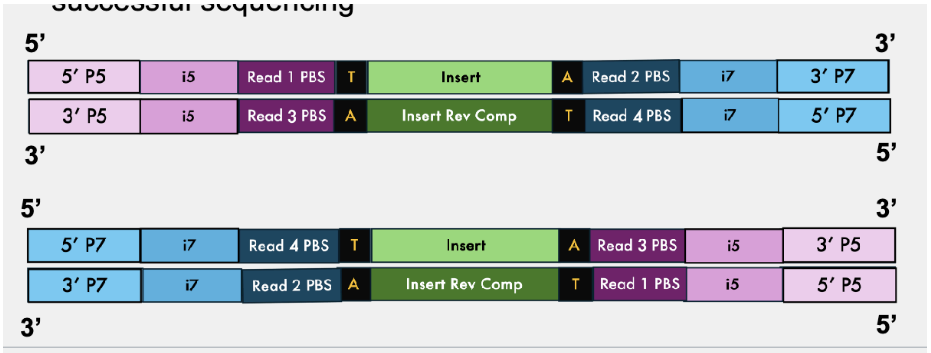
Demultiplexing Lab notebook

Day 1

Initial Data exploration

- Labeling files
    - Read1 & Read 4 are 2 Bio reads.
    - Read 2 & 3 are the 2 Index reads corresponding, 3 index 1(read1) and R3 file is index2 tied to R4 bio file
- Read Length -            head -1 <file> | wc -c. Whatever this is -1
- Phred Encoding - head -16 of a file. Look for the encodings of the quality scores, phred 66 will not have a lot of the symbol characters like # and < >. So I see a few of those and know its phred 33.

Sequence file layout

- 

- Line 1 of the R1 file will correspond to line 1 of the R2 file.

DISTRIBUTION:

- Basic concept / Pseudo Code
    - Open input files 1 at a time
        - Go through each line and each letter, add a phred score to the list. Also add 1 to an ongoing count
        - Take the mean of each entry of the list with count.
        - Plot this information
- Outputs
    - Testing outputs -> R#_test.png
    - Normal -> R#.png
- Script in Assignment the first directory. Using the R2 test file right now for debugging.

UNIT TEST: command below wIll run through every function, and output should be diffed to what i have in the TEST-output

Command for Demultiplexing testing: ./A3_the_third.py
-R1 ../TEST-input_FASTQ/Test_R1.fq.gz
-R2 ../TEST-input_FASTQ/Test_R2.fq.gz
-R3 ../TEST- input_FASTQ/Test_R3.fq.gz
-R4 ../TEST-input_FASTQ/Test_R4.fq.gz

-i ../TEST-input_FASTQ/indexes.txt
-o results

Command run for Demultiplexing final:

```
python3 /home/alho/bgmp/alho/bioinfo/Bi622/Demultiplexing/Assignment-the-third/A3_the_third.py
-R1 /projects/bgmp/shared/2017_sequencing/1294_S1_L008_R1_001.fastq.gz
-R2 /projects/bgmp/shared/2017_sequencing/1294_S1_L008_R2_001.fastq.gz
-R3 /projects/bgmp/shared/2017_sequencing/1294_S1_L008_R3_001.fastq.gz
-R4 /projects/bgmp/shared/2017_sequencing/1294_S1_L008_R4_001.fastq.gz
-i /projects/bgmp/shared/2017_sequencing/indexes.txt
-o results
-os stat_output.txt
```

Job Specifications:

```
#SBATCH --job-name="demultiplex"
#SBATCH --output='demultiu.out'
#SBATCH --account='bgmp'
#SBATCH --partition='bgmp'
```

Output of usr/bin/time -v run   for the cutoff results (there are two results, one for cutoff of 5 and 1 for no cutoff)

```
(base) login4 | alho | ~/bgmp/alho/bioinfo/Bi622/Demultiplexing/Assignment-the-third💰:cat demultiu_c5.out
/var/spool/slurm/job37084740/slurm_script: line 10: mamba: command not found
        Command being timed: "python3
/home/alho/bgmp/alho/bioinfo/Bi622/Demultiplexing/Assignment-the-third/A3_the_third.py -R1
/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R1_001.fastq.gz -R2
/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R2_001.fastq.gz -R3
/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R3_001.fastq.gz -R4
/projects/bgmp/shared/2017_sequencing/1294_S1_L008_R4_001.fastq.gz -i
/projects/bgmp/shared/2017_sequencing/indexes.txt -o results_cutoff5 -os stat_output_cutoff5.txt -c 5"
        User time (seconds): 4496.88
        System time (seconds): 90.88
        Percent of CPU this job got: 62%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 2:02:16
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 246756
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 1
        Minor (reclaiming a frame) page faults: 39879
        Voluntary context switches: 48827
        Involuntary context switches: 10770
        Swaps: 0
        File system inputs: 0
        File system outputs: 0
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
```

Strategy for Demultiplexing

- Initial requirements
    - Read1 file is Bio1, Read2 is index1, Read3 is Index2, and Read4 is Bio2
    - Each line number in all 4 files line up with the other line # in other files. So entry 8 from file 1 (READ 1), is from the same dna strand as entry 8 from file 4 (READ 4)
    - We have 24 indexes that will be the barcodes on the reads, each one of these will need 2 output files, 1 for the read1s and 1 for read2s.
    - Need to take the reverse compliment of read3 barcode, and reverse compliment of read4 sequence. For it to line up with first read & barcode
    - if the barcodes match within a single paired read, put the two reads into the indexes output files.
        - If both barcodes exist, but don't match, you put in an unmatched file, again with two output file 1 and 2 for the unmatched reads
            - For unmatched pairs, write the two indexes <idx1>-<idx2> at the end of the header for both reads
        - If one of the barcodes doesn't exist, put they two reads in the unknown two output files
        - For unmatched pairs
    - 48 index outputs + 2 unmatched outputs + 2 unknown outputs
    - STATS output total read-paris that matched, were unmatched, and were unknown
- Pseudo code

MAIN

- Getargs
    - all 4 files of reads
    - minimum qual score
    - Read length
    - Barcode file input
- Create dictionary of the swapped pairs possible
- Create the sums of the counts of paired, unpaired, unknown
- With open all 4 files
    - Take 4 lines from each file * store them in lists that are 4 length and named after each read
        - Take the reverse compliment of the index3 (with function below)
        - See if index 2 and index 3 match and if they both exist
        - Take the quality scores of the two indexes
            - If either one is below the threshold, then toss em both.
        - If they do and match, open output files with the name of the barcode ‘{barcodeName}_r1.fq’ and ‘{barcodeName}_r2.fq’
            - Write to R1 and R2

          - The header + indexes ‘<index>-<index>’
          - sequences
          - Quality scores
        - Count matched pairs +1
        - Write read 1 to R1, read 2 to R2
      - If they both exist but don't match, open the unmatched files: ‘unmatched_R1.fq’ and ‘unmatched_R2.fq’
        - Count to the dictionary of different unmapped pairs
        - Write to R1 and R2
          - The header + indexes ‘<index>-<index>’
          - sequences
          - Quality scores
        - Count unmatched pairs +1
      - If one or both don't exist, open the unknown output files: ‘unknown_R1.fq’ and ‘unknown_R2.fq’
        - Write to R1 and R2
          - The header + indexes ‘<index>-<index>’
          - sequences
          - Quality scores
        - Write to R1 and R2
        - Count unknown pairs +1
  - RETURN COUNTS OF mapped, unmapped, unknown, and sorted list of the unmatched pairs that are together

FUNCTIONS

  - Get args functions
    - Bring in the 4 files as 4 inputs
    - Minimum quality score
    - Barcode file name

  - Mean Convert fred quality score
    - Converts a sequence to a fred quality score

  - Implement reverse compliment function (takes in dna strand)
    - Returns the reverse compliment

TEST FILES

- Input
  - Finished input CHA-CHING
  - Have 4 records for each of the 4 read files.
    - Sequences are random valid sequences for each of them
    - 1 case where the indexes match correctly

        - 1 case where they are both valid, but don't match
        - 1 case where one is not valid, should be unknown
        - 1 case where the quality score is low for one of them (9 mean), even though the barcodes are good and match
    - Also have indexes input file that is valid for 2 barcodes
- Output
    - Need to make output

Statistics we need to track

- 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).
- Percentage of reads from each sample
- Overall amount of index swapping
- Any figures/any other relevant data your code output