SAM/BAM/CRAM Format

(modified from: https://learn.gencore.bio.nyu.edu)

File extensions : file.sam

The official SAM documentation can be found here.

These formats were introduced to standardize how alignments are reported. Initially there were many different formats, most of them proprietary, which were space inefficient and either held too much or too little information. The first of these to be introduced was Sequence Alignment Map (SAM). With this format not only is the alignment retained but the associated quality scores (both mapping and base quality), the original read itself, paired-end information, sample information, and many more features.

```
1:497:R:-272+13M17D24M  113     1       497     37      37M     15      100338662       0       CGGGTCTGACCTGAGGAGA
19:20389:F:275+18M2D19M 99      1       17644   0       37M     =       17919   314     TATGACTGCTAATAATACCTACACATG
19:20389:F:275+18M2D19M 147     1       17919   0       18M2D19M        =       17644   -314    GTAGTACCAACTGTAAGTC
9:21597+10M2I25M:R:-209 83      1       21678   0       8M2I27M =       21469   -244    CACCACATCACATATACCAAGCCTGGC
```

Example:

SAM Format

The SAM Format is a text format for storing sequence data in a series of tab delimited ASCII columns. Most often it is generated as a human readable version of its sister BAM format, which stores the same data in a compressed, indexed, binary form. SAM format files are generated following mapping of the reads to reference sequence. It is TAB-delimited text format with header and a body. Header lines start with '@' while alignment lines do not. Header hold generic information on SAM file along with version information, if the file is sorted, information on reference sequence, etc. The alignment records constitute the body of the file. Each alignment line/ record has 11 mandatory fields describing essential alignment information.

SAM Header

The header varies in size but adheres to a particular format depending on what information you decide to add. Some example information that can be entered into the header is: command that

Tag	Description
@HD	The header line. The first line if present.
VN*	Format version. *Accepted format*: /^[0-9]+\.[0-9]+$/.
SO	Sorting order of alignments. *Valid values*: **unknown** (default), **unsorted**, **queryname** and **coordinate**. For coordinate sort, the major sort key is the RNAME field, with order defined by the order of @SQ lines in the header. The minor sort key is the POS field. For alignments with equal RNAME and POS, order is arbitrary. All alignments with '*' in RNAME field follow alignments with some other value but otherwise are in arbitrary order.
GO	Grouping of alignments, indicating that similar alignment records are grouped together but the file is not necessarily sorted overall. *Valid values*: **none** (default), **query** (alignments are grouped by QNAME), and **reference** (alignments are grouped by RNAME/POS).
@SQ	Reference sequence dictionary. The order of @SQ lines defines the alignment sorting order.
SN*	Reference sequence name. The SN tags and all individual AN names in all @SQ lines must be distinct. The value of this field is used in the alignment records in RNAME and RNEXT fields. Regular expression: [!-)+-<>-~][!-~]*
LN*	Reference sequence length. *Range*: [1,2^{31}-1]
AH	Indicates that this sequence is an alternate locus.[4] The value is the locus in the primary assembly for which this sequence is an alternative, in the format '*chr*:*start*-*end*', '*chr*' (if known), or '*' (if unknown), where '*chr*' is a sequence in the primary assembly. Must not be present on sequences in the primary assembly.
AN	Alternative reference sequence names. A comma-separated list of alternative names that tools may use when referring to this reference sequence.[5] These alternative names are not used elsewhere within the SAM file; in particular, they must not appear in alignment records' RNAME or RNEXT fields. *Regular expression*: *name*(,*name*)* where *name* is [0-9A-Za-z][0-9A-Za-z*+.@_|-]*
AS	Genome assembly identifier.
M5	MD5 checksum of the sequence. See Section 1.3.1
SP	Species.
UR	URI of the sequence. This value may start with one of the standard protocols, e.g http: or ftp:. If it does not start with one of these protocols, it is assumed to be a file-system path.
@RG	Read group. Unordered multiple @RG lines are allowed.
ID*	Read group identifier. Each @RG line must have a unique ID. The value of ID is used in the RG tags of alignment records. Must be unique among all read groups in header section. Read group IDs may be modified when merging SAM files in order to handle collisions.
CN	Name of sequencing center producing the read.
DS	Description.
DT	Date the run was produced (ISO8601 date or date/time).
FO	Flow order. The array of nucleotide bases that correspond to the nucleotides used for each flow of each read. Multi-base flows are encoded in IUPAC format, and non-nucleotide flows by various other characters. *Format*: /*|[ACMGRSVTWYHKDBN]+/
KS	The array of nucleotide bases that correspond to the key sequence of each read.
LB	Library.
PG	Programs used for processing the read group.
PI	Predicted median insert size.
PL	Platform/technology used to produce the reads. *Valid values*: **CAPILLARY**, **LS454**, **ILLUMINA**, **SOLID**, **HELICOS**, **IONTORRENT**, **ONT**, and **PACBIO**.
PM	Platform model. Free-form text providing further details of the platform/technology used.
PU	Platform unit (e.g. flowcell-barcode.lane for Illumina or slide for SOLiD). Unique identifier.
SM	Sample. Use pool name where a pool is being sequenced.
@PG	Program.
ID*	Program record identifier. Each @PG line must have a unique ID. The value of ID is used in the alignment PG tag and PP tags of other @PG lines. PG IDs may be modified when merging SAM files in order to handle collisions.
PN	Program name
CL	Command line
PP	Previous @PG-ID. Must match another @PG header's ID tag. @PG records may be chained using PP tag, with the last record in the chain having no PP tag. This chain defines the order of programs that have been applied to the alignment. PP values may be modified when merging SAM files in order to handle collisions of PG IDs. The first PG record in a chain (i.e. the one referred to by the PG tag in a SAM record) describes the most recent program that operated on the SAM record. The next PG record in the chain describes the next most recent program that operated on the SAM record. The PG ID on a SAM record is not required to refer to the newest PG record in a chain. It may refer to any PG record in a chain, implying that the SAM record has been operated on by the program in that PG record, and the program(s) referred to via the PP tag.
DS	Description.
VN	Program version
@CO	One-line text comment. Unordered multiple @CO lines are allowed.

generated the SAM file, SAM format version, sequencer name and version. The full list of available header fields can be found below (from: http://samtools.github.io/hts-specs/SAMv1.pdf)

Some terminology used in SAM manual:

Template: The DNA fragment that was measured
Reads: Depending on the methodology a template may produce one or more reads. These reads may cover the entire template or just a subsection of it. Reads originating from the same template typically cover different parts of the template, and, may represent the template itself or the reverse complement of it.
Segments: Each read may produce one or more alignments that in turn will have aligned regions called segments. From these segments it may be possible the infer the size of the original template.

Field Descriptions

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{31}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next read
8	PNEXT	Int	$[0,2^{31}-1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31}+1,2^{31}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

http://samtools.github.io/hts-specs/SAMv1.pdf

Each row contains 11 mandatory fields. The descriptions for them can be found below:

Fields:

Col. 1 QNAME:

Query NAME. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME '*' indicates the information is unavailable. In a SAM file, a read may occupy multiple alignment lines, when its alignment is chimeric .

Col. 2 FLAG:

Combination of bitwise flags.

Bitwise Flag

The bitwise flag is a lookup code to explain certain features about the particular read (exact same concept as Linux permission codes!). It tells you whether the read aligned, is marked a PCR duplicate, if it's mate aligned, etc. and any combination of the available tags, seen below:

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

http://samtools.github.io/hts-specs/SAMv1.pdf

One important thing to note is that any combination of these flags results in one integer, which makes interpreting it a bit difficult. To make it easy you can check here to either encode or decode a bitwise flag.

Col. 3 RNAME:

Name of reference sequence. It generally refers to chromosome number.

Col. 4 POS:

Leftmost mapping position of the first matching base in read. It has 1-based indexing. If pos is set as 0, it represents a unmapped read. For a read pair READ1/1 and READ1/2 and single Read2.

Col. 5 MAPQ:

It indicates MAPpping Quality, or generally how well the read aligned to the reference. MAPQ= -10log10 (Probability of mapping position being wrong). MAPQ=255 indicates mapping quality is unavailable. Different algorithms report it differently but nonetheless, the greater the number the better the alignment (generally).

Col. 6 CIGAR:

CIGAR String

Op	Description
M	Match (alignment column containing two letters). This could contain two different letters (mismatch) or two identical letters. USEARCH generates CIGAR strings containing Ms rather than X's and ='s (see below).
D	Deletion (gap in the target sequence).
I	Insertion (gap in the query sequence).
S	Segment of the query sequence that does not appear in the alignment. This is used with soft clipping, where the full-length query sequence is given (field 10 in the SAM record). In this case, S operations specify segments at the start and/or end of the query that do not appear in a local alignment.
H	Segment of the query sequence that does not appear in the alignment. This is used with hard clipping, where only the aligned segment of the query sequences is given (field 10 in the SAM record). In this case, H operations specify segments at the start and/or end of the query that do not appear in the SAM record.
=	Alignment column containing two identical letters. USEARCH can read CIGAR strings using this operation, but does not generate them.
X	Alignment column containing a mismatch, i.e. two different letters. USEARCH can read CIGAR strings using this operation, but does not generate them.

https://www.drive5.com/usearch/manual/cigar.html

For example:

```
RefPos:     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T  G  A  A  C  T  G  A  C  T  A  A  C
Read: ACTAGAATGGCT
```

Aligning these two:

```
RefPos:     1  2  3  4  5  6  7     8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T     G  A  A  C  T  G  A  C  T  A  A  C
Read:                  A  C  T  A  G  A  A     T  G  G  C  T
```

With the alignment above, you get:

```
POS: 5
CIGAR: 3M1I3M1D5M
```

https://genome.sph.umich.edu/wiki/SAM#What_is_a_CIGAR.3F

This is a shorthand way to encode an entire alignment. Instead of writing the whole alignment out, operators have been defined and are used in combination with numbers to explain which part of the sequence aligns, which doesn't, and everything in between. The definition for the operators can be found here:
The remaining fields are not so important or self explanatory, so no need to continue going through them. If you are interested, Google SAM file format to find out exactly what they define.

Example SAM file

It is much easier to take a quick look at a SAM files using a text editor without soft wrapping. But for completness sake, here is a part of a file:

```
HWI-ST865:416:C6CG0ACXX:1:1313:9073:43827 0 I 2 1 99M * 0 0
CCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTA
AGCCTAAGCCT
@C@DFDEFFHDFFIIJIGIIGIGGIIIIJGHGIJJEEIAHHGGIGFH@HGCFGGGJJIIGDAFG@DGIHHHHHFFBB@CACEC6;?
CDD?CDCAD>>AA AS:i:0 XS:i:0 XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:99 YT:Z:UU
HWI-ST865:416:C6CG0ACXX:1:1215:16359:6484 16 I 9 1 85M * 0 0
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCC
EEEEFFFFDAGHHHIJJIIJJJJJIJJJJIJIJJIGIIJJJJJJJIJJJJJJJIIJJIIJGJJIJJJJJJJHFHHHFFFFFCCB
AS:i:0 XS:i:0 XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:85 YT:Z:UU
HWI-ST865:416:C6CG0ACXX:1:1113:14118:89232 16 I 15 1 100M * 0 0
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAA
GCCTAAGCCTAA CAC@A>C@AADCAC@ACEEC@@BD?
E;@CEHGCEIGIHAFHGGF;FCHBHFBHIGIIIJJJJJJJJJJJJJJJJJJJJIJJJJJJJHHHHHFFFFFCCC AS:i:0
XS:i:0 XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:100 YT:Z:UUHWI-
ST865:412:C6CLLACXX:1:2315:12173:84819 16 I 49 1 70M * 0 0
GCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCT
@7)3CC=)CA;EBC>DAEDBDCDDDCDD@B?<DEDE399CBC<+>EAE<BDCEAE3DADDD<ABDD;1?? AS:i:0 XS:i:0
XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:70 YT:Z:UU
HWI-ST865:412:C6CLLACXX:1:1201:19323:33842 16 I 71 0 100M * 0 0
AAGCCTAAGCCTAAGCCTAAGCCTAAGCCAAATCCCAAGCCTAAGCCTAAGCCTAAGCCTAAGCCAGAGCCTAAGCCTAAGCCTTAGC
CTAAGCCTGATC DDDCCDCCACCCDCCCC>CAA@D@;BDHA3A7(@5/
IIHFIIGEIIIIHEIIIIIGGIGIIIIIFDCIIIIIIIIGGIIGIHFFEIHDDBHFFDDDB@@@ AS:i:-33 XS:i:-33
XN:i:0 XM:i:8 XO:i:0 XG:i:0 NM:i:8 MD:Z:29T2G2T29T0A17A11A1G1 YT:Z:UU
HWI-ST865:412:C6CLLACXX:1:1215:19021:78287 16 I 93 1 66M * 0 0
CTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGCCTAAGC @=87=3GF=.:CFE;D@B3?
3?BD9BC<>CJJJJJJJJJJJJJJJIJJJJIHHHHHFFEDAC@B AS:i:0 XS:i:0 XN:i:0 XM:i:0 XO:i:0 XG:i:0
NM:i:0 MD:Z:66 YT:Z:UU
```

Yeah, pretty difficult to make sense of.

BAM Format

This is the same format except that it encoded in binary which means that it is significantly smaller than the SAM files and significantly faster to read, though it is not human legible and needs to be converted to another format (i.e. SAM) in order to make sense to us.

Some special tools are needed in order to make sense of BAM, such as Samtools, Picard Tools, and IGV which will be discussed in some of the latter sections.

CRAM Format

This is a relatively new format that is very similar to BAM as it also retains the same information as SAM and is compressed, but it is much smarter in the way that it stores the information. It's very interesting and up and coming but is a bit beyond the scope of this course. However, if you're up for it you can read about it here.

What software use these files?

- Alignment algorithms
- Some assemblers
- CRAM/unaligned Bam (uBAM) can be a source of data delivery in some institutions: this cuts down significantly on storage space and transfer speed.
- Alignment viewers
- Variant detection algorithms