Ben Cosgrove

Leslie

Bi622

7 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. Quality score cutoff should vary heavily based on the type of experiment. For index files, the score cutoff should be very high. For biological reads the cutoff may be low depending on the experiment. In an effort to be fairly stringent, I used the lowest mean score of around 32 as a cutoff. 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. First, simlinks were created in the mpx file on talapas. Using zcat 129\* | paste - - - - | cut -f2 |grep -e 'N'|wc -l, I found index 7304664 reads to contain an 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, index 2 , barcode File, quality score’’’

Get all necessary files and set them to variables for use later

def revcomp(seq):

‘’’revcomp creates a reverse compliment of a given string of nucleotides, returns the reverse compliment ‘’’

in: ATCG out: CGAT

Reverse the string, replace with the complimentary base, return the new string

def barcodes(b):

‘’’takes barcodes from a tab separated file and returns an array of known barcodes’’’

in: barcodes tsv out: barcodes in array

opens the tsv, finds the barcodes, adds them to an array and returns it

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

Converts ascii characters to phred scores, averages them across the full read, and see’s if its greater than or equal to the user input quality score

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

add index 1 and index 2 to the end of the header line, return the new header

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

open the four given read files. Initialize a dictionary with unknown:0,index hopped 0. read one line of each file, record one record, check if all reads pass quality scores. If they don’t open two files and write R1 to unknown forward and R2 to unknown reverse with changed headers increase unknown key by one. If they do check if the barcodes match and are in barcodes, if they do open two files and write R1 to barcode name forward and R2 to barcode reverse with changed headers and check if the barcode is in dictionary. If it is, increment by one, if its not add it and set value to one. If they don’t match check if both bar codes are in barcodes and if they are open two files and write R1 to index hopped forward and R2 to index hopped reverse with changed headers and increment index hopped by one. If they don’t open two files and write R1 to unknown forward and R2 to unknown reverse with changed headers and increment unknown key by 1. Open index hopped reads. Create a dictionary of dictionaries that contains all possible index pairs. Add counts from the counts dictionary for barcodes at each of the double barcode locations. Iterate through the index hopped file and increment keys when they occur. Output this to matplotlib to produce a heatmap