Working_Notes

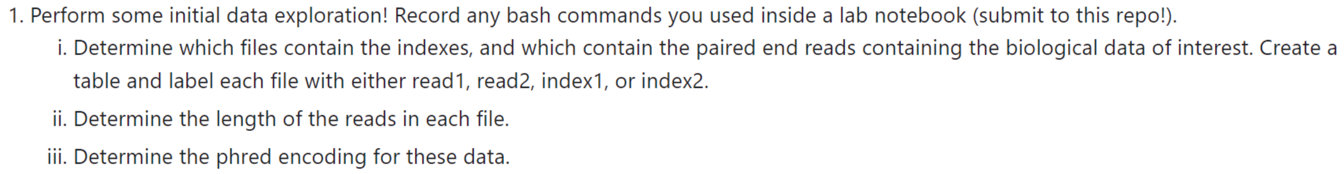
Assignment the first

Talapas destination:
`/projects/bgmp/shared/2017_sequencing`
Contains the following files:

```
1294_S1_L008_R1_001.fastq.gz
1294_S1_L008_R2_001.fastq.gz
1294_S1_L008_R3_001.fastq.gz
1294_S1_L008_R4_001.fastq.gz
```

Don't unzip, don't copy. zcat all the way
All work should be done in Talapas
Note that this is a bulk RNA seq.

Part 1 – Quality Score Distribution per-nucleotide

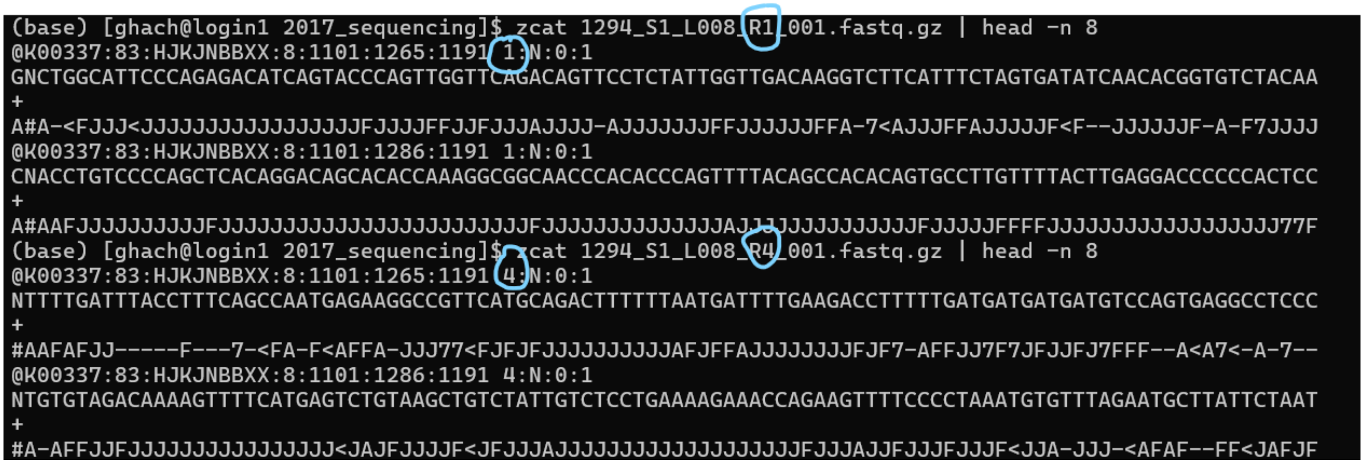


1. Perform some initial data exploration! Record any bash commands you used inside a lab notebook (submit to this repo!).
    i. Determine which files contain the indexes, and which contain the paired end reads containing the biological data of interest. Create a table and label each file with either read1, read2, index1, or index2.
    ii. Determine the length of the reads in each file.
    iii. Determine the phred encoding for these data.

Part 1.1 subpart i:

1294_S1_L008_R1_001.fastq.gz 1294_S1_L008_R4_001.fastq.gz **almost certainly correspond to read 1 and read 2 respectively**, because that's just the convention, but we can tell which are which from the fastq headers. This can be verified with the commands:

```
zcat 1294_S1_L008_RN_001.fastq.gz | head -n 8
```

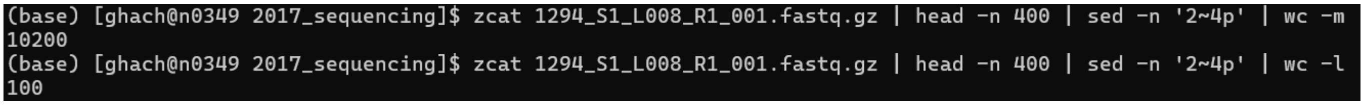


```
(base) [ghach@login1 2017_sequencing]$ zcat 1294_S1_L008_R1_001.fastq.gz | head -n 8
@K00337:83:HJKJNBBXX:8:1101:1265:1191 1:N:0:1
GNCTGGCATTCCCAGAGACATCAGTACCCAGTTGGTTCAGACAGTTCCTCTATTGGTTGACAAGGTCTTCATTTCTAGTGATATCAACACGGTGTCTACAA
+
A#A-<FJJJ<JJJJJJJJJJJJJJJJFJJJJFFJJFJJJAJJJJ-AJJJJJJJFFJJJJJJFFA-7<AJJJFFAJJJJJF<F--JJJJJJF-A-F7JJJJ
@K00337:83:HJKJNBBXX:8:1101:1286:1191 1:N:0:1
CNACCTGTCCCCAGCTCACAGGACAGCACACCAAAGGCGGCAACCCACACCCAGTTTTACAGCCACACAGTGCCTTGTTTTACTTGAGGACCCCCCACTCC
+
A#AAFJJJJJJJJJJFJJJJJJJJJJJJJJJJJJJJJJJJFJJJJJJJJJJJJJAJJJJJJJJJJJJJFJJJJJFFFFJJJJJJJJJJJJJJJJJJ77F
(base) [ghach@login1 2017_sequencing]$ zcat 1294_S1_L008_R4_001.fastq.gz | head -n 8
@K00337:83:HJKJNBBXX:8:1101:1265:1191 4:N:0:1
NTTTTGATTTACCTTTCAGCCAATGAGAAGGCCGTTCATGCAGACTTTTTTAATGATTTTGAAGACCTTTTTGATGATGATGATGTCCAGTGAGGCCTCCC
+
#AAFAFJJ-----F---7-<FA-F<AFFA-JJJ77<FJFJFJJJJJJJJJJAFJFFAJJJJJJJJFJF7-AFFJJ7F7JFJJFJ7FFF--A<A7<-A-7--
@K00337:83:HJKJNBBXX:8:1101:1286:1191 4:N:0:1
NTGTGTAGACAAAAGTTTTCATGAGTCTGTAAGCTGTCTATTGTCTCCTGAAAAGAAACCAGAAGTTTTCCCCTAAATGTGTTTAGAATGCTTATTCTAAT
+
#A-AFFJJFJJJJJJJJJJJJJJJJ<JAJFJJJJF<JFJJJAJJJJJJJJJJJJJJJJJJJJFJJJAJJFJJJFJJJF<JJA-JJJ-<AFAF--FF<JAFJF
```

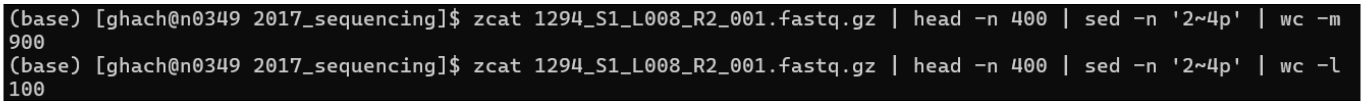
Part 1.1 subpart ii:

```
zcat 1294_S1_L008_RN_001.fastq.gz | head -n 240 | sed -n '2~4p' | wc -<l or m>
```

Now to get read length, use a subset of the file using head and a call to sed:



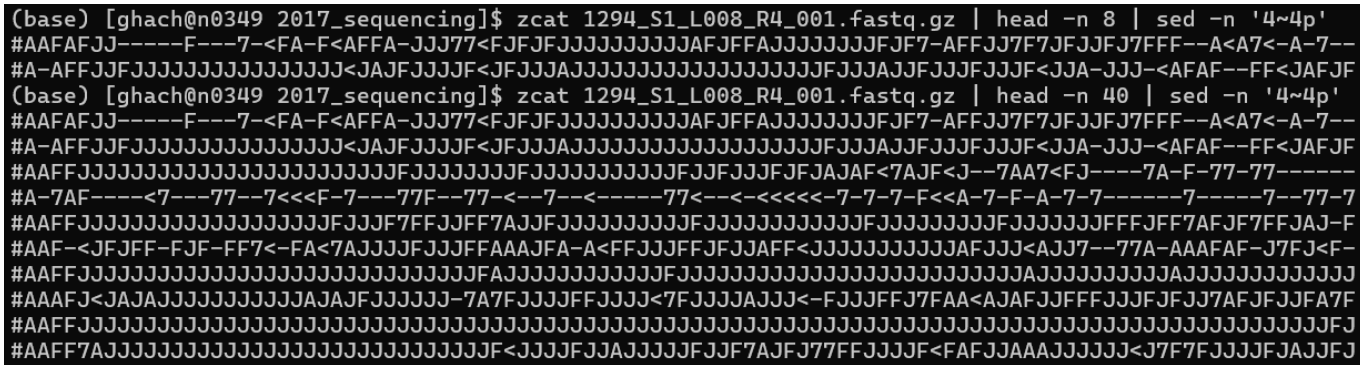
```
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R1_001.fastq.gz | head -n 400 | sed -n '2~4p' | wc -m
10200
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R1_001.fastq.gz | head -n 400 | sed -n '2~4p' | wc -l
100
```



```
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R2_001.fastq.gz | head -n 400 | sed -n '2~4p' | wc -m
900
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R2_001.fastq.gz | head -n 400 | sed -n '2~4p' | wc -l
100
```

Part 1.1 subpart iii:

```
zcat 1294_S1_L008_RN_001.fastq.gz | head -n 40 | sed -n '4~4p'
```



```
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R4_001.fastq.gz | head -n 8 | sed -n '4~4p'
#AAFAFJJ-----F---7-<FA-F<AFFA-JJJ77<FJFJFJJJJJJJJJJAFJFFAJJJJJJJJFJF7-AFFJJ7F7JFJJFJ7FFF--A<A7<-A-7--
#A-AFFJJFJJJJJJJJJJJJJJJJ<JAJFJJJJF<JFJJJAJJJJJJJJJJJJJJJJJJJJJFJJJAJJFJJJFJJJF<JJA-JJJ-<AFAF--FF<JAFJF
(base) [ghach@n0349 2017_sequencing]$ zcat 1294_S1_L008_R4_001.fastq.gz | head -n 40 | sed -n '4~4p'
#AAFAFJJ-----F---7-<FA-F<AFFA-JJJ77<FJFJFJJJJJJJJJJAFJFFAJJJJJJJJFJF7-AFFJJ7F7JFJJFJ7FFF--A<A7<-A-7--
#A-AFFJJFJJJJJJJJJJJJJJJJ<JAJFJJJJF<JFJJJAJJJJJJJJJJJJJJJJJJJJJFJJJAJJFJJJFJJJF<JJA-JJJ-<AFAF--FF<JAFJF
#AAFFJJJJJJJJJJJJJJJJJJJJJJJJJFJJJJJJJFJJJJJJJJJJFJJFJJJFJFJAJAF<7AJF<J--7AA7<FJ----7A-F-77-77------
#A-7AF----<7---77--7<<<F-7---77F--77-<--7--<-----77<--<-<<<<<-7-7-7-F<<A-7-F-A-7-7------7-----7--77-7
#AAFFJJJJJJJJJJJJJJJJJJFJJJF7FFJJFF7AJJFJJJJJJJJJFJJJJJJJJJJJFJJJJJJJJFJJJJJJFFFJFF7AFJF7FFJAJ-F
#AAF-<JFJFF-FJF-FF7<-FA<7AJJJJFJJJFFAAAJFA-A<FFJJJFFJFJJAFF<JJJJJJJJJJAFJJJ<AJJ7--77A-AAAFAF-J7FJ<F-
#AAFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFAJJJJJJJJJJJFJJJJJJJJJJJJJJJJJJJJJJJJJJAJJJJJJJJJAJJJJJJJJJJJJ
#AAAFJ<JAJAJJJJJJJJJJAJAJFJJJJJJ-7A7FJJJJFFJJJJ<7FJJJJAJJJ<-FJJJFFJ7FAA<AJAFJJFFFJJJFJFJJ7AFJFJJFA7F
#AAFFJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJFJ
#AAFF7AJJJJJJJJJJJJJJJJJJJJJJJJJJJJJF<JJJJFJJAJJJJJFJJF7AJFJ77FFJJJJF<FAFJJAAAJJJJJJ<J7F7FJJJJFJAJJFJ
```

2. Generate a per base distribution of quality scores for read1, read2, index1, and index2. Average the quality scores at each position for all reads and generate a per nucleotide mean distribution **as you did in part 1 of PS4 in Bi621**. (NOTE! Do NOT use the 2D array strategy from PS9 - you WILL run out of memory!)
    i. Turn in the 4 histograms.
    ii. What is a good quality score cutoff for index reads and biological read pairs to utilize for sample identification and downstream analysis, respectively? Justify your answer.
    iii. How many indexes have undetermined (N) base calls? (Utilize your command line tool knowledge. Submit the command(s) you used. CHALLENGE: use a one-line command)

Part 1.1 results:

File name	label	Read length	Phred encoding
1294_S1_L008_R1_001.fastq.gz	Read 1	101	Phred+33
1294_S1_L008_R2_001.fastq.gz	Barcodes corresponding to Read 1	8	Phred+33
1294_S1_L008_R3_001.fastq.gz	Barcodes corresponding to Read 2	8	Phred+33
1294_S1_L008_R4_001.fastq.gz	Read 2	101	Phred+33

Part 1.2

Part 1.2 subpart i, development

Okay, now I'm cloning the repo (`/projects/bgmp/ghach/bioinfo/Bi622/`) and getting to making the python script.

I'm going to create a mini test files for my histogram code for this using the command:

```
zcat 1294_S1_L008_R1_001.fastq.gz | head -n 24 >
/projects/bgmp/ghach/bioinfo/Bi622/Demultiplex/Assignment-the-first/mini.fq
```

NOTE! I had issues getting my print statements to work in this script. The issue was resolved by adding flush=True to all print statements. More on buffer issues (which I do not fully understand yet) in the following article:
buffers

Part 1.2, answers

As the histograms show, the quality scores are lowest, by a significant margin, at the start of the reads. After having done the mRNA prep wet lab and gaining some appreciation for how labor-intensive it is (and how underfunded some important areas of biology are) I would like to err on the side of not discarding potentially useful data. For the purpose of downstream analysis, I'd recommend filtering out those biological reads with mean Q scores less than or equal to 30 (though I'd want to know more about what this data would be used for to provide a more detailed recommendation).

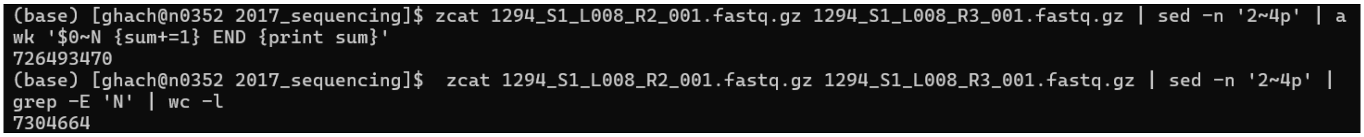
When it comes to the index quality I think it's justifiable to be far more permissive. If the index isn't present in the group of 24, it automatically goes to the undetermined file. Thus, the Q score filtering criterion is relevant **only in the case where the barcode does match one of the 24.** Since there are eight bases per barcode, and 4 possibilities for each base, there are 4096 possible barcodes. Thus, even if the bases are randomly picked out of a hat, the probability that a barcode is misread, but coincidentally reads as another of the 24 is very unlikely, because they represent less than 0.6% of the total space of 8 base sequences.
For these reasons, I will filter only sequences with a mean Q score below 20.

Part 1.2 subpart iii command:

First try:

```
zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p'
| awk '$0~N {sum+=1} END {print sum}'
```

THIS DIDN'T WORK! THIS COUNTED ALL LINES AND I DON'T KNOW WHY. IT WORKED ON A TEST FILE, EVEN WHEN IT WAS GZIPPED

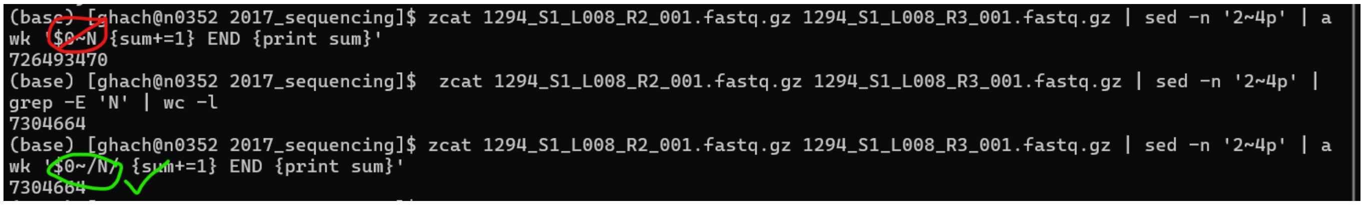


```
(base) [ghach@n0352 2017_sequencing]$ zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' | a
wk '$0~N {sum+=1} END {print sum}'
726493470
(base) [ghach@n0352 2017_sequencing]$  zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' |
grep -E 'N' | wc -l
7304664
```

Second try:

```
zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p'
| grep -E 'N' | wc -l
7304664
```

Arg, okay, I figured it out:



```
(base) [ghach@n0352 2017_sequencing]$ zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' | a
wk '$0~N {sum+=1} END {print sum}'
726493470
(base) [ghach@n0352 2017_sequencing]$  zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' |
grep -E 'N' | wc -l
7304664
(base) [ghach@n0352 2017_sequencing]$ zcat 1294_S1_L008_R2_001.fastq.gz 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' | a
wk '$0~/N/ {sum+=1} END {print sum}'
7304664
```

Part 2 – Develop an algorithm to de-multiplex the samples

1. Define the problem
   - We are seeking to take reads 1- 4 (each from separate fastq files) and export R1 and R4 into two files, belonging to one of three categories:
     1. Those where the barcodes (R2 and R3) are high quality, and match both those in the table, and each other
        - These go to a different file for each pair of barcodes (ex GTAGCGTA_R1.fq)
     2. Those where the barcodes are high quality (meet quality score cutoff and do not contain Ns) but do not match, indicating that index hopping has occurred
        - Reads go into hopped_R1.fq and hopped_R2.fq
     3. Those where the reads do match, but the quality score is below the cutoff, or the indices don't match those in the table
        - Reads go into unknown_R1.fq and unknown_R2.fq
2. Describe output
   - We would also like to keep count how many reads go into each of the three categories defined above
     - This output is short, and so will be printed directly to std out
   - We would also like to keep count of all reads in category 1 and 2 above by their ordered barcode pairs
     - The above output will be longer, and so will be exported to a .tsv file. This file will also have the counts described in the previous bullet point at the top
3. Upload your 4 input FASTQ files and your >=6 expected output FASTQ files.

4. Pseudocode
    - My pseudocode can be found here
5. High level functions. For each function, be sure to include:
    1. Description/doc string
    2. Function headers (name and parameters)
    3. Test examples for individual functions
    4. Return statement
    - My pseudo-functions can be found here

Running mean formula:

$$\text{Mean}: \quad \mu_n = \mu_{n-1} + \frac{x_n - \mu_{n-1}}{n}$$

Running histogram code

Run as sbatch job ID batch job 7790187
fifteenhundreth time is the charm:
Submitted batch job 7791153