demultiplex lab notebook

Folders on talapas:

- cd /projects/bgmp/shared/2017_sequencing —> location of the 2017 BGMP cohort's library preps (R1,R2,R3,R4)
- cd /projects/bgmp/ewi/bioinfo/Bi621/PS/Demultiplex —> my github repo

07/24/2025

purpose of demultiplex:

- we can group different samples with different barcodes so we can mix them and read them all together at the same time
- demultiplexing is computationally seperating each read

demultiplexing should:

- report index matches — the barcodes from r2 and r3 match
- report unknown indexes — barcodes from r2 or r3 either aren't in the database or have too low of a quality score
- index swapping — the barcodes from r2 and r3 both exist in the database, but are not equal (not supposed to happen!)
- to determine how much of each case there is, we must first demultiplex our data!

Develop a strategy to de-multiplex samples to create 52 fastq files

- 48 fastq filesthat contain acceptable index pairs (read1 and read2 for 24 different index pairs)
    - dual matched indexes (same index for read n)
        - index1_R1.fq
        - index1_R2.fq
- 2 fastq files with index-hopped read-pairs
- 2 fastq files undetermined (non-matching or low quality) index-pairs

Initial Data Exploration:

Step 1) look to see if they are properly formatted

- zcat — view and diplay the contents of a compressed file
    - zcat <filename> | wc -l — view the number of lines in a compressed file
- zcat 1294*.gz | wc -l —> 5811947760 lines for ALL 4 FILES
- zcat 1294_S1_L008_R1_001.fastq.gz | wc -l —> 1452986940 for FILE 1
    - 1452986940 × 4 = 5,811,947,760, so all files have the same number of lines

Step 2) print a sample 1st record from each file:
Read 1 command used: zcat 1294_S1_L008_R1_001.fastq.gz | head -4
@K00337:83:HJKJNBBXX:8:1101:1265:1191 1:N:0:1
GNCTGGCATTCCCAGAGACATCAGTACCCAGTTGGTTCAGACAGTTCCTCTATTGGTT
GACAAGGTCTTCATTTCTAGTGATATCAACACGGTGTCTACAA
+
A#A-<FJJJ<JJJJJJJJJJJJJJJJJFJJJJFFJJFJJJAJJJJ-
AJJJJJJJFFJJJJJJFFA-7<AJJJFFAJJJJJF<F--JJJJJJF-A-F7JJJJ

Read 2 command used: zcat 1294_S1_L008_R2_001.fastq.gz | head -4
@K00337:83:HJKJNBBXX:8:1101:1265:1191 2:N:0:1
NCTTCGAC
+
#AA<FJJJ

Read 3 command used: zcat 1294_S1_L008_R3_001.fastq.gz | head -4
@K00337:83:HJKJNBBXX:8:1101:1265:1191 3:N:0:1
NTCGAAGA
+
#AAAAJJF

Read 4 command used: zcat 1294_S1_L008_R4_001.fastq.gz | head -4
@K00337:83:HJKJNBBXX:8:1101:1265:1191 4:N:0:1
NTTTTGATTTACCTTTCAGCCAATGAGAAGGCCGTTCATGCAGACTTTTTTAATGATTT
TGAAGACCTTTTTGATGATGATGATGTCCAGTGAGGCCTCCC
+
#AAFAFJJ-----F---7-<FA-F<AFFA-JJJ77<FJFJFJJJJJJJJJJAFJFFAJJJJJJJJFJF7-
AFFJJ7F7JFJJFJ7FFF--A<A7<-A-7--

Step 3) determine the lengths of the reads in each file (subtract 1 from each wc to account for the newline character)
Read 1 command used: zcat 1294_S1_L008_R1_001.fastq.gz | head -2 | tail -1 | wc

- 102 - 1 = 101

Read 2 command used: zcat 1294_S1_L008_R2_001.fastq.gz | head -2 | tail -1 | wc

- 9 - 1 = 8

Read 3 command used: zcat 1294_S1_L008_R3_001.fastq.gz | head -2 | tail -1 | wc

- 9 - 1 = 8

Read 4 command used: zcat 1294_S1_L008_R4_001.fastq.gz | head -2 | tail -1 | wc

- 102 - 101 = 1

Concluding notes:
Mainly did some initial data exploration for assignment the first, and copied the bioinfo package into my folder for importing.

07/28/2025

Histogram generation

IMPORTANT:

- Each fastq file is very large, and there will be a very long runtime to parse the data for histogram.
- To solve this, create 2 files:
  - One to initialize and save the data from R1, R2, R3, R4 —> will only run once
    - histogram_generator_part1.py
  - One to plot the data saved on a histogram —> can run as many times as you like
    - histogram_generator_part2.py

```
Command being timed: "./histogram_generator_p1.py"
    User time (seconds): 26026.95
    System time (seconds): 4.76
    Percent of CPU this job got: 99%
    Elapsed (wall clock) time (h:mm:ss or m:ss): 7:15:13
    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): 91976
    Average resident set size (kbytes): 0
    Major (requiring I/O) page faults: 0
    Minor (reclaiming a frame) page faults: 166250
    Voluntary context switches: 1199
    Involuntary context switches: 6915
    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: 1
```

Data exploration

How many indexes have undetermined (N) base calls?

- echo $(($(zcat 1294_S1_L008_R2_001.fastq.gz | sed -n '2~4p' | grep -c 'N') + $(zcat 1294_S1_L008_R3_001.fastq.gz | sed -n '2~4p' | grep -c 'N'))) —> 7304664 undetermined base calls

Concluding notes:
Created and finished the histograms and the .py files needed. I used a shell script for the first data initialization script to submit an sbatch for it. The runtime for data generation came to 7 hours and 15 minutes.

07/29/2025

Creating test files
Test files by record (e.g seq1 is record1 for R1,R2,R3,R4):

- seq1 — match GTAGCGTA (1)
- seq2 — match CTCTGGAT (2)
- seq3 — match TCGGATTC (3)
- seq4 — match TGTTCCGT (4)
- seq 5 — match GATCAAGG (5)
- seq6 — match ATCATGCG (6)
- seq7 — unknown (quality score in R2 does not meet cutoff) GCTACTCT
- seq8 — hop TCGAGAGT
- seq9 — match TATGGCAC (7)
- seq10 — hop TACCGGAT
- seq11 — unknown (quality score in R3 does not meet cutoff)GTAGCGTA
- seq12 — unknown (unknown index for R2) GTAGCGTA
- seq13 — hop GTCCTAAG
- seq14 — match GTAGCGTA (1)
- seq15 — match GTAGCGTA (1)

quality scores R2:
1: 31.625
2: 32.25
3: 31.75
4: 30.5
5: 32.375
6: 36.625
7: 14.375
8: 32.875
9: 36.625
10: 32.875
11: 32.875
12: 32.875
13: 32.875

14: 32.875
15: 32.875

quality scores R3:
1: 31.625
2: 32.25
3: 31.75
4: 30.5
5: 32.375
6: 36.625
7: 32.875
8: 32.875
9: 36.625
10: 32.875
11: 16.625
12: 32.875
13: 32.875
14: 32.875
15: 32.875

Concluding notes:
Created and finished the test files. I have 9 matches, 3 unknowns, and 3 index hops. I tried creating a unit test to assert that each files contents are exactly as the test files, but there was a weird thing where a sample code created files were inserting \n newline characters (as expected), but I'm not able to write them into the test files. Because of this, I will visually check if the outputs of the demultiplexer are the same as the test files.

07/31/2025

Concluding notes:
Created and finished the demultiplexer. The demultiplexer takes in 3 arguments: the path to the input directory, name of the output directory (will create one if not found), and the quality score cutoff. I did two trials, one where there is no cutoff (cutoff = 0), and one where the cutoff = 30. After demultiplexing the files, it prints the counts of each match, counts of each index hop, and number of unknown indexes. It also prints the percentage of matches, hops, and unknowns. Below are the time logs for each trial and its corresponding output (i'll only include the percentages here, the entire outputs are rather long):

Cutoff = 0

Command being timed: "./demultiplexing.py -O /projects/bgmp/ewi/bioinfo/Bi621/PS/Demultiplex-/output_demultiplex_NO_CUTOFF -I /projects/bgmp/shared/2017_sequencing/ --qual_score_cutoff_indexes 0"

User time (seconds): 3838.83

System time (seconds): 112.31
Percent of CPU this job got: 93%
Elapsed (wall clock) time (h:mm:ss or m:ss): 1:10:10
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): 246736
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 39266
Voluntary context switches: 47837
Involuntary context switches: 1210
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

% index matches: 91.33049275721639
% index hops: 0.19483726398807136
% index unknown: 8.474669978795541

Cutoff = 30

Command being timed: "./demultiplexing.py -I /projects/bgmp/shared/2017_sequencing/"
User time (seconds): 3814.49
System time (seconds): 112.77
Percent of CPU this job got: 94%
Elapsed (wall clock) time (h:mm:ss or m:ss): 1:09:19
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): 246568
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 39776
Voluntary context switches: 48773
Involuntary context switches: 1433
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

% index matches: 84.0336396708
% index hops: 0.05957546776
% index unknown: 15.9067851369