QAA report: Surbhi Nahata

Table of content:

Goal of the project:

The objectives of this project were to use existing tools for quality assessment and adaptor trimming, and then to compare the quality assessments to those from my own software.

The libraries examined in this project were (R1 and R2 reads in each library):

Part 1

- Following tasks were accomplished in this part:

A) Using FastQC software (s-andrews/FastQC: A quality control analysis tool for high throughput sequencing data (github.com))

1. I Produced plots of the per-base quality score distributions for R1 and R2 reads for each library.
2. I produce plots of the per-base N content and commented on their consistency with quality score plots

B) Using the quality score count software I wrote in a prior class, I generated the per-base quality score distribution plots again for the two reads in each library.

- FastQC summarizes the following information:

1. Basic Statistics
2. Per Base Sequence Quality
3. Per Tile Sequence Quality
4. Per Sequence Quality Scores
5. Per Base Sequence Content
6. Per Sequence GC Content
7. Per Base N Content
8. Sequence Length Distribution
9. Sequence Duplication levels
10. Overrepresented Sequences
11. Adapter Content
12. Kmer Content

- FastQC output plots and explanation on the graph:

1. 2_2B_control_S2_L008

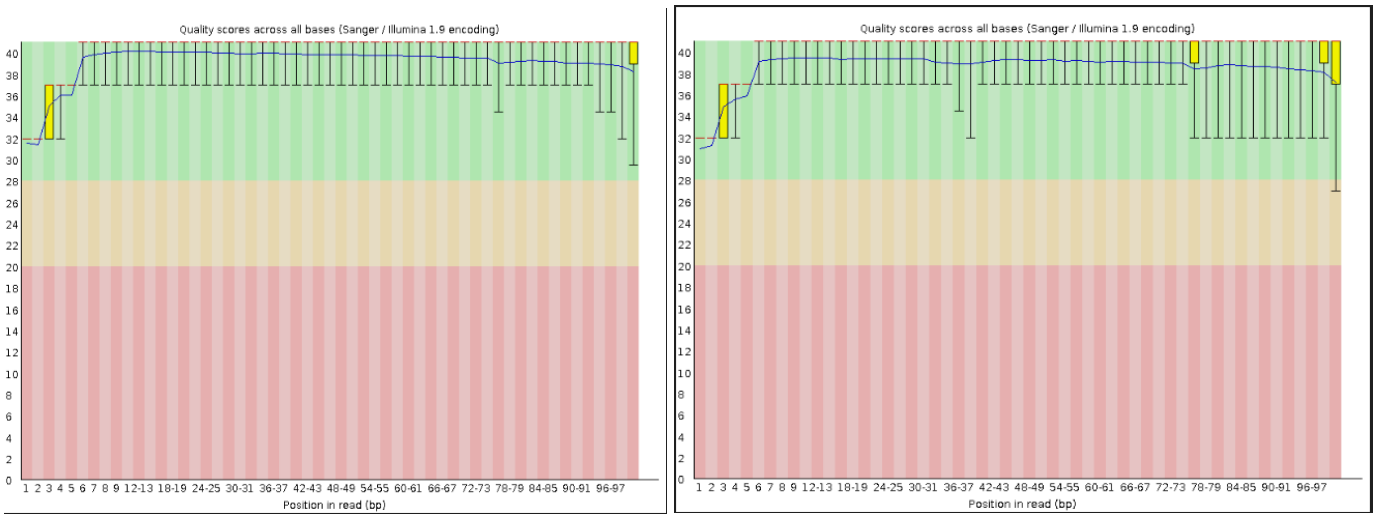

Figure 1: Overall quality score: a) 2_2B_control_S2_L008_R1_001_fastqc (left) b) 2_2B_control_S2_L008_R2_001_fastqc (right): Above is a quality score distribution of all the bases in a read for the library 2_2B_control_S2_L008_001. The x axis shows base per position and y-axis shows the quality scores. Blue line across the graph is the mean quality score and red line is the median. Yellow box is the inter-quartile range and top and bottom whiskers are 10% and 90% points.

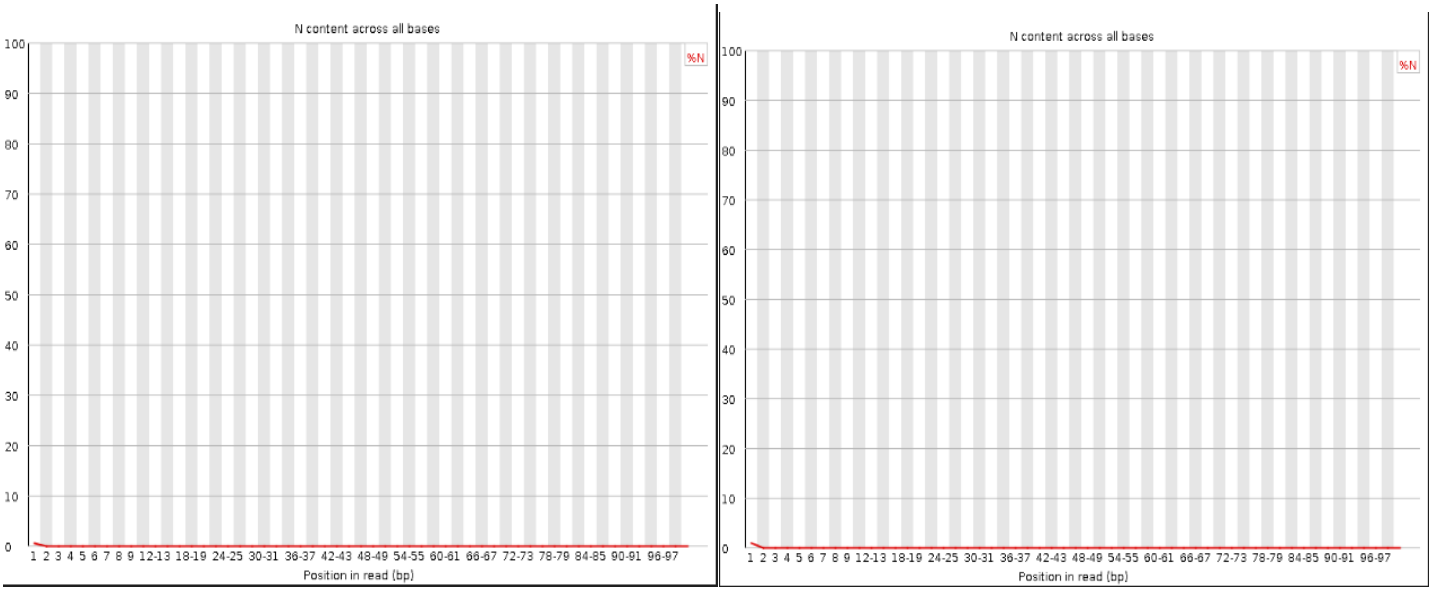

Figure 2: **Per base quality score : a) 2_2B_control_S2_L008_R1_001_fastqc (left) b) 2_2B_control_S2_L008_R2_001_fastqc (right)**: This plot shows per base N content across all base positions. The red line is the N content for each base.

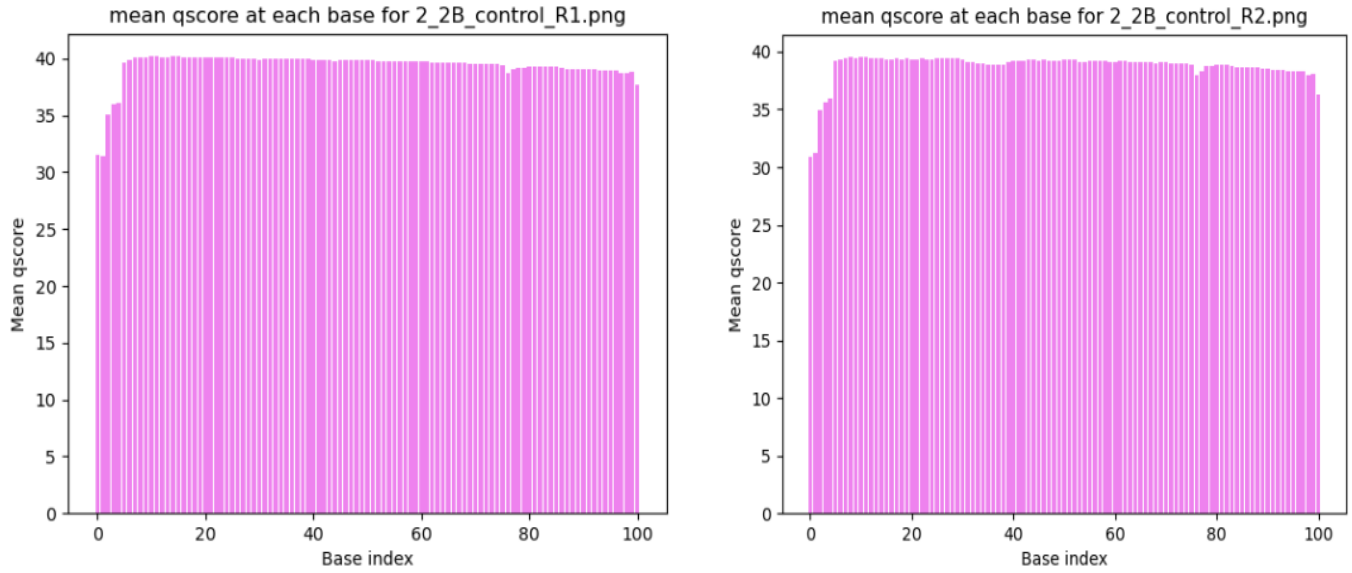

Figure 3: Quality score distribution through my software: a) 2_2B_control_S2_L008_R1_001_fastqc (left) b) 2_2B_control_S2_L008_R2_001_fastqc (right): Quality score of each base has been plotted on the x axis is the base position and on the y axis is the mean qscore at each base.

2. Undetermined_S0_L008

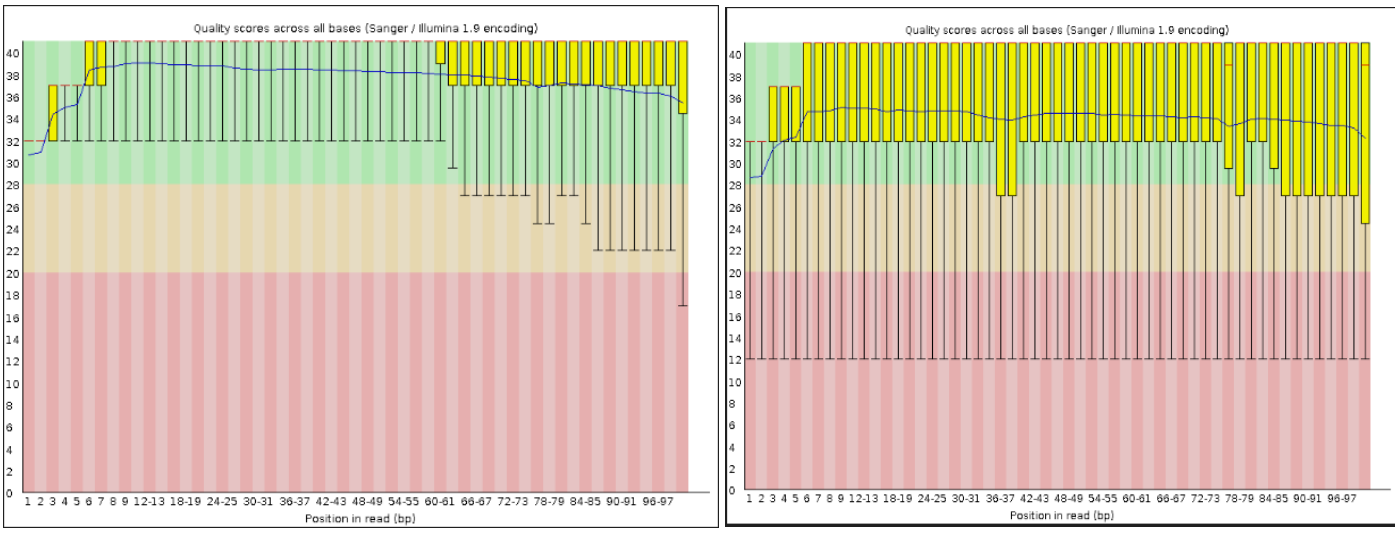

Figure 4: Overall quality score: a) Undetermined_S0_L008_R1_001_fastqc (left) b) Undetermined_S0_L008_R2_001_fastqc (right): Above is a quality score distribution of all the bases in a read for the library 2_2B_control_S2_L008_001. The x axis shows base per position and y-axis shows the quality scores. Blue line across the graph is the mean quality score and red line is the median. Yellow box is the inter-quartile range and top and bottom whiskers are 10% and 90% points.

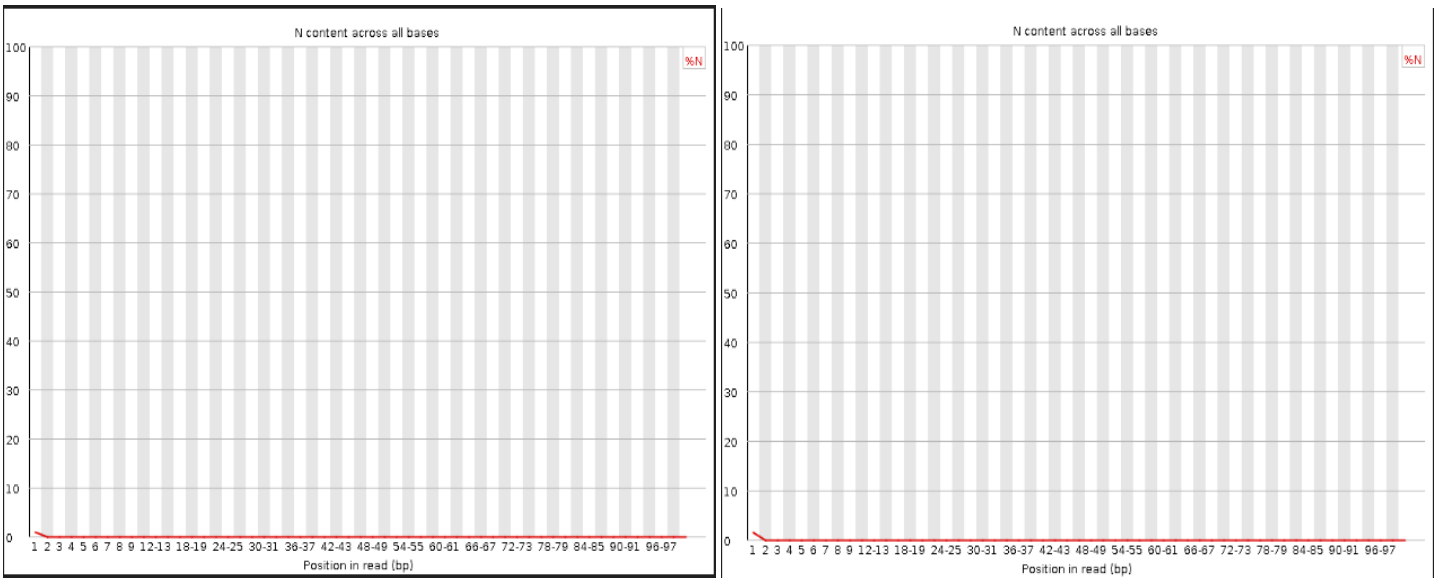

Figure 5: Per N base quality score: a) Undetermined_S0_L008_R1_001_fastqc (left) b) Undetermined_S0_L008_R2_001_fastqc (right)

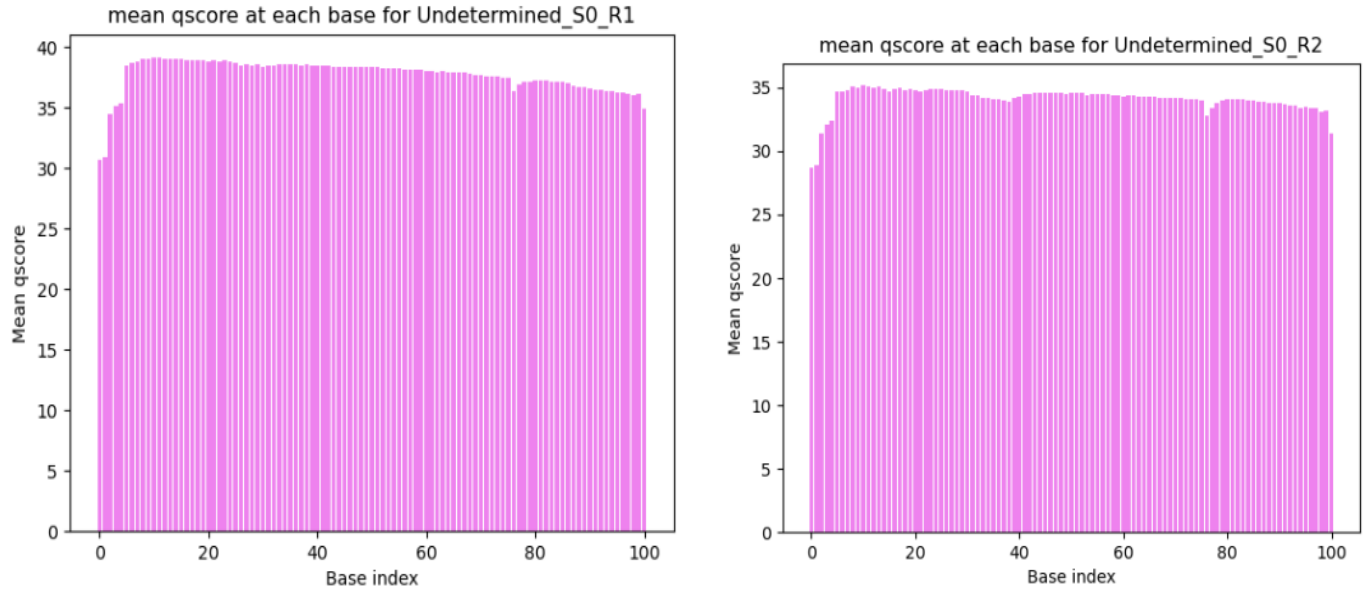

Figure 6:Quality score distribution through my software a) Undetermined_S0_L008_R1_001_fastqc (left) b) Undetermined_S0_L008_R2_001_fastqc (right)

Opinion and interpretation of the plots above:

a) Comment on whether or not N based score are consistent with the quality score plots?
Figure 1 and figure 2 show example of really great quality score. They both have been consistent with each other. Both read one and read two have great quality scores with quality cutoff of 30 which is the Illumina standard. The N based score provide further evidence of good quality score. However, figure 3 and 4 are not very consistent, it looks like the overall average quality score was terrible with 10% of reads having quality score of less than 15. However, looking at N-base score it does look like an example of good read, however, I won't trust the data presented. It could just mean the illumina machine worked fine and something might have gone wrong with the library prep (especially for read 2 where q-score doesn't look very promising).

b) Also, does the runtime differ?
The run time between my software and FastQC differed a lot. FastQC was much more faster than my software and I believe that is the case since FASTQC has been developed to perform the task of looking at quality score faster than my own software.

c) Comment on the overall data quality of your two libraries. Go beyond per-base qscore distributions. Make and justify a recommendation on whether or these data are of high enough quality to use for further analysis?
When I look at basic statistic from FastQC, it looks like my library 1 (i.e. 2_2B_control_S2_L008) is of very high quality as most of the q-score falls above 30. However, my library 2(Undertermined) were not of high quality as it can be seen that 10% of my read fell below the q-score of 10. I would sadly have to re-sequence my library because of low quality reads. Also when I looked at per tile plot, my library one had mostly blue tiles which means it was high quality, however, library 2 had lots of red tiles which means the reading in that particular tile was not great and it makes sense given the quality score graph.

Part 2: Adapter trimming comparison

Cut-adapt results:

For 2_2B_control.fastq.gz:

Finished in 352.78 s (61 us/read; 0.99 M reads/minute)

=== Summary ===

Total read pairs processed: 5,830,665

Read 1 with adapter: 423,128 (7.3%)

Read 2 with adapter: 473,368 (8.1%)

For Undetermined_S0_L008.fastq.gz:

Finished in 1012.48 s (69 us/read; 0.87 M reads/minute).

=== Summary ===

Total read pairs processed: 14,760,166

Read 1 with adapter: 543,021 (3.7%)

Read 2 with adapter: 607,660 (4.1%)

Trimmomatic results:

Unfortunately, due to misread instructions, I was unable to create text files which would have further helped create distribution graphs. Rerunning the command would have taken time, hence I am attaching the results after the trimmomatic was ran successfully. Based on the results below, it looks like read one had more adapter sequences that were removed through the software, but read two had negligible amount of adaptors removed for both the libraries.

```
trimmomatic PE -threads 8
/projects/bgmp/surbhin/bioinfo/Bi623/QAA/v1_2_2B_control_R1.fastq.gz
/projects/bgmp/surbhin/bioinfo/Bi623/QAA/v1_2_2B_control_R2.fastq.gz paired.V1_2_2B_cont
rol_R1.fastq.gz unpaired.V1_2_2B_control_R1.fastq.gz paired.V1_2_2B_control_R2.fastq.gz
unpaired.V1_2_2B_control_R2.fastq.gz LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35
Quality encoding detected as phred33
```

```
Input Read Pairs: 5830665 Both Surviving: 5652541 (96.95%) Forward Only Surviving: 133562
(2.29%) Reverse Only Surviving: 4595 (0.08%) Dropped: 39967 (0.69%)
TrimmomaticPE: Completed successfully

trimmomatic PE /projects/bgmp/surbhin/bioinfo/Bi623/QAA/Undetermined_S0_L008_R1.fastq.gz
/projects/bgmp/surbhin/bioinfo/Bi623/QAA/Undetermined_S0_L008_R2.fastq.gz
pairedundertermined_S0_L008_R1.fastq.gz unpaired.Undetermined_S0_L008_R1.fastq.gz
paired.Undetermined_S0_L008_R2.fastq.gz unpaired.Undetermined_S0_L008_R2.fastq.gz.fastq.gz
LEADING:3 TRAILING:3 SLIDINGWINDOW:5:
15 MINLEN:35
Input Read Pairs: 14760166 Both Surviving: 12160073 (82.38%) Forward Only Surviving: 2511278
(17.01%) Reverse Only Surviving: 31172 (0.21%) Dropped: 57643 (0.39%)
TrimmomaticPE: Completed successfully
```

Part 3: Goal of this section is to look at mapped genes to find the stranded-ness of the sample and to figure out the template strand for the sequencing read

STAR mapping output:

The amount of mapped and unmapped reads can be seen in the figure below. The alignmentfile2_2BAligned.out.sam is from the 2_2B_control_S2_L008 trimmomatic read vs the mouse gtf file (Mus_musculus.GRCm39.110.gtf). The second mapped vs unmapped results with the alignmentfileAligned.out.sam file is the mapping with Undetermined_S0_L008 and Mus_musculus.GRCm39.110.gtf

	Mapped reads	Unmapped reads
Alignmentfile2_2BAligned.out.sam	11078806	226276
AlignmentfileAligned.out.sam	15584495	8735651

Htseq-count output:

Based on the result I got from the htseq-count software, I propose that the sequencing data is stranded and the reverse strand was used as a template for sequencing. If the read was unstranded, both the reverse and strand “yes” option would have shown 50% of mapping. However, that was not the case based on no feature/ambiguous count (results can be seen in the TSV file). Below is a short summary of the no_feature, mapped and counts for the genes. In both the cases (i.e. 2_2B_control_S2_L008 and undertermined_S0_L008) it looks like the stranded options had more “no_feature” count which tells us that reverse strand was used as a template for sequencing as it has less no mapped genes and more of the mapped genes.

```
QAA1) [surbhin@n0349 QAA]$ tail 2_2breverse.tsv
ENSMUSG00002076988      0
ENSMUSG00002076989      0
ENSMUSG00002076990      0
ENSMUSG00002076991      0
ENSMUSG00002076992      0
__no_feature    293950
__ambiguous     89681
__too_low_aQual 4760
__not_aligned   110518
__alignment_not_unique  347951
(QAA1) [surbhin@n0349 QAA]$ tail 2_2bstranded.tsv
ENSMUSG00002076988      0
```

```
ENSMUSG00002076989      0
ENSMUSG00002076990      0
ENSMUSG00002076991      0
ENSMUSG00002076992      0
__no_feature    4960672
__ambiguous     8445
__too_low_aQual 4760
__not_aligned   110518
__alignment_not_unique  347951
(QAA1) [surbhin@n0349 QAA]$ tail undertermined_stranded.tsv
ENSMUSG00002076988      0
ENSMUSG00002076989      0
ENSMUSG00002076990      0
ENSMUSG00002076991      0
ENSMUSG00002076992      0
__no_feature    7064396
__ambiguous     5958
__too_low_aQual 79476
__not_aligned   4325481
__alignment_not_unique  378431
(QAA1) [surbhin@n0349 QAA]$ tail undertermined_reverse.tsv
ENSMUSG00002076988      0
ENSMUSG00002076989      0
ENSMUSG00002076990      0
ENSMUSG00002076991      0
ENSMUSG00002076992      0
__no_feature    630007
__ambiguous     129419
__too_low_aQual 79476
__not_aligned   4325481
__alignment_not_unique  378431
```

Commands on Talapas to receive desired output (just giving one example for one library read but in most cases I have run the command multiple times based on file I used):

Part 1 Commands:

fastqc -o /projects/bgmp/surbhin/bioinfo/Bi623/QAA/fastqc_output/
/projects/bgmp/shared/2017_sequencing/demultiplexed/Undetermined_S0_L008_R2_001.fastq.gz

Part 2 Commands:

conda create -n QAA1

----- conda install -c bioconda cutadapt (cutadapt --version 2.6)

----- conda install -c bioconda trimmomatic (trimmomatic -version 0.39)

cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

-o v1_2_2B_control_R1.fastq.gz -p v1_2_2B_control_R2.fastq.gz

/projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R1_001.fastq.gz

/projects/bgmp/shared/2017_sequencing/demultiplexed/2_2B_control_S2_L008_R2_001.fastq.gz > outputcutadapt

trimmomatic PE /projects/bgmp/surbhin/bioinfo/Bi623/QAA/Undetermined_S0_L008_R1.fastq.gz /projects/bgmp/surbhin/bioinfo/Bi623/QAA/Undetermined_S0_L008_R2.fastq.gz pairedundertermined_S0_L008_R1.fastq.gz unpaired.Undetermined_S0_L008_R1.fastq.gz paired.Undetermined_S0_L008_R2.fastq.gz unpaired.Undetermined_S0_L008_R2.fastq.gz.fastq.gz LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35

Part 3 Commands:

conda install htseq

conda install star

conda install matplotlib

conda install -c anaconda numpy

wget https://ftp.ensembl.org/pub/release-110/fasta/mus_musculus/dna/Mus_musculus.GRCm39.dna.primary_assembly.fa.gz

wget https://ftp.ensembl.org/pub/release-110/gtf/mus_musculus/Mus_musculus.GRCm39.110.gtf.gz

./mapped_vs_unmapped.py -f alignmentfile2_2BAligned.out.sam

htseq-count -c 2_2bstranded.tsv --stranded=yes alignmentfile2_2BAligned.out.sam Mus_musculus.GRCm39.110.gtf

htseq-count -c 2_2breverse.tsv --stranded=reverse alignmentfile2_2BAligned.out.sam Mus_musculus.GRCm39.110.gtf

tail undertermined_reverse.tsv

tail undertermined_stranded.tsv