Bioinformatics Analysis Tools for NGS Data

Read mapping

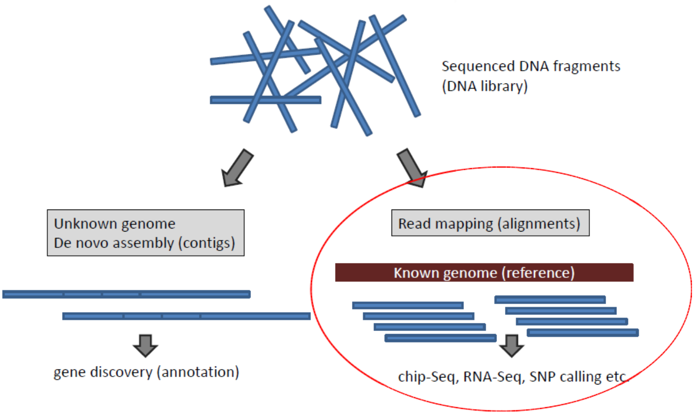
Philipp Rescheneder, Moritz Smolka

April 27, 2016

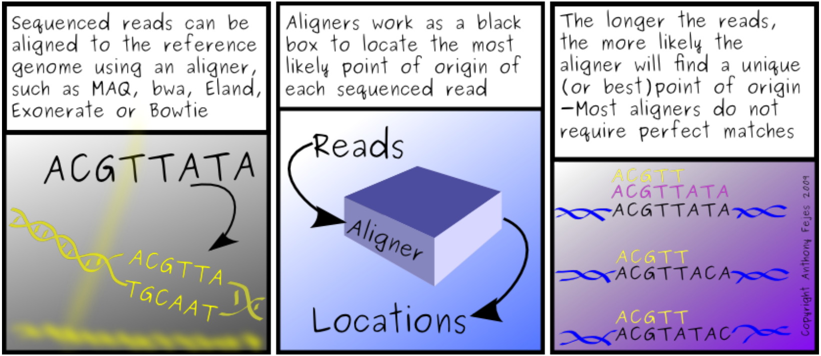
Center for Integrative Bioinformatics Vienna
Max F. Perutz Laboratories

CIBIV
Center for Integrative Bioinformatics Vienna

Making sense of NGS data


Sequenced DNA fragments
(DNA library)
Unknown genome
De novo assembly (contigs)
gene discovery (annotation)
Read mapping (alignments)
Known genome (reference)
chip-Seq, RNA-Seq, SNP calling etc.

Read mapping


Sequenced reads can be aligned to the reference genome using an aligner, such as MAQ, bwa, Eland, Exonerate or Bowtie
ACGTTATA
ACGTTA
TGCAAT
Aligners work as a black box to locate the most likely point of origin of each sequenced read
Reads
Aligner
Locations
The longer the reads, the more likely the aligner will find a unique (or best) point of origin
–Most aligners do not require perfect matches
ACGTT
ACGTTATA
ACGTTATA
ACGTT
ACGTTACA
ACGTT
ACGTATAC
Copyright Anthony Fejes 2009

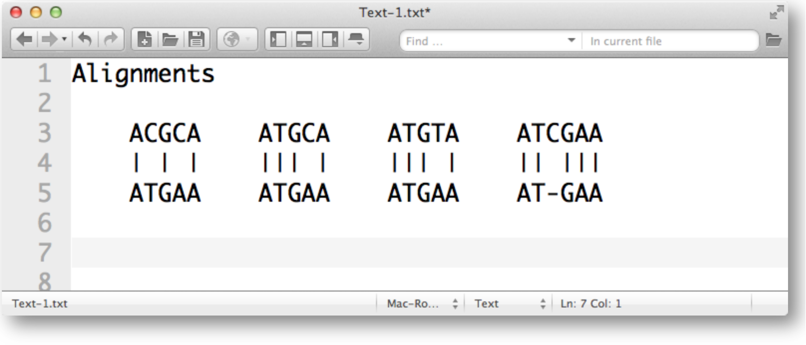
Sequence Alignments

- A way to arrange sequences to identify regions of similarity

```
ATGCAAACAAG
||||  |  ||
ATGCTTATTAG
```

- We look for similarity because it may be a consequence of functional, structural or evolutionary relationship

How to pick the right alignment?



```
Alignments

    ACGCA    ATGCA    ATGTA    ATCGAA
    | | |    ||| |    ||| |    || |||
    ATGAA    ATGAA    ATGAA    AT-GAA
```

We need a scoring scheme:

- positive values to reward matches
- negative values to penalize mismatches

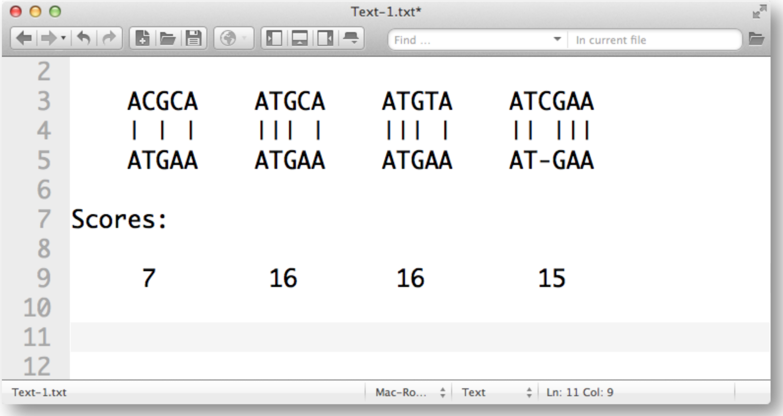
Scoring Alignments

- a match: usually a positive score (5)
- a mismatch: usually a negative score, may depend on the kind of mismatch (-4)
- gap opening: usually the most penalized action (-10)
- gap extension: making the gap longer (-0.5)

Important:

- any two sequences can be aligned, the alignment score represents the sum of the each match/mismatch/gap/gap extension
- There is no universally best alignment only the best alignment for a given scoring scheme
- Most aligners will only report alignments that make some sense

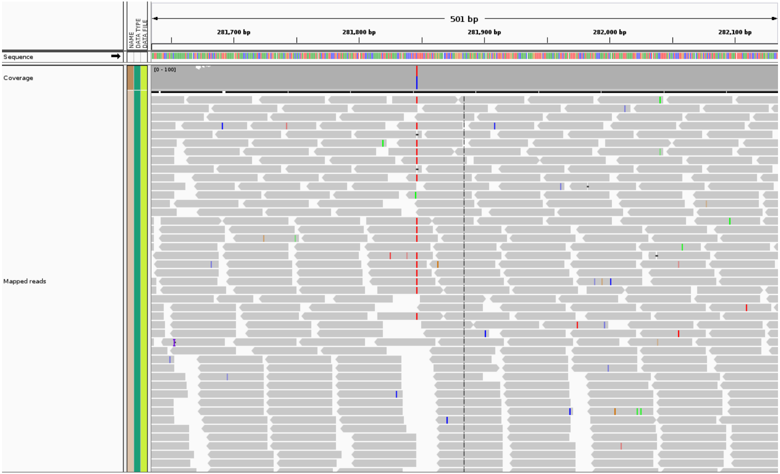
How to pick the right alignment?



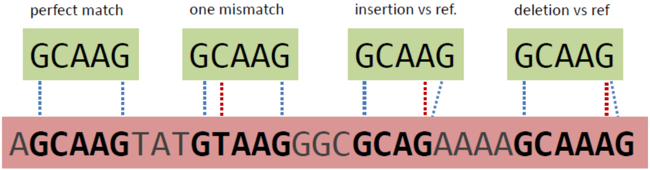
```

     ACGCA      ATGCA      ATGTA      ATCGAA
     | | |      ||| |      ||| |      || |||
     ATGAA      ATGAA      ATGAA      AT-GAA

Scores:

       7         16         16          15



```

Remember: scoring matrices determine which alignment is optimal

Mapping to a genome



Differences between read and reference

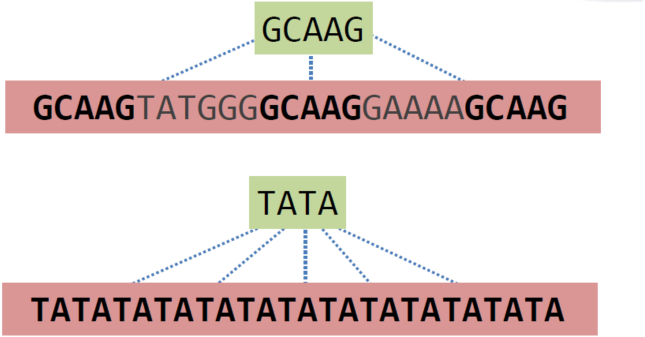


NOTE: mismatches or indels can be longer than 1 base!

It gets complicated very quickly

Alignment scoring depends on mismatch scoring (different across bases!), gap open, gap extension penalties

Repeats



GCAAG

GCAAGTATGGG**GCAAG**GAAAA**GCAAG**

TATA

TATATATATATATATATATATATATA

Short read mappers (aligners)

- Can be **optimal** or heuristic **not all hits will be found**
- Optimal alignment are computationally more demanding
- Not feasible for large genomes and data sets
- Use heuristics to quickly identify locations (hits) where the reads match
- Tradeoff: resource usage vs speed vs accuracy vs usability
- Each domain of applications may have more appropriate tools
- There is no single best tool

- Hash based seed and extend (BLAST)
- Tree/Trie based approach (suffix/prefix trees, burrows wheeler transformation, etc.)

Burrows Wheeler transform methods

- Used by Bowtie2, BWA
- Methodology
  1. Compress the genome into a very efficient data structure
  2. Uses a k difference search
- Advantage:

  Typically faster than seed based methods
- Disadvantage:

  Performance decreases exponentially with the number of mismatches.

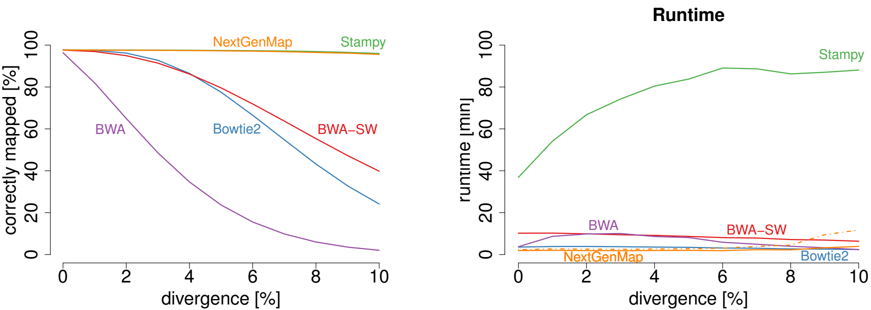
Hash based methods

- Used by Stampy, NextGenMap
- Methodology:
  1. Identify regions with local similarity
  2. Align sub regions of genome with the read
- Advantage:

  More sensitive than BWT
- Disadvantage:

  Typically slower

Read mapper comparison



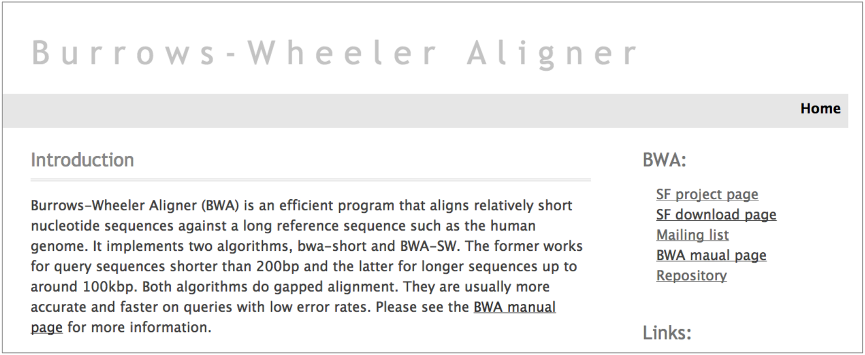
NextGenMap

- Input formats: FASTA/Q, SAM, BAM
- Output formats: SAM, BAM
- Uses GPU(s) to reduce runtime (optional)
- Run:

```
$ ngm -r reference.fasta -q single_end.fastq -o output.sam
$ ngm -r reference.fasta -1 first_mate.fastq -2 second_mate.fastq -o output.sam -t 4
```

- `http://cibiv.github.io/NextGenMap/`

BWA



Burrows-Wheeler Aligner

Introduction

Burrows-Wheeler Aligner (BWA) is an efficient program that aligns relatively short nucleotide sequences against a long reference sequence such as the human genome. It implements two algorithms, bwa-short and BWA-SW. The former works for query sequences shorter than 200bp and the latter for longer sequences up to around 100kbp. Both algorithms do gapped alignment. They are usually more accurate and faster on queries with low error rates. Please see the BWA manual page for more information.

BWA:

- SF project page
- SF download page
- Mailing list
- BWA maual page
- Repository

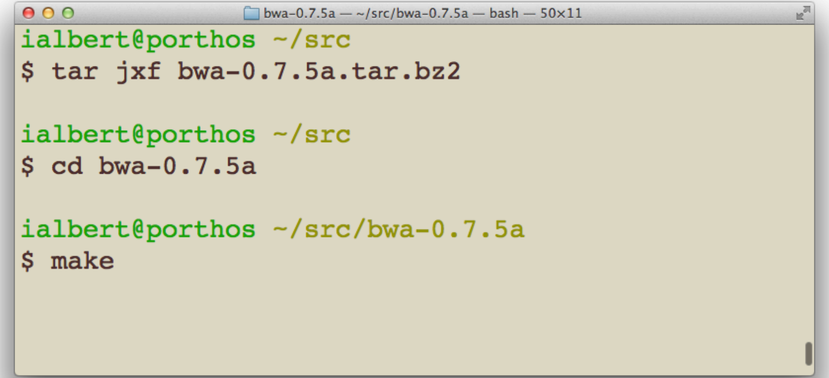
Links:

http://bio-bwa.sourceforge.net/

Download, unpack, compile with make and link to bin

Read also the **bwa-mem** controversy – the **bwa-mem** paper rejection

Install BWA

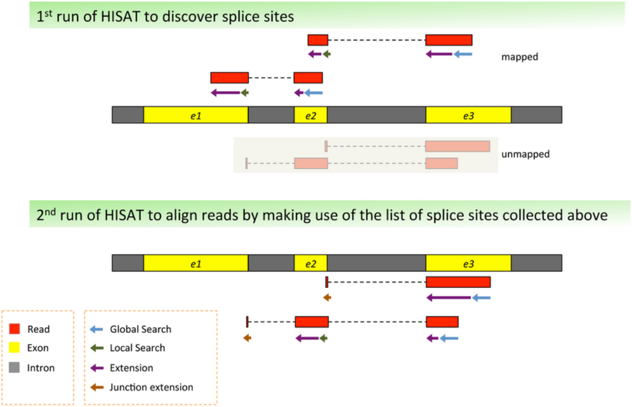


```
ialbert@porthos ~/src
$ tar jxf bwa-0.7.5a.tar.bz2

ialbert@porthos ~/src
$ cd bwa-0.7.5a

ialbert@porthos ~/src/bwa-0.7.5a
$ make
```

Uses **bz2** compression hence the **j** flag (the **z** command would expand a gzip file)

HISAT

- Very efficient RNA-Seq mapper
- Has to split reads that span splice sites
- `http://www.ccb.jhu.edu/software/hisat/index.shtml`



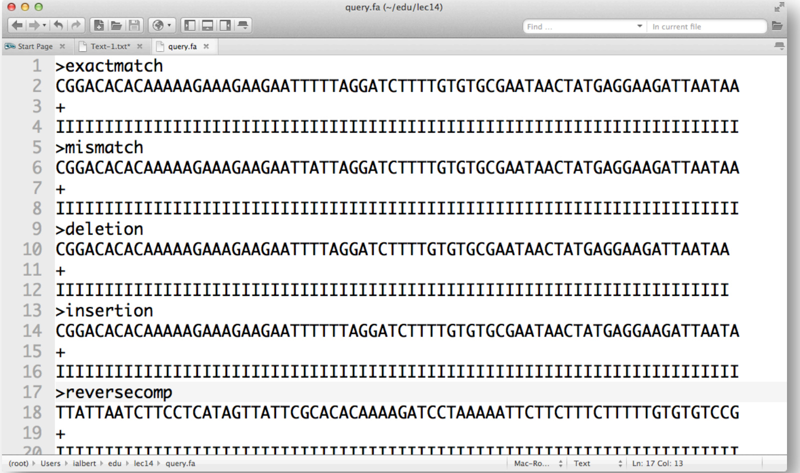
1st run of HISAT to discover splice sites
mapped
e1
e2
e3
unmapped
2nd run of HISAT to align reads by making use of the list of splice sites collected above
e1
e2
e3
Read
Exon
Intron
Global Search
Local Search
Extension
Junction extension

SAM: Sequence Alignment format

SAM/BAM is the de-facto standard for storing (short) read alignments

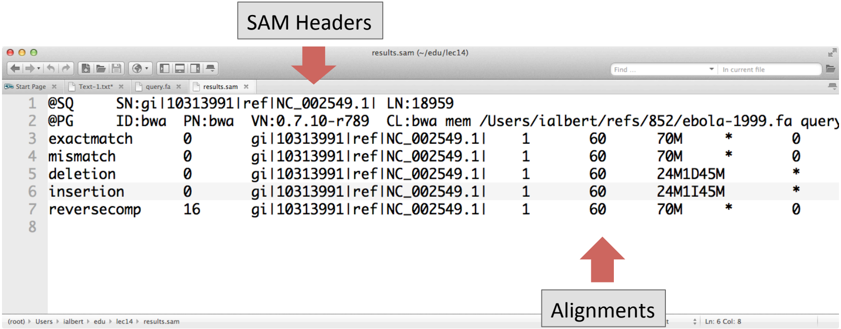
- **SAM:** A TAB-delimited text format consisting of a header section, which is optional, and an alignment section
- **BAM:** compressed version of SAM (same information)
  - Smaller ($<$ 30% of SAM)
  - Not human readable
  - Often used to store unaligned reads as well (instead of FASTQ)

Create test reads



```
>exactmatch
CGGACACACAAAAAGAAAGAAGAATTTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
>mismatch
CGGACACACAAAAAGAAAGAAGAATTATTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
>deletion
CGGACACACAAAAAGAAAGAAGAATTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
>insertion
CGGACACACAAAAAGAAAGAAGAATTTTTTAGGATCTTTTGTGTGCGAATAACTATGAGGAAGATTAATA
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
>reversecomp
TTATTAATCTTCCTCATAGTTATTCGCACACAAAAGATCCTAAAAATTCTTCTTTCTTTTTGTGTGTCCG
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

Resulting SAM file after mapping



SAM Headers

```
@SQ     SN:gi|10313991|ref|NC_002549.1|  LN:18959
@PG     ID:bwa   PN:bwa   VN:0.7.10-r789   CL:bwa mem /Users/ialbert/refs/852/ebola-1999.fa query
exactmatch      0       gi|10313991|ref|NC_002549.1|     1       60      70M     *       0
mismatch        0       gi|10313991|ref|NC_002549.1|     1       60      70M     *       0
deletion        0       gi|10313991|ref|NC_002549.1|     1       60      24M1D45M        *
insertion       0       gi|10313991|ref|NC_002549.1|     1       60      24M1I45M        *
reversecomp     16      gi|10313991|ref|NC_002549.1|     1       60      70M     *       0
```

Alignments

SAM header

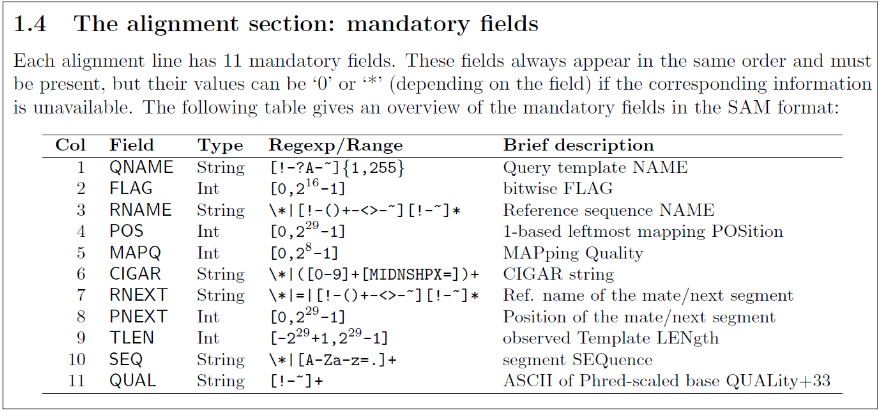
Stores information like:

- File format version
- Information about the reference sequence (name, length, etc.)
- Read group (optional)
- Programs used to create SAM/BAM file including parameters (optional)
- Additional comments (optional)

Example:

```
@HD     VN:1.0  SO:unsorted
@SQ     SN:comp1_c0_seq1        LN:216
@SQ     SN:comp2_c0_seq1        LN:351
@SQ     SN:comp5_c0_seq1        LN:225
```

Reads: 11 tab delimited columns



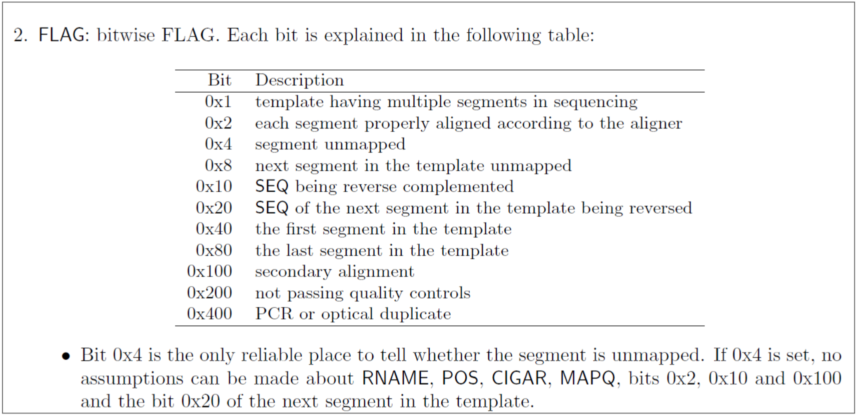
1.4 The alignment section: mandatory fields

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:

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^{29}-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 segment
8	PNEXT	Int	[0,2^{29}-1]	Position of the mate/next segment
9	TLEN	Int	[-2^{29}+1,2^{29}-1]	observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

Column 1 and 2: QNAME and FLAG

QNAME: the name of the query sequence



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

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

- 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.

FLAG: bitwise representation

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

The flag **11** → **1** + **2** + **8** = **0001011** (conditions 1, 2 and 8 satisfied)

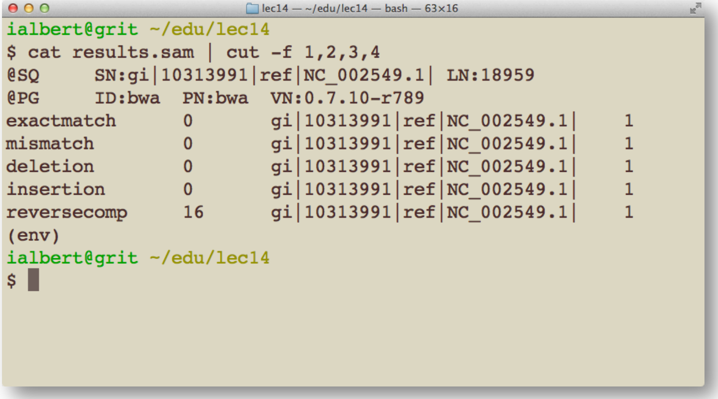
It is used to save space – but it does make things a bit more difficult.

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

Column 3 and 4, RNAME and POS



```
ialbert@grit ~/edu/lec14
$ cat results.sam | cut -f 1,2,3,4
@SQ     SN:gi|10313991|ref|NC_002549.1| LN:18959
@PG     ID:bwa  PN:bwa  VN:0.7.10-r789
exactmatch      0       gi|10313991|ref|NC_002549.1|    1
mismatch        0       gi|10313991|ref|NC_002549.1|    1
deletion        0       gi|10313991|ref|NC_002549.1|    1
insertion       0       gi|10313991|ref|NC_002549.1|    1
reversecomp     16      gi|10313991|ref|NC_002549.1|    1
(env)
ialbert@grit ~/edu/lec14
$ 
```

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)

SAM alignments: mapping quality (MQ)

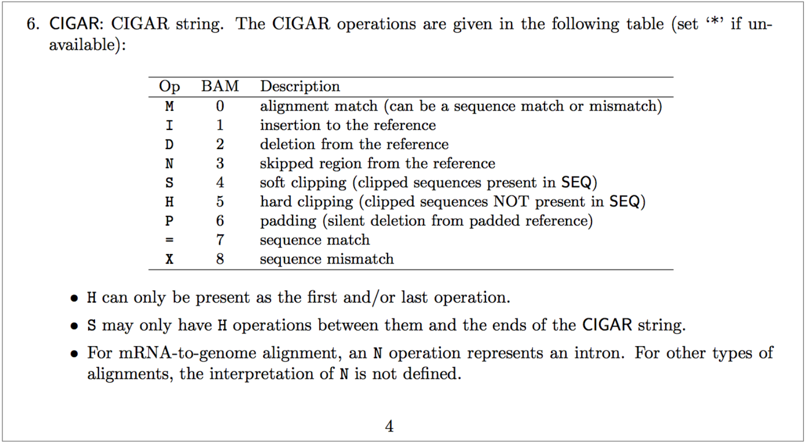
- Phred score, identical to the quality measure in the FASTQ file. Quality Q, probability P

$$P = 10^{\frac{-Q}{10.0}}$$

- if $Q = 30$, $P = \frac{1}{1000} \Rightarrow$ on average, one out of 1000 alignments will be wrong
- As good as this sounds it is not easy to compute such a quality
- Mostly only a very basic estimation. Every program does it differently
- Allowed range in SAM is $0 - 254$. 255 means not available
- Reasons for low MQ include repeats, low base quality, high number of mismatches, etc.
- MQ range typically from 0 to 60

Column 6: CIGAR

- CIGAR = Compact Idiosyncratic Gapped Alignment Report

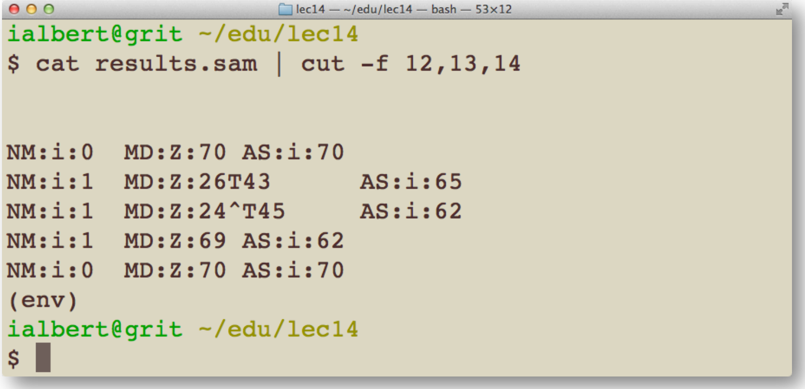


6. CIGAR: CIGAR string. The CIGAR operations are given in the following table (set '*' if unavailable):

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

- 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.

Additional tags



```
ialbert@grit ~/edu/lec14
$ cat results.sam | cut -f 12,13,14


NM:i:0  MD:Z:70 AS:i:70
NM:i:1  MD:Z:26T43      AS:i:65
NM:i:1  MD:Z:24^T45     AS:i:62
NM:i:1  MD:Z:69 AS:i:62
NM:i:0  MD:Z:70 AS:i:70
(env)
ialbert@grit ~/edu/lec14
$ 
```

Specific information about the alignment process that the tools was able to establish.
more details in later lectures

SAM alignments: additional flags

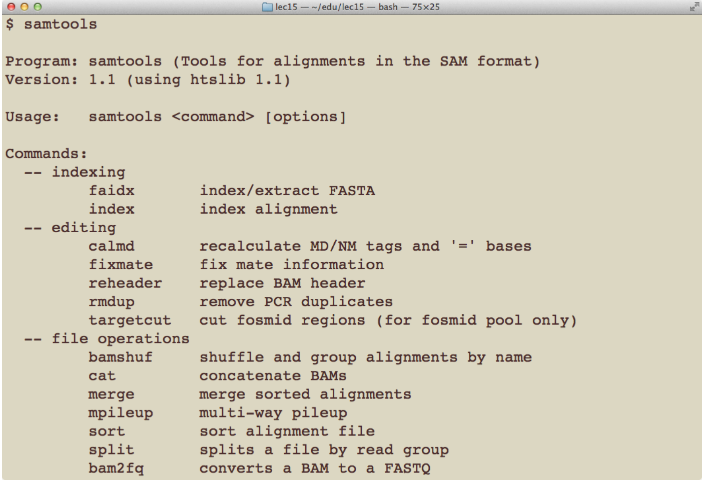
- The SAM/BAM format defines a wide range of optional fields
- NextGenMap for example uses the following ones:

AS	Alignment sore
NM	Number of mismatches in the alignment
XI	Identity of the alignment
X0	Number of equal scoring hits
X1	Number of suboptimal hits found
XE	Number of supported seeds
XR	Number of aligned residues
MD	Mismatched and deleted positions/bases

Working with SAM files: Tools

- Samtools:
  - Command line
  - Convert SAM ↔ BAM
  - SNP calling
  - Various file operations on SAM/BAM
  - Visualization of alignments
  - etc....
- Picard Tools:
  - Various file operations on SAM/BAM files
  - Statistics about mapping

Samtools



```
$ samtools

Program: samtools (Tools for alignments in the SAM format)
Version: 1.1 (using htslib 1.1)

Usage:   samtools <command> [options]

Commands:
  -- indexing
        faidx       index/extract FASTA
        index       index alignment
  -- editing
        calmd       recalculate MD/NM tags and '=' bases
        fixmate     fix mate information
        reheader    replace BAM header
        rmdup       remove PCR duplicates
        targetcut   cut fosmid regions (for fosmid pool only)
  -- file operations
        bamshuf     shuffle and group alignments by name
        cat         concatenate BAMs
        merge       merge sorted alignments
        mpileup     multi-way pileup
        sort        sort alignment file
        split       splits a file by read group
        bam2fq      converts a BAM to a FASTQ
```

different commands

SAM to BAM

transform to bam

```
samtools view –Sb input.sam > tempfile.bam
```

sort bam file

```
samtools sort –f tempfile.bam output.bam
```

Index bam file

```
samtools index output.bam
```

Filtering SAM/BAM files

Required flag (keep if matches)

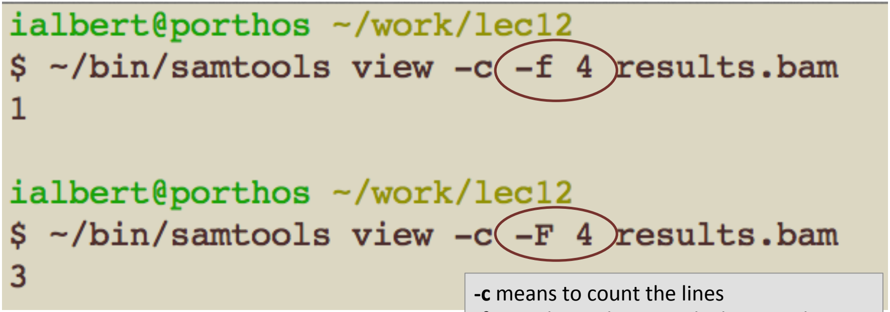
```
samtools view –f
```

Filtering flag (remove if matches)

```
samtools view –F
```

Bitwise flags

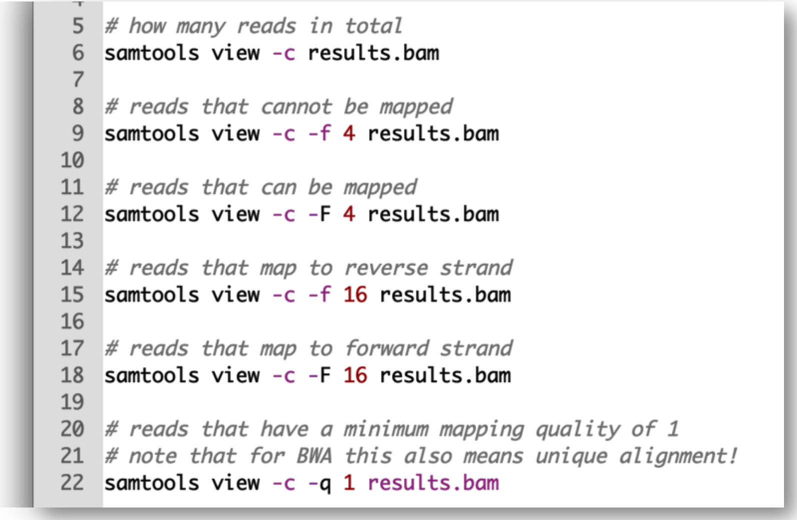
1	= 00000001	→	paired end read
2	= 00000010	→	mapped as proper pair
4	= 00000100	→	unmapped read
8	= 00001000	→	read mate unmapped
16	= 00010000	→	read mapped on reverse strand



```
ialbert@porthos ~/work/lec12
$ ~/bin/samtools view -c -f 4 results.bam
1

ialbert@porthos ~/work/lec12
$ ~/bin/samtools view -c -F 4 results.bam
3
```

-c means to count the lines
-f <number> - keep reads that match
-F <number> - remove reads that match

Samtools examples



```
# how many reads in total
samtools view -c results.bam

# reads that cannot be mapped
samtools view -c -f 4 results.bam

# reads that can be mapped
samtools view -c -F 4 results.bam

# reads that map to reverse strand
samtools view -c -f 16 results.bam

# reads that map to forward strand
samtools view -c -F 16 results.bam

# reads that have a minimum mapping quality of 1
# note that for BWA this also means unique alignment!
samtools view -c -q 1 results.bam
```

Other samtools commands

Flag statistics

```
samtools flagstat data.bam
```

Index stats

```
samtools idxstats data.bam
```

Depth of coverage

```
samtools depth data.bam | head
```

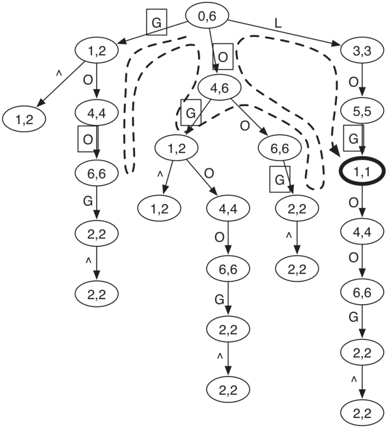
Summary

- Mapping is one of the first steps in data analysis
- Biases introduced here will effect all down-stream analysis steps
- Several different read mappers available. All with different advantages, disadvantages
- SAM/BAM is the standard format for storing short read alignments
- Always look at your data after mapping!

Extra slides

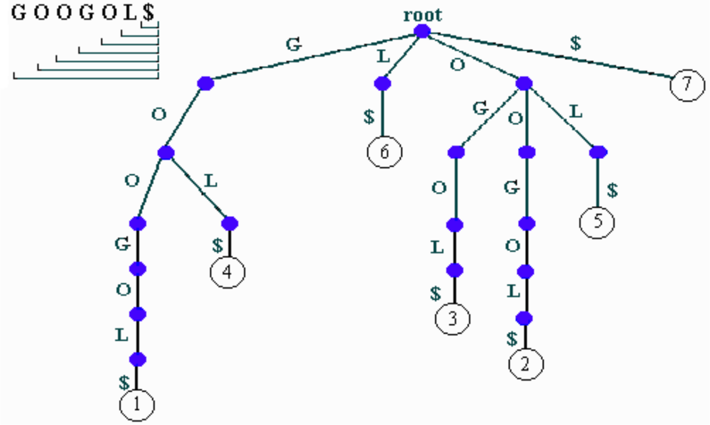
Prefix tree (BWA)

- Search “LOL” in “GOOGOL”



Suffix tree

- Search “GOL” in “GOOGOL”

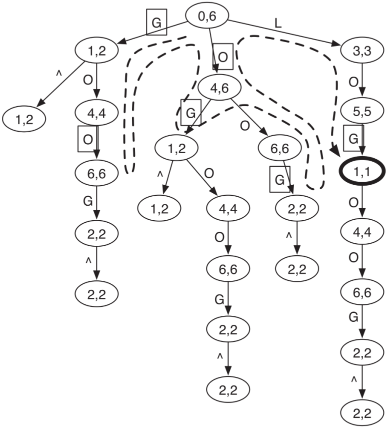


Suffix tree: problems

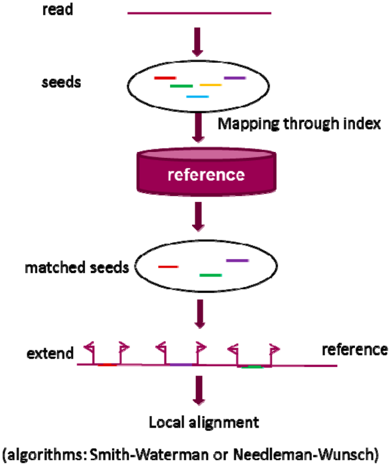
- Only for exact matching strings
- But, we have mismatches, insertions and deletions

Prefix tree

- Search “LOL” in “GOOGOL”



Seed and extend


read
seeds
Mapping through index
reference
matched seeds
extend
reference
Local alignment
(algorithms: Smith-Waterman or Needleman-Wunsch)