Deduper Part 1

Kameron Bates

October 22nd, 2018

Define the problem

To prepare reads for sequencing, PCR amplification is required so that enough sample will be able for the sequencer to detect a read. During this PCR amplification process, some reads may be preferentially amplified compared to others. This is especially important when doing RNAseq as reads that were preferentially amplified will be considered to have increased level of expression due to transcript abundance but is only due to PCR amplification. We need to determine which transripts have high levels of expression and which are due to PCR duplication and eliminate those transcripts. This can be done by looking at Sam files. PCR duplication occurs if the start and stop positions(after adjsuting for soft clipping) of the transcript are the same, it is mapped to the same chromosome, it is on the same strand and finally if it has the same unique molecular identifier. We must remove any occurance of PCR duplication and return only one transcript for the given PCR duplication sequence.

Sample Sam files

Input file

NS500451:154:HWKTMBGXX:1:11101:24260:1121:CTGTTCAC 0 2 7681428436 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:1:11101:24260:1121:CTGTTCAC 0 2 7681428438 2S69M * 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:1:11101:24260:1121:ATCTATCT 0 5 7681428436 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:1:11101:24260:1121:CTGTTCAC 0 2 7681428436 62M * 0 0 TC-
CACCACAATCTTACCATCCTTCCTCCAGACCACATCGCGTTCTTTGTTCAACTCACAGCT
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:1:11101:24260:1121:ATGATGAT 0 2 7681428436 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

Output file

NS500451:154:HWKTMBGXX:1:11101:24260:1121:CTGTTCAC 0 2 7681428436 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:1:11101:24260:1121:ATCTATCT 0 5 7681428436 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:1:11101:24260:1121:CTGTTCAC 0 2 7681428436 62M * 0 0 TC-
CACCACAATCTTACCATCCTTCCTCCAGACCACATCGCGTTCTTTGTTCAACTCACAGCT
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:1:11101:24260:1121:ATGATGAT 0 2 7681428436 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

Pseudocode

```
1
# argparse function where user puts in file and can add addtional inputs of paired vs single read and u
# while true
  # if umi == True
    # create list for the 96 umis
    # creat dictionary whih key is umi and value is a dictionary which will store all unique reads whic

    # use readline
    # if UMI in UMI list
      # if umi_dict[umi] != {}
        # make readline a np array
        # for key in dict
          # turn in np array
          # call chrom checker
          # if true
            # call is_rev_comp
            #if true
              #call start stop checker
                # reads ==
                  # pcr duplicate do not write
                # else
                  # unique read
                  # write to output file
                  # add string to dictionary
            # else
              # unique read
              # write to output file
              # add string to dictionary
          # else
            # on different chrom clear umi dict
            # unique read
            # write to output file
            # add string to dictionary
      # else
        # unique read
        # write to output file
```

```
        # add string to dictionary

  # if umi == flase
    # make empty list for UMi
    # use readline
        # if umi in list
          # go through all steps as above
        # else
          # store umi in list
          # write to file as unique read
```

High level functions

```
1
# def chrom_checker():
  # looks at chromosome and compare
  # returns true if chrom are same
# def is_mapped():
  #looks at bitwise flag 4 to determine if read mapped
  # returns is mapped == True

# def is_rev_comp():
  # checks if is_mapped is True
  # looks at bitwise flag 16 to determine if read was reverse comp
  # returns is rev_comp == True

# def start_stop_checker()
  # looks at cigar string
  # if S in cigar
    # if at start
      # adjust start
    # if at end
      # adjust end
    # sets start and stop
  # checks for indels
    # if N in cigar
      #adjust end
    # if D in cigar
      # adjust end
    # if I in cigar
      # adjust end
    # sets start and stop
  # else
    # sets start and stop positions
  # returns start and stop
```