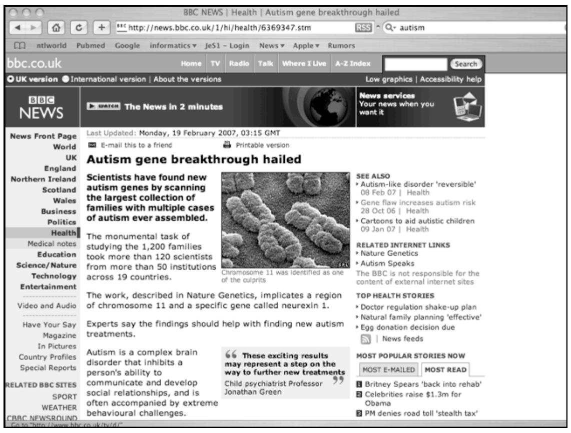
Sequence Alignment

Why?

- Genome sequencing gives us new gene sequences
- Network biology gives us functional information on genes/proteins
- Analysis of mutants links unknown genes to diseases
- *Can we learn anything from other known sequences about our new gene/protein?*



Last Updated: Monday, 19 February 2007, 03:15 GMT

Autism gene breakthrough hailed

Scientists have found new autism genes by scanning the largest collection of families with multiple cases of autism ever assembled.

The monumental task of studying the 1,200 families took more than 120 scientists from more than 50 institutions across 19 countries.

Chromosome 11 was identified as one of the culprits

The work, described in Nature Genetics, implicates a region of chromosome 11 and a specific gene called neurexin 1.

Experts say the findings should help with finding new autism treatments.

Autism is a complex brain disorder that inhibits a person's ability to communicate and develop social relationships, and is often accompanied by extreme behavioural challenges.

“These exciting results may represent a step on the way to further new treatments”
Child psychiatrist Professor Jonathan Green

SEE ALSO
- Autism-like disorder 'reversible' 08 Feb 07 | Health
- Gene flaw increases autism risk 28 Oct 06 | Health
- Cartoons to aid autistic children 09 Jan 07 | Health

RELATED INTERNET LINKS
- Nature Genetics
- Autism Speaks

The BBC is not responsible for the content of external internet sites

TOP HEALTH STORIES
- Doctor regulation shake-up plan
- Natural family planning 'effective'
- Egg donation decision due

MOST POPULAR STORIES NOW
1. Britney Spears 'back into rehab'
2. Celebrities raise $1.3m for Obama
3. PM denies road toll 'stealth tax'

What is it?

```
ACCGGTATCCTAGGAC
ACCTATCTTAGGAC
```

Are these two sequences related?

How similar (or dissimilar) are they?

What is it?

```
ACCGGTATCCTAGGAC
|||  |||| ||||||
ACC--TATCTTAGGAC
```

- Match the two sequences as closely as possible = aligned
- Therefore, alignments need a score

Why do we care?

- DNA and Proteins are based on linear sequences
- Information is encoded in these sequences
- All bioinformatics at some level comes back to matching sequences that might have some noise or variability

Alignment Types

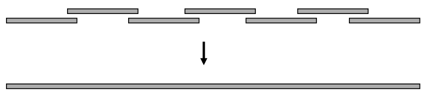
- Global: used to compare to similar sized sequences.
  - Compare closely related genes
  - Search for mutations or polymorphisms in a sequence compared to a reference.

Alignment Types

- Local: used to find shared subsequences.
  - Search for protein domains
  - Find gene regulatory elements
  - Locate a similar gene in a genome sequence.

Alignment Types

- Ends Free: used to find joins/overlaps.
  - Align the sequences from adjacent sequencing primers.



How do we score alignments?

```
ACCGGTATCCTAGGAC
|||  |||| ||||||
ACC--TATCTTAGGAC
```

- Assign a score for each match along the sequence.

How do we score alignments?

```
ACCGGTATCCTAGGAC
|||  |||| ||||||
ACC--TATCTTAGGAC
```

- Assign a score (or penalty) for each substitution.

How do we score alignments?

```
ACCGGTATCCTAGGAC
|||  |||| ||||||
ACC--TATCTTAGGAC
```

- Assign a score (or penalty) for each insertion or deletion.
- insertions/deletions otherwise known as indels

How do we score alignments?

```
ACCGGTATCCTAGGAC
|||  |||| ||||||
ACC--TATCTTAGGAC
```

- Matches and substitutions are 'easy' to deal with.
  - We'll look at substitution matrices later.
- How do we score indels: gaps?

How do we score gaps?

```
ACCGGTATCC---GAC
|||  ||||    |||
ACC--TATCTTAGGAC
```

- A gap is a consecutive run of indels
- The gap length is the number of indels.
- The simple example here has two gaps of length 2 and 3

How do we score gaps?

```
ACCGGTATCC---GAC
|||  ||||    |||
ACC--TATCTTAGGAC
```

- Constant: Length independent weight
- Affine: *Open* and *Extend* weights.
- Convex: Each additional gap contributes less
- Arbitrary: Some arbitrary function on length

Choosing Gap Penalties

- The choice of Gap Scoring Penalty is very sensitive to the context in which it is applied:
  - introns vs exons
  - protein coding regions
  - mis-matches in PCR primers

Substitution Matrices

- Substitution matrices are used to score substitution events in alignments.
- Particularly important in Protein sequence alignments but relevant to DNA sequences as well.
- Each scoring matrix represents a particular theory of evolution

Similarity/Distance

- Distance is a measure of the cost or replacing one residue with another.
- Similarity is a measure of how similar a replacement is.

e.g. replacing a hydrophobic residue with a hydrophilic one.

- The logic behind both are the same and the scoring matrices are interchangeable.

DNA Matrices

Identity matrix

	A	C	G	T
A	1	0	0	0
C	0	1	0	0
G	0	0	1	0
T	0	0	0	1

BLAST

	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

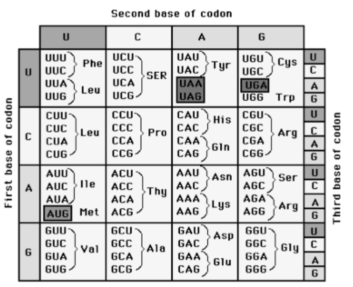
However, some changes are more likely to occur than others (even in DNA). When looking at distance, the ease of mutation is a factor. a.g. A-T and A-C replacements are rarer than A-G or C-T.

Protein Substitution Matrices

How can we score a substitution in an aligned sequence?

- Identity matrix like the simple DNA one.
- Genetic Code Matrix:

  For this, the score is based upon the minimum number of DNA base changes required to convert one amino acid into the other.



Second base of codon

First base of codon	U	C	A	G	Third base of codon
U	UUU } Phe UUC UUA } Leu UUG	UCU } SER UCC UCA UCG	UAU } Tyr UAC UAA UAG	UGU } Cys UGC UGA UGG Trp	U C A G
C	CUU } Leu CUC CUA CUG	CCU } Pro CCC CCA CCG	CAU } His CAC CAA } Gln CAG	CGU } Arg CGC CGA CGG	U C A G
A	AUU } Ile AUC AUA AUG Met	ACU } Thy ACC ACA ACG	AAU } Asn AAC AAA } Lys AAG	AGU } Ser AGC AGA } Arg AGG	U C A G
G	GUU } Val GUC GUA GUG	GCU } Ala GCC GCA GCG	GAU } Asp GAC GAA } Glu CAG	GGU } Gly GGC GGA GGG	U C A G

The genetic code, written by convention in the form in which the Codons appear in mRNA. The three terminator codons, UAA, UAG, and UGA, are boxed in red; the AUG initiator codon is shown in green.

Protein Substitution Matrices

How can we score a substitution in an aligned sequence?

- Amino acid property matrix

  Assign arbitrary values to the relatedness of different amino acids:

  e.g. hydrophobicity , charge, pH, shape, size

Matrices based on Probability

$$S_{ij} = \log (q_{ij}/p_i p_j)$$

Sij is the log odds ratio of two probabilities: amino acids i and j are aligned by evolutionary descent and the probability that they aligned at random**.**

This is the basis for commonly used substitution matrices.

PAM matrices

Dayhoff, Schwarz and Orcutt 1978 took these into consideration when constructing the PAM matrices:

Took 71 protein families - where the sequences differed by no more than 15% of residues (i.e. 85% identical)

Aligned these proteins

Build a theoretical phylogenetic tree

Predicted the most likely residues in the ancestral sequence

PAM Matrices

- Ignore evolutionary direction
- Obtained frequencies for residue X being substituted by residue Y over time period Z
- Based on 1572 residue changes
- They defined a substitution matrix as 1 PAM (point accepted mutation) if the expected number of substitutions was 1% of the sequence length.

PAM Matrices

To increase the distance, they multiplied the the PAM1 matrix.

PAM250 is one of the most commonly used.

PAM - notes

The PAM matrices are rooted in the original datasets used to create the theoretical trees

They work well with closely related sequences

Based on data where substitutions are most likely to occur from single base changes in codons.

PAM - notes

Biased towards conservative mutations in the DNA sequence (rather than amino acid substitutions) that have little effect on function/structure.

Replacement at any site in the sequence depends only on the amino acid at that site and the probability given by the table.This does not represent evolutionary processes correctly. Distantly related sequences usually have regions of high conservation (blocks).

PAM - notes

36 residue pairs were not observed in the dataset used to create the original PAM matrix

A new version of PAM was created in 1992 using 59190 substitutions: Jones, Taylor and Thornton 1992 CAMBIOS 8 pp 275

BLOSUM matrices

Henikoff and Henikoff 1991

Took sets of aligned ungapped regions from protein families from the BLOCKS database.

The BLOCKS database contain short protein sequences of high similarity clustered together. These are found by applying the MOTIF algorithm to the SWISS-PROT and other databases. The current release has 8656 Blocks.

BLOSUM matrices

Sequences were clustered whenever the %identify exceeded some percentage level.

Calculated the frequency of any two residues being aligned in one cluster also being aligned in another

Correcting for the size of each cluster.

BLOSUM matrices

Resulted in the fraction of observed substitutions between any two residues over all observed substitutions.

The resulting matrices are numbered inversely from the PAM matrices so the BLOSUM50 matrix was based on clusters of sequence over 50% identity, and BLOSUM62 where the clusters were at least 62% identical.

BLOSUM 62 Matrix

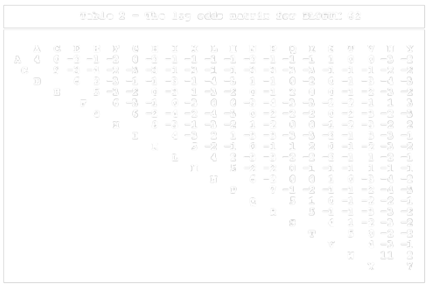


Table 2 - The log odds matrix for BLOSUM 62

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

Summary so far…

- Gaps
  - Indel operations
  - Gap scoring methods
- Substitution matrices
  - DNA largely simple matrices
  - Protein matrices are based on probability
  - PAM and BLOSUM

How do we do it?

- Like everything else there are several methods and choices of parameters
- The choice depends on the question being asked
  - What kind of alignment?
  - Which substitution matrix is appropriate?
  - What gap-penalty rules are appropriate?
  - Is a heuristic method good enough?

Working Parameters

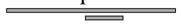
- For proteins, using the affine gap penalty rule and a substitution matrix:

Query Length	Matrix	Gap (open/extend)
<35	PAM-30	9,1
35-50	PAM-70	10,1
50-85	BLOSUM-80	10,1
>85	BLOSUM-62	11,1

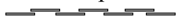
Alignment Types

- Global: used to compare to similar sized sequences.

- Local: used to find similar subsequences.



- Ends Free: used to find joins/overlaps.



Global Alignment

- Two sequences of similar length
- Finds the best alignment of the two sequences
- Finds the score of that alignment
- Includes **ALL** bases from both sequences in the alignment and the score.

- Needleman-Wunsch algorithm

Needleman-Wunsch algorithm

- Gaps are inserted into, or at the ends of each sequence.
- The sequence length (bases+gaps) are identical for each sequence
- Every base or gap in each sequence is aligned with a base or a gap in the other sequence

Needleman-Wunsch algorithm

- Consider 2 sequences *S* and *T*
- Sequence *S* has *n* elements
- Sequence *T* has *m* elements
- Gap penalty ?

How do we score gaps?

```
ACCGGTATCC---GAC
|||  ||||    |||
ACC--TATCTTAGGAC
```

- Constant: Length independent weight
- Affine: *Open* and *Extend* weights.
- Convex: Each additional gap contributes less
- Arbitrary: Some arbitrary function on length
  - Lets score each gap as –1 times length

Needleman-Wunsch algorithm

- Consider 2 sequences *S* and *T*
- Sequence *S* has *n* elements
- Sequence *T* has *m* elements
- Gap penalty –1 per base (arbitrary gap penalty)
- An alignment between base *i* in *S* and a gap in *T* is represented: $(S_i,-)$
- The score for this is represented : $\sigma(S_i,-) = -1$

Needleman-Wunsch algorithm

- Substitution/Match matrix for a simple alignment
- Several models based on probability….

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

Needleman-Wunsch algorithm

- Substitution/Match matrix for a simple alignment
- Simple identify matrix (2 for match, -1 for mismatch)
- An alignment between base *i* in *S* and base *j* in *T* is represented: (S_i,T_j)
- The score for this occurring is represented: $\sigma(S_i,T_j)$

Needleman-Wunsch algorithm

- Set up a array V of size n+1 by m+1
- Row 0 and Column 0 represent the cost of adding gaps to either sequence at the start of the alignment
- Calculate the rest of the cells row by row by finding the optimal route from the surrounding cells that represent a gap or match/mismatch
  - This is easier to demonstrate than to explain

Needleman-Wunsch algorithm

– lets start by trying out a simple example alignment:

```
S = ACCGGTAT
T =  ACCTATC
```

Needleman-Wunsch algorithm

– Get lengths

```
S = ACCGGTAT
T =  ACCTATC
```

Length of *S* = *m* = 8

Length of T = *n* = 7

(lengths approx equal so OK for Global Alignment)

Create array m+1 by n+1

(i.e. 9 by 8)

Add on bases from each sequence

		A	C	C	G	G	T	A	T	(S)
A										
C										
C										
T										
A										
T										
C										
(T)										

Armstrong, 2010

Represent scores for gaps in row/ col 0

		A	C	C	G	G	T	A	T	(S)
	0	→ -1	→ -2							
A										
C										
C										
T										
A										
T										
C										
(T)										

Armstrong, 2010

Represent scores for gaps in row/ col 0

		A	C	C	G	G	T	A	T	(S)
	0	→ -1	→ -2	→ -3	→ -4	→ -5	→ -6	→ -7	→ -8	
A	↓ -1									
C	↓ -2									
C	↓ -3									
T	↓ -4									
A	↓ -5									
T	↓ -6									
C	↓ -7									
(T)										

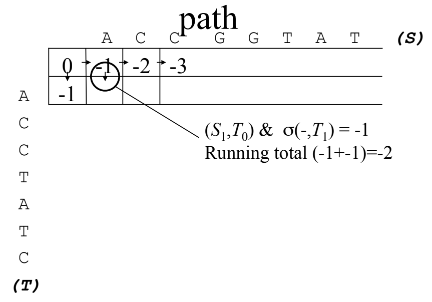
Armstrong, 2010

For each cell consider the 'best' path

		A	C	C	G	G	T	A	T	(S)
	0	→ -1	→ -2	→ -3	→ -4	→ -5	→ -6	→ -7	→ -8	
A	↓ -1	▨								
C	↓ -2									
C	↓ -3									
T	↓ -4									
A	↓ -5									
T	↓ -6									
C	↓ -7									
(T)										

Armstrong, 2010

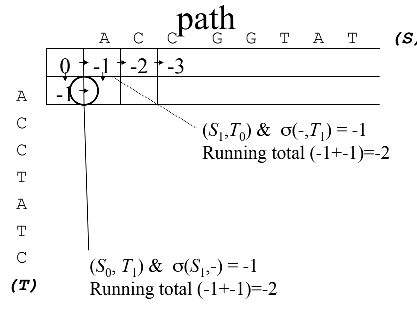
For each cell consider the 'best' path



(S_1,T_0) & σ(-,T_1) = -1
Running total (-1+-1)=-2

Armstrong, 2010

For each cell consider the 'best' path

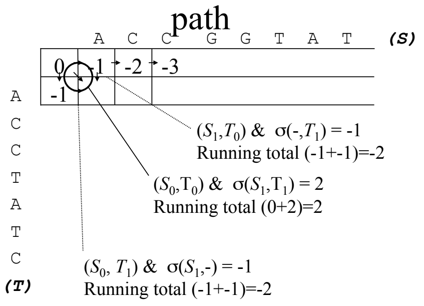


(S_1,T_0) & σ(-,T_1) = -1
Running total (-1+-1)=-2

(S_0, T_1) & σ(S_1,-) = -1
Running total (-1+-1)=-2

Armstrong, 2010

For each cell consider the ‘best’ path



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

(S) (T)

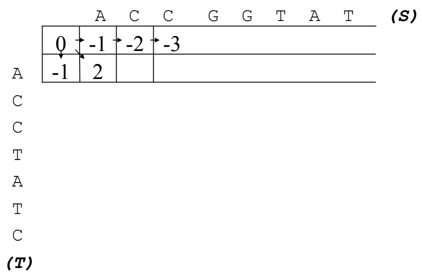
(S_1,T_0) & $\sigma(-,T_1) = -1$
Running total (-1+-1)=-2

(S_0,T_0) & $\sigma(S_1,T_1) = 2$
Running total (0+2)=2

(S_0, T_1) & $\sigma(S_1,-) = -1$
Running total (-1+-1)=-2

Armstrong, 2010

Choose and record ‘best’ path

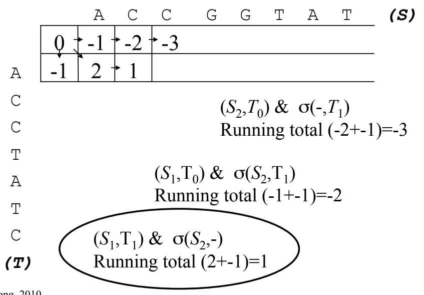


		A	C	C	G	G	T	A	T
	0	-1	-2	-3					
A	-1	2							
C									
C									
T									
A									
T									
C									

(S) (T)

Armstrong, 2010

Choose and record ‘best’ path



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

(S) (T)

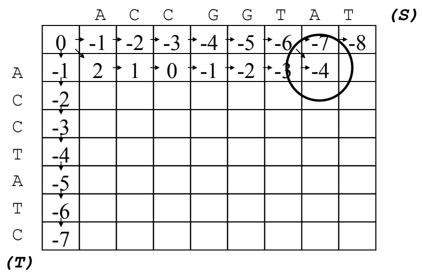
(S_2,T_0) & $\sigma(-,T_1)$
Running total (-2+-1)=-3

(S_1,T_0) & $\sigma(S_2,T_1)$
Running total (-1+-1)=-2

(S_1,T_1) & $\sigma(S_2,-)$
Running total (2+-1)=1

Armstrong, 2010

Continue….

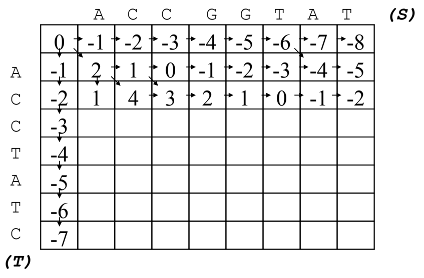


		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	
C	-2								
C	-3								
T	-4								
A	-5								
T	-6								
C	-7								

(S) (T)

Armstrong, 2010

Continue….

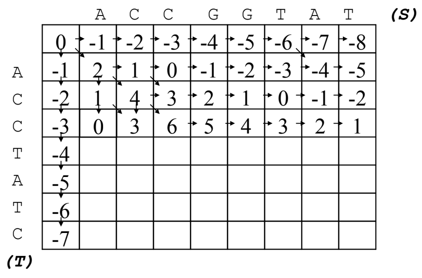


		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3								
T	-4								
A	-5								
T	-6								
C	-7								

(S) (T)

Armstrong, 2010

Continue….



		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4								
A	-5								
T	-6								
C	-7								

(S) (T)

Armstrong, 2010

Continue….

		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5									
T	-6									
C	-7									
(T)										

Armstrong, 2010

Continue….

		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5	-2	1	4	4	3	5	8	7	
T	-6									
C	-7									
(T)										

Armstrong, 2010

Continue….

		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5	-2	1	4	4	3	5	8	7	
T	-6	-3	0	3	3	3	5	7	10	
C	-7									
(T)										

Armstrong, 2010

Finally.

		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5	-2	1	4	4	3	5	8	7	
T	-6	-3	0	3	3	3	5	7	10	
C	-7	-4	-1	2	2	2	4	6	**9**	**= Score**
(T)										

Armstrong, 2010

Finally.

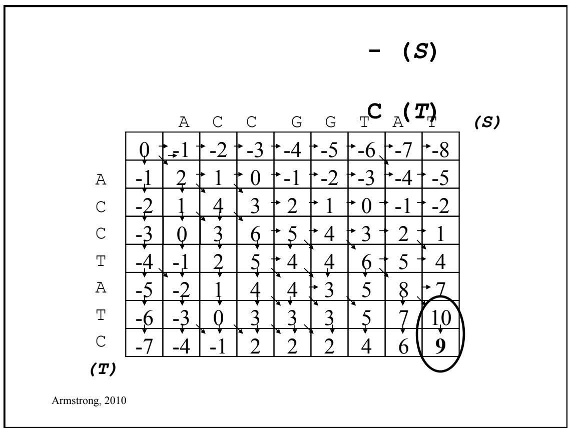
		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5	-2	1	4	4	3	5	8	7	
T	-6	-3	0	3	3	3	5	7	10	
C	-7	-4	-1	2	2	2	4	6	**9**	
(T)										

Armstrong, 2010

We recreate the alignment using by following the pointers back through the array to the origin

		A	C	C	G	G	T	A	T	*(S)*
	0	-1	-2	-3	-4	-5	-6	-7	-8	
A	-1	2	1	0	-1	-2	-3	-4	-5	
C	-2	1	4	3	2	1	0	-1	-2	
C	-3	0	3	6	5	4	3	2	1	
T	-4	-1	2	5	4	4	6	5	4	
A	-5	-2	1	4	4	3	5	8	7	
T	-6	-3	0	3	3	3	5	7	10	
C	-7	-4	-1	2	2	2	4	6	**9**	
(T)										

Armstrong, 2010

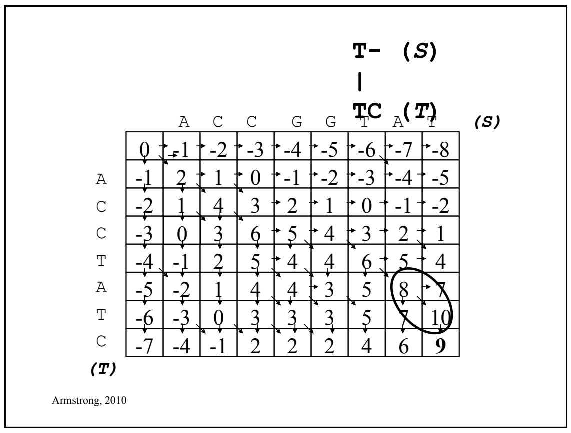


```
- (S)

C (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

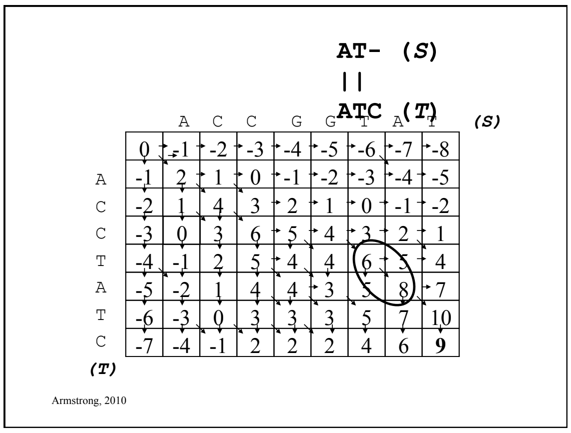
Armstrong, 2010



```
T- (S)
|
TC (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

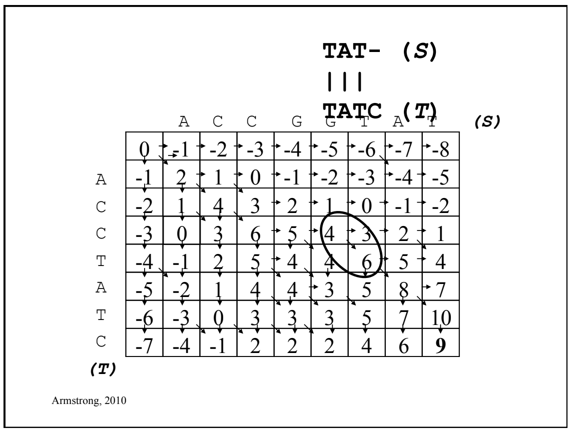
Armstrong, 2010



```
AT- (S)
||
ATC (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

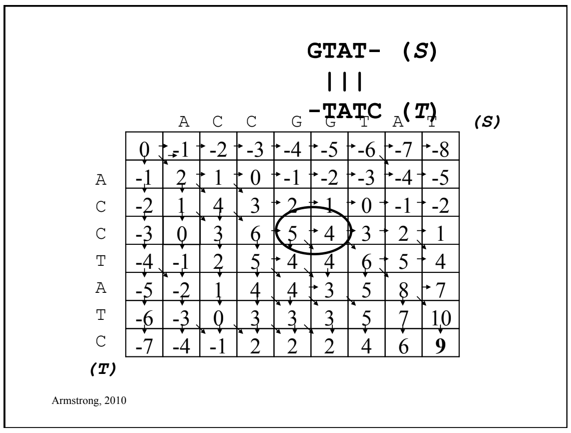
Armstrong, 2010



```
TAT- (S)
|||
TATC (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

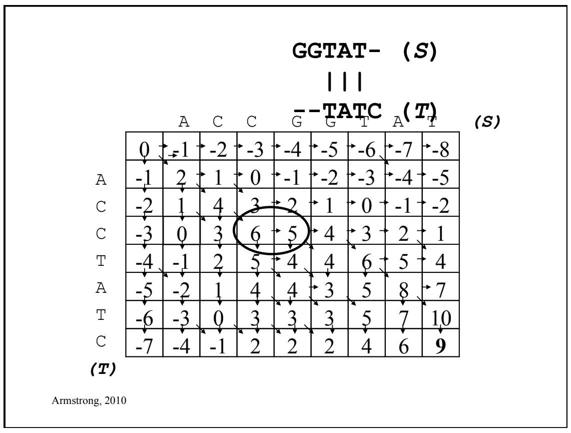
Armstrong, 2010



```
GTAT- (S)
 |||
-TATC (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

Armstrong, 2010



```
GGTAT- (S)
  |||
--TATC (T)
```

		A	C	C	G	G	T	A	T (S)
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**
(T)									

Armstrong, 2010

```
CGGTAT- (S)
|  |||
C--TATC (T)
```

(T) \ (S)		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**

```
CCGGTAT- (S)
||  |||
CC--TATC (T)
```

(T) \ (S)		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**

```
ACCGGTAT- (S)
|||  |||
ACC--TATC (T)
```

(T) \ (S)		A	C	C	G	G	T	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-1	2	1	0	-1	-2	-3	-4	-5
C	-2	1	4	3	2	1	0	-1	-2
C	-3	0	3	6	5	4	3	2	1
T	-4	-1	2	5	4	4	6	5	4
A	-5	-2	1	4	4	3	5	8	7
T	-6	-3	0	3	3	3	5	7	10
C	-7	-4	-1	2	2	2	4	6	**9**

Checking the result

```
ACCGGTAT- (S)
|||  |||
ACC--TATC (T)
```

- Our alignment considers ALL bases in each sequence
- 6 matches = 12 points, 3 gaps = -3 points
- Score = 9 confirmed.

A bit more formally..

Base conditions:

$$V(i,0) = \sum_{k=0}^{i} \sigma(S_k,-)$$

$$V(0,j) = \sum_{k=0}^{j} \sigma(-,T_k)$$

Recurrence relation: for $1<=i <= n$, $1<=j<=m$:

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + \sigma(S_i,T_j) \\ V(i-1,j) + \sigma(S_i,-) \\ V(i,j-1) + \sigma(-,T_j) \end{cases}$$

Time Complexity

- Each cell is dependant on three others and the two relevant characters in each sequence
- Hence each cell takes a constant time
- (n+1) x (m+1) cells
- Complexity is therefore O(nm)

Space Complexity

- To calculate each row we need the current row and the row above only.
- Therefore to get the score, we need O(n+m) space
- However, if we need the pointers as well, this increases to O(nm) space
- This is a problem for very long sequences
  - think about the size of whole genomes

Global alignment in linear space

- Hirschberg 1977 applied a ‘divide and conquer’ algorithm to Global Alignment to solve the problem in linear space.
- Divide the problem into small manageable chunks
- The clever bit is finding the chunks

dividing...

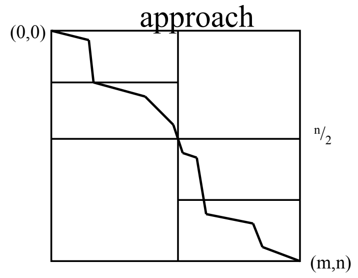
Compute matrix V(A,B) saving the values for $^n/_2$th row
- call this matrix F

Compute matrix $V(A^r,B^r)$ saving the values for $^n/_2$th row
- call this matrix B

Find column k so that the crossing point $(^n/_2,k)$ satisfies:
$F(^n/_2,k) + B(^n/_2,m-k) = F(n,m)$

Now we have two much smaller problems:
$(0,0) \to (^n/_2,k)$ and $(n,m) \to (^n/_2,m-k)$

Hirschberg’s divide and conquer approach



Complexity

- After applying Hirschberg’s divide and conquer approach we get the following:
  - Complexity O(mn)
  - Space O(min(m,n))
- For the proofs, see D.S. Hirschberg. (1977) Algorithms for the longest common subsequence problem. J. A.C.M 24: 664-667

OK where are we?

- The Needleman-Wunsch algorithm finds the optimum alignment and the best score.
  - NW is a dynamic programming algorithm
- Space complexity is a problem with NW
- Addressed by a divide and conquer algorithm
- What about local and ends-free alignments?

Smith-Waterman algorithm

- Between two sequences, find the best two subsequences and their score.
- We want to ignore badly matched sequence
- Use the same types of substitution matrix and gap penalties
- Use a modification of the previous dynamic programming approach.

Smith-Waterman algorithm

- If S_i matches T_j then $\sigma(S_i,T_j)$ >=0
- If they do not match or represent a gap then <=0

- Lowest allowable value of any cell is 0
- Find the cell with the highest value (*i,j*) and extend the alignment back to the first zero value
- The score of the alignment is the value in that cell
- A quick example if best...

min value of any cell is 0

		A	C	C	G	G	T	A	T	*(S)*
	0	0	0	0	0	0	0	0	0	
T	0									
T	0									
G	0									
T	0									
A	0									
T	0									
C	0									
(T)										

min value of any cell is 0

		A	C	C	G	G	T	A	T	*(S)*
	0	0	0	0	0	0	0	0	0	
T	0	0	0	0	0	0	2	1	2	
T	0	0	0	0	0	0	2	1	3	
G	0									
T	0									
A	0									
T	0									
C	0									
(T)										

min value of any cell is 0

		A	C	C	G	G	T	A	T	*(S)*
	0	0	0	0	0	0	0	0	0	
T	0	0	0	0	0	0	2	1	2	
T	0	0	0	0	0	0	2	1	3	
G	0	0	0	0	2	2	1	1	2	
T	0	0	0	0	1	1	4	3	3	
A	0	2	1	0	0	0	3	6	5	
T	0	1	1	0	0	0	2	5	8	
C	0	0	3	4	3	2	1	4	7	
(T)										

Find biggest cell and map alignment from there

		A	C	C	G	G	T	A	T	*(S)*
	0	0	0	0	0	0	0	0	0	
T	0	0	0	0	0	0	2	1	2	
T	0	0	0	0	**0**	0	2	1	3	
G	0	0	0	0	2	**2**	1	1	2	
T	0	0	0	0	1	1	**4**	3	3	
A	0	2	1	0	0	0	3	**6**	5	
T	0	1	1	0	0	0	2	5	**8**	
C	0	0	3	4	3	2	1	4	7	
(T)										

GTAT(*S*)

||||

GTAT(*T*)

		A	C	C	G	G	T	A	T	***(S)***
	0	0	0	0	0	0	0	0	0	
T	0	0	0	0	0	0	2	1	2	
T	0	0	0	0	**0**	0	2	1	3	
G	0	0	0	0	2	**2**	1	1	2	
T	0	0	0	0	1	1	**4**	3	3	
A	0	2	1	0	0	0	3	**6**	5	
T	0	1	1	0	0	0	2	5	**8**	
C	0	0	3	4	3	2	1	4	7	
(T)										

Smith-Waterman cont'd

- Complexity
  - Time is O(nm) as in global alignments
  - Space is O(nm) as in global alignments

  - A mod of Hirschbergs algorithm allows O(n+m) (n +m) as two rows need to be stored at a time instead of one as in the global alignment.

A bit more formally..

Base conditions: $\forall i,j.\ V(i,0) = 0,\ V(0,j) = 0$

Recurrence relation: for $1 \le i \le n$, $1 \le j \le m$:

$$V(i,j) = \max \begin{cases} 0 \\ V(i-1,j-1) + \sigma(S_i,T_j) \\ V(i-1,j) + \sigma(S_i,-) \\ V(i,j-1) + \sigma(-,T_j) \end{cases}$$

Compute i^* and j^* $V(i^*,j^*) = \max_{1<=i<=n,1<=j<=m} V(i,j)$

Ends-free alignment

- Find the overlap between two sequences such start the start of one overlaps is in the alignment and the end of the other is in the alignment.
- Essential to DNA sequencing strategies.
  - Building genome fragments out of shorter sequencing data.
- Another variant of the Global Alignment Problem

Ends-free alignment

- Set the initial conditions to zero weight
  - allow indels/gaps at the ends without penalty
- Fill the array/table using the same recursion model used in global/local alignment
- Find the best alignment that ends in one row or column
  - trace this back

min value row0 & col0 is 0

		G	T	T	A	C	T	G	T	***(S)***
	0	0	0	0	0	0	0	0	0	
C	0	-1	-1	-1	-1	2	1	0	-1	
T	0	-1	1	1	0	1	4	3	2	
G	0	2	1	0	0	0	3	6	5	
T	0	1	4	3	2	1	2	5	8	
A	0	0	3	3	5	4	3	4	7	
T	0	-1	2	5	4	4	6	5	6	
C	0	0	1	4	4	6	5	5	5	
(T)										

Find the best 'end' point in an end col or row

		G	T	T	A	C	T	G	T	*(S)*
	0	0	0	0	0	0	0	0	0	
C	0	-1	-1	-1	-1	2	1	0	-1	
T	0	-1	1	1	0	1	4	3	2	
G	0	2	1	0	0	0	3	6	5	
T	0	1	4	3	2	1	2	5	8	
A	0	0	3	3	5	4	3	4	7	
T	0	-1	2	5	4	4	6	5	6	
C	0	0	1	4	4	6	5	5	5	
(T)										

Armstrong, 2010

Trace the best route from there to the origin and end

		G	T	T	A	C	T	G	T	*(S)*
	0	0	0	0	0	0	0	0	0	
C	0	-1	-1	-1	-1	2	1	0	-1	
T	0	-1	1	1	0	1	4	3	2	
G	0	2	1	0	0	0	3	6	5	
T	0	1	4	3	2	1	2	5	8	
A	0	0	3	3	5	4	3	4	7	
T	0	-1	2	5	4	4	6	5	6	
C	0	0	1	4	4	6	5	5	5	
(T)										

Armstrong, 2010

```
GTTACTGT---(S)
    ||||
----CTGTATC(T)
```

		G	T	T	A	C	T	G	T	*(S)*
	0	0	0	0	0	0	0	0	0	
C	0	-1	-1	-1	-1	2	1	0	-1	
T	0	-1	1	1	0	1	4	3	2	
G	0	2	1	0	0	0	3	6	5	
T	0	1	4	3	2	1	2	5	8	
A	0	0	3	3	5	4	3	4	7	
T	0	-1	2	5	4	4	6	5	6	
C	0	0	1	4	4	6	5	5	5	
(T)										

Armstrong, 2010

A bit more formally..

Base conditions: $\forall i,j.\ V(i,0) = 0,\ V(0,j) = 0$

Recurrence relation: for $1 <= i <= n,\ 1 <= j <= m$:

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + \sigma(S_i,T_j) \\ V(i-1,j) + \sigma(S_i,-) \\ V(i,j-1) + \sigma(-,T_j) \end{cases}$$

Search for i^* such that: $V(i^*,m) = \max_{1<=i<=n,m} V(i,j)$

Search for j^* such that: $V(n,j^*) = \max_{1<=j<=n,m} V(i,j)$

Define alignment score $V(S,T) = \max \begin{cases} V(n,j^*) \\ V(i^*,m) \end{cases}$

Armstrong, 2010

Summary so far...

- Dynamic programming algorithms can solve global, local and ends-free alignment
- They give the optimum score and alignment using the parameters given
- Divide and conquer approaches make the space complexity manageable for small-medium sized sequences

Armstrong, 2010

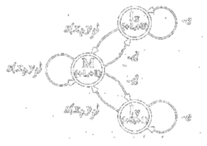
Dynamic Programming Issues

- For huge sequences, even linear space constraints are a problem.
- We used a very simple gap penalty
- The Affine Gap penalty is most commonly used.
  - Cost to open a gap
  - Cost to extend an open gap
- Need to track and evaluate the 'gap' state in the array

Armstrong, 2010

Tracking the gap state

- We can model the matches and gap insertions as a finite state machine:

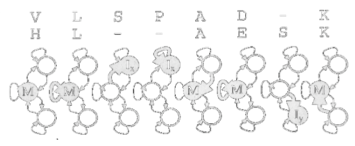


Taken from Durbin, chapter 2.4

Armstrong, 2010

Tracking the gap state

- Working along the alignment process...



Taken from Durbin, chapter 2.4

Armstrong, 2010

Real Life Sequence Alignment

- When searching multiple genomes, the sizes still get too big!
- Several approaches have been tried:
- Use huge parallel hardware:
  - Distribute the problem over many CPUs
  - Very expensive
- Implement in Hardware
  - Cost of specialist boards is high
  - Has been done for Smith-Waterman on SUN

Armstrong, 2010

Real Life Sequence Alignment

- Use a Heuristic Method
  - Faster than ‘exact’ algorithms
  - Give an approximate solution
  - Software based therefore cheap

- Based on a number of assumptions:

Armstrong, 2010

Assumptions for Heuristic Approaches

- Even linear time complexity is a problem for large genomes
- Databases can often be pre-processed to a degree
- Substitutions more likely than gaps
- Homologous sequences contain a lot of substitutions without gaps which can be used to help find start points in alignments

Armstrong, 2010

Conclusions

- Dynamic programming algorithms are expensive but they give you the optimum alignment and exact score
- Choice of GAP penalty and substitution matrix are critically important
- Heuristic approaches are generally required for high throughput or very large alignments

Armstrong, 2010