Garth Kong

BI 624

10/22/2018

De-Duper Pseudocode

Problem: PCR duplicates are defined as overly amplified DNA segments. PCR duplicates arise from the PCR amplification step during library prep for Next-Generation Sequencing (NGS), and they become problematic when specific sequences are overly represented. When homogenous samples are subjected to high-throughput sequencing, the resulting dataset will have an unnecessarily large number of the same DNA fragment. In the case of an RNA-Seq experiment, the overly represented DNA fragments will non-uniformly map to a reference genome and further confound data interpretation. Solving the problems of PCR duplicates will be the motivation for the development of De-Duper – a program that removes PCR duplicates from SAM files.

**Input file**

Input file is in the provided text file for better formatting.

**Output file**

Output file is in the provided text file for better formatting.

**Pre-processing**

Sort SAM files using samtools with respect to leftmost chromosomal position

$ Remove @ headers with Unix

**De-Duper**

#!/usr/env/bin python3

Argparse options.

-f (required) to select files

-p (optional) designates paired-end file

-u (optional) designates a file of UMI’s (unset if randomers use instead of UMI’s)

-h (optional) prints help messages

Define function to get UMI’s from input file: # get\_umi(umi\_file)

‘’’Opens the file containing the UMI’s and saves each of them to a list.’’’

Example: input text file contains “AAAAAAAA” “TTTTTTTT” “GGGGGGGG” “CCCCCCCC”. This approach assumes each UMI is separated by a new line.

Return: [“AAAAAAAA”, “TTTTTTTT”, “GGGGGGGG”, “CCCCCCCC”] this is a python list

Define next line checker function: # NL\_checker(line):

‘’’Returns True or False whether the next line contains certain elements equal to the previous line. Those certain elements are each columns of a SAM file. Must be a recursive loop.’’’

Example: input a line of a SAM file, and return True or False if next line contains same position, barcodes, reads, cigar, etc.

Example 1: line 1 has [QNAME, 0, 2, 500] and line 2 has QNAME, 0, 2, 500]

Example 2: line 1 has [QNAME, 0, 2, 500] and line 2 has [QNAME, 0, 2, 550]

Return: example 1 returns True, example 2 returns False.

get\_umi(umi\_file) # get the UMI’s and save them into a list.

While True: # traverses each line of the file

Initialize empty dictionary.

Read and split a line # input.readline()

QNAME = key, Col 2 – 11 = value. # refreshes at each duplicate.

If read is umapped:

Continue # ignore reads that are unmapped. Check by bitwise flag.

If ‘N’ in QNAME UMI of dictionary:

Continue # ignore barcodes with ambiguous nucleotides.

Set UMI as last 8 characters of the QNAME

If QNAME contains a UMI in provided list of UMI’s and not in dictionary:

Line\_counter = 0

Assign the next line to a variable. # to check for duplicate at second line.

While next line contains the same barcode and read:   
# traverses duplicates and captures them in adjacent lines.

NL\_checker() # recursively finds adjacent duplicates

Line\_counter += 1 # tells how many lines to stop.

If False:

print current dictionary key + values into new file   
# print the first duplicate into a new file.

Scan next 1K lines # accommodates soft-clipping in CIGAR

If (barcode and read is in next 1K lines) and ‘S’/ ‘N’/ ’D’ in CIGAR string:

Delete those lines. # so the while loop doesn’t pick   
 up those lines for comparison.

If line is blank

Break # end program