Part 2

The goal is to look through a lane of sequencing library preps and sort pooled experiments. 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 R1 and R2 fastq with matching index pairs. This 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. Also, output histograms.

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 52 fastq files R1 matching pair index file, R2 matching pair index file, R1 undertermine list file, R2 undertermine list file, R1 index hopped list file and R2 index hopped list file.

2.

a. Create a method to do a reverse complement:

Def rev_comp(seq)

"Takes a sequence and returns the reverse the complement"

-Create list for valid bases, a dictionary of complementary nucleotides and empty list for the result reverse complement sequence.

-Using a for loop check to see if the nucleotide in sequence is valid. If its valid, append the complementary base to the reverse complement list. After for loop is finished, change the reverse complement list into the reverse order and join the list into a string.

Return rev_seq

Input: ACGTACGG

Output: CCGTACGT

b. Create a method that takes a single character as it's parameter and converts that parameter into a phred score, and returns the phred score:

Def convert_phred(letter)

"Converts a single character into a phred score"

- Creates a variable to store a difference of -33 of the unicode point from the instance variable and return it.

Return rev_seq

Input: C

Output: 34

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 R1 undertermine list file. Repeat the same code but with Read4 file and write the concatenated lines to R2 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 R2 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 R1 index hopped list File. Repeat the same code but with Read4 file and write the concatenated lines to R2 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 R1 fastq matching pair index file. Repeat the same code but with Read4 file and write the concatenated lines to R2 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.