Melbourne Bioinformatics

BIOINFORMATICS + DATA SERVICES + INFRASTRUCTURE, FOR LIFE SCIENCES TODAY

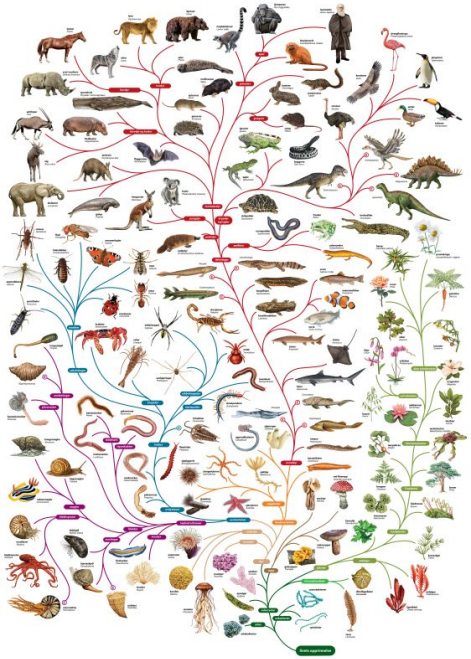
THE UNIVERSITY OF MELBOURNE

COMP90014

Algorithms for Bioinformatics

Week 2B - Sequence Alignment and Mapping II

Sequence Alignment and Mapping II



Local, Semi-Global alignment (previous lecture)

Scoring/Substitution Matrices

Gap penalties

Read mapping

- Seed-extend (short reads)
- Seed-chain-align (long reads)

BLAST

Academic Integrity Statement
DUE TOMORROW

Pairwise Alignment

Levenshtein distance

Forms the basis for the remainder of lecture

<u>Global alignment</u>
Local alignment
Semi-global alignment

These add a few important variations:

Penalties rather than adding edits
Penalties are different for mismatch vs gap
The arrows are stored
(directions we took to calculate each cell)

`SPLATTERING -> PATTERN`

Global	`SPLATTERING` `-P-ATTERN--`
Local	`ATTER` `ATTER`
Semi-global	`PLATTERIN` `P-ATTER-N`

Global Alignment

Algorithm: Needleman Wunsch

```
GCACTGA-
GC-CTGAT
```

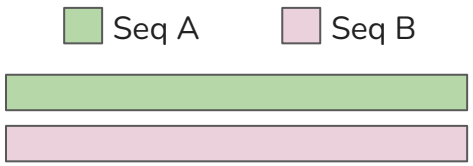
Algorithm 2: Needleman-Wunsch(A, B)

$g \leftarrow GapPenalty;$
for $i = 0$ **to** $length(A)$ **do**
  $S(i, 0) \leftarrow g \times i;$

for $j = 0$ **to** $length(B)$ **do**
  $S(0, j) \leftarrow g \times j;$

for $i = 1$ **to** $length(A)$ **do**
  for $j = 1$ **to** $length(B)$ **do**
    $Match \leftarrow S(i-1, j-1) + Scoring(A_i, B_j);$
    $Insert \leftarrow S(i, j-1) + g;$
    $Delete \leftarrow S(i-1, j) + g;$
    $S(i, j) \leftarrow max(Match, Insert, Delete);$

return $S(length(A), length(B))$



Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	-2	-4	-6	-8	-10	-12	-14
G	-2	1	-1	-3	-5	-7	-9	-11
C	-4	-1	2	0	-2	-4	-6	-8
C	-6	-3	0	1	1	-1	-3	-5
T	-8	-5	-2	-1	0	2	0	-2
G	-10	-7	-4	-3	-2	0	3	1
A	-12	-9	-6	-3	-4	-2	1	4
T	-14	-11	-8	-5	-4	-3	-1	2

del ▶

ins ▶

Semi-global Alignment

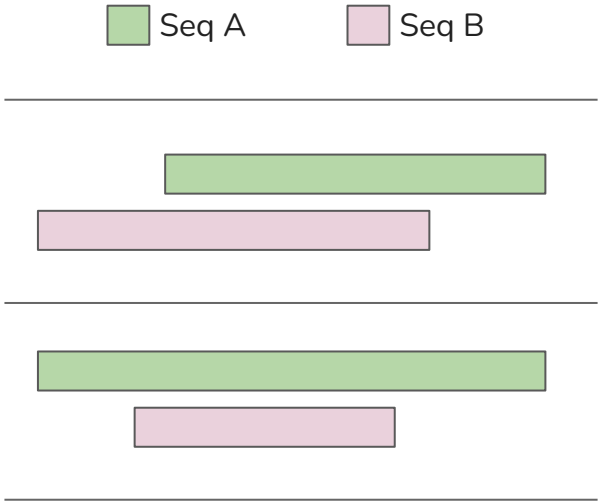
Alignment of complete sequences, where offset is not penalised

All of Seq A
All of Seq B

Best when sequences are expected to have similar overlapping region

eg. Read overlaps in OLC assembly

Returns the full alignment, clipped by best offset



Semi-global Alignment

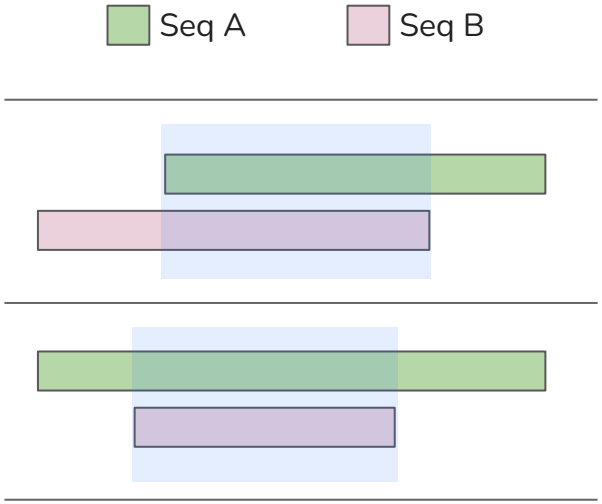
Alignment of complete sequences, where offset is not penalised

All of Seq A
All of Seq B

Best when sequences are expected to have similar overlapping region

eg. Read overlaps in OLC assembly

Returns the full alignment, clipped by best offset



Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

Seq A

		start	G	C	A	C	T	G	A
		start							
		G							
	del ▶	C							
Seq B		C							
		T							
		G							
		A							
	ins ▶	T							

Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column: Why?

Seq A

Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							
T	0							
G	0							
A	0							
T	0							

del ▶

ins ▶

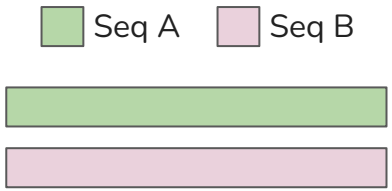
Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column: Why?



Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							
T	0							
G	0							
A	0							
T	0							

del ▶

ins ▶

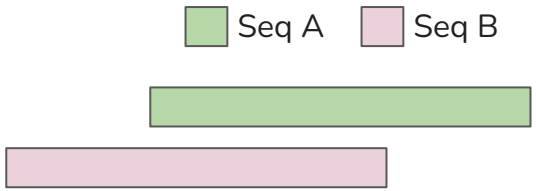
Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column: Why?



Seq A

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							
T	0							
G	0							
A	0							
T	0							

Seq B

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

<u>-> no offset penalties for top row, left column: Why?</u>

Seq A

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							
T	0							
G	0							
A	0							
T	0							

Seq B

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column: Why?

Do not want to penalise beginning shifts

Seq A

		start	G	C	A	C	T	G	A
	start	0	0	0	0	0	0	0	0
	G	0							
	C	0							
del ▶	C	0							
Seq B	T	0							
	G	0							
	A	0							
ins ▶	T	0							

Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

Seq A

		start	G	C	A	C	T	G	A
	start	0	0	0	0	0	0	0	0
	G	0							
	C	0							
del ▶	C	0							
Seq B	T	0							
	G	0							
	A	0							
ins ▶	T	0							

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> <u>the return score</u>:

Max(bottom row), or *<u>Max(right column)</u>*

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

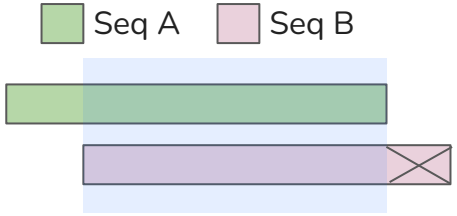
Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> <u>the return score</u>:

Max(bottom row), or *<u>Max(right column)</u>*



Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> the return score:

Max(bottom row), or *Max(right column)*

-> <u>the backtracking:</u>

Starts at max cell

Ends when hit top row, or left column

```
GCACTGA
GC-CTGA
```

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
del ▶ C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
ins ▶ T	0	-1	-2	-1	-1	-1	-1	2

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> the return score:

Max(bottom row), or *Max(right column)*

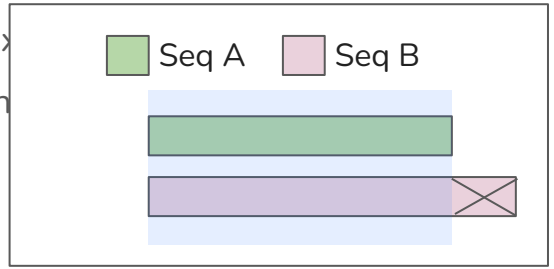
-> <u>the backtracking:</u>

Starts at max

Ends when h

GCACTGA

GC-CTGA



Example to the right

Seq A

Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> the return score:

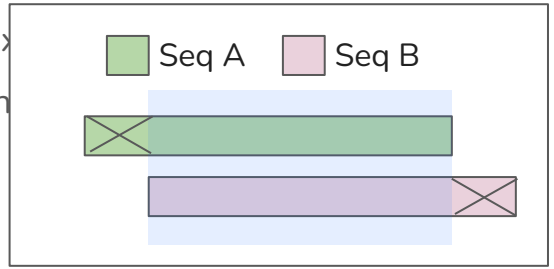
Max(bottom row), or *Max(right column)*

-> <u>the backtracking:</u>

Starts at max

Ends when h

GCACTGA
GC-CTGA



Hit top row

Seq A

Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

Semi-global Alignment

Scoring				(Gaps: -2)
	C	T	A	G
C	+1	-1	-1	-1
T	-1	+1	-1	-1
A	-1	-1	+1	-1
G	-1	-1	-1	+1

Algorithm: Needleman Wunsch (variant)

Same as Needleman-Wunsch except:

-> no offset penalties for top row, left column

-> the return score:

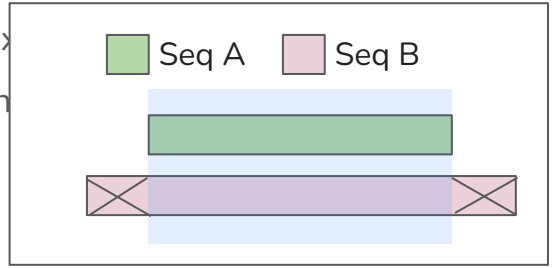
Max(bottom row), or *Max(right column)*

-> <u>the backtracking:</u>

Starts at max

Ends when h

GCACTGA
GC-CTGA



Hit first column

Seq A

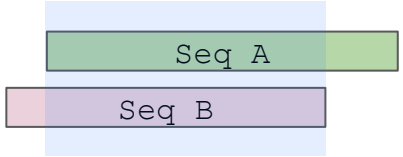
Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	1	-1	-1	-1	-1	1	-1
C	0	-1	2	0	0	-1	-1	0
C	0	-1	0	1	1	-1	-2	-2
T	0	-1	-2	-1	0	2	0	-2
G	0	1	-1	-3	-1	0	3	1
A	0	-1	0	0	-2	-2	1	4
T	0	-1	-2	-1	-1	-1	-1	2

del ▶

ins ▶

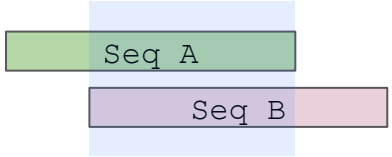
Semi-global Alignment



Seq A

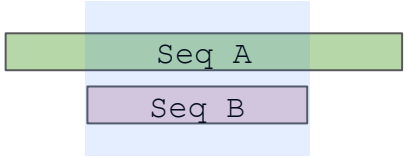
Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							
T	0							
G	0							
A	0							
T	0					max		



Seq A

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0							
C	0							
C	0							max
T	0							
G	0							
A	0							
T	0							



Seq A

	start	G	C	A	C	T	G	A	T	A
start	0	0	0	0	0	0	0	0	0	0
G	0									
C	0									
C	0									
T	0							max		

Local Alignment

Region of best local similarity

Some of Seq A
Some of Seq B

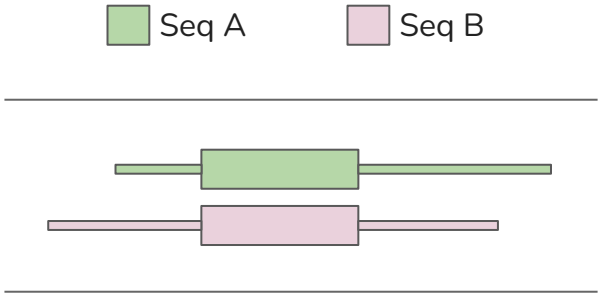
Best when sequences are dissimilar, but contain regions of similarity

eg. BLAST: gene homology

Best region dictated by penalty scores

Returns the alignment in highest scoring region

Algorithm: Smith Waterman



Gene homology between rat and human

Parts of the gene will be similar
(due to evolutionary viability)
(active sites, specific domains)

Parts of the gene will be dissimilar
(those which are more permissive to mutation)

Local Alignment

Region of best local similarity

Some of Seq A
Some of Seq B

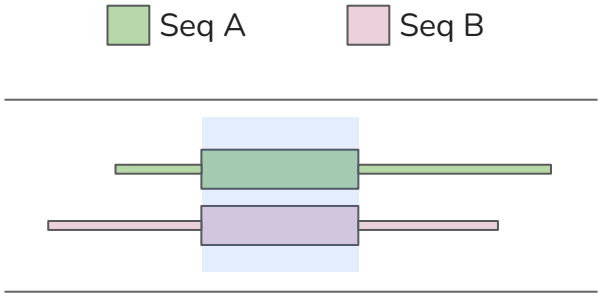
Best when sequences are dissimilar, but contain regions of similarity

eg. BLAST: gene homology

Best region dictated by penalty scores

Returns the alignment in highest scoring region

Algorithm: Smith Waterman



Gene homology between rat and human

Parts of the gene will be similar
(due to evolutionary viability)
(active sites, specific domains)

Parts of the gene will be dissimilar
(those which are more permissive to mutation)

Local Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> First row and first column set to 0

-> Negative score set to 0

-> the return score: *max(S)*

-> the backtracking:

Starts at *max(S)*

Ends when hit score of zero

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	2	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

del ▶

ins ▶

Local Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> <u>First row and first column set to 0</u>

-> Negative score set to 0

-> the return score: *max(S)*

-> the backtracking:

Starts at *max(S)*

Ends when hit score of zero

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	2	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

del ▶

ins ▶

Local Alignment

Scoring (**Gaps:** -2)

	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> First row and first column set to 0

-> <u>Negative score set to 0</u>

-> the return score: *max(S)*

-> the backtracking:

Starts at *max(S)*

Ends when hit score of zero

Seq A

Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	2	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

del ▶

ins ▶

Local Alignment

Scoring (**Gaps:** -2)

	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> First row and first column set to 0

-> Negative score set to 0

-> <u>the return score: *max(S)*</u>

-> the backtracking:

Starts at *max(S)*

Ends when hit score of zero

Seq A

Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	2	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

del ▶

ins ▶

Local Alignment

Scoring				(Gaps: -2)
	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> First row and first column set to 0

-> Negative score set to 0

-> the return score: *max(S)*

-> <u>the backtracking</u>:

Starts at *max(S)*

Ends when hit score of zero

GCACTGA
GC-CTGA

Seq A

Seq B	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	2	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

del ▶

ins ▶

Local Alignment

Scoring (**Gaps:** -2)

	C	**T**	**A**	**G**
C	+2	-1	-1	-1
T	-1	+2	-1	-1
A	-1	-1	+2	-1
G	-1	-1	-1	+2

Algorithm: Smith Waterman

Same as Needleman-Wunsch except:

-> First row and first column set to 0

-> Negative score set to 0

-> the return score: *max(S)*

-> <u>the backtracking</u>:

Starts at *max(S)*

Ends when hit score of zero

ACTGA
CCTGA

Seq A

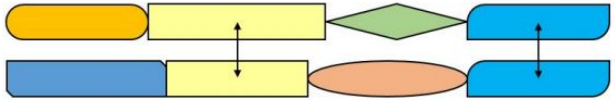
Seq B

	start	G	C	A	C	T	G	A
start	0	0	0	0	0	0	0	0
G	0	2	0	0	0	0	2	0
C	0	0	4	0	2	0	0	0
C	0	0	2	3	4	2	0	0
T	0	0	0	1	2	6	4	2
G	0	2	0	0	0	4	8	6
A	0	0	1	0	0	2	6	10
T	0	0	0	0	0	2	4	8

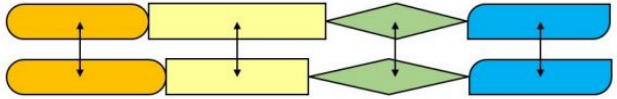
del ▶

ins ▶

Local *vs.* Global



Local Alignment



Global Alignment

	Smith–Waterman algorithm	Needleman-Wunsch algorithm
Initialization	First row and first column are set to 0	First row and first column are subject to gap penalty
Scoring	Negative score is set to 0	Score can be negative
Traceback	Begin with the highest score, end when 0 is encountered	Begin with the cell at the lower right of the matrix, end at top left cell
Complexity	$O(m \times n)$	$O(m \times n)$

Pairwise Alignment

Local Alignment

Finds region of best local similarity

O(m x n)

Global Alignment

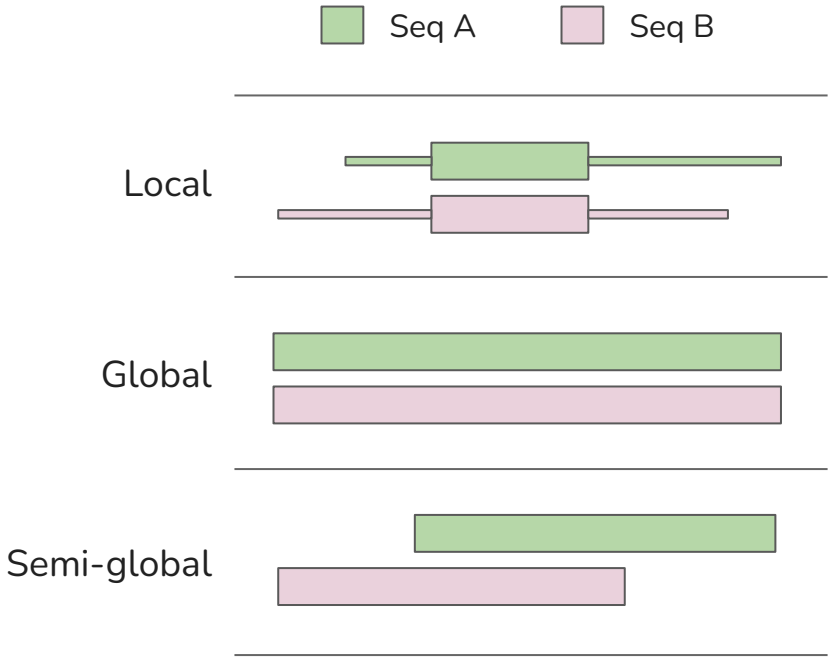
End-to-end alignment of two sequences

O(m x n)

Semi-Global Alignment

Alignment of complete sequences, where offset is not penalised

O(m x n)



Does this scale to large datasets?

Alignment: Scoring / Substitution Matrices

Scoring Alignments

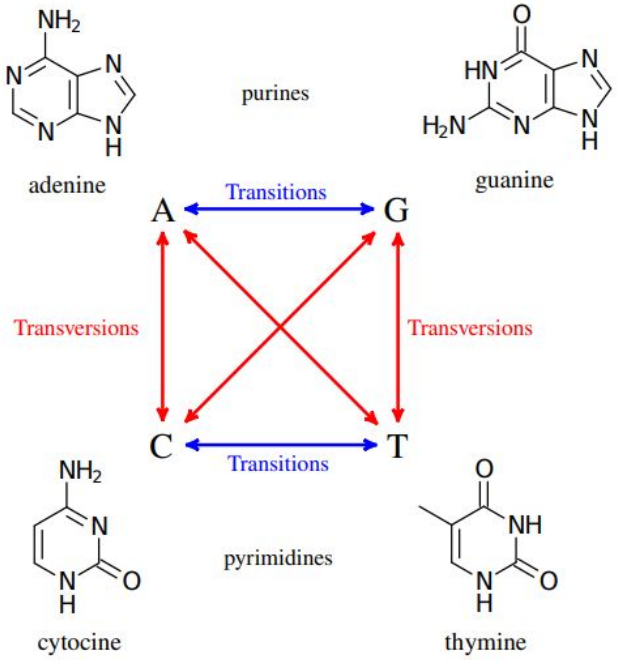
- The optimal alignment depends on our scoring system
  - Matches (reward)
  - Mismatches (penalty)
  - Starting/Extending Gaps (penalty)

- Simple match/mismatch score
  - `Matches:` `+1`
  - `Mismatches/Gaps:` `-1`

- Generalize to a substitution matrix
  - Assign a score to each pair of characters
  - `N×N` Symmetric matrix
    `N=4` Nucleic acids
    `N=20` Proteins
    `M(i,j)` cost/reward to change from i to j

	C	T	A	G
C	1	-1	-1	-1
T	-1	1	-1	-1
A	-1	-1	1	-1
G	-1	-1	-1	1

Why can't we just use a constant score for mismatches?

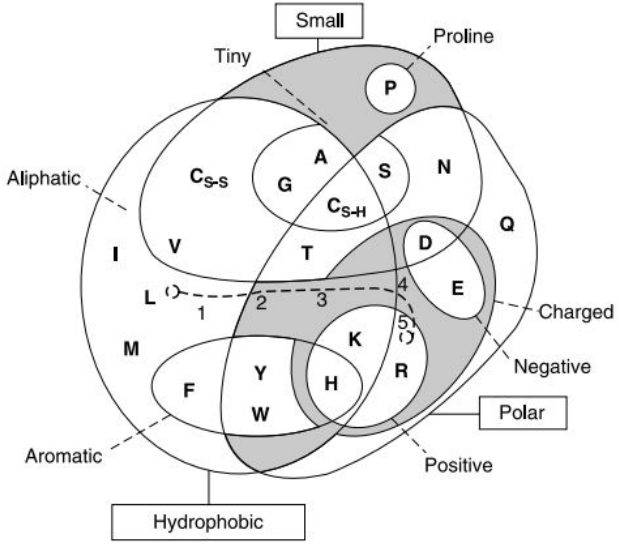


- two types of DNA substitution mutations:
  - transitions (bases with similar shape)
  - transversions (different number of rings)
- transition mutations occur more frequently
- transitions are less likely to result in amino acid substitutions (often synonyms)
- we can capture this in a substitution matrix

	C	T	A	G
C	2	1	-1	-1
T	1	2	-1	-1
A	-1	-1	2	1
G	-1	-1	1	2

Krishnavedala via Wikimedia Commons

Protein substitution matrix



- protein substitution matrices are more complex than DNA scoring matrices
- proteins are composed of 20 different amino acids
- varied physicochemical properties
  - compare a `D` to `E` with a `D` to `W`
- scoring matrices reflect:
  - chemical similarity
  - observed mutation frequencies
  - how protein sequences evolve

Betts & Russell, 2003. *Amino Acid Properties and Consequences of Substitutions*. 10.1002/0470867302.ch14.

Protein substitution matrices

How often is one amino acid substituted for another in related proteins?

PAM	BLOSUM
Point **A**ccepted **M**utation	**Blo**cks **Su**bstitution **M**atrix

PAM1 matrix

- Margaret Dayhoff in *Atlas of protein sequence and structure* (1978)
- the first widely-used amino acid substitution matrix

1	2	3	4	5	6	7	8	9
A	C	G	C	T	A	F	K	I
G	C	G	C	T	A	F	K	I
A	C	G	C	T	A	F	K	L
G	C	G	C	T	G	F	K	I
G	C	G	C	T	L	F	K	I
A	S	G	C	T	A	F	K	L
A	C	A	C	T	A	F	K	L

- derived from global alignments of closely related sequences
  - 71 groups of protein sequences
  - minimum 85% identity
  - functional proteins
  - 1572 amino-acid changes/mutations
- evolutionary model
  - assumes symmetry: A → B = B → A
  - assumes substitutions observed over short periods of time can be extrapolated to long periods of time (mathematically)

Extrapolating PAM matrices to longer distances

- Margaret Dayhoff in *Atlas of protein sequence and structure* (1978)
- Family of matrices: PAM1, PAM80, PAM120, PAM250

- *evolutionary interval is the time taken for n mutations to occur per 100 amino acids*
- the number represents the evolutionary distance between the sequences
- higher numbers denote greater distances

- PAM matrices for larger evolutionary distances are extrapolated from PAM1
- probabilities are calculated by matrix multiplication, *e.g.*
  $M_2 = M_1 \times M_1$
  $M_{250} = M_1^{250}$
- PAM250 is the substitution matrix calculated from M_{250}

Limitations of PAM matrices

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	O	T	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-3	-4	-3	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	0	-1	-3	0	-1	0	0	-2	-3	-1	-2	-5	6					
O	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

- inferred from a small dataset with $\geq$85 % identity
- mainly small globular proteins
- doesn't account for different evolutionary rates between conserved and non-conserved regions

BLOSUM (**BLO**cks **SU**bstitution **M**atrix)

	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
A	0	1	0	4																	A
G	-3	0	-2	0	6																G
P	-3	-1	-1	-1	-2	7															P
D	-3	0	-1	-2	-1	-1	6														D
E	-4	0	-1	-1	-2	-1	2	5													E
Q	-3	0	-1	-1	-2	-1	0	2	5												Q
N	-3	1	0	-2	0	-2	1	0	0	6											N
H	-3	-1	-2	-2	-2	-2	-1	0	0	1	8										H
R	-3	-1	-1	-1	-2	-2	-2	0	1	0	0	5									R
K	-3	0	-1	-1	-2	-1	-1	1	1	0	-1	2	5								K
M	-1	-1	-1	-1	-3	-2	-3	-2	0	-2	-2	-1	-1	5							M
I	-1	-2	-1	-1	-4	-3	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-1	-4	-3	-4	-3	-2	-3	-3	-2	-2	2	2	4					L
V	-1	-2	0	0	-3	-2	-3	-2	-2	-3	-3	-3	-2	1	3	1	4				V
W	-2	-3	-2	-3	-2	-4	-4	-3	-2	-4	-2	-3	-3	-1	-3	-2	-3	11			W
Y	-2	-2	-2	-2	-3	-3	-3	-2	-1	-2	2	-2	-2	-1	-1	-1	-1	2	7		Y
F	-2	-2	-2	-2	-3	-4	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	1	3	6	F
	C	S	T	A	G	P	D	E	Q	N	H	R	K	M	I	L	V	W	Y	F	

- Henikoff & Henikoff, *Amino acid substitution matrices from protein blocks* (1992). 10.1073/pnas.89.22.10915
- alignments of 500 distantly related protein families
- scores derived from frequencies of substitutions in blocks of ungapped local alignments
- BLOSUM*x* is based on sequences that share at least *x*% identity
  - e.g. BLOSUM62 was constructed from aligned sequences sharing no more than 62 % identity

Image: Paul Gardner via Wikimedia Commons

PAM *vs.* BLOSUM

BLOSUM80	BLOSUM62	BLOSUM45
PAM1	PAM120	PAM250

Less divergent → More divergent

- For closely related proteins: lower PAM matrices or higher BLOSUMs
- For distantly related proteins higher PAM matrices or lower BLOSUMs
- BLOSUM62 is commonly used for database searching (BLAST default)

- **In general**:
  - BLOSUMs perform well for local similarity searches
  - PAM matrices perform well for global alignments
- BLOSUMs are calculated from observed frequencies
- higher PAM matrices are extrapolated mathematically

Alignment: Gap Penalties

Why penalize gaps?

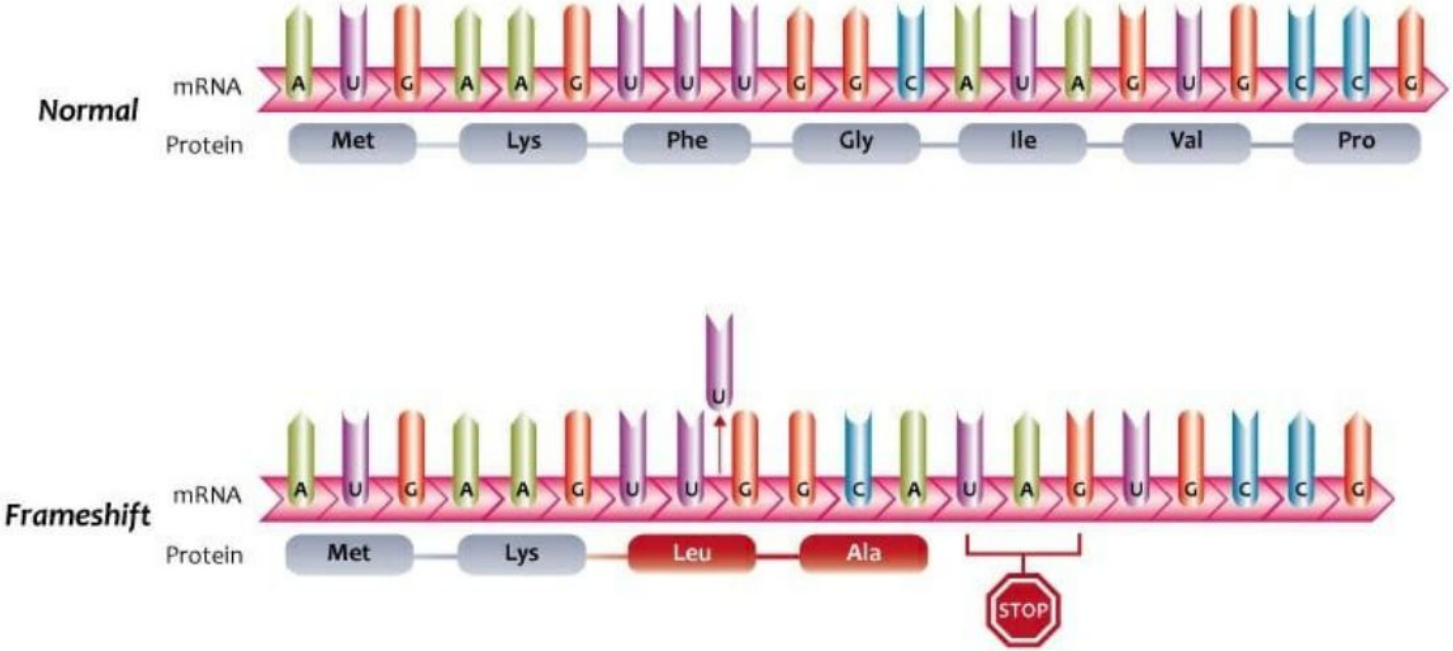
- allowing gaps with no cost results in misleading alignments

V D S – C Y
V E S L C Y

optimal alignment: **maximizes** the number of matches, and **minimizes** the number of gaps

- tradeoff: adding gaps reduces mismatches
- penalising gaps heavily forces alignments to have fewer gaps

Why penalize gaps?



Adapted from Campbell NA (ed). Biology, 2nd ed, 1990.

How do we penalise gaps?

Negative score:

- Gap penalty should be several times greater than the mismatch penalty
  - Proteins: an insertion/deletion could interrupt the entire polymer chain
  - DNA: shift the reading frame

A naïve approach:

- fixed penalty (σ) for every gap/indel
  - -σ for one indel
  - -2σ for two consecutive indels
  - -3σ for three consecutive indels
- what is the problem with that?

```
V D S - C Y
V E S L C Y
```

Fixed gap penalty

Alignment 1:

```
ATGTAGTGTATAGTACATGCA
ATGTAG-------TACATGCA
```

- **an indel of length *k* is more likely to occur as a single event than as *k* events each of length 1**
- *i.e.* alignment 1 is a better representation of homology

Alignment 2:

```
ATGTAGTGTATAGTACATGCA
ATGTA--G--TA---CATGCA
```

- a fixed gap penalty would give the same score for both
- treat gap initiation and gap extension differently

```
V D S - C Y
V E S L C Y
```

Affine gap penalties

affine: **linear** (in this context).
i.e. the penalty grows at the same rate as the length of the gap.

- large penalty for opening a gap
- much smaller penalty for gap extension
- an indel of length k has a penalty $W(k)$
  - $W(k) = -(\rho + \sigma \times k)$
  - ρ: penalty to open a gap
  - σ: penalty to extend a gap
- *e.g.* used by BLAST

Read Mapping: Seed-Extend

Recap

Kmers

Subsequences of length K

Used extensively in bioinformatics

Indexing

Method to quickly identify matching regions of two sequences

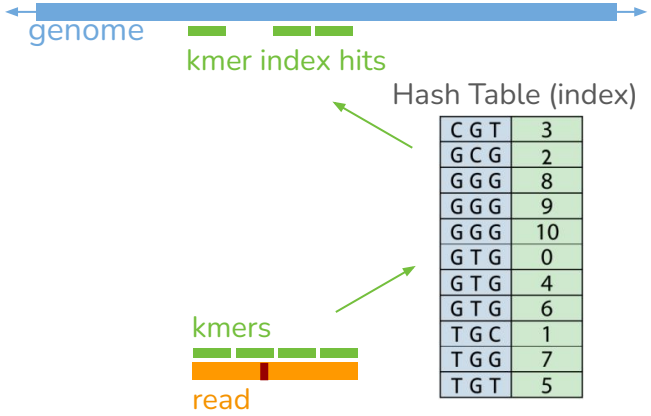
Key: Value store

Kmers: Occurrences

Alignment

Finding optimal match between two sequences

Considers matches, mismatches, gaps



	start	G	C	A	C	T	G	A
start	0	-2	-4	-6	-8	-10	-12	-14
G	-2	1	-1	-3	-5	-7	-9	-11
C	-4	-1	2	0	-2	-4	-6	-8
C	-6	-3	0	1	1	-1	-3	-5
T	-8	-5	-2	-1	0	2	0	-2
G	-10	-7	-4	-3	-2	0	3	1
A	-12	-9	-6	-3	-4	-2	1	4
T	-14	-11	-8	-5	-4	-3	-1	2

Alignment

```
GCACTGA-
||.||||.
GC-CTGAT
```

Read Mapping: Seed-Extend

Mapping reads to human reference genome

Ideally, we could use semi-global alignment

Is this feasible for average dataset?

-> each read is ~ 100 bp
-> reference length ~ 3.2 billion bp
-> number of reads ~ 300 million (30x coverage)

Semi-global alignment time and space complexity

-> quadratic: $O(n \times m)$
-> single read: 300 billion operations
-> average dataset: 10^{20}

Not feasible!

We need a heuristic approach

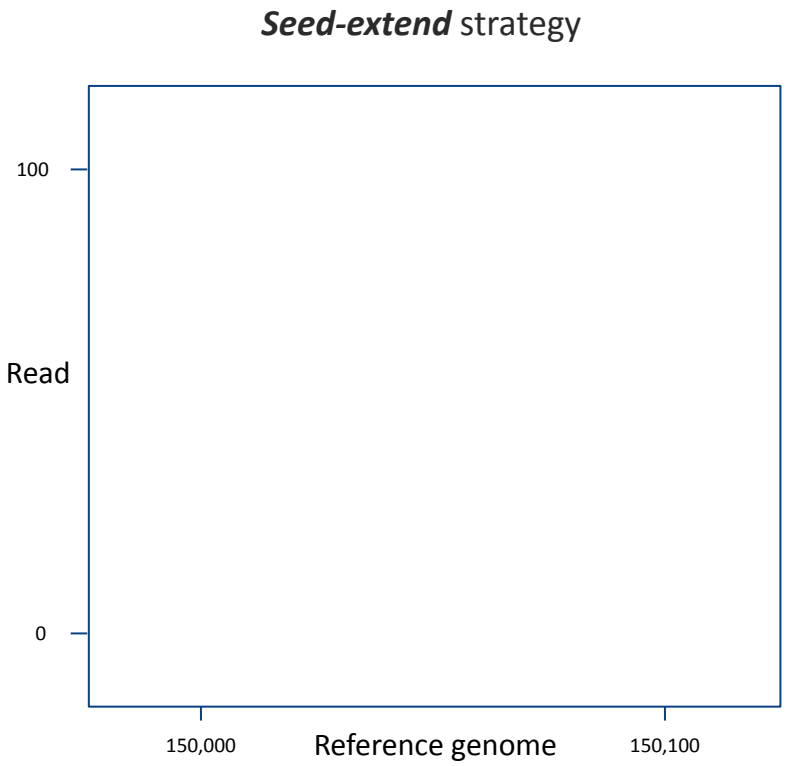
Needle in haystack via wikipedia commons

Read Mapping: Seed-Extend

Seed-Extend

- Combines indexing and alignment
- Used for aligning short, accurate sequences

Process



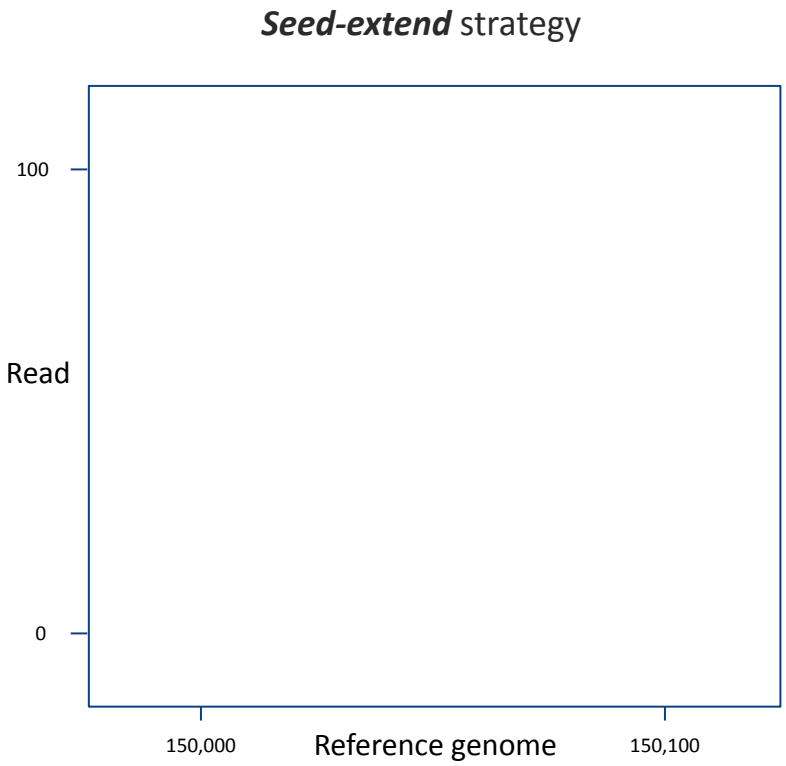
Read Mapping: Seed-Extend

Seed-Extend

- Combines indexing and alignment
- Used for aligning short, accurate sequences

Process

- Index the reference genome using kmers



Read Mapping: Seed-Extend

Seed-Extend

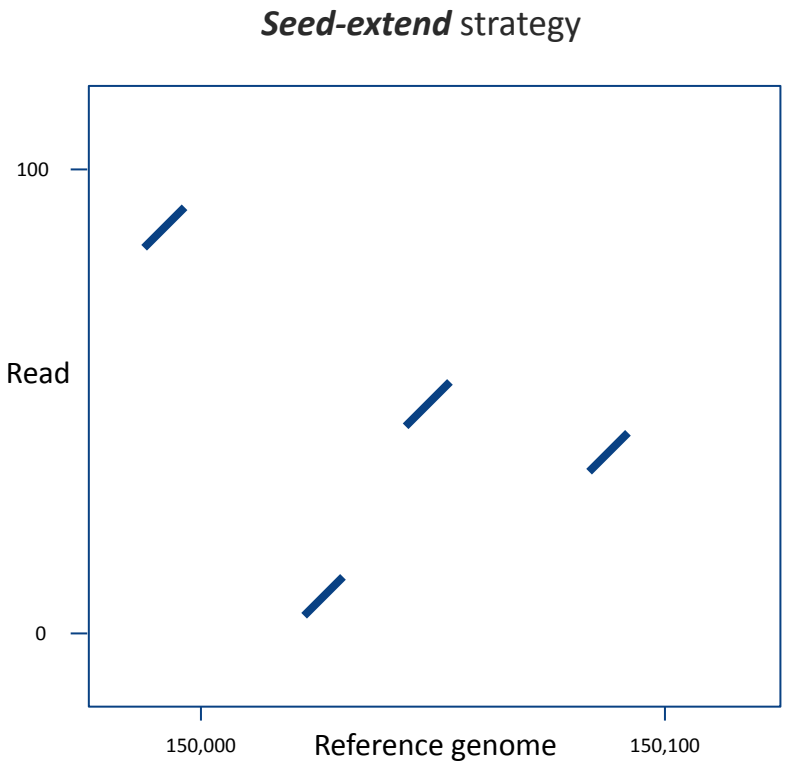
Combines indexing and alignment

Used for aligning short, accurate sequences

Process

Index the reference genome using kmers

Use the index to find “seed” matches for each read
(kmer hits between read and index)



Read Mapping: Seed-Extend

Seed-Extend

Combines indexing and alignment

Used for aligning short, accurate sequences

Process

Index the reference genome using kmers

Use the index to find “seed” matches for each read
(kmer hits between read and index)

Extend the match ends using alignment
(Local alignment and / or gap-free alignment)

Read Mapping: Seed-Extend

Seed-Extend

Combines indexing and alignment

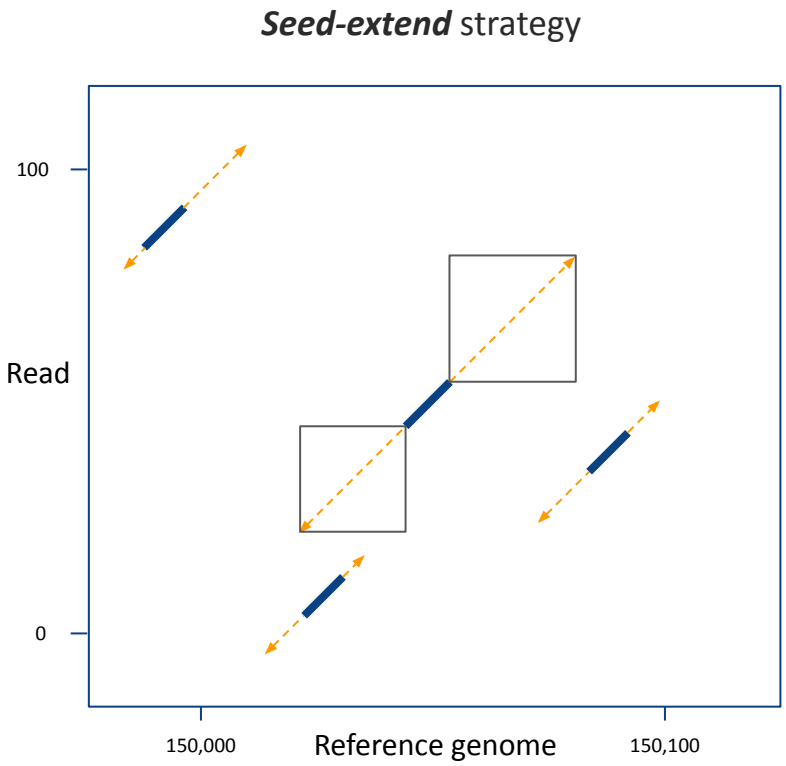
Used for aligning short, accurate sequences

Process

Index the reference genome using kmers

Use the index to find “seed” matches for each read
(kmer hits between read and index)

Extend the match ends using alignment
(Local alignment and / or gap-free alignment)



Read Mapping: Seed-Extend

Seed-Extend

Combines indexing and alignment

Used for aligning short, accurate sequences

Process

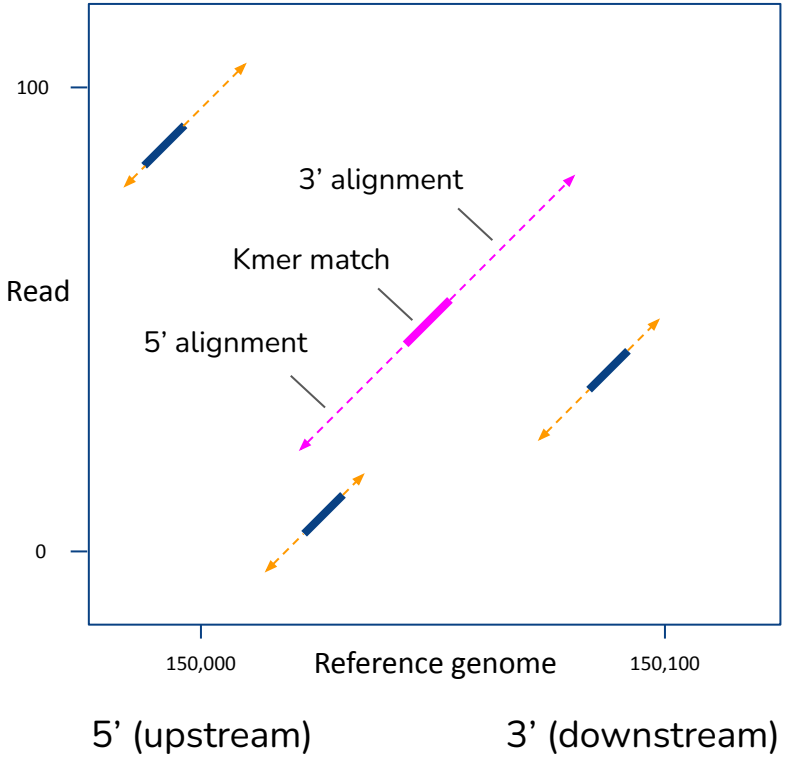
Index the reference genome using kmers

Use the index to find “seed” matches for each read
(kmer hits between read and index)

Extend the match ends using alignment
(Local alignment and / or gap-free alignment)

Return the best location
(5’ alignment + kmer match + 3’ alignment)

Seed-extend strategy



Read Mapping: Seed-Extend

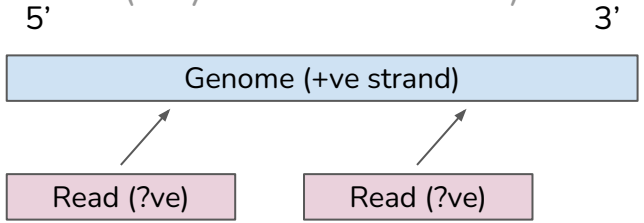
What about reverse strand?

Genome: Double Stranded DNA (except some viruses)

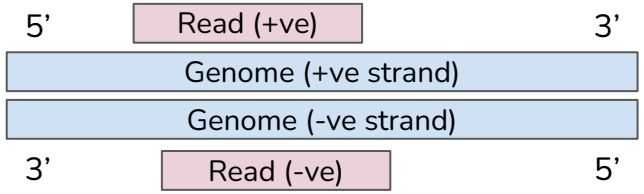
DNA Seq. Reads: Can originate from +ve or -ve strand

Reference genome: Only the +ve strand

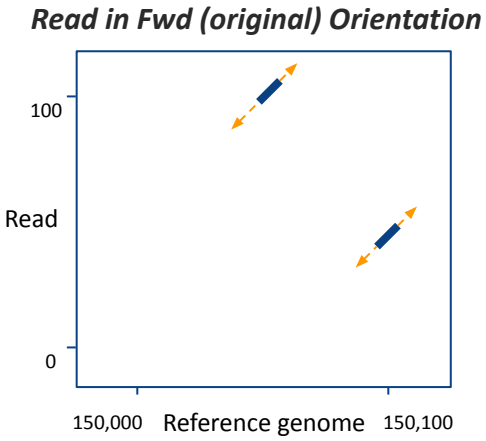
Reads are mapped to genome
(only +ve strand stored)



But reads originate from both strands of genome!



Read Mapping: Seed-Extend



What about reverse strand?

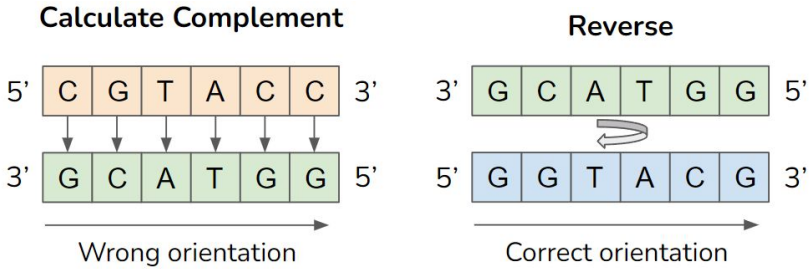
Genome: Double Stranded DNA (except some viruses)

DNA Seq. Reads: Can originate from +ve or -ve strand

Reference genome: Only the +ve strand

For each read (sequence to align)

1. Do Seed-Extend (original - forward orientation).
2. <u>Flip it (reverse complement).</u>



Read Mapping: Seed-Extend

What about reverse strand?

Genome: Double Stranded DNA (except some viruses)

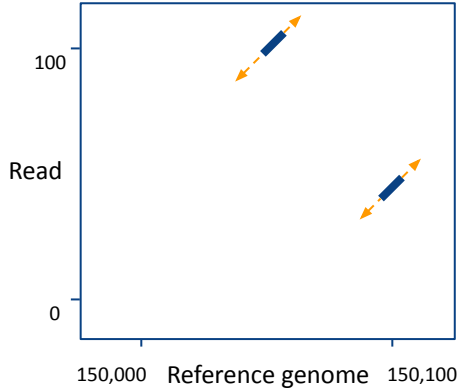
DNA Seq. Reads: Can originate from +ve or -ve strand

Reference genome: Only the +ve strand

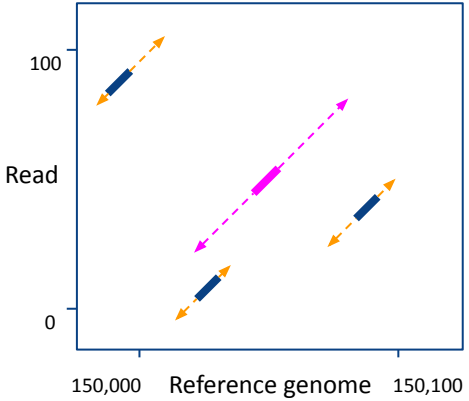
For each read (sequence to align)

1. Do Seed-Extend (original - forward orientation).
2. <u>Flip it (reverse complement).</u>
3. Do Seed-Extend (flipped - reverse orientation).
4. Take the best alignment from Fwd & Rev orientation.
5. If it's Rev, report seq as originating from the -ve strand .

Read in Fwd (original) Orientation



Read in Rev (flipped) Orientation



Read Mapping: Seed-Chain-Align

Read Mapping: Seed-Chain-Align

Aligning long, noisy sequences

Long-read data has unique properties due to sequencing technology

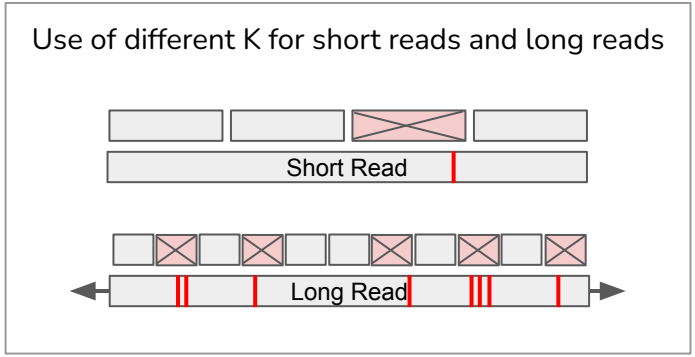
Read Mapping: Seed-Chain-Align

Aligning long, noisy sequences

Long-read data has unique properties due to sequencing technology

Error rate

-> Long-read data more noisy
-> ~1 error every 10 bp (improved by 2023)
-> Kmer size needs to be lower



Read Mapping: Seed-Chain-Align

Aligning long, noisy sequences

Long-read data has unique properties due to sequencing technology

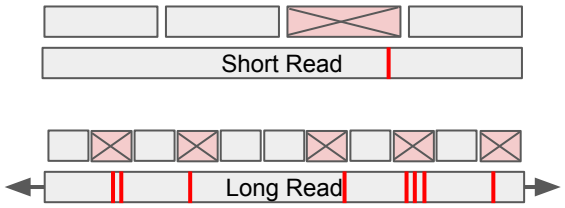
Error rate

-> Long-read data more noisy
-> ~1 error every 10 bp (improved by 2023)
-> Kmer size needs to be lower

Read length

-> Length ~10kb
-> Many kmers -> seeds for a single read (thousands)
-> Alignments for each seed: very expensive

Use of different K for short reads and long reads

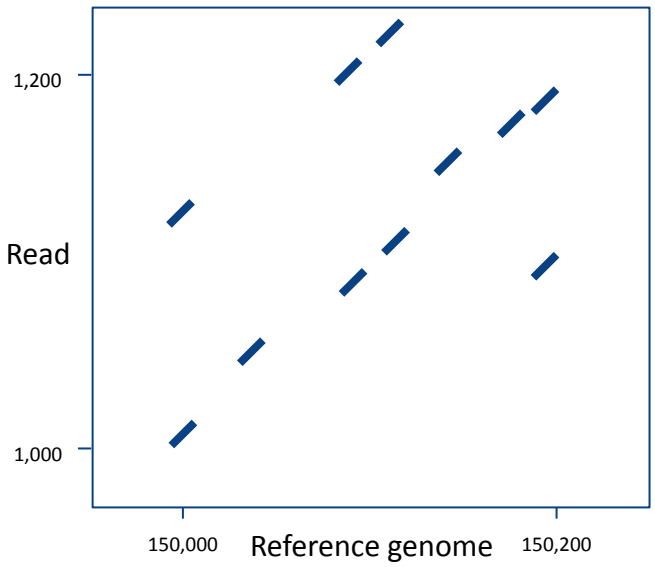


Specificity vs Sensitivity when varying K

Read Mapping: Seed-Chain-Align

Can we use this to our advantage?

- For a given read, many kmers will match the genome index
- Expect the relative kmer positions to be similar in the read and genome



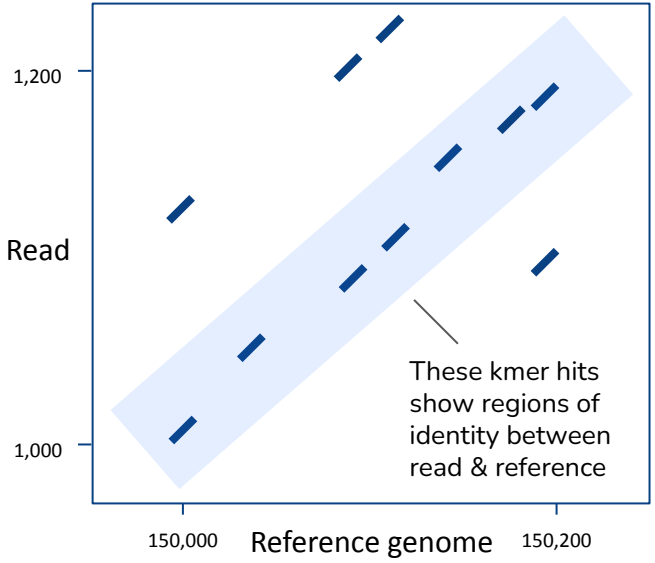
Read Mapping: Seed-Chain-Align

Can we use this to our advantage?

For a given read, many kmers will match the genome index

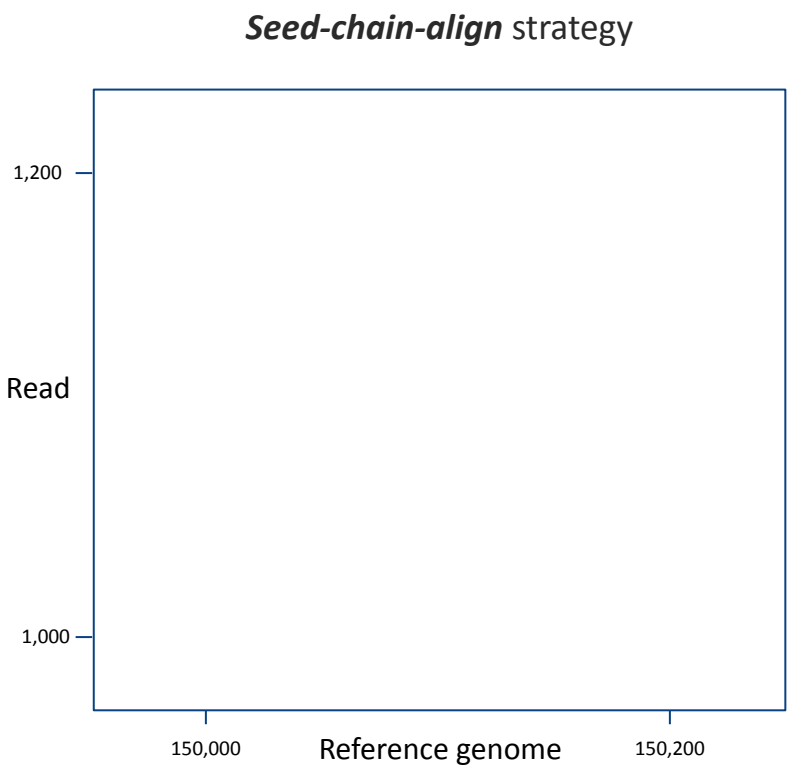
Expect the relative kmer positions to be similar in the read and genome

Chaining



Read Mapping: Seed-Chain-Align

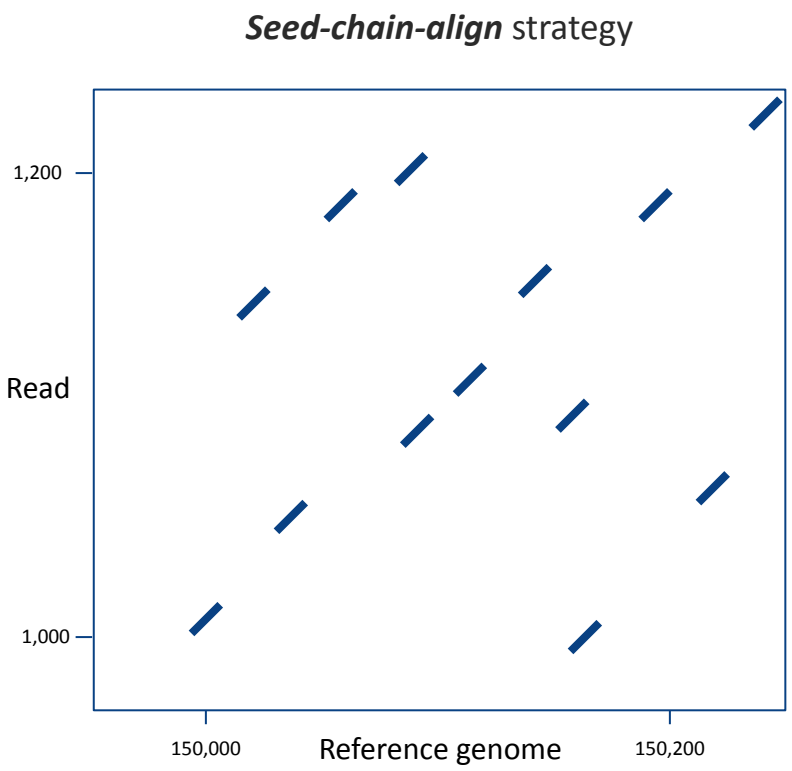
Seed-Chain-Align



Read Mapping: Seed-Chain-Align

Seed-Chain-Align

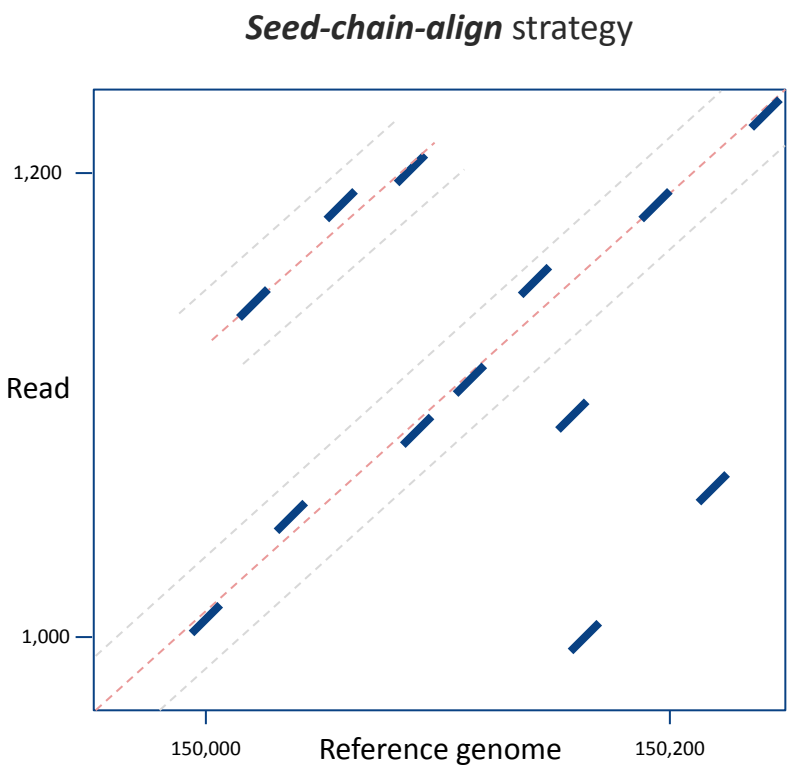
1. Break the read into k-mers and look up their genomic positions in the index to find seeds
   Note: don't need to extract every possible kmer from the read – can jump a little in between. Common to extract a kmer every few – tens of base pairs.



Read Mapping: Seed-Chain-Align

Seed-Chain-Align

1. Break the read into k-mers and look up their genomic positions in the index to find seeds
   Note: don't need to extract every possible kmer from the read – can jump a little in between. Common to extract a kmer every few – tens of base pairs.
2. Identify colinear chains

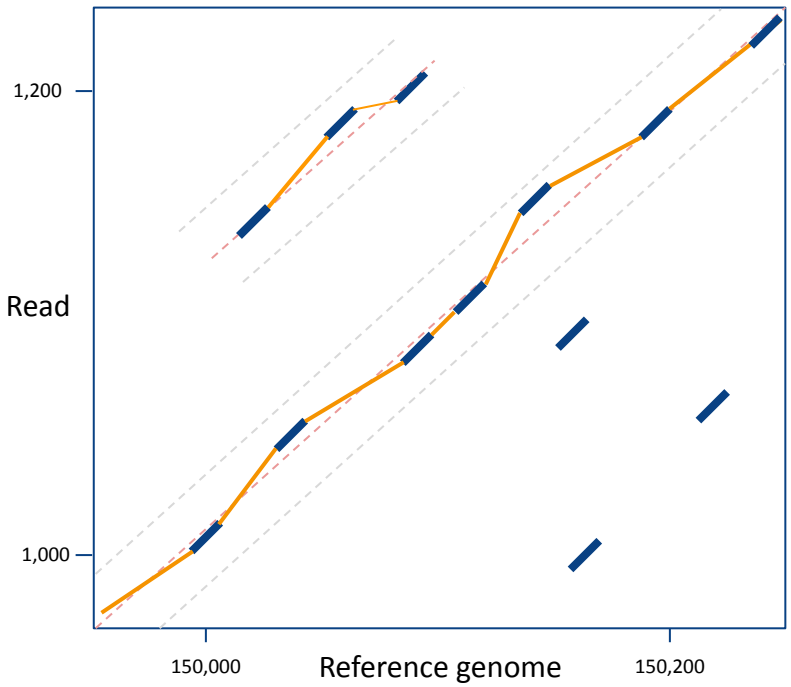


Read Mapping: Seed-Chain-Align

Seed-Chain-Align

1. Break the read into k-mers and look up their genomic positions in the index to find seeds
   Note: don't need to extract every possible kmer from the read – can jump a little in between. Common to extract a kmer every few – tens of base pairs.
2. Identify colinear chains
3. For each: base-level alignments to fill gaps

Seed-chain-align strategy

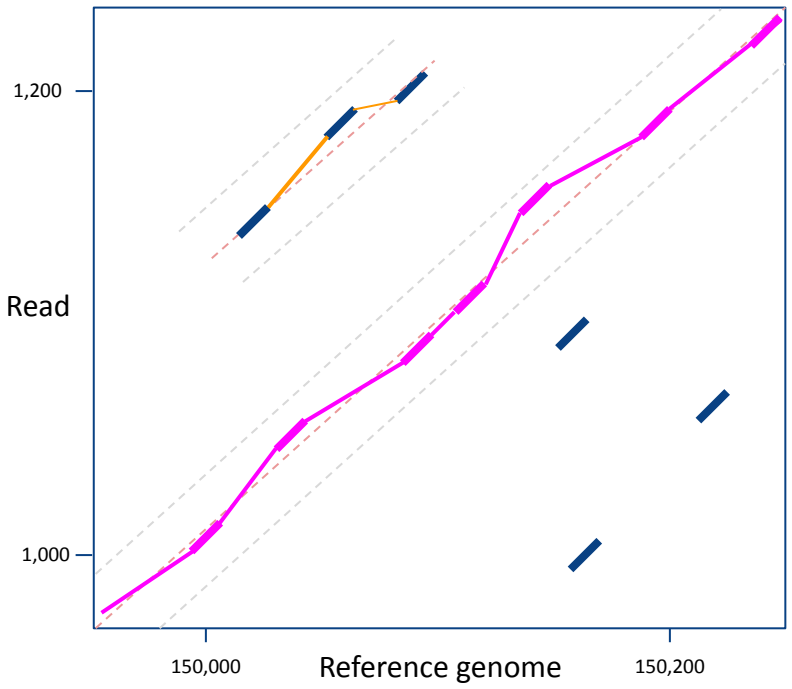


Read Mapping: Seed-Chain-Align

Seed-Chain-Align

1. Break the read into k-mers and look up their genomic positions in the index to find seeds
   Note: don't need to extract every possible kmer from the read – can jump a little in between. Common to extract a kmer every few – tens of base pairs.
2. Identify colinear chains
3. For each: base-level alignments to fill gaps
4. Return the best location
   (gap-filled chain with highest score)

Seed-chain-align strategy



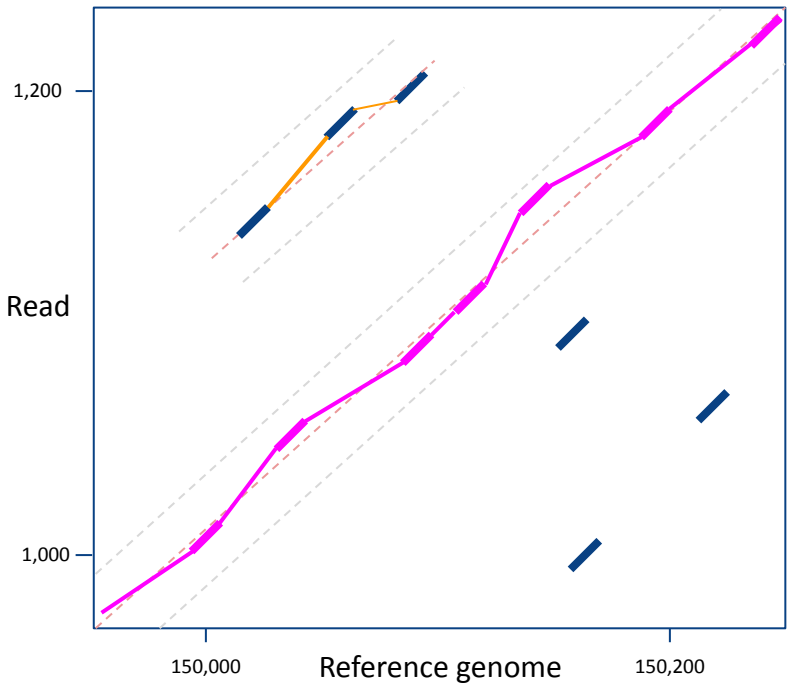
Read Mapping: Seed-Chain-Align

Seed-Chain-Align

1. Break the read into k-mers and look up their genomic positions in the index to find seeds
   Note: don't need to extract every possible kmer from the read – can jump a little in between. Common to extract a kmer every few – te
   base pairs.
2. Identify colin
3. alignments to fill gaps
   turn the best location
   (gap-filled chain with highest score)

Expanded upon in week 3

Seed-chain-align strategy

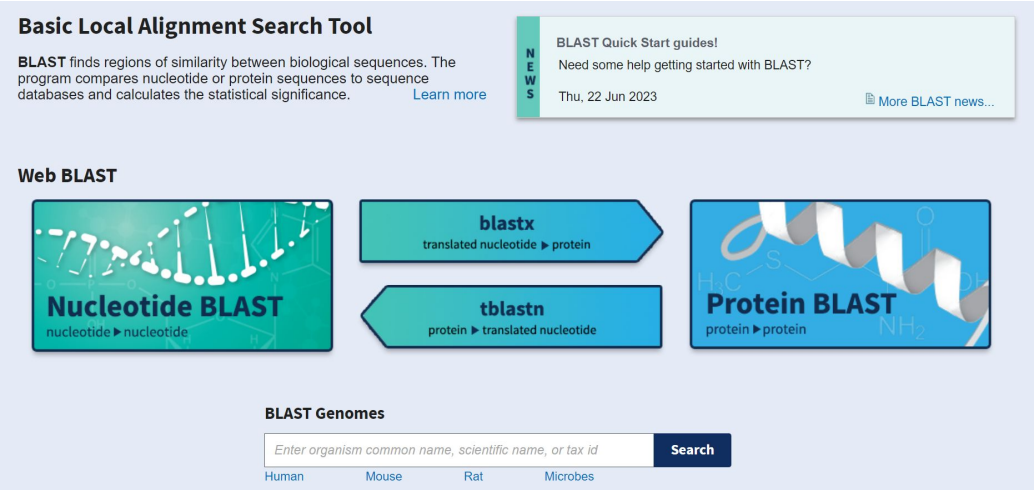


BLAST

BLAST

Basic Local Alignment Search Tool (BLAST) - Extreme efficiency heuristics!

- Finding conserved sequences (eg. genes)
- Query sequence -> Massive database
- Somewhere between Seed-Extend and Seed-Chain-Align



BLAST

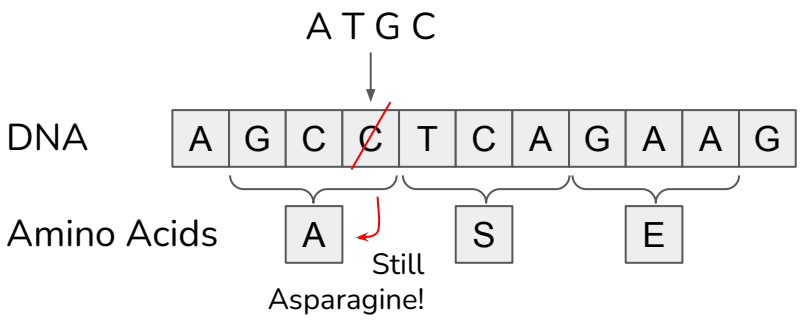
Extreme efficiency heuristics

BLAST

Extreme efficiency heuristics

1. DNA translated to protein seq for use

   -> AA seq more conserved than DNA seq
   -> Degeneracy / codon wobble
   -> 3rd base in codon: multiple different nucleotides encode same AA
   -> Eliminates meaningless DNA mismatches
   -> More useful kmer hits
   (mismatches, but coding seq undisturbed)



BLAST

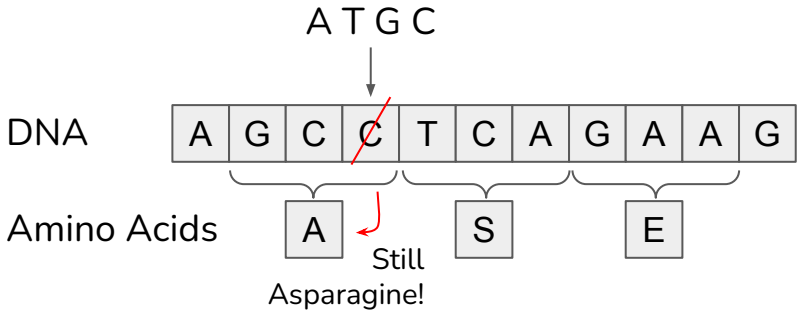
Extreme efficiency heuristics

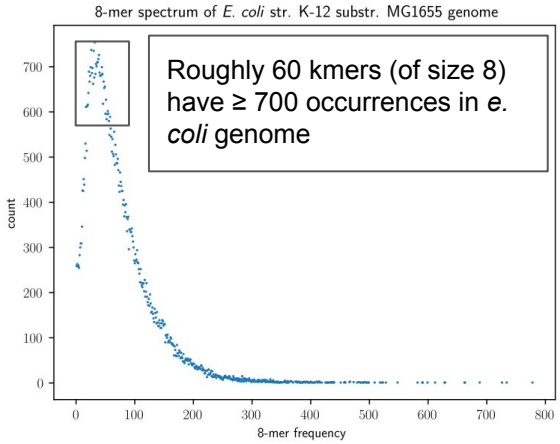
1. DNA translated to protein seq for use

   -> AA seq more conserved than DNA seq
   -> Degeneracy / codon wobble
   -> 3rd base in codon: multiple different nucleotides encode same AA
   -> Eliminates meaningless DNA mismatches
   -> More useful kmer hits
   (mismatches, but coding seq undisturbed)

2. Low-complexity regions removed

   -> Repetitive DNA
   -> Functional elements generally not repetitive
   -> Leads to uninformative kmer seeds (if retained)





Ytngargar via wikipedia commons

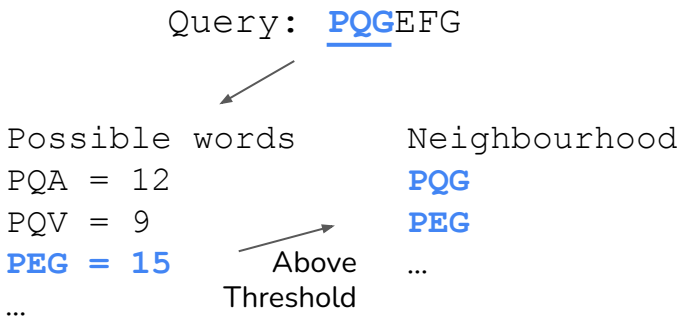
BLAST

Extreme efficiency heuristics

3. Kmers are fast, let's maximise their use

-> Generate kmer (word) 'neighbourhood' for query kmers
-> Referred to as
-> Allows mismatches in the seed step
(prev. only exact kmer matches in this lecture)
-> Similar to short ungapped alignment
-> Kmer matches must be above threshold score
(high scoring words)
-> Words in neighbourhood looked up in index

Generating Word Neighbourhood



BLAST

Extreme efficiency heuristics

3. Kmers are fast, let's maximise their use

   -> Generate kmer (word) 'neighbourhood' for query kmers
   -> Referred to as
   -> Allows mismatches in the seed step
   (prev. only exact kmer matches in this lecture)
   -> Similar to short ungapped alignment
   -> Kmer matches must be above threshold score
   (high scoring words)
   -> Words in neighbourhood looked up in index

4. Identify nearby word hits on same diagonal

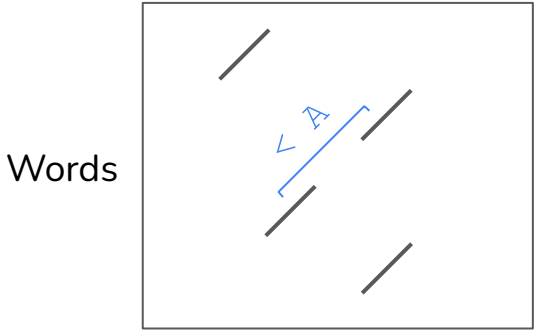
   -> Similar to Chaining in Seed-Chain-Align
   -> Word distance < A
   -> Extends matching region to a:
   "High-scoring Segment Pair (HSP)"

Generating Word Neighbourhood

Query: **PQG**EFG

Possible words		Neighbourhood
PQA = 12		**PQG**
PQV = 9		**PEG**
PEG = 15	Above Threshold	...
...		

Matching words on diagonal



BLAST

Extreme efficiency heuristics

5. Ungapped extension

-> Extend HSP on either end

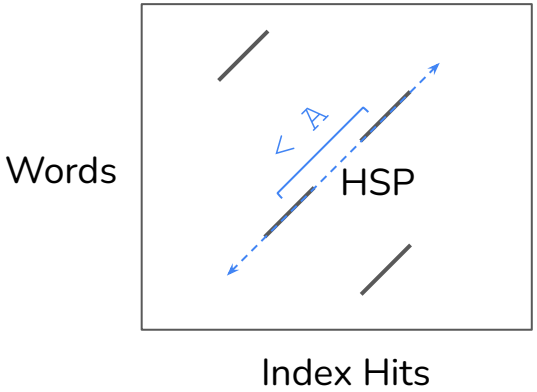
-> How far do we extend?
Stop when score < threshold (efficiency)

-> Rank HSPs by E-score & do cutoff

How likely this segment would appear in random sequence, same size of our database?

Probability that this segment appears by simple chance. Lower is better.

Ungapped extension



BLAST

Extreme efficiency heuristics

5. Ungapped extension

-> Extend HSP on either end

-> How far do we extend?
Stop when score < threshold (efficiency)

-> Rank HSPs by E-score & do cutoff

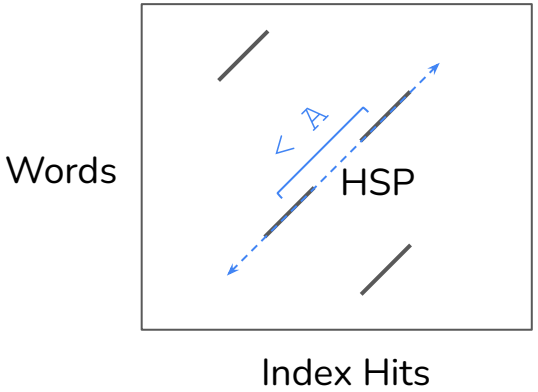
How likely this segment would appear in random sequence, same size of our database?
Probability that this segment appears by simple chance.
Lower is better.

6. Gapped alignment

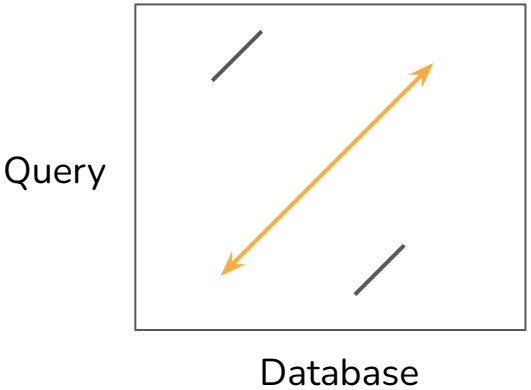
-> Local-alignment for HSP + end extension (early termination)

-> Recalculate E-score & report alignments greater than threshold

Ungapped extension



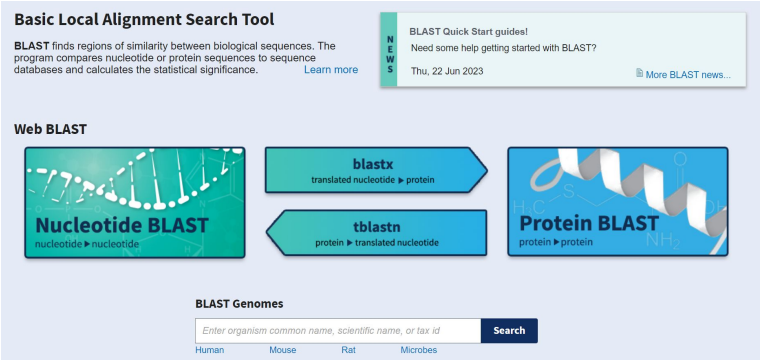
Final Alignment



BLAST

Summary

- Iterative approach
- Ruthlessly & Continuously reduces search space
- Very fast!
- Clever thought applied to properties of genomic data.



Sequence Alignment

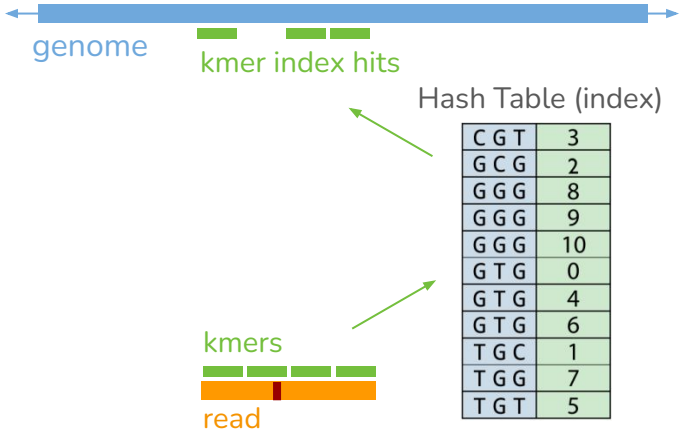
Kmers

breaking sequence into smaller pieces

Indexing

Use of kmers + hash tables

Fast lookup of subsequence matches

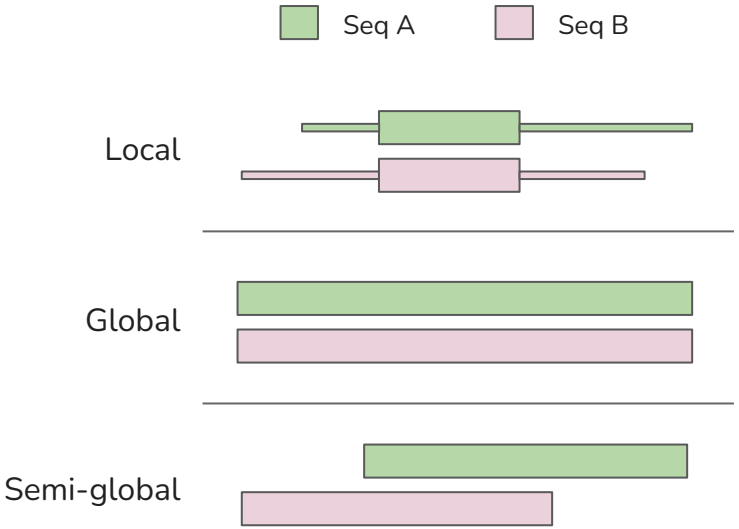


C G T	3
G C G	2
G G G	8
G G G	9
G G G	10
G T G	0
G T G	4
G T G	6
T G C	1
T G G	7
T G T	5

Alignment

Different variations on Levenshtein distance

...for different tasks



Melbourne Bioinformatics
BIOINFORMATICS + DATA SERVICES + INFRASTRUCTURE, FOR LIFE SCIENCES TODAY

THE UNIVERSITY OF MELBOURNE

Thank you!

Don’t forget your signed academic integrity statement
Due tomorrow!!

Today: Sequence Alignment II

Next time: Comparing Sequences I