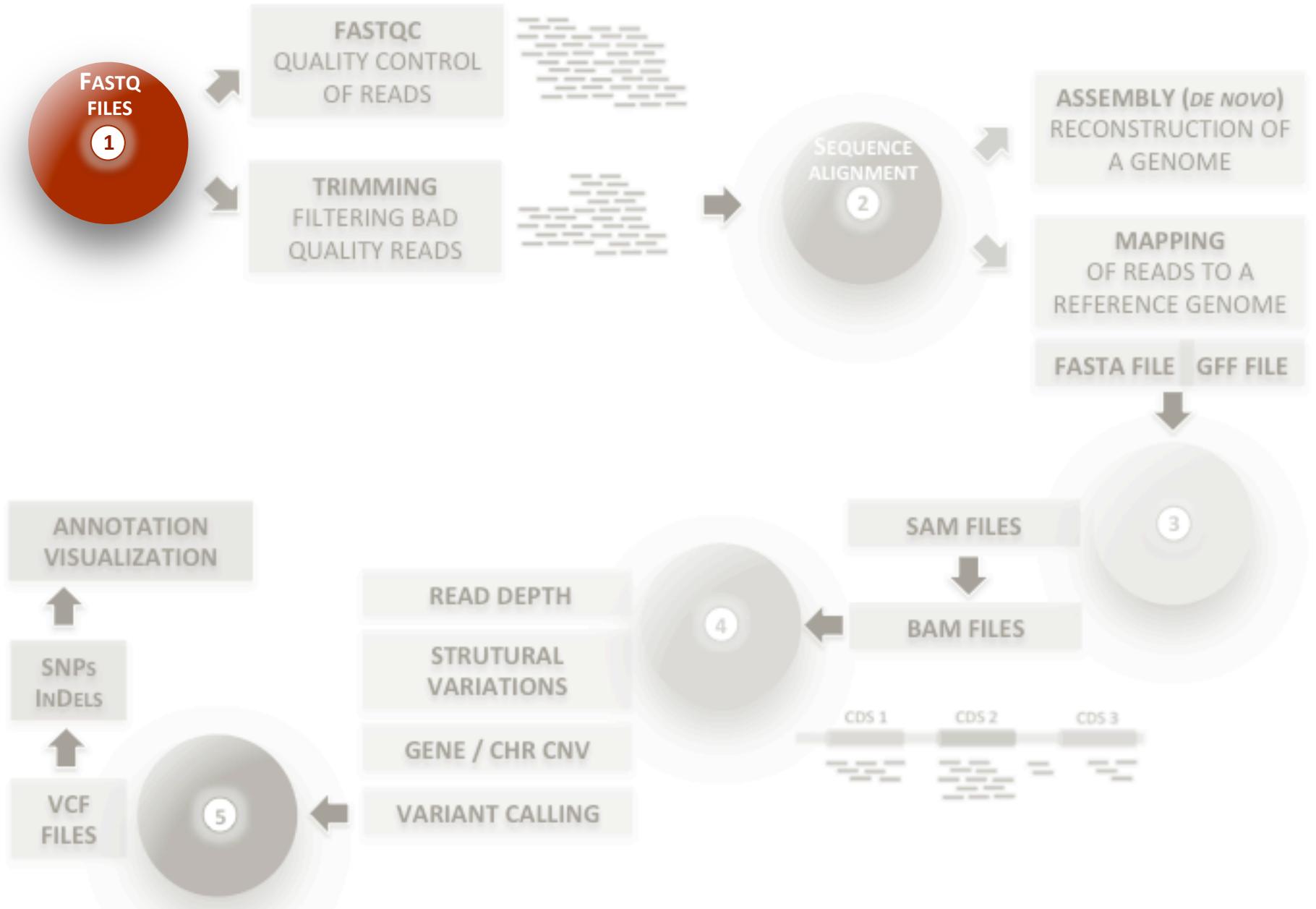


File formats

Understanding your mapping

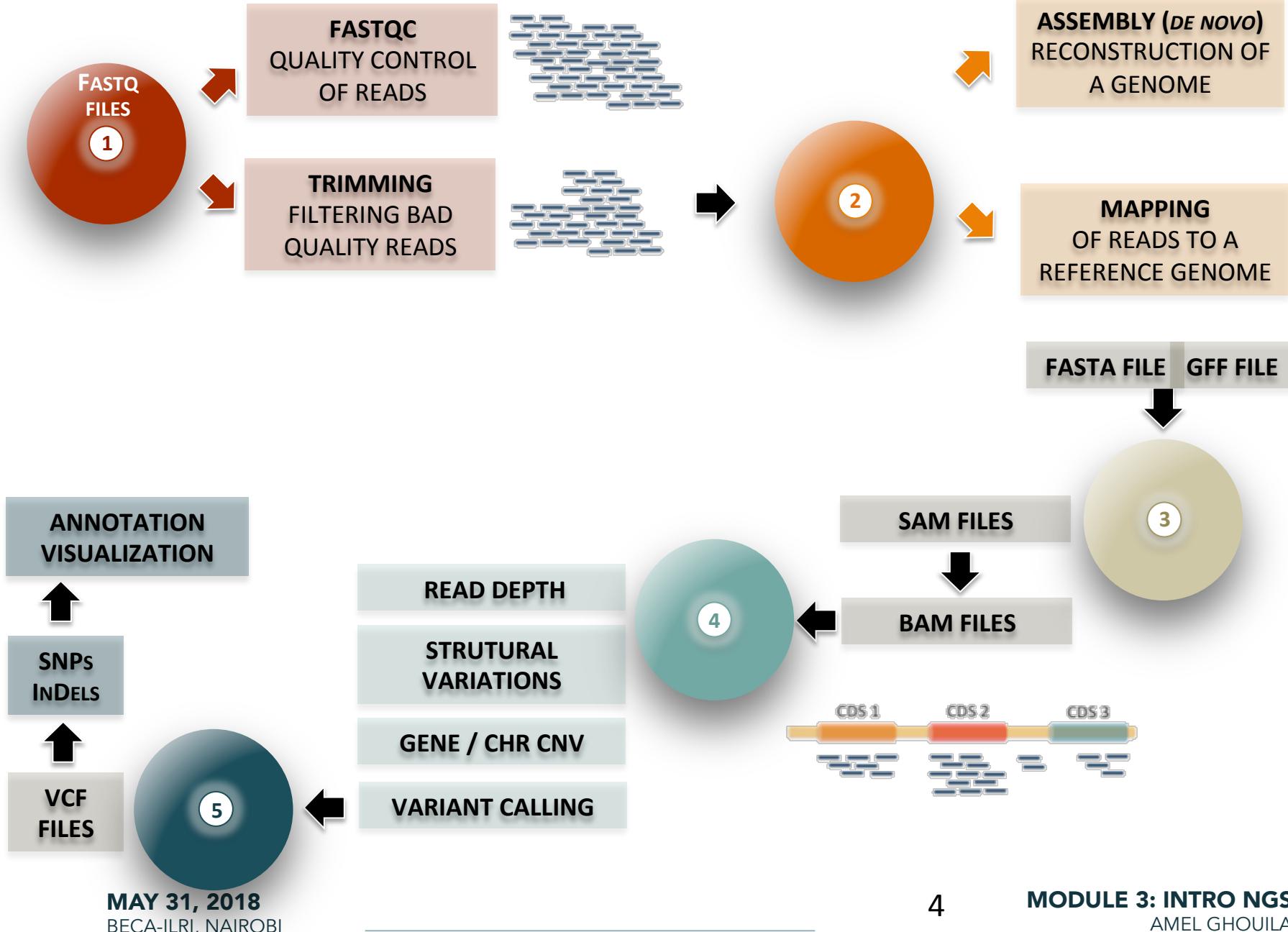




Part I: File formats



Overview of analysis workflow



FASTQ FILE FORMAT

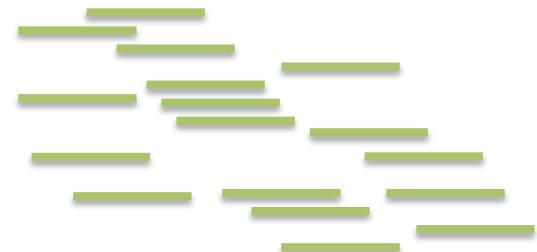
read file format

What is a FastQ file?

FASTQ= FASTA + Quality

FastQ format is a text-based format for storing both a **biological sequence** and the corresponding per base **quality scores**

-> Most common output provided by sequencing platforms





FASTQ FILE FORMAT

read file format

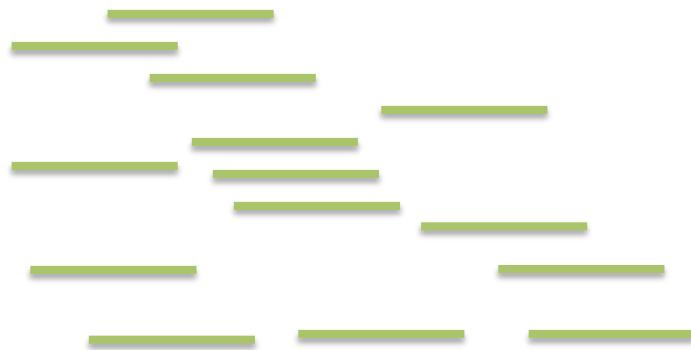
What is a FastQ file?

FASTQ= FASTA + Quality

FastQ format is a text-based format for storing both a **biological sequence** and its corresponding **quality scores**



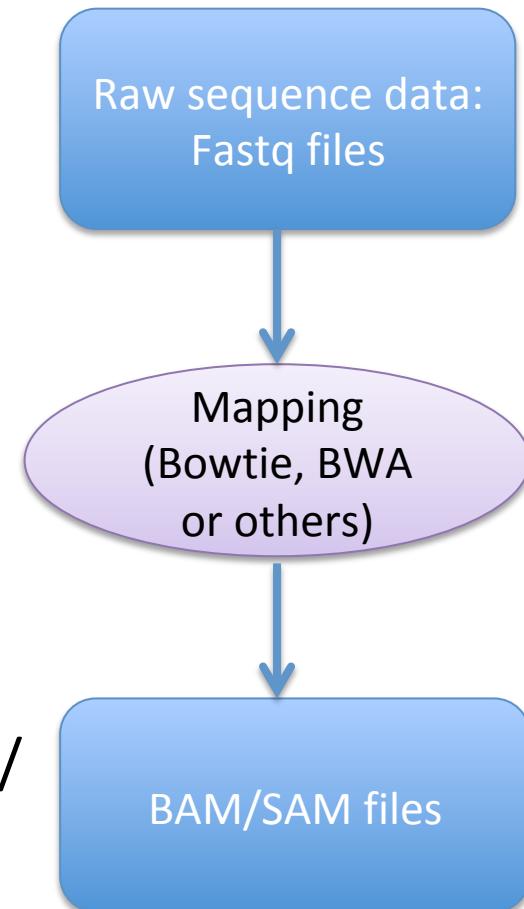
Raw sequence
data: FastQ
files



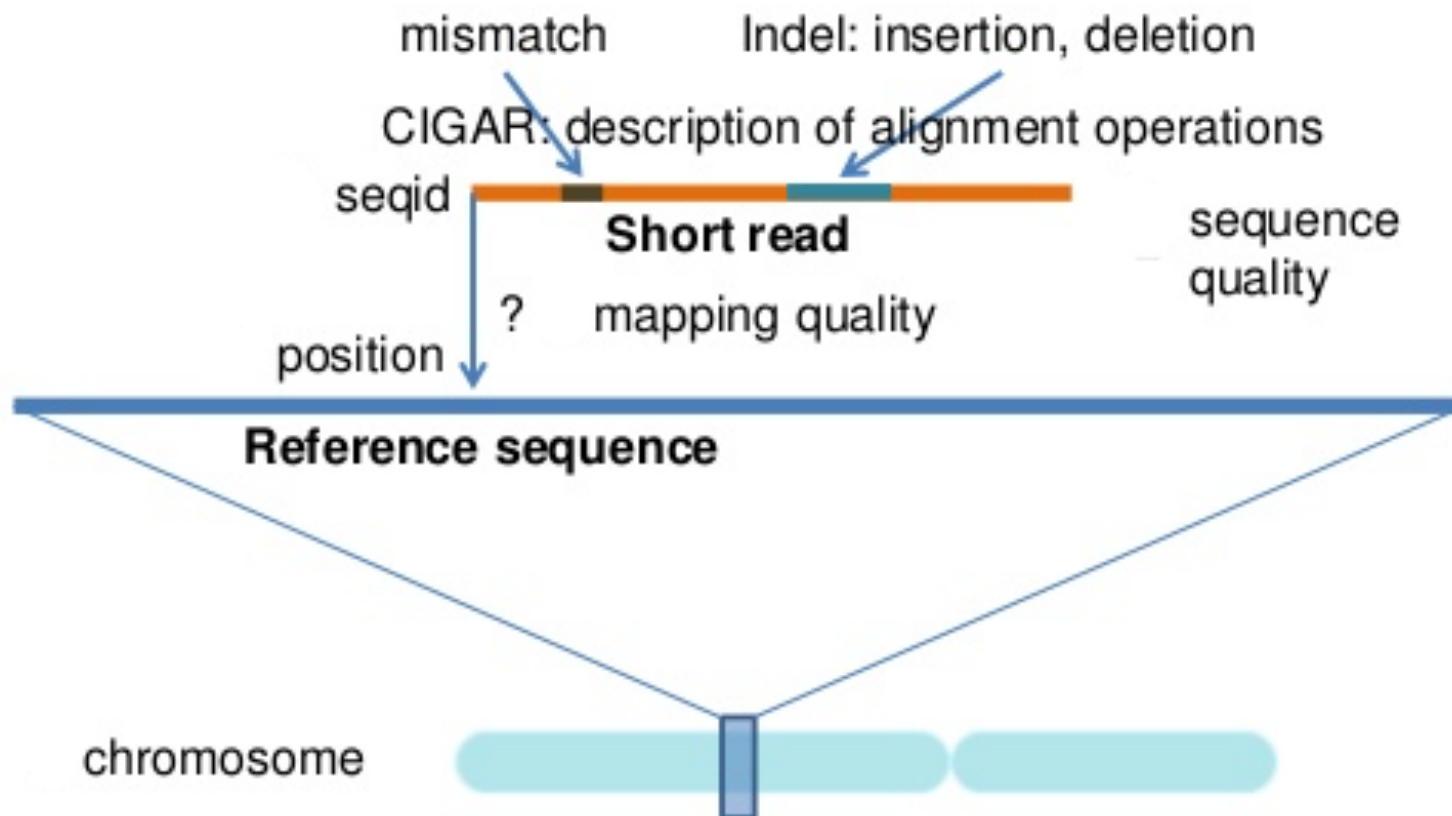
SAM and BAM formats

SAM/BAM files

- After mapping the FASTQ file to the reference genome you will end up with a SAM or BAM alignment file
- SAM stands **for Sequence Alignment/Map format**
- A single SAM file can store mapped, unmapped, and even QC-failed reads from a sequencing run, and indexed to allow rapid access. This means that the raw sequencing data can be fully recapitulated from the SAM/BAM file.



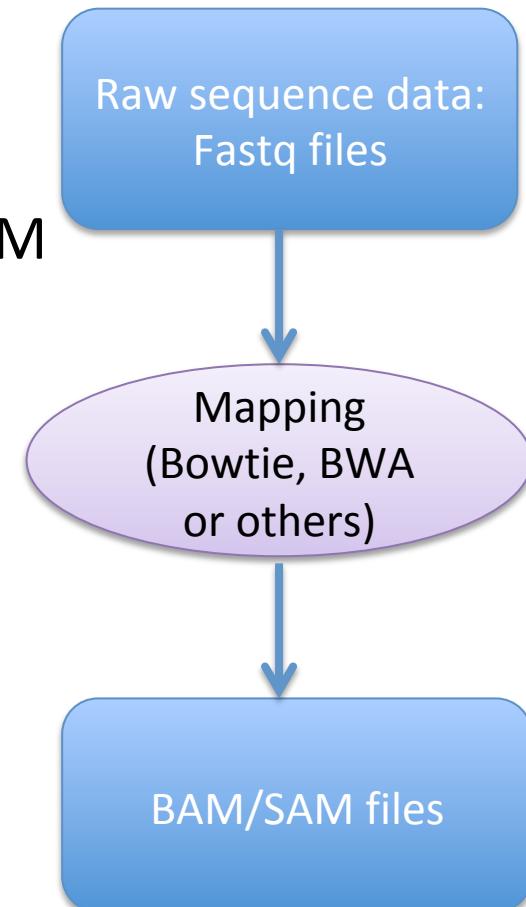
SAM/BAM files



Li Shen, 2014

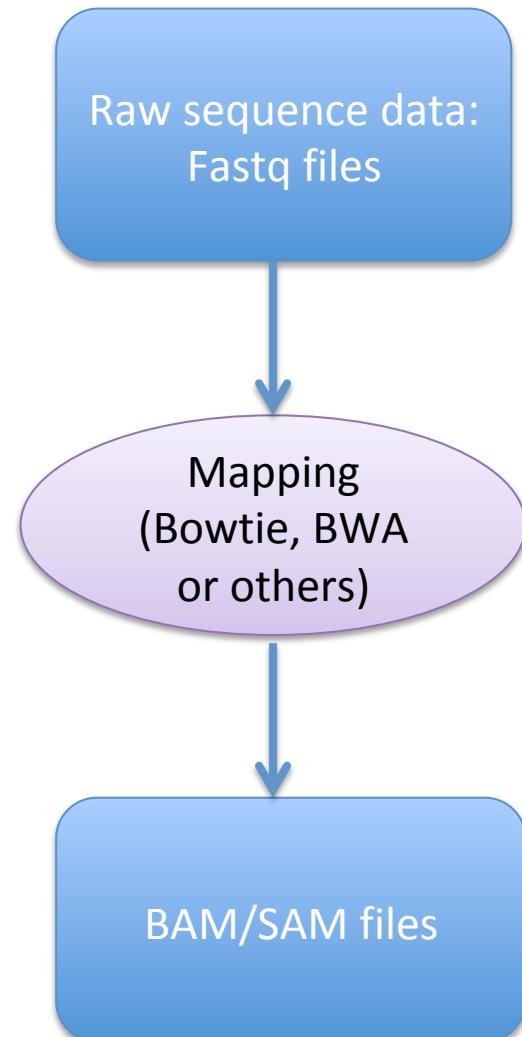
SAM/BAM files

- SAM is rarely helpful and really takes up too much space which is why we use only the BAM in principle
- A BAM file (.bam) is the **binary version** of a SAM file (saving storage and faster manipulation)



SAM/BAM files

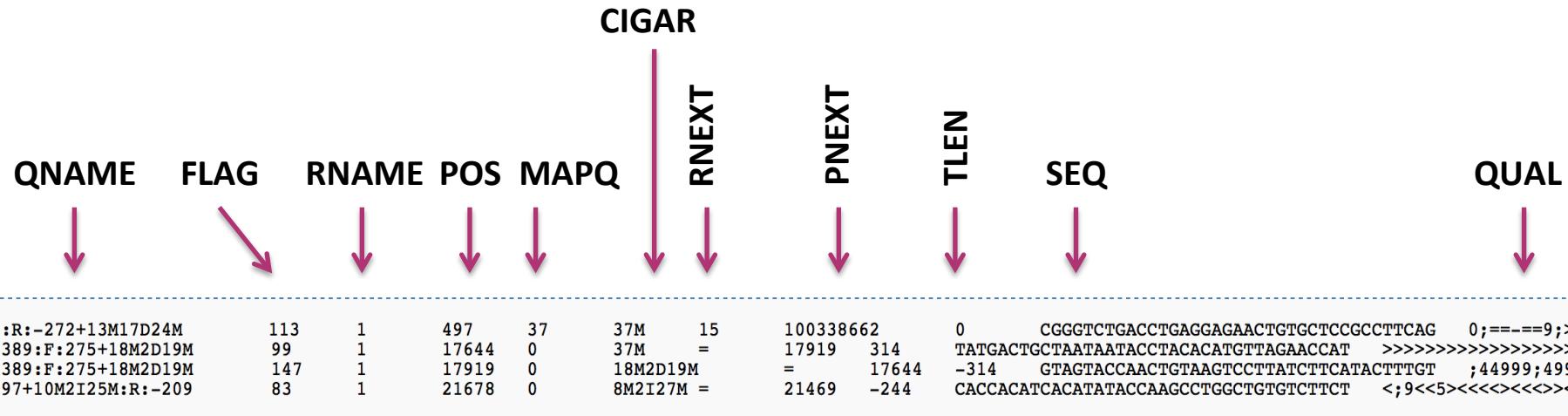
- A SAM file (.sam) is a tab-delimited text file that contains sequence alignment data
- SAM files can be opened using a text editor or viewed using the UNIX "more" command
- Most alignment programs will supply:
 - a **header**: describing the format version, sorting order of the reads, genomic sequences to which the reads were mapped
 - an **alignment section**: contains the information for each sequence about where/how it aligns to the reference genome



SAM files

Header:	Alignment section 11 columns (tab-separated)										
@SQ SN:chr9_random LN:449483	*	0	0	*	*	0	0	0	0	0	0
@SQ SN:chrM LN:16299	*	0	0	*	*	0	0	0	0	0	0
@SQ SN:chrUn_random LN:5900358	*	0	0	*	*	0	0	0	0	0	0
@SQ SN:chrX LN:166658296	*	0	0	*	*	0	0	0	0	0	0
@SQ SN:chrY LN:1785075	*	0	0	*	*	0	0	0	0	0	0
@SQ SN:chrY_random LN:58682461	*	0	0	*	*	0	0	0	0	0	0
HWI-EAS038:6:1:23:122#0 4	*	0	0	*	*	0	0	0	0	0	0
B											
HWI-EAS038:6:1:25:283#0 0	chr14	27002726	0	33M	*	0	0	0	0	0	0
BBBBBBBBBBB XT:A;R NM:i:1	X0:i:3	X1:i:0 XM:i:1	X0:i:0	XG:i:0	MD:Z:18T22						
HWI-EAS038:6:1:26:649#0 0	chr9	27884899	37	33M	*	0	0	0	0	0	0
_QWoo'YXS XT:A;U NM:i:0	X0:i:1	X1:i:0 XM:i:0	X0:i:0	XG:i:0	MD:Z:33						
HWI-EAS038:6:1:30:918#0 16	chr17	95265601	0	33M	*	0	0	0	0	0	0
_aaaa'a' XT:A;R NM:i:2	X0:i:3	X1:i:0 XM:i:2	X0:i:0	XG:i:0	MD:Z:368T28						
HWI-EAS038:6:1:32:150#0 0	chr13	57505480	37	33M	*	0	0	0	0	0	0
'bbab_([W]_bb_M_b XT:A;U NM:i:0	X0:i:1	X1:i:0 XM:i:0	X0:i:0	XG:i:0	MD:Z:33						
HWI-EAS038:6:1:32:298#0 4	*	0	0	*	*	0	0	0	0	0	0
B											
HWI-EAS038:6:1:32:193#0 0	0	chr7	65636851	37	33M	*	0	0	0	0	0
ZHMHWZEVFO][8888 XT:A;U NM:i:1	X0:i:1	X1:i:0 XM:i:1	X0:i:0	XG:i:0	MD:Z:31G1						
HWI-EAS038:6:1:32:861#0 4	*	0	0	*	*	0	0	0	0	0	0
B											
HWI-EAS038:6:1:32:1814#0 0	chr2	98506740	0	33M	*	0	0	0	0	0	0
WJPYVV\YRMR[SUZSST XT:A;R NM:i:1	X0:i:12	X1:i:44 XM:i:1	X0:i:0	XG:i:0	MD:Z:14G18						
HWI-EAS038:6:1:34:280#0 0	chr10	97252488	37	33M	*	0	0	0	0	0	0
OHDT^_BBBBBBBBBBB XT:A;U NM:i:1	X0:i:1	X1:i:0 XM:i:1	X0:i:0	XG:i:0	MD:Z:29T3						
HWI-EAS038:6:1:37:667#0 0	chrX	90652654	37	33M	*	0	0	0	0	0	0
BBBBBBBBBBB XT:A;U NM:i:1	X0:i:1	X1:i:0 XM:i:1	X0:i:0	XG:i:0	MD:Z:19C13						
HWI-EAS038:6:1:37:123#0 0	4	*	0	0	*	*	0	0	0	0	0
BBBBBBBBBBB											
HWI-EAS038:6:1:37:262#0 16	chr2	338658723	33M	*	0	0	0	0	0	0	0
a XT:A;U NM:i:1 X0:i:1	X1:i:1 XM:i:1	X0:i:0	XG:i:0	MD:Z:6C26							
HWI-EAS038:6:1:38:385#0 0	chr9	35113013	25	33M	*	0	0	0	0	0	0
BBBBBBBBBBB XT:A;U NM:i:2	X0:i:1	X1:i:0 XM:i:2	X0:i:0	XG:i:0	MD:Z:16G9C6						
HWI-EAS038:6:1:38:37#0 16	chr16	49998240	37	33M	*	0	0	0	0	0	0
'^'R')a_a XT:A;U NM:i:0	X0:i:1	X1:i:0 XM:i:0	X0:i:0	XG:i:0	MD:Z:33						
HWI-EAS038:6:1:40:991#0 16	chr13	75619559	0	33M	*	0	0	0	0	0	0
WOU[^\V_\^ XT:A;R NM:i:0	X0:i:6619	XM:i:0	X0:i:0	XG:i:0	MD:Z:33						
HWI-EAS038:6:1:40:767#0 0	chr11	34713793	25	33M	*	0	0	0	0	0	0

SAM files



<http://samtools.sourceforge.net/SAM1.pdf>
<http://genome.sph.umich.edu/wiki/SAM>

SAM files

The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	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>)

- **QNAME:** Query template NAME. Reads/segments having identical QNAME are regarded to come from the same template. A QNAME '*' indicates the information is unavailable.
- Used to group/identify alignments that are together, like paired alignments or a read that appears in multiple alignments.

SAM files

The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	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>)

FLAG: FLAG: bitwise FLAG (ideal for compression).

11 boolean flags all stored in a single column

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

SAM files

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

SAM file

(b) @SQ SN:ref LN:45
r001 163 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTA *



read mapped to position 7:

FLAG 163 (=1 + 2 + 32 + 128):

- Read is the second read in the pair (128)
- Read is properly paired (1 + 2)
- its mate is mapped to 37 on the reverse strand (32)

SAM files

Explain flag tool:

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

The screenshot shows a web-based utility titled "Decoding SAM flags". The main heading states: "This utility makes it easy to identify what are the properties of a read based on its SAM flag value, or conversely, to find what the SAM Flag value would be for a given combination of properties." Below this, instructions say: "To decode a given SAM flag value, just enter the number in the field below. The encoded properties will be listed under Summary below, to the right." A text input field is labeled "SAM Flag:" followed by an "Explain" button. To the right of the input field is a link "Switch to mate" and a note "Toggle first in pair / second in pair". On the left, there's a section titled "Find SAM flag by property:" with a list of checkboxes for various read properties. On the right, there's a "Summary:" section which is currently empty. The overall interface has a light blue header and a white body.

Decoding SAM flags

This utility makes it easy to identify what are the properties of a read based on its SAM flag value, or conversely, to find what the SAM Flag value would be for a given combination of properties.

To decode a given SAM flag value, just enter the number in the field below. The encoded properties will be listed under Summary below, to the right.

SAM Flag: Explain

Toggle first in pair / second in pair

Find SAM flag by property:

To find out what the SAM flag value would be for a given combination of properties, tick the boxes for those that you'd like to include. The flag value will be shown in the SAM Flag field above.

read paired
 read mapped in proper pair
 read unmapped
 mate unmapped
 read reverse strand
 mate reverse strand
 first in pair

Summary:

SAM files

The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	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>)

It equals $-10 \log_{10} \text{Pr}\{\text{mapping position is wrong}\}$, rounded to the nearest integer.

The **MAPQ** value can be used to figure out how unique an alignment is in the genome.

- ✓ Large number, >10 indicates it's likely the alignment is unique.
- ✓ 255 indicates that the mapping quality is not available

SAM files

- The CIGAR string is a sequence of numbers and letters representing the associated information on bases alignment used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and if there are insertions that are not in the reference

More information about these formats available here:

<http://samtools.sourceforge.net>

<https://samtools.github.io/hts-specs/SAMv1.pdf>

SAM files

Mapped and unmapped reads are imported into SAM/BAM format

The standard CIGAR description of pairwise alignment defines three operations:

'M' for **alignment match**, 'I' for insertion compared with the reference and 'D' for deletion.

(NB: The POS indicates that the read aligns starting at position 5 on the reference)

The CIGAR :

3M = 3 bases in the read sequence align with the reference.

1I = The next base in the read does not exist in the reference.

1D = The reference base does not exist in the read sequence

RefPos:	1	2	3	4	5	6	7	8	9	10	11	12	13
Reference:	C	C	A	T	A	C	T	G	A	A	C	T	G
Read:	ACTAGAAATG												
RefPos:	1	2	3	4	5	6	7	8	9	10	11	12	13
Reference:	C	C	A	T	A	C	T	G	A	A	C	T	G
Read:			A	C	T	A	G	A	A		T	G	

POS: 5

CIGAR: 3M1I3M1D2M

BECA-ILRI, NAIROBI

<http://genome.sph.umich.edu/wiki/SAM>

SAM files

Examples of CIGAR strings for different types of alignments

Alignments

(a) coor 12345678901234 5678901234567890123456789012345
ref AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGGCCAT

r001+ TTAGATAAAGGATA*CTG
r002+ aaaAGATAA*GGATA
r003+ gectaAGCTAA
r004+ ATAGCT.....TCAGC
r003- tagetTAGGC
r001- CAGCGCCAT

SAM file

(b) @SQ SN:ref LN:45
r001 163 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTA *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5H6M * 0 0 AGCTAA * NM:i:1
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 16 ref 29 30 6H5M * 0 0 TAGGC * NM:i:0
r001 83 ref 37 30 9M = 7 -39 CAGCGCCAT *

(Li et al., 2009)

SAM files

The following table gives an overview of the mandatory fields in the SAM format:

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	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]	MAppling 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]	o ⁱ s ^r ve T ^{em} plate 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>)

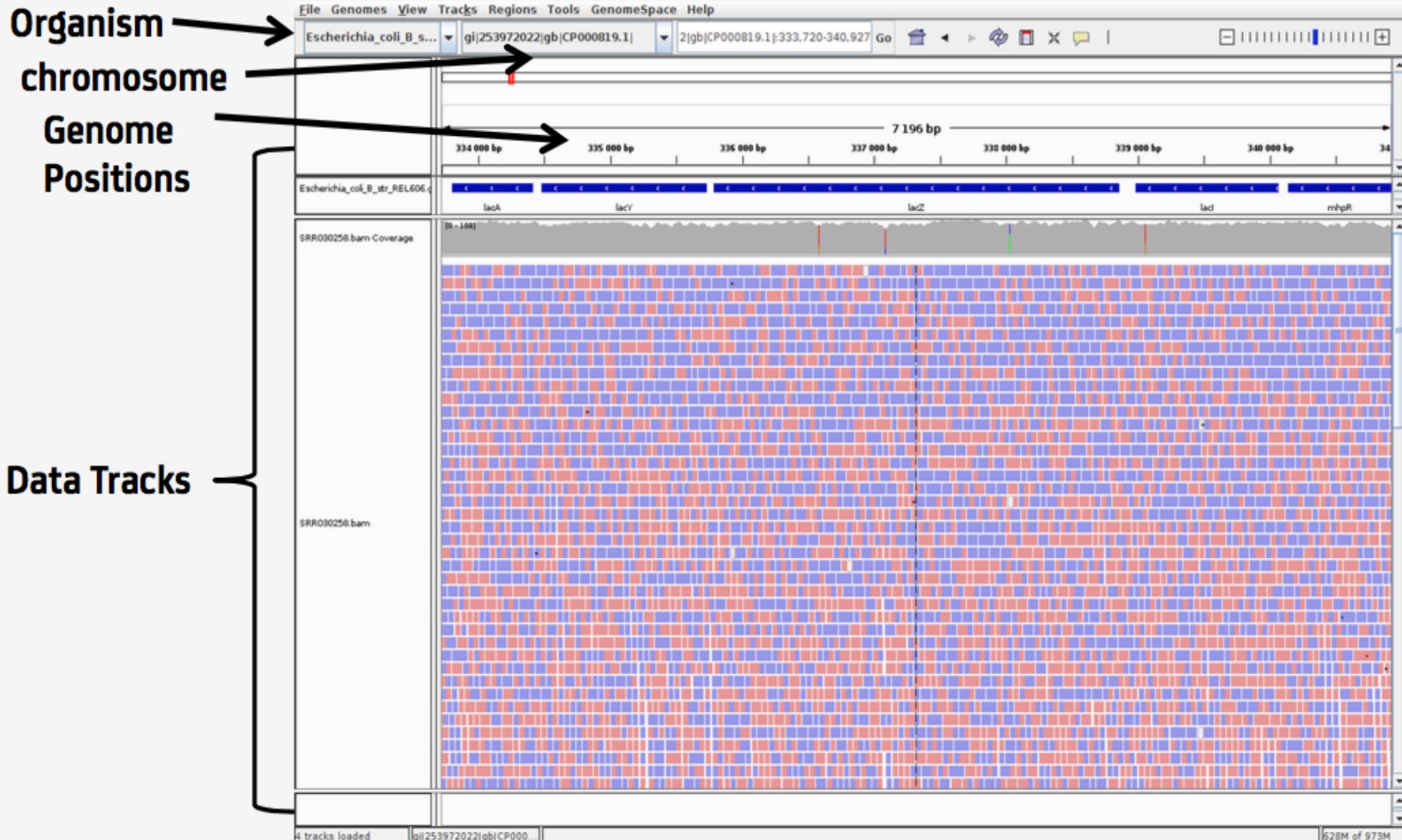
Name of mate (mate pair information for paired-end sequencing)
Position of mate (mate pair information)

Obviously, the chromosome and position are important. The CIGAR string is also important to know where insertions (i.e. introns) might exist in your read.

Part II: Visualizing mapping results



IGV: Integrated Genome Viewer



Part III: Exploring SAM/BAM files with Samtools





- A free software package for manipulating SAM/BAM files
- **SAMtools** provides the utilities for:
 - ✓ Viewing and formatting
 - ✓ Extracting statistics
 - ✓ Indexing
 - ✓ Manipulating SAM/BAM files
 - ✓ Editing

Usage: **samtools <command> [options]**

► samtools

Different samtools options available:

<http://www.htslib.org/doc/samtools.html>

Usage: samtools <command> [options]

Command:	view	SAM<->BAM conversion
	sort	sort alignment file
	mpileup	multi-way pileup
	depth	compute the depth
	faidx	index/extract FASTA
	tview	text alignment viewer
	index	index alignment
	idxstats	BAM index stats (r595 or later)
	fixmate	fix mate information
	flagstat	simple stats
	calmd	recalculate MD/NM tags and '=' bases
	merge	merge sorted alignments
	rmdup	remove PCR duplicates
	reheader	replace BAM header
	cat	concatenate BAMs
	targetcut	cut fosmid regions (for fosmid pool only)
	phase	phase heterozygotes



One of the most used tools since BAM files are often the input files needed for many different analysis programs

samtools view

#from SAM to BAM

```
 samtools view -b test.sam > test.bam
```

#from BAM to SAM

```
 samtools view test.bam > test.sam
```

Use options –h and –H to deal with the header

Always sort your BAM files; many downstream programs only take sorted BAM files.

samtools sort

#sorting a bam file

```
 samtools sort test.bam –o test.bam
```

#converting SAM directly to a sorted BAM file

```
 samtools view test.sam | samtools sort –o test.bam
```

Some programs require that for faster access we need a companion file (often called index) for the different formats

- ✓ FASTA (.fa & .fai) ([samtools faidx](#))
- ✓ BAM and BAI formats (suffixes .bam & .bai) ([samtools index](#))
- ✓ VCF (.vcf & .vcf.idx)

- `samtools flagstat file.bam`
- Does a full pass through the input file to calculate and print statistics such as:
 - ✓ % reads mapped
 - ✓ % unmapped reads
 - ✓ % reads properly paired
 - ✓ Other information

1	6874858 + 0 in total (QC-passed reads + QC-failed reads)
2	90281 + 0 duplicates
3	6683299 + 0 mapped (97.21%)
4	6816083 + 0 paired in sequencing
5	3408650 + 0 read1
6	3407433 + 0 read2
7	6348470 + 0 properly paired (93.14NaV)
8	6432965 + 0 with itself and mate mapped
9	191559 + 0 singletons (2.81NaV)
10	57057 + 0 with mate mapped to a different chr
11	45762 + 0 with mate mapped to a different chr (mapQ>=5)



Some programs require that for faster access we need a companion file (often called index) for the different formats

FASTA (.fa & .fai)

BAM and BAI formats (suffixes .bam & .bai)

VCF (.vcf & .vcf.idx)



Many tools require a BAM Index file to more efficiently access reads in a BAM file.

To create a BAM index:

- You must **first sort** the BAM file to create a sorted.bam
- Run **samtools index** with the sorted.bam as input
- This will create a file named **sorted.bam.bai** which contains the index

Example:

samtools view file.sam >file.bam

samtools sort file.bam -o file_sorted.bam

samtools index file_sorted.bam file_sorted.bai

Filtering out unmapped reads from BAM files

```
samtools view -h -F 4 file.bam > file_only_mapped.sam  
# output back to BAM  
samtools view -h -F 4 -b file.bam > file_only_mapped.bam
```

Extracting SAM entries mapping to a specific region

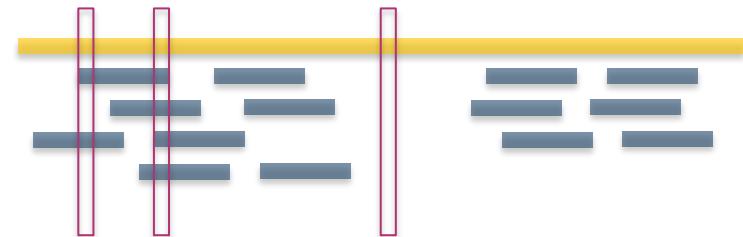
```
#index the bam file first  
samtools index file.bam  
samtools view file.bam chr1:200000-500000  
#all reads mapping on chr1 as another bam  
samtools view -b file.bam chr1 > file_chr1.bam
```

Samtools allows computing the depth at each position

Samtools depth

#syntax

samtools depth options file.bam



-a allows to output all positions (including those with zero depth)
samtools depth -a test.bam

#**-q INT** only count reads with base quality greater than *INT*
samtools depth -q int test.bam



Samtools allows computing the read depth per genomic region, as specified in the supplied BED file

Samtools bedcov

#syntax

 samtools bedcov options region.bed file.bam

Exercises:

https://github.com/bixcop18/module_3_intro_NGS/blob/master/exploremappingresults.md



GFF and BED formats

BED files

BED = (Browser Extensible Data)

(<http://genome.ucsc.edu/FAQ/FAQformat>)

BED starts are zero-based and BED ends are one-based

- Tab-delimited files
- 3 first fields required, others optional

File format (BEDPE): describes disjoint genome features, such as structural variations or paired-end sequence alignments.

CHR	START-END	FEATURE NAME	SCORE*	STRAND
chr7	127471196	Pos1	0	+
chr7	127472363	Pos2	0	+
chr7	127473530	Pos3	0	+
chr7	127474697	Pos4	0	+
chr7	127475864	Neg1	0	-
chr7	127477031	Neg2	0	-
chr7	127478198	Neg3	0	-
chr7	127479365	Pos5	0	+
chr7	127480532	Neg4	0	-

* Any string could be used (p-value, enrichment score...)

7. thickStart - The starting position at which the feature is drawn thickly.

8. thickEnd - The ending position at which the feature is drawn thickly.

9. itemRgb - An RGB value of the form R,G,B (e.g. 255,0,0).

10. blockCount - The number of blocks (exons) in the BED line.

11. blockSizes - A comma-separated list of the block sizes.

12. blockStarts - A comma-separated list of block starts.



BED files

(A) **BED3:** A BED file where each feature is described by **chrom**, **start**, and **end**.

For example: chr1 11873 14409

(B) **BED4:** A BED file where each feature is described by **chrom**, **start**, **end**, and **name**.

For example: chr1 11873 14409 uc001aaa.3

(C) **BED5:** A BED file where each feature is described by **chrom**, **start**, **end**, **name**, and **score**.

For example: chr1 11873 14409 uc001aaa.3 0

(D) **BED6:** A BED file where each feature is described by **chrom**, **start**, **end**, **name**, **score**, and **strand**.

For example: chr1 11873 14409 uc001aaa.3 0 +

(E) **BED12:** A BED file where each feature is described by all twelve columns listed above.

For example: chr1 11873 14409 uc001aaa.3 0 + 11873
11873 0 3 354,109,1189, 0,739,1347,

BED files

- ▶ **BEDTools** support a wide range of operations for interrogating and manipulating genomic features.

intersectBed

Returns overlapping features between two BED/GFF/VCF files.

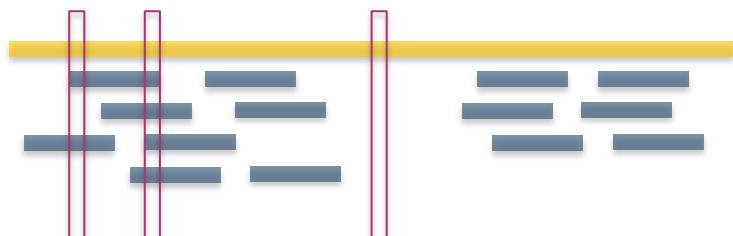
genomeCoverageBed

Histogram or a “per base” report of genome coverage.

...

▶ Coverage

The number of times each nucleotide is « read »
→ Fold Coverage (numberX)



GFF files

GFF3

The two most widely used formats for representing genome features are the BED and GFF formats.

GFF (Generic Feature Format) is a standard file format for storing genomic features in a text file.

Careful : GFF has several versions, the most recent is **GFF3**.

GFF start at a 1-based position and ends at a 1-based position.

- ▶ 1 line for 1 feature
- ▶ tab-delimited file (tab-separated columns)
- ▶ 9 columns + optional additional information

SEQ-ID	SOURCE	TYPE	START-END	SCORE	STRAND	PHASE	ATTRIBUTES
ctg123	mRNA		1300 9000	.	+	.	ID=mrna0001;Name=sonichedgehog
ctg123	exon		1300 1500	.	+	.	ID=exon00001;Parent=mrna0001
ctg123	exon		1050 1500	.	+	.	ID=exon00002;Parent=mrna0001
ctg123	exon		3000 3902	.	+	.	ID=exon00003;Parent=mrna0001
ctg123	exon		5000 5500	.	+	.	ID=exon00004;Parent=mrna0001
ctg123	exon		7000 9000	.	+	.	ID=exon00005;Parent=mrna0001

<http://gmod.org/wiki/GFF3>
<http://www.ensembl.org/>

Bedtools genomecov

- ▶ **bedtools genomecov** computes histograms per-base reports (-d) and BEDGRAPH (-bg) summaries of feature coverage

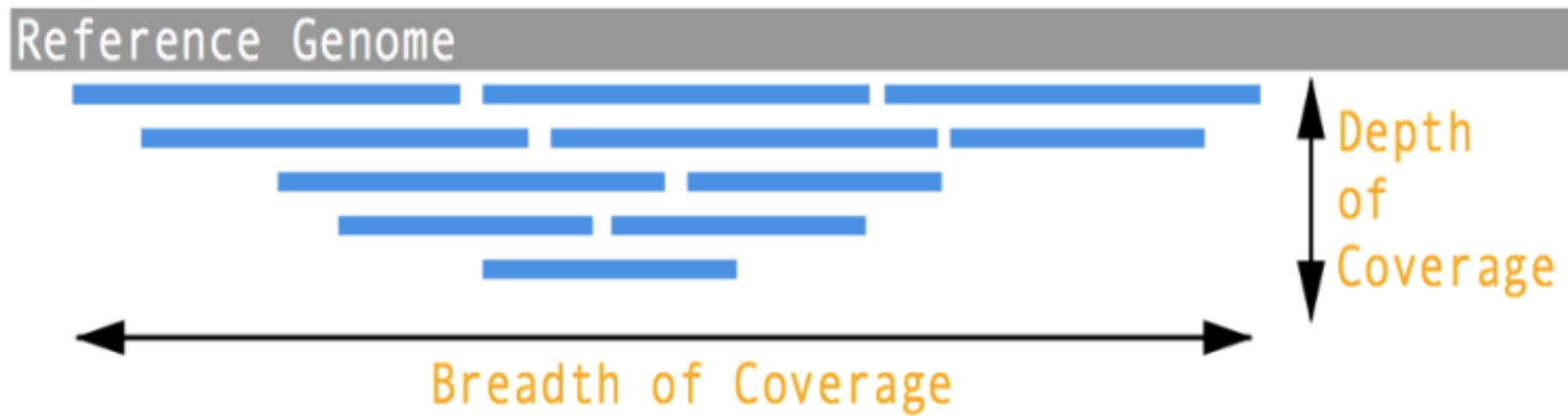
Command: bedtools genomecov -bg -ibam yourbam.bam

Check out this link:

<http://bedtools.readthedocs.io/en/latest/content/tools/genomecov.html>

for more examples and different parameters

Depth and breadth of coverage



From a bam file computes:

- Depth of coverage
- Breadth of coverage
- Sum of coverage at everyposition
- Number of bases mapped

<http://www.danielecook.com/calculate-depth-coverage-bam-file/>

Sequencing depth and coverage

Theoretical sequencing coverage $C = LN/G$

G: length of the haploid genome

L: the read length

N: the number of reads

Coverage/read count calculator:

<http://apps.bioconnector.virginia.edu/covcalc/>

Compute depth coverage:

- <http://www.danielecook.com/calculate-depth-coverage-bam-file/>
- <https://bioconductor.org/packages/devel/bioc/vignettes/CoverageView/inst/doc/CoverageView.pdf>

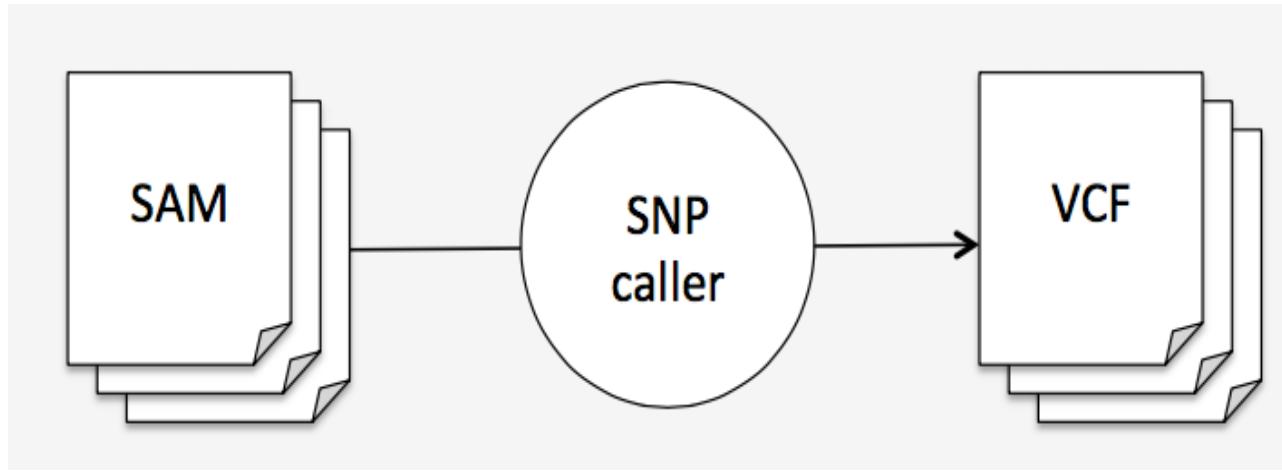
► Sequencing depth and coverage

Check these papers:

- [https://www.illumina.com/documents/products/technotes/
technote coverage calculation.pdf](https://www.illumina.com/documents/products/technotes/technote_coverage_calculation.pdf)
- <https://www.nature.com/articles/nrg3642#df1>
- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5737511/>

VCF format

VCF files



Many tools for variants detection: [GATK](#), [Samtools \(mpileup\)](#), [FreeBayes](#), [PICARD](#), etc.

VCF files

VCF file (.vcf) = “Variant Calling format”

Tab-delimited text file format that can store information about variants: the differences between your sample and the reference genome.

All this information can be easily queried (**VCFtools**).

- Format validation.
- SNV annotation.
- VCF comparison.
- Statistics.
- ...

VCF files

VCF Format coordinates are 1-based

Types of variants

SNPs

Alignment	VCF representation		
ACGT	POS	REF	ALT
ATGT	2	C	T

Insertions

Alignment	VCF representation		
AC-GT	POS	REF	ALT
ACTGT	2	C	CT

Deletions

Alignment	VCF representation		
ACGT	POS	REF	ALT
A--T	1	ACG	A

Complex events

Alignment	VCF representation		
ACGT	POS	REF	ALT
A-TT	1	ACG	AT

Large structural variants

VCF representation
POS REF ALT INFO
100 T SVTYPE=DEL;END=300

Example

#fileformat=VCFv4.0									
#fileDate=20100707									
#source=VCFtools									
#reference=NCBI36									
#INFO=<ID=AA,Number=1>Type=String,Description="Ancestral Allele">									
#INFO=<ID=H2,Number=0>Type=Flag,Description="HapMap2 membership">									
#FORMAT=<ID=GT,Number=1>Type=String,Description="Genotype">									
#FORMAT=<ID=GQ,Number=1>Type=Integer,Description="Genotype Quality (phred score)">									
#FORMAT=<ID=GL,Number=3>Type=Float,Description="Likelihoods for RR,RA,AA genotypes (R=ref,A=alt)">									
#FORMAT=<ID=DP,Number=1>Type=Integer,Description="Read Depth">									
#ALT=<ID=DEL>Description="Deletion">									
#INFO=<ID=SVTYPE,Number=1>Type=String,Description="Type of structural variant">									
#INFO=<ID=END,Number=1>Type=Integer,Description="End position of the variant">									
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1 SAMPLE2									
1	1	.	ACG	A,AT	.	PASS	.	GT:DP	1/2:13
1	2	rs1	C	T,CT	.	PASS	H2;AA=T	GT:GQ	0/1:100
1	5	.	A	G	.	PASS	.	GT:GQ	1/0:77
1	100	.	T		.	PASS	SVTYPE=DEL;END=300	GT:GQ:DP	1/1:12:3

Deletion

SNP

Large SV

Insertion

Other event

Mandatory header lines

Optional header lines (meta-data about the annotations in the VCF body)

Reference alleles (GT=0)

Alternate alleles (GT>0 is an index to the ALT column)

Phased data (G and C above are on the same chromosome)



VCF files

No need to write your own parser, you can use GATK Variants to table

[https://software.broadinstitute.org/gatk/documentation/tooldocs/current/
org_broadinstitute_gatk_tools_walkers_variantutils_VariantsToTable.php](https://software.broadinstitute.org/gatk/documentation/tooldocs/current/org_broadinstitute_gatk_tools_walkers_variantutils_VariantsToTable.php)

<http://wiki.bits.vib.be/index.php/NGS-formats>

Variants annotation

Variant annotation programs: SnpEff

- A variant annotation and effect prediction tool.
- Annotates and predicts the effects of variants on genes: Are they in a gene? In an exon? Do they change protein coding? Do they cause premature stop codons?

```
$ java -jar snpEff.jar  
SnpEff version SnpEff 4.1 (build 2015-01-07), by Pablo Cingolani  
Usage: snpEff [command] [options] [files]
```

Run 'java -jar snpEff.jar command' for help on each specific command

Available commands:

[eff ann]	: Annotate variants / calculate effects (you can use either 'ann' or 'eff')
build	: Build a SnpEff database.
buildNextProt	: Build a SnpEff for NextProt (using NextProt's XML files).
cds	: Compare CDS sequences calculated from a SnpEff database to the one in a
closest	: Annotate the closest genomic region.
count	: Count how many intervals (from a BAM, BED or VCF file) overlap with eac
databases	: Show currently available databases (from local config file).
download	: Download a SnpEff database.
dump	: Dump to STDOUT a SnpEff database (mostly used for debugging).
genes2bed	: Create a bed file from a genes list.
len	: Calculate total genomic length for each marker type.
protein	: Compare protein sequences calculated from a SnpEff database to the one
spliceAnalysis	: Perform an analysis of splice sites. Experimental feature.

Variants annotation

Here is an example of a file before and after being annotated using SnpEff:

VCF file before annotations

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
1	889455	.	G	A	100.0	PASS	AF=0.0005
1	897062	.	C	T	100.0	PASS	AF=0.0005

VCF file after being annotated using SnpEff

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
1	889455	.	G	A	100.0	PASS	AF=0.0005;EFF=STOP_GAINED(HIGH NONSENSE Cag/Tag Q236*)
1	897062	.	C	T	100.0	PASS	AF=0.0005;EFF=STOP_GAINED(HIGH NONSENSE Cag/Tag Q141*)

As you can see, SnpEff added an 'EFF' tag to the INFO field (eighth column).

|749|NOC2L||CODING|NM_015658|)
|642|KLHL17||CODING|NM_198317|)

<http://wiki.bits.vib.be/index.php/NGS-formats>