Part 1
1.

1294_S1_L008_R1_001.fastq.gz	Read 1
1294_S1_L008_R2_001.fastq.gz	index 1
1294_S1_L008_R3_001.fastq.gz	Index 2
1294_S1_L008_R4_001.fastq.gz	Read 2

2b. From looking at the average means scores histogram in the index reads. A good cut off quality score is 29. The reason for the score cutoff is that it will include all the average base pairs ,which was 30, for each position. The same reason was used for the biological reads, the cut off was 28 since the average base pairs was 29.

2c.

Input: file2="/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R2_001.fastq.gz"
/usr/bin/time -v zcat $file2 | sed -n '2~4p' | grep -c "N"

Output: 3976613

Input: file3="/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R3_001.fastq.gz"
/usr/bin/time -v zcat $file3 | sed -n '2~4p' | grep -c "N"

Output: 3328051

Part 2

The goal is to look through a lane of sequencing library preps. These preps were made by the 2017 bgmp cohort. A concern besides getting low quality data is index-hopping. Index-hopping is a problem because it can result in miss assigning the wrong index leading to misalignments. Another concern is to filter out undetermined unknown barcodes. If these concerns are not addressed they can cause negative impact on data quality.

The output that would be informative would be the forward and reverse fastq with matching index pairs. This is is necessary for downstream analyses . Also, the report of the number of read pairs with properly matched index, number of read pairs with index-hopping and read pairs with unknown would be informative to get a sense of workflow efficiency. For example, If

the efficiency is a lot of index hopping then something needs to be changed in the library preparations.

Pseudocode:

1. Initialize argparse to take 4 fastq files to be read. Ex Read1 file, Read2 file, Read3 file and Read4 file.
   Initialize argparse to output names for 6 fastq files forward matching pair index file, reverse matching pair index file, forward undertermine list file, reverse undertermine list file, forward index hopped list file and reverse index hopped list file.
2. Create a method to do a reverse complement: Def rev_comp(seq)
   "Takes a sequence and returns the reverse the complement"

   Return rev_seq

3. Create a list of 24 indexed (dual matched) libraries. Index list
4. Using a while loop open the 4 files. Extract the 4 lines of Read1 into variables. Extract the second line of Read2 into a variable. Extract the second line of Read3 into Variable. Extract the 4 lines of Read4 into variables.

5. Check the index files quality score. If it's below cutoff, they go to undetermined. Then check biological reads cutoff.

6. Check if the 2nd line in Read2 is in Index list. If false, then write the 4 lines from Read1 file with the 2nd line concatenated with Read 2 line and Read 3 line into forward undertermine list file. Repeat the same code but with Read4 file and write the concatenated lines to reverse undertermine list file. Have a counter to increment after every undetermined found.

   7. A nested if statement to check If the 2nd line in Read2 does not equal the reverse complement of 2nd line in Read 3 (The rev_comp method gets called here). If true, then write the 4 lines from Read1 file, with the 2nd line concatenated with Read2 line and Read3 line into forward index hopped list File. Repeat the same code but with Read4 file and write the concatenated lines to reverse index hopped list. Have a counter to increment after every index hopped found.

   8. Else statement that writes the 4 lines from Read1 file, with the 2nd line concatenated with Read2 line and Read3 line into the forward fastq matching pair index file. Repeat the same code but with Read4 file and write the concatenated lines to reverse fastq matching pair index file. Have a counter to increment after every matching pair index found.

9. Break loop after all files are read.

10. Print statement for number of counts of undetermined found, index hopped found and matching pair index.