Beagan Nguy Assignment QAA

Part 1 – Read quality score distributions

1. 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.

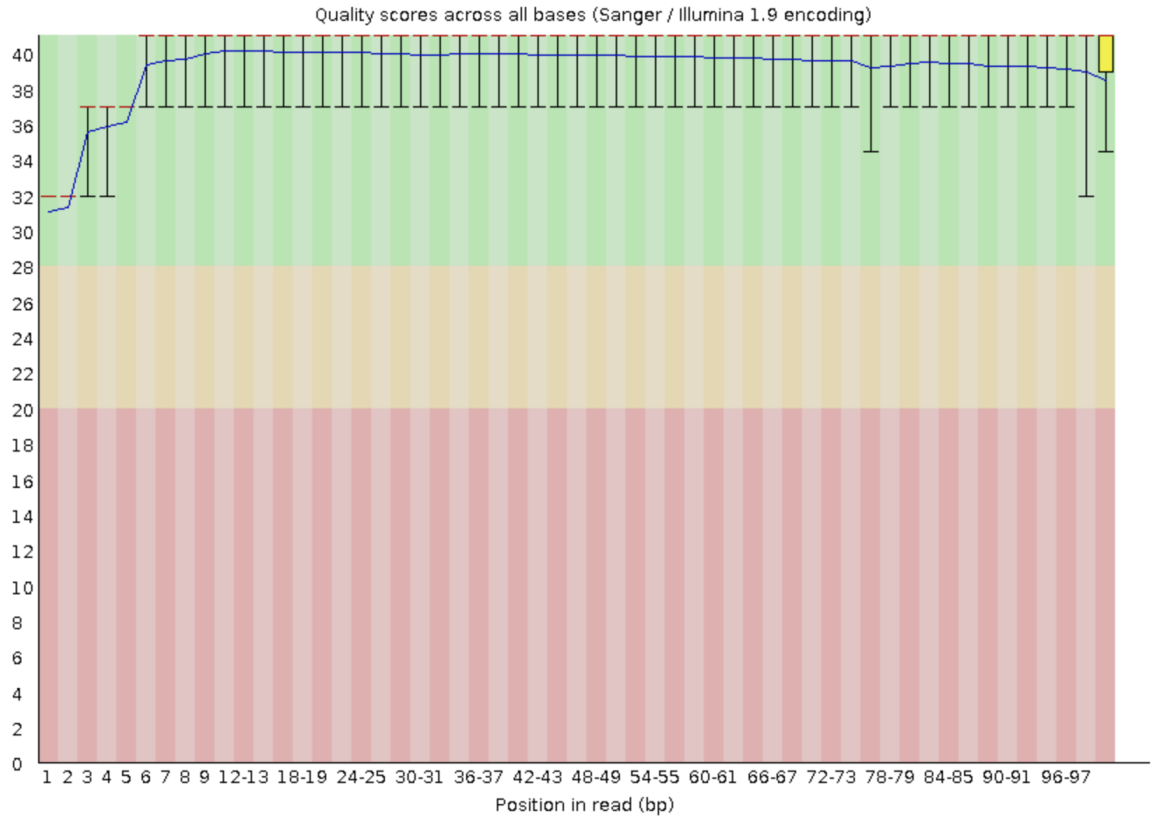
fastqc -o

`fastq files examples: /projects/bgmp/shared/2017_sequencing/demultiplexed/`
`34_4H_both_S24_L008_R2_001.fastq.gz`

The per-base N content is somewhat consistent with the quality score plots. There is a slight amount of N per base at the starting bases compared to the quality score plot where there is a quality drop of the beginning. Meaning there is an inverse relationship at the starting bases of both the plots. However for the quality score plot there is a consistent decrease as the base pair increased whereas the quality score plot stayed uniform. However, the base calls after the few bases are still good quality since most of the base calls have an average quality score of 38-40.

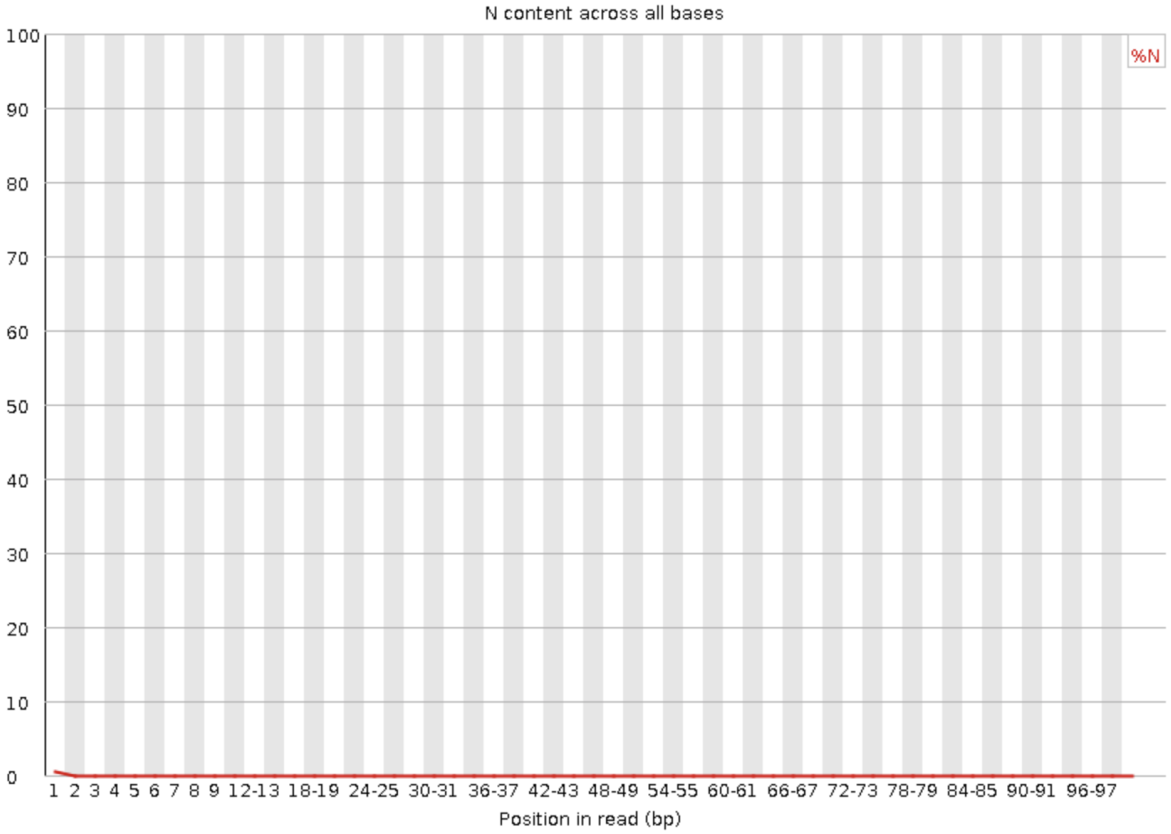
4_2C_mbnl_S4_read_1

Per base sequence quality



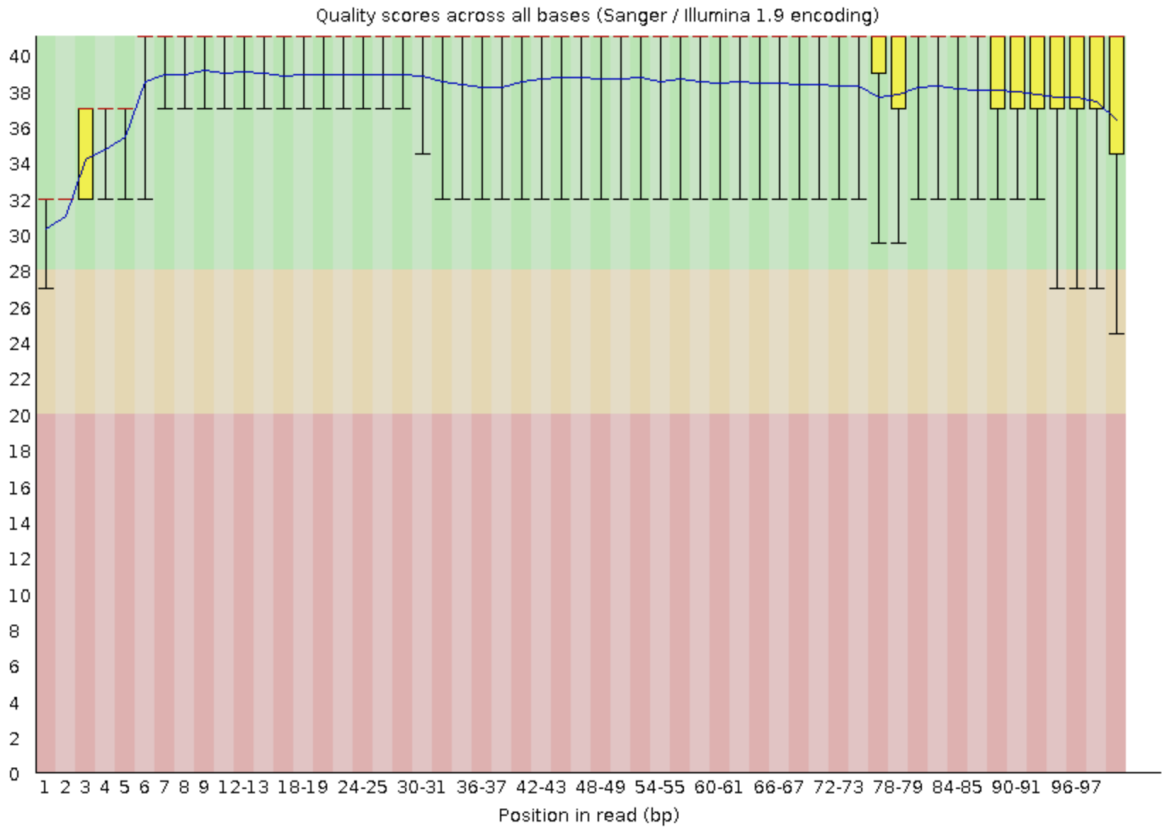
4_2C_mbnl_S4_read_1

Per base N content



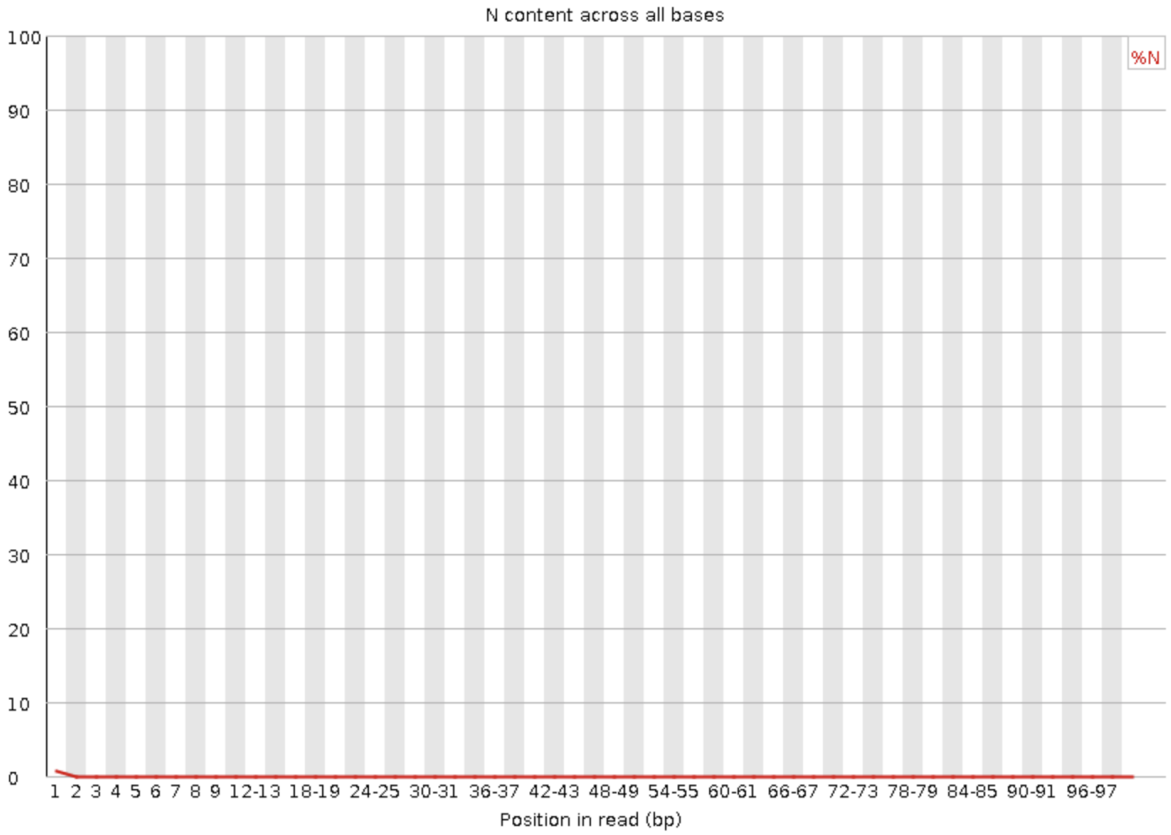
4_2C_mbnl_S4_read_2

Per base sequence quality



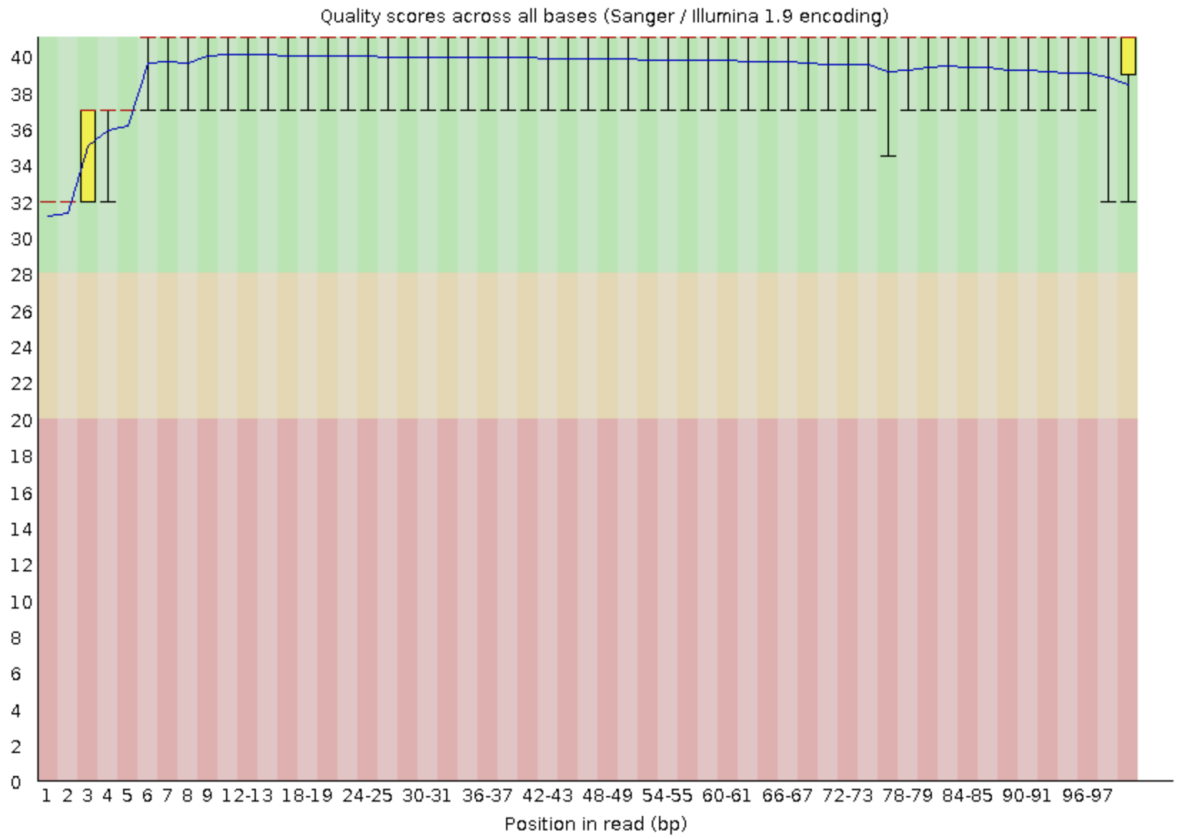
4_2C_mbnl_S4_read_2

Per base N content



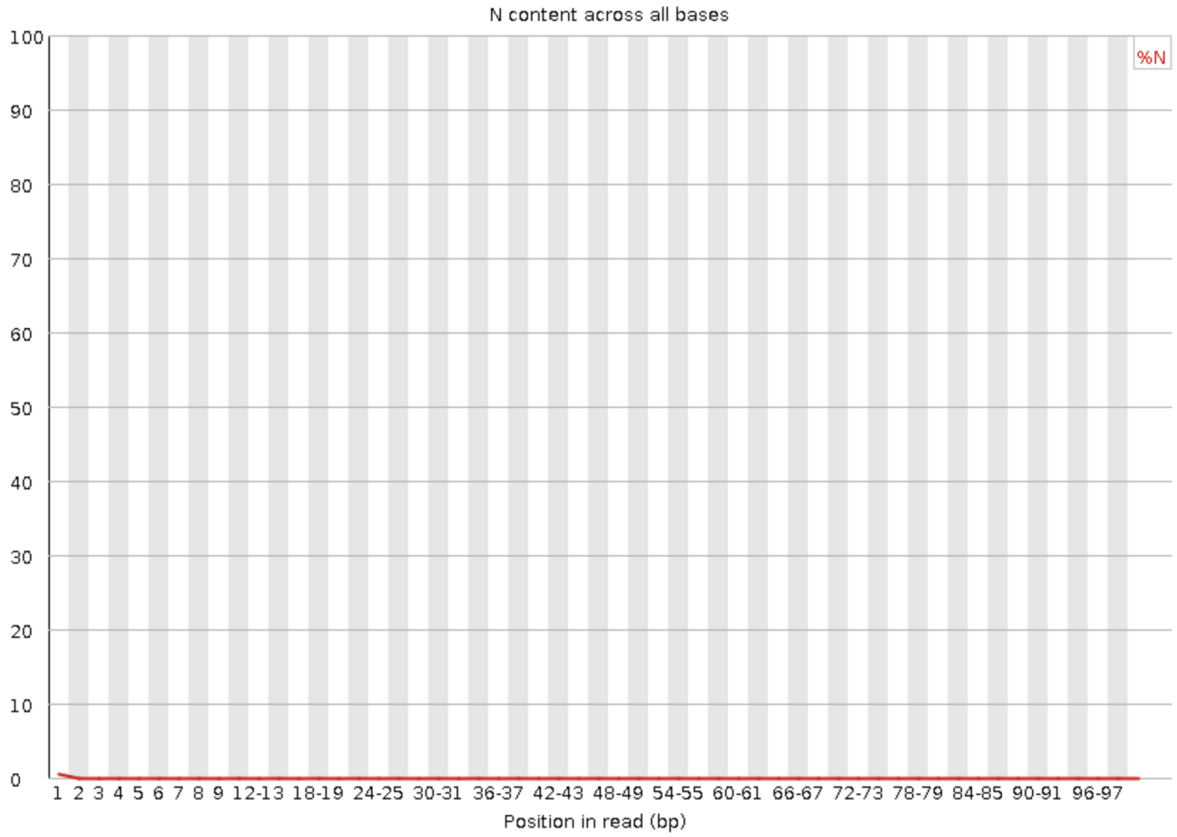
both_s24_read_1

Per base sequence quality



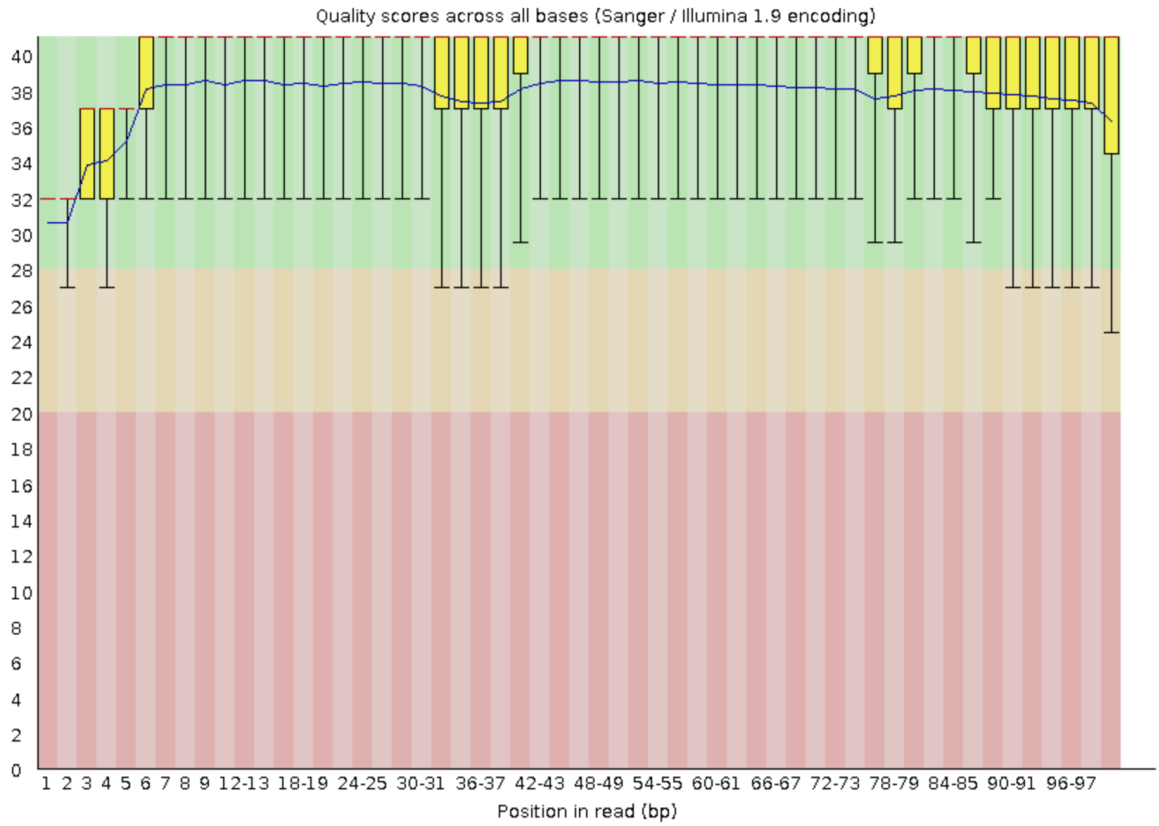
both_s24_read_1

Per base N content



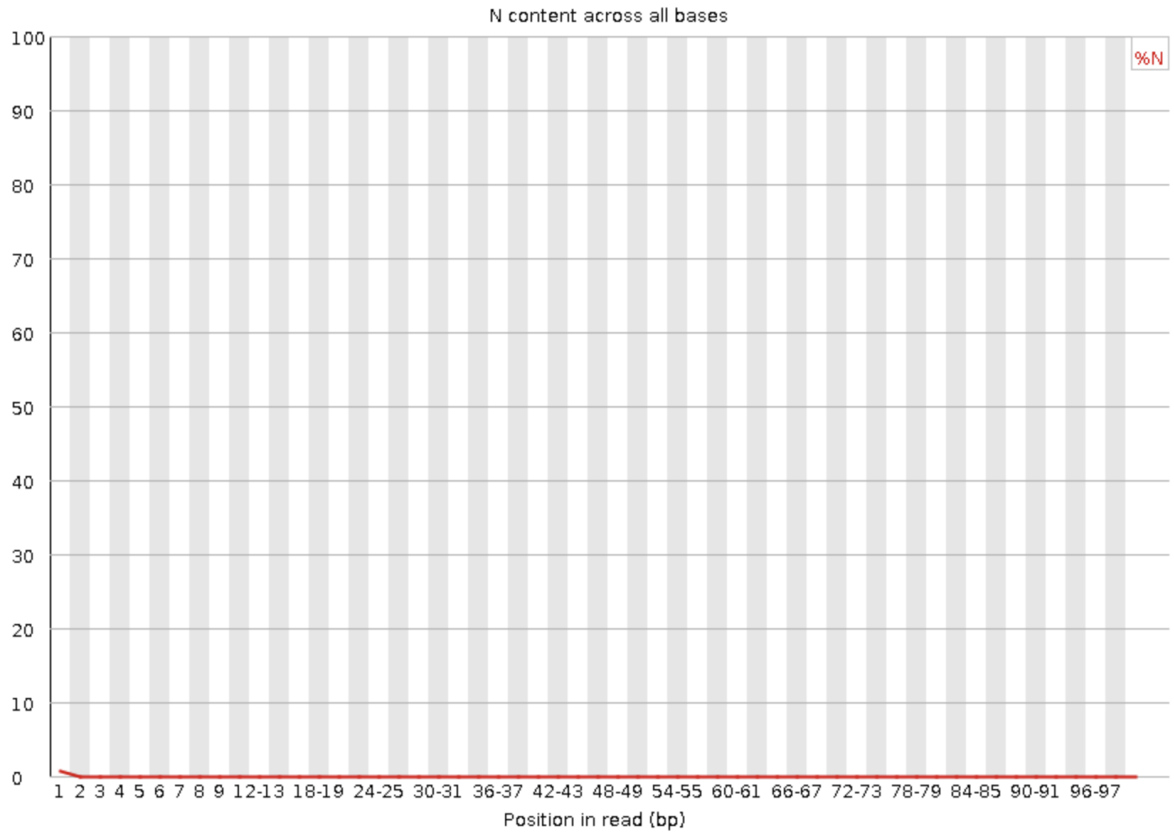
both_s24_read_2

Per base sequence quality

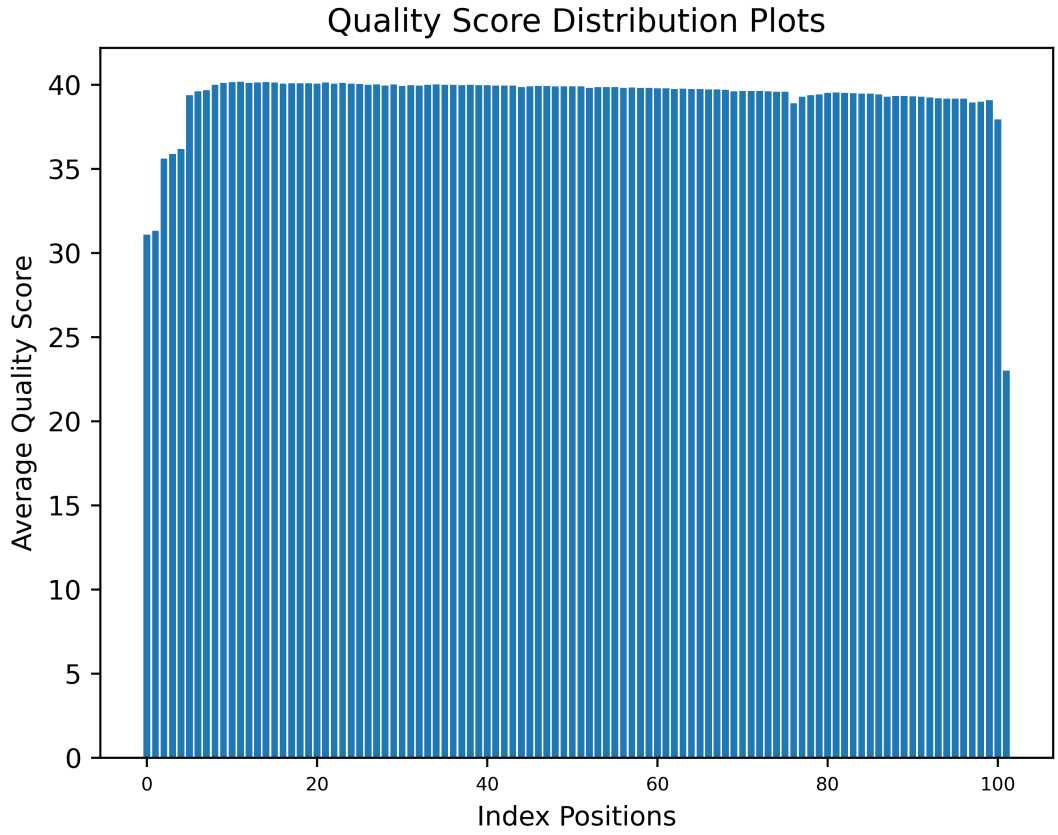


both_s24_read_2

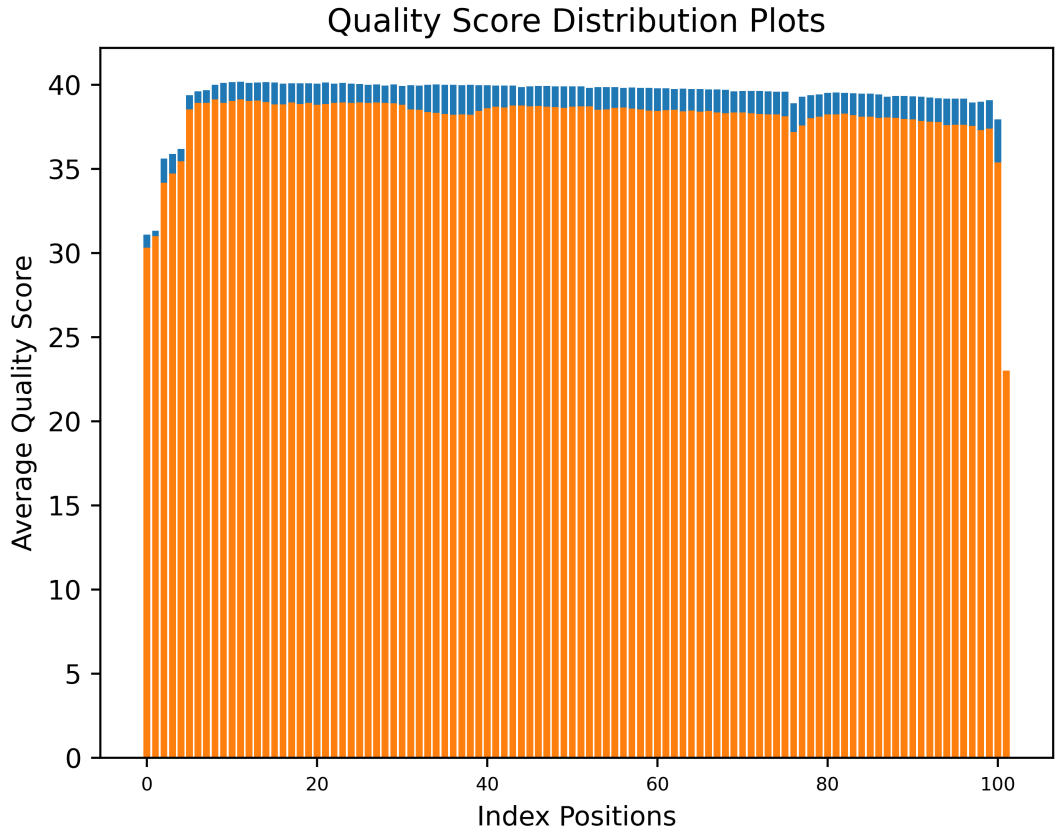
Per base N content



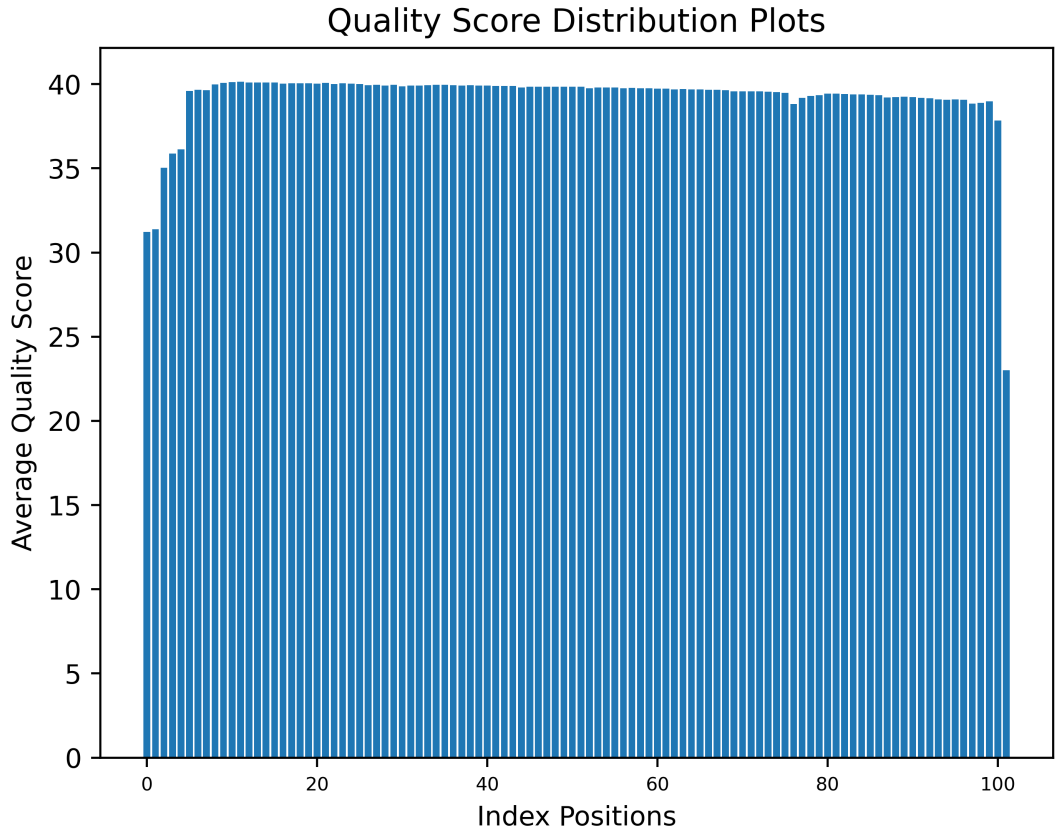
mbnl_S4_read_1



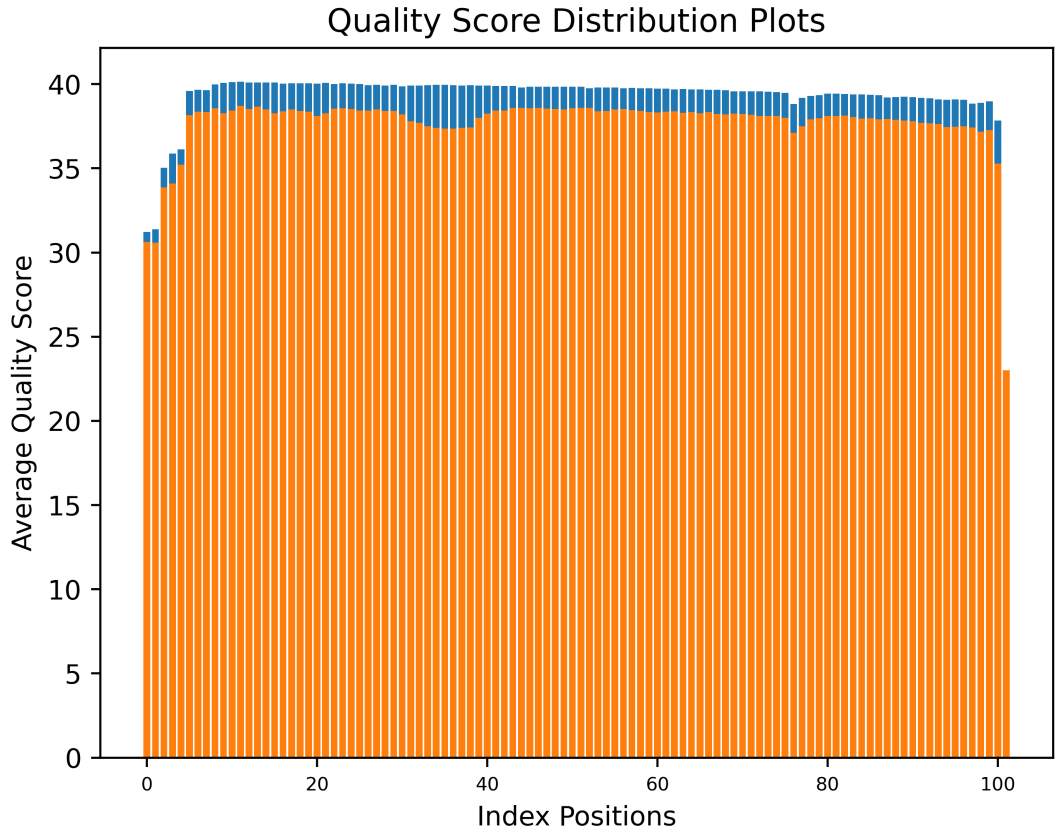
mbnl_S4_read_2



both__S24__read__1



both_S24_read_2



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?

The FastQC generate a similar mean distribution to the plots generated with my demultiplex plotting script. This includes a quality drop at the starting base positions of both plots. The runtime differs much between the FastQC run time of 1 minute 8 seconds and my demultiplex script average runtime per plot of 12 minute 16 seconds. The difference in runtime could be due to ineffiencies in my script. Another attribute to consider is that my script is written in Python whereas FastQC is written in Java. Java can sometimes have a faster runtime than Python because Java is a compiled language whereas Python is a interpreted language.

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

The overall data quality of both my libraries were good because the quality scores were consistently high and the per-base N content were low for both r1 and r2. Though r1 has slightly better overall data quality.

Part 2 - Adaptor trimming comparison

4. Create a new conda environment called QAA and install cutadapt and Trimmomatic. Google around if you need a refresher on how to create conda environments.

5. 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?

34_4H_both_S24

Total read pairs processed: 9,040,597

Read 1 with adapter: 819,166 (9.1%)

Read 2 with adapter: 886,595 (9.8%)

4_2C_mbnl_S4

Total read pairs processed: 9,265,284

Read 1 with adapter: 570,274 (6.2%)

Read 2 with adapter: 637,307 (6.9%)

```
cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o
34_4H_both_S24_L008_R1_001.fastq.gz -p 34_4H_both_S24_L008_R2_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/34_4H_both_S24_L008_R1_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/34_4H_both_S24_L008_R2_001.fastq.gz

cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o
4_2C_mbnl_S4_L008_R1_001.fastq.gz -p 4_2C_mbnl_S4_L008_R2_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/4_2C_mbnl_S4_L008_R1_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/4_2C_mbnl_S4_L008_R2_001.fastq.gz
```

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.

```
# zgrep --color <adapter sequence> <read.fastq.gz>
```

This command searches for sequences in color and for our case the adapter sequence. Typically, the adapter sequence is located towards the right end of each read followed by the barcode. Thus, this shows polymerase synthesize from the 5' to 3' direction and the adapters are located at the 3' end.

6. Use Trimmomatic to quality trim your reads.

```
trimmomatic PE /home/bnguy2/bgmp/Bi623/final_qaa/34_4H_both_S24_L008_R1_001.fastq.gz
/home/bnguy2/bgmp/Bi623/final_qaa/34_4H_both_S24_L008_R2_001.fastq.gz -baseout 34_4H.fastq.gz
LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35

trimmomatic PE /home/bnguy2/bgmp/Bi623/final_qaa/4_2C_mbnl_S4_L008_R1_001.fastq.gz
/home/bnguy2/bgmp/Bi623/final_qaa/4_2C_mbnl_S4_L008_R2_001.fastq.gz -baseout 4_2C_mbnl.fastq.gz
LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35
```

7. Plot the trimmed read length distributions for both R1 and R2 reads (on the same plot).

Use linux commands to count the read length

```
zcat 34_4H_1P.fastq.gz | awk "NR % 4 == 2 { print length()}" > plot_r1_34_4H.txt
zcat 34_4H_2P.fastq.gz | awk "NR % 4 == 2 { print length()}" > plot_r2_34_4H.txt
zcat 4_2C_mbnl_1P.fastq.gz | awk "NR % 4 == 2 { print length()}" > plot_r1_4_2C.txt
zcat 4_2C_mbnl_2P.fastq.gz | awk "NR % 4 == 2 { print length()}" > plot_r2_4_2C.txt
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(ggplot2)
library(knitr)

setwd("/Users/beagannguy/Desktop/R_script/Assignment_4")
getwd()
```

```
## [1] "/Users/beagannguy/Desktop/R_script/Assignment_4"
```

```
r1_4_2C <- read_tsv("plot_r1_4_2C.txt")
```

```
## Rows: 9251394 Columns: 1

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (1): 101
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
r2_4_2C <- read_tsv("plot_r2_4_2C.txt")
```

```
## Rows: 8653140 Columns: 1
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (1): 100
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
r1_4_2C
```

```
## # A tibble: 9,251,394 x 1
##    `101`
##    <dbl>
##  1   101
##  2    83
##  3   101
##  4    85
##  5   101
##  6   101
##  7   101
##  8   101
##  9   101
## 10   100
## # ... with 9,251,384 more rows
```

```
r1_34_4H <- read_tsv("plot_r1_34_4H.txt")
```

```
## Rows: 8653140 Columns: 1
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (1): 101
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
r2_34_4H <- read_tsv("plot_r2_34_4H.txt")
```

```
## Rows: 8653140 Columns: 1
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (1): 100

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
r1_4_2C %>%
  mutate(Read = "1") -> r1_4_2C
r2_4_2C %>%
  mutate(Read = "2") -> r2_4_2C

r1_34_4H %>%
  mutate(Read = "1") -> r1_34_4H
r2_34_4H %>%
  mutate(Read = "2") -> r2_34_4H

colnames_reads <- c("Lengths", "Read")
colnames(r1_4_2C) <- colnames_reads
colnames(r2_4_2C) <- colnames_reads

colnames(r1_34_4H) <- colnames_reads
colnames(r2_34_4H) <- colnames_reads

merge_4_2C <- rbind(r1_4_2C, r2_4_2C)
merge_4_2C
```

```
## # A tibble: 17,904,534 x 2
##    Lengths Read
##      <dbl> <chr>
##  1     101 1
##  2      83 1
##  3     101 1
##  4      85 1
##  5     101 1
##  6     101 1
##  7     101 1
##  8     101 1
##  9     101 1
## 10     100 1
## # ... with 17,904,524 more rows
```

```
merge_34_4H <- rbind(r1_34_4H, r2_34_4H)
merge_34_4H
```

```
## # A tibble: 17,306,280 x 2
##    Lengths Read
##      <dbl> <chr>
##  1     101 1
##  2      75 1
##  3     101 1
```

```
##  4     101 1
##  5     101 1
##  6     101 1
##  7     101 1
##  8     101 1
##  9      65 1
## 10     101 1
## # ... with 17,306,270 more rows
```

```
#head(merge_rbh, 10)
#tail(merge_rbh, 10)
```

```
merge_4_2C %>%
  ggplot(aes(x=Lengths, fill=Read, colour="#006f9e")) + geom_histogram(position="dodge") +
  labs(title="Distribution of Quality Checked Reads") + scale_y_continuous(trans = "log10")
```

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


```
#merge_34_4H %>%
#  ggplot(aes(x=Lengths, fill=Read, colour=Read)) + geom_density(alpha = 0.2) +labs(title="Distribution

merge_34_4H %>%
  ggplot(aes(x=Lengths, fill=Read, colour="#006f9e")) + geom_histogram(position="dodge") +
  labs(title="Distribution of Quality Checked Reads") + scale_y_continuous(trans = "log10")
```

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

From these plots, read 2 looks to be trimmed more extensively on both samples. We can expect this to happen since samples are gradually degraded over time on the illumina plates as they go through chemical processes. Read 1 is often seqeunced first then read 2, so when its time to sequence read 2 it has already been degraded.

Install sofware.

9. 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.

Align database using STAR. To build the STAR database, a reference fasta file and gtf file were downloaded from ensembl

```
/usr/bin/time -v STAR --runThreadN 8 \
--runMode genomeGenerate \
--genomeDir /home/bnguy2/bgmp/Bi623/final_qaa/Mus_musculus.GRCm39.dna.ens104.STAR.2.7.9a \
--genomeFastaFiles Mus_musculus.GRCm39.dna.primary_assembly.fa \
--sjdbGTFfile Mus_musculus.GRCm39.104.gtf
```

To align the fastq files

```
/usr/bin/time -v STAR --runThreadN 8 --runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 --alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn /home/bnguy2/bgmp/Bi623/final_qaa/34_4H_1P.fastq.gz
/home/bnguy2/bgmp/Bi623/final_qaa/34_4H_2P.fastq.gz  \
--genomeDir Mus_musculus.GRCm39.dna.ens104.STAR.2.7.9a \
--outFileNamePrefix Mus_musculus_mbnl_s4_
```

```
/usr/bin/time -v STAR --runThreadN 8 --runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 --alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn /home/bnguy2/bgmp/Bi623/final_qaa/4_2C_mbnl_1P.fastq.gz
/home/bnguy2/bgmp/Bi623/final_qaa/4_2C_mbnl_2P.fastq.gz  \
--genomeDir Mus_musculus.GRCm39.dna.ens104.STAR.2.7.9a \
--outFileNamePrefix Mus_musculus_mbnl_s4_
```

10. Using your script from PS8 in Bi621, report the number of mapped and unmapped reads from each of your 2 sam files.

Run ps8.py on both sam files.

Mus_musculus_both_s24_Aligned.out.sam

Total: 17306282

Number of Mapped Reads: 16822672, Number of Unmapped Reads 483610

Mus_musculus_mbnl_s4_Aligned.out.sam

Total: 17960762

Number of Mapped Reads: 17172669, Number of Unmapped Reads 788093

11. 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.

Sort Sam files using htseq-count

```
samtools view -u  Mus_musculus_both_s24_Aligned.out.sam | samtools sort -n | samtools view -h -o
Mus_musculus_sorted_both_s24_Aligned.out.sam
```

```
samtools view -u  Mus_musculus_mbnl_s4_Aligned.out.sam | samtools sort -n | samtools view -h -o
Mus_musculus_sorted_mbnl_s4_Aligned.out.sam
```

Run htseq count

```
/usr/bin/time -v htseq-count --stranded=yes Mus_musculus_sorted_both_s24_Aligned.out.sam
Mus_musculus_sorted_mbnl_s4_Aligned.out.sam Mus_musculus.GRCm39.104.gtf > htseq_count_yes_stranded.out

__no_feature    7559101 7807229
__ambiguous 8234    5212
__too_low_aQual 11962   9632
__not_aligned   235491  388842
__alignment_not_unique  369262  420417

/usr/bin/time -v htseq-count --stranded=no Mus_musculus_sorted_both_s24_Aligned.out.sam
Mus_musculus_sorted_mbnl_s4_Aligned.out.sam Mus_musculus.GRCm39.104.gtf > htseq_count_no_stranded.out

__no_feature    471489  707441
__ambiguous 410505  409736
__too_low_aQual 11962   9632
__not_aligned   235491  388842
__alignment_not_unique  369262  420417

/usr/bin/time -v htseq-count --stranded=reverse Mus_musculus_sorted_both_s24_Aligned.out.sam
Mus_musculus_sorted_mbnl_s4_Aligned.out.sam Mus_musculus.GRCm39.104.gtf > htseq_count_reverse_stranded.

__no_feature    682220  789010
__ambiguous 146358  141694
__too_low_aQual 11962   9632
__not_aligned   235491  388842
__alignment_not_unique  369262  420417

awk '{total+=$2} $1 ~/__no_feature|__ambiguous/ {nomap+=$2} {map=total-nomap} {per=(map/total)*100}
ENDFILE {printf "\nTotalRead:%d NoMap:%d TotalMapReads:%d %TotalRead:%f\n", total, nomap, map, per}'
htseq_count_yes_stranded.out

TotalRead:8653141 NoMap:7567335 TotalMapReads:1085806 %TotalReadMapped:12.548114

awk '{total+=$2} $1 ~/__no_feature|__ambiguous/ {nomap+=$2} {map=total-nomap} {per=(map/total)*100}
ENDFILE {printf "\nTotalRead:%d NoMap:%d TotalMapReads:%d %TotalRead:%f\n", total, nomap, map, per}'
htseq_count_no_stranded.out

TotalRead:8653141 NoMap:881994 TotalMapReads:7771147 %TotalReadMapped:89.807239

awk '{total+=$2} $1 ~/__no_feature|__ambiguous/ {nomap+=$2} {map=total-nomap} {per=(map/total)*100}
ENDFILE {printf "\nTotalRead:%d NoMap:%d TotalMapReads:%d %TotalRead:%f\n", total, nomap, map, per}'
htseq_count_reverse_stranded.out

TotalRead:8653141 NoMap:828578 TotalMapReads:7824563 %TotalReadMapped:90.424541

Notes.
The amound of reads that maps to stranded and reverse stranded strands rougly adds totals the reads
that mapped to unstranded strands.
```

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 samples came from a stranded kit because 12.54% of the reads are stranded as opposed to 89.80% of the reads are unstranded. If it was unstranded there would be a roughly equal percentage of stranded and unstranded reads, instead our samples are not.