RNA-Seq Quality Assessment

Bea Meluch

2022-09-08

Objectives

- Use existing tools for quality assessment and adaptor trimming
- Compare assessments between existing tools and own software
- Summarize other important information about RNA-seq data

Data

All work except for R markdown was done on Talapas in `/projects/bgmp/bmeluch/bioinfo/Bi623/QAA` folder.

My assigned files:

14_3B_control_S10_L008
8_2F_fox_S7_L008

Paths:

/projects/bgmp/shared/2017_sequencing/demultiplexed/8_2F_fox_S7_L008_R1_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/8_2F_fox_S7_L008_R2_001.fastq.gz

/projects/bgmp/shared/2017_sequencing/demultiplexed/14_3B_control_S10_L008_R1_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/14_3B_control_S10_L008_R2_001.fastq.gz

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.

The quality plots show lower quality for the first 6 bases or so, though 'lower' is relative - the median scores are still in the green range on the plot. The per-base N content plots show a tiny percentage of Ns at the beginning of the reads and almost no Ns throughout the rest of the read. These plots complement each other. Ns are called when a base is low quality, so Ns are expected more in the beginning of these reads than throughout the rest of the lengths. Read 2 appears to be lower quality in both samples.

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

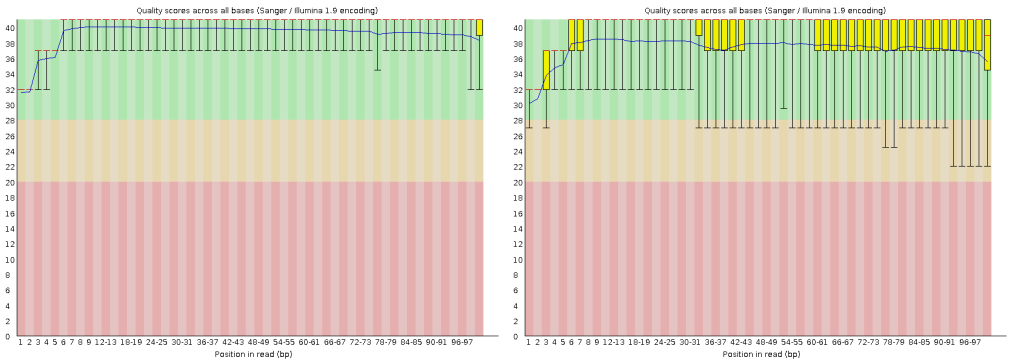

Figure 1: Sample 7: Read quality score distributions (Read 1, Read 2)

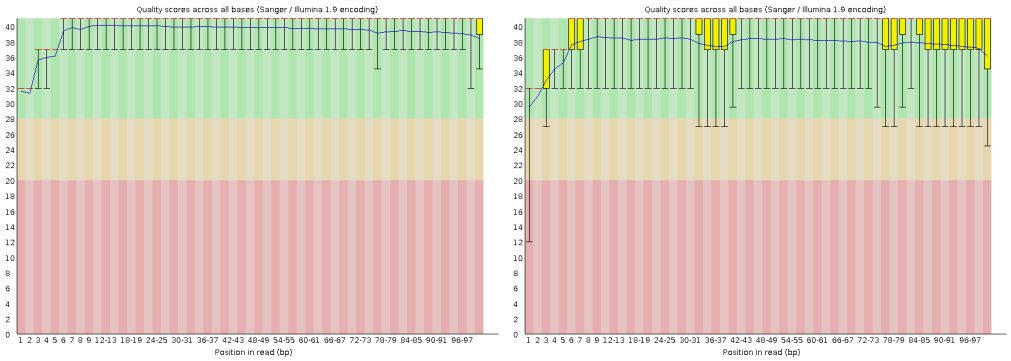

Figure 2: Sample 10: Read quality score distributions (Read 1, Read 2)

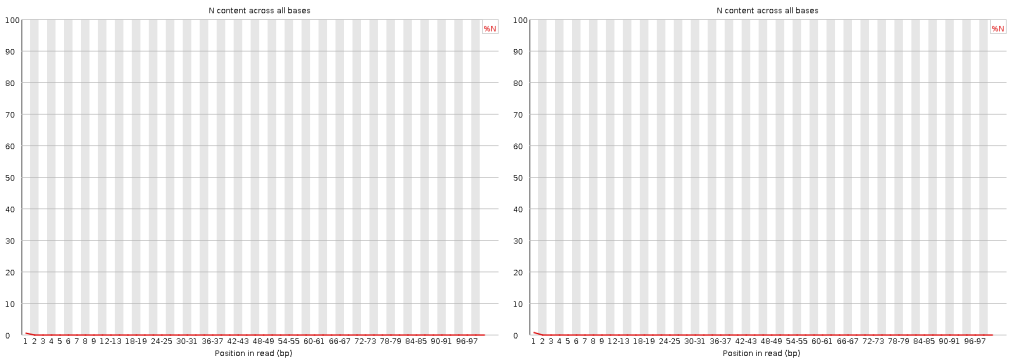

Figure 3: Sample 7: Per-base N content (Read 1, Read 2)

To produce my own plots, I used `base_distribution.py` from Demultiplex Assignment 1. My histograms closely match the shape of the FastQC quality plots. They show the same trends and dips across the length of the reads.

```
/projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part1/slurm/qual_hist.srun

/usr/bin/time -v /projects/bgmp/bmeluch/bioinfo/Bi622/Demultiplex/
  Assignment-the-first/base_distribution.py \
-f <file path> \
-o /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part1/my_plots/<file prefix>_qual_hist.png
```

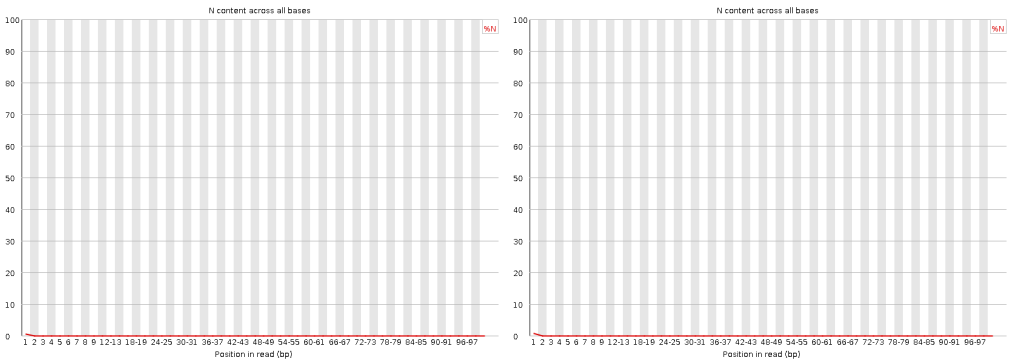

Figure 4: Sample 10: Per-base N content (Read 1, Read 2)

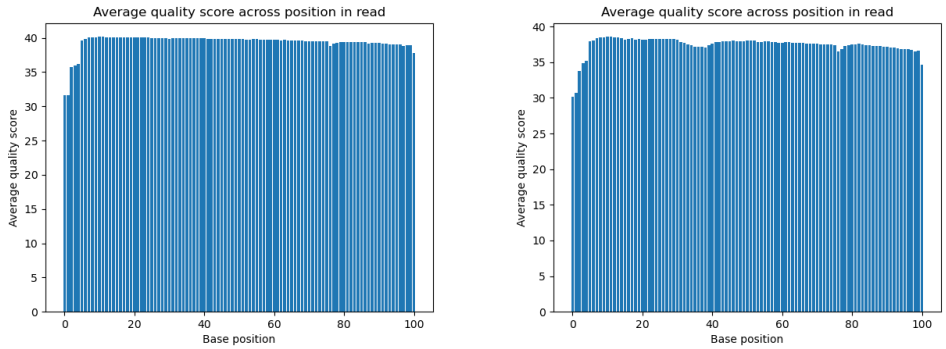

Figure 5: Sample 7: Read quality score distribution (Read1, Read 2) using Demux script

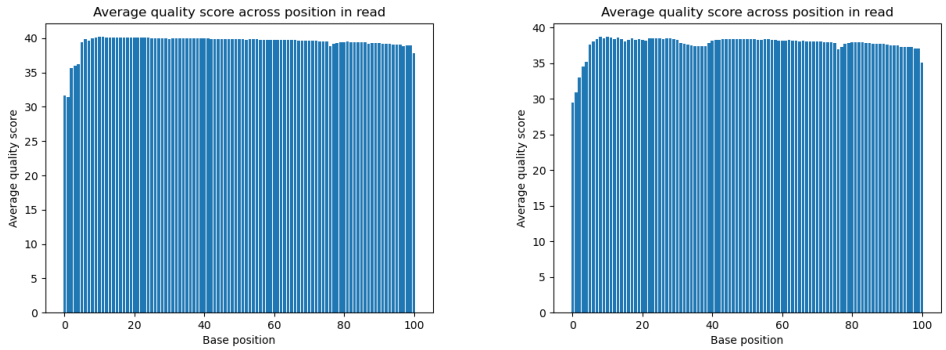

Figure 6: Sample 10: Read quality score distribution (Read1, Read 2) using Demux script

The time required for each of the four plots was obtained from `/usr/bin/time -v` commands in `qual_hist.srun`.

```
/projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part1/slurm/qual_hist.srun
8_2F_fox_S7_L008_R1_001
Elapsed (wall clock) time (h:mm:ss or m:ss): 13:39.74
8_2F_fox_S7_L008_R2_001
Elapsed (wall clock) time (h:mm:ss or m:ss): 13:47.79
14_3B_control_S10_L008_R1_001
```

```
Elapsed (wall clock) time (h:mm:ss or m:ss): 1:43.22
14_3B_control_S10_L008_R2_001
Elapsed (wall clock) time (h:mm:ss or m:ss): 1:40.57
```

The runtime for Sample 7 files was much longer than the runtime for Sample 10 files. The files for Sample 7 are much bigger:

```
(base) [bmeluch@talapas-ln1 bgmp]$ zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
8_2F_fox_S7_L008_R1_001.fastq.gz | wc -l
145930404
(base) [bmeluch@talapas-ln1 bgmp]$ zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
14_3B_control_S10_L008_R1_001.fastq.gz | wc -l
17761512
```

Dividing the line count by 4 gives the number of reads in the file. Sample 7 has 36,482,601 reads, while Sample 10 has 4,440,378 reads.

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

The two libraries are overall very high quality, as shown on the FastQC quality plots. Median read quality stays within the green zone for the entire length of the read. All reads are low quality for the first few bases, then jump up to the 38-40 range, then slowly decline towards the end. Read 2 is overall lower quality than Read 1 for both libraries (overall still high but with more variability into the lower region).

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. Recommend doing this in an interactive session, not the login node! Make sure you check your installations with:
    - `cutadapt --version` (should be 4.1)
    - `trimmomatic -version` (should be 0.39)

```
(bgmp_py310) [bmeluch@talapas-ln1 fastqc_out]$ conda create -n QAA python=3.10.5
(bgmp_py310) [bmeluch@talapas-ln1 fastqc_out]$ conda activate QAA
(QAA) [bmeluch@talapas-ln1 QAA]$ conda install cutadapt
(QAA) [bmeluch@talapas-ln1 QAA]$ conda install Trimmomatic
(QAA) [bmeluch@talapas-ln1 QAA]$ cutadapt --version
4.1
(QAA) [bmeluch@talapas-ln1 QAA]$ trimmomatic -version
0.39
```

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 R1 and R2) 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.

To determine the adapter sequence to search for, I looked at the FastQC "Adapter Content" plots to see which adapter it thinks it found. All of the plots show a red line for "Illumina Universal Adapter" content near the end of the read. According to the FastQC documentation (https://github.com/golharam/FastQC/blob/master/Configuration/adapter_list.txt), FastQC identifies the sequence "AGATCGGAAGAG" as "Illumina Universal Adapter". Illumina's website (https://support.illumina.com/bulletins/2016/12/what-sequences-do-i-use-for-adapter-trimming.html) has many adapter sequences to use with trimming software. The first two begin with "AGATCGGAAGAG", so that is what I used in my cutadapt commands.

To check that this sequence was present near the ends of the reads, I used `awk` to search for the locations of the sequence:

```
(base) [bmeluch@talapas-ln1 QAA]$ zcat /projects/bgmp/shared/2017_sequencing/demultiplexed/
8_2F_fox_S7_L008_R1_001.fastq.gz |
awk '/AGATCGGAAGAG/ {print index($0, "AGATCGGAAGAG" )}' | head -20
74
79
78
78
52
79
79
72
48
87
82
75
87
90
84
59
89
88
62
84
```

I repeated the command for all four input files. The output showed that when the adapter was located, its first base was generally at index 70 or higher (with some exceptions). Adapter content is expected to increase along the length of the reads, and indeed this is what is shown in the plot, so it makes sense that adapter start indices would be in the last 30% of the read length.

Since I found the sequence "AGATCGGAAGAG", I proceeded to run cutadapt. I feel it is important to note that cutadapt has the cutest ASCII scissors progress animation.

```
(QAA) [bmeluch@n278 QAA]$ cutadapt -a AGATCGGAAGAG -A AGATCGGAAGAG
-o <cutadapt output read 1>
-p <cutadapt output read 2>
<input fastq from shared folder read 1>
<input fastq from shared folder read 2>
```

I copied the terminal output into text files `/QAA/part2/14_3B_control_S10_cutadapt_output.txt` and `/QAA/part2/8_2F_fox_S7_cutadapt_output.txt`.

Cutadapt reported that 5.9% and 6.7% of reads were trimmed for Read 1 and Read 2 of Sample 7, respectively. Similarly, 6.0% and 6.8% of reads were trimmed for Read 1 and Read 2 of Sample 10, respectively. In both cases slightly more reads had adapter content in Read 2 than the corresponding Read 1.

```
Sample 7
Total read pairs processed:         36,482,601
  Read 1 with adapter:               2,156,351 (5.9%)
  Read 2 with adapter:               2,426,159 (6.7%)
Pairs written (passing filters):    36,482,601 (100.0%)


Sample 10
Total read pairs processed:          4,440,378
  Read 1 with adapter:                 265,268 (6.0%)
  Read 2 with adapter:                 302,372 (6.8%)
Pairs written (passing filters):     4,440,378 (100.0%)
```

6. Use `Trimmomatic` to quality trim your reads.

Trimmomatic was run on the trimmed files, using paired-end mode with the specified options:

```
/usr/bin/time -v trimmomatic PE \
    cutadapt_<sample>_R1_001.fastq.gz \
    cutadapt_<sample>_R2_001.fastq.gz \
    -baseout trimmomatic_<sample>.fastq.gz \
    LEADING:3 TRAILING:3 SLIDINGWINDOW:5:15 MINLEN:35
```

Trimmomatic reported that ~95% of reads for both samples came back with both reads in the pair passing quality trimming.

```
Sample 7

Input Read Pairs: 36482601
Both Surviving: 34780223 (95.33%)
Forward Only Surviving: 1638786 (4.49%)
Reverse Only Surviving: 31878 (0.09%)
Dropped: 31714 (0.09%)

Sample 10

Input Read Pairs: 4440378
Both Surviving: 4245444 (95.61%)
Forward Only Surviving: 183589 (4.13%)
Reverse Only Surviving: 3781 (0.09%)
Dropped: 7564 (0.17%)
```

7. Plot the trimmed read length distributions for both R1 and R2 reads (on the same plot). Comment on whether you expect R1s and R2s to be adapter-trimmed at different rates.

I expect Read 2 to be adapter-trimmed at a higher rate, since Cutadapt identified more adapter content in Read 2. The plots show that Read 2 has more reads at shorter lengths than Read 1. Read 1 adapter content appears to increase steadily from ~35 bp onwards, but Read 2 adapter content jumps sharply and stays higher.

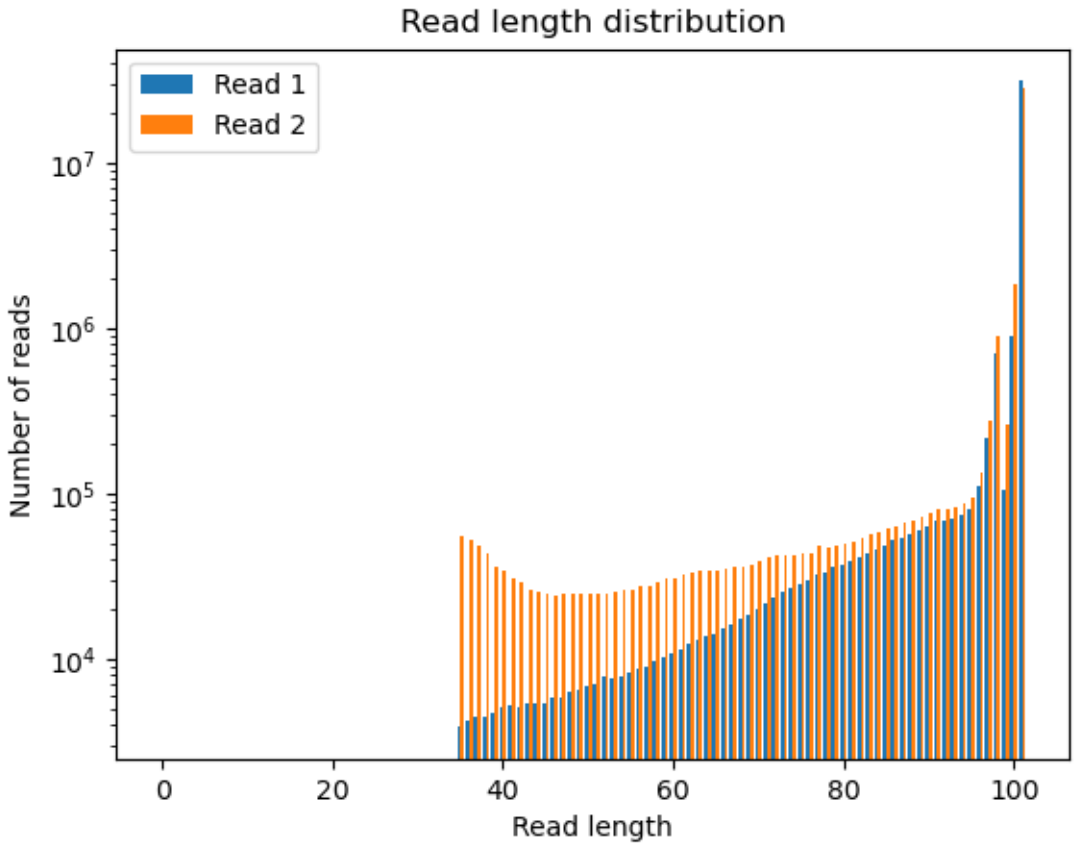

Figure 7: Read length distribution for Sample 7

Part 3 – Alignment and strand-specificity

8. Install sofware. In your QAA environment, use conda to install:
    - star
    - numpy
    - pysam
    - matplotlib
    - htseq

```
(QAA) [bmeluch@n278 slurm]$ conda install star -c bioconda
(QAA) [bmeluch@n278 slurm]$ STAR --version
2.7.10a
(QAA) [bmeluch@n278 slurm]$ conda install numpy
(QAA) [bmeluch@n278 slurm]$ conda install pysam
>>> print(pysam.__version__)
0.19.1
(QAA) [bmeluch@n278 QAA]$ conda install matplotlib
(QAA) [bmeluch@n278 QAA]$ conda install htseq -c bioconda
Downloading and Extracting Packages
htseq-2.0.2
```

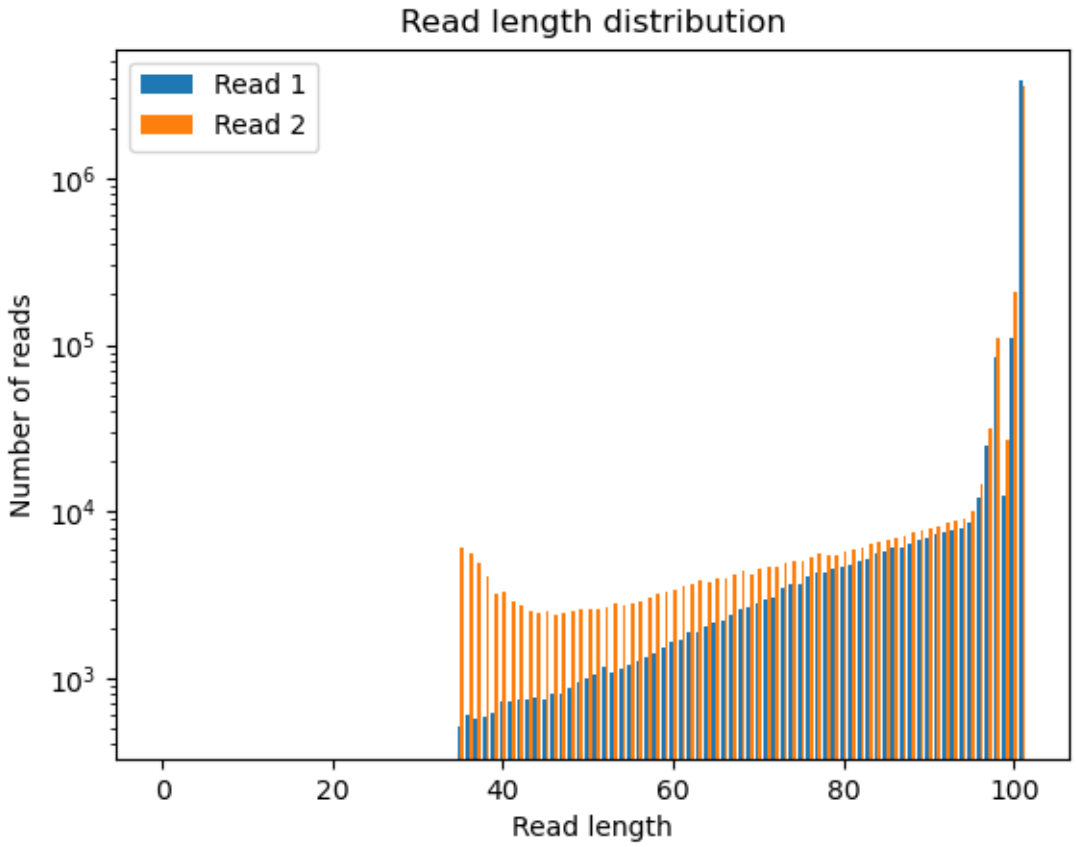

Figure 8: Read length distribution for Sample 10

8. Find publicly available mouse genome fasta files (Ensemble release 107) 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.

   Hint - you will need to use gene models to perform splice-aware alignment, see PS8 from Bi621.

Mouse genome files were downloaded to Talapas using FTP from the following Ensembl sources. Files were unzipped using gzip.

```
(QAA) [bmeluch@n278 part3]$ wget http://ftp.ensembl.org/pub/release-107/gtf/
  mus_musculus/Mus_musculus.GRCm39.107.gtf.gz
(QAA) [bmeluch@n278 part3]$ wget http://ftp.ensembl.org/pub/release-107/fasta/
  mus_musculus/dna/Mus_musculus.GRCm39.dna.primary_assembly.fa.gz
```

STAR database generation and alignment was performed according to the instructions in PS8.

STAR database generation

```
/projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/slurm/star_database_generation.srun

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

```
--genomeDir /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/
  Mus_musculus.GRCm39.107.STAR_2.7.1a \
--genomeFastaFiles /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/
  Mus_musculus.GRCm39.dna.primary_assembly.fa \
--sjdbGTFfile /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/
  Mus_musculus.GRCm39.107.gtf

Elapsed (wall clock) time (h:mm:ss or m:ss): 17:45.66
```

STAR alignment

```
/projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/slurm/star_align.srun

/usr/bin/time -v STAR --runThreadN 8 --runMode alignReads \
--outFilterMultimapNmax 3 \
--outSAMunmapped Within KeepPairs \
--alignIntronMax 1000000 --alignMatesGapMax 1000000 \
--readFilesCommand zcat \
--readFilesIn <trimmomatic output read 1>.fastq.gz \
<trimmomatic output read 2>.fastq.gz \
--genomeDir /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/
  Mus_musculus.GRCm39.107.STAR_2.7.1a \
--outFileNamePrefix
  /projects/bgmp/bmeluch/bioinfo/Bi623/QAA/part3/aligned/<sample>


8_2F_fox_S7: Elapsed (wall clock) time (h:mm:ss or m:ss): 4:11.52
14_3B_control_S10: Elapsed (wall clock) time (h:mm:ss or m:ss): 0:36.88
```

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

My modified script from PS8 reported that Sample 7 had 33556914 mapped reads and 1289546 unmapped reads. Sample 10 had 4157957 mapped reads and 94209 unmapped reads.

10. Count reads that map to features using `htseq-count`. You should run htseq-count twice: once with `--stranded=yes` and again with `--stranded=reverse`. Use default parameters otherwise.

	S7_Yes	S7_Rev	S10_Yes	S10_Rev
no_feature	30597190	3347799	3798204	231033
ambiguous	26264	528115	3337	72917
too_low_aQual	61680	61680	6303	6303
not_aligned	1223309	1223309	87487	87487
alignment_not_unique	1590113	1590113	182321	182321

11. Are the data from strand-specific RNA-seq libraries?

These data are from stranded library preps. If the library was unstranded, then htseq-count would report an equal number of features when using the options `--stranded=yes` and `--stranded=reverse`. An unstranded library would include roughly 50% of insert molecules on the + strand and 50% on the - strand for a given

read. This would result in reads mapping to features in the "yes" setting as well as in the "reverse" setting. According to the htseq output shown above, 33498556 reads did not successfully map to features in the "yes" analysis of Sample 7. Only 6751016 reads did not successfully map to features when the "reverse" option was used. This means that roughly five times (4.96) as many reads failed to map in one direction as compared to the other. Similarly, 7 times as many reads failed to map in the "yes" as compared to the "reverse" analysis for Sample 10. If strandedness matters in this analysis, then the library was prepared using a stranded method.