Reference-based PCR Duplicate Removal

Review

November 27, 2017

What does a PCR-duplicate look like?

- Same alignment position
  - Chromosome — **RNAME** (SAM col 3)
  - Position — **POS** (SAM col 4)
  - Strand (strand specific?) — **FLAG** (SAM col 2)
- Soft Clipping — **CIGAR** (SAM col 6)
- Same Unique Molecular Index (UMI or “randomer”) — **QNAME** (SAM col 1)
- Single-end vs Paired-end? — **FLAG** (SAM col 2)

SAM: alignment section mandatory fields

- SAM format: https://samtools.github.io/hts-specs/SAMv1.pdf

```
NS500451:154:HWKTMBGXX:1:11101:25391:8012321:N:0:GGCCTAAT 0 chr22_hg38
21875238 37 67M * 0
AAACCATCCTGTACCTGAGATTCATGGGGGTGGCTGTTCTCCAAATCCCACAATCCCGCAGGGAGAC
ABEAAEEEEEEEEEEEEEEEEFEEEEEEEECCCAAAEEEEEEEEEEEEEEEAEAEAEAEEAA
MD:Z:67 NH:i:1 HI:i:1 NM:i:0 SM:i:37 XQ:i:40 X2:i:0 XO:Z:UU XG:Z:A
```

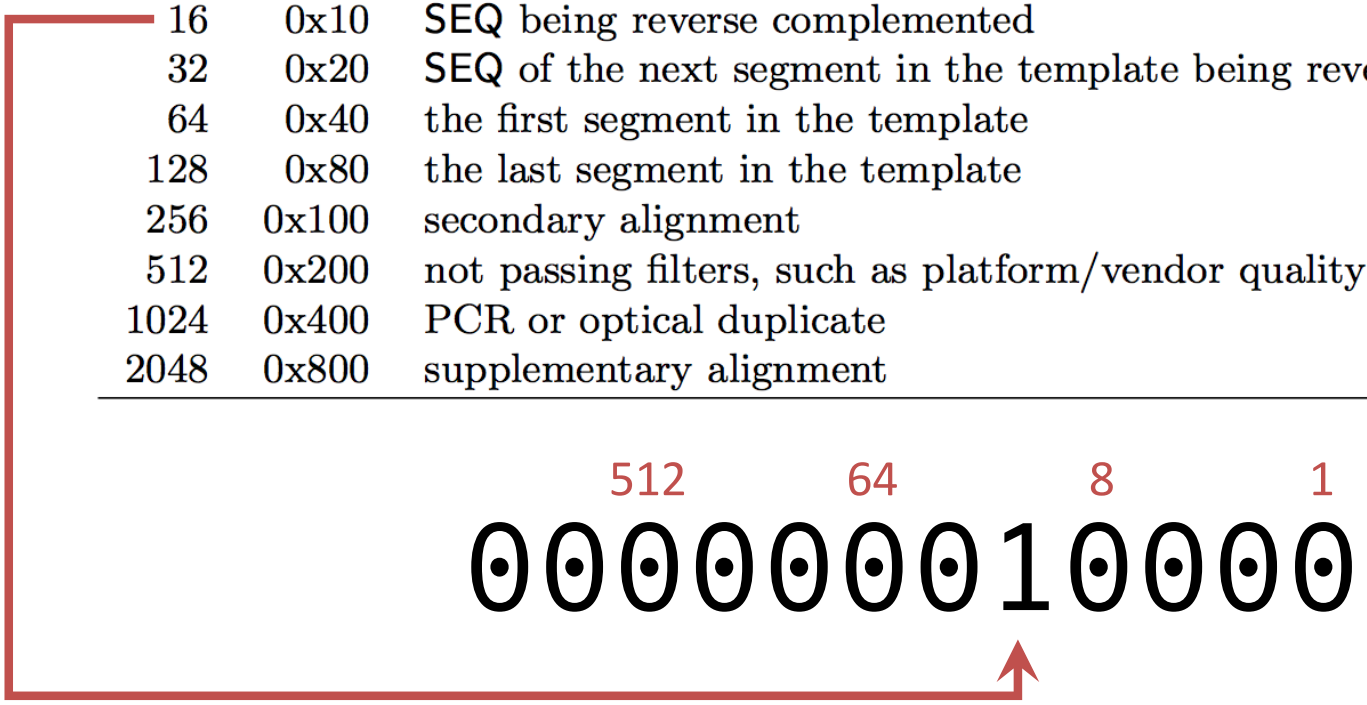
Col	Field	Type	Description
1	QNAME	String	Query template NAME
2	FLAG	Int	bitwise FLAG
3	RNAME	String	Reference sequence NAME
4	POS	Int	1-based leftmost mapping POSition
5	MAPQ	Int	MAPping quality
6	CIGAR	String	CIGAR string
7	RNEXT	String	Ref. name of the mate/next read
8	PNEXT	Int	Position of the mate/next read
9	TLEN	Int	observed Template LENgth
10	SEQ	String	segment SEQuence
11	QUAL	String	ASCII of Phred-scaled base QUALity+33

SAM: parsing the bitwise FLAG

https://broadinstitute.github.io/picard/explain-flags.html

```
if ((flag & 4) != 4):
    mapped = True
```

Bit		Description
1	0x1	template having multiple segments in sequencing
2	0x2	each segment properly aligned according to the aligner
4	0x4	segment unmapped
8	0x8	next segment in the template unmapped
16	0x10	SEQ being reverse complemented
32	0x20	SEQ of the next segment in the template being reverse complemented
64	0x40	the first segment in the template
128	0x80	the last segment in the template
256	0x100	secondary alignment
512	0x200	not passing filters, such as platform/vendor quality controls
1024	0x400	PCR or optical duplicate
2048	0x800	supplementary alignment



Soft clipping

- What is it?
- What does it look like?
- Why would something be soft clipped?
  - Sequence error/heterozygosity
  - Over-penalizing indels
  - Splicing with just a few nucleotides in an exon
  - Novel splicing
- Where in the alignment could soft clipping occur?

How do you know if your sequence was soft clipped? The CIGAR string!

CIGAR: CIGAR string. The CIGAR operations are given in the following table (set '*' if unavailable):

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

10M

```
...ATTGTCCATT...
ATTGTCCATT
```

10M

```
...ATTGTCCATT...
ATTGACCATT
```

2S8M

```
...ATTGTCCATT...
GGTGTCCATT
```

5M1024N5M

```
...ATTGTAGT...GCCCATT...
ATTGT         CCATT
```

Example CIGAR strings from real data

1S27M113N38M	13M1185N53M	23M686N43M	40M1I25M
1S36M284N29M	13M83N53M	25M246N41M	43M32325N23M
1S45M301N20M	13M85N53M	25M470N41M	44M5064N22M
1S61M4S	14M903N52M	27M2645N39M	46M2001N20M
1S64M1S	15M470N51M	28M138N38M	49M211N17M
2S31M745N33M	16M1540N50M	28M1794N38M	51M97N15M
2S62M2S	17M1097N49M	28M3349N38M	52M1198N14M
2S63M1S	18M12872N48M	29M2D37M	52M1904N14M
2S64M	19M20545N47M	30M1244N36M	54M12S
3S18M104N45M	20M2979N46M	31M1043N35M	54M407N12M
3S63M	20M456N46M	31M271N35M	55M954N11M
4S62M	20M631N46M	35M113N31M	56M10S
5S61M	21M982N45M	35M128N31M	57M5055N9M
6S42M284N18M	22M103N44M	35M289N31M	58M8S
6S60M	22M138N44M	35M5284N31M	62M4S
7S59M	22M4529N44M	36M103N30M	63M3S
11M4797N55M	23M1290N25M2D18M	36M12071N29M1S	65M1S
11M592N55M	23M15018N43M	36M1496N30M	66M

Your algorithm!

Given a SAM file of uniquely mapped reads, remove all PCR duplicates (retain only a single copy of each read)

- Samtools sort
- Adjust for soft clipping
- Single-end reads
- Known UMIs
- Considerations:
  - Millions of reads – avoid loading everything into memory!
  - Be sure to utilize functions appropriately
  - Appropriately commend code and include doc strings
- CHALLENGE: Include options for
  - Single-end vs paired-end
  - Known UMIs vs randomers
  - Choice of duplicate written to file

You MUST

- Write Python 3 compatible code
- Include the following argparse options
  - -f, --file required arg absolute file path
  - -p, --paired optional arg designates file is paired end (not single-end)
  - -u, --umi optional arg designates file has UMIs (not randomers)
  - -h, --help optional arg prints a USEFUL help message (see argparse docs)
    - If your script is not capable of dealing with a particular option (ex: no paired-end functionality), your script should print an error message and quit
- Output the first read encountered if duplicates are found
  - You may include an additional argument to designate output of a different read (highest quality or random or ???)
- Output a properly formatted SAM file with "`_deduped`" appended to the filename
- Name your python script `<your_last_name>_deduper.py`