TOPIC 4: Sequence alignment

Bill 525D - Bioinformatics for Evolutionary Biology

Learning Goals

- Be able to define the two main methods of alignment.
- Understand the two main algorithms for NGS alignment, including strengths and weaknesses.
- Be able to read SAM format

Sequence alignment

- Sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences.

Pairwise alignment

- Alignment of two sequences is a relatively straightforward computational problem, but…
    - there are many possible alignments
    - there can be a very large reference
- NOTE: Two sequences can always be aligned and there can be more than one optimal solution

Methods of alignment

- By hand
- Mathematical approach
  - Dynamic programming (slow, but optimal)
- Heuristic methods (fast, but approximate)
  - BLAST, short read aligners

Align by hand

TGCAGTT
TGGAATCGTT

In groups, align these two sequences
and justify your result

Alignment by hand

```
 TGCAGTT
TGGAATCGTT
```

Scoring:
Matching letter: +1
Mismatch letter: -1
Inserting gap: -2

Alignment by hand

```
              TGCA---GTT
              TGGAATCGTT
   Matches    +1+1  +1      +1+1+1
Mismatches        -1
      Gaps             -2-2-2
```

Total score = -1

Scoring:
Matching letter: +1
Mismatch letter: -1
Inserting gap: -2

Alignment by hand

```
              TGCA---GTT
              TGGAATCGTT
Matches      +1+1  +1        +1+1+1
Mismatches        -1
Gaps                 -2-2-2
```

Total score = -1

Scoring:
Matching letter: +1
Mismatch letter: -1
Inserting gap: -2

What is the best alignment if the gap penalty is zero?

Dynamic programming

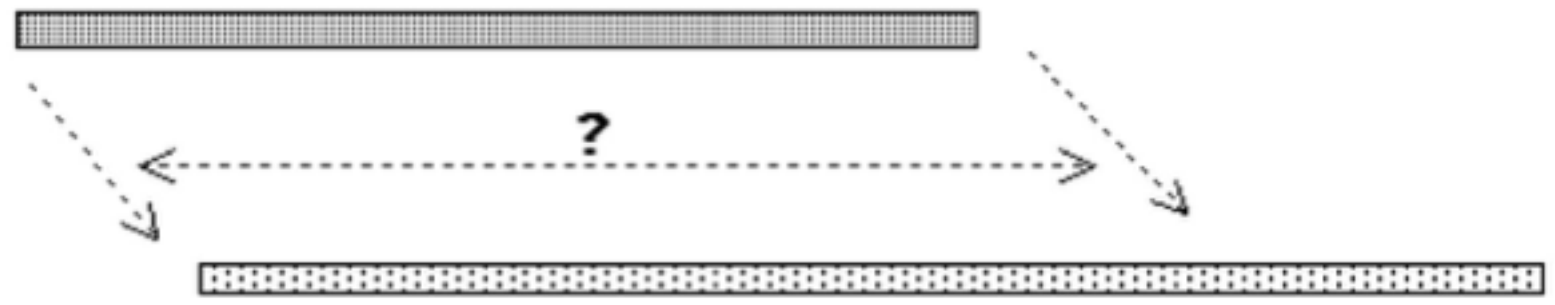
- Dynamic programming is a general programming technique.
- It structures a large search space into a succession of stages
    - The initial stage contains trivial solutions to sub-problems
    - Each partial solution in a later stage can be calculated by recurring a fixed number of partial solutions in an earlier stage
    - The final stage contains the overall solution

Global vs Local alignments

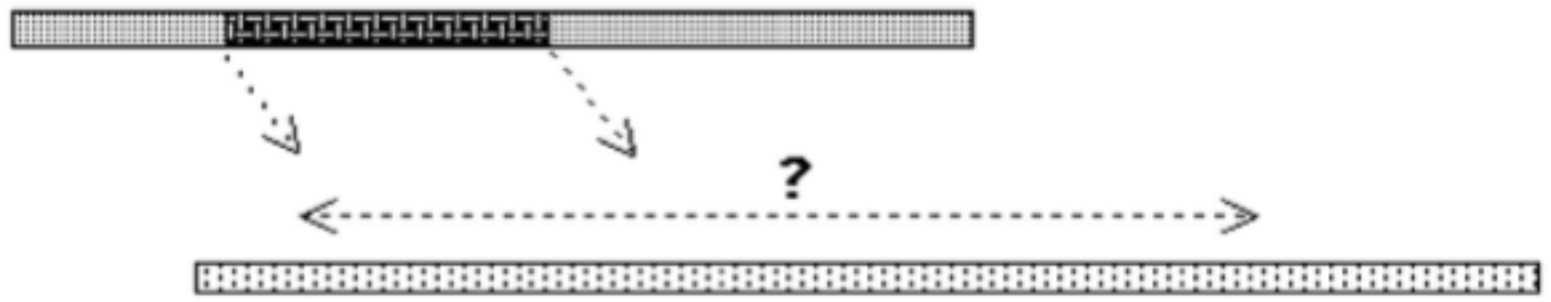
- Global alignment algorithms start at the beginning of two sequences and add gaps to each until the end of one is reached (Needleman-Wunsch).

- Local alignment algorithms finds the region (or regions) of highest similarity between two sequences and build the alignment outward from there (Smith-Waterman).

Global Alignment



Local Alignment



Think-Pair-Share

- When would you use global alignment over local alignment?
- Which one is faster?

Basic principles of dynamic programming

- There are too many comparisons to try them all so instead:
    - Build alignment path matrix
    - Stepwise calculation of score values
    - Backtracking (evaluation of optimal path)

Build an alignment path matrix

- For sequences *x(1:i)* and *y(1:j)*:
  - If $F(i-1,j-1)$, $F(i-1,j)$ and $F(i,j-1)$ are known we can calculate $F(i,j)$
  - Three possibilities:
    - x_i and y_j are aligned, $F(i,j) = F(i-1,j-1) + s(x_i, y_j)$
    - x_i is aligned to a gap, $F(i,j) = F(i-1,j) - d$
    - y_j is aligned to a gap, $F(i,j) = F(i,j-1) - d$
      - The best score up to (i,j) will be the **largest** of the three options
    - d = gap penalty

Dynamic Programming

- Global alignment (Needleman-Wunsch) algorithm
- Example: align GATC to GAC

<u>Scoring system:</u>
Match =+1
Mismatch = -1
Gap = -2

	-	G	A	T	C
-	0				
G					
A					
C					

Dynamic Programming

- Global alignment (Needleman-Wunsch) algorithm
- Example: align GATC to GAC

<u>Scoring system:</u>
Match =+1
Mismatch = -1
Gap = -2

	-	G	A	T	C
-	0 →	-2 →	-4 →	-6 →	-8
G	↓ -2				
A	↓ -4				
C	↓ -6				

Dynamic Programming

- Global alignment (Needleman-Wunsch) algorithm
- Example: align GATC to GAC

<u>Scoring system:</u>

Match =+1

Mismatch = -1

Gap = -2

	-	G	A	T	C
-	0 ↘	-2 ↓	-4	-6	-8
G	-2 →	1			
A	-4				
C	-6				

Dynamic Programming

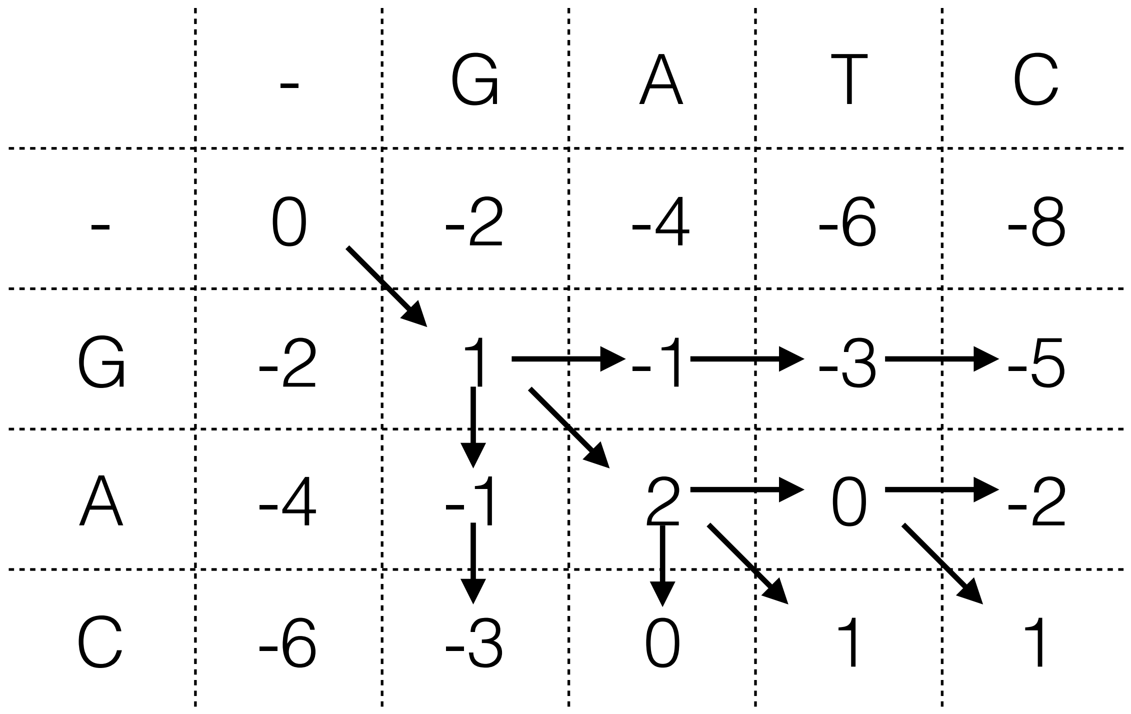
- Global alignment (Needleman-Wunsch) algorithm
- Example: align GATC to GAC

<u>Scoring system:</u>

Match =+1

Mismatch = -1

Gap = -2



	-	G	A	T	C
-	0	-2	-4	-6	-8
G	-2	1	-1	-3	-5
A	-4	-1	2	0	-2
C	-6	-3	0	1	1

Dynamic Programming

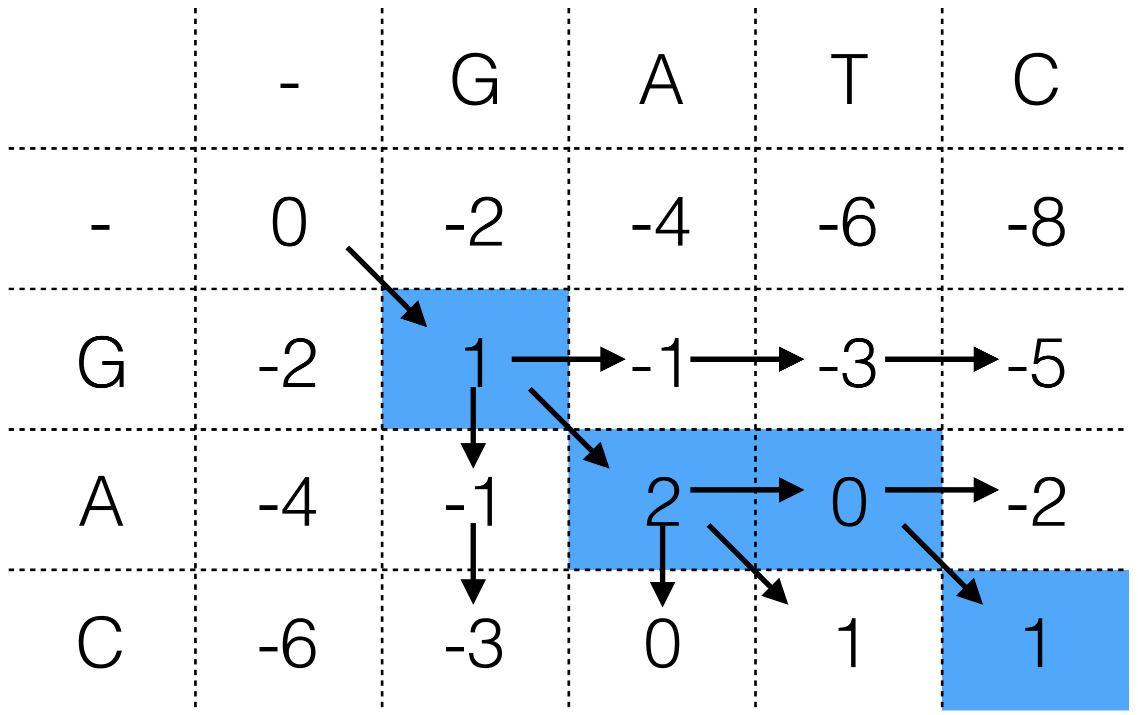
- Global alignment (Needleman-Wunsch) algorithm
- Example: align GATC to GAC

Scoring system:

Match =+1

Mismatch = -1

Gap = -2



	-	G	A	T	C
-	0	-2	-4	-6	-8
G	-2	1	-1	-3	-5
A	-4	-1	2	0	-2
C	-6	-3	0	1	1

```
GATC
GA-C
```

Smith-Waterman local alignment

- Variation on the Needleman-Wunsch algorithm that guarantees best local alignment of any possible length.

Scoring methods

- Scoring systems:
  - Each symbol pairing is assigned a numerical value, based on a symbol comparison table.
    - nucleotides
    - amino acids (PAM, BLOSUM)
- Gap penalties:
  - Opening: The cost of introducing a gap.
  - Extension: The cost to elongate a gap.

Gap penalties

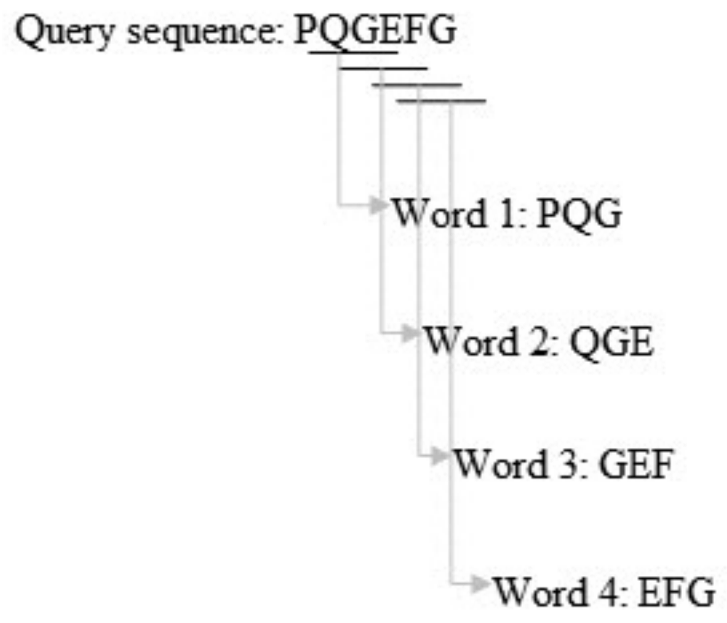
- Too little gap penalty gives nonsense non-homologous alignments.

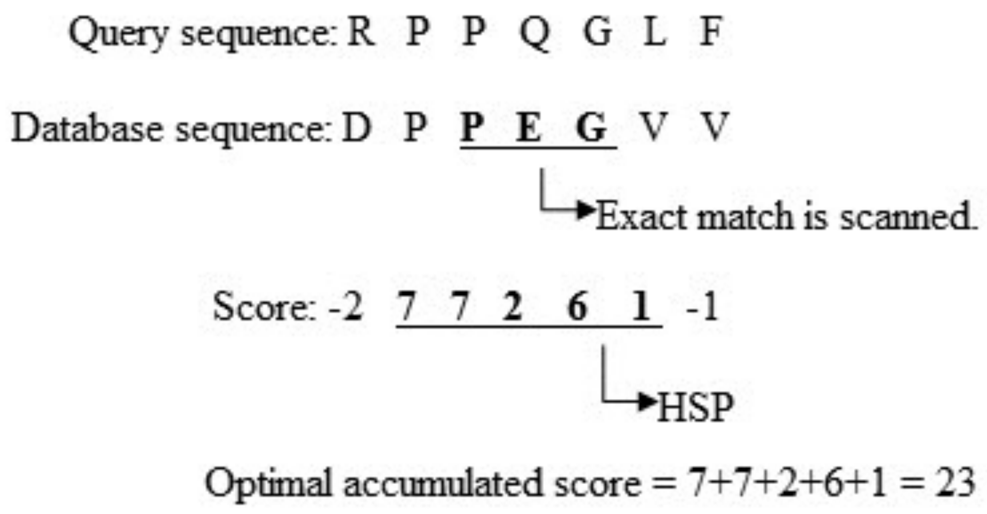
- Gaps are common, so too high gap penalty removes real alignments.

- “Affine” gap penalty has a large penalty to introduce a gap and a smaller penalty to extend one.

BLAST - Best Local Alignment Search Tool

- Designed to identify homologous sequences.
- Hashed seed-extend algorithm
- First finds highly conserved or identical sequences which are then extended with a local alignment

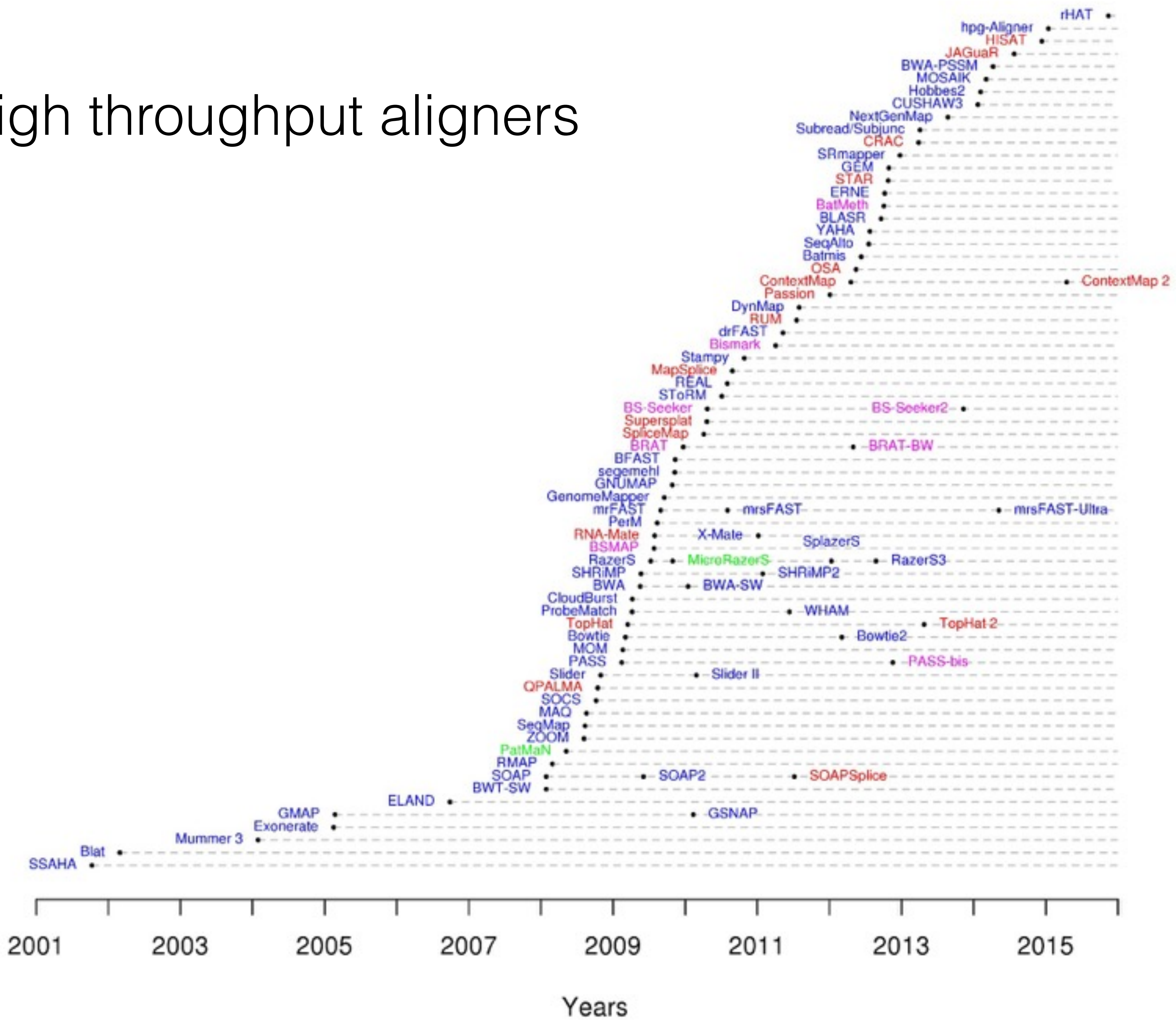



Query sequence: R P P Q G L F
Database sequence: D P P E G V V
Exact match is scanned.
Score: -2 7 7 2 6 1 -1
HSP
Optimal accumulated score = 7+7+2+6+1 = 23

BLAST

- Why not use BLAST for short read data?
  - Typically takes 0.1 to 1 second to search 1 sequence against a database
  - 60 million reads equates to 70 CPU days

High throughput aligners

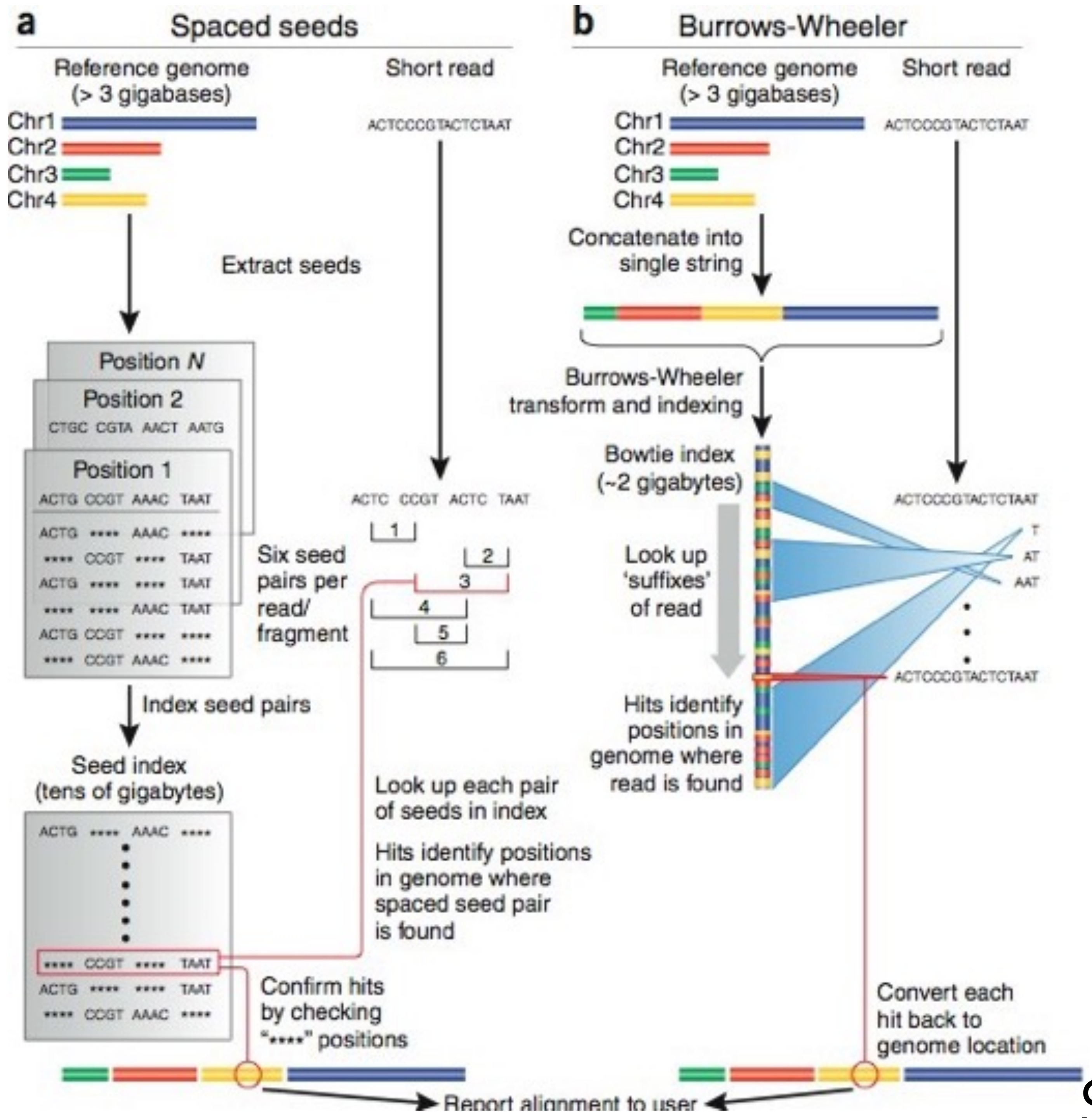


http://www.ebi.ac.uk/~nf/hts_mappers/

Short read alignment is hard

- Billions of short sequences aligned to a very long reference
- Short reads contain less information and are less likely to have a unique mapping location

Approaches to align short reads


a
Spaced seeds
Reference genome (> 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Extract seeds
Position N
Position 2
CTGC CGTA AACT AATG
Position 1
ACTG CCGT AAAC TAAT
ACTC CCGT ACTC TAAT
Six seed pairs per read/ fragment
Index seed pairs
Seed index (tens of gigabytes)
Look up each pair of seeds in index
Hits identify positions in genome where spaced seed pair is found
Confirm hits by checking "****" positions
b
Burrows-Wheeler
Reference genome (> 3 gigabases)
Short read
Concatenate into single string
Burrows-Wheeler transform and indexing
Bowtie index (~2 gigabytes)
Look up 'suffixes' of read
Hits identify positions in genome where read is found
Convert each hit back to genome location
Report alignment to user

Trapnell & Salzberg 2009

Hashed seed-extend algorithms

- Two step process:
    - Identify a match to the seed sequence in the reference
    - Extend match using sensitive (but slow) Smith-Waterman algorithm

Seed-extend algorithm

Reference sequence:

…GATCTCGATCGATGATCGTAGGATTGATCAGCTA…

Short read:

TCGATCGATGATCGAAGGATTGATCAG

Seed-extend algorithm

Reference sequence:

…GATCTCGATCGATGATCGTAGGATTGATCAGCTA…

Short read:

TCGATCGAT	GATCG$\underline{A}$AGG	ATTGATCAG
9bp seed	9bp seed	9bp seed

The algorithm will try to match each seed to the reference. If there is a match with any seed, it performs a local alignment

Seed-extend algorithm

Reference sequence:

```
                    seed     ->Extend with Smith-Waterman->
 …GATCTCGATCGATGATCGTAGGATTGATCAGCTA…
       TCGATCGATGATCGAAGGATTGATCAG
```

Short read:

TCGATCGAT	GATCGAAGG	ATTGATCAG
9bp seed	9bp seed	9bp seed

Here there is a match with at least one seed

Seed-extend algorithm

Reference sequence:

```
...GATCTCGATCGATGATCGTAGGATTGATCAGCTA...
```

Short read:

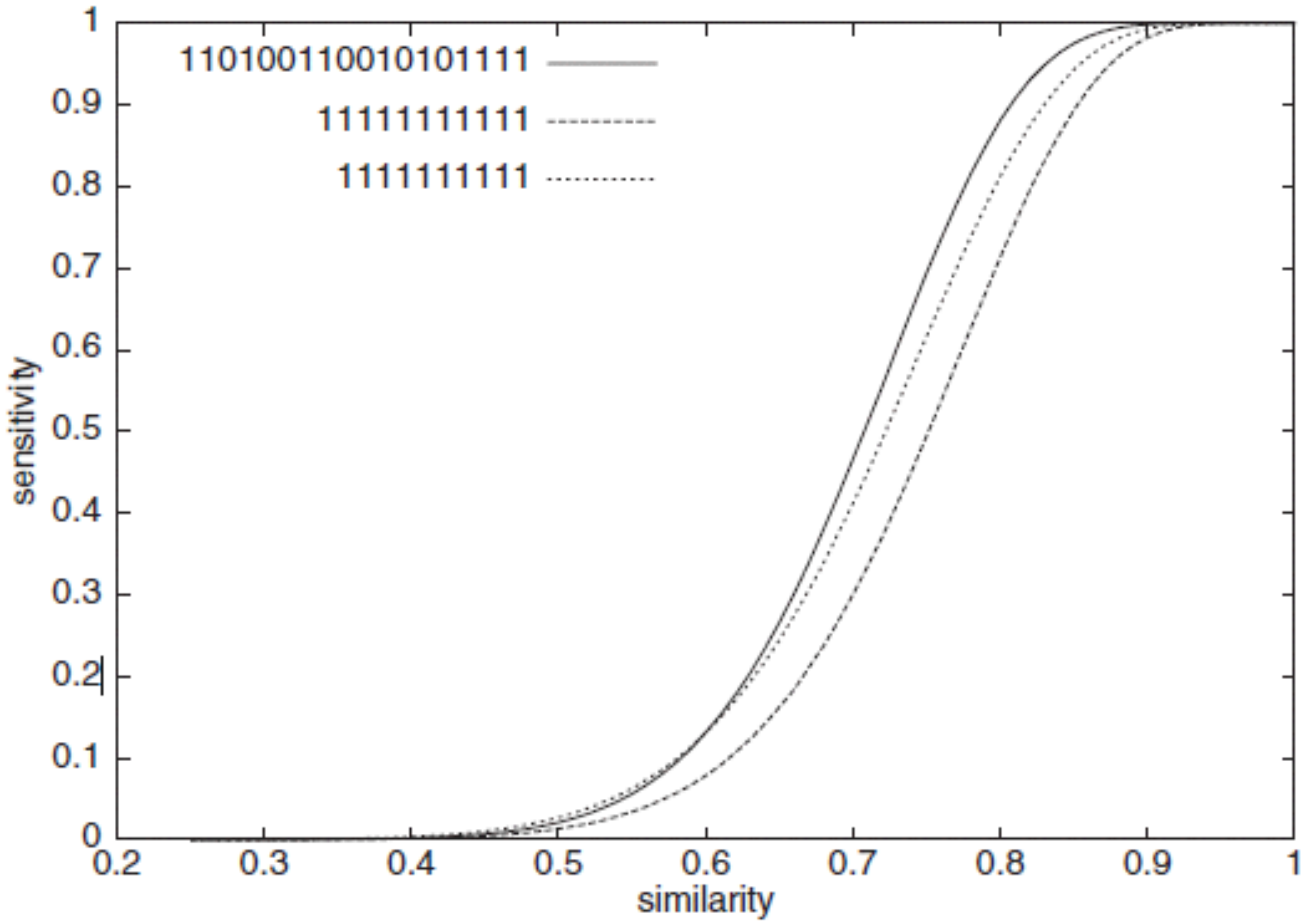
```
TAGATCGAT     GATCGAAGG     ATTGAGCAG
```

9bp seed   9bp seed   9bp seed

With three sequencing errors/SNPs, there can be no matches

Spaced seeds

- To increase sensitivity we can used spaced-seeds:



Spaced seeds

- To increase sensitivity we can used spaced-seeds:

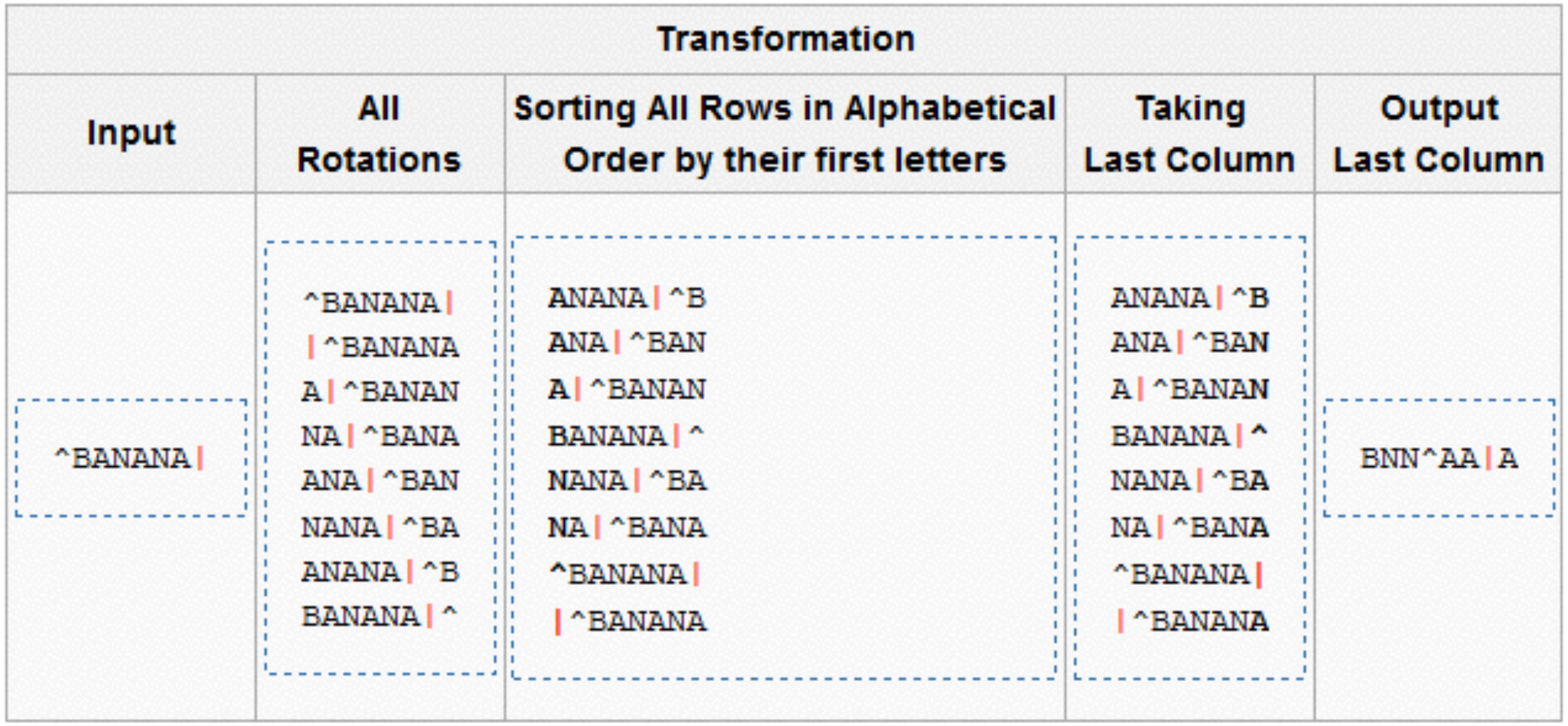
```
   111111111     Consecutive seed template with length 9bp
GATAGCTAGCTAAT   Reference
   AGCTAGCTA     Query


10101101011011   Consecutive seed template with weight 9bp
GATAGCTAGCTAAT   Reference
GATAGCGAGCTAAT   Query
```

Suffix-Prefix Trie

- A family of methods which uses a Trie structure to search a reference sequence (e.g. Bowtie, BWA, SOAP2)
- Trie – data structure which stores the suffixes (i.e. ends of a sequence)
- Key advantage over hashed algorithms:
    - Alignment of multiple copies of an identical sequence in the reference only needs to be done once
    - Use of an FM-Index to store Trie can drastically reduce memory requirements (e.g. Human genome can be stored in 2Gb of RAM)
    - Burrows Wheeler Transform to perform fast lookups

Burrows-Wheeler Algorithm

- Encodes data so that it is easier to compress
- Can be reversed to recover the original word



Transformation				
Input	**All Rotations**	**Sorting All Rows in Alphabetical Order by their first letters**	**Taking Last Column**	**Output Last Column**
^BANANA|	^BANANA|	ANANA|^B	ANANA|^B	BNN^AA|A
	|^BANANA	ANA|^BAN	ANA|^BAN	
	A|^BANAN	A|^BANAN	A|^BANAN	
	NA|^BANA	BANANA|^	BANANA|^	
	ANA|^BAN	NANA|^BA	NANA|^BA	
	NANA|^BA	NA|^BANA	NA|^BANA	
	ANANA|^B	^BANANA|	^BANANA|	
	BANANA|^	|^BANANA	|^BANANA	

Comparison

Hash referenced spaced seeds (NextGenMap)

- Requires more RAM
- Runs slower
- Simpler to program
- More sensitive

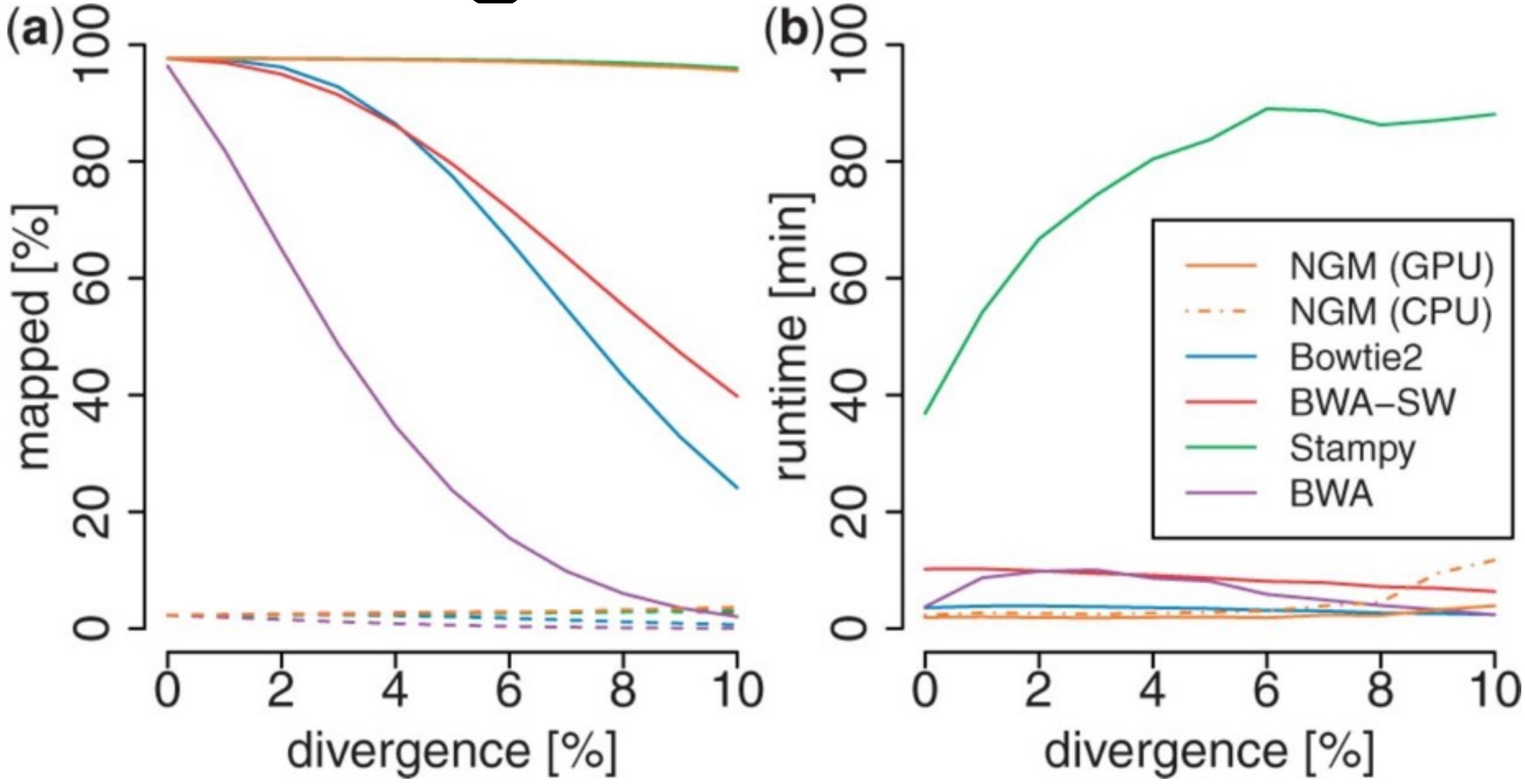
Suffix/Prefix Trie (BWA)

- Requires less RAM
- Runs much faster
- Complicated to program
- Less sensitive

Popular short read aligners

Program	Algorithm	Speed	Accuracy in for divergent sequences
Bowtie2	FM-index	Very fast	Low
BWA	FM-index	Fast	Medium
Stampy	Hashing ref	Slow	High
Soap2	FM-index	Fast	Low
Novoalign	Hashing ref	Slow	High
NextGenMap	Hashing ref	Fast	High

Alignment stats



*From NextGenMap paper

Alignment choice

- Speed needed?
- How divergent is sequence from reference? Same species or relative?
- How much variation in your samples?
- Genome size of reference?

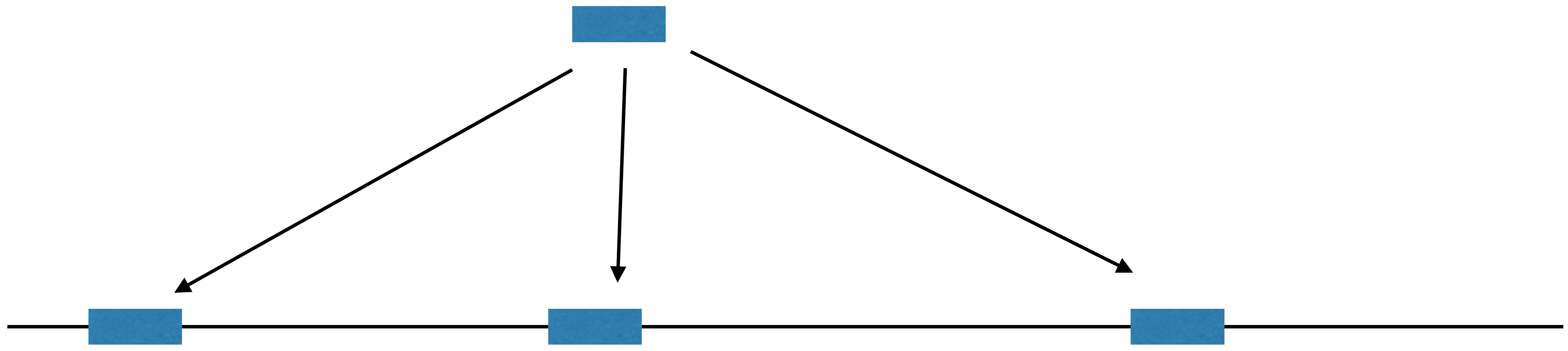
Other considerations

- PCR duplicates
- Multi-mapping reads
- Spliced-read mapping

PCR duplicates

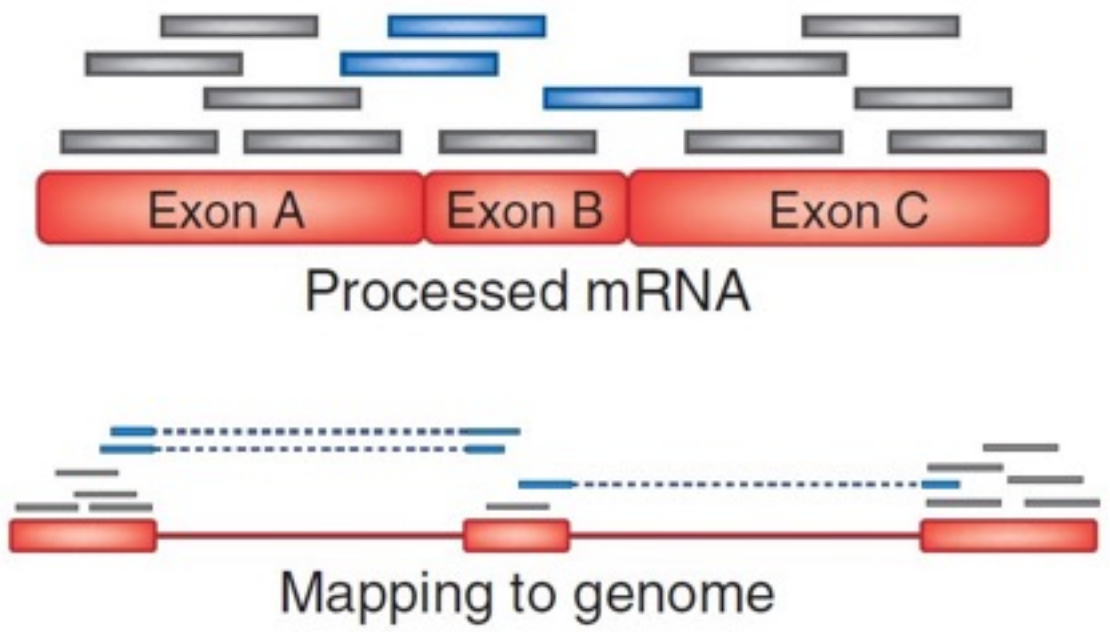
- Most library preps have at least one PCR amplification step
  - PCR can introduce errors and then sequencing multiple copies makes it seem like a real SNP
  - SAMtools and Picard can flag or remove these duplicates based on alignment location
    - Samples with same start and stop position are considered duplicates
    - Don’t flag duplicates for GBS (set start and stop)

Multiple mapping reads



- A single read may occur more than once in a reference genome, due to gene/chromosome duplication or repetitive elements
- Reads may be assigned to one random location
- Affects mapping quality

Spliced-read mapping



- Need to account for splicing
- Examples: TopHat, SubRead, Star

SAM (BAM) format

- Sequence Alignment/Map format
  - Universal standard.
  - Generally aligned to reference, but not necessarily
  - Human-readable (SAM) and compressed (BAM) forms
- Structure:
  - Header: Version, sort order, reference sequences, read groups, program/processing history
  - Alignment records

SAM format

Sort order

Reference sequence name and length

Read group information

```
@HD	VN:1.5  GO:none SO:coordinate
@SQ	SN:cp_gi_88656873	LN:151104
@SQ	SN:mt_gi_571031384	LN:300945
@SQ	SN:rDNA_gi_563582565	LN:9814
@SQ	SN:Ha1  LN:175985764
@SQ	SN:Ha2  LN:209013747
@SQ	SN:Ha3  LN:203472901
@SQ	SN:Ha4  LN:216026857
@SQ	SN:Ha5  LN:271056985
@SQ	SN:Ha6  LN:100519666
@SQ	SN:Ha7  LN:109221022
@SQ	SN:Ha8  LN:192129815
@SQ	SN:Ha9  LN:253478808
@SQ	SN:Ha10 LN:327788049
@SQ	SN:Ha11 LN:208730832
@SQ	SN:Ha12 LN:208068730
@SQ	SN:Ha13 LN:239367298
@SQ	SN:Ha14 LN:230295834
@SQ	SN:Ha15 LN:202246870
@SQ	SN:Ha16 LN:226777971
@SQ	SN:Ha17 LN:267415242
@SQ	SN:Ha0_73Ns	LN:359367108
@RG	ID:HI.2034.006.Index_18.W70_NHK_2013_5  LB:Anomalus	PL:ILLUMINA	SM:HI.2034.006.Index_18.W70_NHK_2013_5  PU:Anomalus
@PG	ID:ngm  PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --dualstrand 1
@PG	ID:ngm.1	PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
@PG	ID:ngm.2	PN:ngm  CL:" --affine 0 --argos_min_score 0 --bam 1 --block_multiplier 2 --bs_cutoff 6 --bs_mapping 0 --cpu_threads 11 --
```

Program information

SAM format

Read lines

SRR035022 163 chr16 59999 37 22D54M = 60102 179 CCAACCCAAC... >AAA=>?AA... XT:A:M XN:i:2 SM:i:37

<QNAME> <FLAG> <RNAME> <POS> <MAPQ> <CIGAR> <MRNM> <MPOS> <ISIZE> <SEQ> <QUAL> [<TAG>]

Mapping Quality

- MapQ = Qs = $-10 \log_{10}(P)$
- P = probability that this mapping is NOT the correct one
- MapQ = 0 = equally likely to map somewhere else
- Different programs use different formulas for P