Week 6, Lecture 11

István Albert

Biochemistry and Molecular Biology
and Bioinformatics Consulting Center

Penn State
An query file with changes
(modify increasing number of bases)
Create a script that runs the aligner

```bash
# producing alignment data with bwa is a two step process
#
# create the alignment (this is a binary file)
#!/bin/bwa aln refs/sc.fasta query.fq > query.sai

# format the alignment in a certain way
#!/bin/bwa samse refs/sc.fasta query.sai query.fq > results.sam
```
The structure of the SAM file

SAM Headers

Alignments
SAM format: tabular text format

Published as
The Sequence Alignment/Map format and SAMtools by Heng Li et al
Bioinformatics 25, Volume 25, Issue 16, 2009

“A TAB-delimited text format consisting of a header section, which is optional, and an alignment section.”
The majority of data analyses split off after generating a SAM file.
The SAM Format Specification (v1.4-r985)

The SAM Format Specification Working Group

September 7, 2011

1 The SAM Format Specification

SAM stands for Sequence Alignment/Map format. It is a TAB-delimited text format consisting of a header section, which is optional, and an alignment section. If present, the header must be prior to the alignments. Header lines start with '@', while alignment lines do not. Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.

1.1 An example

Suppose we have the following alignment with bases in lower cases clipped from the alignment. Read r001/1 and r001/2 constitute a read pair; r003 is a chimeric read; r004 represents a split alignment.

```
Coor  12345678901234  5678901234567890123456789012345
ref   AGCATGTTAGATAA**GATAGCTGTGCTAGTAGCAGTCAGC5CCAT
```
An alignment consists of 11 tab delimited columns.

### 1.4 The alignment section: mandatory fields

Each alignment line has 11 mandatory fields. These fields always appear in the same order and must be present, but their values can be ‘0’ or ‘*’ (depending on the field) if the corresponding information is unavailable. The following table gives an overview of the mandatory fields in the SAM format:

<table>
<thead>
<tr>
<th>Col</th>
<th>Field</th>
<th>Type</th>
<th>Regexp/Range</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>QNAME</td>
<td>String</td>
<td>![^-?A-~]1,255</td>
<td>Query template NAME</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>Int</td>
<td>[0, 2^16 - 1]</td>
<td>bitwise FLAG</td>
</tr>
<tr>
<td>3</td>
<td>RNAME</td>
<td>String</td>
<td>*</td>
<td>![^-()+-&lt;&gt;-<del>]!/^-</del>]*</td>
</tr>
<tr>
<td>4</td>
<td>POS</td>
<td>Int</td>
<td>[0, 2^29 - 1]</td>
<td>1-based leftmost mapping POSition</td>
</tr>
<tr>
<td>5</td>
<td>MAPQ</td>
<td>Int</td>
<td>[0, 2^8 - 1]</td>
<td>MAPping Quality</td>
</tr>
<tr>
<td>6</td>
<td>CIGAR</td>
<td>String</td>
<td>*</td>
<td>![0-9]+[MIDNSHPX=]+</td>
</tr>
<tr>
<td>7</td>
<td>RNEXT</td>
<td>String</td>
<td>*</td>
<td>![^-()+-&lt;&gt;-<del>]!/^-</del>]*</td>
</tr>
<tr>
<td>8</td>
<td>PNEXT</td>
<td>Int</td>
<td>[0, 2^29 - 1]</td>
<td>Position of the mate/next segment</td>
</tr>
<tr>
<td>9</td>
<td>TLEN</td>
<td>Int</td>
<td>![2^29 +1, 2^29 - 1]</td>
<td>observed Template LENgth</td>
</tr>
<tr>
<td>10</td>
<td>SEQ</td>
<td>String</td>
<td>![A-Za-z=.]+</td>
<td>segment SEQuence</td>
</tr>
<tr>
<td>11</td>
<td>QUAL</td>
<td>String</td>
<td>![^-~]+</td>
<td>ASCII of Phred-scaled base QUALity+33</td>
</tr>
</tbody>
</table>
A tour of the columns in a SAM file

```bash
bialbert@porthos ~/work/lec11
$ cat results.sam | grep -v "SQ" | cut -f 1,2,3,4
0changes 0    chrI    1
1changes 0    chrI    1
2changes 0    chrI    1
4changes 4    *       0

bialbert@porthos ~/work/lec11
$ cat results.sam | grep -v "SQ" | cut -f 5,6,7,8
37   80M    *       0
37   80M    *       0
37   13M1D66M *       0
0    *       *       0
```

bialbert@porthos ~/work/lec11
$
Column 1 and 2: QNAME and FLAG (Query name and bitwise flags)

QNAME: the name of the query sequence

2. FLAG: bitwise FLAG. Each bit is explained in the following table:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>template having multiple segments in sequencing</td>
</tr>
<tr>
<td>0x2</td>
<td>each segment properly aligned according to the aligner</td>
</tr>
<tr>
<td>0x4</td>
<td>segment unmapped</td>
</tr>
<tr>
<td>0x8</td>
<td>next segment in the template unmapped</td>
</tr>
<tr>
<td>0x10</td>
<td>SEQ being reverse complemented</td>
</tr>
<tr>
<td>0x20</td>
<td>SEQ of the next segment in the template being reversed</td>
</tr>
<tr>
<td>0x40</td>
<td>the first segment in the template</td>
</tr>
<tr>
<td>0x80</td>
<td>the last segment in the template</td>
</tr>
<tr>
<td>0x100</td>
<td>secondary alignment</td>
</tr>
<tr>
<td>0x200</td>
<td>not passing quality controls</td>
</tr>
<tr>
<td>0x400</td>
<td>PCR or optical duplicate</td>
</tr>
</tbody>
</table>

- Bit 0x4 is the only reliable place to tell whether the segment is unmapped. If 0x4 is set, no assumptions can be made about RNAME, POS, CIGAR, MAPQ, bits 0x2, 0x10 and 0x100 and the bit 0x20 of the next segment in the template.
Column 2: FLAG
the bitwise representation

1  = 00000001  \rightarrow  paired end read
2  = 00000010  \rightarrow  mapped as proper pair
4  = 00000100  \rightarrow  unmappable read
8  = 00001000  \rightarrow  read mate unmapped
16 = 00010000  \rightarrow  read mapped on reverse strand

The flag 11 \rightarrow 1 + 2 + 8 = 0001011 (conditions 1, 2 and 8 satisfied)

It is used to save space – but it does make things a bit more difficult.

Usually very few flags are needed in practice – 0, 4, 16 are the most generic ones

If you need to construct a more complex flag search for explain SAM flags:

http://picard.sourceforge.net/explain-flags.html
Columns 3, 4: RNAME and POS Reference and Position

Column 4 POS: **1-based leftmost mapping POSition of the first matching base.**

Very important to remember later when we need to find the 5’ end (the actual start)
Column 5: MAPQ - Mapping Quality

- Phred score, identical to the quality measure in the fastq file. Quality $Q$, probability $P$:

$$ P = 10^{(-Q/10.0)} $$

If $Q=30$, $P=1/1000 \rightarrow$ on average, one of out 1000 alignments will be wrong

As good as this sounds it is not easy to compute such a quality.
Details of the mapping quality computation – hard to find good answers

- Tool specific – there is no standard of what it should be

- The repeat structure of the reference. Reads falling in repetitive regions usually get very low mapping quality.

- The base quality of the read. Low quality means the observed read sequence is possibly wrong, and wrong sequence may lead to a wrong alignment.

- The sensitivity of the alignment algorithm. The true hit is more likely to be missed by an algorithm with low sensitivity, which also causes mapping errors.

- Paired end or not. Reads mapped in pairs are more likely to be correct.

(from the MAQ manual)
A read alignment with a mapping quality 30 or above usually implies:

- The overall base quality of the read is good.
- The best alignment has few mismatches.
- The read has few or just one `good' hit on the reference, which means the current alignment is still the best even if one or two bases are actually mutations or sequencing errors.
BWA specific low scores

Surprisingly difficult to track down the exact behavior

• Q=0 → if a read can be aligned equally well to multiple positions, BWA will randomly pick one position and give it a mapping quality zero.

• Q=25 → the edit distance equals mismatches and is greater than zero
Column 6: CIGAR

- **CIGAR** = Compact Idiosyncratic Gapped Alignment Report

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

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

- H can only be present as the first and/or last operation.
- S may only have H operations between them and the ends of the CIGAR string.
- For mRNA-to-genome alignment, an N operation represents an intron. For other types of alignments, the interpretation of N is not defined.
Columns 7, 8, 9: RNEXT, PNEXT, TLEN (used in paired end read sequencing)

- **RNEXT**: the name of the pair
- **PNEXT**: the position of the pair
- **TLEN**: the distance between the leftmost positions of the pairs

We will discuss these in more detail later.

These can show the position of reads that are distant – allow us to infer genomic variations
Column 10, 11

- SEQ: the query sequence
- QUAL: the phred encoded quality sequence

SEQ may contain the original sequence or the segment it was aligned to. Not all tools do the same thing.
Specific information about the alignment process that the tools was able to establish.
Generate a SAM file using dataset **lect-11.gz** as query and the yeas genome as reference then answer the following questions:

1. How many reads are in the data?
2. How many reads are unmapped?
3. How may different types of quality scores can you observe? (hint: cut, sort, uniq as before)?
4. What is the number of reads on both strands?
5. How many reads have a MAPQ (mapping quality) of 0 and what does that value mean in a SAM file from BWA?