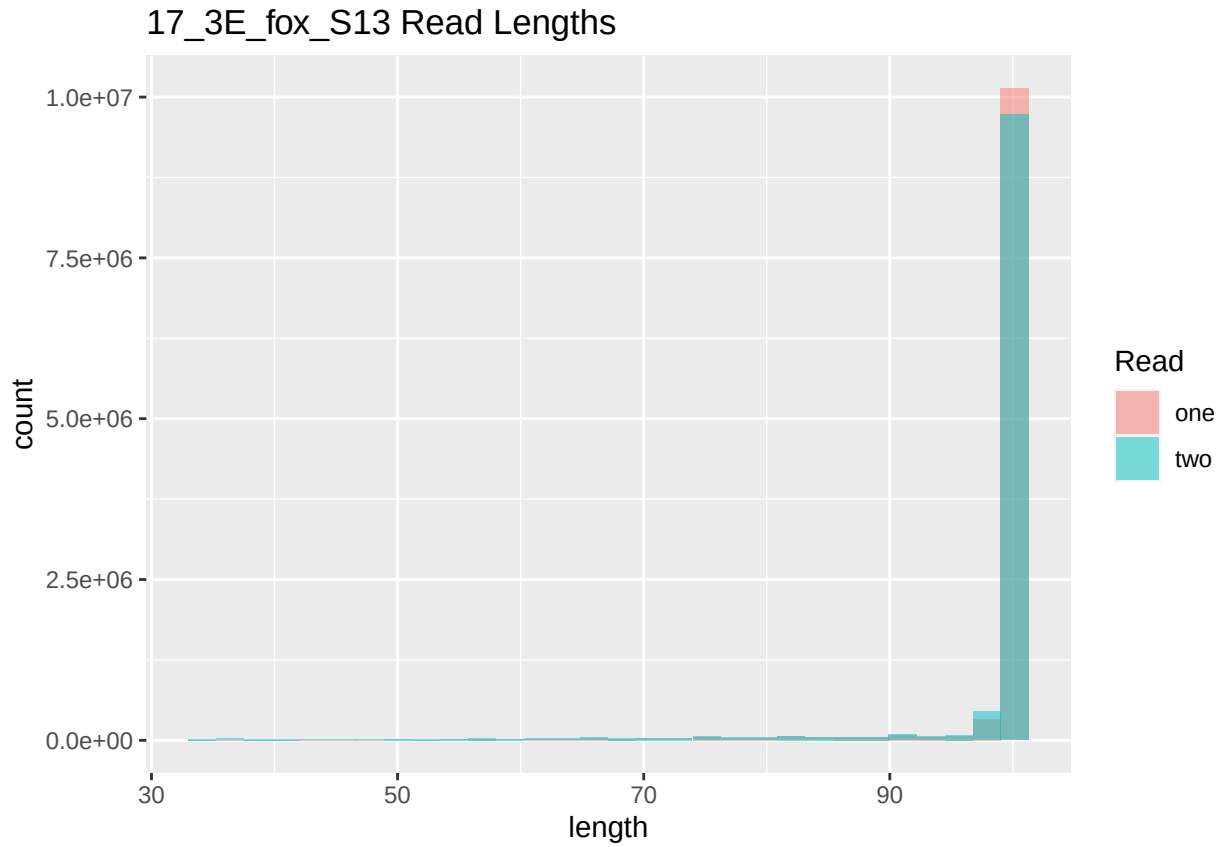
Read 2 was trimmed more extensively than Read 1. This is because Read 2 is generally lower quality than Read 1 and so adapter read-through is more common.

```
S1_R1_trim <- read_tsv("S1_R1.trimmed.nsv", show_col_types = FALSE)
S1_R2_trim <- read_tsv("S1_R2.trimmed.nsv", show_col_types = FALSE)
S13_R1_trim <- read_tsv("S13_R1.trimmed.nsv", show_col_types = FALSE)
S13_R2_trim <- read_tsv("S13_R2.trimmed.nsv", show_col_types = FALSE)

colnames(S1_R1_trim) <- c("length")
colnames(S1_R2_trim) <- c("length")
colnames(S13_R1_trim) <- c("length")
colnames(S13_R2_trim) <- c("length")

S1_R1_trim$Read <- 'one'
S1_R2_trim$Read <- 'two'
S13_R1_trim$Read <- 'one'
S13_R2_trim$Read <- 'two'

S1_Reads <- rbind(S1_R1_trim, S1_R2_trim)
S13_Reads <- rbind(S13_R1_trim, S13_R2_trim)
S1_plot <- ggplot(S1_Reads, aes(length, fill = Read)) + geom_histogram(alpha = 0.5,
aes(y= ..count..), position = 'identity') + ggtitle("1_2A_control_S1 Read Lengths")
S13_plot <- ggplot(S13_Reads, aes(length, fill = Read)) + geom_histogram(alpha = 0.5,
aes(y= ..count..), position = 'identity') + ggtitle("17_3E_fox_S13 Read Lengths")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



PART 3: ALIGNMENT AND STRAND SPECIFICITY

QUESTION 8:

```
Install STAR: conda install star -c bioconda
Install CUTADAPT: conda install -c bioconda cutadapt
INSTALL TRIMMOMATIC: conda install -c bioconda trimmomatic
INSTALL NUMPY: conda install numpy -c bioconda
INSTALL PYSAM: conda install pysam -c bioconda
pip install HTSeq
```

QUESTION 9: I went out to the Ensembl website and downloaded these files directly to my directory on Talapas by grabbing their url and using "wget *url*" Navigating the website: Ensembl > Downloads > FTP Downloads > Mouse > DNA Fasta & GTF Fasta

SLURM to generate alignment database

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=STAR_generate_db
#SBATCH --output=STAR_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=2:00:00
#SBATCH --cpus-per-task=8

/usr/bin/time -v STAR --runThreadN 8 \
--runMode genomeGenerate \
--genomeDir /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/
Mus_musculus.GRCm39.dna.ens104.STAR_2.7.9a \
--genomeFastaFiles /projects/bgmp/nkearns/bioinformatics/Bi623/
QAA/Mus_musculus.GRCm39.dna.primary_assembly.fa \
--sjdbGTFfile /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/
Mus_musculus.GRCm39.104.gtf
--outFileNamePrefix Align_OUTPUT
```

SLURM to Align Reads to Mouse Genomic Database

S1

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=STAR_align
#SBATCH --output=STAR_align_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=2:00:00
#SBATCH --cpus-per-task=8

/usr/bin/time -v STAR --runThreadN 8 \
--runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 \
--alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S1_R1.trimmed.fastq.gz
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S1_R2.trimmed.fastq.gz \
--genomeDir /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/
```

```
Mus_musculus.GRCm39.dna.ens104.STAR_2.7.9a \
--outFileNamePrefix S1_Align_to_ref
```

S13

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=STAR_S13_align
#SBATCH --output=STAR_S13_align_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=2:00:00
#SBATCH --cpus-per-task=8

/usr/bin/time -v STAR --runThreadN 8 \
--runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 --alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S13_R1.trimmed.fastq.gz
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S13_R2.trimmed.fastq.gz \
--genomeDir /projects/bgmp/nkearns/bioinformatics/Bi623/QAA/
Mus_musculus.GRCm39.dna.ens104.STAR_2.7.9a \
--outFileNamePrefix S13_Align_to_ref
```

Mapped and Unmapped Reads

Ran python script.

1_2A_control_S1

Mapped reads: 15627450 Unmapped reads: 305236

17_3E_fox_S13

Mapped reads: 21532850 Unmapped reads: 948692

HTSEQ

```
#!/bin/bash
#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=htseq
#SBATCH --output=S1_htseq_%j.out
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=8:00:00
#SBATCH --cpus-per-task=8

/usr/bin/time -v htseq-count -f sam --stranded=yes --samout=S1_htseq_stranded_out \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S1_Alignment/S1_Align_sorted \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/Mus_musculus.GRCm39.104.gtf

/usr/bin/time -v htseq-count -f sam --stranded=no --samout=S1_htseq_out \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S1_Alignment/S1_Align_sorted \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/Mus_musculus.GRCm39.104.gtf

/usr/bin/time -v htseq-count -f sam --stranded=yes --samout=S13_htseq_stranded_out \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S13_Alignment/S13_Align_sorted \
```

```
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/Mus_musculus.GRCm39.104.gtf

/usr/bin/time -v htseq-count -f sam --stranded=no --samout=S13_htseq_out \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/S13_Alignment/S13_Align_sorted \
/projects/bgmp/nkearns/bioinformatics/Bi623/QAA/Mus_musculus.GRCm39.104.gtf
```

12. I propose that these are from an unstranded library prep. There are substantially more "no_feature" reads in the stranded = yes condition than in the stranded = no condition. More reads map to features in the unstranded kit than in the stranded kit because they can map to both the coding and template strand of the reference. I obtained these counts by running "tail" on the the htseq output files.

S1_stranded no_feature 7163565 = 94.5 %

S1_NOT_stranded no_feature 408922 = 5.5% %

S13_stranded no_feature 9781740 = 90.8 %

S13_NOT_stranded no_feature 994711 = 1.8 %