Demultiplexing and Index Swapping – Assignment the First

Goals: Our goal is to look through a lane of sequencing generated from the 2017 BGMP cohort's library preps and determine the level of index swapping and undetermined index-pairs, before and after quality filtering of index reads. In order to do this, we must first demultiplex the data. In Assignment the first, we will **develop a strategy** to de-multiplex samples to create 48 FASTQ files that contain acceptable index pairs (read1 and read2 for 24 different index pairs), two FASTQ files with index-hopped reads-pairs, and two FASTQ files undetermined (non-matching or low quality) index-pairs.

De-multiplexing is necessary for downstream analyses.

We submitted 24 indexed (dual matched) libraries. The indexes are:

B1	GTAGCGTA	A11	CTAGCTCA	C10	TCTTCGAC
A5	CGATCGAT	C7	CACTTCAC	A2	ATCATGCG
C1	GATCAAGG	B2	GCTACTCT	C2	ATCGTGGT
B9	AACAGCGA	A1	ACGATCAG	A10	TCGAGAGT
C9	TAGCCATG	B7	TATGGCAC	B8	TCGGATTC
C3	CGGTAATC	A3	TGTTCCGT	A7	GATCTTGC
B3	CTCTGGAT	B4	GTCCTAAG	B10	AGAGTCCA
C4	TACCGGAT	A12	TCGACAAG	A8	AGGATAGC

4 FASTQ files are:

```
      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
            in /projects/bgmp/shared/2017_sequencing/
```

Part 1 – Quality Score Distribution per-nucleotide

1. 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.
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 Leslie's class).
    a. Turn in the 4 histograms.
    b. What is a good quality score cutoff for index reads and biological read pairs to utilize for sample identification and downstream analysis, respectively?
    c. How many indexes have undetermined (N) base calls? (Utilize your command line tool knowledge. Submit the command you used. CHALLENGE: use a one-line command)

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

Write up a strategy (NOT A SCRIPT) for writing an algorithm to de-multiplexing files and reporting index-hopping. That is, given four input FASTQ files (2 with biological reads, 2 with index reads) and the 24 known indexes above, demultiplex reads by index-pair, outputting one forward FASTQ file and one reverse FASTQ file per matching index-pair, another two FASTQ files for non-matching index-pairs (index-hopping), and two additional FASTQ files when one or both index reads are unknown or low quality (do not match the 24 known indexes or do not meet a quality score cutoff). Add the sequence of the index-pair to the header of BOTH reads in all of your FASTQ files for all categories (e.g. add "AAAAAAAA-CCCCCCCC" to the end of headers of every read pair that had an index1 of AAAAAAAA and an index2 of CCCCCCCC; this pair of reads would be in the unknown category as one or both of these indexes doesn't match the 24 known indexes).

Additionally, your algorithm should report the number of read-pairs with properly matched indexes (per index-pair), the number of read pairs with index-hopping observed, and the number of read-pairs with unknown index(es). You should strive to report values for each possible pair of indexes (both swapped and dual matched). **You should not write any code** for this portion of the assignment. ***Be sure to***:

- Define the problem
- Determine/describe what output would be informative
- Write examples (unit tests!):
  - Include four properly formatted input FASTQ files with read pairs that cover all three categories (dual matched, index-hopped, unknown index)
  - Include the appropriate number of properly formatted output FASTQ files given your input files
- Develop your algorithm using pseudocode
- Determine high level functions
  - Description/doc string – What does this function do?
  - Function headers (name and parameters)
  - Test examples for individual functions
  - Return statement

Turn in:
Answers to questions, Python script for part 1, plots, and anything outlined in part 2 (NOT CODE!) to GitHub.