

Alignment (II)

Bioinformatics Applications (PLPTH813)

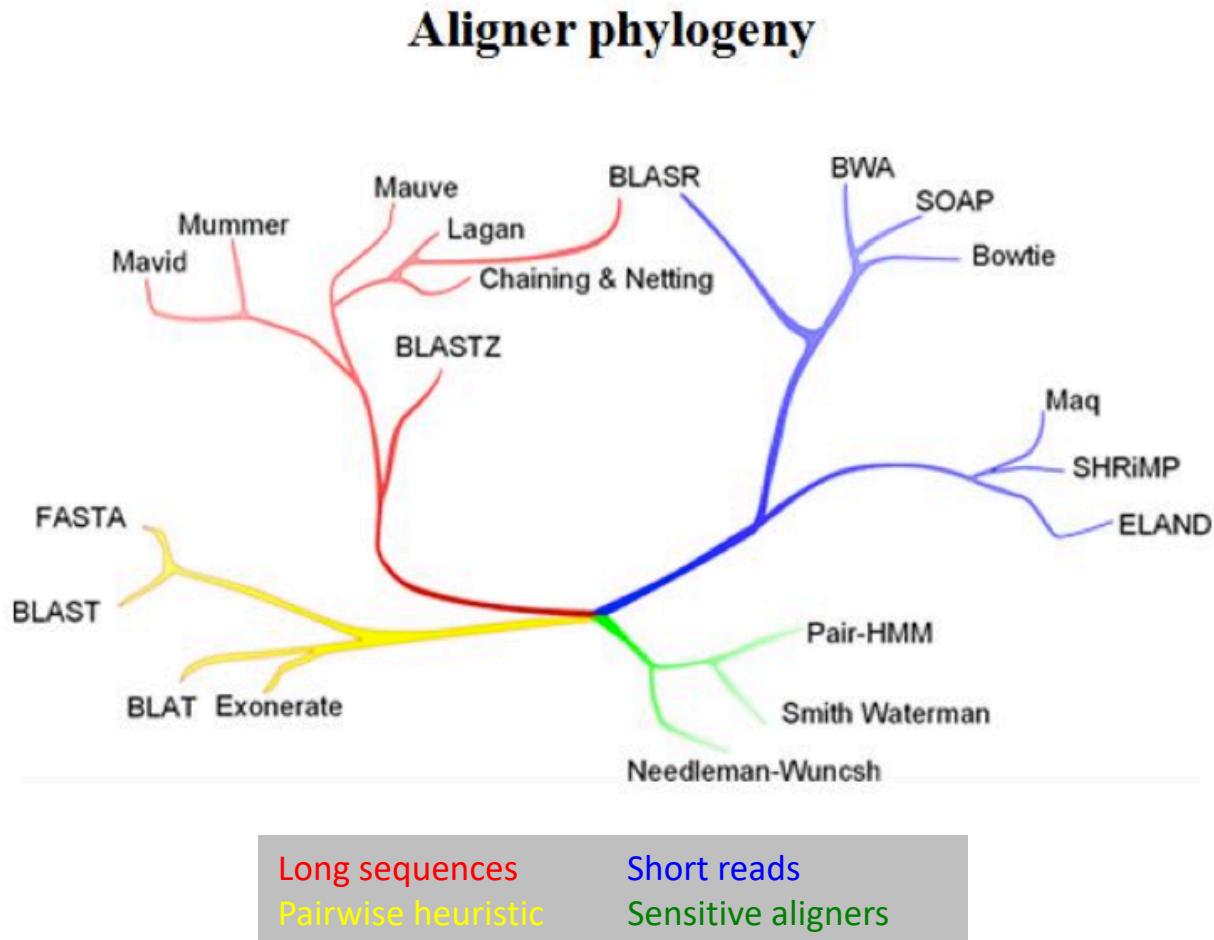
Sanzhen Liu

3/2/2021

Alignment I - review

- The local alignment algorithm, **Smith–Waterman algorithm**, ensures the best performance on accuracy and the most precise results with respect to its scoring scheme but it is very time-consuming.
- **BLAST** is a seed-extension algorithm for database searching.
 1. Identify a match for the seed sequence
 2. Extend match using a local alignment
- However, BLAST is still not fast enough for efficient alignments of millions of NGS reads.

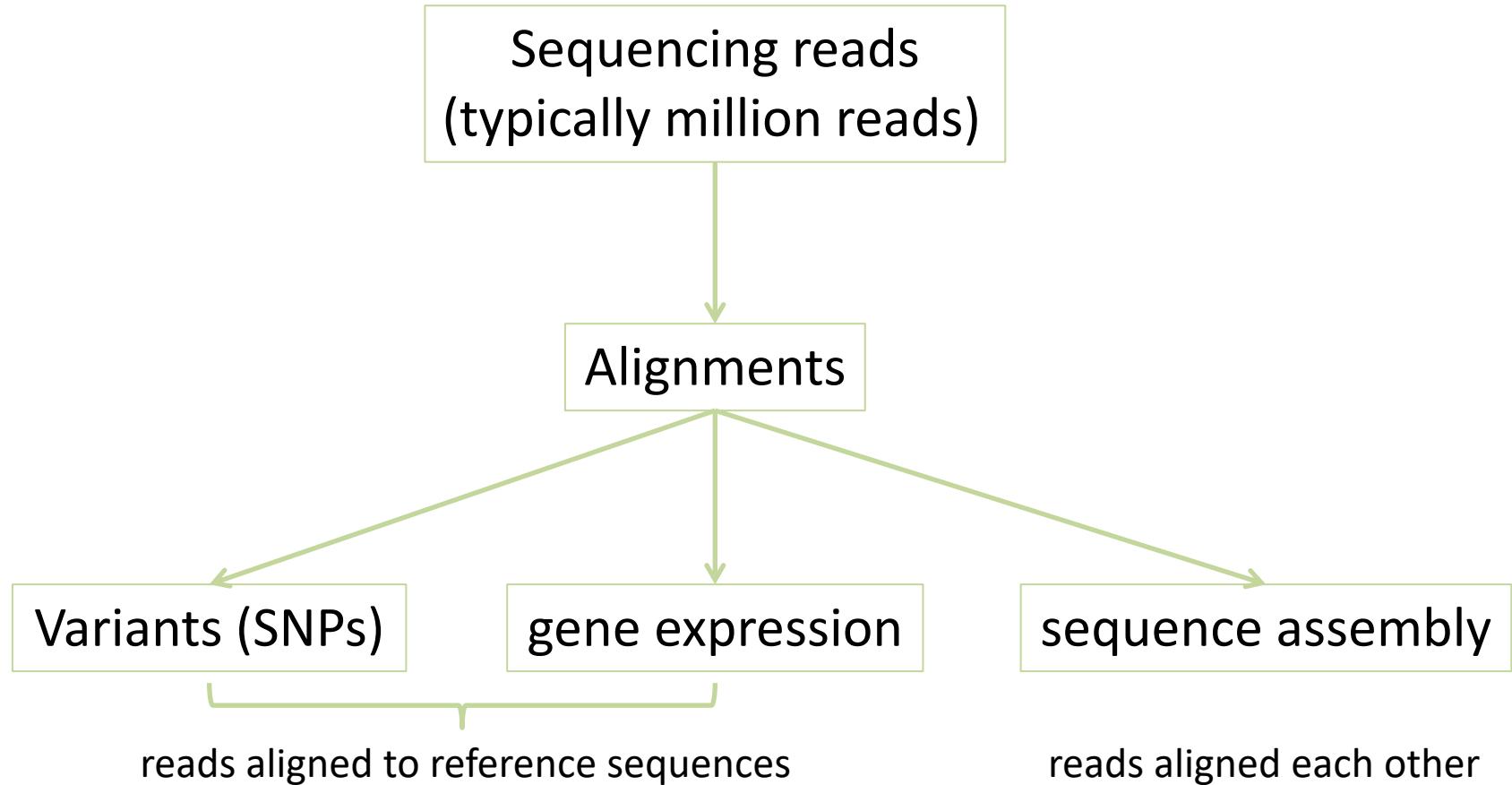
Relationship among different algorithms



Outline

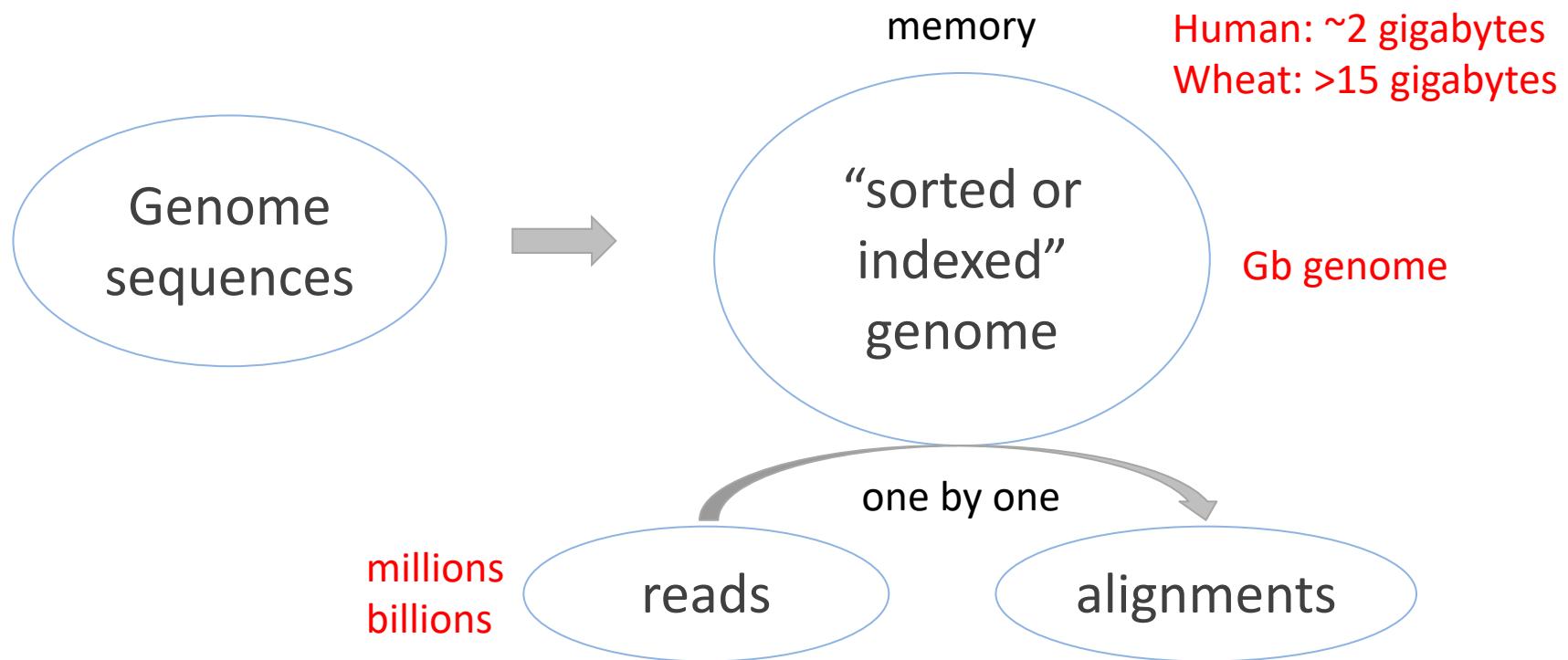
- One of short-read alignment algorithms – BWT
- A short-read aligner - BWA
- Standard alignment format – SAM and BAM
- A tool to handle alignments - SAMtools
- An alignment visualization tool - IGV

Alignment is a central step for most NGS analyses



Short-read alignments

1. “Sorting or indexing” genome sequences in a smart way to accelerate the alignment
2. Keeping the “Sorting or indexing” data at a reasonable size



3. A good aligner needs to find a good tradeoff between Speed and Accuracy.

Indexing speeds up searching



un-indexed library



indexed library

Indexing algorithms

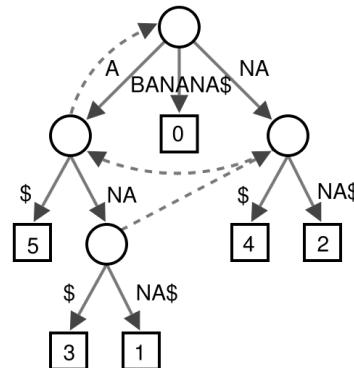
A benefit of string indexing is to avoid scanning the whole reference. Therefore alignment searching is much faster.

BWT

Input	BWT
^BANANA	BNN^AA A

<2 gigabytes

Suffix Tree



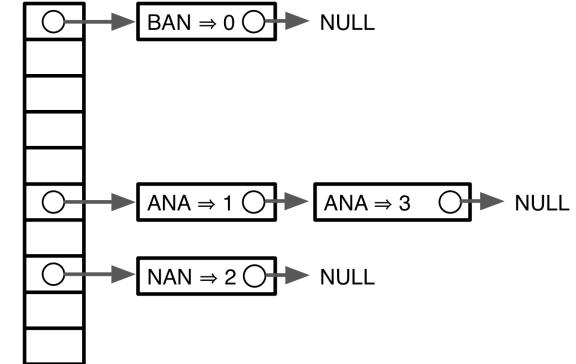
>35 gigabytes

Suffix Array

6	\$
5	A\$
3	ANA\$
1	ANANA\$
0	BANANA\$
4	NA\$
2	NANA\$

>12 gigabytes

Hash table



Human genome memory footprint*

Burrows–Wheeler transform

Transformation algorithm*

Input	All Rotations	Sorting all rows	Taking last character	BWT output
acaacg\$	acaacg\$	\$acaacg	\$acaacg	gc\$aaac
	\$acaacg	aacg\$ac	aacg\$ac	
	g\$acaac	acaacg\$	acaacg\$	
	cg\$acaa	acg\$aca	acg\$aca	
	acg\$aca	caacg\$a	caacg\$a	
	aacg\$ac	cg\$acaa	cg\$acaa	
	caacg\$a	g\$acaac	g\$acaac	



Before

After

itsgravybaby.com

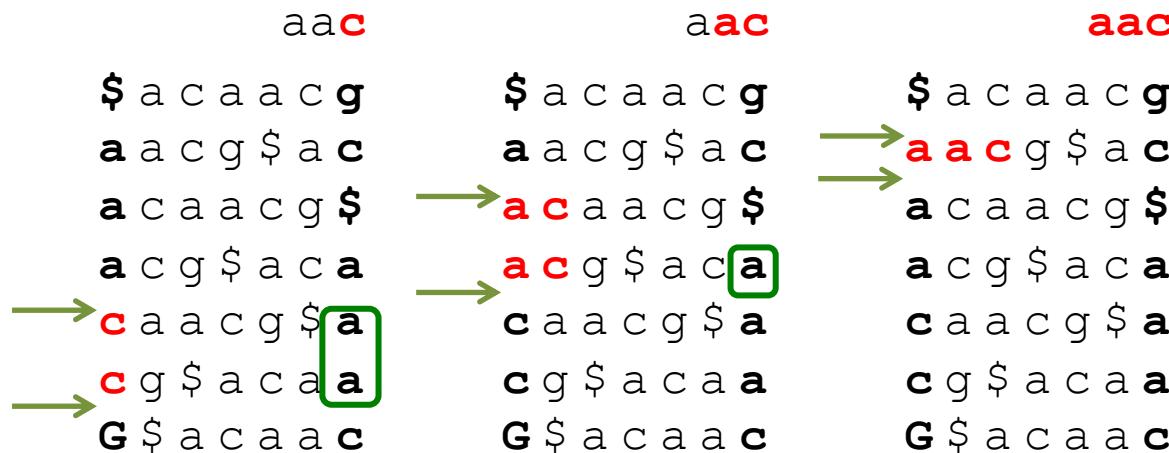
acaacg\$	\$ a c a a c g	g c \$ a a a c
	a a c g \$ a c	
	a c a a c g \$	
	a c g \$ a c a	
	c a a c g \$ a	
	c g \$ a c a a	
	G \$ a c a a c	

- The transformation is **reversible**, without needing to store any additional data.
- BWT is a **compression** technique to find repeated patterns and encoding the duplications more compactly. (“bzip” is based on this compression)

Matching search

Find exact match of aac in the sequence of acaacg

1. BWT(acaacg) is gc\$aaac
2. Backward searching



Ultrafast and memory-efficient alignment of short DNA sequences
to the human genome

Ben Langmead, Cole Trapnell, Mihai Pop and Steven L Salzberg

Bowtie

Fast and accurate short read alignment with Burrows–Wheeler transform

Heng Li and Richard Durbin*

BWA

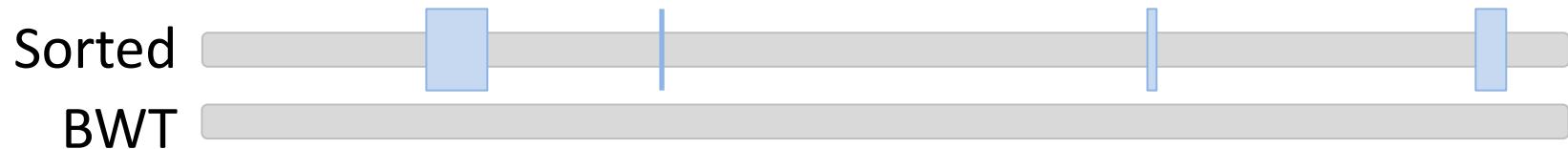
BWT searching



5' CGTAGCA 3'

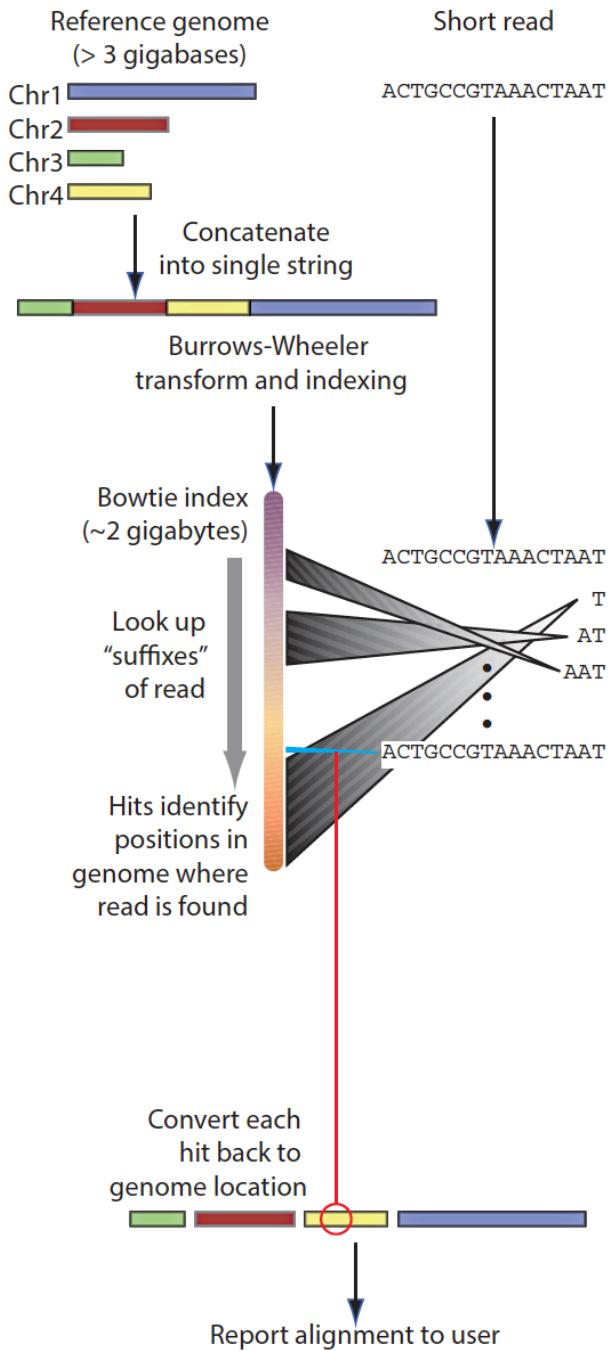
The sequence is shown in black text with a 5' and 3' orientation. The last four bases, GCA, are highlighted with orange vertical bars, indicating they are being searched for in the BWT data.

backward searching



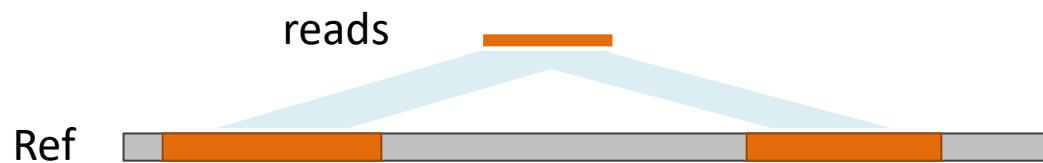
5' CGTAGCA 3'

Working flow



Alignment issues

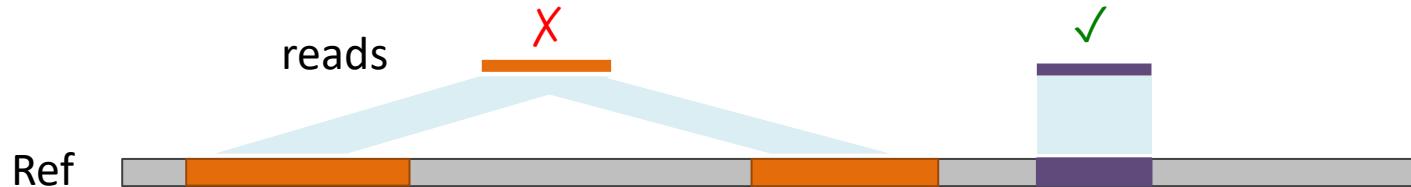
- Repeats



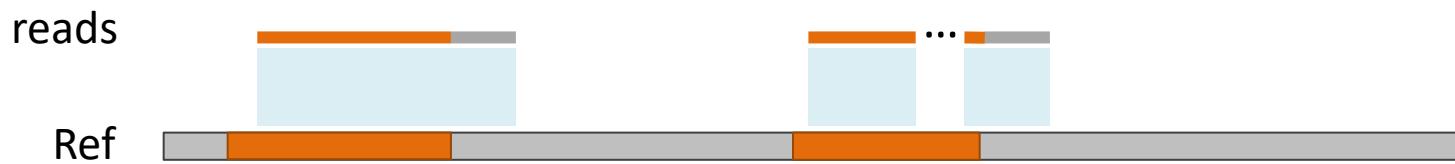
- Sequencing errors
- Polymorphisms (reference and sequenced sample)
- Quality of reference genomes (mis-assembly and incomplete genome)

Solutions

- Unique mapped reads



- Longer reads or Paired-end reads



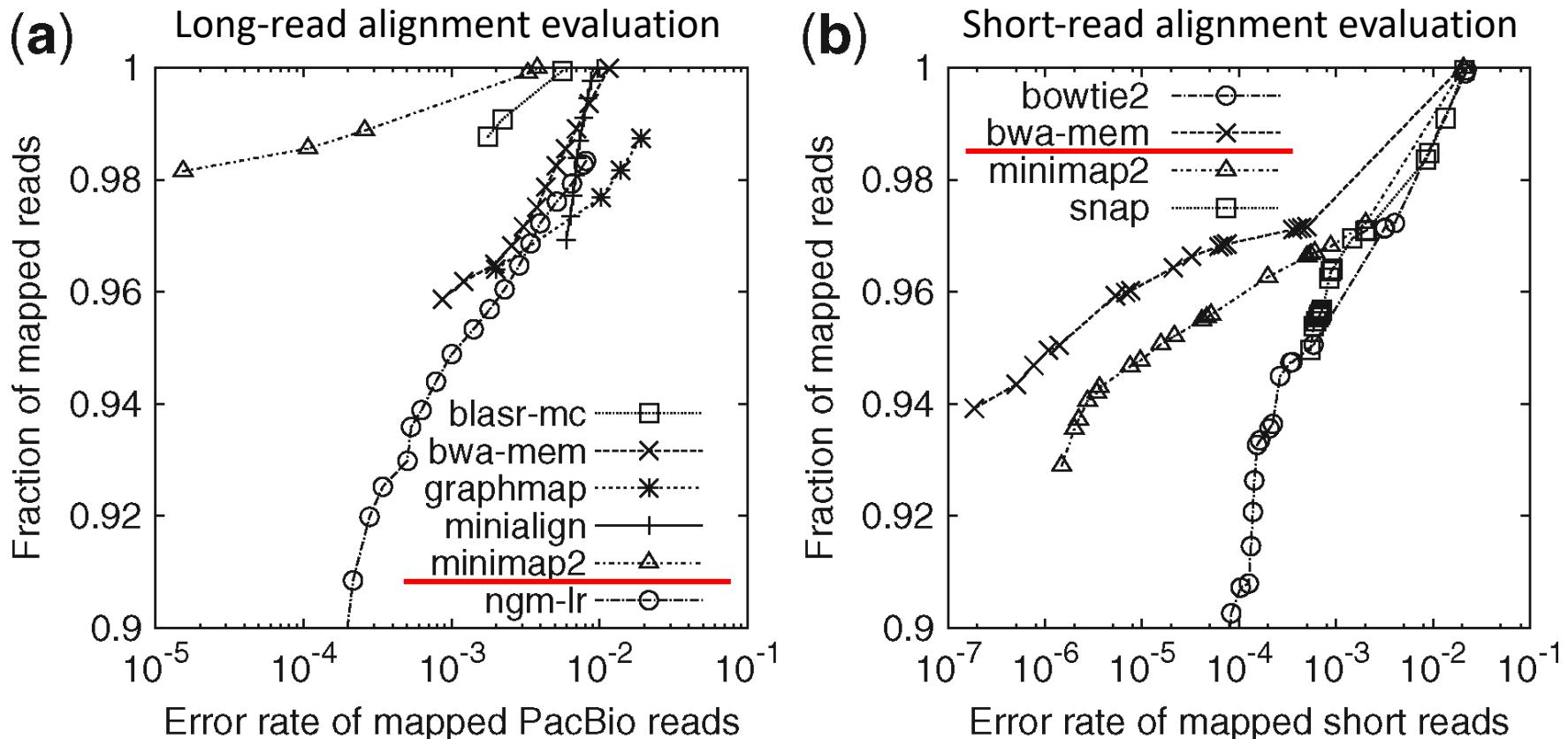
- Tolerance of mismatches or gaps for each alignment



The aligner: BWA

- BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome.
- It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM.
- BWA-backtrack is designed for Illumina sequence reads up to 100bp, while BWA-SW and BWA-MEM are for longer sequences ranged from 70bp to 1Mbp.
- **BWA-MEM**, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate.

bwa, bowtie2, minimap2



Align queries (reads) to the reference with BWA-MEM

- Alignment using “mem”

```
bwa mem ref.fa reads.fq > se.sam
```

```
bwa mem ref.fa pair1.fq pair2.fq > pe.sam
```

Build a BWA sequence index database

- Index reference sequences

```
bwa index ref.fa
```

example

```
bwa index MG1655.fasta
```

-----output-----

MG1655.fasta.amb

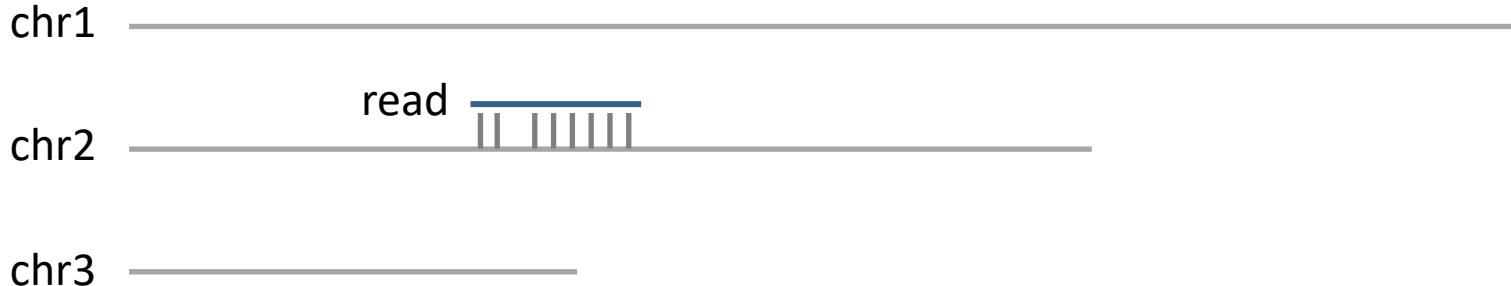
MG1655.fasta.ann

MG1655.fasta.bwt

MG1655.fasta.pac

MG1655.fasta.sa

Question



What parameters/items are needed to describe an alignment between a read and a reference?



And how about an alignment of a paired-end read to the reference?

Alignment format: SAM

- **SAM** stands for Sequence Alignment/Map format.
- The Sequence Alignment/Map (SAM) format is a generic alignment format for storing read alignments against reference sequences.
- The SAM/BAM format, together with **SAMtools**, enables a generic and modular approach to the analysis of genomic sequencing data.

More about SAM/BAM format and samtools

Heng Li's blog

Archive Categories Pages Tags

SAM/BAM/samtools is 10 years old

21 December 2018

Feature Year first available Target Feature descriptions

1	2009	BAM	End-of-file marker to detect truncated files
2	2009	SAM	"="/"X" CIGAR operators to encode sequence matches or mismatches
3	2011	SAM	"B" tag type to efficiently store binary arrays
4	2012	BAM	CSI index for chromosomes longer than 512Mbp
5	2012	htslib	BCFv2 as an efficient binary representation of VCF files
6	2013	SAM	Supplementary alignment to encode split alignment
7	2014	htslib	CRAMv3 as a more compact binary representation of SAM
8	2015	htslib	Direct access to remote files over internet
9	2017	BAM	BAM extension to support CIGARs with >65535 operators
10	2017	htslib	Htsget protocol for efficient database access
11	2017	htslib	CRC error checking to detect internally corrupted BAM files

Examples of SAM alignments

@SQ SN:1 LN:307041717

@SQ SN:2 LN:244442276

3

HISEQ:446:C9KP3ANXX:3:1102:8028:81847 99 1 162 40 63M1I61M = 285 234
AATGGTCGTTCATGGCTATTTCGACAAAAATGGGGGTTGTGTGGCAATTGATCATCGACCAGAAGCT
CATACACCTCACCCCCACATATGTTCCTGCCATAGATCACATTCTTGGATTCTGG
BBCBBGDEGGGGGG>FGGGGGGGGFGGGGGEGEGGG9C<FGGGDGBGGGGGGFG>@GGGD
GGGC@GDGGGG0CG>GBG0<CCGGGGGFGGCFG0CCGGGG=8800:8@CDD/88@C8DDG
NM:i:2 MD:Z:46C77 AS:i:112 XS:i:112 RG:Z:A188-1

SAM section 1: headers

In the header section, the line starts with '@'

- @SQ Reference sequence dictionary.
- @RG Read group.
- ...

@SQ SN:1 LN:307041717
@SQ SN:2 LN:244442276

SAM section 2: the alignment section

In SAM, each alignment line has 11 mandatory fields and a variable number of optional fields.

```
EAS600_70:5:1:3215:930 99      REF     2767401      60      100M    =      2767797      498
NTGATATTAACTTGTCCAATATGATCAAATAGCATTACCCCCCCTCACAAACGTCCTGCATAGGGAACACGTTTCCCTGTGCACCCACGACTAAATT
!++*+87777@0000000000000000<::<<99989::32222298&)--28888589179@000# ##########
NM:i:1      MD:Z:0A99      AS:i:99      XS:i:0
```

No.	Name	Description	
1	QNAME	Query NAME of the read or the read pair	EAS600_70:5:1:3215:930
2	FLAG	Bitwise FLAG (pairing, strand, mate strand, etc.)	99
3	RNAME	Reference sequence NAME	REF
4	POS 1	1-Based leftmost POSition of clipped alignment	2767401
5	MAPQ	MAPping Quality (Phred-scaled)	60
6	CIGAR	Extended CIGAR string (operations: MIDNSHP)	100M
7	MRNM	Mate Reference NaMe ('=' if same as RNAME)	=
8	MPOS	1-Based leftmost Mate POSition	2767797
9	ISIZE	Inferred Insert SIZE	498
10	SEQ	Query SEQuence on the same strand as the reference	NTGATA...
11	QUAL	Query QUALity (ASCII-33=Phred base quality)	!++*+8...

optional field

FLAG (for single-end or paired-end reads)

Bitwise flag values

Hexadecimal Value	Decimal Value	Description
0x0001	1	The read is paired in sequencing, no matter whether it is mapped in a pair
0x0002	2	The read is mapped in a proper pair (depends on the protocol, normally inferred during alignment)
0x0004	4	The query sequence itself is unmapped
0x0008	8	The second read is unmapped
0x0010	16	The read is reverse complemented
0x0020	32	The second read is reversed
0x0040	64	The read is the first read in a pair
0x0080	128	The read is the second read in a pair
0x0100	256	The alignment is not primary (a read having split hits may have multiple primary alignment records)
0x0200	512	The read fails platform/vendor quality checks
0x0400	1024	The read is either a PCR duplicate or an optical duplicate

16 Reverse complemented

$1 + 4 + 128 = 133$ paired; segment unmapped;

Problem: explain these decimal FLAG numbers

73

81

192

Hexadecimal Value	Decimal Value	Description
0x0001	1	The read is paired in sequencing, no matter whether it is mapped in a pair
0x0002	2	The read is mapped in a proper pair (depends on the protocol, normally inferred during alignment)
0x0004	4	The query sequence itself is unmapped
0x0008	8	The second read is unmapped
0x0010	16	The read is reverse complemented
0x0020	32	The second read is reversed
0x0040	64	The read is the first read in a pair
0x0080	128	The read is the second read in a pair
0x0100	256	The alignment is not primary (a read having split hits may have multiple primary alignment records)
0x0200	512	The read fails platform/vendor quality checks
0x0400	1024	The read is either a PCR duplicate or an optical duplicate

MAPQ: Mapping quality

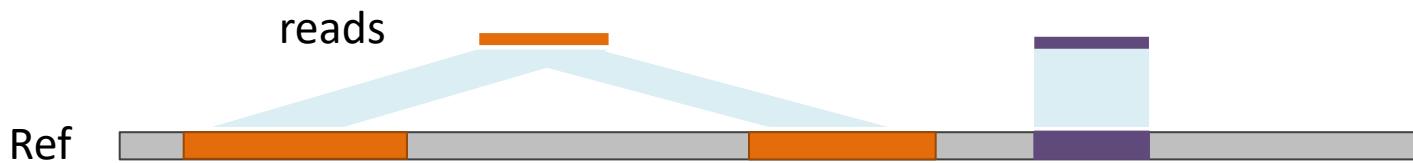
- $\text{MAPQ} = -10 \log_{10} \text{Prob}\{\text{error mapping}\}$, rounded to the nearest integer.
 - 0: $\text{Prob}(\text{error mapping}) = 10^{0/(-10)} = 1$
 - 10: $\text{Prob}(\text{error mapping}) = 10^{10/(-10)} = 0.1$
 - 30: $\text{Prob}(\text{error mapping}) = 10^{30/(-10)} = 0.001$
 - 40: $\text{Prob}(\text{error mapping}) = 10^{40/(-10)} = 0.0001$

Factors influencing mapping quality

- Alignment quality (matches, mismatches, gaps)

s : CTGTTGCT
| | | | |
t : ATGCTGCA

- Mapping ambiguity



Mapping quality ≠ Alignment quality

A read with a high mapping score

- A read alignment with a mapping quality 30 or above usually implies:

The read is mapped at the location with **a high alignment score** and the read is not mapped to anywhere else or alignments at other locations are not as good as this alignment.

CIGAR

Example 1: 100M

Example 2: 24M**188N83M**

Example 3: 80M**1I20M2D60M**

Example 4: 98M**2S**

Operation	Description
M	alignment match (can be a sequence match or mismatch)
I	insertion to the reference
D	deletion from the reference
N	skipped region from the reference
S	soft clipping (clipped sequences present in SEQ)
H	hard clipping (clipped sequences NOT present in SEQ)
P	padding (silent deletion from padded reference)
=	sequence match
X	sequence mismatch

For mRNA-to-genome alignment, an N operation represents an intron.

P, =, and X are rarely used.

Binary Alignment/Map format (BAM)

- To improve the performance, a companion format Binary Alignment/Map (BAM) was designed.
- BAM is the binary representation of SAM and keeps exactly the same information as SAM.

Table. File size of two example files

File type	Storage usage
SAM	313 M
BAM	97 M

SAMtools

- SAMtools is a popular open-source software package for sequence alignment manipulation. It is a reliable tool for almost all bioinformaticists who work on NGS data.

```
-- indexing          -- stats
faidx      index/extract FASTA   bedcov     read depth per BED region
index       index alignment    depth      compute the depth
-- editing           flagstat    simple stats
calmd       recalculate ...   idxstats   BAM index stats
fixmate     fix mate information phase      phase heterozygotes
reheader    replace BAM header stats     generate stats (former bamcheck)
rmdup      remove PCR duplicates
targetcut   cut fosmid regions
-- file operations
bamshuf    shuffle and group ...
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
-- viewing
flags      explain BAM flags
tview      text alignment viewer
view       SAM<->BAM<->CRAM conversion
```

Conversion between SAM and BAM

For efficient computation and storage, SAM is often converted to BAM. Sorting and indexing the alignment facilitate the following analyzing procedures.

- **converting:**

```
samtools view aln.sam -o aln.bam
```

- **sorting:**

```
samtools sort aln.bam -o alnsort.bam
```

- **indexing:**

```
samtools index alnsort.bam
```

Visualization

```
 samtools tview alnsort.bam ref.fasta
```

Integrative Genomics Viewer (IGV)

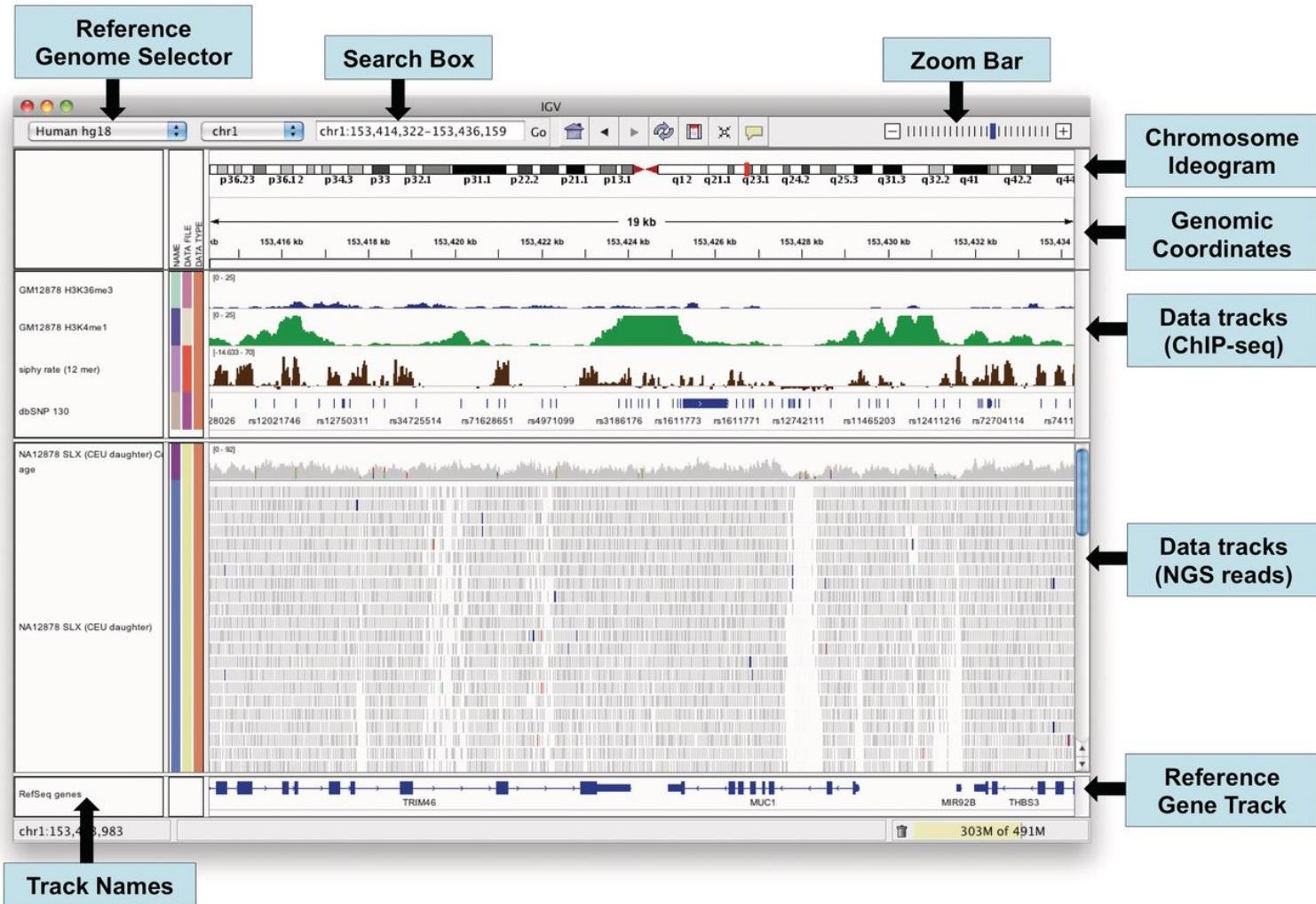
- The Integrative Genomics Viewer (IGV) is a high-performance visualization tool for interactive exploration of large, integrated genomic datasets.
- IGV focus on the integrative nature of genomic studies, with support for both array-based and next-generation sequencing data, and the integration of phenotypic data.

James T. Robinson, et al., Nature Biotechnology 29:24–26 (2011)

Helga Thorvaldsdóttir, et al., Briefings in Bioinformatics 14:178-192 (2013)

www.broadinstitute.org/igv

The IGV application window.



Data for IGV

- Reference genomes (build-in or external references)
- Data format:
 1. Non-indexed formats: GFF, BED and WIG.
 2. Indexed formats: BAM (sequence alignments)

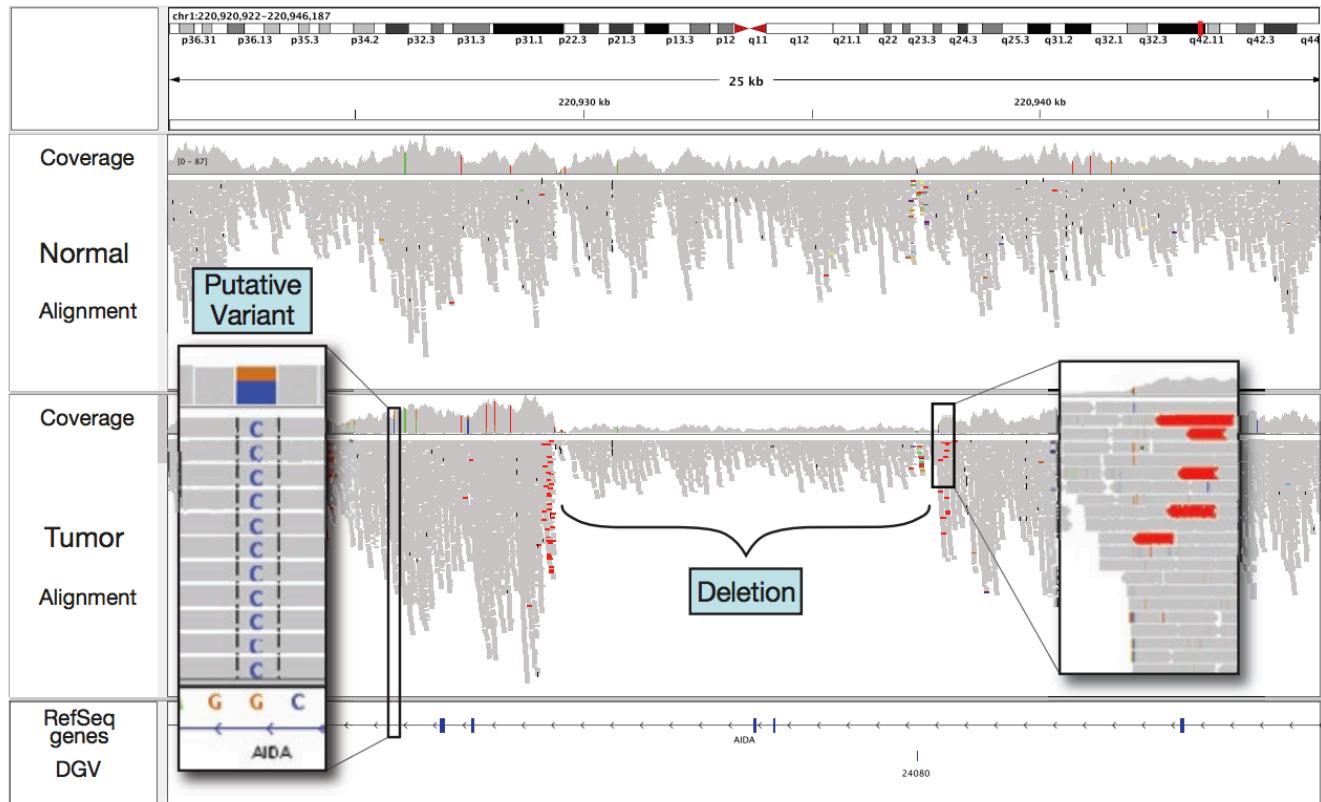
For example:

A reference genome: .fasta

A genome annotation file: .gff3 (optional)

An alignment file: .bam (plus .bai index file)

IGV example



1. Loci with a large percentage of mismatches relative to the reference are flagged in the coverage plot as color-coded bars.
2. Alignments with unexpected inferred insert sizes are indicated by color.
3. Evidence for a ~10-kb deletion in the tumor sample not present in the normal.

Review

- One of short-read alignment algorithms – BWT
- An aligner - BWA
- Alignment standard format – SAM and BAM
- A tool to handle alignments - SAMtools
- An alignment visualization tool - IGV