PCR_Duplicate

Sneha Challa

10/14/2021

PCR DUPLICATES

During RNA seq library preparation, PCR is done to add the sequencing adapters to the fragments and to amplify the fragments if the starting quantity is low. During this process, the same fragment might get amplified multiple times to produce PCR duplicates. It is important to differentiate molecular duplicates (indicating expression levels) from PCR duplicates to avoid downstream analysis issues. PCR duplication is best done after aligining to a reference genome. Adding an UMI to each fragment before the PCR step, helps to identify the PCR duplicates. Fragments with same UMI are considered PCR duplicates. The goal is to retain only one unique read.

Why is PCR duplicate a problem? - It incorrectly increases the coverage, giving a false confidence in the base call. - While variant calling, if PCR duplicates are not removed, we might incorrectly call a sequencing error as a true variant due to a false increase in coverage and hence confidence.

Identifying a PCR Duplicate PCR Duplicates that came from the same template: - have the same QNAME (UMI) - same Chrom - RNAME pos - POS strand - FLAG {multiple reads when aligned to the same start position = PCR Duplicate} - soft clipping

STRATEGY 1. Create a list of the 96 UMI's from the STL96.txt.
2. Adjust for soft clipped positions (if CIGAR has "S", pos= pos-n, where n is the number of nt to be adjusted)
3. Sort the SAM file by chromosome, position and UMI
4. Open the SAM file and iterate through the file reading one line at a time
5. The header lines starting with @ can be ignored or written to a new file For each read line in the alignment section:
6. Check for the strand position (if flag &16!=16) and write to two separate files - one file for stranded sorted reads and second file for unstranded sorted reads.
7. Store QNAME(UMI), POS, strand, RNAME(chrm) in different variables
8. Core Logic: Iterate through both files (stranded and unstranded):
a) if current_UMI(from variable) is in the list:
UMI += 1
else:
write to an unknown/garbage file.
b) concatenate the UMI, pos, strand from the 1st read and store in a list and write to the output file.
c) Compare the next reads with this list.
if same UMI,pos,strand:
skip the read to get unique reads in the output file. else:
write to the output file.
Add to the list and repeat c.

TEST FILES: *Input SAM File:*

NS500451:154:HWKTMBGXX:1:11101:24260:1121:CTGTTCAC 0 2 76814284 36 71M * 0 0 TCCACCA-
CAATCTTACCATCCTTCCTCCAGACCACATCGCGTTCTTTGTTCAACTCACAGCTCAAGTACAA
6AEEEEEEAEEAEEEEAAEEEEEEEEEAEEAEEAAEE<EEEEEEEEEAEEEEEEEAAEEAAAEAEEAEAE/

MD:Z:71 NH:i:1 HI:i:1 NM:i:0 SM:i:36 XQ:i:40 X2:i:0 XO:Z:UU NS500451:154:HWKTMBGXX:1:11101:18996:1145:CTGTTC
0 2 76814284 36 71M * 0 0 GTCTCTTAGTTTATTATAAACCAGCTTCATAGGCCACAGAGGAAAAAG-
GACTATATACATACAGCCTTTTG 6AEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEE
MD:Z:53G16 NH:i:1 HI:i:1 NM:i:2 SM:i:36 XQ:i:40 X2:i:0 XO:Z:UU NS500451:154:HWKTMBGXX:1:11101:25533:1187:GTTC
0 2 76743835 36 71M * 0 0 CTTGGTAACTTTCAGAGAATTAGTCACAACTTCTGAAGCAAC-
CACAGTCCATGCAAGTCGACTGGTTTCTC 6AEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEAEEEEE
MD:Z:71 NH:i:1 HI:i:1 NM:i:0 SM:i:36 XQ:i:40 X2:i:0 XO:Z:UU NS500451:154:HWKTMBGXX:1:11101:25533:1187:GTTCAC
0 2 76743835 36 2S71M * 0 0 CTTGGTAACTTTCAGAGAATTAGTCACAACTTCTGAAGCAAC-
CACAGTCCATGCAAGTCGACTGGTTTCTC 6AEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEAEEEEE
MD:Z:71 NH:i:1 HI:i:1 NM:i:0 SM:i:36 XQ:i:40 X2:i:0 XO:Z:UU

Output SAM File: NS500451:154:HWKTMBGXX:1:11101:24260:1121:CTGTTCAC 0 2 76814284 36
71M * 0 0 TCCACCACAATCTTACCATCCTTCCTCCAGACCACATCGCGTTCTTTGTTCAACT-
CACAGCTCAAGTACAA 6AEEEEEEAEEAEEEEAAEEEEEEEEEAEEAEEAAEE<EEEEEEEEEAEEEEEEEEAAEEAA
MD:Z:71 NH:i:1 HI:i:1 NM:i:0 SM:i:36 XQ:i:40 X2:i:0 XO:Z:UU

Functions: a) fn name: soft_clip_pos
fn description: corrects the soft clipped position of the CIGAR string input = CIGAR String , expected output = corrected CIGAR String return CIGARstring

b) fn name: strandedness fn description: takes a read and identifies if it is stranded or unstranded. Writes it to the respective file input = Read
expected output = two files return stranded_file, unstranded_file