Ben Cosgrove

Leslie

Bi622

6 August 2019

Demultiplexing

1.

|  |  |  |  |
| --- | --- | --- | --- |
| 1294\_S1\_L008\_R1\_001.fastq.gz | 1294\_S1\_L008\_R4\_001.fastq.gz | 1294\_S1\_L008\_R2\_001.fastq.gz | 1294\_S1\_L008\_R3\_001.fastq.gz |
| Read 1 | Read 2 | Index 1 | Index 2 |

2.b. The lowest mean scores seem to fall around 32. While having the median score would be more effective in estimating a proper cutoff, I feel that setting the cutoff for 32 would be reasonable. This may be too high of a cutoff as roughly a quarter of the reads seem to be excluded.

c. Using zcat $filename | paste - - - - | cut -f2 |grep -e 'N'|wc -l, I found index 1 to have 3976613 reads with a N and index 2 to have 3328051 reads containing a N.

Demultiplexing algorithm

The problem: Since Illumina flow cells have millions of reads per lane, it becomes feasible to run multiple experiments in a single lane. Each of these experiments has a unique barcode associated with it, however Illumina outputs a file containing all reads. Here we need to create a total of 52 files, one forward and one reverse file for each of 24 barcodes, for index hopped reads, and for reads that don’t meet a quality cutoff. The barcodes also need to be appended to the end of each headers, and a counter for each of the types of reads is necessary.

I believe the most informative output would be a bar graph that presents comparative counts for each barcode, unknown reads, and index hopped reads in addition to the 52 files.

Examples: 4 test.fq files are included along with 6 expected output .fq files, forward and reverse match for barcode, index hopped, and unknown. An expected graph is also included. Please note: I would use my srun script to zip the files at the end of demultiplexing, I left them open to be easier to read. All test files are contained in mpxTest directory

Pseudocode:

def get\_args():  
 #get file paths for read 1, read 2, index 1, and index 2

def revcomp(seq):  
 #revcomp creates a reverse compliment of a given string of nucleotides, returns the reverse compliment

in: ATCG out: CGAT

def barcodes(b):  
 #takes barcodes from a tab separated file and returns an array of known barcodes

in: barcodes tsv out: barcodes in array

def qual(q,seq):  
 #takes a quality score and checks if the mean phred score of a given score line is greater and returns true if it is  
 #and false if not

in: 32, LLLLLL out: True

def header(header,i1,i2):  
 #appends \_i1\_i2 to the end of the header line, returns the adjusted header

in:Hell,o,world out:Hell\_o\_world

def demult(r1,r2,i1,i2,bar,q):  
 #takes read and index files record by record, checks if indexes contain an N, and if it does places it in low quality,  
 #checks if average read quality is above or equal to user input threshold, if not places record in low quality files,  
 #checks if index 1 is the reverse compliment of index two, if not places in unmatched index files, if true places  
 #in that barcode file. Also includes counters for each file type in a dictionary. All headers are appended to include the barcodes at the start. Returns counts for each index, unknown indexes, and index hopped indices, a graph of these values, as well as creating files

in: four files, barcodes from tsv, and quality score, out: 50 files, counts of each barcode and a graph of those counts