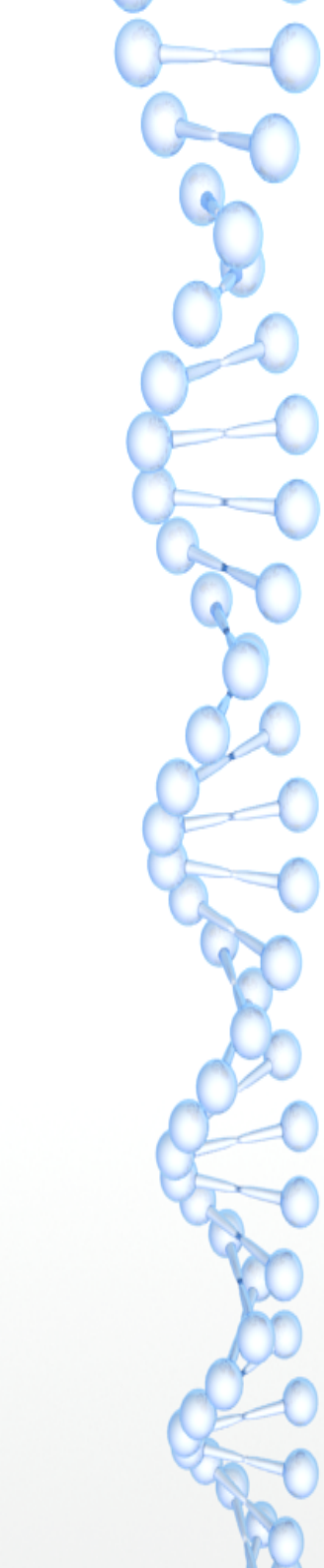


Bioinformatica II
LM Biologia Evoluzionistica
A.A. 2016-2017
Università di Padova

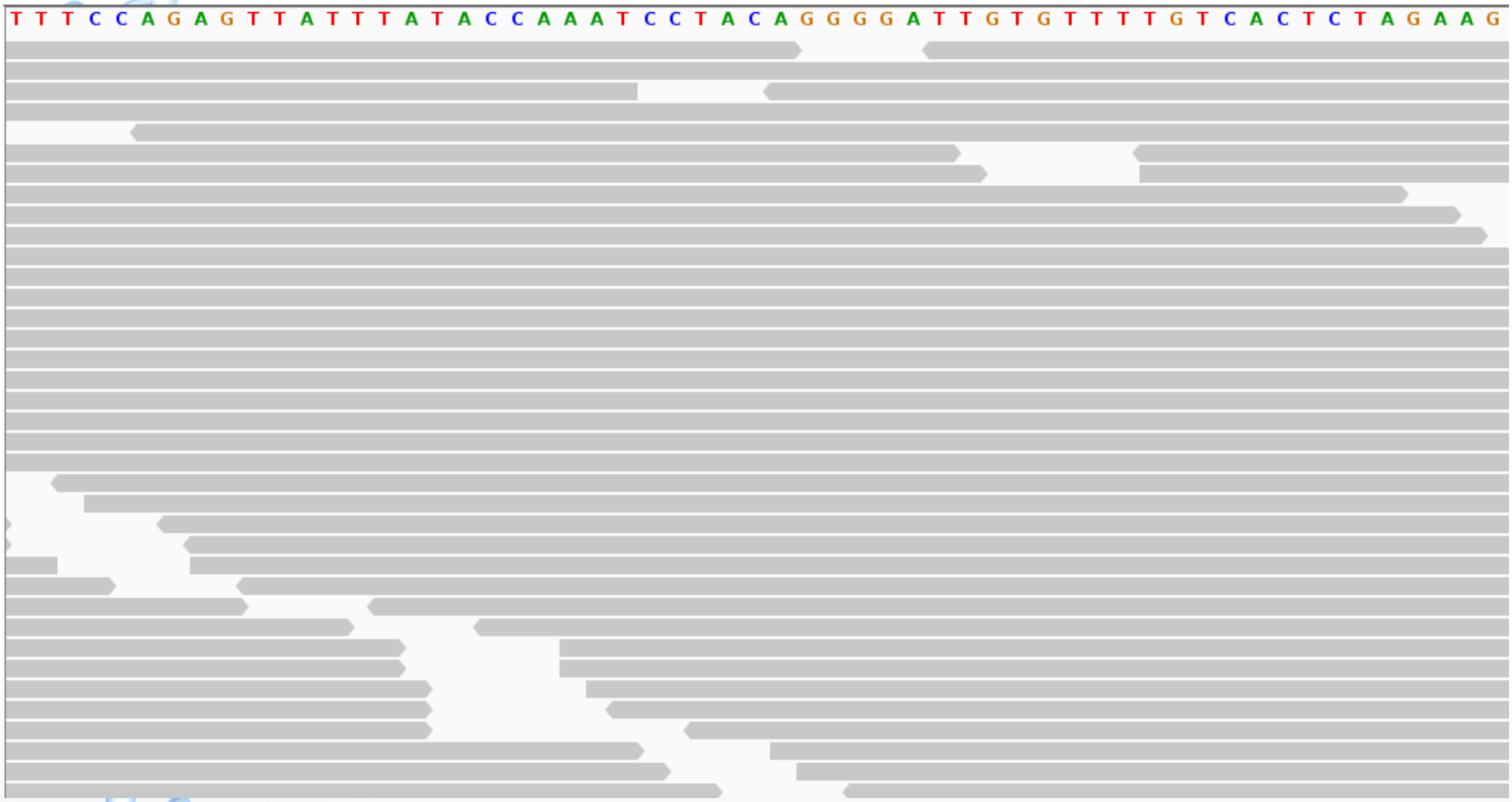
Esercitazione 6
Strumenti per analisi di dati NGS

Docente: S. Bortoluzzi
Assistenti: A. Coppe, E. Gaffo, A. Binatti

Formati dei file

- Risultati del sequenziamento: milioni di brevi sequenze (reads) di 35÷300 nucleotidi
  - **FASTQ** (visto nella lezione teorica): 4 righe per read
  - Se "**paired-end**" abbiamo due file, o un singolo file con le reads interlacciate
  - Generalmente nominati con estensione del file .fastq o .fq
- Sequenza delle basi del genoma di riferimento
  - **FASTA**: un file per ogni (sequenza di) cromosoma, o
  - **Multi-FASTA**: un singolo file con tutti i cromosomi
  - Generalmente nominati con estensione del file .fasta o .fa
- Allineamenti delle sequenze al genoma di riferimento: coordinate delle posizioni sul genoma
  - **SAM**
- Spesso i file di testo (quali sono FASTA, FASTQ, GFF) sono compressi per risparmiare (molto) spazio, quindi li troviamo generalemnte con estensione **.gz**, **.bz2** a seconda dell'algoritmo di compressione usato.
  Ad es: reads_1.fq.gz, hg38.fa.bz2, etc.
  Il formato SAM ha però anche un suo formato compresso specifico:
  - **BAM**
- Annotazioni dei geni e trascritti: coordinate delle posizioni sul genoma
  - **GFF/GTF**

Allineare le read al genoma di riferimento



Allineare read di sequenziamento genomico

- BWA = Burrows-Wheeler Aligner [1]
  - Metodo che sfrutta una indicizzazione del genoma per trovare i punti in cui le read si allineano
- Comandi per ottenere gli allineamenti:
  - Creare l'indice della sequenza del genoma
    - **`bwa index –p ref ref.fasta`**
  - Effettuare gli allineameti
    - **`bwa mem ref 1.fastq 2.fastq > mapping.sam`**

1. Li, H., and Durbin, R. (2009). *Fast and accurate short read alignment with Burrows–Wheeler transform*. Bioinformatics 25, 1754–1760.

Sequence Alignment/Map (SAM) Format

- https://samtools.github.io/hts-specs/SAMv1.pdf
- TAB-delimited text
- **header** section (optional): lines start with '@'
- **alignment** section with 11 mandatory fields

```
Coor     12345678901234  5678901234567890123456789012345
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                     ATAGCT..............TCAGC
-r003                           ttagctTAGGC
-r001/2                                        CAGCGGCAT
```

The corresponding SAM format is:[1]

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

SAM fields e SAM flags

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

FLAG: Combination of bitwise FLAGs.[4] Each bit is explained in the following table:

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

Maneggiare gli allineamenti: **samtools**

- Estrarre dal file degli allineamenti le reads che (di fatto) non sono state allineate: SAMflag ‘4’ = read non allineata
  - **samtools view –f 4 mapping.sam**
  - Quante reads abbiamo allineato?
    1. Contare le righe dei FASTQ (e dividere per 4)
    2. Contare i QNAME (univoci) del SAM, escludendo le read non mappate
- Risparmiare spazio usando il formato BAM
  - **samtools view –bh mapping.sam > mappings.bam**

Visualizzare gli allineamenti

1. Ordinare gli allineamenti per posizione
   - **`samtools sort mapping.bam > mapping.bam`**
2. Creare un indice del file degli allineamenti
   - **`samtools index mapping.bam`**
3. Caricare il file degli allineamenti su IGV

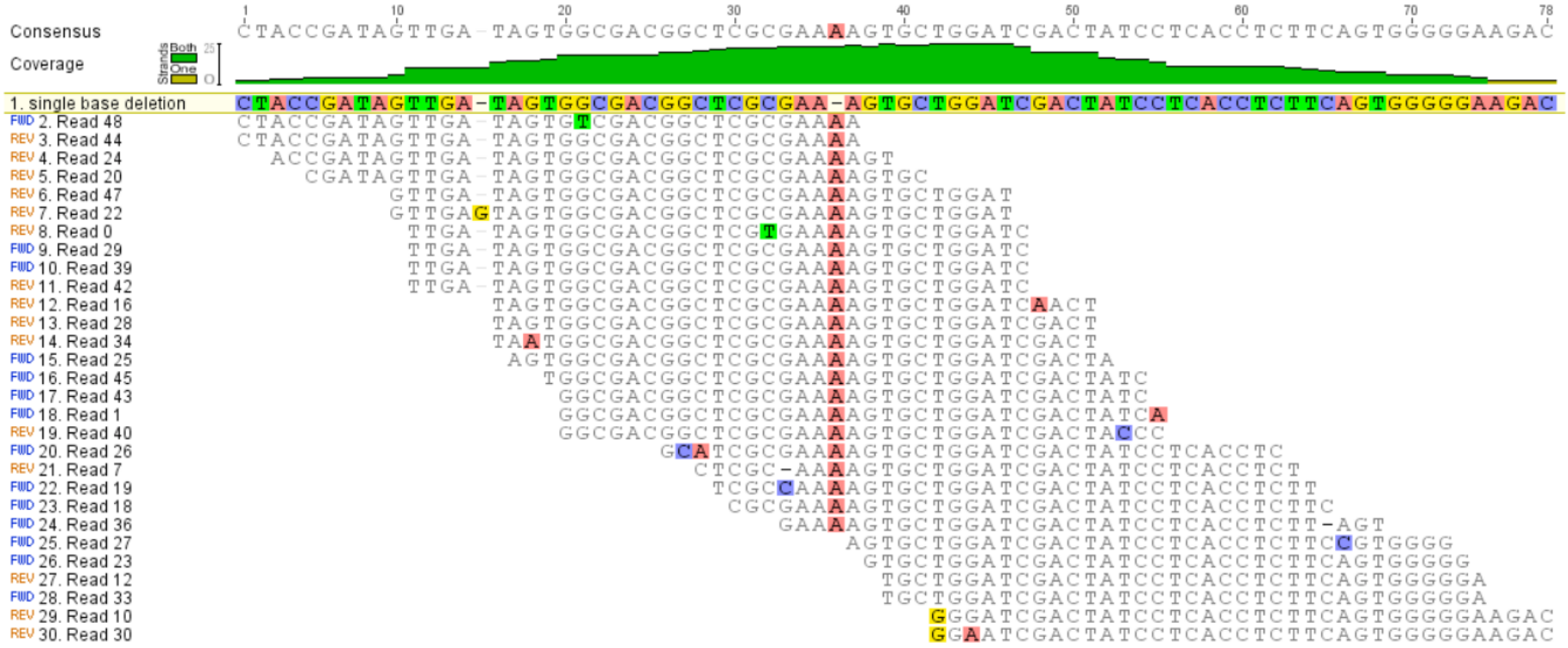
Statistiche: coverage del genoma

Metodo grossolano per calcolare il coverage medio degli allineamenti

1. Ottenere il coverage per ogni posizione:
   - **`samtools depth mapping.bam > cov.csv`**
2. Aprire cov.csv con un foglio di calcolo e sommare l'ultima colonna
3. Prendere dall'header del file SAM la lunghezza del genoma e usarla come dividendo della somma calcolata al punto precedente
   - **`samtools view –h mapping.bam | grep '@SQ'`**

Chiamata delle varianti

- Esaminare la basi allineate in ogni posizione e cercare le differenze con il genoma di riferimento
- Fattori da considerare:
  - Qualità della basi (riportati nel SAM dai FASTQ)
  - Prossimità ad altre varianti (indel) e regioni omopolimeriche
  - Qualità degli allineamenti delle read che supportano la variante (read lunghe e paired-end migliorano la qualità - MQ scores in SAM)
  - Profondità (depth) del sequenziamento



Chiamata delle varianti

- Preparazione dei dati con `samtools`
  - **`samtools mpileup -v -u -f ref.fasta mapping.bam > mapping.vcf`**
- Altri metodi sono necessari per ottenere le varianti, ad es. `bcftools` (stesso pacchetto di samtools)
  - **`bcftools call -O v -c -v mapping.vcf > variants.vcf`**
- Altri metodi più sofisticati:
  - GATK,
  - FreeBayes,
  - VarScan

Variant Call Format (VCF)

Header

Lines starting with ##: arbitrary number of meta-information lines

Line starting with #: column definition (8 mandatory):

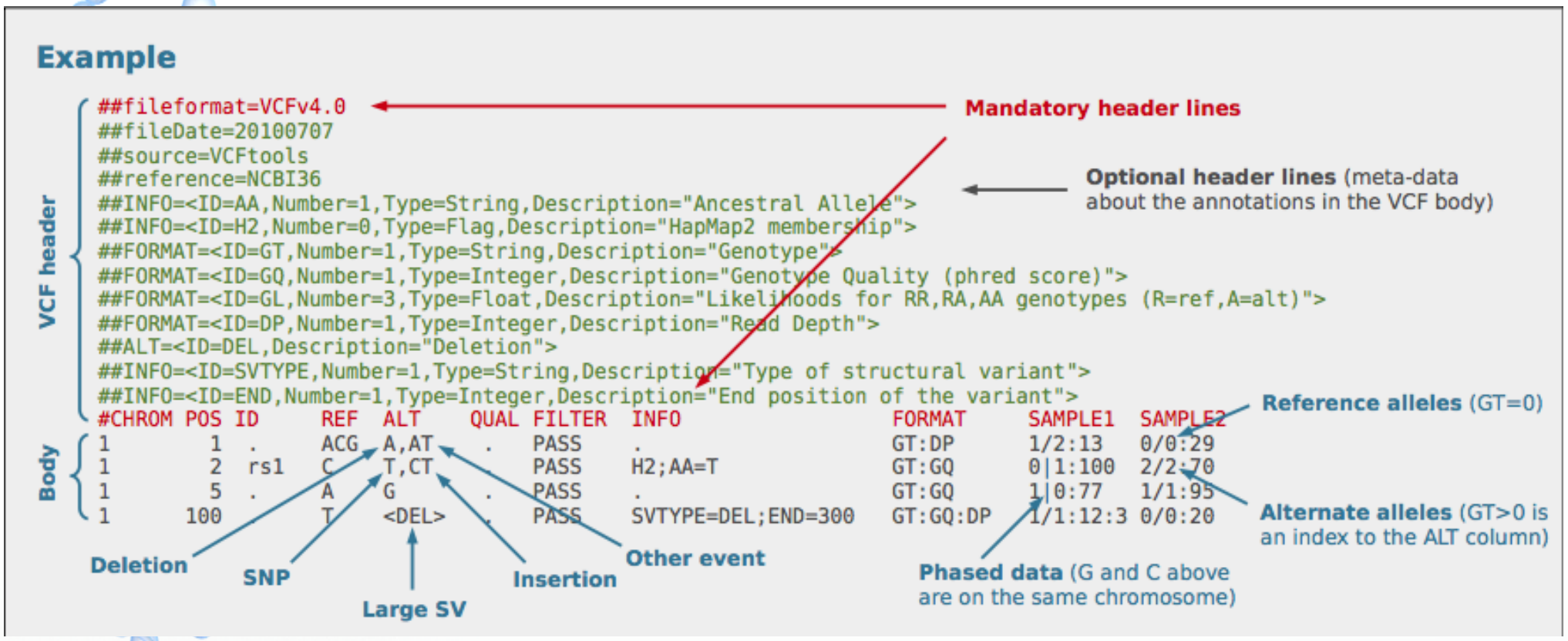
- CHROM = chromosome
- POS = start position of the variant
- ID = unique identifier of the variant (e.g. Number for SNPs)
- REF = reference allele
- ALT = comma separated list of alternate alleles
- QUAL = phred-scaled quality score
- FILTER = site filtering information
- INFO = user extensible annotation (e.g. snpEff, Annovar)
- FORMAT = an (optional) extensible list of fields for describing the SAMPLE column
- SAMPLE COLUMN = free

Data

One line per site (all columns described above per line); useful information per site and per sample

http://samtools.github.io/hts-specs/VCFv4.2.pdf

Esempio di un file VCF



Varianti in IGV