QAA

Wesley Rosales

9/7/2021

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.

See fastqc_run.sbatch for details

Two libraries assessed: 11_2H_both_S9_L008, 29_4E_fox_S21_L008

11_2H_both_S9_L008

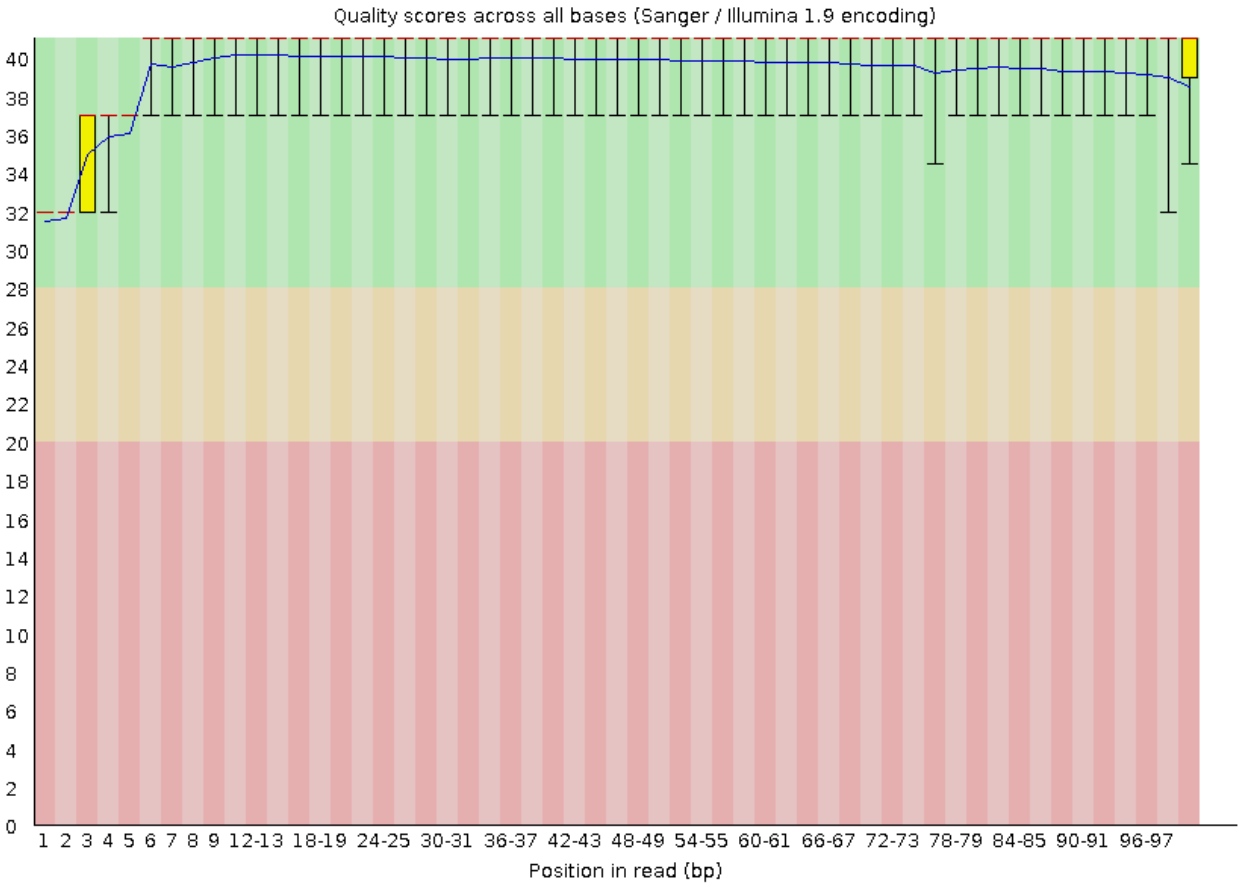

Figure 1: "11_2H_both_S9_L008 Read 1 Quality Distribution"

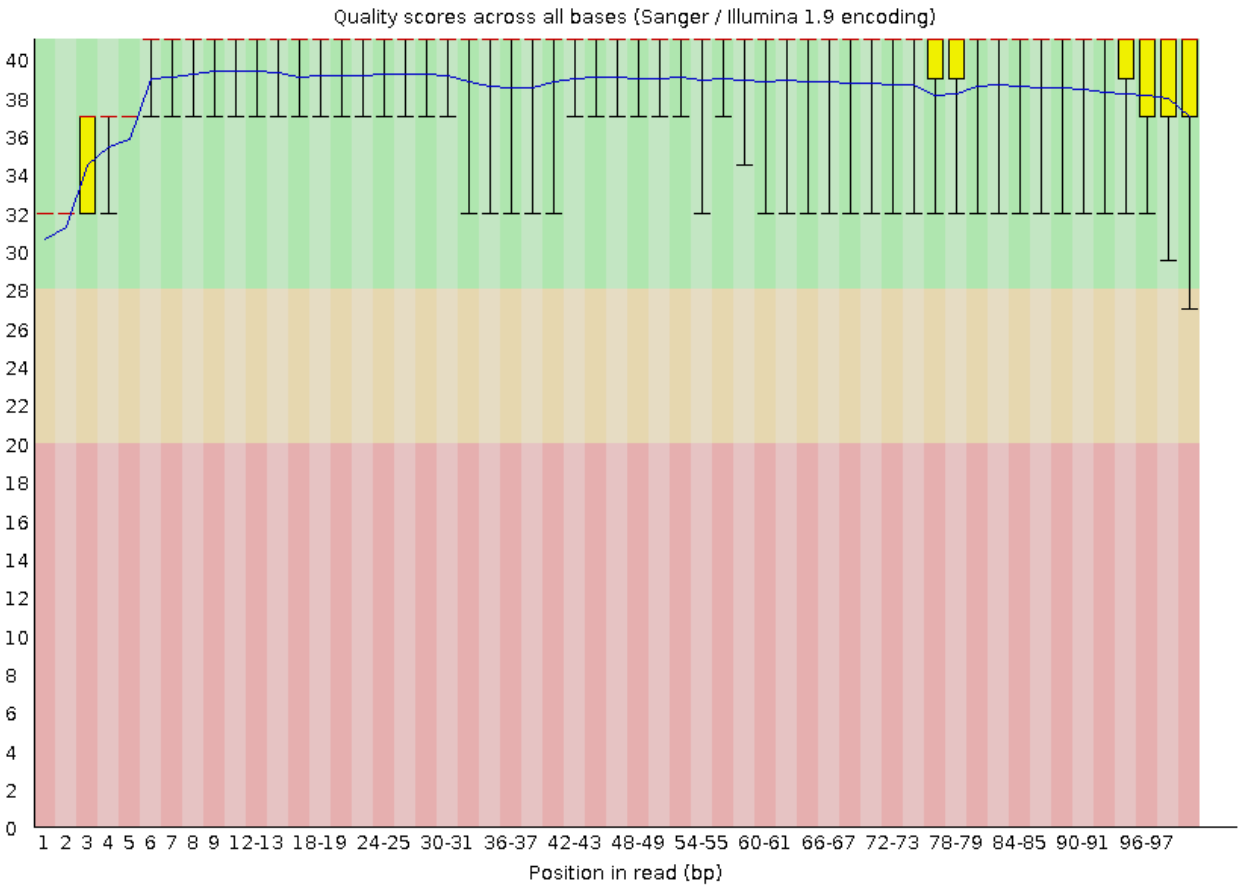

Figure 2: "11_2H_both_S9_L008 Read 2 Quality Distribution"

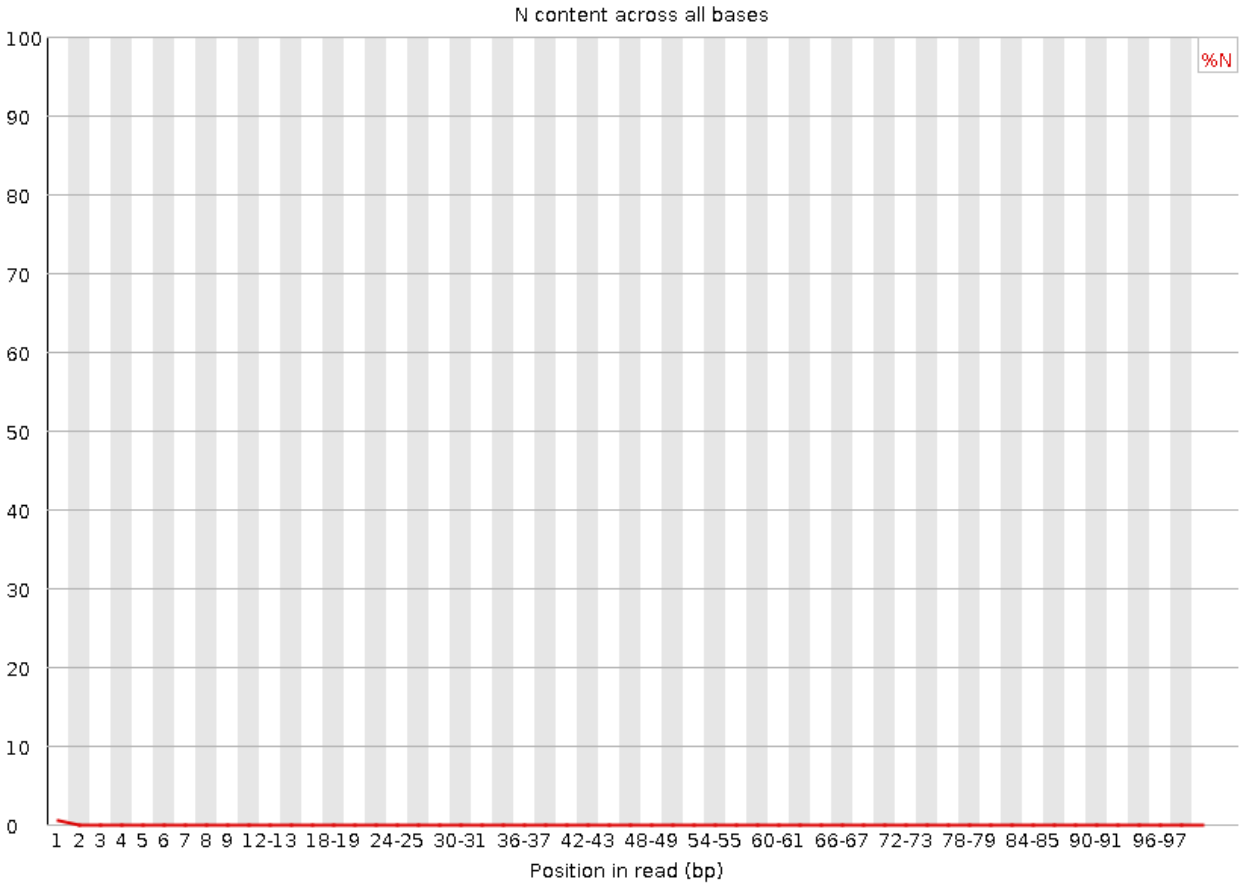

Figure 3: "11_2H_both_S9_L008 Read 1 Per Base N Content"

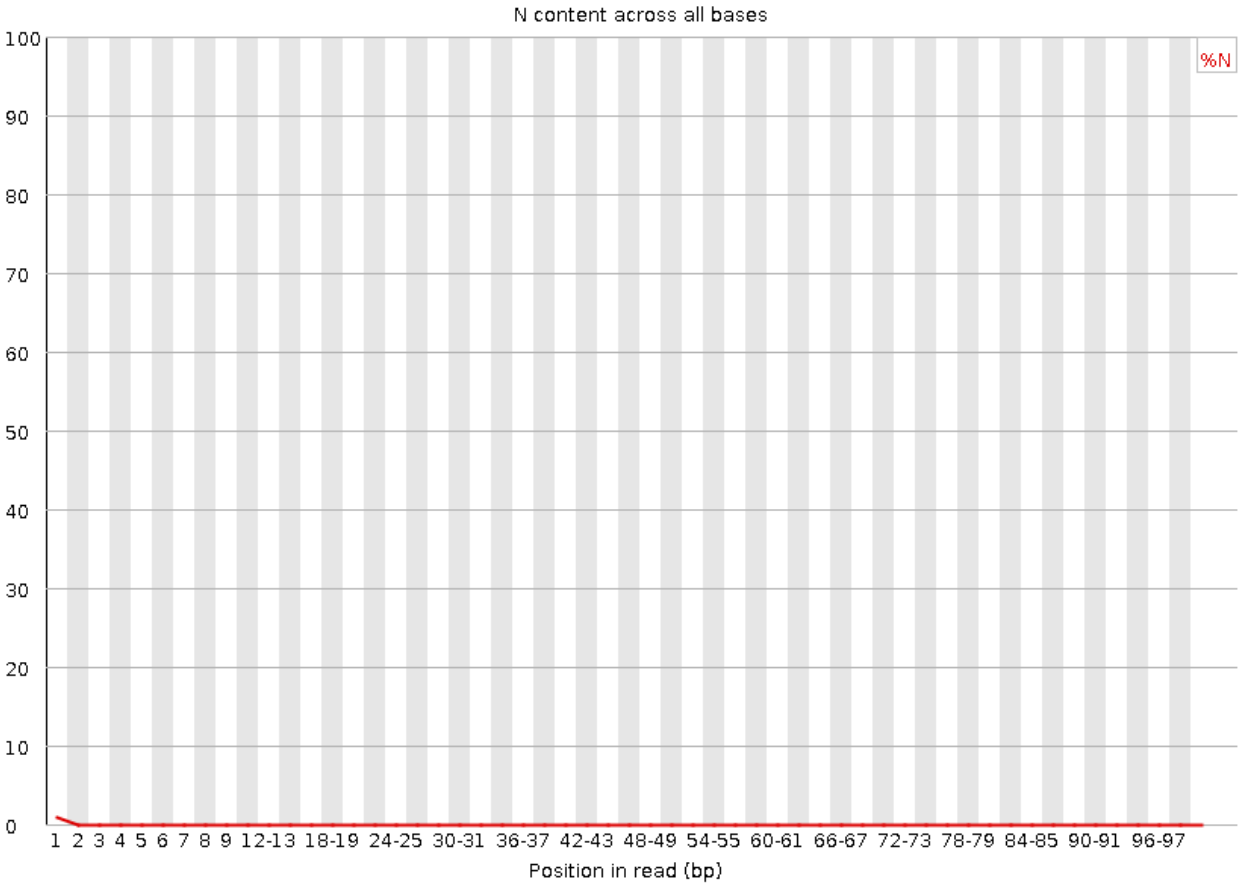

Figure 4: "11_2H_both_S9_L008 Read 2 Per Base N Content"

29_4E_fox_S21_L008

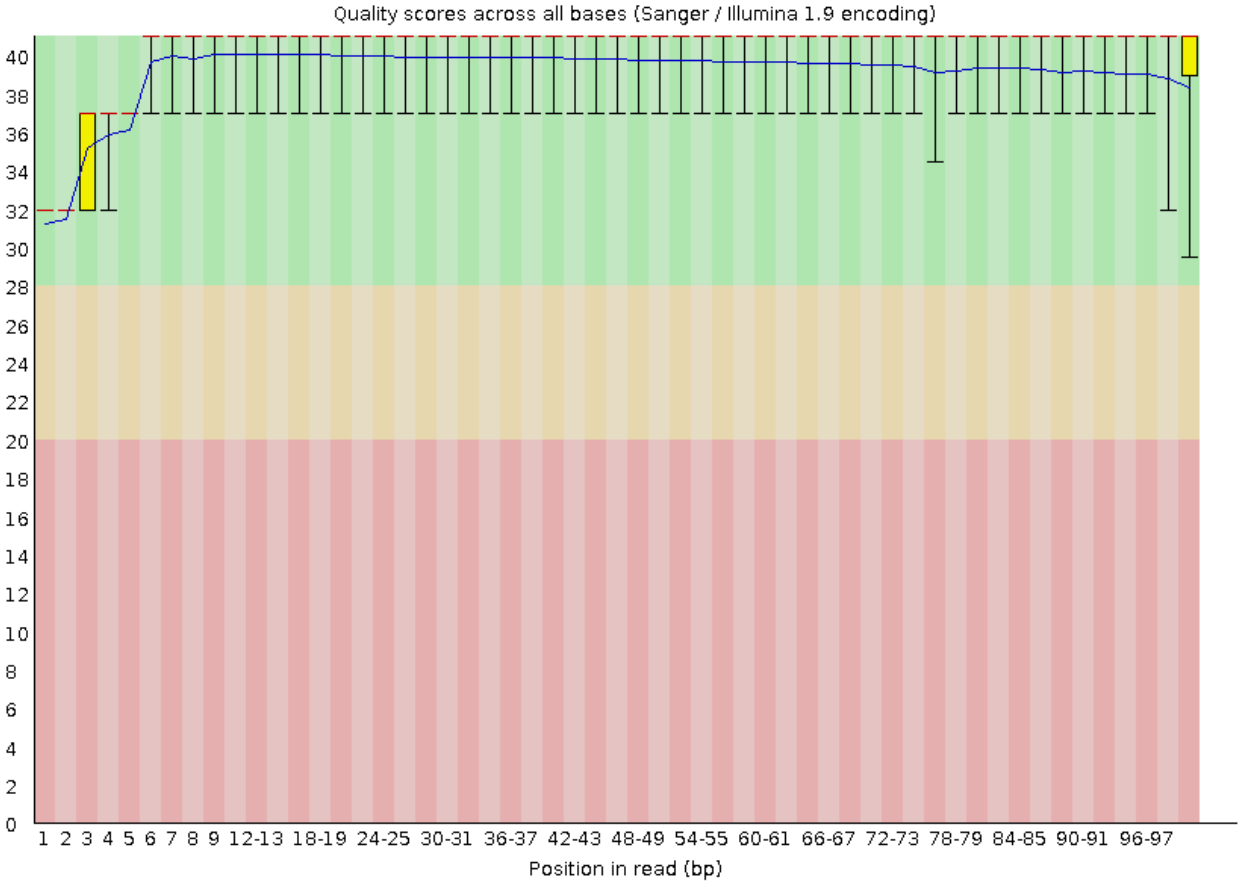

Figure 5: "29_4E_fox_S21_L008 Read 1 Quality Distribution"

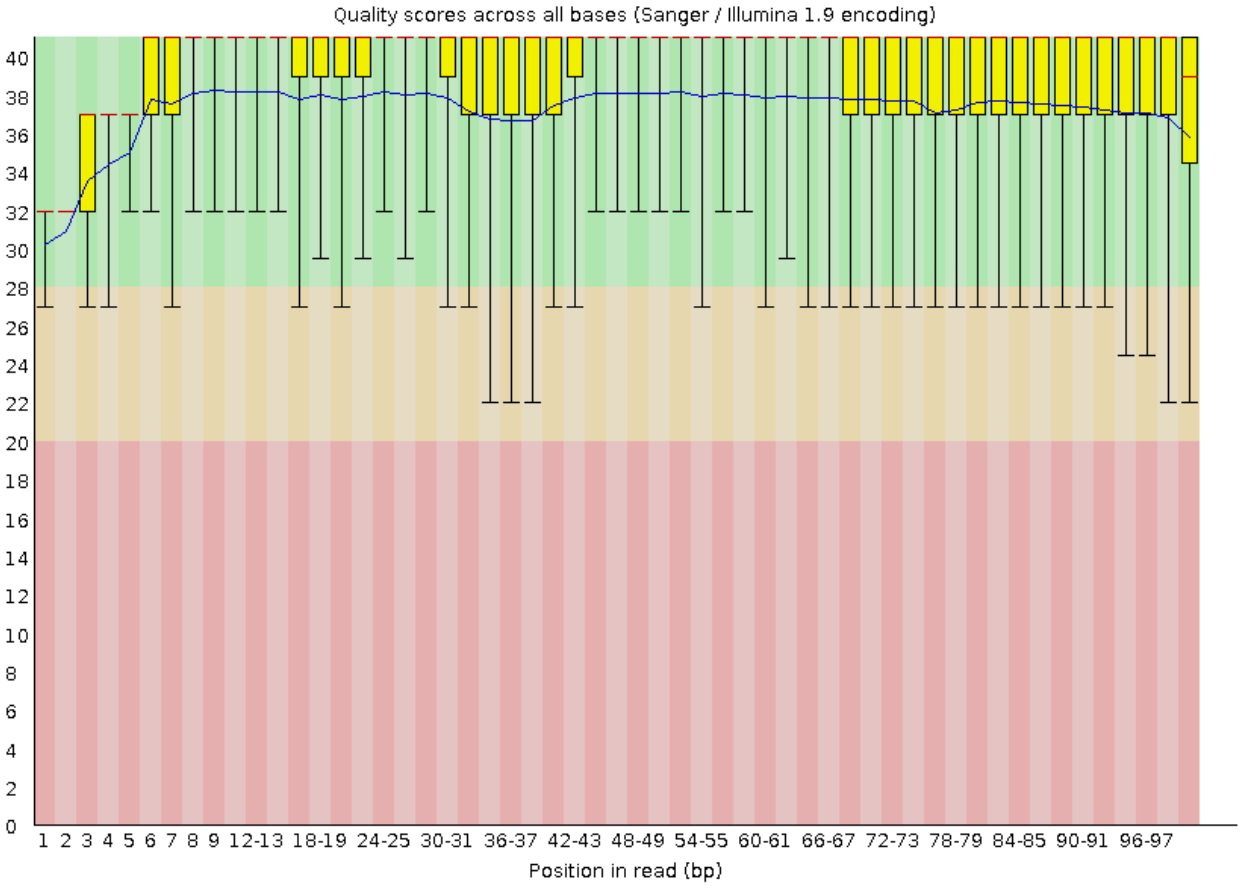

Figure 6: "29_4E_fox_S21_L008 Read 2 Quality Distribution"

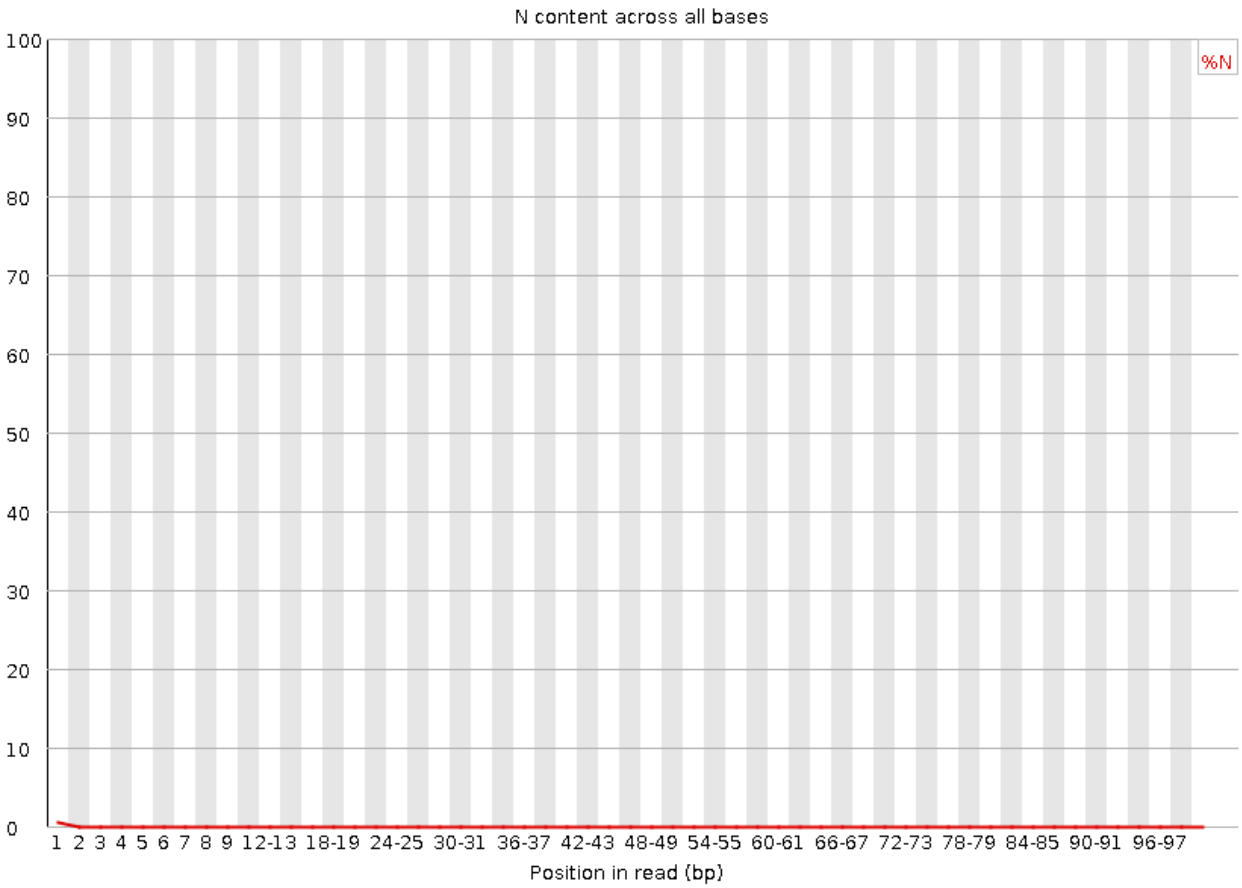

Figure 7: "29_4E_fox_S21_L008 Read 1 Per Base N Content"

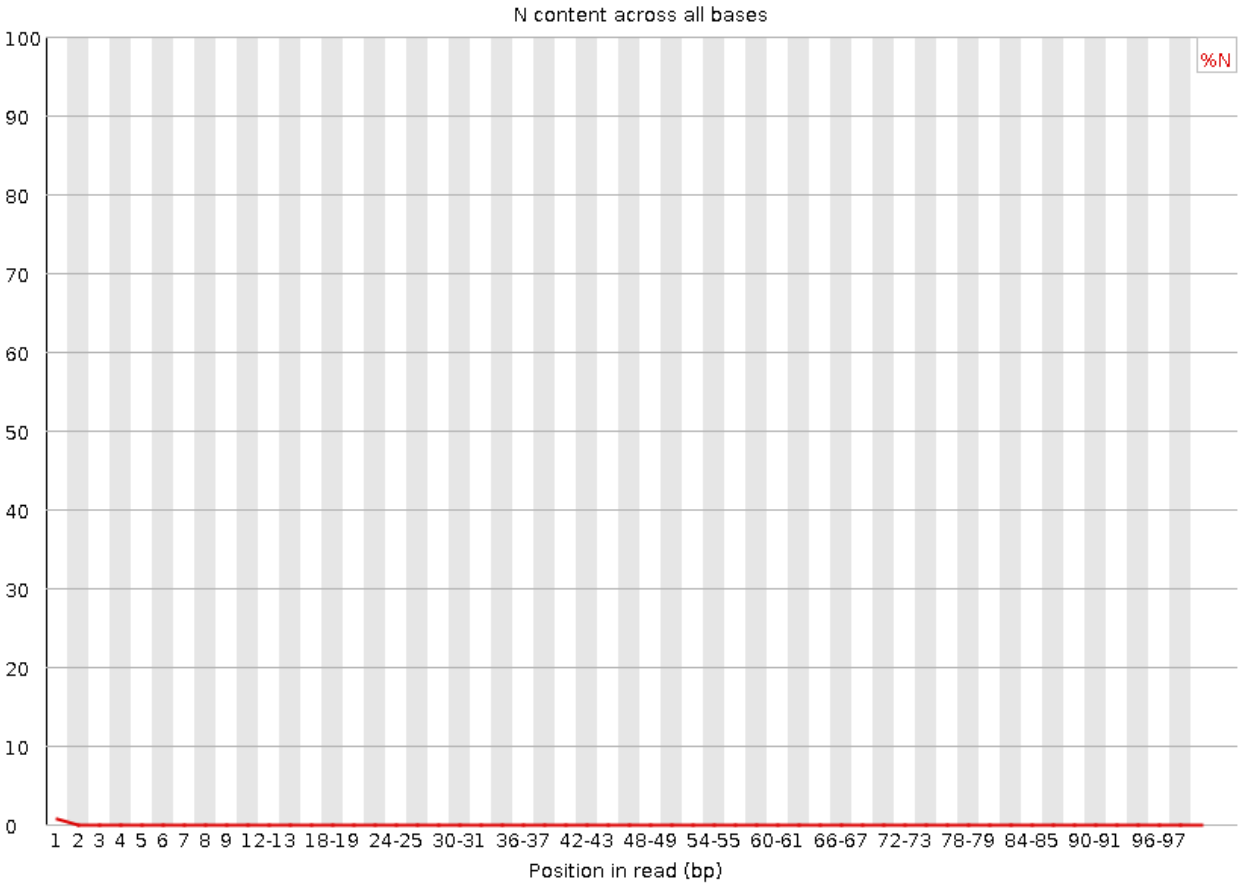

Figure 8: "29_4E_fox_S21_L008 Read 2 Per Base N Content"

The per-base N content graphs for both libraries are consistent with their respective quality score distributions. The lowest quality bases are the beginning few and the highest per-base N content is also in the first few bases.

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?

See qual_score.sbatch for details

Figure 9: "11_2H_both_S9_L008 Read 1 Quality Distribution Custom Script"

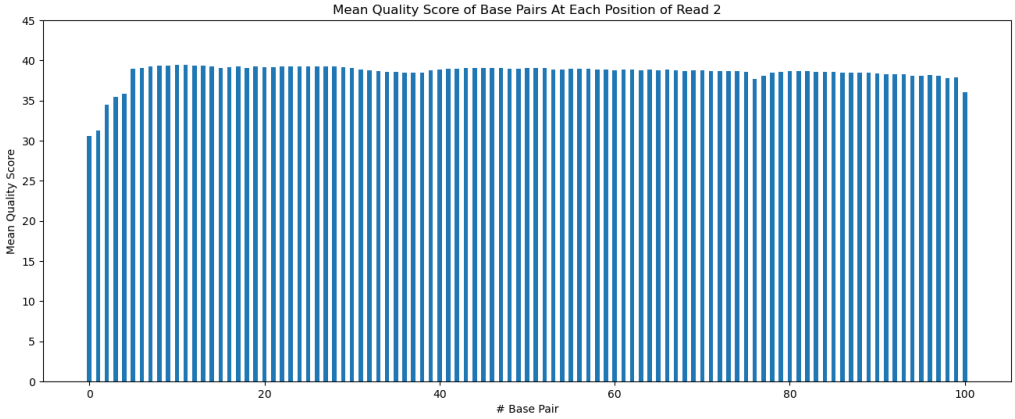

Figure 10: "11_2H_both_S9_L008 Read 2 Quality Distribution Custom Script"

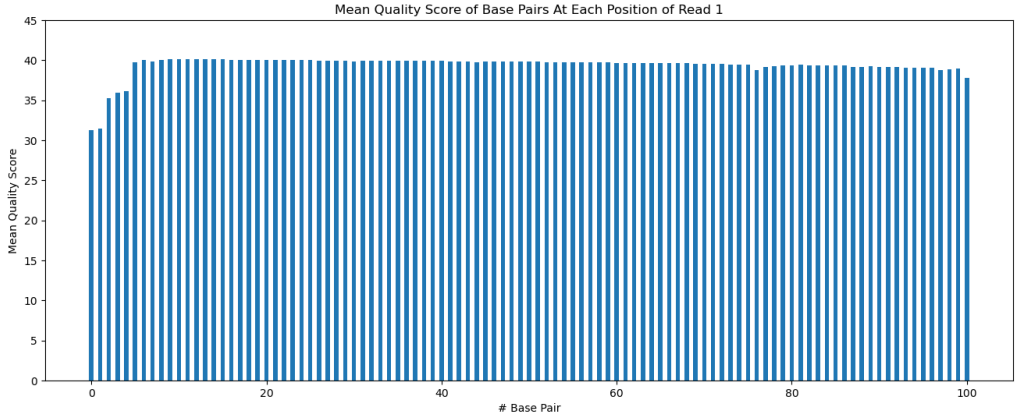

Figure 11: "29_4E_fox_S21_L008 Read 1 Quality Distribution Custom Script"

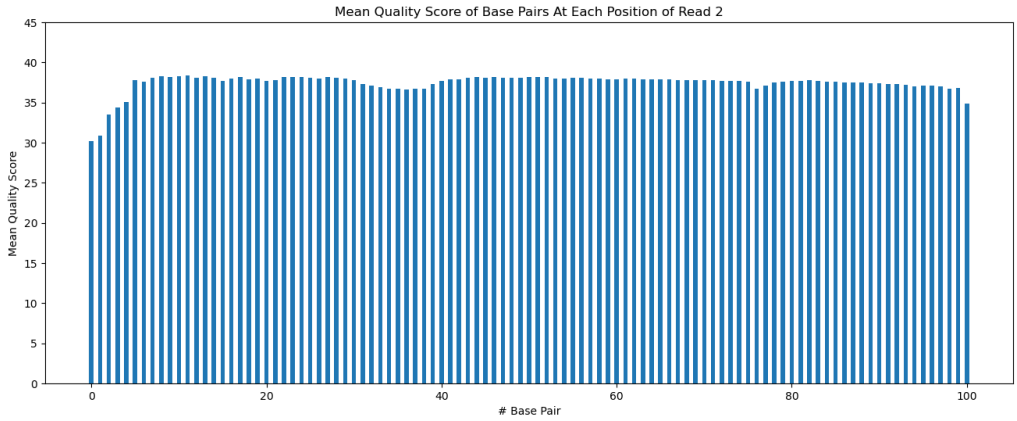

Figure 12: "29_4E_fox_S21_L008 Read 2 Quality Distribution Custom Script"

The Fastqc quality distributions match the quality distributions output by my custom python script (qual_score_dist.py). The runtime of the programs, though, differs substantially. For the 11_2H_both_S9_L008 library, the Fastqc runtime was 174.70 second and 52.38 seconds for the 29_4E_fox_S21_L008 library. My script runtime was 769.47 seconds for 11_2H_both_S9_L008 and 234.90 seconds for the 29_4E_fox_S21_L008 library. Considering that the fastqc command also outputs numerous other graphs, my code is much slower than fastqc. This could be due to many slow sections of my code (multiple for loops, converting from binary to human readable text) that fastqc may not have.

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

Both of my libraries have mostly good quality reads. The only potential issue in terms of quality may come from the read 2 section of the 29_4E_fox_S21_L008 library since some of the qualities may have a Q-score below 28, though none of them seem to have a Q-score below 20 in any position which is good. Both the libraries also seem to have very low N content per base, even the first base which usually has the highest N content.

Part 2 – Adaptor trimming comparison

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

See cutadapt.sbatch for details

Fastqc output suggests that Illumina universal adapters were used

Read 1 Adapter: AGATCGGAAGAGCACACGTCTGAACTCCAGTCA

Read 2 Adapter: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

11_2H_both_S9_L008 Forward read trimmed and total counts

```
zcat 11_2H_both_S9_L008_trim_outputs/11_2H_both_S9_L008_R1_cutadapt_output.fastq.gz | \
sed -n "2~4p" | awk '{if(length != 101){print length}}' | wc -l
874706
zcat 11_2H_both_S9_L008_trim_outputs/11_2H_both_S9_L008_R1_cutadapt_output.fastq.gz | \
sed -n "2~4p" | wc -l
17919193
```

11_2H_both_S9_L008 reverse reads trimmed counts

```
zcat 11_2H_both_S9_L008_trim_outputs/11_2H_both_S9_L008_R2_cutadapt_output.fastq.gz | \
sed -n "2~4p" | awk '{if(length != 101){print length}}' | wc -l
1016991
```

29_4E_fox_S21_L008 forward reads trimmed and total counts

```
zcat 29_4E_fox_S21_L008_trim_outputs/29_4E_fox_S21_L008_R1_cutadapt_output.fastq.gz | \
sed -n "2~4p" | awk '{if(length != 101){print length}}' | wc -l
361886
zcat 29_4E_fox_S21_L008_trim_outputs/29_4E_fox_S21_L008_R1_cutadapt_output.fastq.gz | \
sed -n "2~4p" | wc -l
4827433
```

29_4E_fox_S21_L008 reverse reads trimmed counts

```
zcat 29_4E_fox_S21_L008_trim_outputs/29_4E_fox_S21_L008_R2_cutadapt_output.fastq.gz | \
sed -n "2~4p" | awk '{if(length != 101){print length}}' | wc -l
400819
```

```
11_2H_both_S9_L008 proportion trimmed:
Forward reads: 874706/17919193 = 0.04881392
Reverse reads: 1016991/17919193 = 0.05675429

29_4E_fox_S21_L008 proportion trimmed:
Forward reads: 361886/4827433 = 0.07496448
Reverse reads: 400819/4827433 = 0.08302943
```

Similar proportions of forwards and reverse reads were trimmed.

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.

I searched for the adapters specified in the smaller 29_4E_fox_S21_L008 library (as shown by the faster runtimes for fastqc and my qual_score_dist.py script). I searched for both the adapters in each of the .fastq.gz files and only found hits for read 1 adapter in the read 1 file and the read 2 adapter in the read 2 file. I then searched for the reverse complement of both adapters in each read file and found 0 hits. Altogether, this informs me that the orientation of the adapters is as shown.

```
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R1_001.fastq.gz | \
sed -n "2~4p" | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
39701
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R1_001.fastq.gz | \
sed -n "2~4p" | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R2_001.fastq.gz | \
sed -n "2~4p" | grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
39929
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R2_001.fastq.gz | \
sed -n "2~4p" | grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
0
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R2_001.fastq.gz | \
sed -n "2~4p" | grep "ACACTCTTTCCCTACACGACGCTCTTCCGATCT" | wc -l
0
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R1_001.fastq.gz | \
sed -n "2~4p" | grep "ACACTCTTTCCCTACACGACGCTCTTCCGATCT" | wc -l
0
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R1_001.fastq.gz | \
sed -n "2~4p" | grep "TGACTGGAGTTCAGACGTGTGCTCTTCCGATCT" | wc -l
0
zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/29_4E_fox_S21_L008_R2_001.fastq.gz | \
sed -n "2~4p" | grep "TGACTGGAGTTCAGACGTGTGCTCTTCCGATCT" | wc -l
0
```

6. Use Trimmomatic to quality trim your reads. Specify the following, in this order:

LEADING: quality of 3

TRAILING: quality of 3

SLIDING WINDOW: window size of 5 and required quality of 15

MINLENGTH: 35 bases

Be sure to output compressed files and clear out any intermediate files.

See trimmomatic.sbatch for details

7. 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.

See QAA_hist.sbatch for details

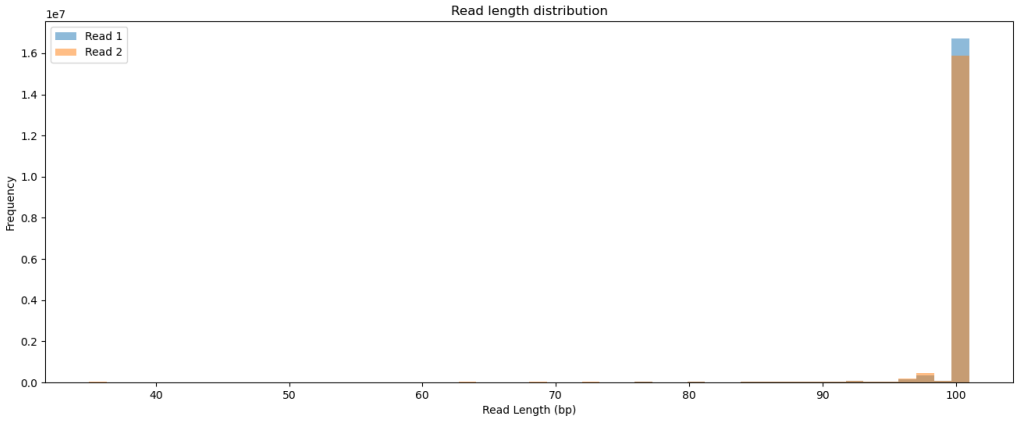

Figure 13: "11_2H_both_S9_L008 Post-Trimming Read Lengths"

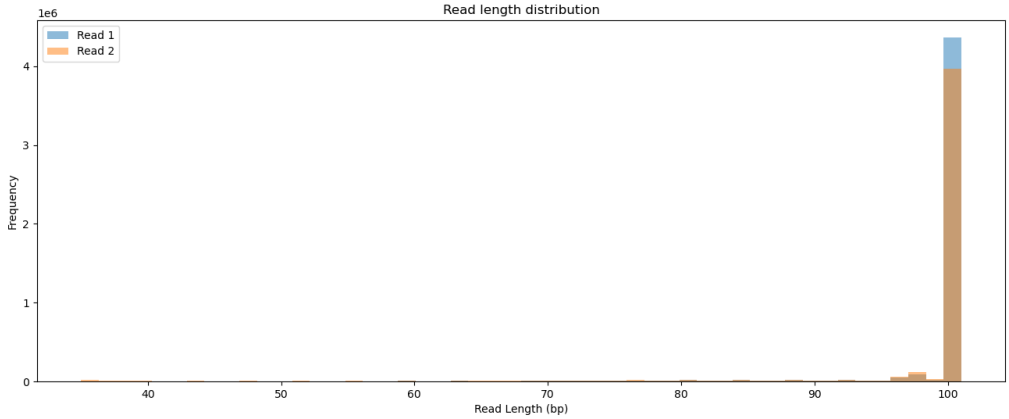

Figure 14: "29_4E_fox_S21_L008 Post-Trimming Read Lengths"

The trimming of both libraries yielded similar patterns in which Read 2 portion of the library was trimmed to a greater extent than the read 1 portion. This makes sense because in Illumina sequencing, the sequencing for read 2 occurs after all the other sequencing is complete and the DNA molecule is more heavily damaged. This causes more error in sequencing and lower quality reads that may be trimmed.

Part 3 – Alignment and strand-specificity

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.

Code used to download mouse assembly and gtf files:

wget http://ftp.ensembl.org/pub/release-104/fasta/mus_musculus/dna/Mus_musculus.GRCm39.dna.primary_assembly.fa.gz .

wget http://ftp.ensembl.org/pub/release-104/gtf/mus_musculus/Mus_musculus.GRCm39.104.gtf.gz .

See STAR_db.sbatch and STAR_align.sbatch for details

10. 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).

```
Output of mapped counting of 11_2H_both_S9_L008 sam file:
mapped count: 33637698
unmapped count: 1293608

Output of mapped counting of 29_4E_fox_S21_L008 sam file:
mapped count: 8883016
unmapped count: 260792
```

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.

See htseq_count.sbatch for details

12.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.").

RNA-seq library strandedness determination: When stranded=no in htseq-count, the program searches for mapping to a feature regardless if the read is on the same or opposite strand. When stranded=yes, paired-end reads need to have the first strand mapping to the same strand as the feature and the second strand needs to map to the opposite strand of the feature. When running htseq-count with the parameter stranded=no, a much larger proportion of reads map to features for both libraries (11_2H_both_S9_L008: 0.771992 stranded=no vs. 0.0333735 stranded=yes; 29_4E_fox_S21_L008: 0.820646 stranded=no vs. 0.0398086 stranded=yes). This informs me that the library was not stranded. If it were stranded, a higher proportion of reads would map to features for the htseq-count where stranded=yes.

Evidence: This command line will track two sums: the total sum of all not mapped to genes and the sum of all reads mapped to genes. It will then calculate the proportion of reads mapped to genes compared to all reads.

```
cat 11_2H_both_S9_L008_stranded_htseq_out.txt | \
awk '{if($1 !~ "^__+") {sum2 += $2} else {sum1 += $2}} END{print (sum2/(sum1+sum2))}'
0.0333735
cat 11_2H_both_S9_L008_nonstranded_htseq_out.txt | \
awk '{if($1 !~ "^__+") {sum2 += $2} else {sum1 += $2}} END{print (sum2/(sum1+sum2))}'
0.771992
cat 29_4E_fox_S21_L008_stranded_htseq_out.txt | \
awk '{if($1 !~ "^__+") {sum2 += $2} else {sum1 += $2}} END{print (sum2/(sum1+sum2))}'
0.0398086
cat 29_4E_fox_S21_L008_nonstranded_htseq_out.txt | \
awk '{if($1 !~ "^__+") {sum2 += $2} else {sum1 += $2}} END{print (sum2/(sum1+sum2))}'
0.820646
```