Garth Kong

BI 622

8/8/2018

Demultiplex

1. R1 and R4 files contain the paired-end reads.

R2 and R3 files contain the indexes.

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

1. Figures for average Q-score per nucleotide base position.

b. A quality score threshold for indices would be 30 because 40 would be too stringent of a filter, while anything under 30 would be too permissive. Although the average quality score is around 30 for the first 2 nucleotides, the quality score threshold will remove the quality scores < 30 and as a result raise the average score for the first 2 nucleotide positions higher. This will likely yield a more uniform, high-quality score per base pair position for better confidence in the data and its implications.

c. To obtain the number of indices with at least one N in its base calls, the following commands were used:

**# Number of indices in file 2 with N is: 3976613, or 1.10% of all reads**

zcat 1294\_S1\_L008\_R2\_001.fastq.gz | sed -n '2~4p' | grep "N" | wc -l

**# Number of indices in file 3 with N is: 3328051, or 0.92% of all reads**

zcat 1294\_S1\_L008\_R3\_001.fastq.gz | sed -n '2~4p' | grep "N" | wc -l

Part 2 Demultiplex Pseudocode

Problem: samples in Illumina reads are multiplex together into one lane, but identified by separate barcodes. However, prior studies have shown there has been a phenomenon called ‘index hopping,’ where the barcodes are switched between different (e.g. treatment) groups. We want to demultiplex the data and find the rate of index hopping.

Desired output: reads separated in both the forward or reverse direction with respect to both forward and reverse bar codes. All barcodes that do not meet a certain quality score threshold and/or have ambiguous sequences will be binned in a separate file separated by forward or reverse direction.

Unit Test:

Unit test files are submitted along with this document.

Key to unit test files: r1, r2, r3, r4 correspond to 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, respectively.

Summary of expected outcome is described below:

Number of input files: 4

Number of output files: 10

Expected output files:

Read1\_AAAAAAAA\_TTTTTTTT.fq

Read2\_AAAAAAAA\_TTTTTTTT.fq

Read1\_TTTTTTTT\_AAAAAAAA.fq

Read2\_TTTTTTTT\_AAAAAAAA.fq

Read1\_CCCCCCCC\_GGGGGGGG.fq

Read2\_CCCCCCCC\_GGGGGGGG.fq

Read1\_GGGGGGGG\_CCCCCCCC.fq

Read2\_GGGGGGGG\_CCCCCCCC.fq

Read1\_ambiguous.fq

Read2\_ambiguous.fq

Expected contents of files

**Read1\_AAAAAAAA\_TTTTTTTT.fq**

|  |
| --- |
| @header\_AAAAAAAA\_TTTTTTTT |
| GGGG |
| + |
| FFFF |

**Read2\_AAAAAAAA\_TTTTTTTT.fq**

|  |
| --- |
| @header\_AAAAAAAA\_TTTTTTTT |
| CCCC |
| + |
| FFFF |

**Read1\_TTTTTTTT\_AAAAAAAA.fq**

|  |
| --- |
| @header\_TTTTTTTT\_AAAAAAAA |
| GGGG |
| + |
| FFFF |

**Read2\_TTTTTTTT\_AAAAAAAA.fq**

|  |
| --- |
| @header\_TTTTTTTT\_AAAAAAAA |
| CCCC |
| + |
| FFFF |

**Read1\_CCCCCCCC\_GGGGGGGG.fq**

|  |
| --- |
| @header\_CCCCCCCC\_GGGGGGGG |
| AAAA |
| + |
| FFFF |

**Read2\_CCCCCCCC\_GGGGGGGG.fq**

|  |
| --- |
| @header\_CCCCCCCC\_GGGGGGGG |
| TTTT |
| + |
| FFFF |

**Read1\_GGGGGGGG\_CCCCCCCC.fq**

|  |
| --- |
| @header\_GGGGGGGG\_CCCCCCCC |
| TTTT |
| + |
| FFFF |

**Read2\_GGGGGGGG\_CCCCCCCC.fq**

|  |
| --- |
| @header\_GGGGGGGG\_CCCCCCCC |
| AAAA |
| + |
| FFFF |

**Read1\_ambiguous.fq**

|  |
| --- |
| @header\_AAAAAAAN\_TTTTTTTT |
| GGGG |
| + |
| FFFF |
| @header\_TTTTTTTT\_AAAAAAAN |
| CCCC |
| + |
| FFFF |
| @header\_NAAAAAAA\_NTTTTTTT |
| CCCC |
| + |
| FFFF |
| @header\_CCCCCCCC\_GGGGGGGG |
| AAAA |
| + |
| FFFF |
| @header\_CCCCCCCC\_GGGGGGGG |
| TTTT |
| + |
| FFFF |
| @header\_TTTTTTTT\_AAAAAAAA |
| GGGG |
| + |
| FFFF |

**Read2\_ambiguous.fq**

|  |
| --- |
| @header\_AAAAAAAN\_TTTTTTTT |
| CCCC |
| + |
| FFFF |
| @header\_TTTTTTTT\_AAAAAAAN |
| GGGG |
| + |
| FFFF |
| @header\_NAAAAAAAA\_NTTTTTTTT |
| GGGG |
| + |
| FFFF |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| @header\_CCCCCCCC\_GGGGGGGG |
| TTTT |
| + |
| FFFF |
| @header\_CCCCCCCC\_GGGGGGGG |
| AAAA |
| + |
| FFFF |
| @header\_TTTTTTTT\_AAAAAAAA |
| CCCC |
| + |
| FFFF |

Pseudocode for demultiplexing DNA

#!/usr/env/bin python3

Import gzip

Define a reverse complement function

Function header: def reverse\_complement(sequence)

Description: Reverse\_complement() takes sequence input (string) and returns the complementary sequence (string).

Example: Input of “ATGC” sequence

Return: returns “TACG” output

Define convert\_phred(x) function

Function header: def convert\_phred(x)

Convert\_phred(x) takes a character from the quality score line and returns it’s phred score on 33 Illumina scale.

Example: convert\_phred(“C”), convert\_phred(“D”), convert\_phred(“E”)

Return: return 34, 35, 36 as integer

Define barcodes with dictionary

Key = barcode code and value = barcode sequence.

Example: key = A1, B1, C1, D1, etc. and value = AAAAAAAA, TTTTTTTT, GGGGGGGG, CCCCCCCC, etc., respectively.

Define bad quality score

Example: bad quality score = 30

Define no\_index\_hop with dictionary

Key = barcodes from file 2 and 3 and value = count for number of matching complementary strands (inferred as no index hopping, assuming the reads pass other filters too).

Example: AAAAAAAA\_TTTTTTTT : 6, TTTTTTTT\_AAAAAAAA : 2,

GGGGGGGG\_CCCCCCCC : 8, CCCCCCCC\_GGGGGGGG : 3

Note: all pair of barcodes must be reverse complement to each other. A:T and G:C.

Define yes\_index\_hop with dictionary

Key = barcodes from file 2 and 3 and value = count for number of index hopping.

Example: AAAAAAAA\_TTTTGGCA : 15, AAAAAAAA\_NTTTTTTTT : 25, NAAAAAAA\_NTTTTTTTT : 4

Note: bad QS, N in barcode, barcode mis-match, will fall into this if they don’t pass the filters.

Gzip open all 4 files:

While True

Assign lines to variables with file.readline command across all 4 files.

# filter out bad barcode sequences

If ‘N’ in barcode sequence of file R2:

Print read information from R1 into r1\_ambiguous file

Print read information from R4 into r2\_ambiguous file

If ‘N’ in barcode sequence of file R3:

Print read information from R1 into r1\_ambiguous file

Print read information from R4 into r2\_ambiguous file

# filter out bad quality scores

If a quality score in r2 less than or equal to bad quality score threshold:

Print read information from R1 into r1\_ambiguous

Print read information from R4 into r2\_ambiguous

Elif a quality score in r3 less than or equal to bad quality score threshold:

Print read information from R1 into r1\_ambiguous file

Print read information from R4 into r2\_ambiguous file

# after passing through all the filters

If barcode 2 == reverse\_complement(barcode 3): # because indices are the same on both ends, they should be the reverse complement of each other.

Print read information from R1 into r1\_barcode2\_barcode3

Print read information from R4 into r2\_barcode2\_barcode3

Update the no index hopping dictionary respect to the pair of barcodes.

Else:

Print read information from R1 into r1\_barcode2\_barcode3

Print read information from R4 into r2\_barcode2\_barcode3

Update the yes index hopping dictionary respect to the pair of barcodes.