Maddy Griswold

Demultiplexing

|  |  |
| --- | --- |
| **File** | **Type** |
| R1 | Read1 |
| R2 | Index1 |
| R3 | Index2 |
| R4 | Read2 |

Line count (Anna ran) – (zcat 1294\*R1\* | wc -l ) 1,452,986,940

Read count = (1,452,986,940/4) = 363,246,735

Reads with N - zcat /projects/bgmp/shared/2017\_sequencing/1294\_S1\_L008\_R(2/3)\_001.fastq.gz | sed -n '2~4p' | grep "N" | wc -l

R2: 3,976,613

R3: 3,328,051

Quality Score Overall Means:

R1: 39.3156853325 R2: 36.1485259059 R3: 34.9760626854 R4: 37.4276004263

Quality Score Cut Off: 32

**Problem**:

Many samples are run on machines at a time. To identify different samples, barcodes or indices are added to each sample. Identifying these different samples are called demultiplexing. There is a phenomenon called index hopping where indices ‘hop’ to different samples. This algorithm needs to A) demultiplex and B) determine the rate of index hopping from dual indexed pair-end reads.

**Output**:

* 2 fastq files with matching indices, 1 reverse, 1 forward (per index, 48 total)
* 2 fastq files with unmatched indices and any read with low quality or undetermined reads
* Return read count per index and percent line hopping

**Unit Tests**:

**Pseudocode**:

file = original

finalFile – original - IH - N - QS

Open all files and be on the same line in each file

For line in files:

Read four lines at a time of read files. Save each line to an array, read1 and read2

[0] header, [1] sequence, [2] +, [3] quality scores

Read four lines at a time of index files. Save only the sequence (index) line as index1 and index2

Concat corresponding index to + line of read

If either index contains an N

Write both reads to the respective bad file and concat N onto the + line

Else if indexes don’t match (reverse reciprocal)

Write both reads to the respective bad file and concat IH for index hopping onto the + line

Else if the mean quality score is lower than the deemed cutoff

Write both reads to the respective bad file and concat QS onto the + line

else

Write to the best file corresponding to the index (index.f/r.txt)

**Functions:**

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: matching

Description: to compare two indexes and determine if they are the same (index2 reverse compliment of index1

Parameters: index1 – index from first read

Index2 – index from second read

Returned: matched – bool

TRUE – if matched

FALSE – if not matched

Test Case: T1 = matching("AAAAAAAA", "TTTTTATT")

T2 = matching("AAAAAAAA", "AAAAAAAA")

T3 = matching("AAAAAAAA", "TTTTTTTT")

T4 = matching("NGTTCCGT", "NCGGAACA")

T5 = matching("GATTACAA", "TTGTAATC")

T6 = matching("AAAAAAAA", "TAAAAAAA")

Results:

T1 = FALSE

T2 = FALSE

T3 = TRUE

T4 = TRUE

T5 = FALSE

T6 = FALSE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: concatIndex

Description: concatenate index onto the “+” line of the sequence read

Output –

Header

Sequence

+index

Quality Scores

Parameters: read – read to add index to

Index – index to add to read

Returned: None

Test Case: read1 = [@headerline, CATACTGGTCA , +, DDDDDDDDDDD]

read2 = [@headerline2, CATACTCAGTA, +, ;;;;;;;;;;;]

read3 = [@headerline3, ACGCTGCGATC, +, HHHHHHHHHHH]

index1 = AAAAAAAA

index2 = TTTTTTTT

index3 = GGGGGGGG

read1 = concatIndex(read1, index1)

read2 = concatIndex(read2, index2)

read3 = concatIndex(read3, index3)

Results:

read1 = [@headerline, CATACTGGTCA , +NAAAAAAA, DDDDDDDDDDD]

read2 = [@headerline2, CATACTCAGTA, +TTTTTTTT, ;;;;;;;;;;;]

read3 = [@headerline3, ACGCTGCGATC, +GGGGGGGG, HHHHHHHHHHH]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: writeToBadFile

Description: write reads that don’t meet standards to file

Reason concatenated to +index line in file

Parameters: outputFile – fastq file that the reads are being written to

read – the read needing to be put in output file

reason – reason for being output, concat onto +index line in []

options:

QS – low quality score

N – index contains Ns

IH – index hopping

Returned: None

Test Case: outfile = "badFile.txt"

read1 = [@headerline, CATACTGGTCA , +NAAAAAAA, DDDDDDDDDDD]

read2 = [@headerline2, CATACTCAGTA, +TTTTTTTT, JJJJJJJJJJJ]

read3 = [@headerline3, ACGCTGCGATC, +GGGGGGGG, HHHHHHHHHHH]

writeToBadFile(outfile, read1, "N")

writeToBadFile(outfile, read2, "QS")

writeToBadFile(outfile, read3, "IH")

Results:

@headerline

CATACTGGTCA

+NAAAAAAA N

DDDDDDDDDDD

@headerline2

CATACTCAGTA

+TTTTTTTT QS

;;;;;;;;;;;

@headerline3

ACGCTGCGATC

+GGGGGGGG IH

HHHHHHHHHHH

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: writeToGoodFile

Description: write reads that meet standards to file

Parameters: outputFile – fastq file that the reads are being written to

read – the read needing to be put in output file

Returned: None

Test Case: outfile = "goodFile.txt"

read1 = [@headerline, CATACTGGTCA , +NAAAAAAA, DDDDDDDDDDD]

read2 = [@headerline2, CATACTCAGTA, +TTTTTTTT, ;;;;;;;;;;;]

read3 = [@headerline3, ACGCTGCGATC, +GGGGGGGG, HHHHHHHHHHH]

writeToGoodFile(outfile, read1)

writeToGoodFile(outfile, read2)

writeToGoodFile(outfile, read3)

Results:

@headerline

CATACTGGTCA

+AAAAAAAA

DDDDDDDDDDD

@headerline2

CATACTCAGTA

+TTTTTTTT

;;;;;;;;;;;

@headerline3

ACGCTGCGATC

+GGGGGGGG

HHHHHHHHHHH

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: meanQS

Description: determine the average quality score in the read

Parameters: read – the read that contains the quality scores

Returned: mean – mean of quality scores (float)

Test Case: read1 = [@headerline, CATACTGGTCA , +NAAAAAAA, DDDDDDDDDDD]

read2 = [@headerline2, CATACTCAGTA, +TTTTTTTT, ;;;;;;;;;;;]

read3 = [@headerline3, ACGCTGCGATC, +GGGGGGGG, HHHHHHHHHHH]

QS1 = meanQS(read1)

QS2 = meanQS(read2)

QS3 = meanQS(read3)

Results:

QS1 = 35

QS2 = 26

QS3 = 39

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: compareMeans

Description: compare mean quality scores to designated cutoff

Parameters: mean – mean of quality scores of specific read

Returned: above – bool

TRUE – if above or equal to

FALSE – if below

Test Case: QS1 = 35

QS2 = 26

QS3 = 39

cutoffQS = 32

b1 = compareMeans(QS1)

b2 = compareMeans(QS2)

b3 = compareMeans(QS3)

Results:

b1 = TRUE

b2 = FALSE

b3 = TRUE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: containsN

Description: determine if the index contains an N

Parameters: index – the index of the read

Returned: N – bool

TRUE – if contains N

FALSE – if does not contain N

Test Case: index1 = NAAAAAAA

index2 = TTTTTTTT

index3 = GGGGGGGG

b1 = containsN(index1)

b2 = containsN(index2)

b3 = containsN(index3)

Results:

b1 = TRUE

b2 = FALSE

b3 = FALSE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ```

``` \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: convertPhred

Description: convert letter to phred score

Parameters: letter – quality score from fastq file

Returned: score – phred score value (int)

Test Case: phred1 = convertPhred("A")

phred2 = convertPhred("#")

phred3 = convertPhred("I")

Results:

phred1 = 32

phred2 = 2

phred3 = 40

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*