Assignment 2 - QAA (FastQC) Lab Notebook

General Notes

Upload report both to GitHub and Canvas. Check that everything got uploaded.
Answers should be in the final high level report pdf.
It should be written in R markdown.
THE FILE SHOULD BE KNITTED WITH LATEX
Read 2 is lower quality because it is the last to come off the sequencer so it has had more time to degrade.
Adapter content increases toward the end of a sequence because if you have short inserts, adapters may overlap.
If you see adapter sequences, you must trim them.

Data can be found at:
`/projects/bgmp/shared/2017_sequencing/demultiplexed/`
Library assignments are found here:
`/projects/bgmp/shared/Bi623/QAA_data_assignments.txt`

Each of us are assigned two demultiplexed files to work with.
My assignment:

```
Fileset 1                Fileset 2
32_4G_both_S23_L008     3_2B_control_S3_L008
```

Objectives:

The objectives of this assignment are to use existing tools for quality assessment and adaptor trimming, compare the quality assessments to those from your own software, and to demonstrate your ability to summarize other important information about this RNA-Seq data set in a high-level report. That is, you should create a cohesive, well written report for your "PI" about what you've learned about/from your data.

Data Exploration

1. Number of lines (# of records)
2. Are our data good enough to proceed?
    1. QC Ideas:

1. Graph of per base quality distribution
2. Per base N content
3. Per base GC
4. Length distribution
5. Adapter content
6. Overrepresented sequences
7. Encoding i.e. Phred+33

Will use FastQC for quality control. Not making our own program.

Procedures

Part 1

1. Created a new GitHub repo called QAA
2. Created a new environment called QAA
    1. `conda create –n QAA`
3. Installed fastqc inside QAA environment
    1. `conda install fastqc`
4. To activate the QAA environment:
    1. `conda activate QAA`
5. Cloned the QAA repo onto Bi623 directory on Talapas
6. Made a fastqc output directory called FASTQC_OUTPUT
7. General Syntax for fastqc command:
    1. `fastqc –o <name_of_output_directory> <name_of_input_file>`
    2. In this example, the -o flag is used to indicate that the next group of characters following the space represent the name of the output directory (FASTQC_OUTPUT), immediately followed by the name/location of the input file
8. Command to run:

```
fastqc –o ~/bgmp/bioinfo/Bi623/QAA/FASTQC_OUTPUT/ –t 4
32_4G_both_S23_L008_R1_001.fastq.gz 32_4G_both_S23_L008_R2_001.fastq.gz
3_2B_control_S3_L008_R1_001.fastq.gz 3_2B_control_S3_L008_R2_001.fastq.gz
```

This multithreads the fastqc command so all four files can be run at once.
Used srun with mem = 30GB and -c 6
Although, with only four files processed, only four cores were needed.
Took ~2 minutes

Running Demultiplexing quality score distribution script:

```
sbatch sbatch_qual_dist
```

slurm-16029502.out

Downloaded the plots to desktop via git:

```
~/bioinfo/Bi623/QAA/my_qual_score_distributions/32_4G_both_S23_L008_R1_001_d
istribution.png
~/bioinfo/Bi623/QAA/my_qual_score_distributions/32_4G_both_S23_L008_R2_001_d
istribution.png
~/bioinfo/Bi623/QAA/my_qual_score_distributions/3_2B_control_S3_L008_R1_001.
png
~/bioinfo/Bi623/QAA/my_qual_score_distributions/3_2B_control_S3_L008_R2_001.
png
```

Part 2

The adapters:
Index 1 (i7) from Illumina TruSeq universal adapter online
(A)GATCGGAAGAGCACACGTCTGAACTCCAGTCAC
Index 1 from Leslie:
AGATCGGAAGAGCACACGTCTGAACTCCAGTCA
Index 2:
ACACTCTTTCCCTACACGACGCTCTTCCGATCT
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT

Using Cutadapt to trim the adapters:

```
cutadapt –a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA –o
sans_adapter/sans_adapt_32_4G_both_S23_L008_R1_001.fasta.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/32_4G_both_S23_L008_R1_0
01.fastq.gz
```

```
cutadapt –a AACCGGTT –o output.fastq input.fastq
```

Inside QAA/sans_adapter/

```
cutadapt -j 12 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o
sans_adapt_3_2B_control_S3_L008_R1_001.fastq.gz -p
sans_adapt_3_2B_control_S3_L008_R2_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/3_2B_control_S3_L008_R1_
001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/3_2B_control_S3_L008_R2_
001.fastq.gz
```

```
cutadapt -j 12 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A
AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -o
sans_adapt_32_4G_both_S23_L008_R1_001.fastq.gz -p
sans_adapt_32_4G_both_S23_L008_R2_001.fasta.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/32_4G_both_S23_L008_R1_0
01.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/32_4G_both_S23_L008_R2_0
01.fastq.gz
```

All files are between 5 and 9 MB bigger after cutadapt adapter trimming. VERY weird. Could not find anything online about this. This appears to be extremely unusual. Ran it without multithreading and without paired end option and all returned the same result.

3_2B_R1

```
cutadapt -j 12 -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -o
sans_adapt_3_2B_control_S3_L008_R1_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/3_2B_control_S3_L008_R1_
001.fastq.gz
```

`ls -lhart /projects/bgmp/shared/2017_sequencing/demultiplexed/ | grep 3_2B_control`

```
-A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT
sans_adapt_3_2B_control_S3_L008_R2_001.fastq.gz
/projects/bgmp/shared/2017_sequencing/demultiplexed/3_2B_control_S3_L008_R2_
001.fastq.gz
```

Trimmomatic

- LEADING: quality of 3
- TRAILING: quality of 3
- SLIDING WINDOW: window size of 5 and required quality of 15
- MINLENGTH: 35 bases
  Trimmomatic worked!

```
trimmomatic PE -threads 12
../sans_adapater/sans_adapt_3_2B_control_S3_L008_R1_001.fastq.gz
../sans_adapater/sans_adapt_3_2B_control_S3_L008_R2_001.fastq.gz
sans_adapt_3_2B_control_S3_L008_R1P_001.fastq.gz
sans_adapt_3_2B_control_S3_L008_R1U_001.fastq.gz
sans_adapt_3_2B_control_S3_L008_R2P_001.fastq.gz
sans_adapt_3_2B_control_S3_L008_R2U_001.fastq.gz LEADING:3 TRAILING:3
SLIDINGWINDOW:5:15 MINLEN:35
```

job ID: 16031653 using `sbatch_trim` sbatch script for trimmomatic for control and both.

Plotting read length distributions
Compute the distribution of lengths

```
zcat sans_adapt_32_4G_both_S23_L008_R1P_001.fastq.gz | grep -E '^[ATGCN]+$'|
awk '{print length($1)}' |sort -g | uniq -c >
sans_adapt_32_4G_both_S23_L008_R1P_001.read_length
```

```
zcat sans_adapt_32_4G_both_S23_L008_R2P_001.fastq.gz | grep -E '^[ATGCN]+$'
| awk '{print length($1)}' |sort -g | uniq -c >
sans_adapt_32_4G_both_S23_L008_R2P_001.read_length
```

```
zcat sans_adapt_3_2B_control_S3_L008_R1P_001.fastq.gz | grep -E '^[ATGCN]+$'
| awk '{print length($1)}' |sort -g | uniq -c >
sans_adapt_3_2B_control_S3_L008_R1P_001.read_length
```

```
zcat sans_adapt_3_2B_control_S3_L008_R2P_001.fastq.gz | grep -E '^[ATGCN]+$'
| awk '{print length($1)}' |sort -g | uniq -c >
sans_adapt_3_2B_control_S3_L008_R2P_001.read_length
```

I plotted these read length distribution files in the Rmd file.

Part 3:

Generating the genome with star:

sbatch script: `genome_database.sh`

```
STAR --runThreadN 12 --runMode genomeGenerate --genomeDir ./ --
genomeFastaFiles ../Mus_musculus.GRCm39.dna.primary_assembly.fa --
sjdbGTFfile Mus_musculus.GRCm39.112.gtf
```

Job ID: 16037599
Worked!

Alignment with Star
`sbatch star_sbatch.sh`
Job ID: 16038560
Included star alignment commands for control and both.
Result:

```
/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/3_2B_control_S3_L008/Mus_mu
sculus.GRCm39.dna.primary_assemblyAligned.out.sam
```

While the correct sam file is named `Mus_musculus.GRCm39.dna.primary_assemblyAligned.out.sam` which sounds like data from ensembl, it is actually the data aligned to the database in its respective folder.

Counting gene map counts using my Bi621 PS8 script:

Ran `mapcounts.py` copied from PS8. Hard-coded the input file names from above and ran it twice (once each).
`32_4G_both_S23_L008/Mus_musculus.GRCm39.dna.primary_assemblyAligned.out.sam`:
Number of mapped counts: 168582
Number of unmapped counts: 23431994
Total = 168582 + 23431994 = 23600576

`3_2B_control_S3_L008/Mus_musculus.GRCm39.dna.primary_assemblyAligned.out.sam`:
Number of mapped counts: 12359963
Number of unmapped counts: 496075
Total = 12359963 + 496075 = 12856038

Counting gene mappings using HTSeq-count:

You should run htseq-count twice: once with `--stranded=yes` and again with `--stranded=reverse`. Use default parameters otherwise.

```
htseq-count --stranded=yes
/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/3_2B_control_S3_L008/Mus_mu
sculus.GRCm39.dna.primary_assemblyAligned.out.sam
../Mus_musculus.GRCm39.112.gtf
```

```
htseq-count --stranded=reverse
/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/3_2B_control_S3_L008/Mus_mu
sculus.GRCm39.dna.primary_assemblyAligned.out.sam
../Mus_musculus.GRCm39.112.gtf
```

```
htseq-count --stranded=yes
/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/32_4G_both_S23_L008/Mus_mus
culus.GRCm39.dna.primary_assemblyAligned.out.sam
../Mus_musculus.GRCm39.112.gtf
```

```
htseq-count --stranded=reverse
/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/32_4G_both_S23_L008/Mus_mus
culus.GRCm39.dna.primary_assemblyAligned.out.sam
../Mus_musculus.GRCm39.112.gtf
```

Job ID: 16039341.
Completed successfully with `slurm-16039341.out`
output:
no_feature 5664164
ambiguous 5228
too_low_aQual 15298
not_aligned 239783
alignment_not_unique 281888
no_feature 526681
ambiguous 104201
too_low_aQual 15298
not_aligned 239783
alignment_not_unique 281888

no_feature 323
ambiguous 6
too_low_aQual 0
not_aligned 11715997
alignment_not_unique 83681
no_feature 318
ambiguous 9
too_low_aQual 0
not_aligned 11715997
alignment_not_unique 83681

HTseq-Count makes use of the information in the CIGAR string which is also what our own script uses to count gene features. The output does not include total number of reads that map to the reference assembly. Must calculate it myself.

I created a python script to parse this slurm output into four separate files, one for each htseq-count run. it also only includes the gene counts per gene in tsv format. This script is found: `/home/asol/bgmp/bioinfo/Bi623/QAA/star/alignment/htseq-count_parse.py`

Read the files in R to sum the counts with
Frequency Read_length

Counts via HTSeq-count
Stranded control:
5664164 + 5228 + 15298 + 239783 + 281888 = 6206361
Reverse control:
526681 + 104201 + 15298 + 239783 + 281888 = 1167851
Stranded both:
323 + 6 + 0 + 11715997 + 83681 = 11800007
Reverse both:
318 + 9 + 0 + 11715997 + 83681 = 11800005

If stranded, the `(unmapped + mapped) * 2 = Total from my script`
Basically, the total count from `HTSeq-count*2` = total from `mapcounts.py`

Total count from HTSeq-count	**Control**	**Both**
Stranded	6428019	12021662
Reverse	1389506	12021660
Sum	7817525	24043322
Doubled	--	--
Stranded	12856038	24043324

Total count from HTSeq-count	Control	Both
Reverse	2779012	24043320

Stranded control doubled = the total from my script above.

Percent of genes mapped generated by HTSeq-count, calculated in QAA_report.rmd:
3.44831
[1] 15.95207
[1] 1.843797
[1] 1.843797

	3_2B_control	32_4G_both
HTSeq-count forward	6,428,019	12,021,662
HTSeq-count reverse	1,389,506	12,021,660
sum	7,817,525	24,043,322
`mapcounts.py`	12,856,038	23,600,576
	3_2B_control	**32_4G_both**
HTSeq-count forward	6,428,019	12,021,662
HTSeq-count reverse	1,389,506	12,021,660
sum	7,817,525	24,043,322
`mapcounts.py`	12,856,038	23,600,576

Table 3: Percent of genes successfully mapped to the mouse reference genome counted with HTSeq-count.

	Control	Both
Total count `mapcounts.py`	12856038	23600576

Because the total count of genes in control doubled are equal to the total count of genes from `mapcounts.py`, we can confidently say that the control sample reads are stranded because `mapcounts.py` does not have a stranded option so it does not split up the counts, you must double the total counts from HTSeq-count to account for strandedness.
Additionally, because the reads from the "Both" samples doubled do not equal the counts from mapcounts.py, we can confidently say that the reads are not stranded. In addition, the percent of genes that "Both" samples map to the mouse reference database counted by `HTSeq-count`