Bioinformatics for Evolutionary Biology

Alignments: algorithms and tools

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.

Note:

- When we align sequences we assume they are similar
- Protein sequence conservation and DNA sequence conservation

Orthologous and paralogous

- Orthologous sequences differ because they are found in different species
- Paralogous sequences differ due to a gene duplication event
- Homoeologous sequences originated by a process of polyploidization
- Sequences may be all three

Types of alignments

- Pair-wise alignments - comparing two sequences at a time.

- Multiple alignments - comparing more than two sequences: dymanic programming, progressive alignment (iterative), HMM, ...

Pairwise alignment

The alignment of two sequences (DNA or protein) is a relatively straightforward computational problem.

Challenge1: there are many possible alignments.

Challenge2: large database/reference.

- Note:
- Two sequences can **always** be aligned.
- Often there is **more than one** solution with the same score.

Methods of alignment

- By hand - slide sequences on two lines of a word processor
- Dot plot
  - with windows
- Mathematical approach
  - Dynamic programming (slow, optimal)
- Heuristic methods (fast, approximate)
  - BLAST
  - Short-reads aligners

Align by hand

GATCGCCTA_TTACGTCCTGGAC <--

--> AGGCATACGTA_GCCCTTTCGC

You still need some kind of scoring method to find the best alignment

Alignment cost

Points for a matching letter: 1

Points for a non-matching letter: -1

Points for inserting a gap: -2

ATCGAT-ACG

ATGGATTACG

Pair-wise alignment

ATCGAT-ACG

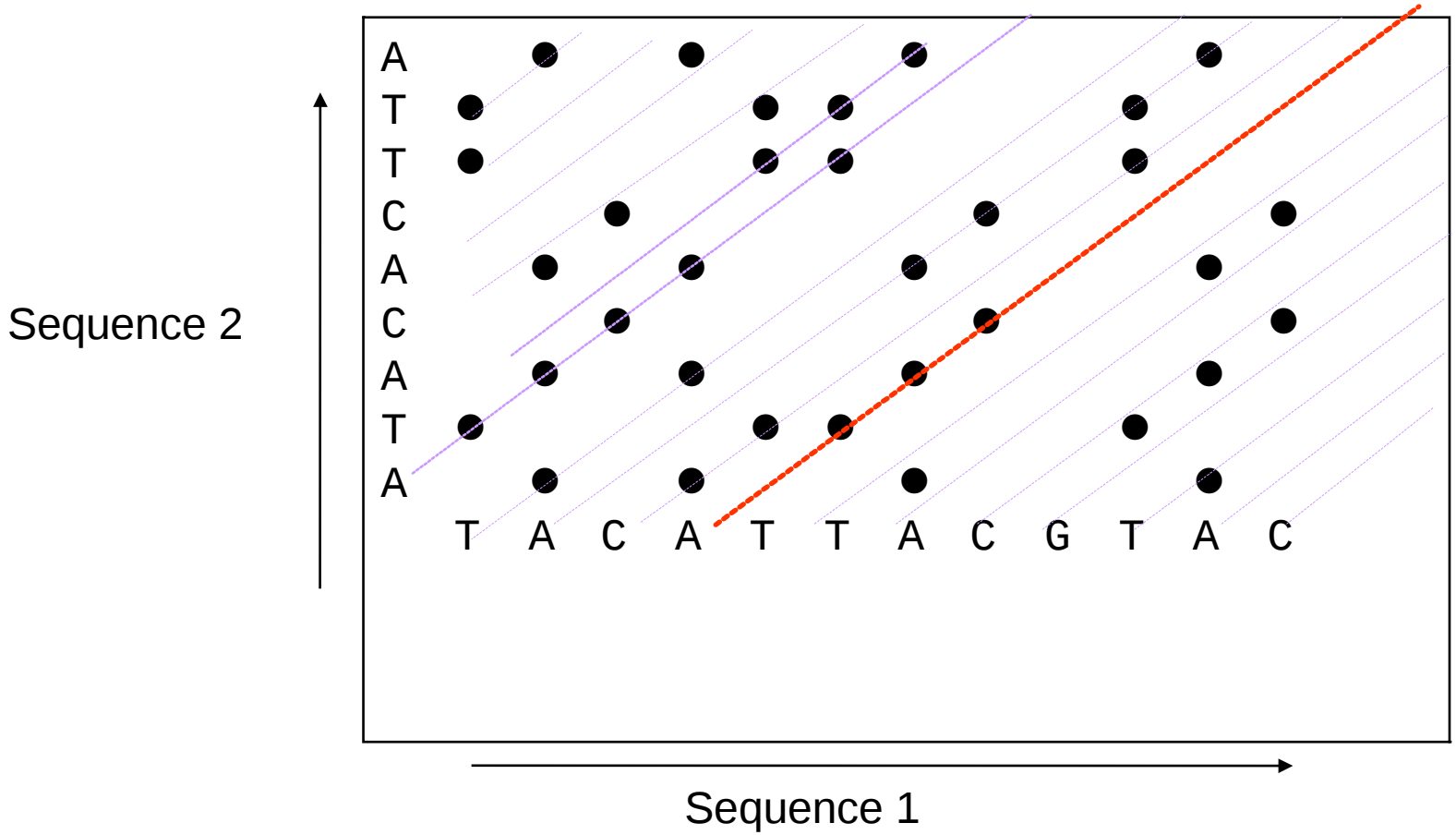
ATGGATTACG

Matches:	+1+1 +1 +1+1 +1 +1+1	= +8
Mismatches:	-1	= -1
Gaps:	-2	= -2
		Total score = +5

Dotplot

A dotplot gives an overview of all possible alignments

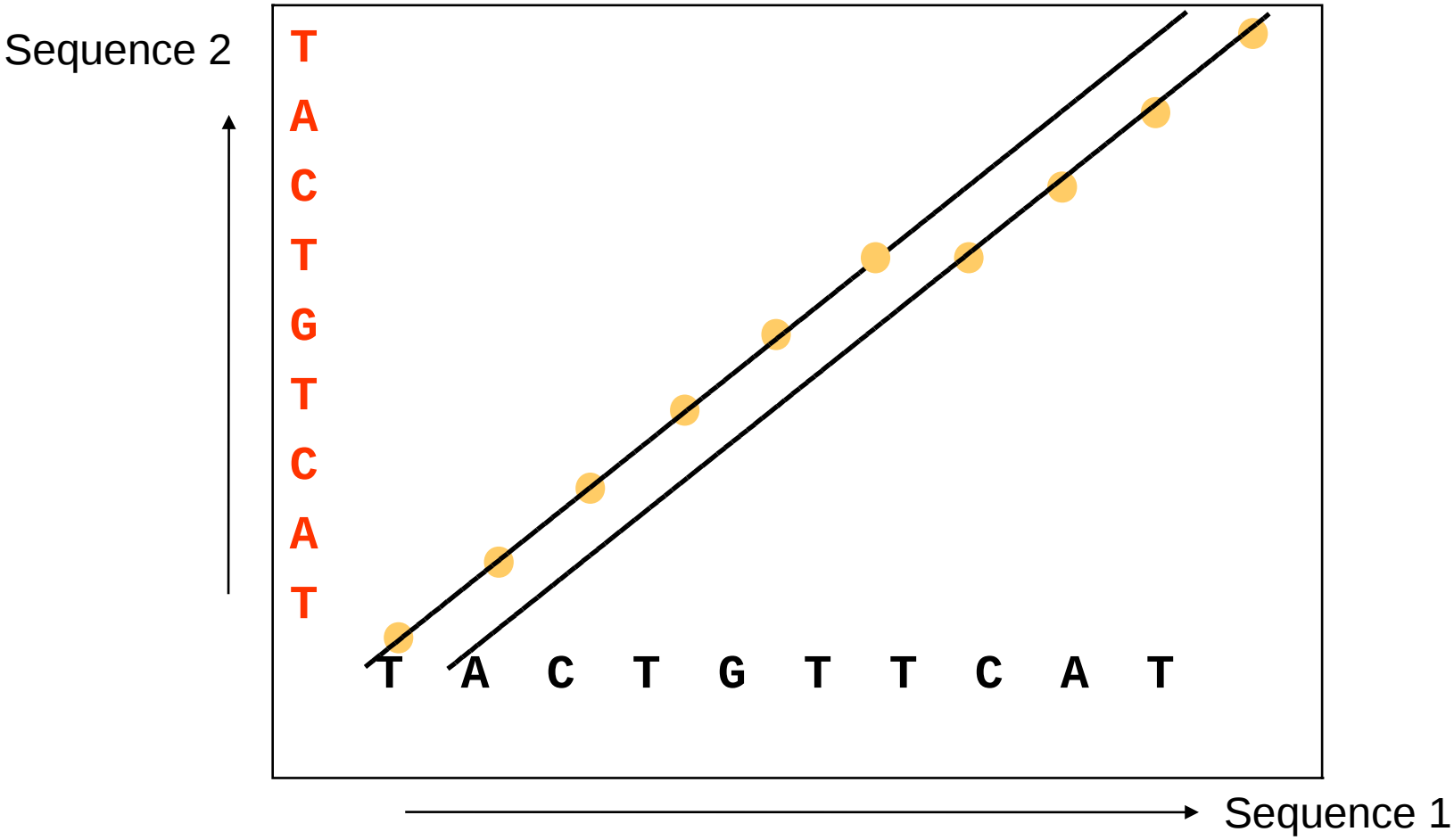
In a dotplot each diagonal corresponds to a possible (ungapped) alignment



One possible alignment:

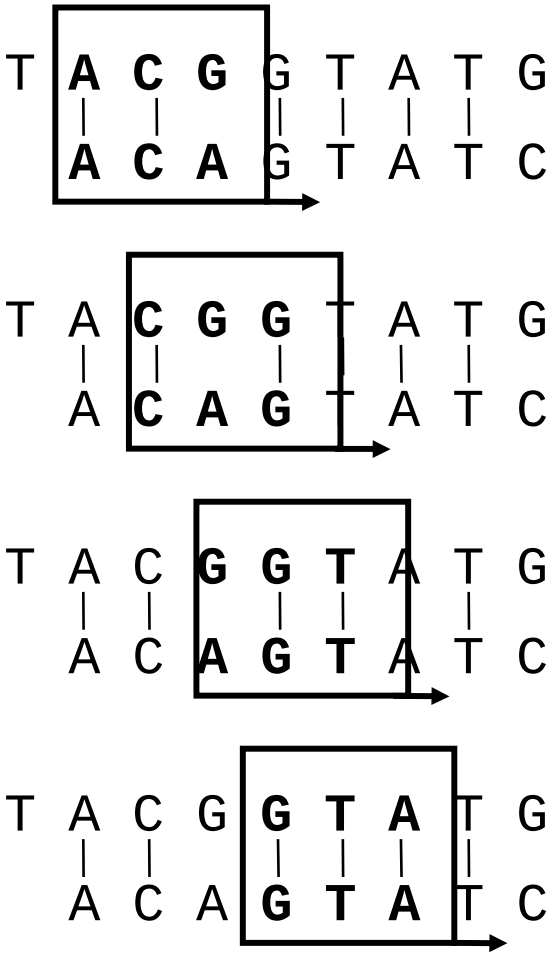
```
T A C A T T A C G T A C
        | | |
      A T A C A C T T A
```

Insertions / Deletions in a Dotplot

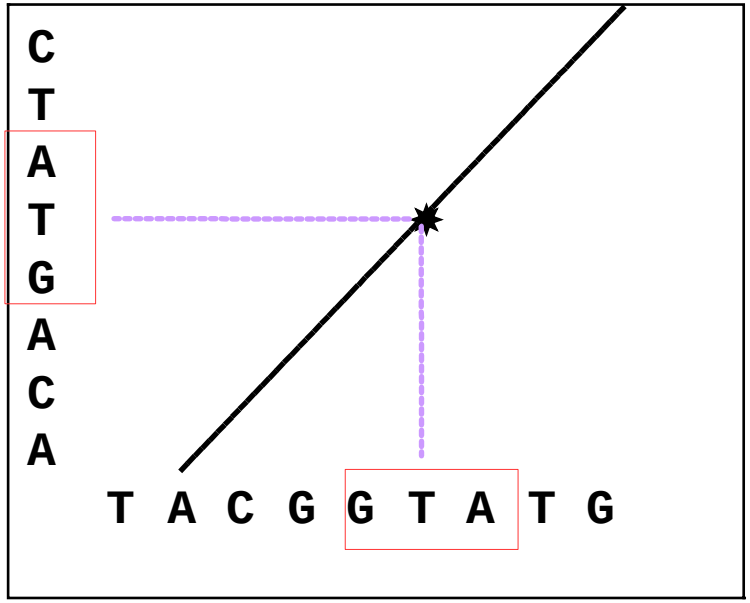



```
T A C T G - T C A T
| | | | |   | | | |
T A C T G T T C A T
```

Word size and stringency



Word Size = 3



Window / Stringency

Score = 11

```
PTHPLASKTQILPEDLASEDLTI
|| |||     |     ||  ||
PTHPLAGERAIGLARLAEEDFGM
```

Score = 11

```
PTHPLASKTQILPEDLASEDLTI
 | | | ||     |    ||  ||
PTHPLAGERAIGLARLAEEDFGM
```

Score = 7

```
PTHPLASKTQILPEDLASEDLTI
|||| |     |    || ||
PTHPLAGERAIGLARLAEEDFGM
```

Scoring Matrix Filtering

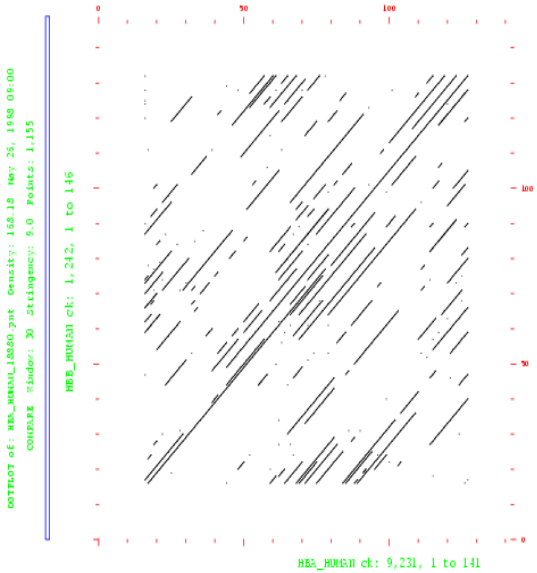
Matrix: PAM250

Window = 12

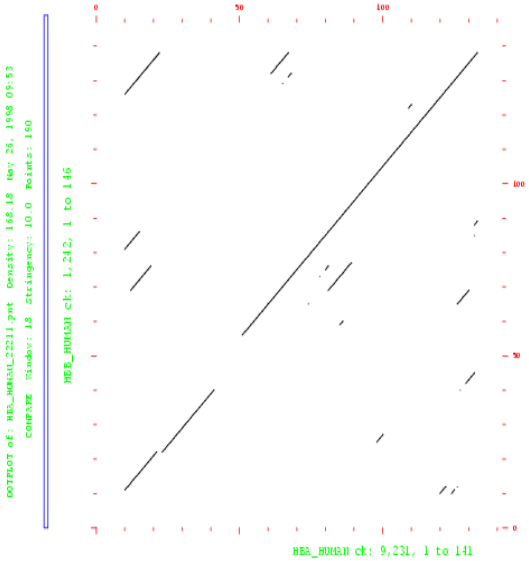
Stringency = 9

Dotplot

Window = 130
Stringency = 9



Window = 18
Stringency = 10



Dotplot Considerations

- The smaller the window, the larger the weight of statistical (unspecific) matches.
- With large windows the sensitivity for short sequences is reduced.
- Insertions/deletions are not treated explicitly.

Dynamic programming

- Dynamic programming is a very general programming technique.
- It is applicable when a large search space can be structured into a succession of stages, such that:

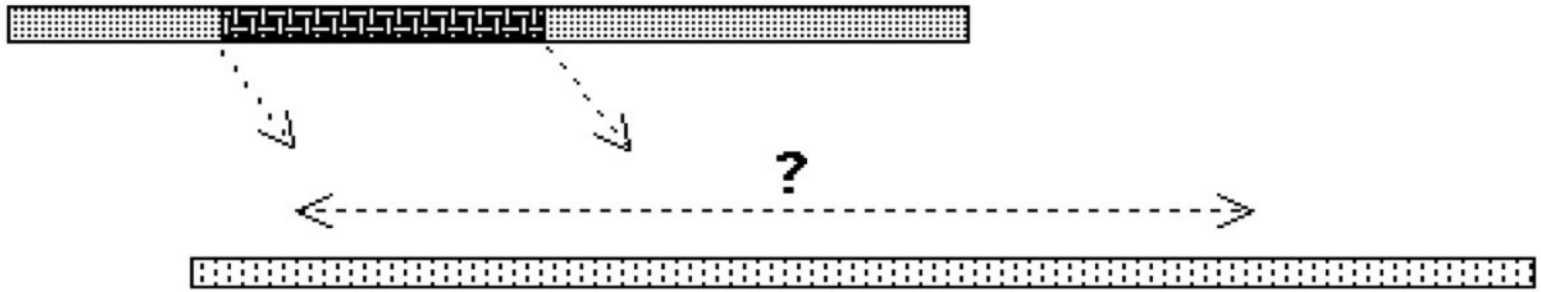
  - The initial stage contains trivial solutions to sub-problems
  - Each partial solution in a later stage can be calculated by recurring a fixed number of partial solutions in an earlier stage
  - The final stage contains the overall solution

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



Basic principles of dynamic programming

There are about $2^{2N}/2N$ comparisons

$N=300 \rightarrow 10^{179}$ different alignments

- **Build alignment path matrix**
- **Stepwise calculation of score values**
- **Backtracking (evaluation of the optimal path)**

Build an alignment path matrix

- For sequences *x(1:i)* and *y(1:j):*
- If $F(i\text{-}1,j\text{-}1)$, $F(i\text{-}1,j)$ and $F(i,j\text{-}1)$ are known we can calculate $F(i,j)$

- Three possibilities:
  - x_i and y_j are aligned, $F(i,j) = F(i\text{-}1,j\text{-}1) + s(x_i, y_j)$
  - x_i is aligned to a gap, $F(i,j) = F(i\text{-}1,j) - d$
  - y_j is aligned to a gap, $F(i,j) = F(i,j\text{-}1) - d$

- The best score up to (i,j) will be the **largest** of the three options

Formal description of dynamic programming algorithm

For two sequences **a** = $a_1, a_2, ..a_i$ and **b** = $b_1, b_2, ..b_j$, where $S_{ij} = S(a_1, \ldots a_i, b_1, \ldots b_j)$ then

$$
\begin{aligned}
S_{ij} = \max \{ & S_{i-1, j-1} + s(a_i b_j), \\
& \max (S_{i-x, j} - w_x), \\
& x \geq 1 \\
& \max (S_{i\, j-y} - w_x), \\
& y \geq 1 \\
& \}
\end{aligned}
$$

where S_{ij} is the score at position at *i* in sequence **a** and *j* in sequence **b**, $s(a_i b_j)$ is score for aligning the character at positions *i* and *j*, w_x is the penalty for a gap of length *x* in sequence **a**, and w_x is the penalty for a gap of length *y* in sequence **b**.

Note: S_{ij} is a type of running best score as the algorithm moves through every position in the matrix – optimal alignment

Dynamic Programming

Global alignment (Needleman-Wunsch) algorithm

Example – align GATC to GAC

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -2

0	-	G	A	T	C
-	0				
G					
A					
C					

Dynamic Programming

Global alignment (Needleman-Wunsch) algorithm

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -2

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

Dynamic Programming

Global alignment (Needleman-Wunsch) algorithm

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -2

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

Dynamic Programming

Global alignment (Needleman-Wunsch) algorithm

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -2

0	-	G	A	T	C
-	0 +1	-2	-4	-6	-8
G	-2	Max= 1			
A	-4				
C	-6				

+ MATCH + GAP

+ GAP

Dynamic Programming

Global alignment (Needleman-Wunsch) algorithm

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -2

0	-	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

Backtracking and final alignment

Global alignment (Needleman-Wunsch algorithm)

Scoring system:
Points for match = +1
Points for mismatch = -1
Points for a gap insertion = -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 of the Needleman-Wunsch algorithm and as such it guarantied to find the best local alignment with respect to the scoring system used.

The main difference to the Needleman–Wunsch algorithm is that negative scoring matrix cells are set to zero.

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix}$$

$$T = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ G & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ C & 0 & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow \end{pmatrix}$$

Scoring method

Scoring Systems:

- Each symbol pairing is assigned a numerical value, based on a symbol comparison table.

  - nucleoties

  - amino acids (PAM, BLOSUM)

Gap Penalties:

- Opening: The cost to introduce a gap
- Extension: The cost to elongate a gap

Protein scoring systems

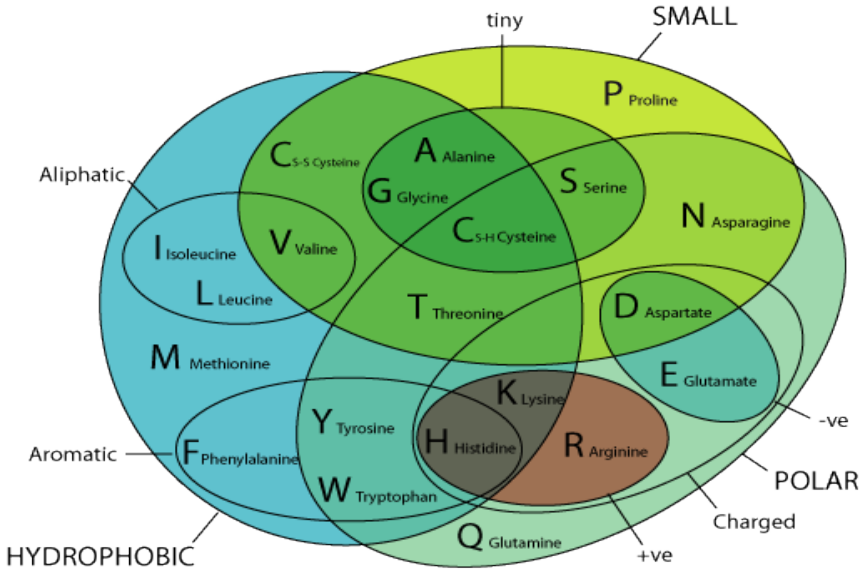
Amino acids have different biochemical and physical properties that influence their relative replaceability in evolution.

PAM1	PAM120	PAM250
BLOSUM80	BLOSUM62	BLOSUM45

← →

Less divergent ***More divergent***

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4



Livingstone & Barton 1993

Scoring gaps

Insertions and Deletions

```
  A T G T A A T G C A
  | | | |
T A T G T G G A A T G A

  A T G T - - A A T G C A
  | | | |     | | | |
T A T G T G G A A T G A
```

insertion / deletion

Introduction of a gap is **penalized** with a negative score value.

Why gap penalties?

Gaps not permitted **Score: 0**

```
1 GTGATAGACACAGACCGGTGGCATTGTGG 29
  |||   |  | |||    |   || || |
1 GTGTCGGGAAGAGATAACTCCGATGGTTG 29
```

Match = 5
Mismatch = -4

Gaps allowed but not penalized **Score: 88**

```
1 GTG.ATAG.ACACAGA..CCGGT..GGCATTGTGG 29
  ||| || | | | |||  ||  |  |  || || |
1 GTGTAT.GGA.AGAGATACC..TCCG..ATGGTTG 29
```

Why Gap Penalties?

- The optimal alignment of two similar sequences usually **maximizes** the number of matches and **minimizes** the number of gaps.
- There is a trade-off between these two - adding gaps reduces mismatches.
- Permitting the insertion of arbitrarily many gaps can lead to high scoring alignments of **non-homologous** sequences.
- Penalizing gaps forces alignments to have relatively few gaps.

Gap Penalties

How to balance gaps with mismatches?

- Gaps must get a steep penalty, or else you'll end up with nonsense alignments.
- In real sequences, multi-base (or amino acid) gaps are quit common (genetic insertion/deletion)
- **"Affine"** gap penalties give a big penalty for each new gap, but a much smaller "gap extension" penalty.

Scoring Insertions and Deletions

match = 1
mismatch = 0

Total Score: 4

```
  A T G T T A T A C
  | | | |
T A T G T G C G T A T A
```

Total Score: 8 - 3.2 = 4.8

<u>Gap parameters:</u>

$d = 3$ (gap opening)

$e = 0.1$ (gap extension)

$g = 3$ (gap length)

$\xi(g) = -d - (g-1)\, e$

$\xi(g) = -3 - (3 - 1)\, 0.1 = -3.2$

```
  A T G T - - - T A T A C
  | | | |       | | | |
T A T G T G C G T A T A
```

insertion / deletion

BLAST – Best Local Alignment Search Tool

- Primarily designed to identify homologous sequences.
- Blast is a hashed seed-extend algorithm.
- Finding seeds significantly increases the speed of BLAST compared to doing a full local alignment over a whole sequence.
- BLAST first finds highly conserved or identical sequences which are then extended with a local alignment.

BLAST - Original version

<u>Example:</u>

Seed size = 4,

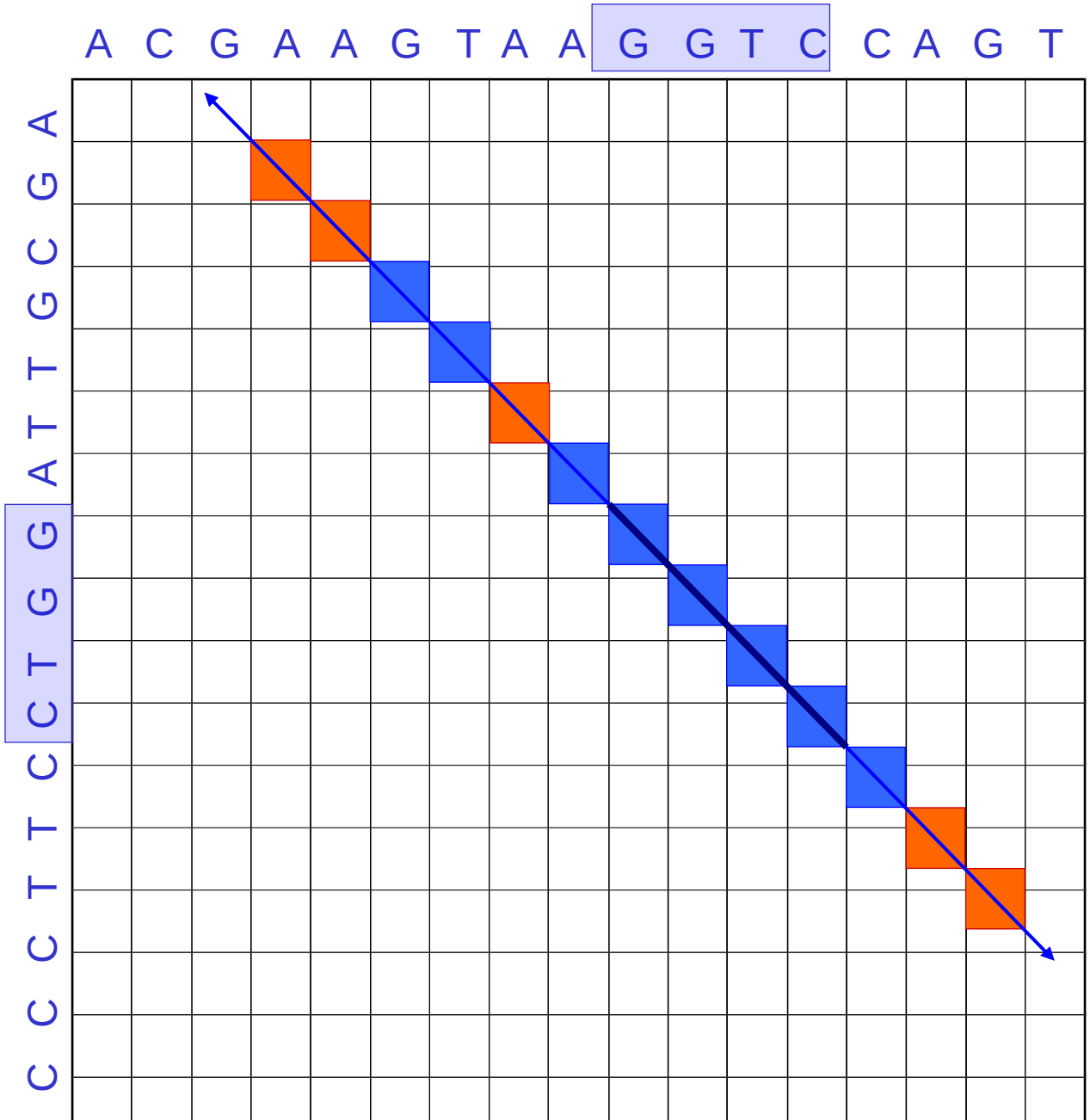
The matching word GGTC initiates an alignment

Extension to the left and right until alignment score falls below 50%

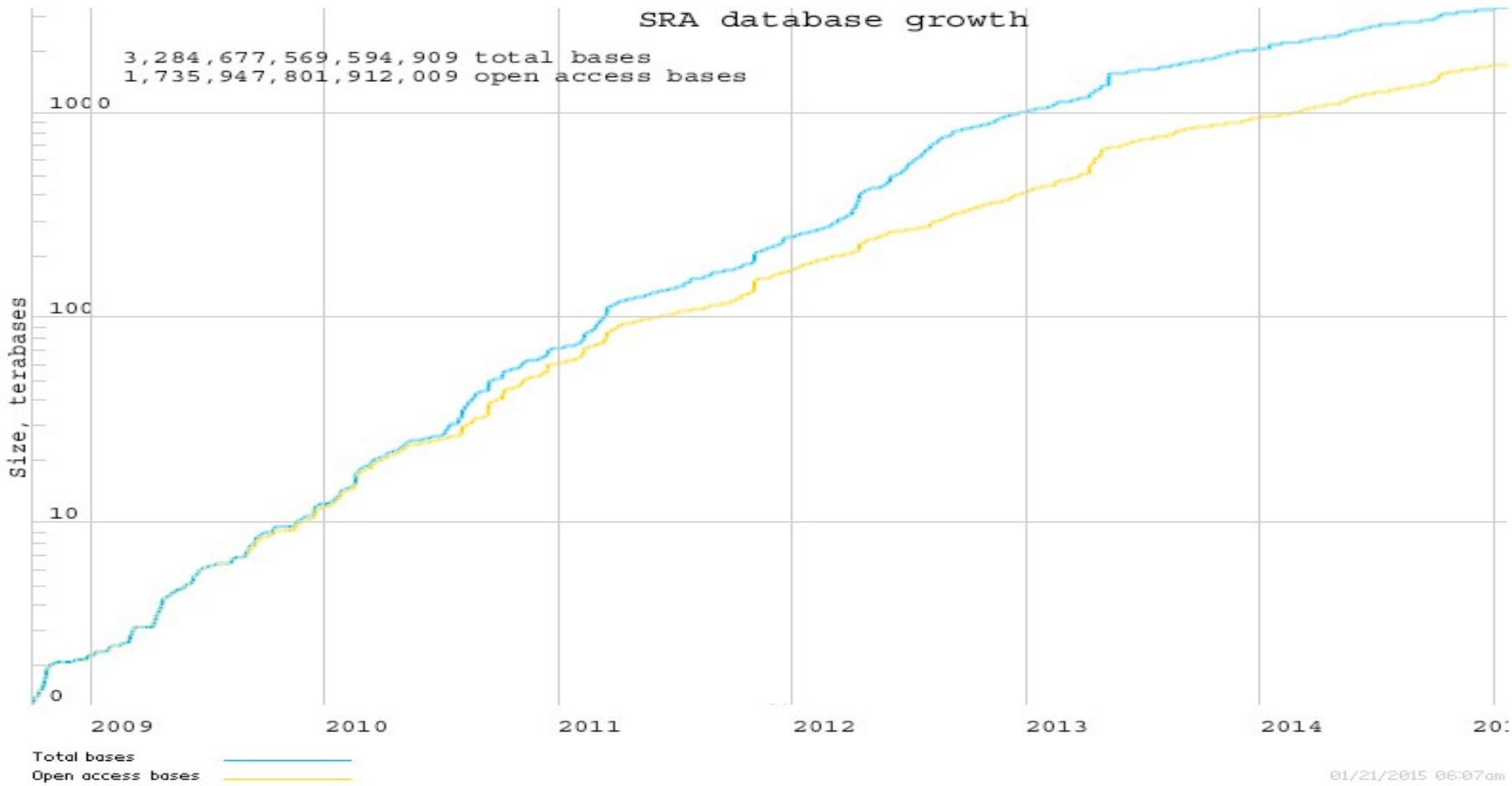
<u>Output:</u>

GTAAGGTCC

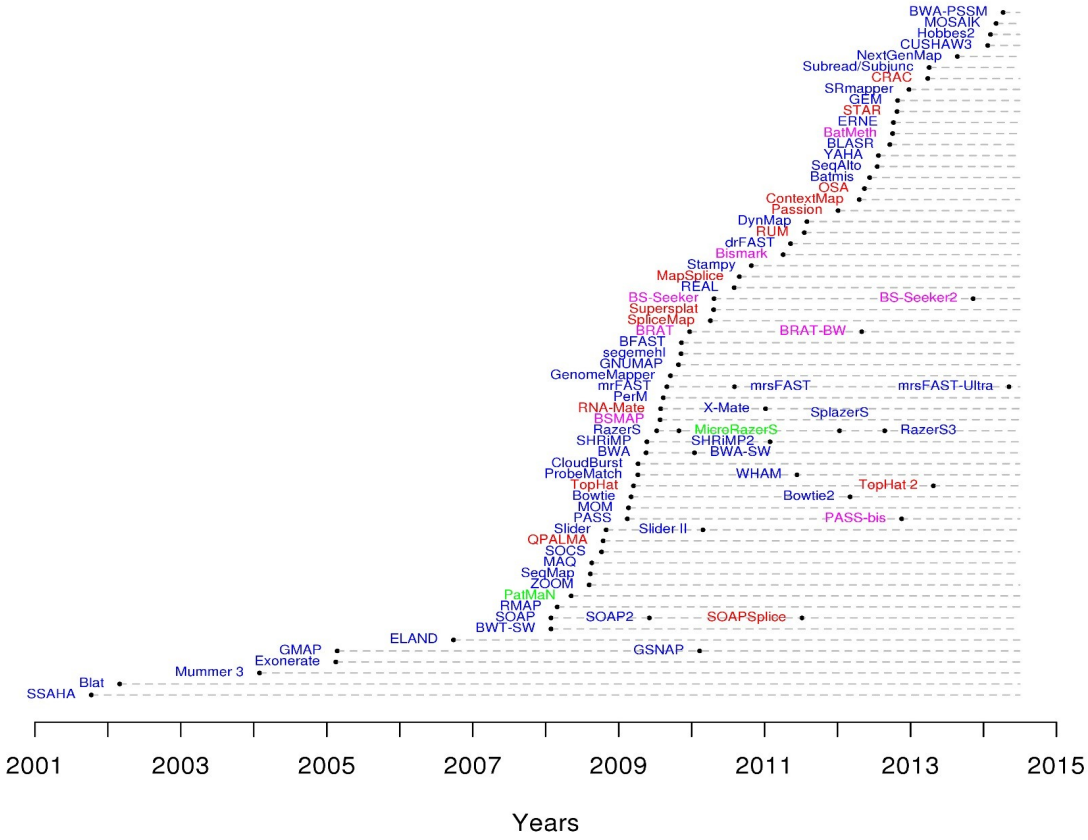
GTTAGGTCC



Next Generation Sequencing
More sequences

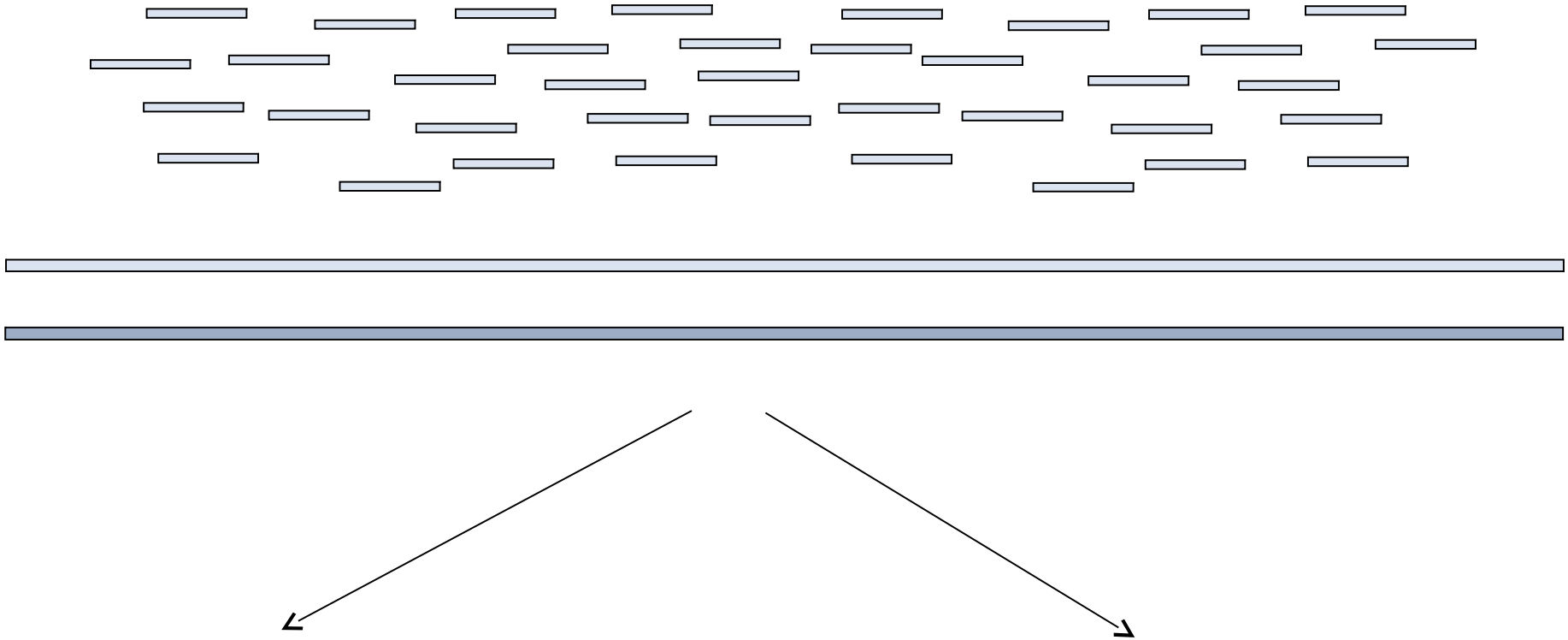


More aligners



http://www.ebi.ac.uk

Alignment of short reads to a reference



..ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA.. Reference

..ACTGGGTCATCGTACGATCGATAGATCGATCGATCGCTAGCTAGCTA.. Sample

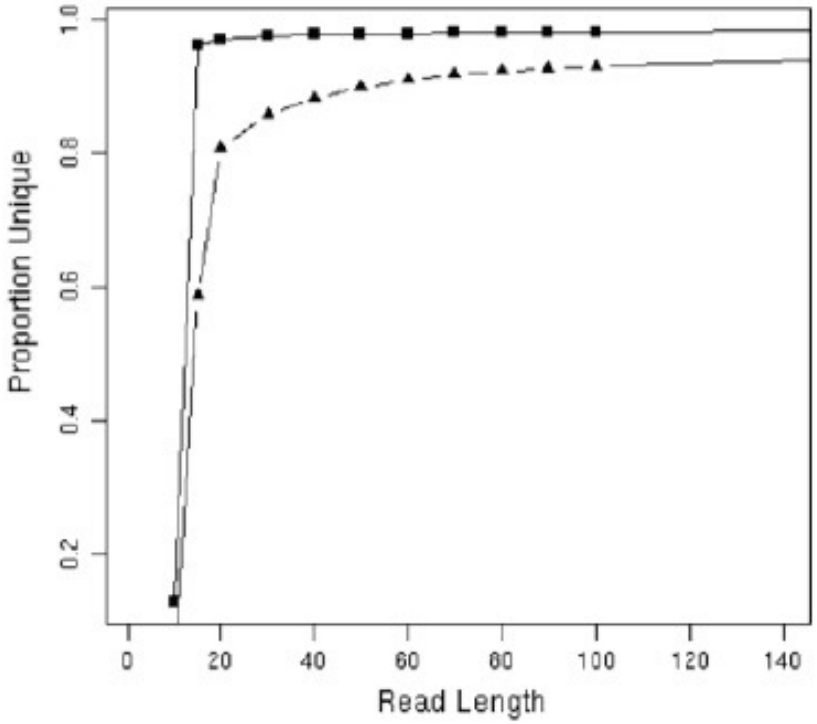
Why not using BLAST – Speed

- Typically BLAST will take approximately 0.1 – 1 second to search 1 sequence against a database
- Depends on size of database, e-value cutoff and number of hits to report selected
- 60 million reads equates to 70 CPU days!
- Even on multi-core systems this is too long!
- Especially if you have multiple samples!
- This is still true with new implementations of BLAST

Why is short read alignment hard?

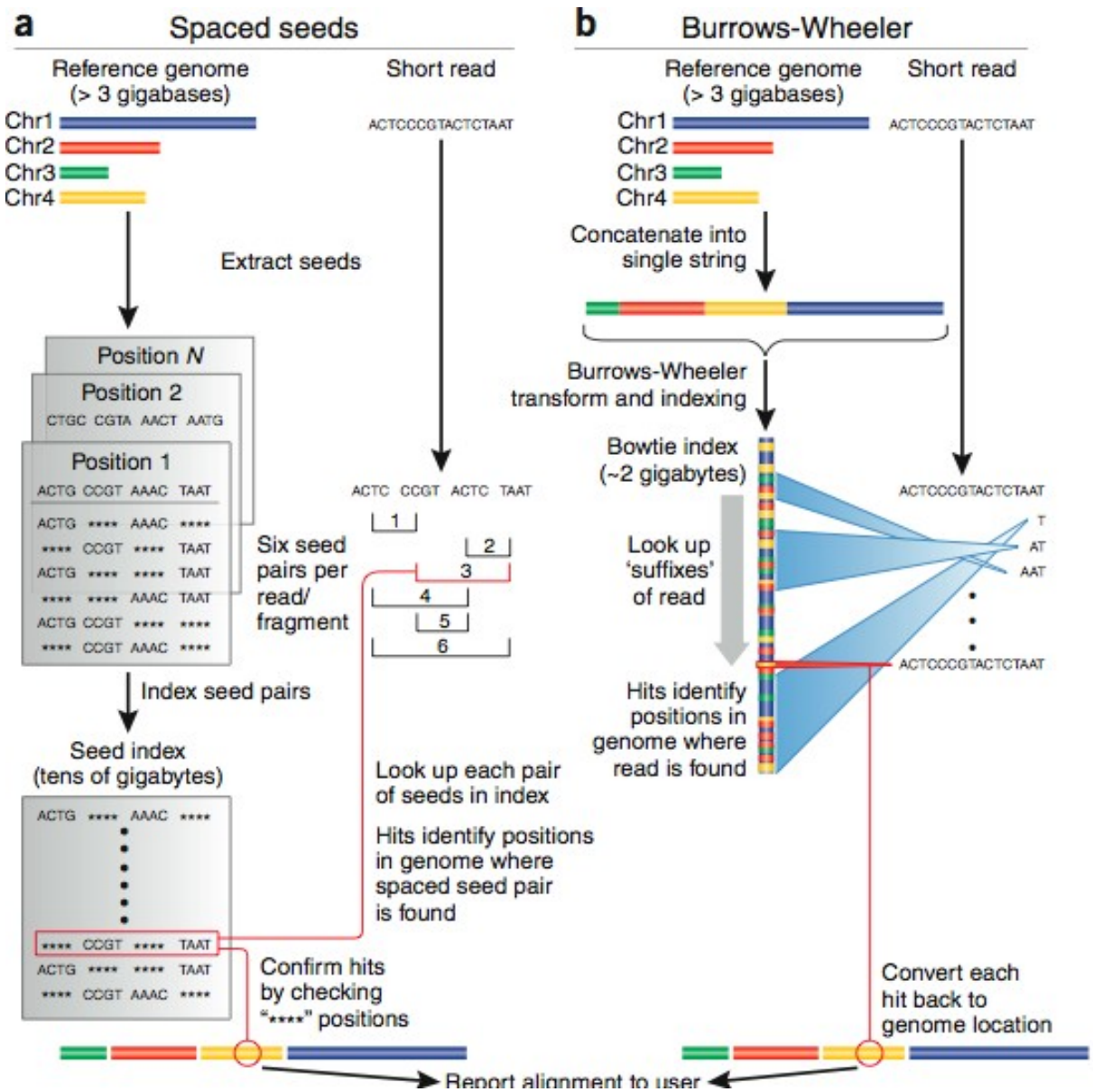
Many sequences should be aligned quickly to a very long reference

Short reads contain inherently limited information and the shorter a read, the less likely it is to have a unique match to a reference sequence



Turner et al. 2009

Approaches to align short reads



Trapnell & Salzberg 2009

Hashed seed-extend algorithms

2 step process

- Identify a match to the seed sequence in the reference
- Extend match using sensitive (but slow) Smith-Waterman algorithm (dynamic programming)

Seed-extend algorithm

Reference sequence:

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

Short read:

GTCATCGTACGATCGATAGATCGATCGATCGGCTA

Note that the short read has 1 difference from the reference

Seed-extend algorithm

Reference sequence:

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

Short read:

GTCATCGTACG ATCGATAGATCG ATCGATCGGCTA

11bp word 11bp word 11bp word

The algorithm will try to match each word to the reference. If there is a match at with any single word it will perform a local alignment to extend the match

Seed-extend algorithm

Reference sequence:

Seed

Extend with Smith Waterman →

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

GTCATCGTACG ATCGAACGATCGATCGATCGGCTA

Short read:

GTCATCGTACG ATCGATAGATCG ATCGATCGGCTA

Here the algorithm is able to match the short read with a word length of 11bp

Seed-extend algorithm

Reference sequence:

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

Short read:

GTCATCGTACGATCGATCGATCGATCGATCGGCAA

Note that the short read has 3 differences
possibly sequencing errors, possibly SNPs

Seed-extend algorithm

Reference sequence:

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

Short read:

GTCATCGTACG
11bp word

ATCGATCGATCG
11bp word

ATCGATCGGCAA
11bp word

Note that the short read has 3 differences

Seed-extend algorithm

Reference sequence:

...ACTGGGTCATCGTACGATCGATCGATCGATCGATCGGCTAGCTAGCTA...

Short read:

GTCATCGTACG ATCGATCGATCG ATCGATCGGCAA

No seeds match

Therefore the algorithm would find no hits at all!

Consecutive seed

Consecutive seed 9bp with no mismatches:

ACTCCCATCGTCATCGTACTAGGGATCGTAACA Reference sequence

CCACTGTCCTCCTACATAGGAACGA SNP ‘heavy’ read

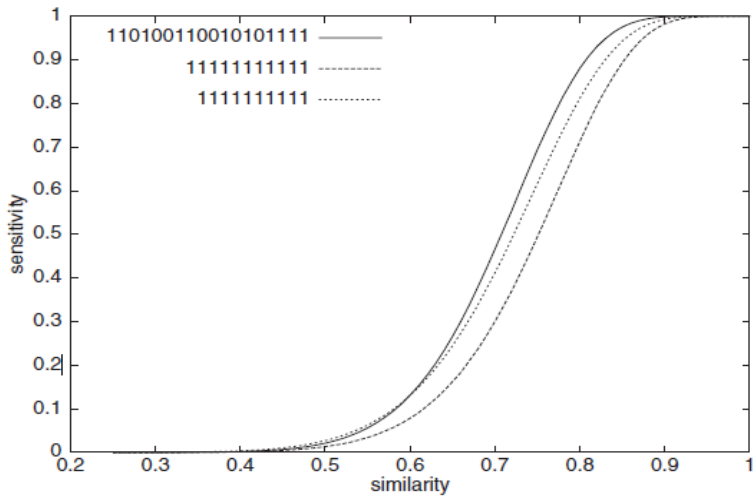
TC**A**TC**G**TAC

TC**C**TC**C**TAC Cannot find seed match

Even allowing for 2 mismatches
in the seed - no seeds match.
No hits!

Spaced seeds

To increase sensitivity we can used spaced-seeds:



Up to 55% more sensitive than BLAST's default template for two sequences of 70% similarity

Ma et al. 2002

```
11111111111
ACTATCATCGTACACAT
TCATCGTAC
```

Consecutive seed template with *length* 9bp

Reference

Query

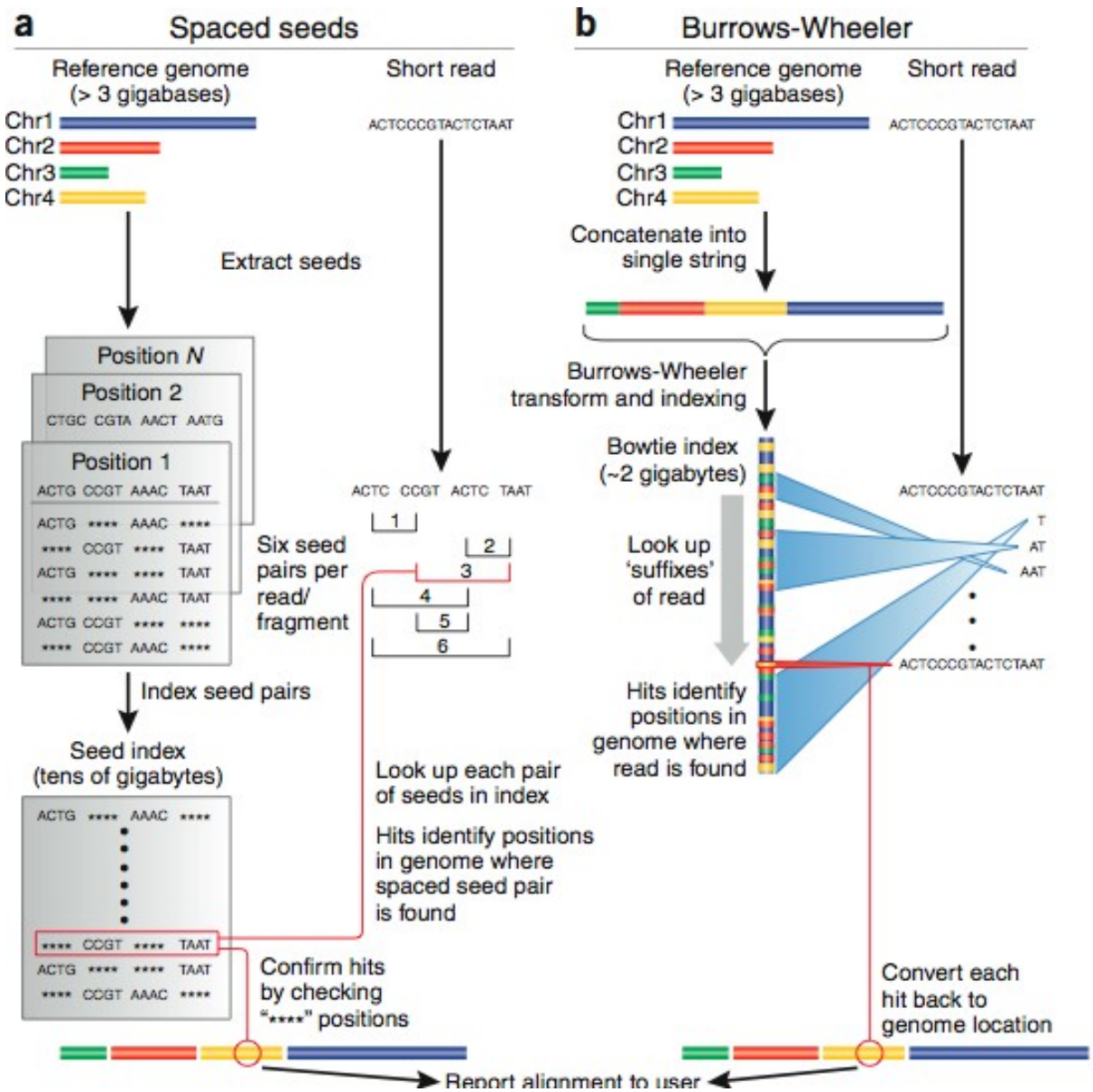
```
1100110011001 1001
ACTATCATCGTACACAT
ACTCTCACCGTACACAT
```

Spaced-seed template with *weight* 9bp

Reference

Query

Approaches to align short reads



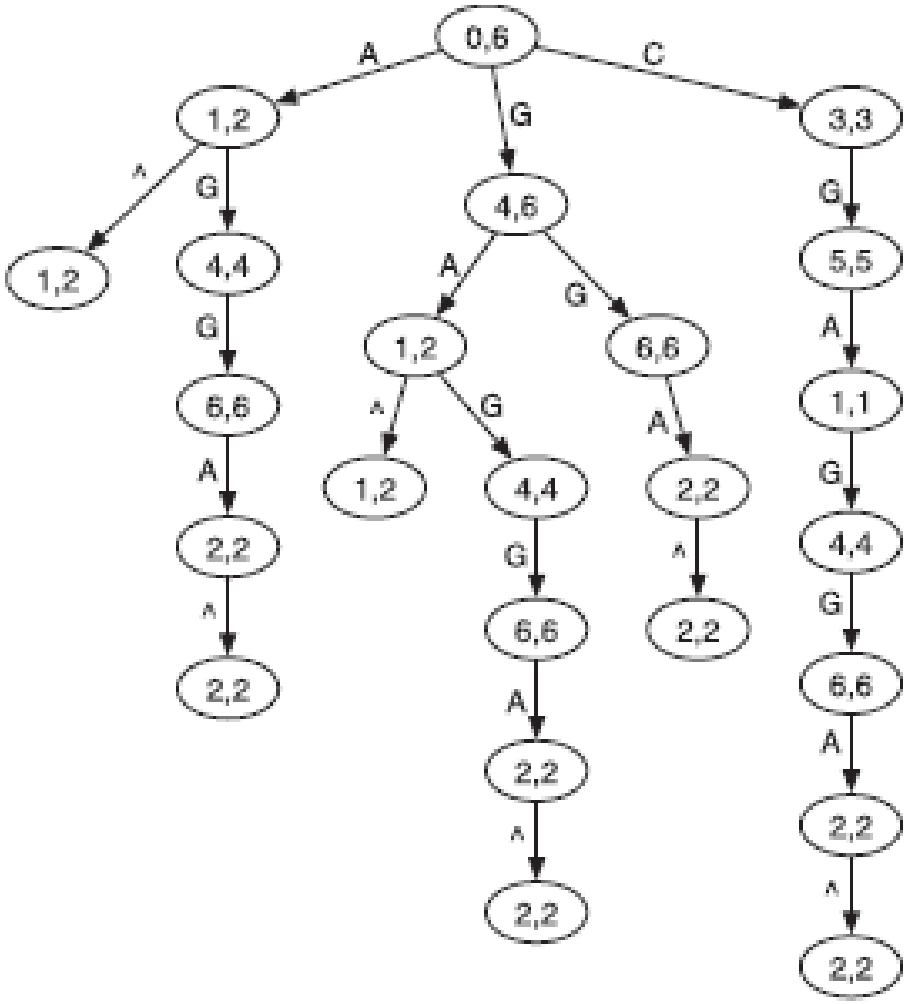
Trapnell & Salzberg 2009

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

Suffix Trie

CGAGG^A
CGAG^GA
CGA^GGA
CG^AGGA
C^GAGGA
^CGAGGA



Li &Homer 2010

Burrows-Wheeler Algorithm

- Encodes data so that it is easier to compress
- Burrows-Wheeler transform of the word BANANA
- Can later 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| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

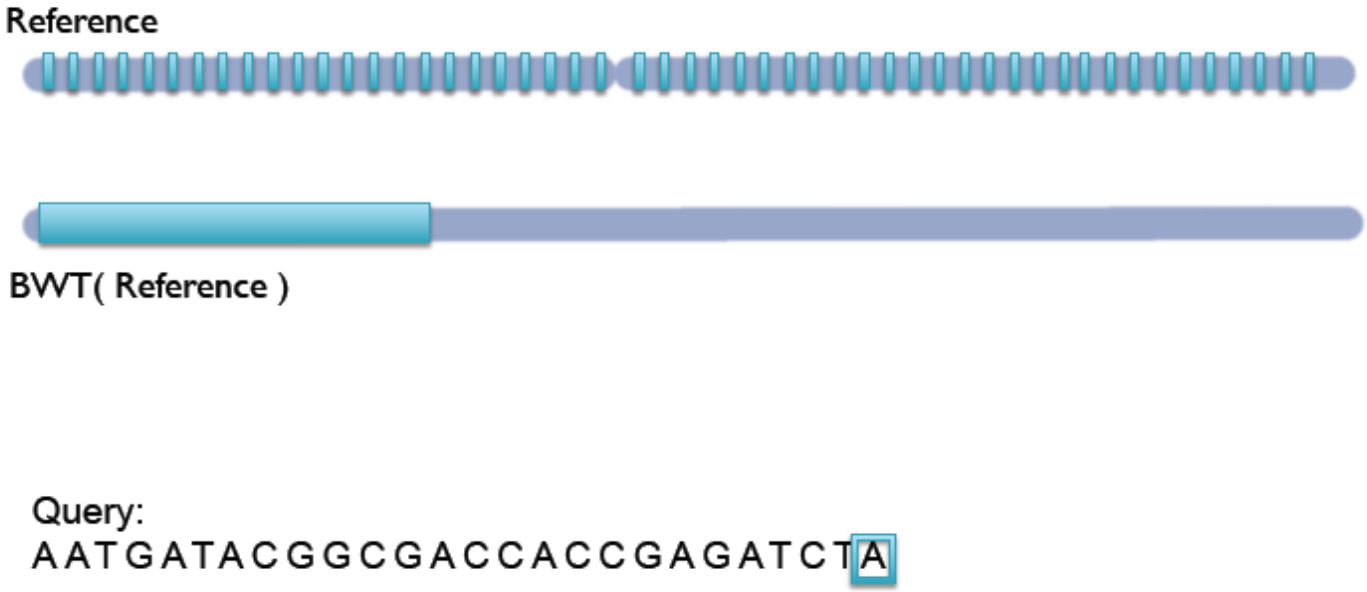
Bowtie/Soap2 example

Reference

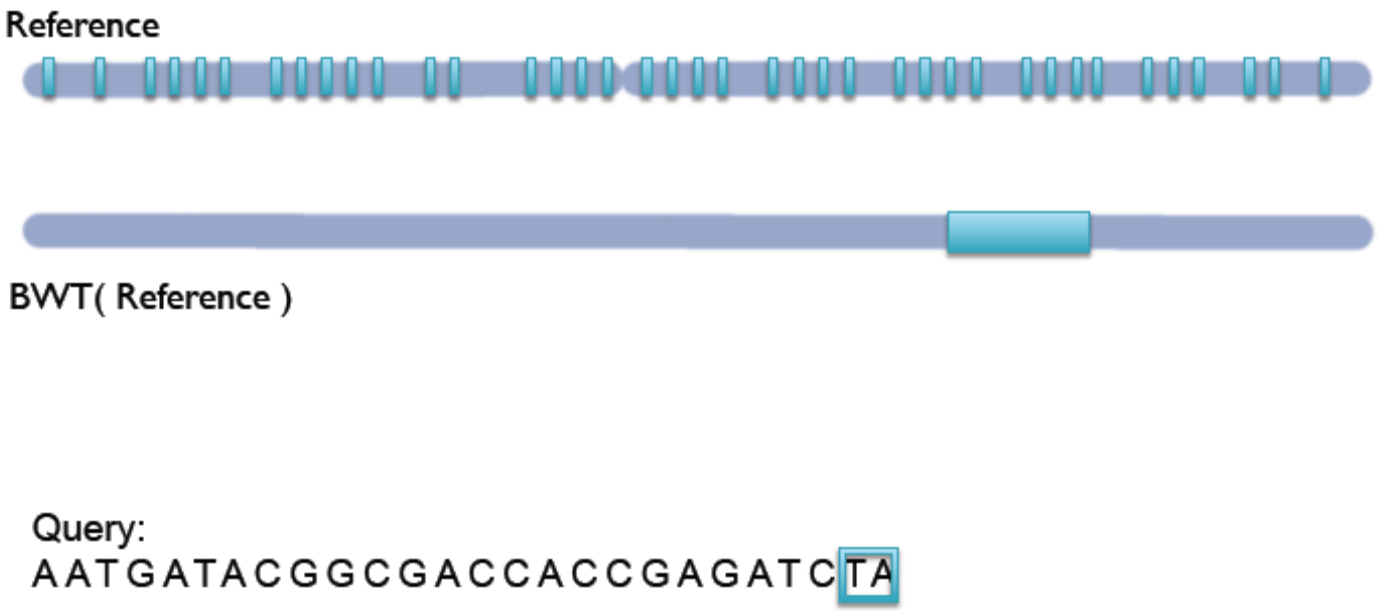
BWT(Reference)

Query:
AATGATACGGCGACCACCGAGATCTA

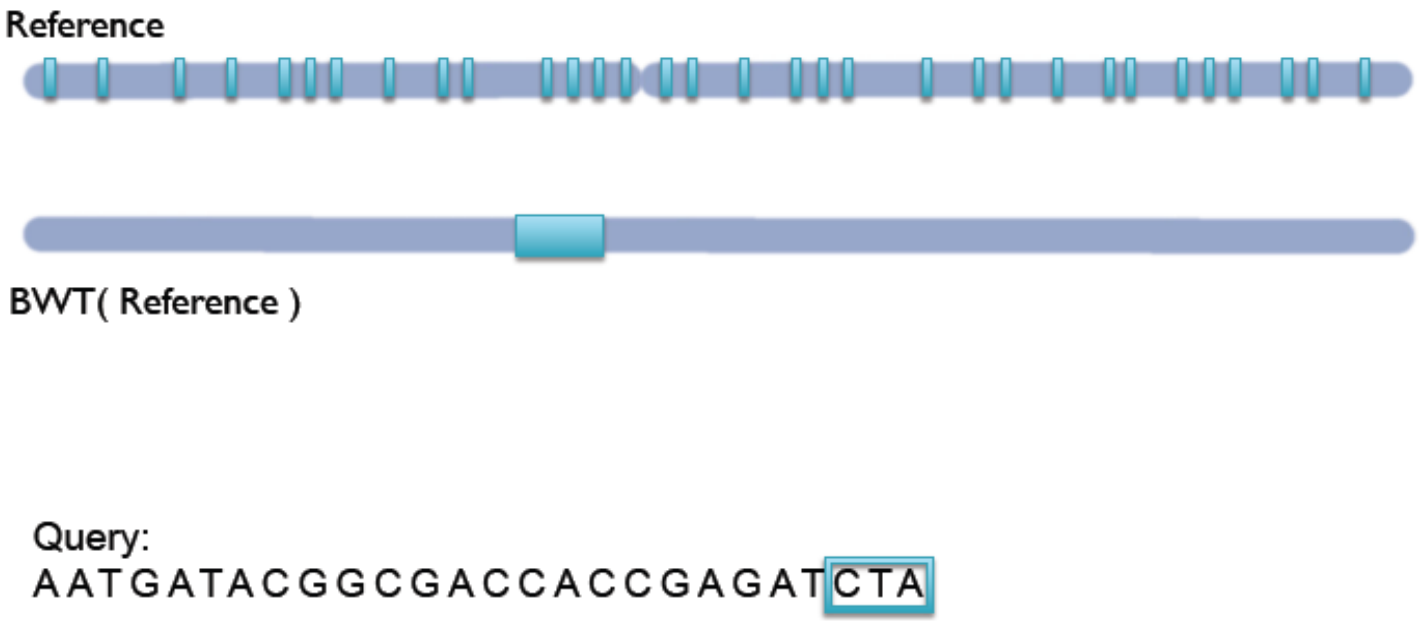
Bowtie/Soap2 example



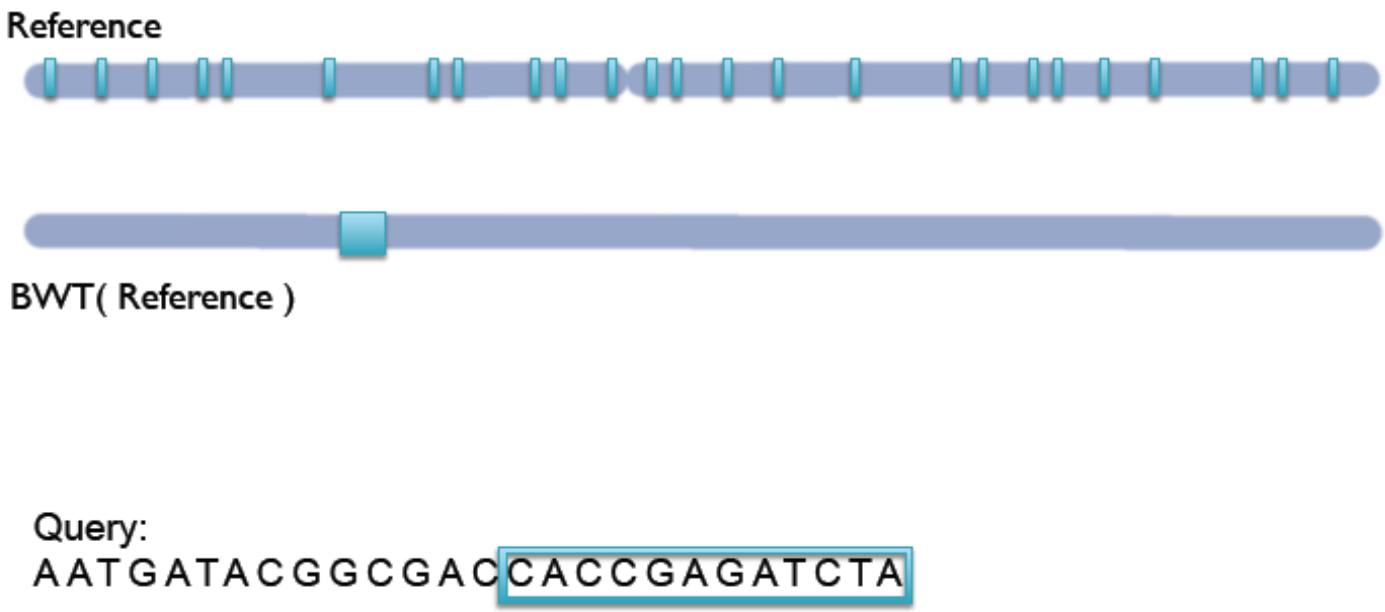
Bowtie/Soap2 example



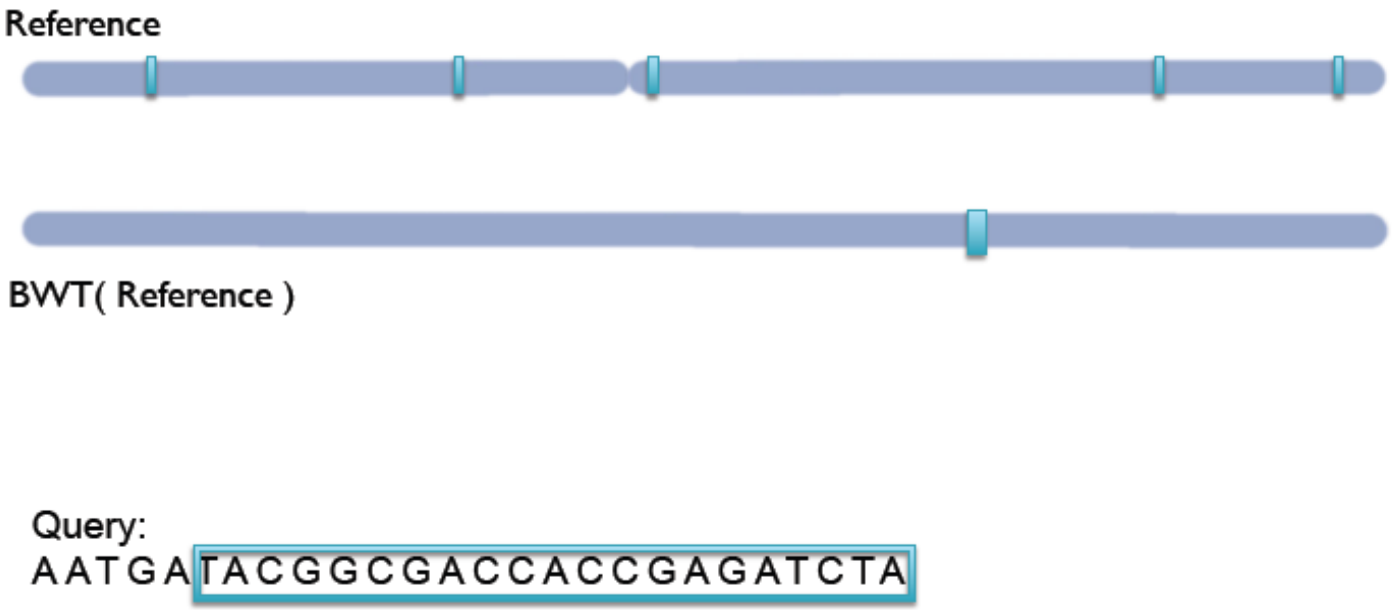
Bowtie/Soap2 example



Bowtie/Soap2 example



Bowtie/Soap2 example



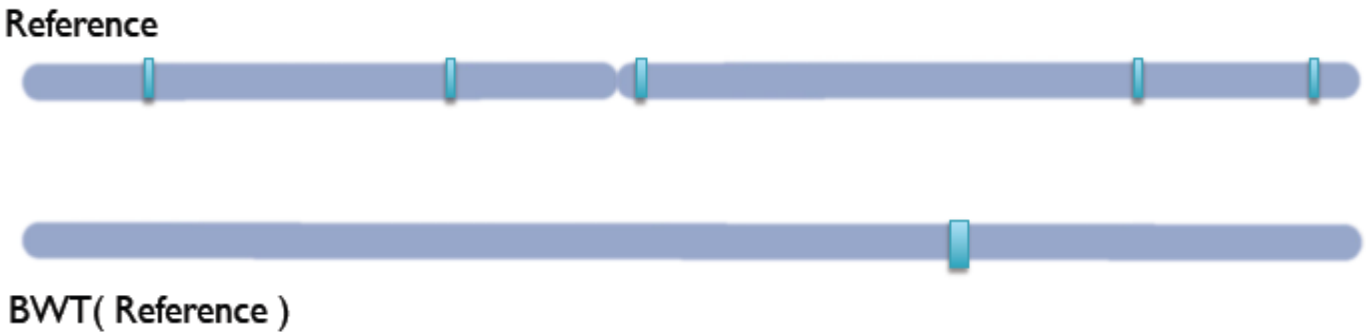
Bowtie/Soap2 example

Reference

BWT(Reference)

Query:
AATGATACGGCGACCACCGAGATCTA

Bowtie/Soap2 example



Query:
AATGTTACGGCGACCACCGAGATCTA

Bowtie/Soap2 example

Query:

AATGTTACGGCGACCACCGAGATCTA

Bowtie/Soap2 vs. BWA

Bowtie (not Bowtie2) and Soap2 cannot handle gapped alignments
- no indel detection => Many false SNP calls

BWA:

```
ACTCCCATTGTCATCGTACTTGGGATCGTAACA   Reference
        CCATTGTCATCGTACTTGGGATC-TA
                TCATCGTACTTGGGATC-TA
                        TTGGGATC-TA
```

Comparison

Hash referenced spaced seeds

- Requires ~50Gb of memory
- Runs 30-fold slower
- Simpler to program
- More sensitive

Suffix/Prefix Trie

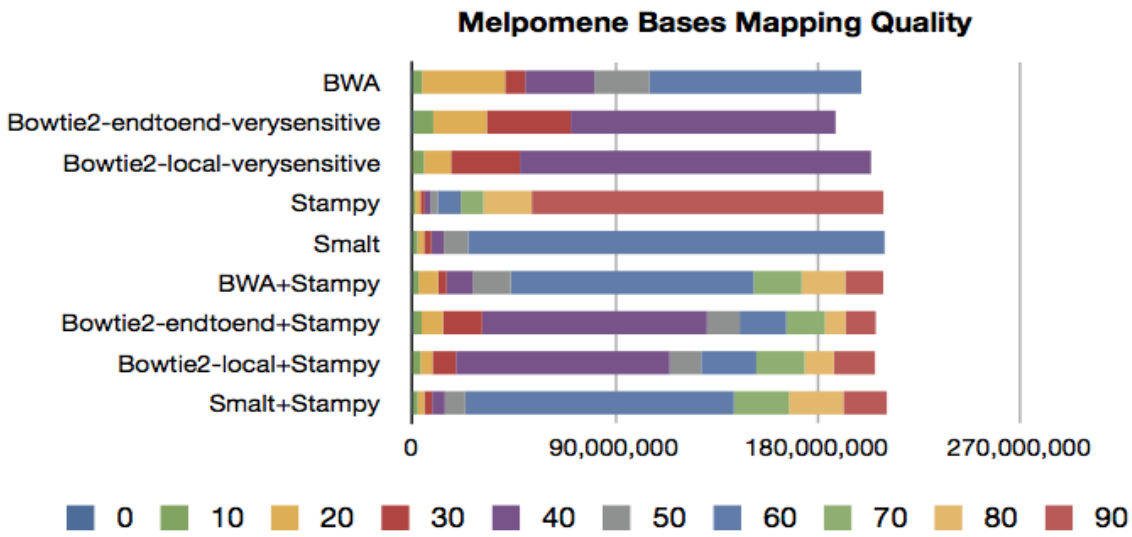
- Requires <2Gb of memory
- Runs 30-fold faster
- Complicated to program
- Less sensitive

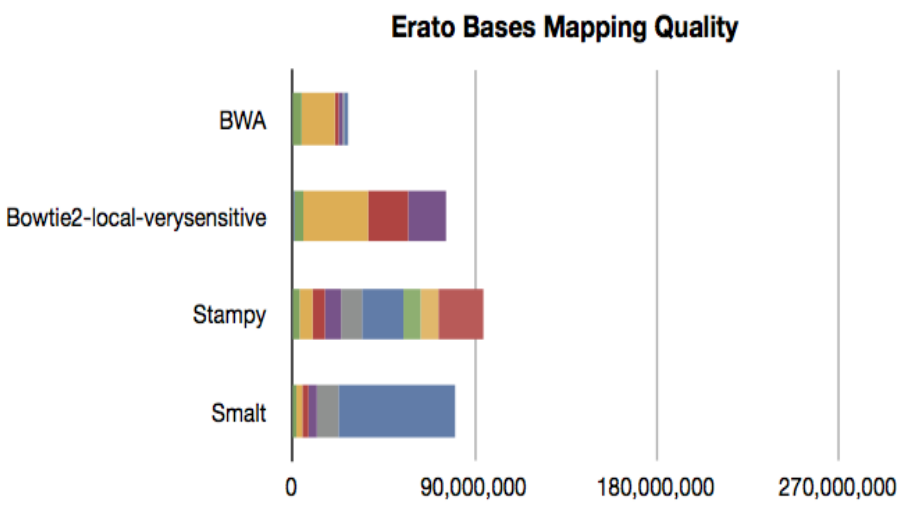
Open-source short read alignment programs

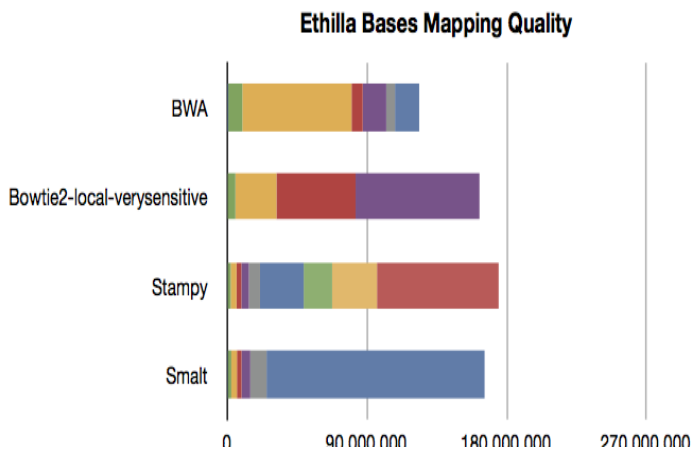
Program	Algorithm	SoLID	Long reads	Gapped alignment	Paired-end	Quality scores used?
Bfast	Hashing ref	Yes	No	Yes	Yes	No
Bowtie2	FM-Index	Yes	No	Yes	Yes	Yes
Blat	Hashing ref	No	Yes	Yes	No	No
BWA	FM-Index	Yes	Yes	Yes	Yes	Yes
MAQ	Hashing reads	Yes	No	Yes	Yes	Yes
STAMPY	Hashing ref	No	Yes	Yes	Yes	Yes
SMALT	Hashing ref	No	Yes	Yes	Yes	Yes
Shrimp2	Hashing ref	Yes	Yes	Yes	Yes	Yes
SOAP2	FM-Index	No	No	Yes	Yes	Yes
SSAHA2	Hashing ref.	No	No	No	Yes	Yes

Adjusted from Li & Homer 2010

Comparing aligners







John Davey: http://www.heliconius.org/

Other alignment considerations

- Indel detection
- Effect of paired-end alignments
- Using base quality to inform alignments
- PCR duplicates
- Multi-mapping reads
- Aligning spliced-reads from RNA-seq experiments

Indel detection

Spaced seed with weight 9bp and no mismatches:

```
ACTCCCATTGTCATCGTACTTGGGATCGTAACA
      CCATTGTCATGTACTTGGGATCGT
```

Reference sequence

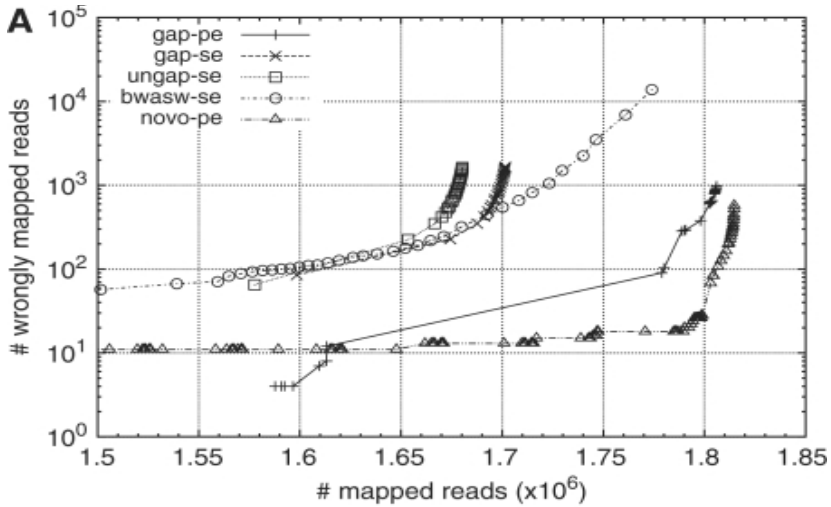
Read containing a deletion

```
CCATTGTCATCGTACAT
CCXXTGXXATXXACXXG
```

Seed not matched due to frame shift caused by gap

No seed match. No alignment!

Effect of paired-end alignments

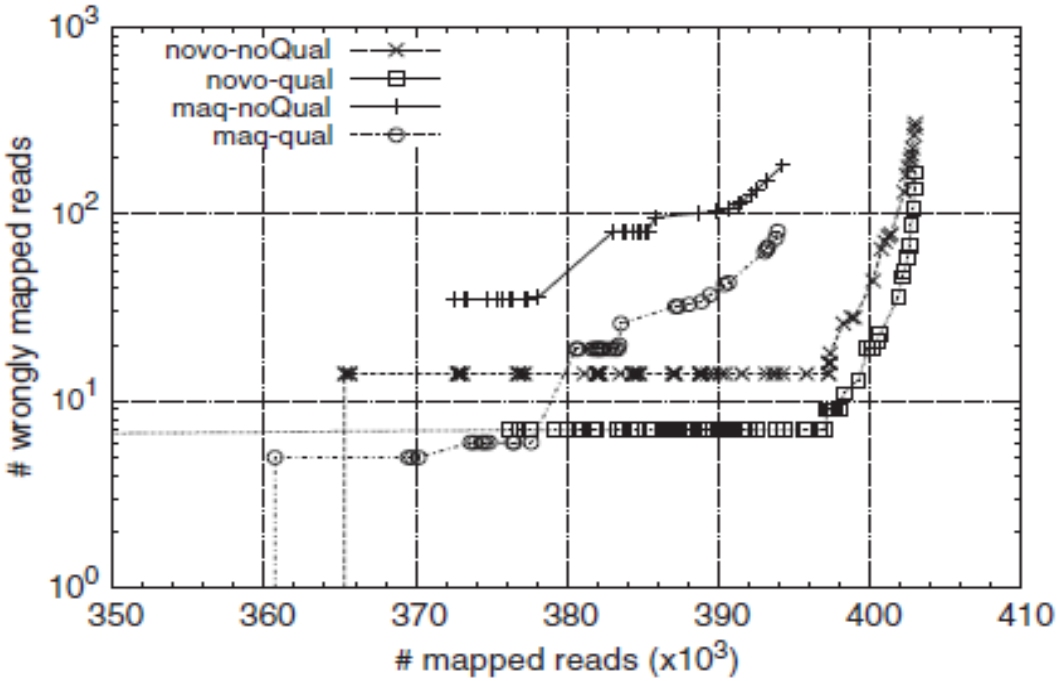


Li & Homer 2010

Paired read maps uniquely

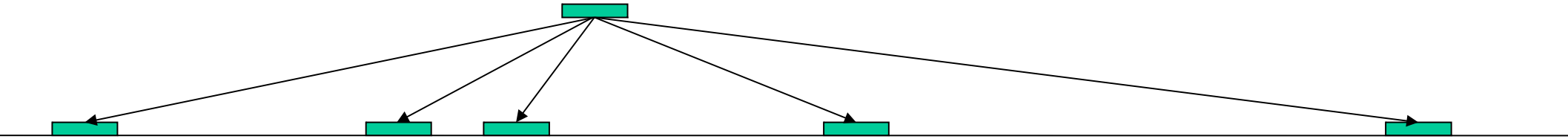
Repetitive sequence

Base quality impacts on read mapping



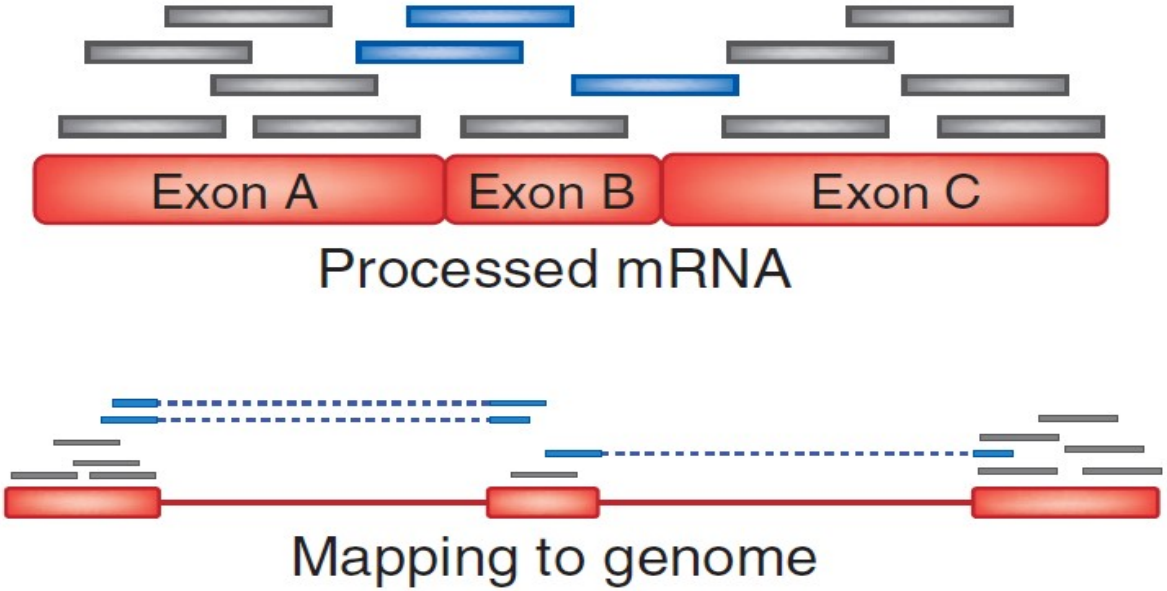
Li & Homer 2010

Multiple mapping reads



- A single read may occur more than once in the reference genome.
- This may be due to gene or whole chromosome duplication or repetitive sequences
- Aligners generally allow you to chose how these are dealt with
- Some aligners automatically assign a multi-mapping read to one of the locations at random (e.g. MAQ) – how random is random

Spliced-read mapping



- Need packages which can account for splice variants
- Examples: TopHat, SubRead, Star

PCR duplicates

2nd generation sequencers have at least one PCR amplification step

- Can result in duplicate DNA fragments
- This can bias SNP calls or introduce false SNPs
- Good libraries have < 2-3% of duplicates
- SAMtools and Picard can identify and remove these when aligned against a reference genome

SAM (BAM) Format

Sequence Alignment/Map format

- Universal standard
- Human-readable (SAM) and compact (BAM) forms

Structure

- Header

  version, sort order, reference sequences, read groups, program/processing history
- Alignment records

SAM format

Header lines

Sort order

```
@HD   VN:1.0  SO:coordinate
@SQ   SN:1    LN:249250621
@SQ   SN:2    LN:243199373
@SQ   SN:3    LN:198022430
@RG   ID:UM0098:1     PL:ILLUMINA     PU:HWUSI-EAS1707 LB:80   DT:2010-05-05T20     SM:SD37743      CN:UMCORE
```

Reference sequence name and length

read groups information: library and sample

Alignment 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>]

sam/bam FLAG

Bit	Description
0x1 (1)	template having multiple segments in sequencing
0x2 (2)	each segment properly aligned according to the aligner
0x4 (4)	segment unmapped
0x8 (8)	next segment in the template unmapped
0x10 (16)	SEQ being reverse complemented
0x20 (32)	SEQ of the next segment in the template being reversed
0x40 (64)	the rst segment in the template
0x80 (128)	the last segment in the template
0x100 (256)	secondary alignment
0x200 (512)	not passing quality controls
0x400 (1024)	PCR or optical duplicate

147 =1+2+16+128

http://broadinstitute.github.io/picard/explain-flags.html

Calculating mapping quality

MapQ = Q_s = -10 $\log_{10}(P)$

P = probability that this mapping is NOT the correct one
How to calculate p?

bowtie

- 255 = unique mapping
- Descriptive: 2 locations - P = 0.5
  3 locations - P = 2/3
  4-9 locations - P = ¾
  ≥10 locations – P = 0
- MapQ = -10 log10(0.5) = 3
  -10 log10(2/3) = 1.76 (rounds to 2)
  -10 log10(3/4) = 1.25 (rounds to 1)

Bwa (same as in MAQ)

- 255 = MQ is not available

Z = read sequence
X = reference sequence
q = sum of qualities for mismatches

$$p_s(u|x,z) = \frac{p(z|x,u)}{\sum_{v=1}^{L-l+1} p(z|x,v)}$$

$$Q_s = \min\{q_2 - q_1 - 4.343 \log n_2, 4 + (3-k')(\bar{q} - 14) - 4.343 \log p_1(3-k', 28)\}.$$

Li et al. 2008

CIGAR string format

Option	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 clip on the read (clipped sequence present in <seq>)
H	Hard clip on the read (clipped sequence NOT present in <seq>)
P	Padding (silent deletion from the padded reference sequence)

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

Exercise

1) Generate fastq files with different levels of similarity to the reference:

- cp_350PE_01Err: sunflower chloroplast, 100bp paired-end with 350 (sd=30) insert-size, 1% error, 30x coverage, 0.01% mutations, 10% indels, 3 Ns allowed, min quality = 5
- cp_350PE_05Err: sunflower chloroplast, 100bp paired-end with 350 (sd=30) insert-size, 5% error, 30x coverage, 0.01% mutations, 10% indels, 3 Ns allowed, min quality = 5
- cp_350PE_03INDL: sunflower chloroplast, 100bp paired-end with 350 (sd=30) insert-size, 1% error, 30x coverage, 0.01% mutations, 30% indels, 3 Ns allowed, min quality = 5

2) Align to reference (advanced users can change default parameters or use another aligner

- Align with bwa mem/bowtie2
- How much time did it take?
- How many reads were aligned?
- How many correctly paired?
- View
- Calculate depth
- How many mismaches/Indels?