RNA-seq Quality Assessment Assignment

Gera

10/12/2019

Bi 624 (Fall 2019) – Problem Set 1

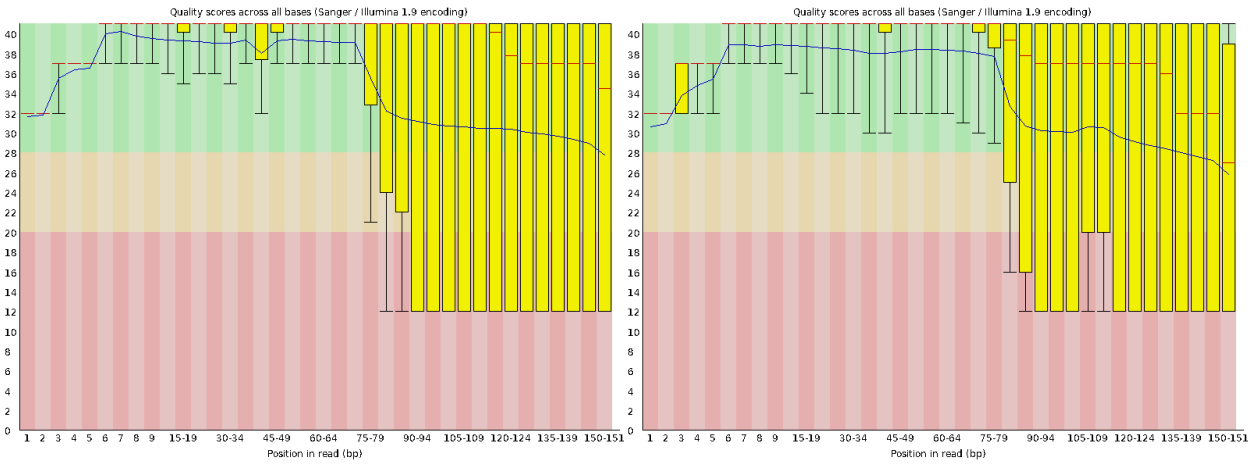
1. Using FastQC on Talapas, produce plots of quality score distributions for forward and reverse reads. Also, produce plots of the per-base N content

Running Fastqc on talapas

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=fastqc
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-01:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=7

module load fastqc/0.11.5
ml

/usr/bin/time -v fastqc /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R1_001.fastq.gz
/usr/bin/time -v fastqc /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R2_001.fastq.gz
/usr/bin/time -v fastqc /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R1_001.fas
/usr/bin/time -v fastqc /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R2_001.fas
```

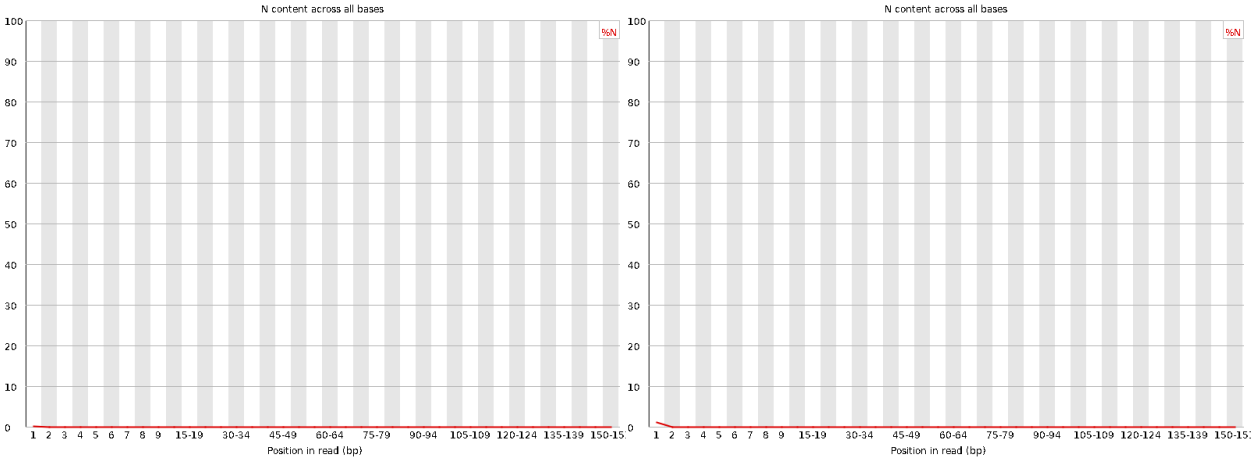
Figure 1: RAL306W-F3_S79_L006 fastqc

a. per base quality



Fastqc Per base sequence quality distributions output png files. Left is RAL306W-F3_S79_L006_R1_001.Right is RAL306-F3_S79_L006_R2_001

b. Per base n content



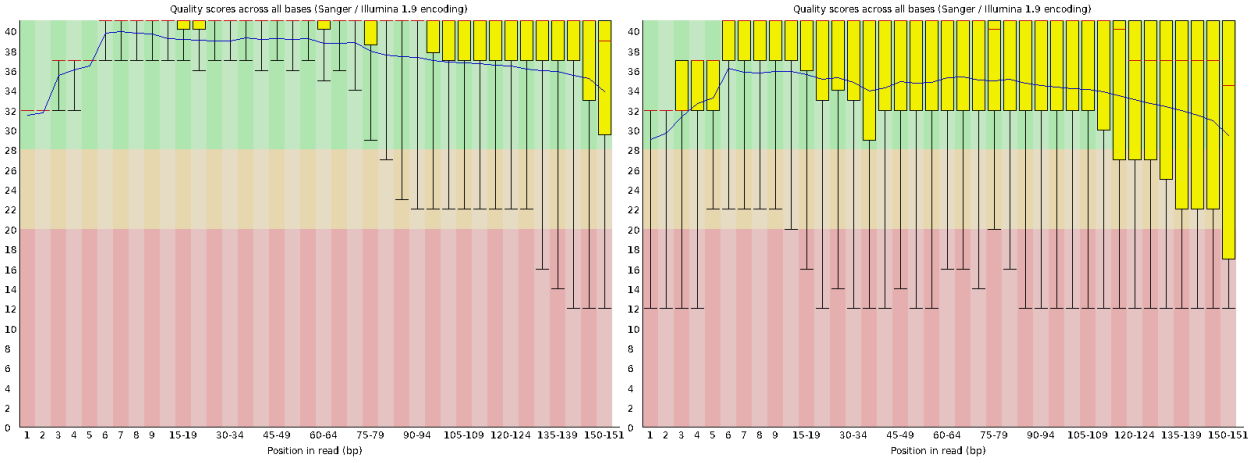
Fastqc Per base n content output png files. Left is RAL306W-F3_S79_L006_R1_001. Right is RAL306-F3_S79_L006_R2_001

and comment on whether or not they are consistent with the quality score plots.

The N cotent across the bases was consistent in which there was no N bases. The base quality scores were good, specifically 1 to 75-79 in RAL306W-F3_S79_L006 reads.

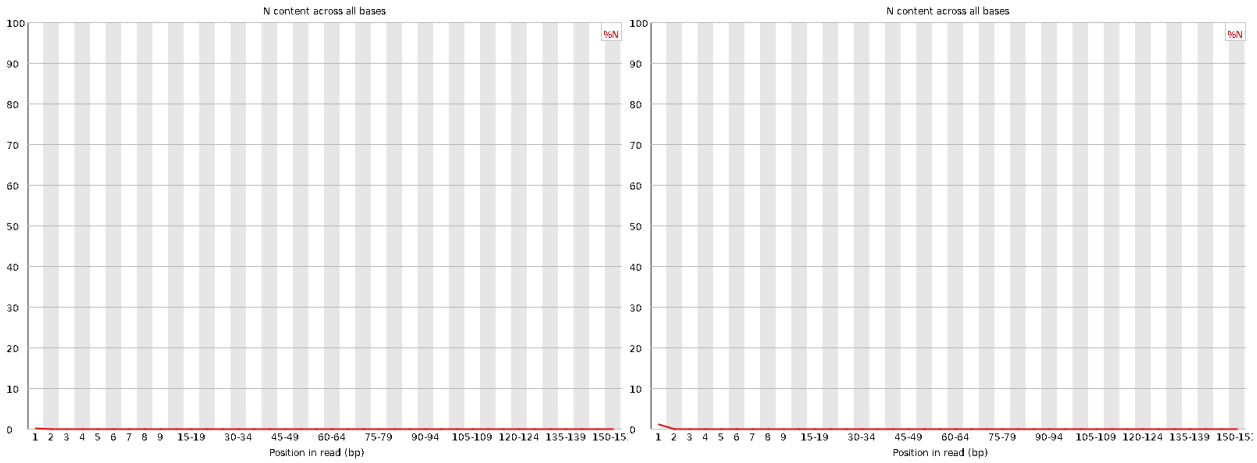
Figure 2: RAL306W_plus_M3_S78_L006 fastqc

a. per base quality



Fastqc Per base sequence quality distributions output png files. Left is RAL306W_plus_M3_S78_L006_R1_001. Right is RAL306W_plus_M3_S78_L006_R2_001

b. Per base n content



Fastqc Per base n content output png files. Left is RAL306W_plus_M3_S78_L006_R1_001. Right is RAL306W_plus_M3_S78_L006_R2_001

and comment on whether or not they are consistent with the quality score plots.

The N cotent across the bases was consistent in which there was no N bases. The mean curves look the same. RAL306W_plus_M3_S78_L006_R1_001 base quality scores were good from 1 to 75-79 but R2 was not as good. There was no N cotent across the bases in both reads.

Run your quality score plotting script from your Demultiplexing assignment from Bi 622

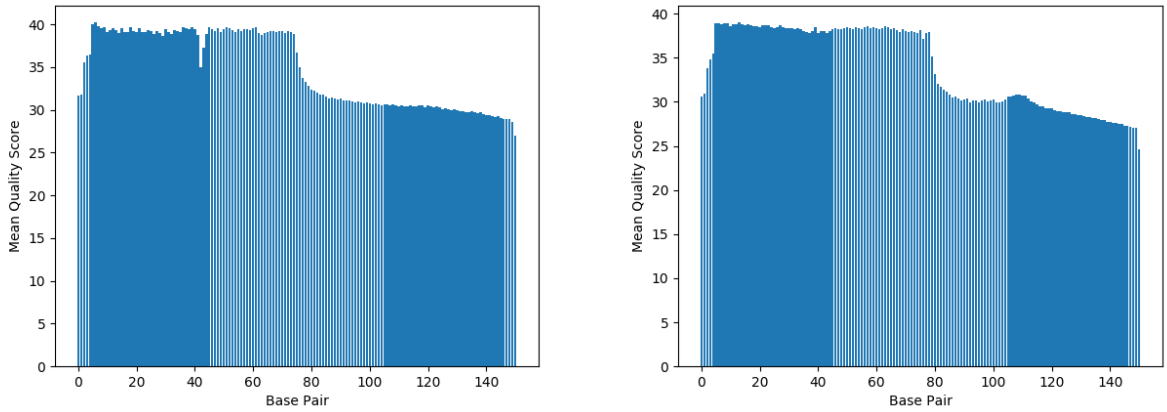
Running quality score plotting using python script

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=demu_F3_R1
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3



file1="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R1_001.fastq.gz"


/usr/bin/time -v python /home/gperez8/bgmp/projects/bgmp/gperez8/Bio622/c_count.py -f $file1 -o

RAL306W-F3_S79_L006_R1_001_hist
```

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=demu_F3_R2
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3




file1="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R2_001.fastq.gz"

#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=demu_M3_R1
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3




file1="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R1_001.fastq.gz"

#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=demu_M3_R2
#SBATCH --output=slurm-%j-%x.out
```

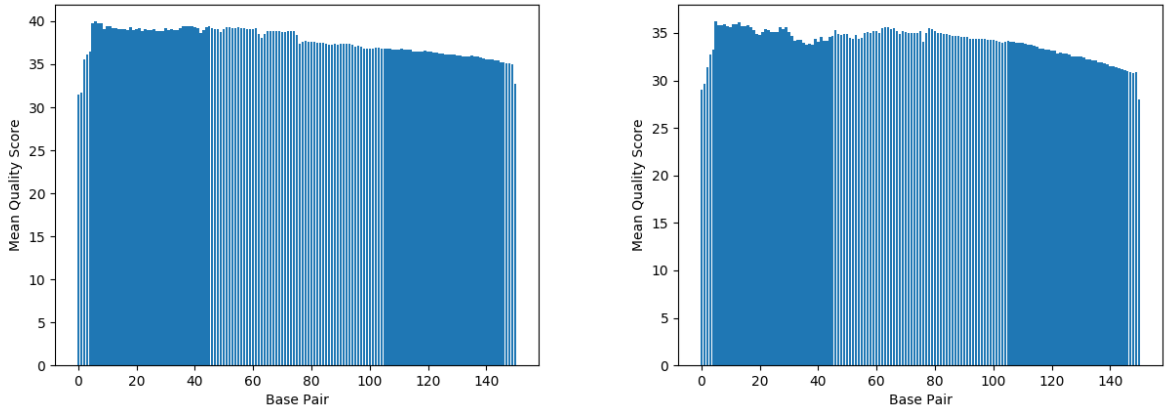
```
#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3



file1="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R2_001.fastq.gz"
```

Figure 3: RAL306W-F3_S79_L006 python script



Python script for quality score along each of the basepair distributions. Left is RAL306W-F3_S79_L006_R1_001. Right is RAL306-F3_S79_L006_R2_001

Figure 4: RAL306W_plus_M3_S78_L006 python script



Python script for quality score along each of the basepair distributions. Left is RAL306W_plus_M3_S78_L006_R1_001. Right is RAL306W_plus_M3_S78_L006_R2_001

Describe how the FastQC quality score distribution plots compare to your own

The mean score curve in the plots betwwen fastqc and Demultiplixing code looks about the same. The run time was different fastqc was 3-5 fold faster then the demultiplexing code. This can be due to fastqc using bins of base pairs instead of individual base pairs. Also, multithreading can make fastqc more efficient. In addition, fastqc uses the language Java which is fatser than python. Java is a compiled language where it runs faster compared to python as interpreted language.

Table 1

```
##                                 fast_qc (h:mm:ss)
## RAL306W-F3_S79_L006_R1_001      "0:14.15"
## RAL306W-F3_S79_L006_R2_001      "0:13.13"
## RAL306W_plus_M3_S78_L006_R1_001 "4:21.43"
## RAL306W_plus_M3_S78_L006_R2_001 "4:37.23"
##                                 demultiplexing_code (h:mm:ss)
## RAL306W-F3_S79_L006_R1_001      "0:48.04"
## RAL306W-F3_S79_L006_R2_001      "0:50.98"
## RAL306W_plus_M3_S78_L006_R1_001 "21:09.52"
## RAL306W_plus_M3_S78_L006_R2_001 "20:44.86"
```

Table shows the differeces in times running fastqc and python script for the from the R1 and R2 of the 2 demultiplexed file pairs.

Comment on the overall data quality of your two libraries.

The overall quality data of RAL306W-F3_S79_L006 reads were good. The mean scores curve for 1 to 75-79 where in the green then started to dip into the abnormal. Fastqc summary states that they are good. For RAL306W_plus_M3_S78_L006 R1, the mean score curve was good, good reads overall but started slightly deeping 1 to 75-79. Where for RAL306W_plus_M3_S78_L006 R2 it was not as good as R1 but the mean score curve did stay in the green started to fall 30-34.

Part 2 – Adaptor trimming comparison

Look into the adaptor trimming options for cutadapt and Trimmomatic (all on Talapas), and briefly describe the differences.

Trimmomatic can read single and paired ends. Removes adapters, leading or trailing low quality bases and N bases. Can cut adapter and other illumina-specific sequences, can cut bases off the start or the end of the read by a threshold quality. Has a sliding window trimming. Can drop read by a specific length. Has the convenience that it decides what adapter to use but can cut out overrepresented areas which can cut more than your adapter

Cutadapt can read single and paired ends. Removes adapter primers, sequences, poly-A tails and other unwanted sequences. Can modify and filter reads in various ways. IUPAC wildcard characters are allowed in adapter sequences. In cutadapt, you have to specify the adapters. It throws out reads that are less than the adapters size.

For the this assignment I used cutadapt because it was easier to get it running compared to trimmomatic.

What proportion of reads (both forward and reverse) was trimmed?

Running cutadapt

```
#!/bin/bash

#SBATCH --account=bgmp
```

```
#SBATCH --partition=bgmp
#SBATCH --job-name=cutadapt_pair
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3

module load easybuild  icc/2017.1.132-GCC-6.3.0-2.27  impi/2017.1.132
module load cutadapt/1.18-Python-3.7.0




file1="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R1_001.fastq.gz"
file2="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R2_001.fastq.gz"
file3="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R1_001.fastq.gz"
file4="/projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R2_001.fastq.gz"







/usr/bin/time -v cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
 -o RAL306W-F3_S79_L006_R1_001_cutadapt.fastq.gz   -p RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz \
 $file1 $file2 -m 30
/usr/bin/time -v cutadapt -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA -A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-o RAL306W_plus_M3_S78_L006_R1_001_cutadapt.fastq.gz -p RAL306W_plus_M3_S78_L006_R2_001_cutadapt.fastq.g
$file3 $file4 -m 30
```

According cutadapt for RAL306W-F3_S79_L006 R1 and R2 pairs 767,924 of 1,153,925, resulting in 66.5% of the reads were trimmed.

According cutadapt for RAL306W_plus_M3_S78_L006 R1 and R2 pairs 29,169,013 of 30,943,940 resulting in 94% of the reads were trimmed.

setwd("/Users/gerardoperez/Documents/shell/Bi624")

Sanity check: Use your Unix skills to search for the adapter sequences in your datasets and confirm the expected sequence orientations. You may want to refer to Graded Assignment 4 from Bi623. Report the commands you used

```
#PRETRIMMED

#R1 adapter to RAL306W-F3_S79_L006_R1_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R1_001.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
```

```
40111

#R1 adapter to RAL306W-F3_S79_L006_R2_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R2_001.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
1


#R2 adapter to RAL306W-F3_S79_L006_R1_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R1_001.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0

#R2 adapter to RAL306W-F3_S79_L006_R2_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W-F3_S79_L006_R2_001.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
38046

#R1 adapter to RAL306W_plus_M3_S78_L006_R1_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R1_001.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
341962

#R1 adapter to RAL306W_plus_M3_S78_L006_R2_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R2_001.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
5

#R2 adapter to RAL306W_plus_M3_S78_L006_R1_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R1_001.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0

#R2 adapter to RAL306W_plus_M3_S78_L006_R2_001
zcat /projects/bgmp/shared/2019_Drosphila_RNAseq/RAL306W_plus_M3_S78_L006_R2_001.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
341211


#POSTRIMMED

#R1 adapter to RAL306W-F3_S79_L006_R1_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R1_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
0

#R1 adapter to RAL306W-F3_S79_L006_R2_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
0

#R2 adapter to RAL306W-F3_S79_L006_R1_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R1_001_cutadapt.fastq.gz| \
```

```
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0

#R2 adapter to RAL306W-F3_S79_L006_R2_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0

#R1 adpater to RAL306W-F3_S79_L006_R1_001_cutadapt
 zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R1_001_cutadapt.fastq.gz| \
 grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
0

#R1 adpater to RAL306W-F3_S79_L006_R2_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCACACGTCTGAACTCCAGTCA" | wc -l
0

#R2 adapter to RAL306W_plus_M3_S78_L006_R1_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W_plus_M3_S78_L006_R1_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0

#R2 adapter to RAL306W_plus_M3_S78_L006_R2_001_cutadapt
zcat /home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W_plus_M3_S78_L006_R2_001_cutadapt.fastq.gz| \
grep "AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT" | wc -l
0
```

the reasoning behind them, and how you confirmed the adapter sequences.

The reason behind this was to confirm by greping and word count line of the Illumina adapter sequences, from the R1 and R2 of the 2 demultiplexed file pairs, are in the correct orientation pre-trimmed and absent post trimmed. The Illumina's TruSeq LT and TruSeq HT-based kits Read 1 and Read 2 adapter sequences were used. The confirmation came from having a large value in the grep word count line using Read 1 adapter on RAL306W-F3_S79_L006_R1_001 and doing the same for Read 2 adapter on RAL306W-F3_S79_L006_R2_001. Which did result into outputing large values. For coriousity, the same commands were used using Read 1 adapter on RAL306W-F3_S79_L006_R2_001 and Read 2 adapter on RAL306W-F3_S79_L006_R1_001 to confirm there was 0 which was the result. This same approach was used for RAL306W_plus_M3_S78_L006 R1 and R2 files. Furthermore, using postrimmed files, these same commands were run and resulted in 0 which is what we expected.

https://support.illumina.com/bulletins/2016/12/what-sequences-do-i-use-for-adapter-trimming.html

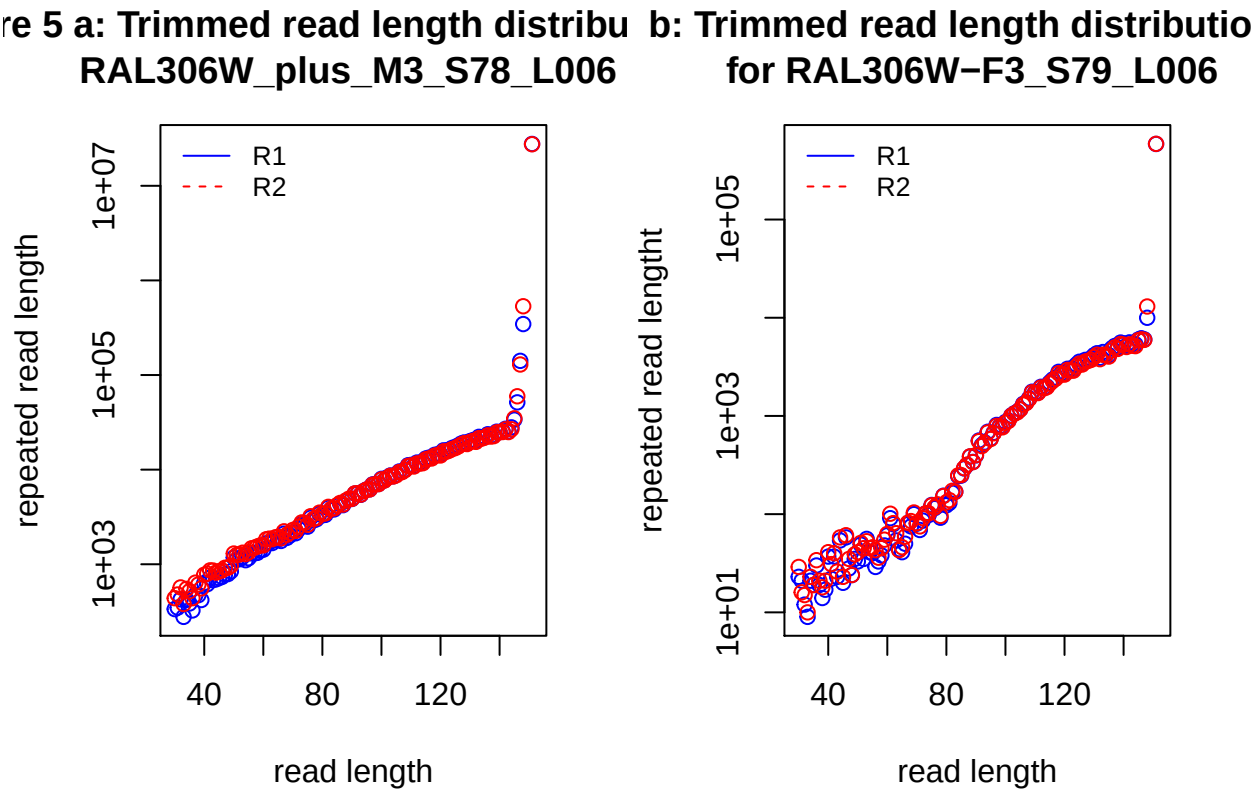
Plot the trimmed read length distributions for both forward and reverse reads (on the same plot). You can produce 2 different plots for your 2 different RNA-seq samples

Commands to get the read length distributions.

```
zcat RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz | sed -n '2~4p' | awk '{print length($0)}'|\
sort| uniq -c > RAL306W-F3_S79_L006_R2_001_cutadapt_read_len.txt

zcat RAL306W_plus_M3_S78_L006_R1_001_cutadapt.fastq.gz | sed -n '2~4p' | awk '{print length($0)}'|\
sort| uniq -c > RAL306W_plus_M3_S78_L006_R1_001_cutadapt_read_len.txt
```

Creating Trimmed read length distribution plots

```
M3_R1_txt <- read.table("RAL306W_plus_M3_S78_L006_R1_001_cutadapt_read_len.txt")
M3_R2_txt <- read.table("RAL306W_plus_M3_S78_L006_R2_001_cutadapt_read_len.txt")
F3_R1_txt <- read.table("RAL306W-F3_S79_L006_R1_001_cutadapt_read_len.txt")
F3_R2_txt <- read.table("RAL306W-F3_S79_L006_R2_001_cutadapt_read_len.txt")

par(mfrow=c(1,2))
plot(M3_R1_txt[,2],M3_R1_txt[,1], log="y", col="blue", xlab="read length",
     ylab="repeated read length", main="Figure 5 a: Trimmed read length distributions for \nRAL306W_plus
points(M3_R2_txt[,2],M3_R2_txt[,1], col="red")
legend("topleft", legend=c("R1", "R2"),
       col=c("blue", "red"), lty=1:2, cex=0.8,
       box.lty=0)

plot(F3_R1_txt[,2],F3_R1_txt[,1], log="y", col="blue", xlab="read length",
     ylab="repeated read lengtht", main="b: Trimmed read length distributions \nfor RAL306W-F3_S79_L006
points(F3_R2_txt[,2],F3_R2_txt[,1], col="red")
legend("topleft", legend=c("R1", "R2"),
       col=c("blue", "red"), lty=1:2, cex=0.8,
       box.lty=0)
```



Trimmed read length distributions for R1 and R2 of the 2 demultiplexed file pairs. Figure 5 a shows at the beginning read length R1 having less repeated read lengths compared to R2 but then overlapping until last few points. Figure 5 b shows at the beginning reads scattered but some overlap and more overlap as the reads length increases until the last few points.

Comment on whether you expect R1s and R2s to be adapter-trimmed at different rates

We should definitely see expect R1s and R2s to be adapter-trimmed at different rates.

Part 3 – rRNA reads and strand-specificity

Find publicly available Drosophila melanogaster rRNA sequences and generate an alignment database (e.g. STAR) from them.

Downloaded the Drosophila_melanogaster.BDGP6.22.ncrna.fa.gz file from ensembl.org. and did a gunzip.

bash commands to filter rRNA from Drosophila_melanogaster.BDGP6.22.ncrna.fa.gz file

```
awk '/^>/ {printf("\n%s\n",$0);next; } { printf("%s",$0);}  END {printf("\n");}' \
< Drosophila_melanogaster.BDGP6.22.ncrna.fa > out.fa

grep -A 1 "rRNA" out.fa  > Drosophila_melanogaster.BDGP6.22.ncrna_RNA.fa
```

Runnin STAR to create Drosophila_melanogaster rRNA database

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=Star_database
#SBATCH --output=slurm-%j-%x.out
#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=7

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3



genomeDir="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/Drosophila_melanogaster.BDGP6.22._RNA_.STAR_2
genomeFastaFiles="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/Drosophila_melanogaster.BDGP6.22.RNA.fa


mkdir -p $genomeDir



/usr/bin/time -v STAR --runThreadN 7 --runMode genomeGenerate \
    --genomeDir $genomeDir \
    --genomeFastaFiles $genomeFastaFiles --genomeSAindexNbases 4
```

Align the reads to your fly rRNA database (e.g. STAR) and report the proportion of reads that likely came from rRNAs.

Runnin STAR to align the 2 trimmed demultiplexed file pairs reads to Drosophila_melanogaster rRNA.

```
#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=star_aligner
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=7

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3


file1="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R1_001_cutadapt.fastq.gz"
file2="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W-F3_S79_L006_R2_001_cutadapt.fastq.gz"
genomeDir="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/Drosophila_melanogaster.BDGP6.22._RNA_.STAR_2




/usr/bin/time -v STAR --runThreadN 7 --runMode alignReads --outFilterMultimapNmax 3 \
  --outSAMunmapped Within KeepPairs \
  --readFilesCommand zcat \
  --readFilesIn $file1 $file2   --genomeDir $genomeDir \
  --outFileNamePrefix aligned_RAL306W-F3_S79_L006_rRNA____ --genomeSAindexNbases 4


#!/bin/bash

#SBATCH --account=bgmp
#SBATCH --partition=bgmp
#SBATCH --job-name=star_aligner
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=7

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3


file3="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W_plus_M3_S78_L006_R1_001_cutadapt.fastq.gz
```

```
file4="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/RAL306W_plus_M3_S78_L006_R1_001_cutadapt.fastq.gz
genomeDir="/home/gperez8/bgmp/projects/bgmp/gperez8/Bi624/Drosophila_melanogaster.BDGP6.22._RNA_.STAR_2
```

```
/usr/bin/time -v STAR --runThreadN 7 --runMode alignReads --outFilterMultimapNmax 3 \
  --outSAMunmapped Within KeepPairs \
  --readFilesCommand zcat \
  --readFilesIn $file3 $file4   --genomeDir $genomeDir \
  --outFileNamePrefix aligned_RAL306W_plus_M3_S78_L006_rRNA____ --genomeSAindexNbases 4
```

and report the proportion of reads that likely came from rRNAs

From the alignnment result and looking into the aligned_RAL306W-F3_S79_L006_rRNA Log.final.out file, rRNA uniquely mapped reads were 1.13%. As for, the aligned_RAL306W_plus_M3_S78_L006_rRNA Log.final.out file, rRNA uniquely mapped reads were 0%.

These numbers state that 2 trimmed demultiplexed file pair reads of rRNA proportions were small to nothing. This is hard to beleive. There is most likely an incorrect error in running STAR that needs to be explored and corrected. Due to time, I went forward with these alignments to explore further questions.

Demonstrate convincingly whether or not the data are from "strand-specific" RNA-Seq libraries. There are a few possible strategies to address this problem, but you need only implement one

From the alignment output result sam file, we can use the bit wise flag 64, which is "First in pair" and 16, which is "Read reverse strand" to determine the stran-specific. From running this, if we see an even distrubution in forward and reverse, this would mean no strandness. If we see a lopsided distrubution then it would mean strandness.

Python script to counts up the number of reads that are properly forward and reverse from a sam file

```
#!/usr/bin/env python

# Program Header
# Course: Bi621
# Name:   Gerardo Perez
# Description: Python script that that counts up the number
# of reads that are properly forward and reverse from a sam file.
# to find "strand-specific".
#
#
#
# Development Environment: Atom 1.38.2
# Version: Python 3.7.3
# Date:  06/19/2019
################################################

# Imports modules
import re
import argparse

# Creates an arguement passer
parser = argparse.ArgumentParser(description="Program parses the contents of your SAM file. \
counts up the number of reads that are properly forward and reverse from sam file")
```

```
# Adds arguemets by calling the arguement passer
parser.add_argument("-f", "--filename", help="specify the filename", required=True)


# Parses arguemets through using the parse args method.
args = parser.parse_args()

# Creates a variable
forward=0
unmapped=0
reverse=0
count=0

#Opens a text file to read. stores the file as a variable.
with open(args.filename, 'r') as fh:

    # A for loop that goes through each line in the file.
    for line in fh:

        # If statement that uses regex to find the first the line with a specific first charcter.
        if re.match(r'^K', line):

            # splits the line into an array everytime it encounters a tab.
            parts = line.split('\t')

            # converts the index one in the array  into to an int and stores it as a variable.
            flags=int(parts[1])

            # if statement that checks if check the current read is forward and counts for forward and
            if ((flags & 4)==4):
                    if((flags & 64) == 64):
                        if(flags & 16) == 16:
                            forward=forward+1
                        else: reverse=reverse+1
            else: unmapped=unmapped+1
            count=count+1

# opens a text file to read and stores the text file as a variable.
f1 = open("forward_vs_reverse.tsv", "w")

# writes the formatted output to the tsv file
f1.write("forward:{0}\treverse:{1}\n".format(forward,reverse))

# Prints a statement in a specific format.
print("forward:", forward)
print("reverse:", reverse)
print("unmapped:", unmapped)
```

Using bash to run the python script

```
#!/bin/bash

#SBATCH --account=bgmp
```

```
#SBATCH --partition=bgmp
#SBATCH --job-name=parse_sam
#SBATCH --output=slurm-%j-%x.out

#SBATCH --time=0-12:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=7

conda deactivate
conda deactivate
conda deactivate
conda deactivate
conda activate bgmp_py3



/usr/bin/time -v python parse_sam_strand.py -f aligned_RAL306W-F3_S79_L006_rRNA____Aligned.out.sam

/usr/bin/time -v python parse_sam_strand.py -f aligned_RAL306W_plus_M3_S78_L006_rRNA____Aligned.out.sam
```

Running the python script using bash, the output for running the aligned_RAL306W-F3_S79_L006_rRNA sam file was forward: 0, reverse: 758684, unmapped: 19747. As for running aligned_RAL306W_plus_M3_S78_L006_rRNA sam file was forward: 0, reverse: 29169000, unmapped: 52. From these results, the data are from a strand specific RNA-Seq libraries.