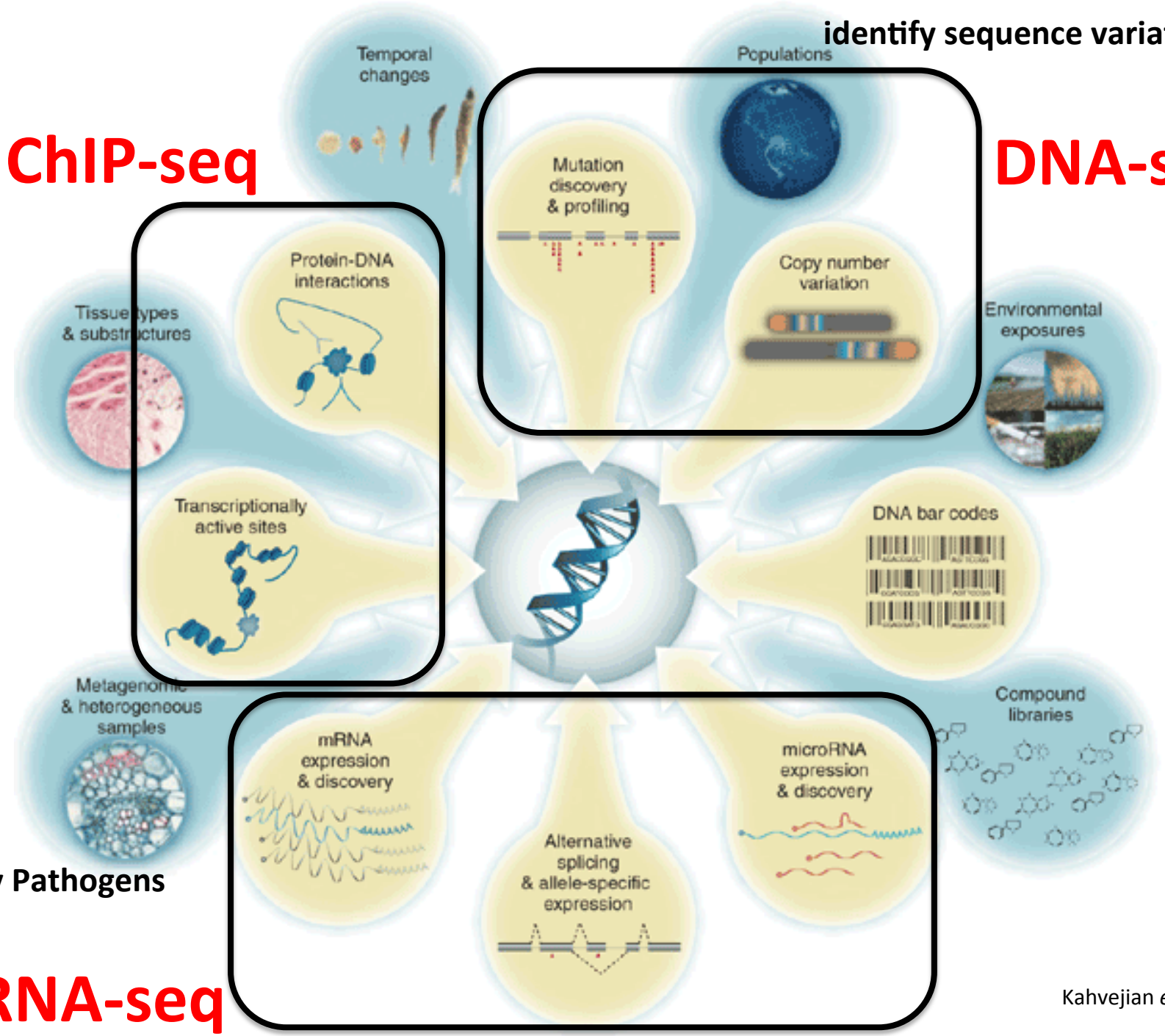
Next-generation sequencing

Lecture 9

identify sequence variations



Identify Pathogens

Kahvejian *et al*, 2008

Applications of RNA-seq

- Study of mRNA gene expression profile and discovery of differentially expressed genes.
- Study of microRNA profiling.
- Discovery of new transcripts, such as long non-coding RNA and circular RNA.

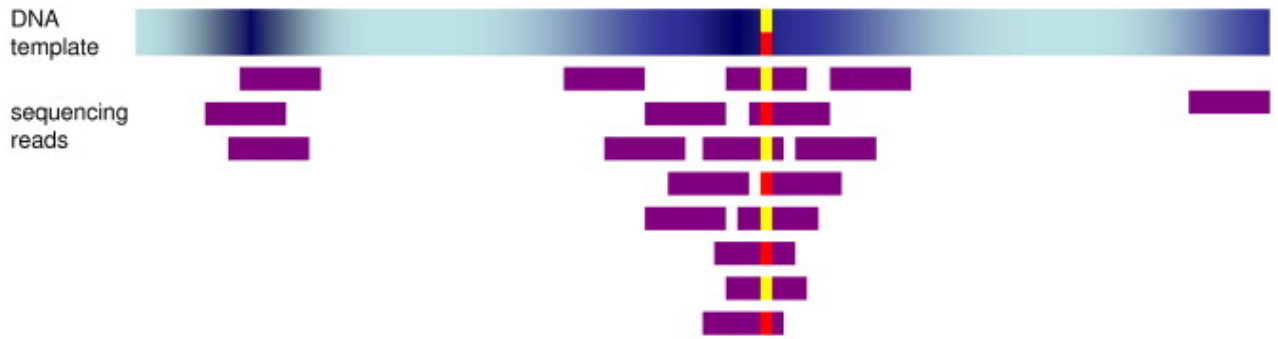
Short Read Alignment

- short reads have to be aligned (mapped) to a reference sequence.

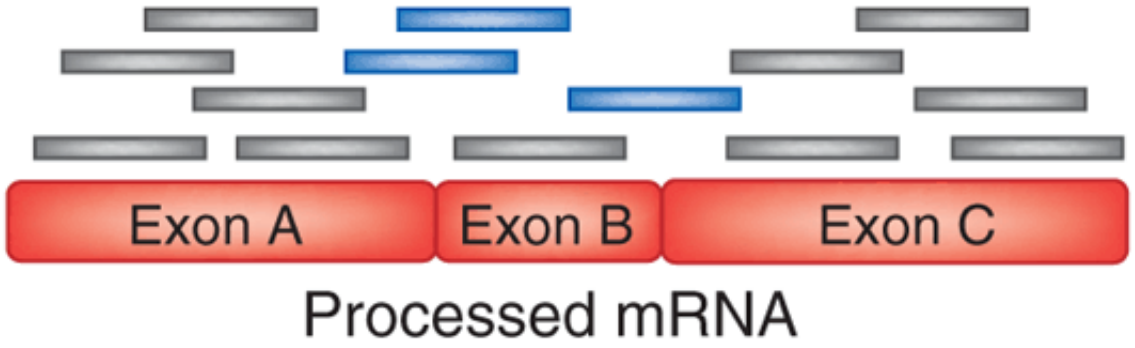
  To genomes
  To transcriptomes

Short Read Applications

- To genomes (DNA-seq)



- To transcriptomes (RNA-seq, ChIP-seq, MethyI-seq)



Measure significant peaks

Short Read Alignment

```
GCACTTCACAAATTAATGACCATGAGCTCGTTTTTGATAAACTCCAACTACATCGAGCCC
            ||||||||||||  ||||||||||
            ACCATGAGCTCGATTTTGATAAA
```

GOAL: to efficiently find the true location of each read from a potentially large quantity of reference data while distinguishing between technical sequencing errors and true genetic variation within the sample.

1. Efficient
2. True location
3. Distinguishing between technical sequencing errors and true genetic variation

Traditional methods would not work well

- BLAST: a sequence similarity search algorithm (fast for searching a single sequence)
  - Not specifically designed for mapping millions of query sequences
  - Take very long time
    - Can take 2 days to map half million sequences to 70MB reference genome
    - Very memory expensive (indexing the whole genome)

Performance Requirements

- **Aligning tens of millions of sequences requires:**
  - Ultra fast search algorithms (100-1000x faster than BLAST)
  - Small memory usage with economic data structures and containers for programming
- **Alignment requirements**
  - The requirements for short-read mapping applications are very different from traditional sequence database search approaches for ortholog identification.
  - Most short-read alignment algorithms will not work for longer sequences! However, most of them are more sensitive for short-reads than BLAST, because they lack its word size limitation.
  - Only best hits with almost perfect alignments are required. Lower scoring alternative hits (more mismatches) are less interesting.
  - Allow 1-2 mismatches and only sometimes very short gaps.

Alignment

- “Mapping the vast quantities of short sequence fragments produced by next-generation sequencing platforms is a challenge. What programs are available and how do they work?” (Trapnell et al. Nature Biotechnology, 2009.)

Mapping Algorithms

- **Hash Table** (Lookup table)
  - FAST, but requires perfect matches.
- Array Scanning
  - Can handle mismatches, but not gaps.
- **Dynamic Programming** (Smith-Waterman)
  - Indels
  - Mathematically optimal solution
  - Slow (most programs use Hash Mapping as a pre-filter)
- **Merge Sorting**

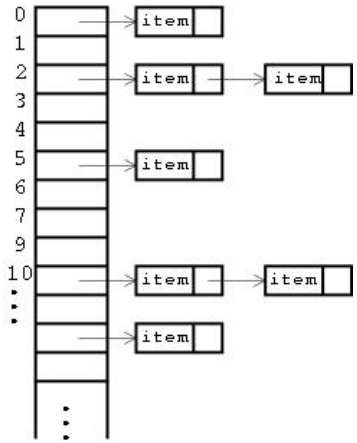
  - i.e. Slider [Malhis et al 2009]
- **Indexing method**, i.e. Burrows-Wheeler Transform (BW Transform)
  - FAST. [O(m + N)] (without mismatch/gap)
  - Memory efficient.
  - But for gaps/mismatches, it lacks sensitivity

Dynamic programming

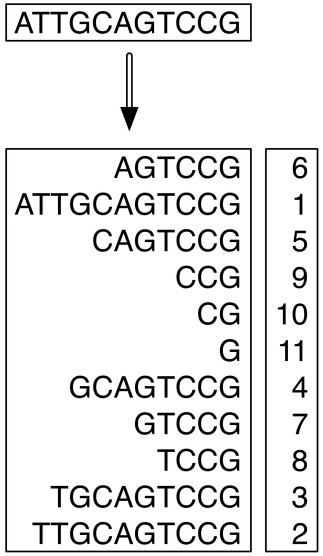
- A dynamic programming can be used to find the local alignments between a text T and a pattern P in O(|T|,|P|) time, but need memory about O(|T|x|P|) .
- It is used for two sequence comparison developed by Smith and Waterman.
- The genome is too big for this approach.

Indexing

- Lengths of genome sequences and numbers of reads are too large for direct approaches
- *Indexing* is required



Hash tables, such as spaced seed



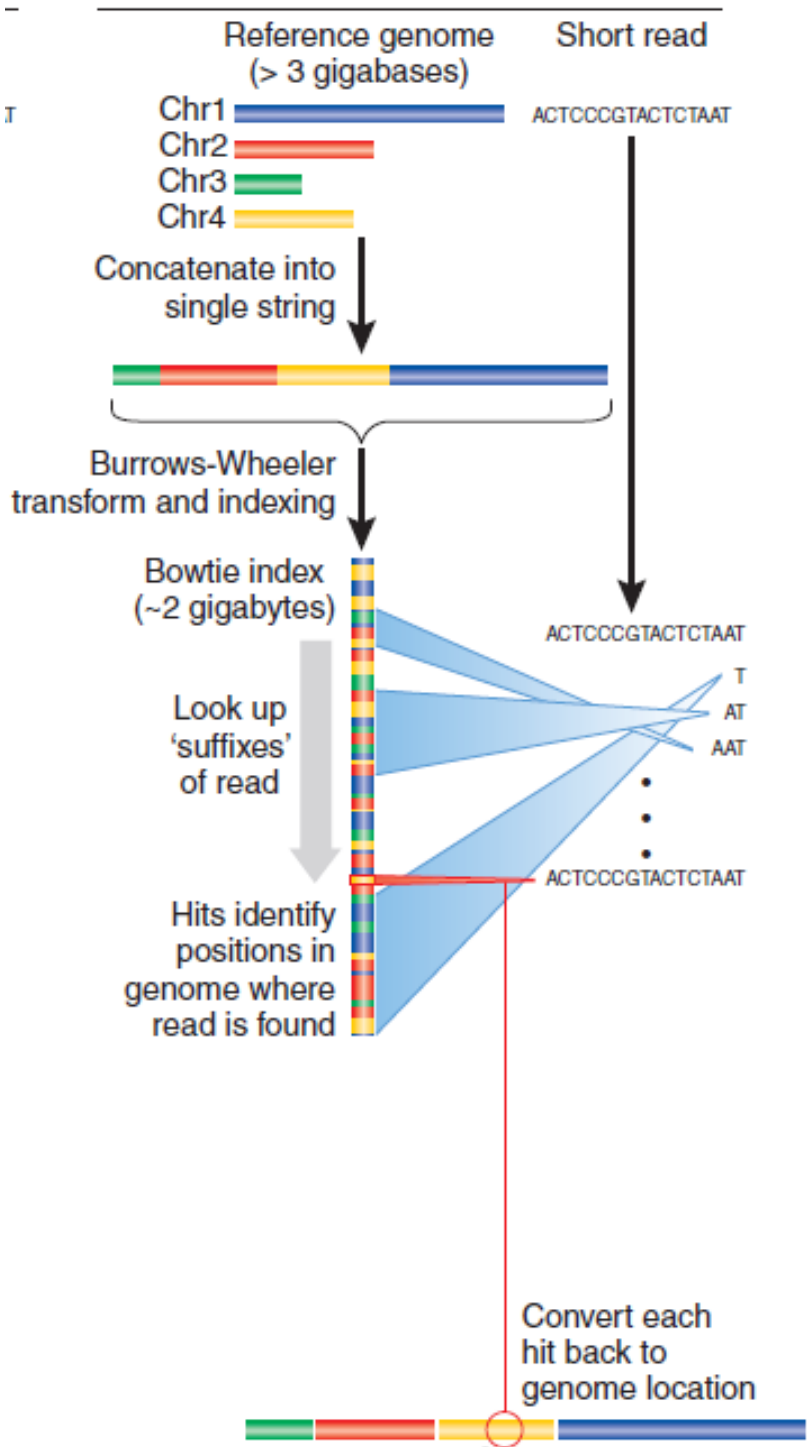
Suffix array or Burrows-wheeler Transform

Short-Read Alignment Tools with indexing

- **Indexing Reads with Hash Tables**
  - ZOOM: uses spaced seeds algorithm [Lin et al 2008]
  - RMAP: simpler spaced seeds algorithm [Smith et al 2008]
  - SHRiMP: employs a combination of spaced seeds and the Smith-Waterman
  - MAQ [Li et al 2008b]
  - Eland (commercial Solexa Pipeline)
- **Indexing Reference with Hash Tables**
  - SOAPv1 [Li et al 2008]
- **Indexing Reference with Sux Array/Burrows-Wheeler**
  - Bowtie [Langmead et al 2009]
  - BWA
  - SOAPv2

MAQ, indexing with Hashing tables

- **Algorithm [Li et al 2008b]**
  - Uses a hashing technique that guarantees to found alignments with up to two mismatches in the first 28 bp of the reads.
  - It indexes the read sequences in six hash tables and scans the reference genome sequence for seed hits that are subsequently extended and scored.
  - The commercial Eland alignment program uses a very similar approach.
- **Performance**
  - Slower than Bowtie and SOAP.
- **Features**
  - Versatile pipeline for SNP detection.
  - Can report all hits for queries with multiple ones.
  - Performs on single reads only ungapped alignments. Gaps only possible for paired end reads by applying Smith-Waterman algorithm on small candidate set.



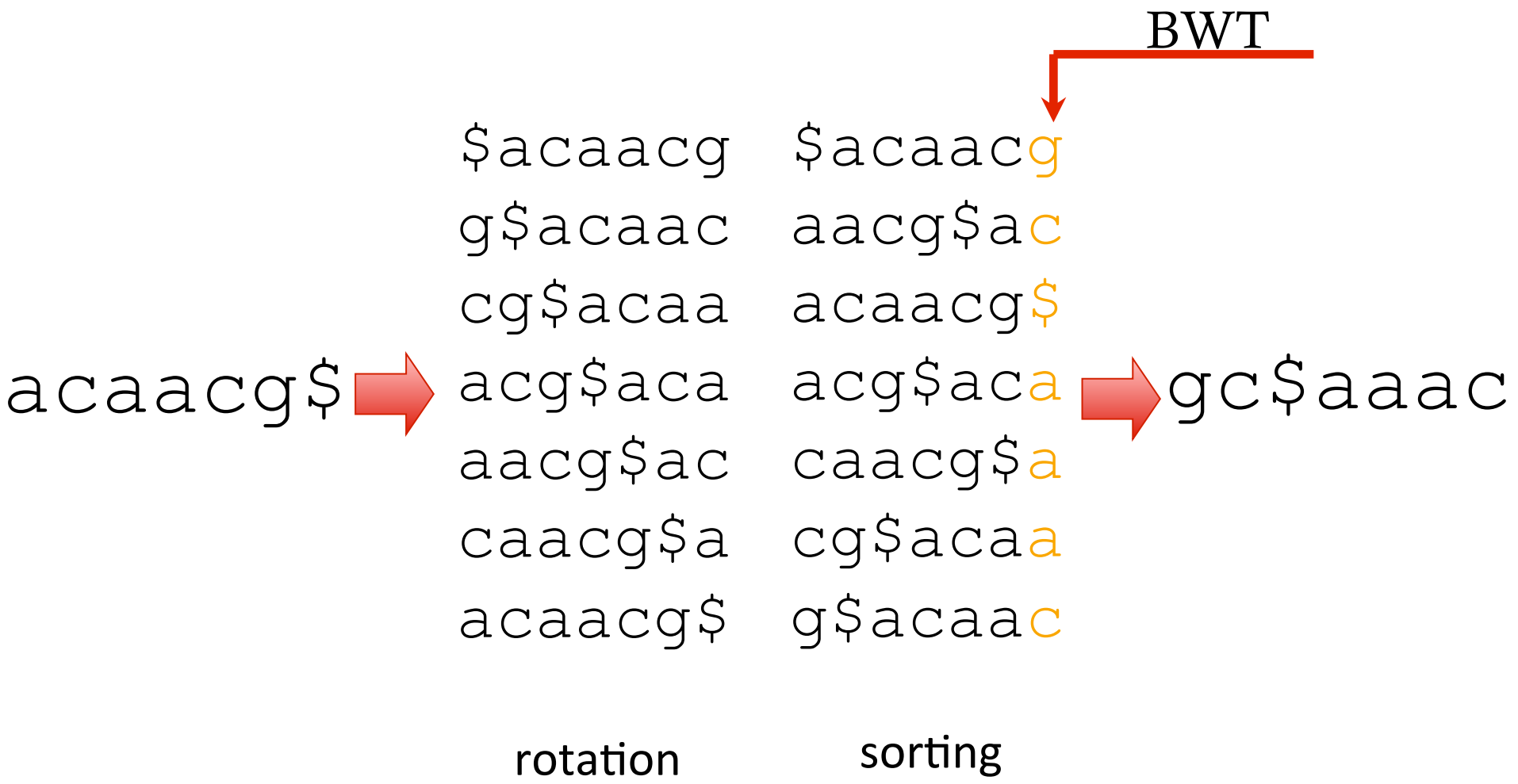
Bowtie, Burrows-Wheeler Transformation (BWT)

- Store entire reference genome.
- Align tag base by base from the end.
- When tag is traversed, all active locations are reported.
- If no match is found, then back up and try a substitution.

Burrows-Wheeler Transform (BWT)

- BWT is the sequence of the last characters of sorted list of rotations of the string (notice that they are sorted in the same order as the prefixes).

Burrows-Wheeler Transform (BWT)



Key observation

$a^1c^1a^2a^3c^2g^1\$^1$

“last first (LF) mapping”

The *i*-th occurrence of character X in the last column corresponds to the same text character as the *i*-th occurrence of X in the first column.

$^1\$acaacg^1$

$^2aacg\$ac^1$

$^1acaacg\$^1$

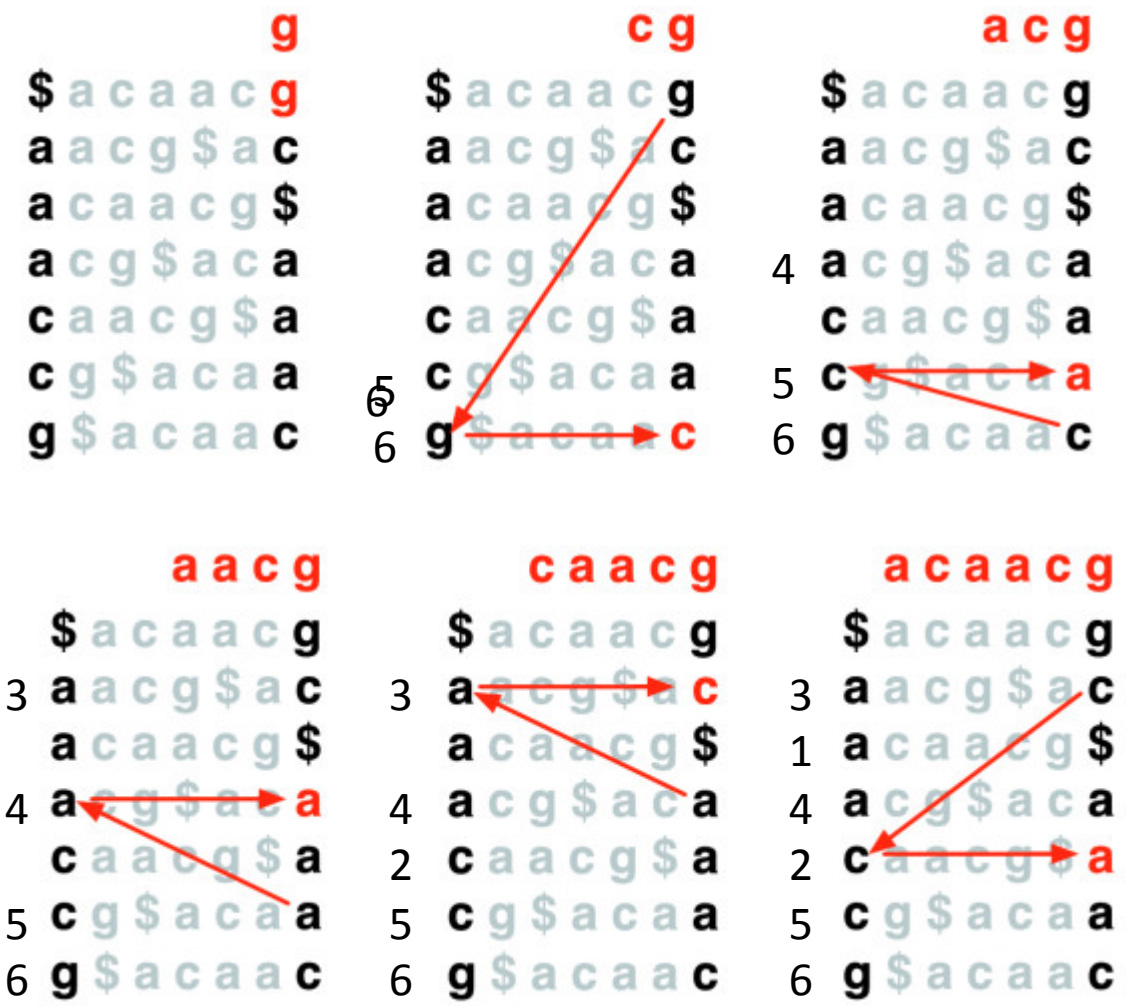
$^3acg\$aca^2$

$^1caacg\$a^1$

$^2cg\$acaa^3$

$^1g\$acaac^2$

Recover text



$a^1c^1a^2a^3c^2g^1\$^1$

$^1\$acaacg^1$

$^2aacg\$ac^1$

$^1acaacg\$^1$

$^3acg\$aca^2$

$^1caacg\$a^1$

$^2cg\$acaa^3$

$^1g\$acaac^2$

One issue

- For BWT to search perfect matches, it is simple and fast
- For imperfect mapping, such as mismatches or insertion/deletion, BWT is very slow.
- Bowtie uses greedy searching, and does not always give the optimal alignment.
- TopHat and Cufflinks based on bowtie are even worse.

Main advantage of BWT against suffix array

- BWT very compact:
  - Approximately 1/4 byte per base (2 bits)
  - As large as the original text, plus a few "extras"
  - Can fit onto a standard computer with 2GB of memory
  - BWT needs less memory than suffix array
- For example, human genome, $m = 3 \times 10^9$:
  - Suffix array: $m\log_2(m)$ bits = 4m bytes = 12GB
  - BWT: m/4 bytes plus extras = 1 - 2 GB
- Linear-time search algorithm
  - proportional to length of query for exact matches

Comparison

Indexing Reads with Hash Tables

- Requires ~50Gb of memory.
- Runs 30-fold slower.
- Is much simpler to program.

MAQ

Burrows-Wheeler

- Requires <2Gb of memory.
- Runs 30-fold faster.
- Is much more complicated to program.

Bowtie, BWA etc.

Short-read mapping software

Software	Technique	Developer	License
SOAP	Hashing refs	BGI	Academic
Maq	Hashing reads	Sanger (Li, Heng)	GNUPL
Bowtie	BWT	Salzberg/UMD	GNUPL
BWA	BWT	Sanger (Li, Heng)	GNUPL
SOAP2	BWT & hashing	BGI	Academic

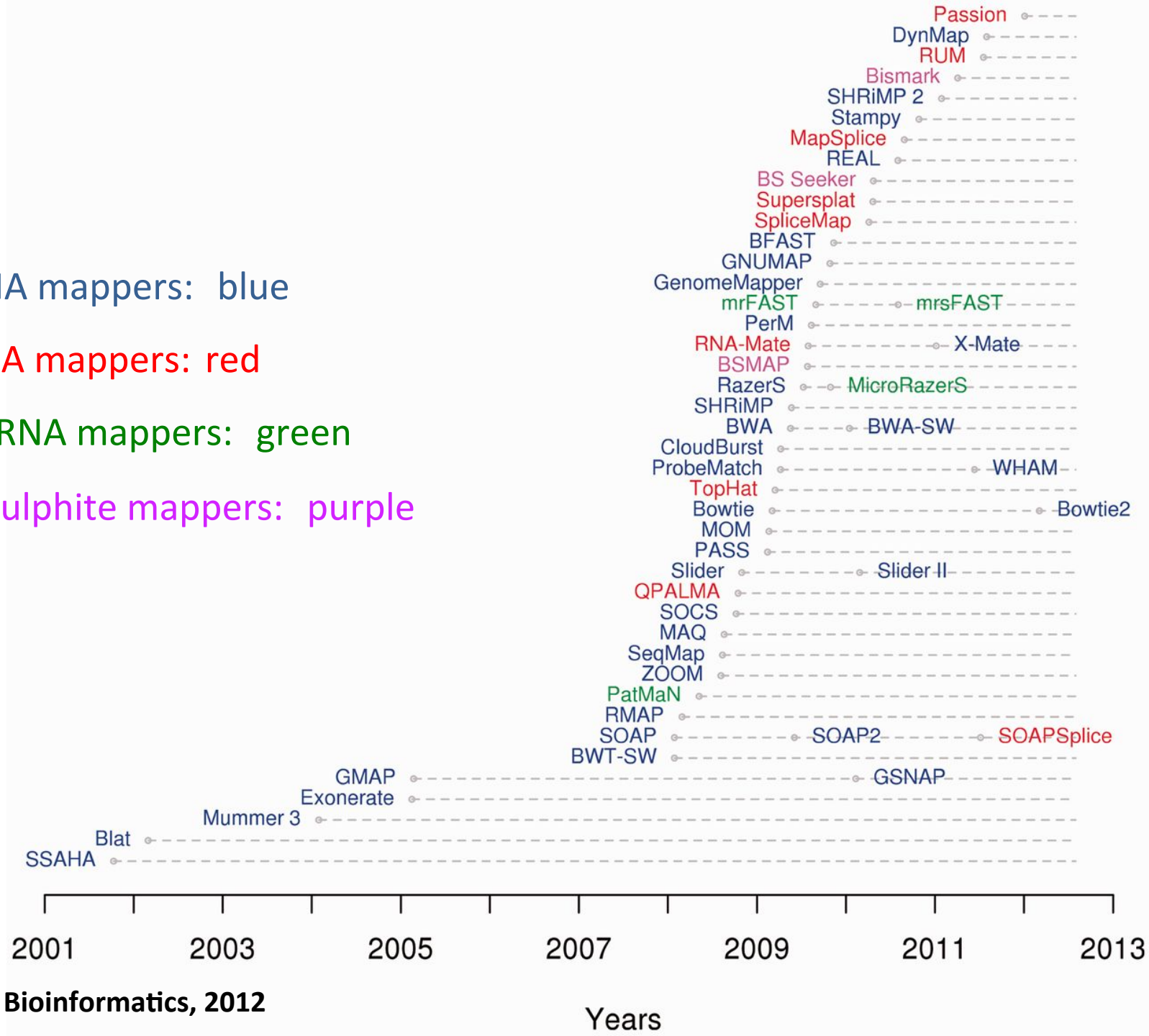
http://www.oxfordjournals.org/our_journals/bioinformatics/nextgenerationsequencing.html

DNA mappers: blue

RNA mappers: red

miRNA mappers: green

Bisulphite mappers: purple



Fonseca, Bioinformatics, 2012

Mapper	Data	Seq.Plat.	Input	Output	Avail.	Version	Cit.	$\frac{Citations}{Years}$	Reference
BFAST	DNA	I,So,4, Hel	(C)FAST(A/Q)	SAM TSV	OS	0.7.0	94	37.11	Homer *et al.* (2009)
Bismark	Bisulphite	I	FASTA/Q	SAM	OS	0.7.3	7	6.21	Krueger and Andrews (2011)
BLAT	DNA	N	FASTA	TSV BLAST	OS	34	2844	275.67	Kent (2002)
Bowtie	DNA	I,So,4,Sa,P	(C)FAST(A/Q)	SAM TSV	OS	0.12.7	1168	363.42	Langmead *et al.* (2009)
Bowtie2	DNA	I,4,Ion	FASTA/Q	SAM TSV	OS	2.0beta5		0.00	Langmead and Salzberg (2012)
BS Seeker	Bisulphite	I	FASTA/Q	SAM	OS		19	9.26	Chen *et al.* (2010)
BSMAP	Bisulphite	I	FASTA/Q	SAM TSV	OS	2.43	31	11.06	Xi and Li (2009)
BWA	DNA	I,So,4,Sa,P	FASTA/Q	SAM	OS	0.6.2	738	224.20	Li and Durbin (2009)
BWA-SW	DNA	I,So,4,Sa,P	FASTA/Q	SAM	OS	0.6.2	160	67.69	Li and Durbin (2010)
BWT-SW	DNA	N	FASTA	TSV	OS	20070916	45	10.42	Lam *et al.* (2008)
CloudBurst	DNA	N	FASTA	TSV	OS	1.1	146	46.97	Schatz (2009)
DynMap	DNA	N	FASTA	TSV	OS	0.0.20		0.00	Flouri *et al.* (2011)
ELAND	DNA	I	FASTA	TSV	Com	2	7	1.09	Unpublished[a]
Exonerate	DNA	N	FASTA	TSV	OS	2.2	255	34.69	Slater and Birney (2005)
GEM	DNA	I, So	FASTA/Q	SAM, counts	Bin	1.x	4	1.35	Unpublished[b]
GenomeMapper	DNA	I	FASTA/Q	BED TSV	OS	0.4.3	31	11.66	Schneeberger *et al.* (2009)
GMAP	DNA	I,4,Sa,Hel,Ion,P	FASTA/Q	SAM, GFF	OS	2012-04-27	217	29.52	Wu and Watanabe (2005)
GNUMAP	DNA	I	FASTA/Q Illumina	SAM TSV	OS	3.0.2	15	5.73	Clement *et al.* (2010)
GSNAP	DNA	I,4,Sa,Hel,Ion,P	FASTA/Q	SAM	OS	2012-04-27	72	31.61	Wu and Nacu (2010)
MapReads	DNA	So	FASTA/Q	TSV	OS	2.4.1		0.00	Unpublished[c]
MapSplice	RNA	I	FASTA/Q	SAM BED	OS	1.15.2	50	28.17	Wang *et al.* (2010)
MAQ	DNA	I,So	(C)FAST(A/Q)	TSV	OS	0.7.1	957	251.66	Li *et al.* (2008a)
MicroRazerS	miRNA	N	FASTA	SAM TSV	OS	0.1	7	2.75	Emde *et al.* (2010)
MOM	DNA	I,4	FASTA	TSV	Bin	0.6	18	5.55	Eaves and Gao (2009)
MOSAIK	DNA	I,So,4,Sa,Hel,Ion,P	(C)FAST(A/Q)	BAM	OS	2.1	4	1.18	Unpublished[d]
mrFAST	miRNA	I	FASTA/Q	SAM	OS	2.1.0.4	158	58.34	Alkan *et al.* (2009)
mrsFAST	miRNA	I,So	FASTA/Q	SAM	OS	2.3.0	32	18.03	Hach *et al.* (2010)
Mummer 3	DNA	N	FASTA	TSV	OS	3.23	683	81.58	Kurtz *et al.* (2004)
Novoalign	DNA	I,So,4,Ion,P	(C)FAST(A/Q) Illumina	SAM TSV	Bin	V2.08.01	137	34.49	Unpublished[e]
PASS	DNA	I,So,4	(C)FAST(A/Q)	SAM GFF3 BLAST	Bin	1.62	45	13.67	Campagna *et al.* (2009)
Passion	RNA	I,4,Sa,P	FASTA/Q	BED	OS	1.2.0		0.00	Zhang *et al.* (2012)
PatMaN	miRNA	N	FASTA	TSV	OS	1.2.2	38	9.36	Prüfer *et al.* (2008)
PerM	DNA	I,So	(C)FAST(A/Q)	SAM TSV	OS	0.4.0	30	10.88	Chen *et al.* (2009)
ProbeMatch	DNA	I,4,Sa	FASTA	ELAND	OS		6	1.92	Kim *et al.* (2009)

(continued)

Performance: DNA

Aligner	Time	Memory	Read #
BFAST	39	21.4	561348
BLAT	93	3.8	950220
Bowtie	169	5	798566
BWA	97	7.6	928093
BSMAP	25	8.3	802430
SOAP2	82	5.3	798565
MOSAIK	22	15.6	267173

For DNA (100bp) reads, 1 Million reads, time is in minutes, memory is in GB.

Performance: miRNA

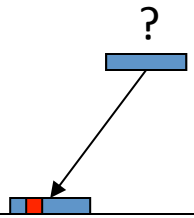
Aligner	Time	Memory	Read #
Bowtie	119	5	983951
BWA	91	7.4	996470
GSNAP	19	7.6	966802
MicroRazerS	49	1.6	979464
mrFAST	48	0.8	982049
PASS	28	15.8	999989
PERM	57	13.4	982545

For miRNA (20bp) reads, 1 Million reads, time is in minutes, memory is in GB.

Issues about Alignment

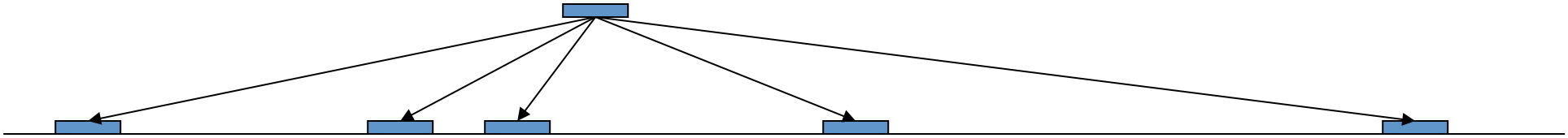
- A reference genome is used when available.
- Typically, a maximum number of mismatches (1 or 2) are allowed when aligning reads.
- There are many challenges, such as dealing with alternative splicing and reads that match multiple places in the genome.
- Data in its rawest form is huge and requires substantial computing power to manage.

Mismatch



- An observed short read may not exactly match any position in the reference genome.
- Such mismatches may represent a bad read-out.
- The user can specify the maximum number of mismatches, or a phred-style quality score threshold.
- As the number of allowed mismatches goes up, the number of mapped tags increases, but so does the number of incorrectly mapped tags.

Multiple mapping



- A single short read may occur more than one time in the reference genome.
- The user may choose to ignore tags that appear many times.
- As the number of aligned positions gets large, you get more data, but also more noise in the data.

A short summary

- One can specify how many mismatches are to be tolerated.
- This can also be quantified by accounting for quality scores.
- A typical criterion might be 1-2 mismatches for 36bp reads.
- From the raw sequences, ~50-80% of the reads are typically aligned to the genome
  - sequencing errors
  - multiple matches in the genome
  - deviations from the reference genome (SNPs, insertions, etc)
  - problems with the aligner
- This % of mapped reads is a good measure of data quality.

Output: SAM format

A SAM file consists of two parts:

- Header
  - contains meta data (source of the reads, reference genome, aligner, etc.)
  - All header lines start with "@".
  - Header fields have standardized two-letter codes for easy parsing of the information.
  - Most current tools omit and/or ignore the header.
- Alignment section
  - A tab-separated table with at least 11 columns
  - Each line describes one alignment

http://samtools.sourceforge.net/SAM1.pdf

SAM: multiple alignments and paired-end reads

- Each line represents one *alignments.*
- Multiple alternative alignments for the same read take multiple lines. Only the read ID allows to group them.
- Paired-end alignments take two lines.

- All these reads are not necessarily in adjacent lines.

A SAM file

[...]

HWI-EAS225_309MTAAXX:5:1:689:1485 0 XIII 863564 25 36M * 0 0 GAAATATATACGTTTTTATCTATGTTACGTTATATA CCCCCCCCCCCCCCCCCCCCCCCC4CCCB4CA?AAA< NM:i:0 X0:i:1 MD:Z:36

HWI-EAS225_309MTAAXX:5:1:689:1485 16 XIII 863766 25 36M * 0 0 CTACAATTTTGCACATCAAAAAAGACCTCCAACTAC =8A=AA784A9AA5AAAAAAAAAAA=AAAAAAAAAA NM:i:0 X0:i:1 MD:Z:36

HWI-EAS225_309MTAAXX:5:1:394:1171 0 XII 525532 25 36M * 0 0 GTTTACGGCGTTGCAAGAGGCCTACACGGGCTCATT CCCCCCCCCCCCCCCCCCCCCC?CCACCACA7?<???NM:i:0 X0:i:1 MD:Z:36

HWI-EAS225_309MTAAXX:5:1:394:1171 16 XII 525689 25 36M * 0 0 GCTGTTATTTCTCCACAGTCTGGCAAAAAAAAGAAA 7AAAAAA?AA<AA? AAAAA5AAA<AAAAAAAAAAA NM:i:0 X0:i:1 MD:Z:36

SAM format: alignment section

The columns are:

- ✓ QNAME: ID of the read ("query")
- ✓ FLAG: alignment flags
- ✓ RNAME: ID of the reference (typically: chromosome name)
- ✓ POS: Position in reference (1-based, left side)
- ✓ MAPQ: Mapping quality (as Phred score)
- ✓ CIGAR: Alignment description (gaps etc.) in CIGAR format
- ✓ MRNM: Mate reference sequence name [for paired end data]
- ✓ MPOS: Mate position [for paired end data]
- ✓ ISIZE: inferred insert size [for paired end data]
- ✓ SEQ: sequence of the read
- ✓ QUAL: quality string of the read
- ✓ extra fields

SAM alignment section

Chr1-239951 0 Chr1 1211705 255 50M * 0 0
ACCAATTCTCCTTGTGTTGTCACTGAAATTGTCTCCATGGAATTTTATTT
B@BC,CE;*BCEF@=EDA,C+AB?,F>,B53@((2(?33?;D992)5+55 XA:i:1
MD:Z:32T17 NM:i:1

1. Read Name
2. SAM flag (0=same orientation, 16=complimentary)
3. chromosome (if read is has no alignment, there will be a "*" here)
4. position (1-based index, "left end of read")
5. MAPQ (mapping quality)
6. CIGAR string (describes the position of insertions/deletions/matches in the alignment, encodes splice junctions, for example)
7. Name of mate (mate pair information for paired-end sequencing)
8. Position of mate (mate pair information)
9. Template length (always zero)
10. Read Sequence
11. Read Quality (Phred scores)
12. extra fields Program specific Flags (i.e. HI:i:0, MD:Z:66G6, etc.)

Li et al, Bioformatics, 2009

SAM flag field

- FLAG field: a number, read as binary

bit	hex	decimal	
0	00 01	1	read is a paired-end read
1	00 02	2	read pair is properly matched
2	00 04	4	read has not been mapped
3	00 08	8	mate has not been mapped
4	00 10	16	read has been mapped to “-” strand
5	00 20	32	mate has been mapped to “-” strand
6	00 40	64	read is the first read in a pair
7	00 80	128	read is the second read in a pair
8	01 00	256	alignment is secondary
9	02 00	512	read did had not passed quality check
10	04 00	1024	read is a PCR or optical duplicate

SAM: CIGAR strings

- Alignments contain gaps (e.g., in case of an indel, or, in RNA-Seq, when a read straddles an intron).
- Then, the CIGAR string gives details.
- Example: “M10 I4 M4 D3 M12” means
  - ✓ the first 10 bases of the read map (“M10”) normally (not necessarily perfectly)
  - ✓ then, 4 bases are inserted (“I4”), i.e., missing in the reference
  - ✓ then, after another 4 mapped bases (“M4”), 3 bases are deleted (“D3”), i.e., skipped in the query.
  - ✓ Finally, the last 12 bases match normally.
- There are further codes (N, S, H, P), which are rarely used

SAM: optional fields

Extra fields:

- Always triples of the format TAG : VTYPE : VALUE
- some important tag types:
  - NH: number of reported alignments
  - NM: number of mismatches
  - MD: positions of mismatches

Bowtie specified part

NM:i:<N> Aligned read has an edit distance of <N>. (N mismatches)

CM:i:<N> Aligned read has an edit distance of <N> in colorspace. This field is present in addition to the NM field in -C/--color mode, but is omitted otherwise.

MD:Z:<S> For aligned reads, <S> is a string representation of the mismatched reference bases in the alignment. See SAM format specification for details. For colorspace alignments, <S> describes the decoded nucleotide alignment, not the colorspace alignment.

XA:i:<N> Aligned read belongs to stratum <N>.

XM:i:<N> For a read with no reported alignments, <N> is 0 if the read had no alignments. If -m was specified and the read's alignments were supressed because the -m ceiling was exceeded, <N> equals the -m ceiling + 1, to indicate that there were at least that many valid alignments (but all were suppressed). In -M mode, if the alignment was randomly selected because the -M ceiling was exceeded, <N> equals the -M ceiling + 1, to indicate that there were at least that many valid alignments (of which one was reported at random).

A useful specific tag

- NH:i:<N> Number of reported alignments that contains the query in the current record

- Bowtie does not have this output. How can we find the uniquely aligned reads?
  - Search whole sam file with a read ID. If this ID occurs only once time, it is uniquely aligned.

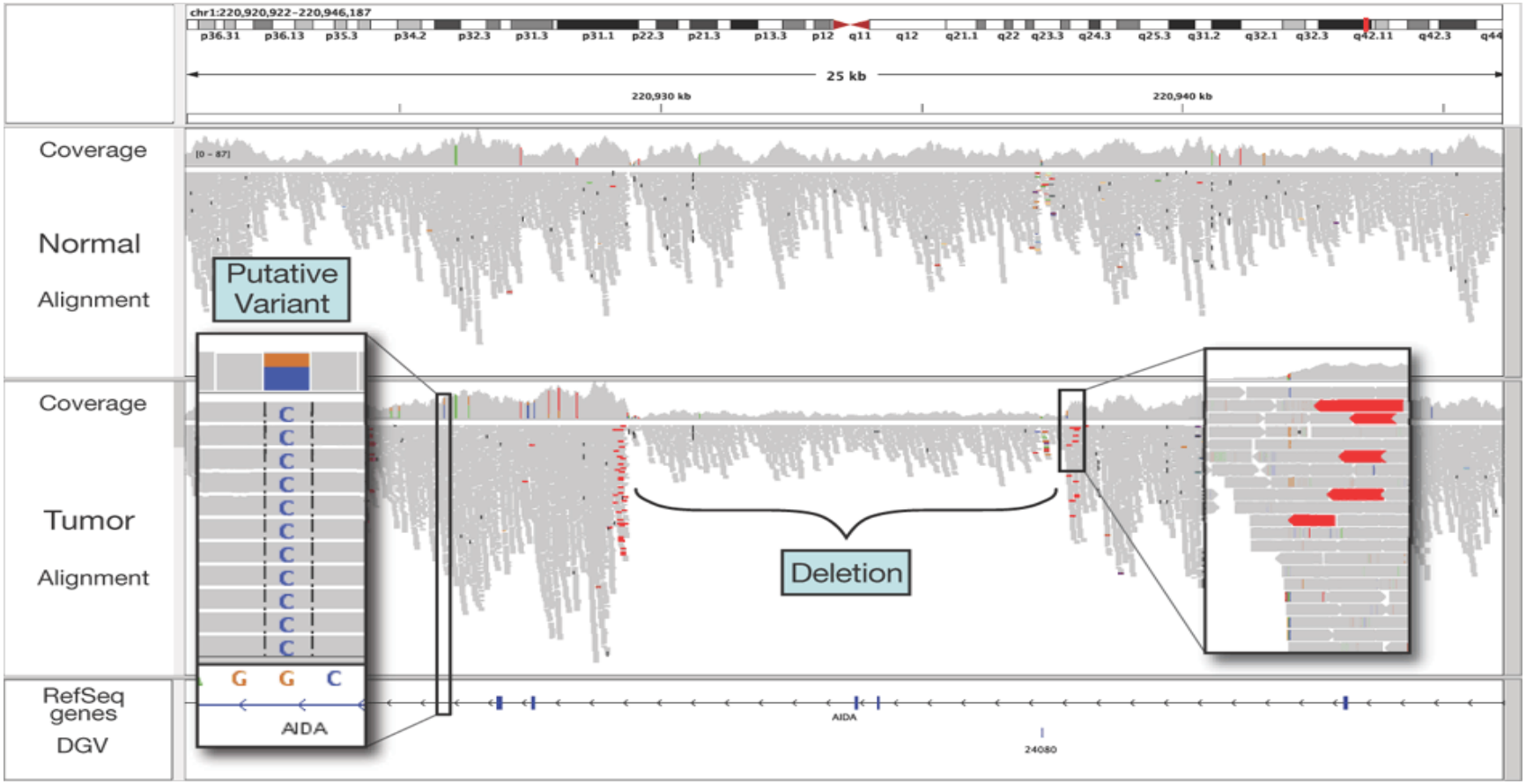
Other SAM/BAM files

- Text SAM files (.sam): standard form
- BAM files (.bam): binary representation of SAM
  - more compact, faster to process, random access and indexing possible
- BAM index files (.bai) allow random access in a BAM file that is sorted by position

SAMtools

- The SAMtools are a set of simple tools to
  - convert between SAM and BAM
  - sort and merge SAM files
  - index SAM and FASTA files for fast access
  - calculate tallies (“flagstat”)
  - view alignments (“tview”)
  - produce a “pile-up”, i.e., a file showing
    - local coverage
    - mismatches and consensus calls
    - indels
- http://samtools.sourceforge.net/

Visualization of SAM files.



- Integrative Genomics Viewer (IGV): Robinson et al., Broad Institute

Homework 4 (next week)

- Download data from course website
  - Reference genome and 50 reads
- Download Bowtie and install it
- Indexing and alignment
- Find uniquely aligned reads
  - If you make a program to this, you can get extra credits.
  - If using perl, you may use “hash” to determine the uniquely aligned reads.