De-duper 1

David Ho

10/18/2017

Define the problem, write examples

During library prep for sequencing, generally PCR is used to amplify the starting library so there is enough DNA molecules to sequence. This is an intential part of the prep but it also means that some molecules are overrepresented because the are already high in abundant to start with or kinetically amplify better than others.

One way to identify these PCR duplicates and remove them so you are not biasing your analysis is to attach a Unique Molecular Identifier (UMI) or a randomer to the original library molecules before PCR. Reads that have the same UMI/randomer, align to the same left-most start position on the same chromosome, on the same strand are therefore duplicates of each other.

Develop your algorithm using pseudocode

Simple case, single-end, we have a known list of UMIs used in library prep – How do we know if something is a PCR duplicate? (1) Same UMI/Randomer (2) Same left-most start position (corrected by CIGAR if necessary) (3) Same chromosome

```
SAM --> BAM
sort BAM          ## by chromosome and bp start
BAM --> SAM


de_dupe.py (single-end)

argparse
    -f INPUT_SAM -u UMI_file

definition DUPE_CHECK(SAM_line):        ## a function to extract information necessary t
o determine PCR duplicate
    split SAM_line by field (tab)

    UMI = last part of 1st field        ## assuming the UMI is part of the first field

    POS = 4th field

    if 5th field (CIGAR) has "S":        ## if the CIGAR string is soft-clipped, adjust
 the POS
        POS = POS - #S

    CHROM = 3rd field                    ## maybe if we sorted with samtools, this does
n't matter...

    DUPE_CHECK = UMI_POS_CHROM           ## combine the info into a string



umi_dictionary                          ## store known UMIs in a dictionary
dict = {}                               ## initiate a dict

for line in INPUT_SAM:                    ## read in the sam file and go line by line af
ter the @s

    if line starts with @, write out to OUTPUT.sam   ## write out the headers

    if line does not start with @:
        DUPE_CHECK(line)                    ## pass each line through the function esta
blished before

        if UMI in umi_dictionary:           ## if UMI in dictionary...

            if DUPE_CHECK not in dict:           ## if DUPE_CHECK is not in the dict, th
is means that it's the first time we see it
                write out line to OUTPUT.SAM     ## write this line out to a new SAM fil
e
                clear dict                       ## clear the dictionary, because we sor
ted SAM file, then PCR dupes should be next to each other
                dict[DUPE_CHECK] = 1             ## add this line's DUPE_CHECK to the di
ct
            else
                continue                         ## if DUPE_CHECK is already IN the dic
```

```
t, then it is a PCR dupe, continue to the next line

        else continue                           ## if UMI not in dictionary, then skip l
ine
```

Determine high level functions

In my pseudocode, I have just one function in which the argument to pass through is a line of a SAM file. The function splits the line up into parts and assigns variables to each relavant part. The final thing it does is parses together a string of the relavant variables to compare to the next line. (maybe this is more like a `method` in Python…)

In actually writing the code, I might switch to writing multiple functions: one to get the UMI, one to check to the CIGAR string and edit the bp position, and one to look at the bitwise flag.

See example SAM files in the respository.