Department of Biology Penn University of Pennsylvania

Introduction to Bioinformatics

ITMAT Bioinformatics

Topic Six

Nucleic Acid Alignment: Local and Multiple

Lecturer
Gregory R. Grant

Fall 2023

Teaching Assistants

Gregory R. Grant

Genetics Department

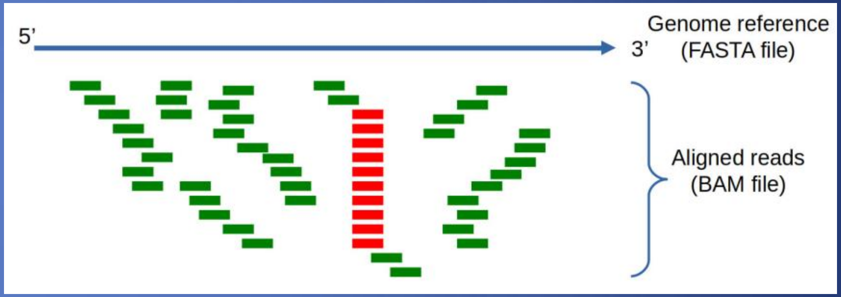
ggrant@pennmedicine.upenn.edu

ITMAT Bioinformatics Laboratory

University of Pennsylvania

Local Alignment

- Global alignment only works when the two sequences are related end-to-end.
- But more often one sequence is a subsequence of another.
  - A gene is a subsequence of a chromosome.
  - An exon is a subsequence of a transcript.
  - A sequencing read is just a small piece of a gene or a chromosome.



- If you want to locate a short 100-base sequencing read in a 100 million base chromosome, Needleman-Wunch would be useless.
  - Not just because of efficiency, but because a global alignment makes no sense in this context.

Local Alignment

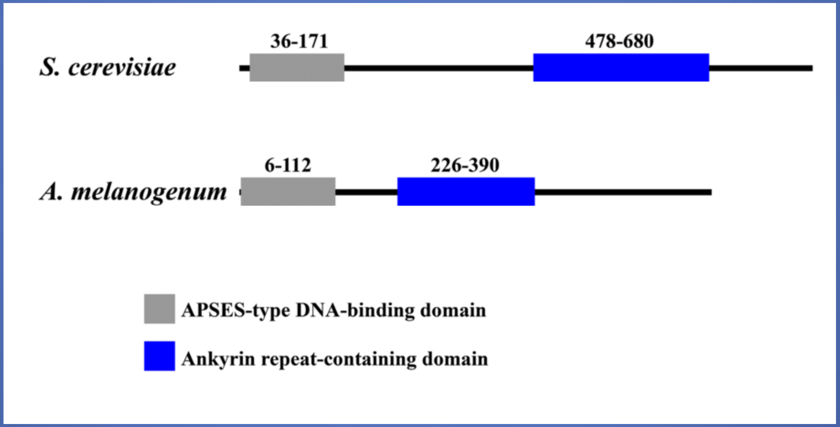
A more flexible algorithm would be one that doesn’t require all bases of both sequences to be involved in the alignment.

Even so, the problem of finding a short read in a long chromosome is so particular that it requires its own algorithm, which we will look at later.

For now, we will maintain our focus on aligning two relatively short sequences like two proteins or two RNAs.

Functional Domains

- Typically, a protein has a few important stretches of amino acids that perform some vital function (called ‘functional domains’) and the rest of the amino acids separating these stretches are just backbone.


36-171
478-680
S. cerevisiae
6-112
226-390
A. melanogenum
APSES-type DNA-binding domain
Ankyrin repeat-containing domain

- This figure shows a gene with two functional domains
  - Those are the only conserved part of the gene’s sequence between distantly related species of yeast.
  - Due to how DNA can shuffle, the parts in between may not even be related.

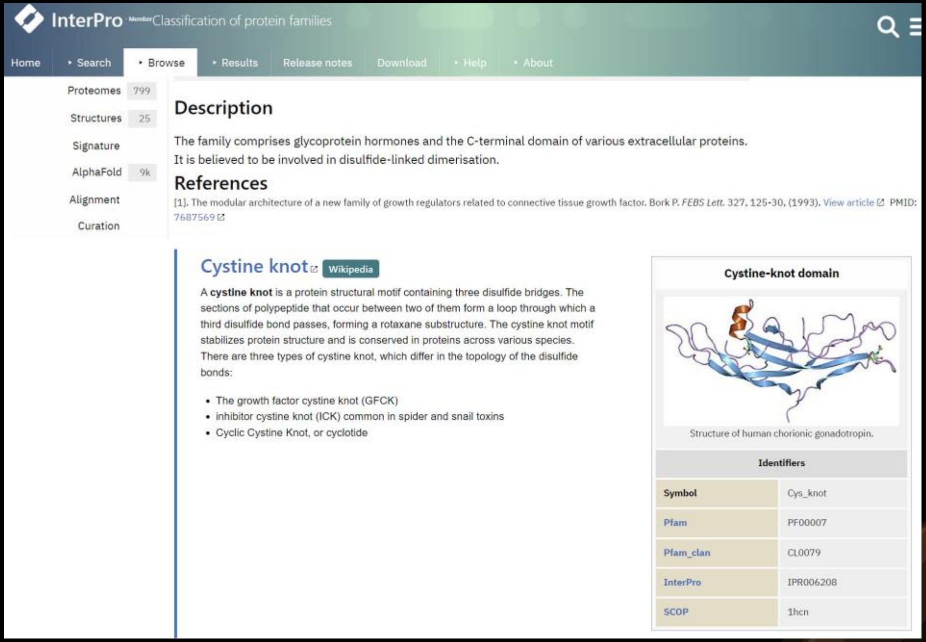
Local Alignment

- It could be that the stretches of sequence between the functional domains has drifted so much that their similarity is down to roughly 25%.
- At that point alignment between these segments is impossible.
  - Relatedness needs to be established by the conserved domains.
- We need an algorithm where we can give both sequences and it will find just the parts that align.
  - We call this "local alignment"

Pfam

- Some argue that the fundamental units of interest should not be whole genes or proteins, but rather “functional domains”
  - These combine in different ways to make full length proteins.

- This is the perspective taken by Pfam where proteins are classified by their functional (and structural) domains.
  - The domains are modelled by Hidden Markov Models, which we will talk about briefly later in this lecture.



Description

The family comprises glycoprotein hormones and the C-terminal domain of various extracellular proteins.
It is believed to be involved in disulfide-linked dimerisation.

References

[1]. The modular architecture of a new family of growth regulators related to connective tissue growth factor. Bork P. *FEBS Lett.* 327, 125-30, (1993). View article PMID: 7687569

Cystine knot

A **cystine knot** is a protein structural motif containing three disulfide bridges. The sections of polypeptide that occur between two of them form a loop through which a third disulfide bond passes, forming a rotaxane substructure. The cystine knot motif stabilizes protein structure and is conserved in proteins across various species. There are three types of cystine knot, which differ in the topology of the disulfide bonds:

- The growth factor cystine knot (GFCK)
- inhibitor cystine knot (ICK) common in spider and snail toxins
- Cyclic Cystine Knot, or cyclotide

Cystine-knot domain

Structure of human chorionic gonadotropin.

Identifiers	
Symbol	Cys_knot
Pfam	PF00007
Pfam_clan	CL0079
InterPro	IPR006208
SCOP	1hcn

Complexity

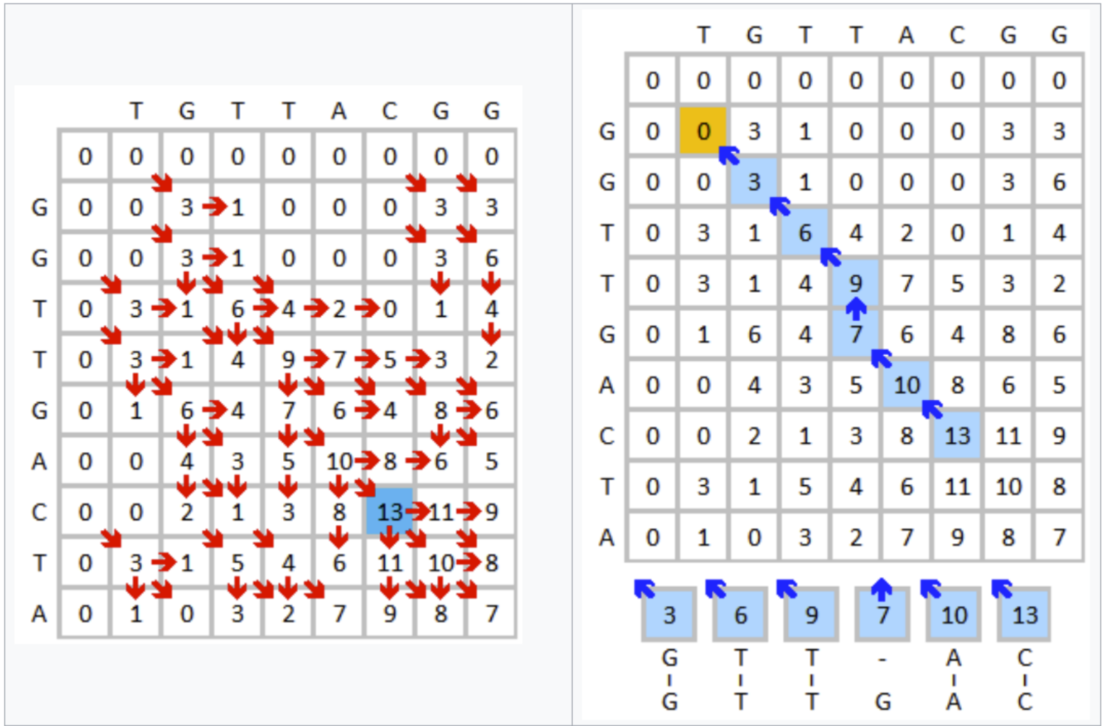
- There are orders of magnitude more possible local alignments between two sequences than possible global alignments.
  - All the global ones also count as local, plus a whole lot more.
- Yet there are still algorithms that achieve n^2 complexity.
- That does not *necessarily* mean they run at the same speed.
  - Just because two algorithms are both big O of n^2 that does not mean they run at the same speed.
  - If the constant that bounds the limit is twice as big, then the run time is still twice as long.
  - But the constant does not depend on n and that's the important thing.
- Yet, the local alignment algorithms actually *do* run at the same speed as global.
  - Because they require filling in a very similar table.

Smith-Waterman

- The workhorse of ***optimal*** pairwise local alignment is the Smith-Waterman algorithm, first published in 1981.
- Smith-Waterman is very similar to Needleman-Wunsch.
- There are two main differences.
  - First, if the maximum of the three scores for a cell is negative, then we put 0 in the cell, not the actual max score.
  - Second, you don't (necessarily) start the traceback in the bottom right corner and you don't (necessarily) end in the top left. See next slide.

Smith-Waterman

- Assume we have a scoring scheme.
  - Larger numbers are better.
- The difference with Needleman-Wunsch traceback is that instead of starting at the bottom right corner, you start in the cell with the highest number and traceback only until you reach a cell that is 0 or negative.
- The initialization of the first row and column are all zeros.
- This example, copied from Wikipedia, illustrates the process.



		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Finished scoring matrix (the highest score is in blue)

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

```
 3   6   9   7   10   13
 G   T   T   -   A    C
 |   |   |       |    |
 G   T   T   G   A    C
```

Traceback process and alignment result

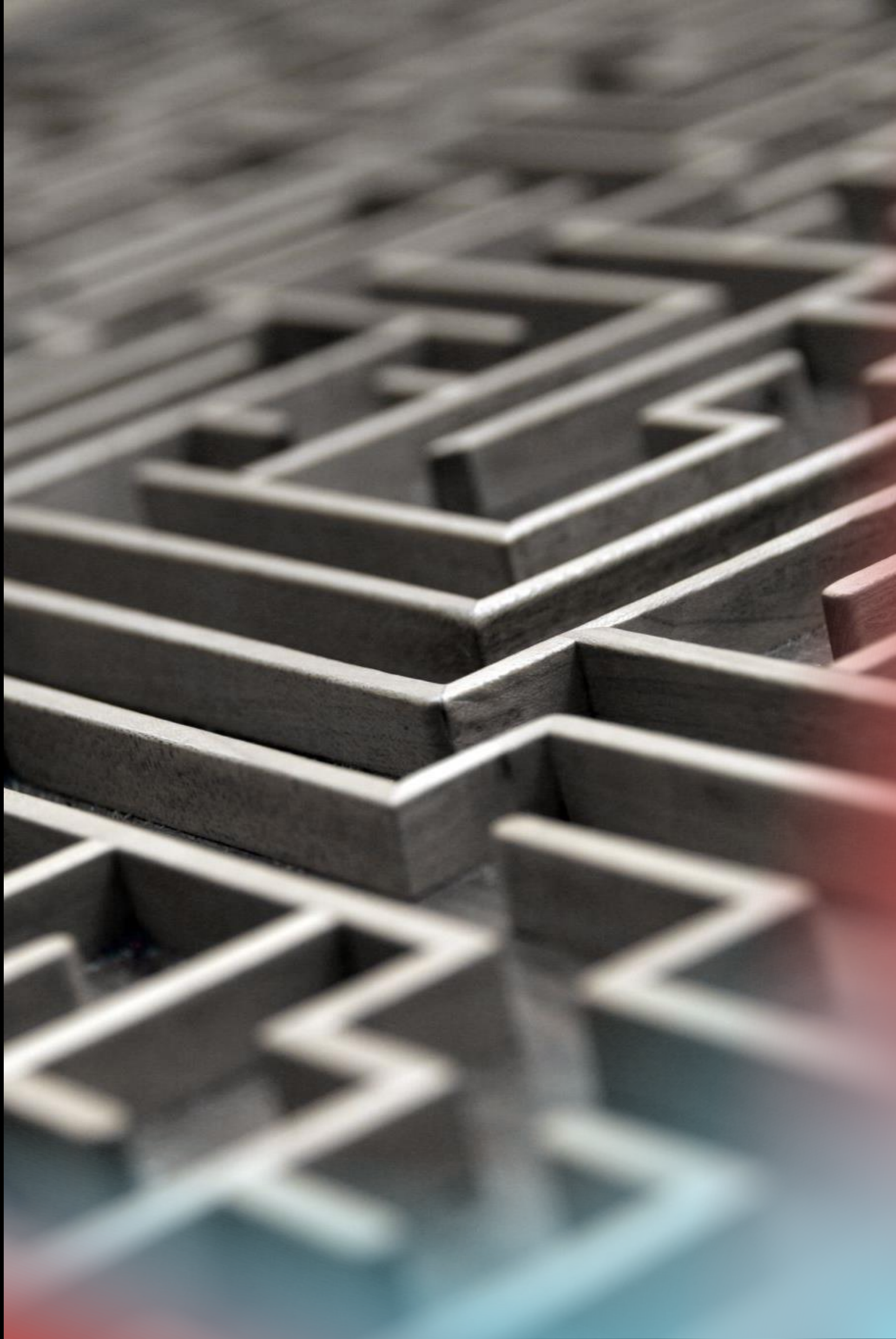
Greedy Algorithms

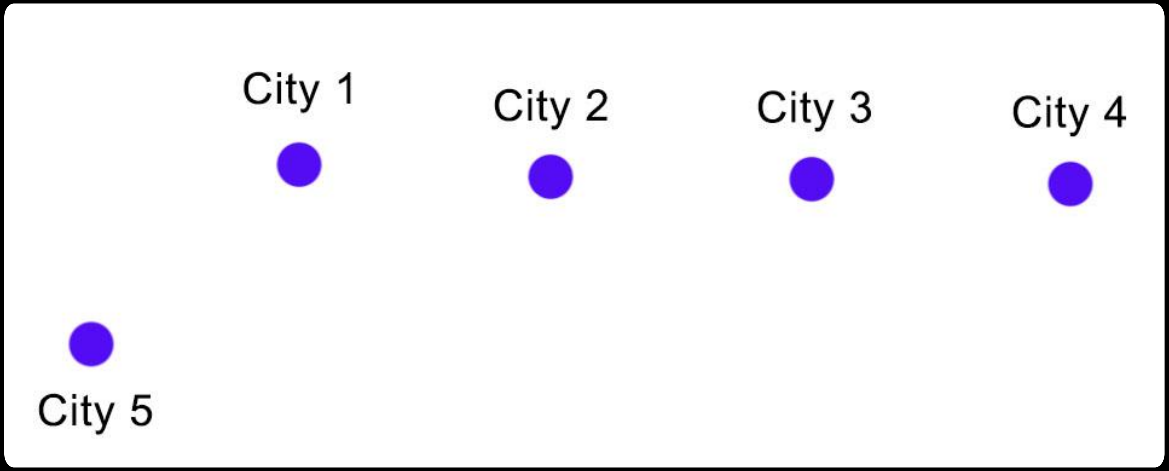
- We’re searching a (greater than) exponential search space in polynomial operations.

- Why does this work?

- It works because alignment is a special class of problem which can be broken down into a finite sequence of problems where:

1. Finding the optimal solution at each step along the way yields a global optimal solution.
2. Given the optimal solution at step *n* there’s a simple procedure for finding the optimal solution for step *n+1.*



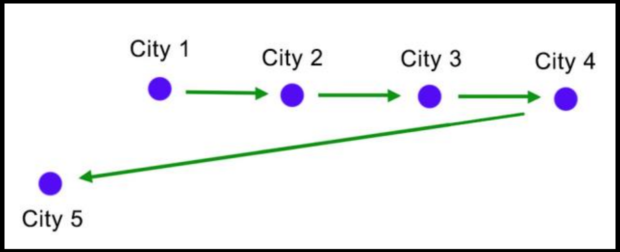


Greedy Algorithms

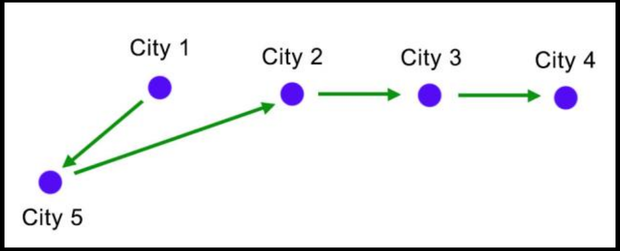
- Just because we construct an iterative algorithm that appears to do the optimal thing at each step, that does not mean it will be 'globally' optimal.
- It is the case with sequence alignment, but there's a mathematical proof we must do to know that for sure.
- For an example where a greedy strategy failes, consider the travelling salesman problem of finding the shortest route to visit *n* cities.

Greedy Algorithms

- Suppose you must start in City 1.
- The greedy algorithm is as follows:
  - Travel to the nearest city at each step.
- This algorithm does not guarantee the shortest route overall.



This is the route the greedy algorithm would return



But this is the shortest route

Nucleic Acid Alignment

- There are many applications of nucleic acid alignment.
  - In different contexts it requires very different considerations.   There's can be no "one algorithm fits all"
  - Here are some examples to illustrate how varied the problem is.

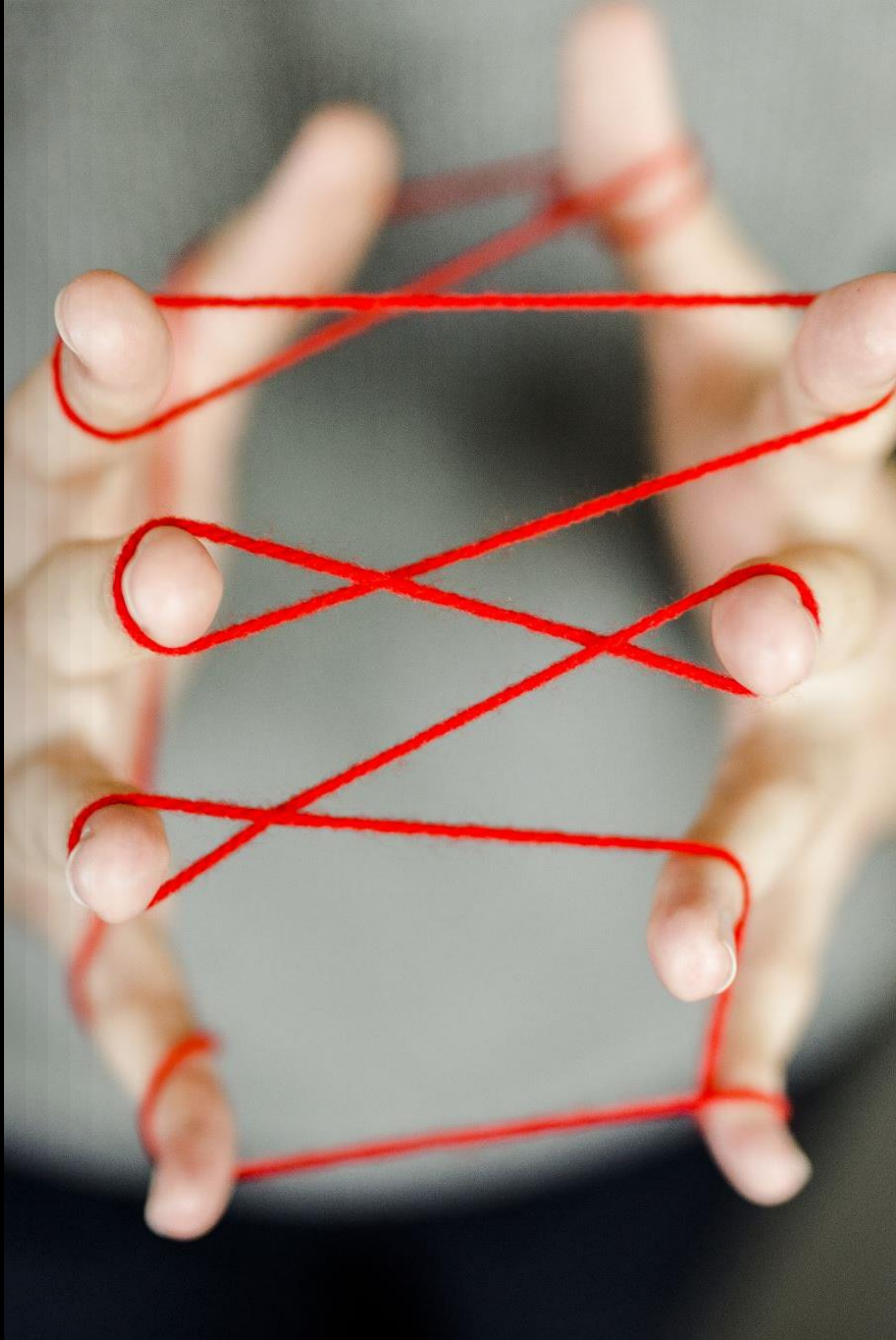
1. Align short (gene length or shorter sequences) to each other, to infer their evolutionary relationship.
   - In this case identity may be quite low.
   - Use this to search a database to find related sequences to a sequence in hand.
2. Align sequenced fragments of a gene or chromosome to each other, to assemble them into "contigs".
   - In this case identity should be quite high, near 100%
3. Find the location of a gene in a genome.
   - We may have the RNA of an unknown gene and want to find it in the DNA.
   - In this case identity should be high.
4. Comparative Genomics
   - Align entire chromosomes from different species.

Low Complexity Sequence

- Low complexity sequence is sequence with short patterns.
- A simple example is:

  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

- Suppose you are searching for a gene which had this sequence of A's in it and this stretch of A's was all that aligned.
  - We would not be very convinced the sequences are related, even though it's 30 bases long, so should not occur more than once in random sequence, it's obviously not random.



Modeling DNA/RNA with Random Sequence

- We tend to model DNA and RNA by random independently and identically distributed bases.
  - Imagine rolling a four-sided die a bunch of times in a row.
- We use this “random” model to assess the significance of things.
  - You’ll see, this will be a recurring theme.
- In random sequence, a string of 30 A’s in a row would be unlikely.
  - But in DNA it happens frequently, because DNA is not random.
- Therefore, we must be careful when using this model to assess likelihoods.

Random Sequence

Suppose the problem at hand is to find the location of a small sequence *S* within a large sequence *G*.

- 'S' for sequence, 'G' for genome.

How many sequences are there of length 3?

- $4^3 = 64$

If *S* has length 3 and *G* is a random sequence of length 100, what is the probability of finding *S* in *G*?

- To 3 decimal places, it's 0.786

Random Sequence

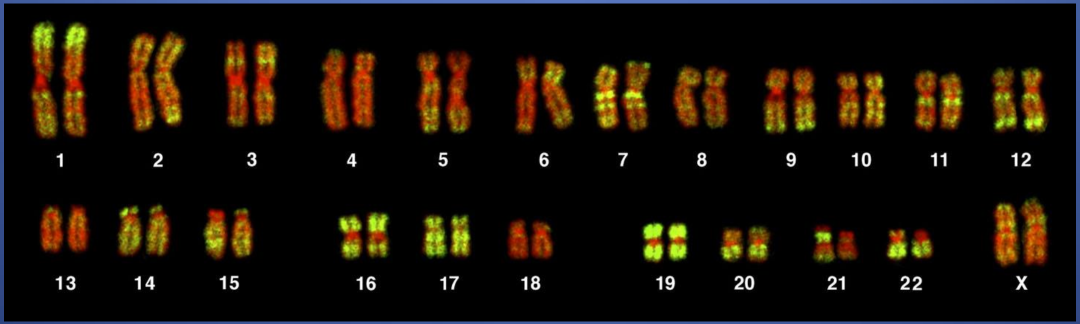
Given the human genome is 3.4 Gb, how long does a subsequence need to be in order to be unlikely to be found just by chance?

- There are 1,048,576 sequences of length 10.
- There are 4,294,967,296 sequences of length 16.
- There are 1,099,511,627,776 sequences of length 20.

So somewhere between 16 and 20.

Repeat Sequence

- Nucleic acid sequence as found in nature is not random.
  - It is highly structured.
  - And it is full of so-called “repeats” and “low complexity sequence”.
- Some elements in the genome like to duplicate themselves, ALU for example, a sequence of about 300 bases that occurs over 1,000,000 times in the human genome.



Human genome stained for ALU elements

P-Values

- An alignment is not interesting if we would not be surprised to find it between two unrelated sequences of the same sizes.

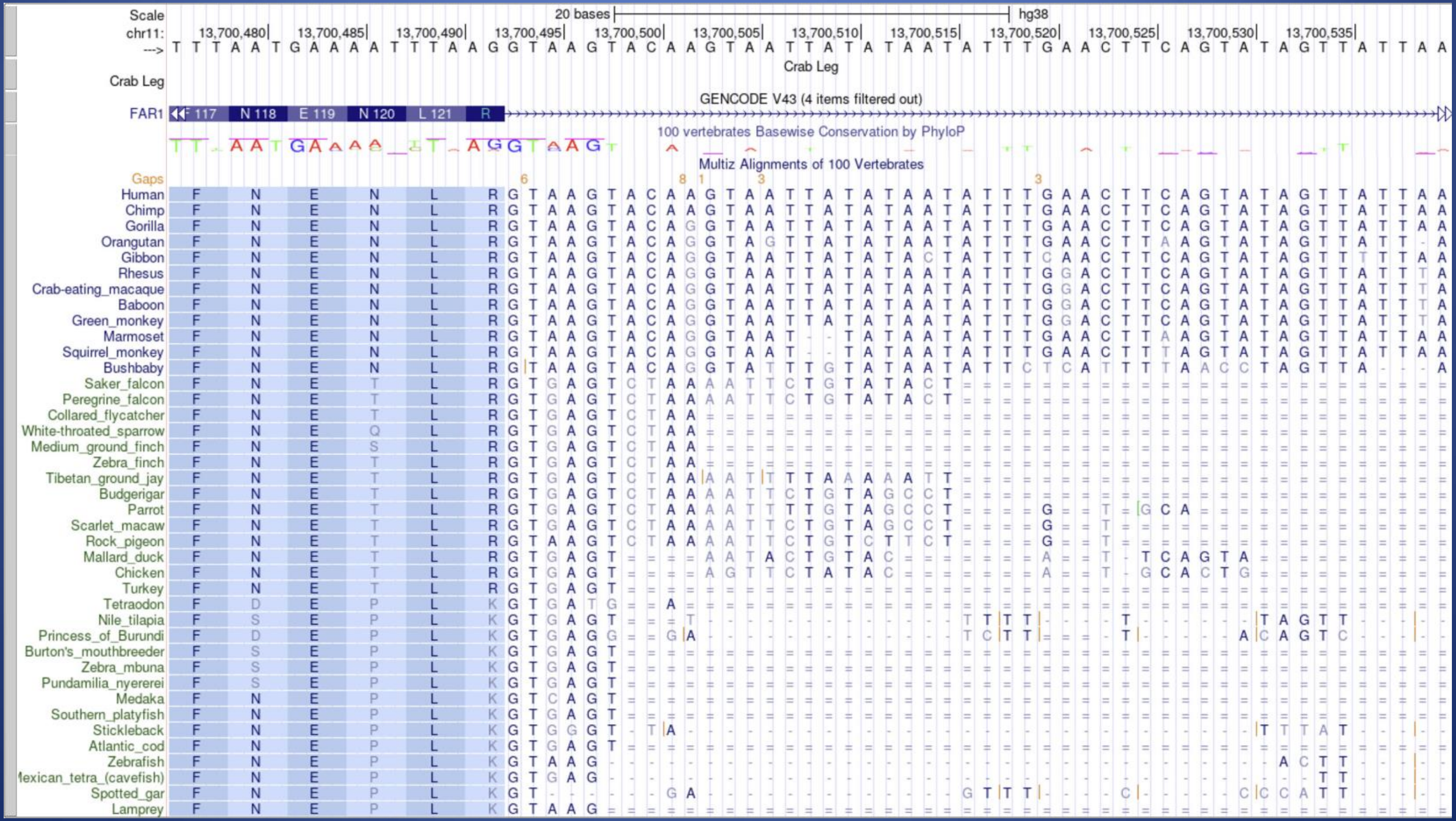
- In the age of big data, alignments with high scores happen if you give them enough chances to happen.

- To handle this, we will require *p*-values.
  - Which will require a null probabilistic model of sequence.
  - That's where random sequence comes in.

- When we do this, it will be particularly important to handle repeat and low-complexity sequence properly.
  - Otherwise, an alignment of a stretch of 25 A's in a row would be considered highly significant.

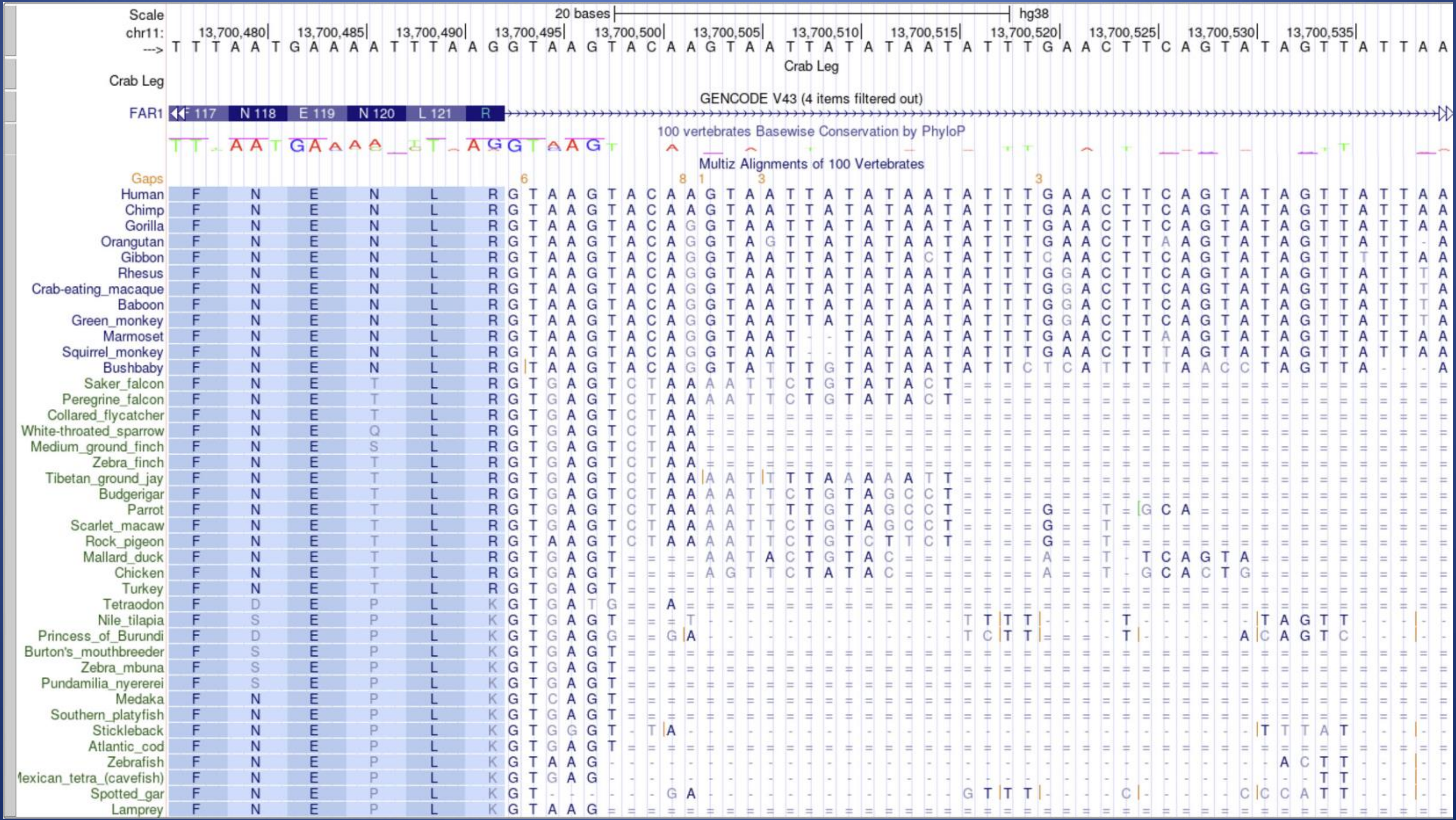
Multiple Alignment

- If you zoom in far enough on the conservation track, it shows a multiple sequence alignment across many species.



Multiple Alignment

- You may never realize how important those first two nucleotide bases of the intron are until you see this.
  - Nothing else in the intron is conserved across primates, fish and birds.



Algorithms

- Let’s try to lift the concepts developed for pairwise alignment to the problem of multiple alignment.
- First off, we need a scoring scheme.
- How should that work when we have multiple sequences?
- To build some intuition, let’s first consider three sequences aligned together. Could set:
  - +1 if all three are equal at a location.
  - -1 if two are equal and one is different
  - -2 if all three are different.
- That’s at least natural.
  - Might get complicated to extend this scoring scheme to many sequences.

Sum-of-Pairs Score (one position)

- Instead of scoring all possible N-tuples at a position, we can just sum all pairs using the usual pairwise-alignment score.
- Suppose $s(x, y)$ is a pairwise alignment scoring function, with linear gap penalty, so one of x or y could be an indel.
- For example:

$$s(x, y) = \begin{cases} +1 \text{ if } x = y \\ -1 \text{ if } x \neq y \\ -1 \text{ if one of } x \text{ or } y \text{ is an indel} \end{cases}$$

- Then

$$s(a_1, a_2, \ldots, a_n) = \sum_{i \neq j} s(a_i, a_j)$$

Sum-of-Pairs Score (full alignment)

- The function *s* on the previous slide is to score on position of a multiple sequence alignment.
- To score the entire alignment you must sum *s* over all positions.
- This score works for any number of sequences.

EXAMPLE

$$\begin{array}{ccccc} A & T & C & A & G \\ A & G & T & - & G \\ A & G & G & A & G \end{array}$$

Position 1: $\sum_{i \neq j} s(a_i, a_j) = s(A, A) + s(A, A) + s(A, A) = 3$

Position 2: $\sum_{i \neq j} s(a_i, a_j) = s(T, G) + s(T, G) + s(G, G) = -1$

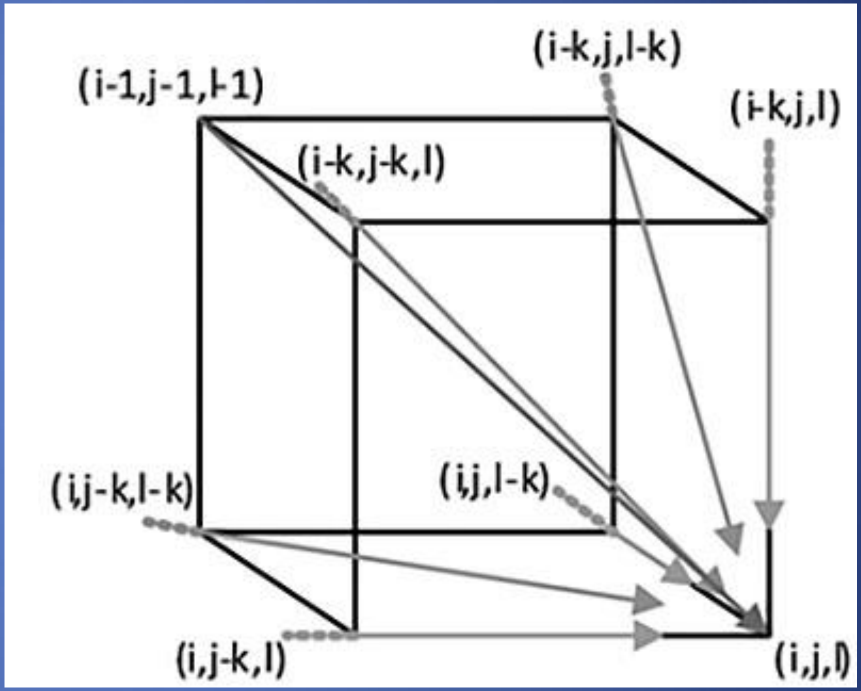
Position 3: $\sum_{i \neq j} s(a_i, a_j) = s(C, T) + s(C, G) + s(T, G) = -3$

Position 4: $\sum_{i \neq j} s(a_i, a_j) = s(A, -) + s(A, A) + s(-, A) = -1$

Position 5: $\sum_{i \neq j} s(a_i, a_j) = s(G, G) + s(G, G) + s(G, G) = 3$

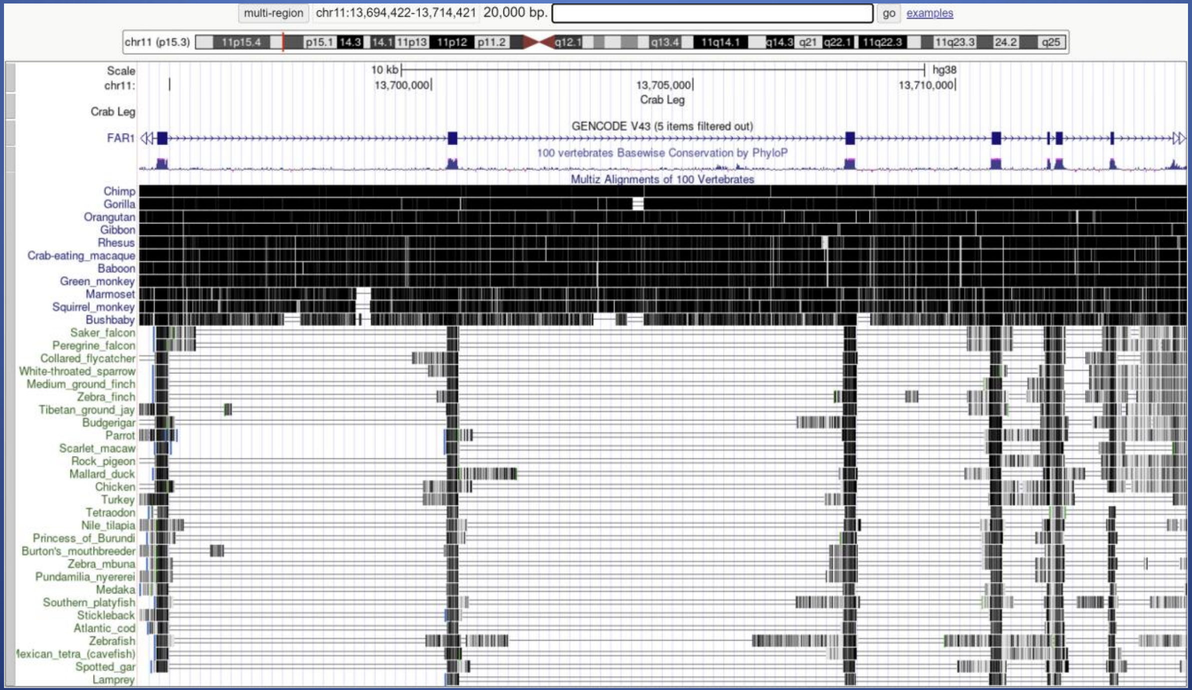
Smith-Waterman

- With any scoring scheme, one could generalize Smith-Waterman to find a local multiple alignment with optimal score.
- If there are *N* sequences, this would involve filling in all the cells of an *N*-dimensional table.
- Then if aligning *N* sequences of length *m*, the complexity of this algorithm is $O(m^N)$.
- That quickly gets out of hand.



Many Long Sequences

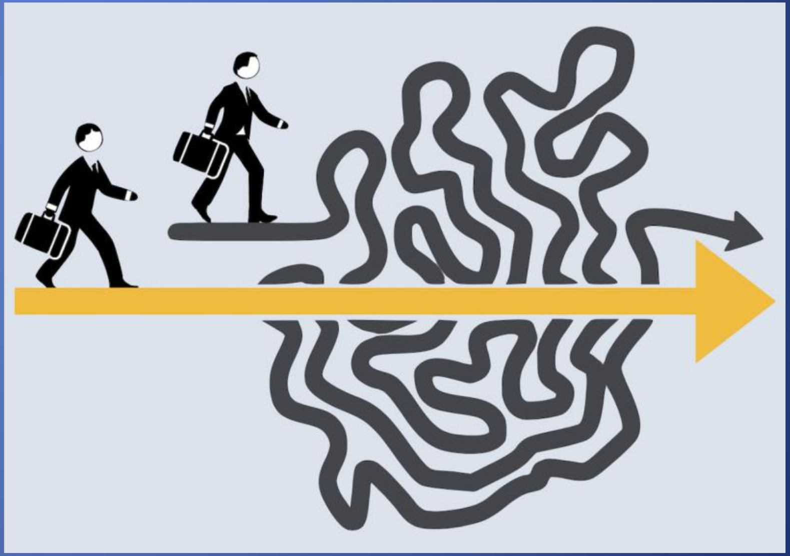
- The multiple alignment shown in the genome browser involves large *N* and *m*.
- Here for example is a 20,000 base sequence of genome aligned across 39 species, as shown in the conservation track.
  - This was not done with Smith-Waterman, or any other method that guarantees to find an optimal scoring alignment.



Heuristic Approaches

- A heuristic approach is one that does not guarantee to find the optimal.
  - This is different from an “approximation algorithm” because those give you a handle on how far the solution is from optimal.

- In heuristics, you have no handle on that.
  - But they can still be very useful when no fast alternative can be found.



Multiple Alignment Heuristics

Several completely different approaches have been taken to this problem.

Some examples are:

- Hidden-Markov Models
- Gibbs Sampling
- Tree-Guided Progressive Alignment
- Maximum Parsimony

We will focus on one of these that does not involve heavy machinery.

Progressive Alignment

- This is a strategy for building multiple alignments that starts by aligning two of the sequences.
- Then it aligns a third sequence to the pairwise alignment.
  - We'll have to talk about how that's done, but it's relatively straightforward.
- Then one by one it adds sequences to the multiple alignment.

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

Order Matters

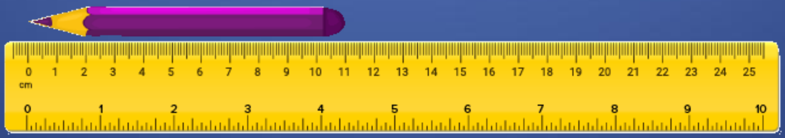
- Progressive alignment works best when you start with the two most similar sequences and align them first.
- Then add the third sequence that is closest to the alignment of the first two.
- Then add the fourth sequence that is closest to the alignment of the first three.
- Etc.

Distance Between Sequences

- To define "close" we must have some concept of "distance" between two (related) sequences.
- Let S_1 and S_2 be two sequences.
- Suppose we have a pairwise scoring scheme $s(S_1 , S_2)$.
  - First, align the two sequences S_1 and S_2 with Needleman-Wunch to find an alignment with optimal (highest) score.
- Now define the distance between S_1 and S_2 to be:

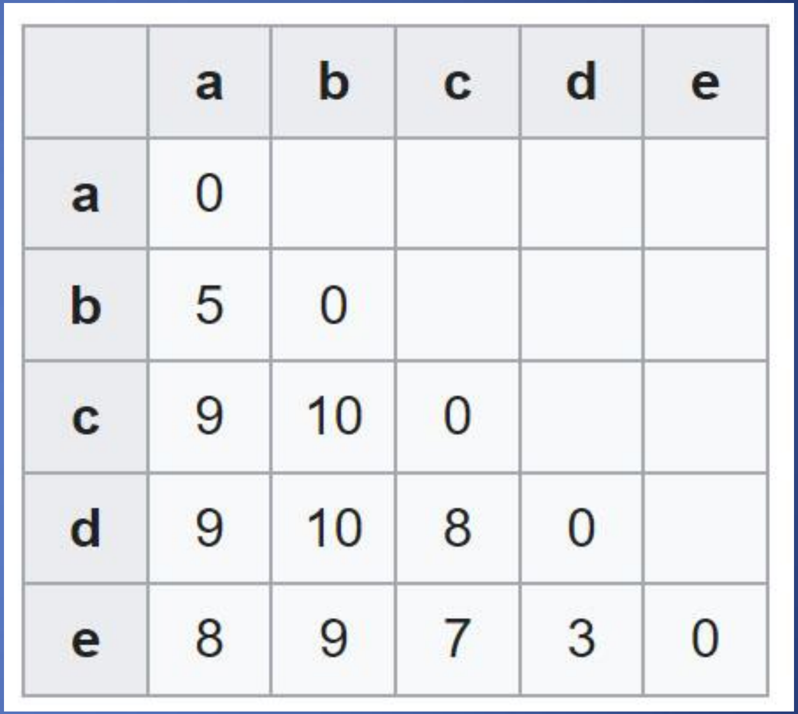
  $d(S_1 , S_2)$ = number of mismatches/indels

  In the optimal alignment
  - We still needed Needelman-Wunch to align them, but we do not use the score as distance.
- This behaves like a distance more than s does, because:
  1. d is always positive
  2. $d(S_1 , S_2) = 0$ if and only if $S_1 = S_2$.
  3. The more distantly related the sequences, the bigger d is.



Distance Matrix

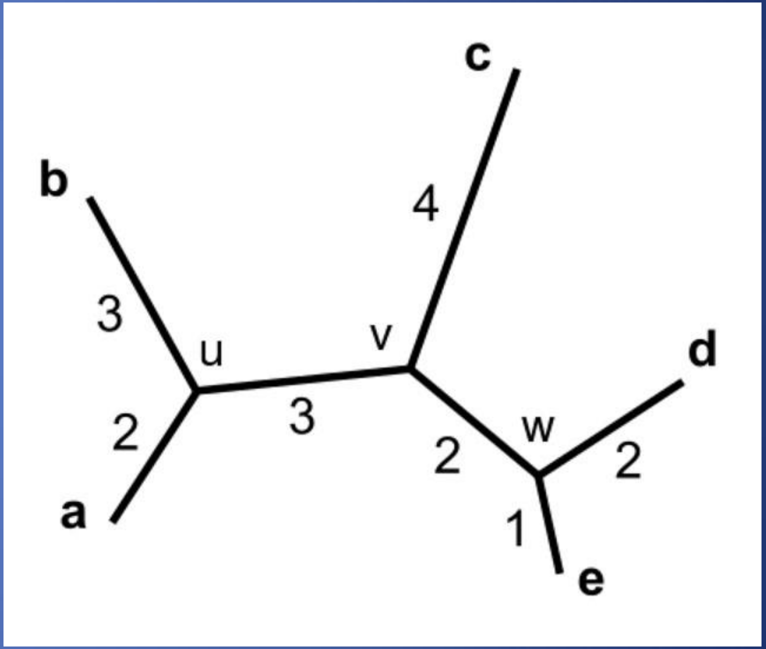
- Calculating the pairwise distance between all pairs of sequences gives us a matrix of distances.
- For example, suppose we have five sequences *a, b, c, d, e.*
  - The matrix could look like the one shown here.



	a	b	c	d	e
a	0				
b	5	0			
c	9	10	0		
d	9	10	8	0	
e	8	9	7	3	0

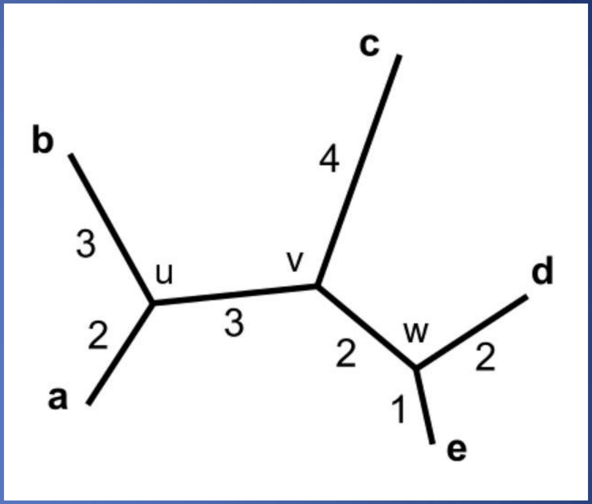
Phylogenetic Tree

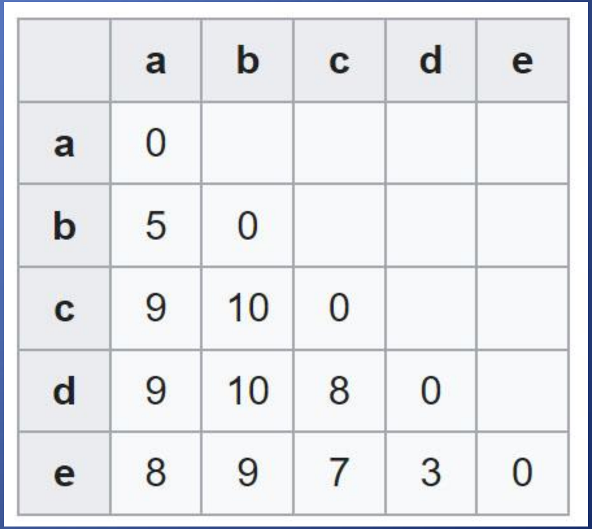
- Suppose we had the phylogenetic tree showing the evolutionary relationship between the five sequences.
  - This is called an “unrooted” tree, we do not know where inside this tree the ancestral species sits, nor the direction of evolution along the internal branches.
  - There are three inferred ancestral “sequences” *u, v* and *w.*
  - The numbers represent evolutionary distance, in some units.



Phylogenetic Tree Distances

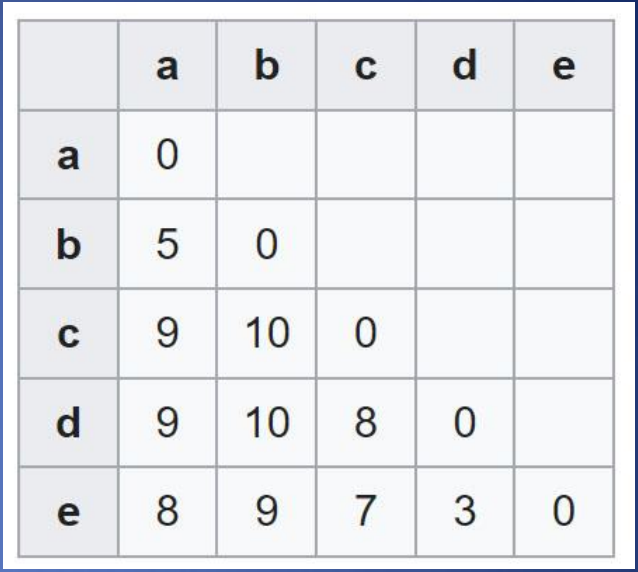
- We calculate the distance between nodes in the tree by summing the distances of the edges connecting them.
- For example, the distance between *a* and *c* is 2+3+4=9.
- Notice, that is the same distance in the distance matrix that we found from the alignments.
  - We say this is a "tree derived" distance.
- The trick is to go backwards, from the matrix to the tree.



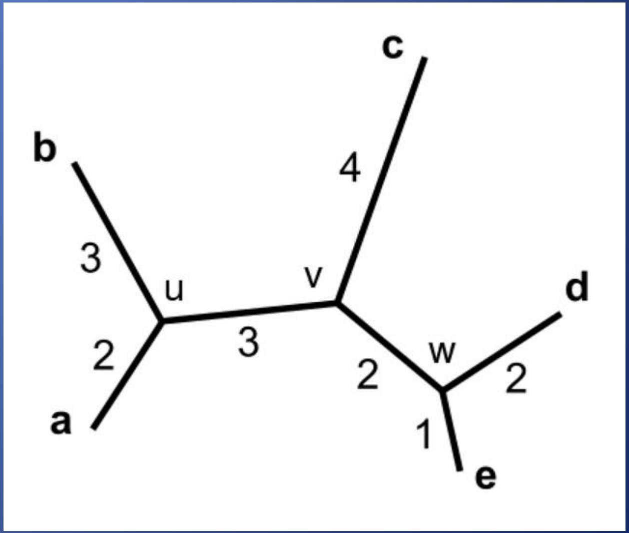


	a	b	c	d	e
a	0				
b	5	0			
c	9	10	0		
d	9	10	8	0	
e	8	9	7	3	0

Phylogenetic Tree Distances



	a	b	c	d	e
a	0				
b	5	0			
c	9	10	0		
d	9	10	8	0	
e	8	9	7	3	0



- If there really is a tree that recovers the same distances as in the matrix, then (happily) we can find it.
- If there is no tree that gives the exact distance matrix, then the goal is to get as close as possible.
- **This is the business of phylogenetics.**
- A big business we don't have time to go into very far.
- We will just look at one algorithm because it's used in multiple sequence alignment.

Neighbor Joining

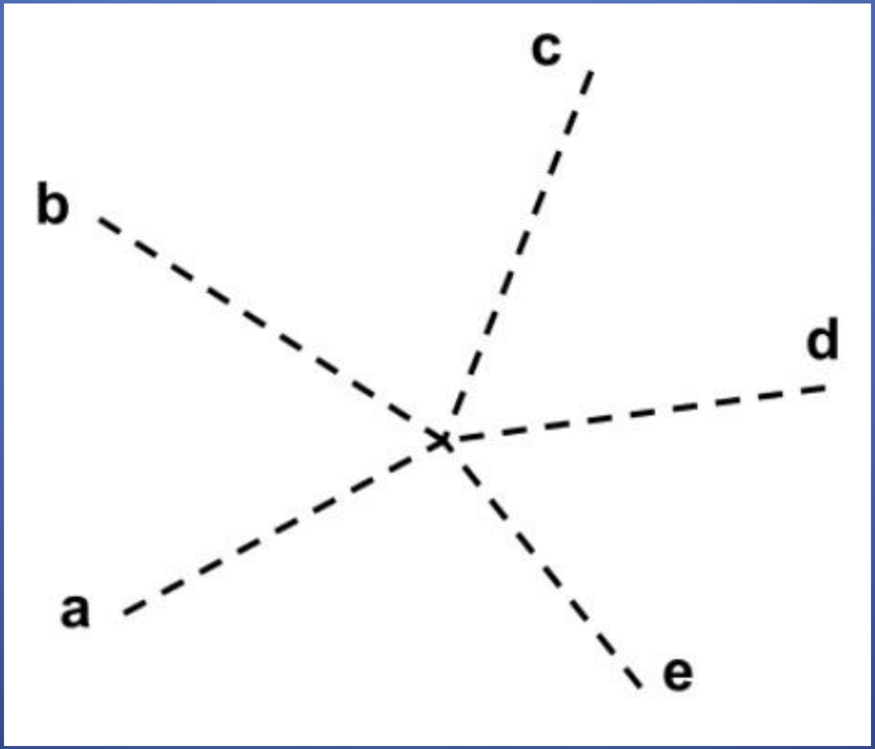
- Neighbor Joining depends on the following theorem:
- Suppose we have a set of sequences and a distance function *d(x,y).*
- Define $\delta(x, y)$ as follows.

$$\delta(x,y) = (N-4)d(x,y) - \sum_{z \neq x,y} (d(x,z) + d(y,z))$$

- Then the *x* and *y* such that $\delta(x, y)$ is smallest are necessarily neighbors in the tree.

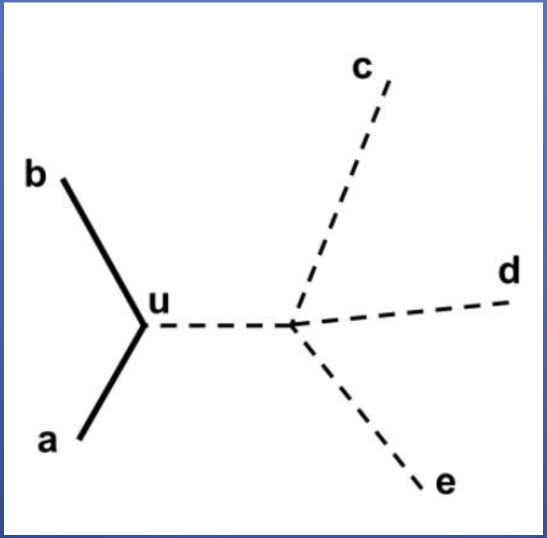
Neighbor Joining

- Start with the with a completely unresolved tree, whose topology corresponds to that of a star.



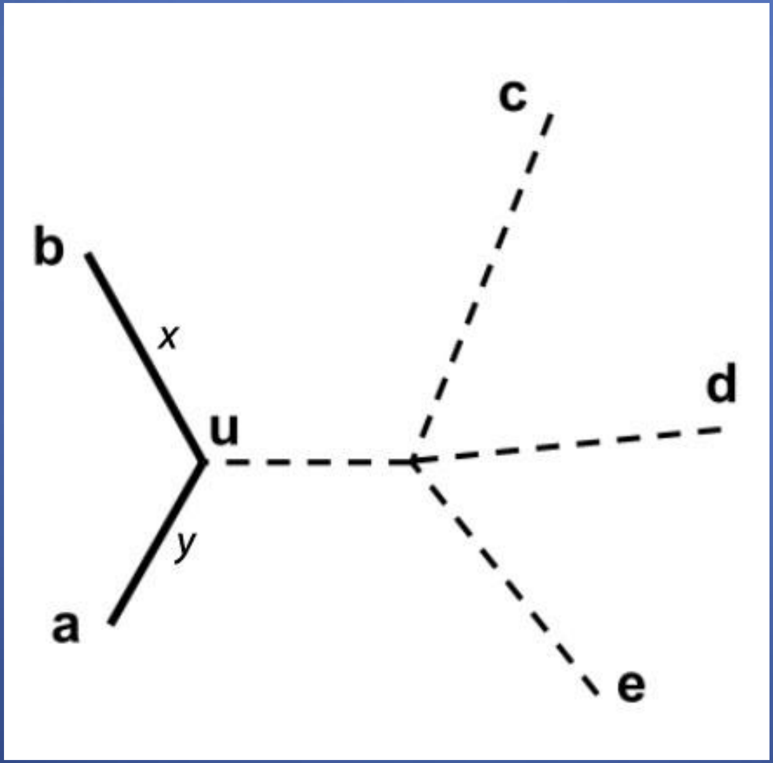
Finding First Neighbors

- Suppose *a* and *b* are the pair of sequences with minimal δ.
  - In other words, $\delta(a, b) \leq \delta(x, y)$ for any *x* and *y*.
- Then we know *a* and *b* are neighbors, so that means the tree topology has to be like this.
  - For some ancestral node *u*.



Adding Distances

- We next need to determine the distances *x* and *y* to the ancestral node *u* to *a* and *b*.



Clever Calculation #1

- We know

$$x + y = d(a,b) = 5.$$

- And,

$$d(b,c) + d(b,d) + d(b,e) - d(a,c) - d(a,d) - d(a,e)$$

$$= 3x - 3y.$$

- We can look those six distances up in the distance matrix, giving

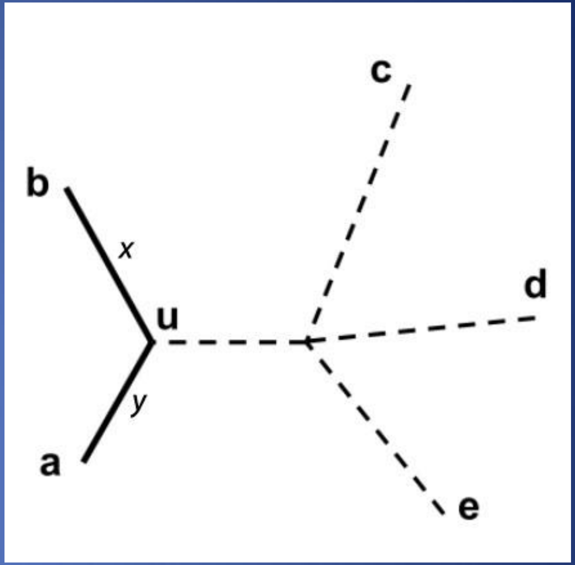
$$10 + 10 + 9 - 9 - 9 - 8 = 3x - 3y$$

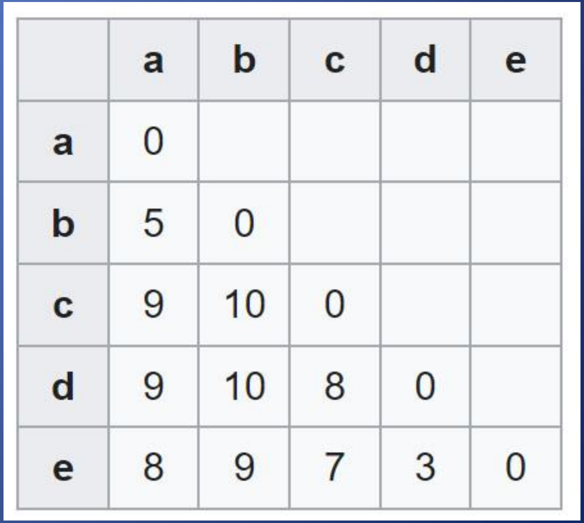
- Which simplifies to

$$x - y = 1$$

Combine this with $x + y = 5$ gives:

$$x = 3, y = 2$$

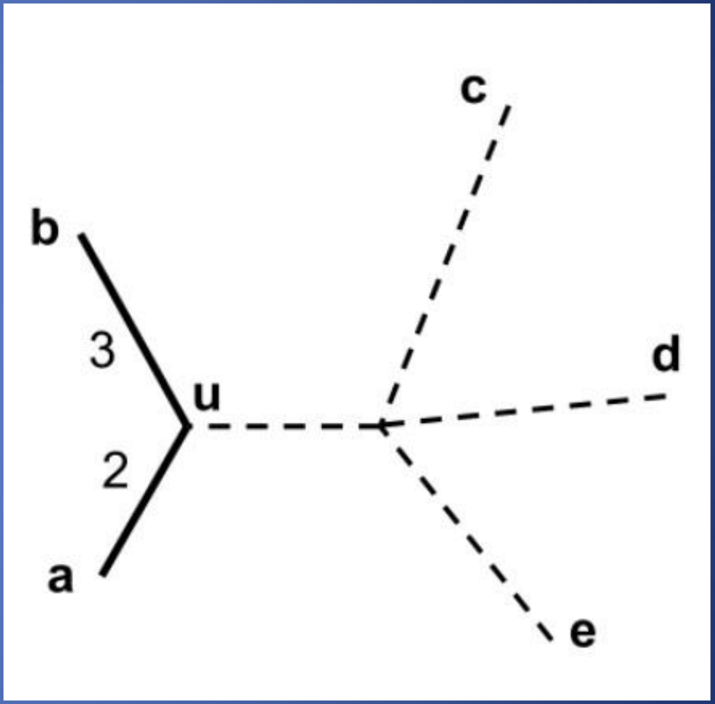




	a	b	c	d	e
a	0				
b	5	0			
c	9	10	0		
d	9	10	8	0	
e	8	9	7	3	0

Lather, Rinse, Repeat

- Now we need to determine the distance from *u* to the remaining nodes *c, d, e.*
  - Once we know that, we have turned our five-node problem into a four-node problem.
  - We repeat the process, until we get down to the last two nodes which must be neighbors..



Calculation #2

- We need to determine $x + y$ because that is $d(u,c)$.

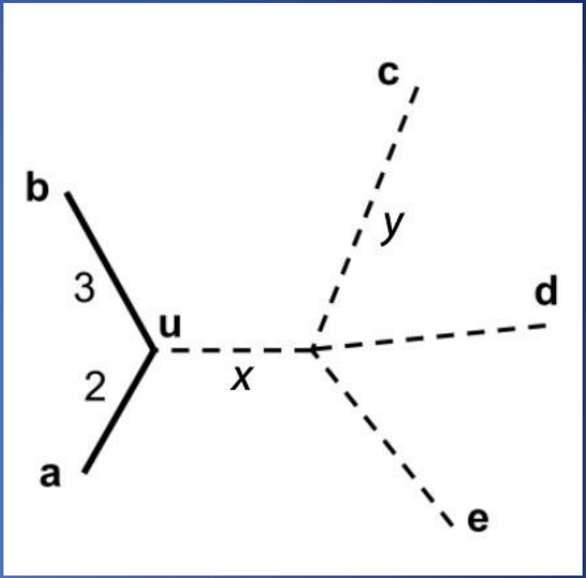
$$d(b,c) = 3 + x + y$$

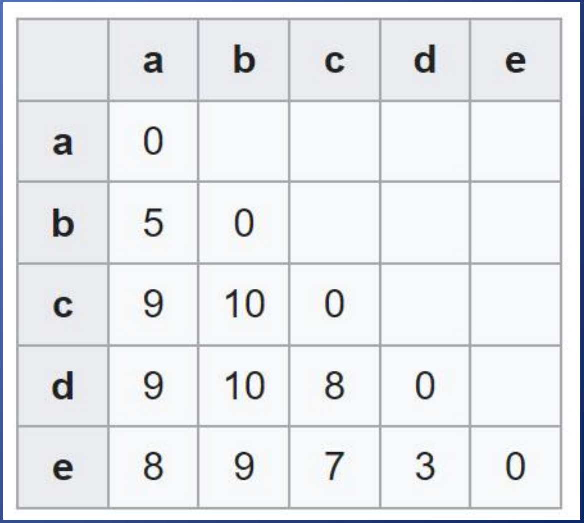
And we know $d(b,c)=10$ from the distance matrix.

Therefore,

$$x + y = 7$$

- Thus, we have shown that $d(u,c) = 7$.
  - Likewise, we can find $d(u,d)$ and $d(u,e)$.



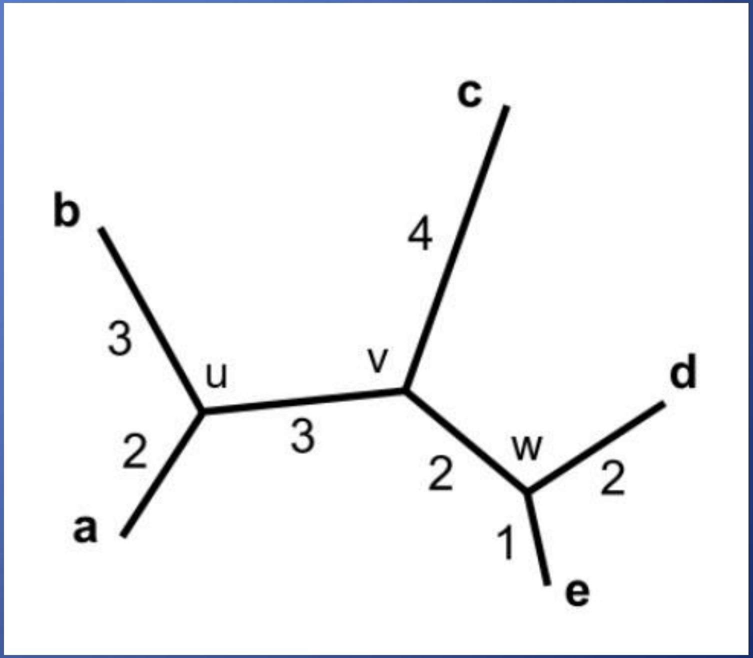


	a	b	c	d	e
a	0				
b	5	0			
c	9	10	0		
d	9	10	8	0	
e	8	9	7	3	0

Next Step

- Proceeding with the four nodes *c, d, e* and *u* the next pair of neighbors are *d* and *e.*
- Making similar calculations as before gives the final (unrooted) tree.

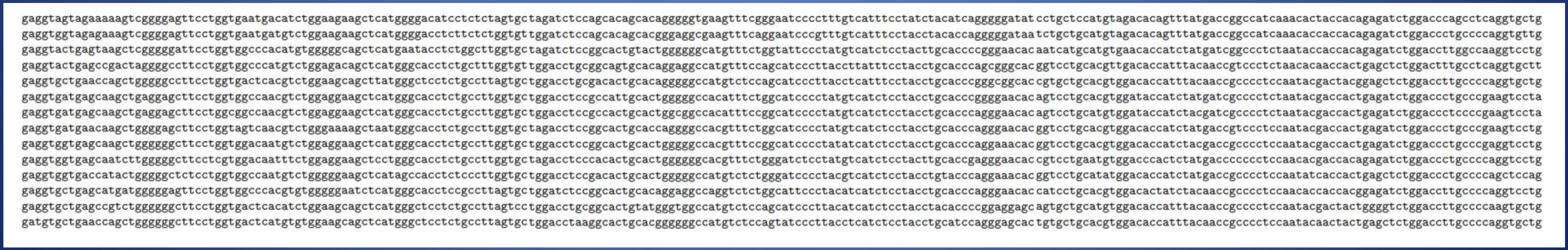
- This is a very efficient algorithm.
  - Can align many long sequences very fast.



Example

- Let’s build a tree from 14 species.
  - We’ll use the gene “interphotoreceptor retinoid binding protein” which is found in all 14.
  - Here’s a piece of the alignment:

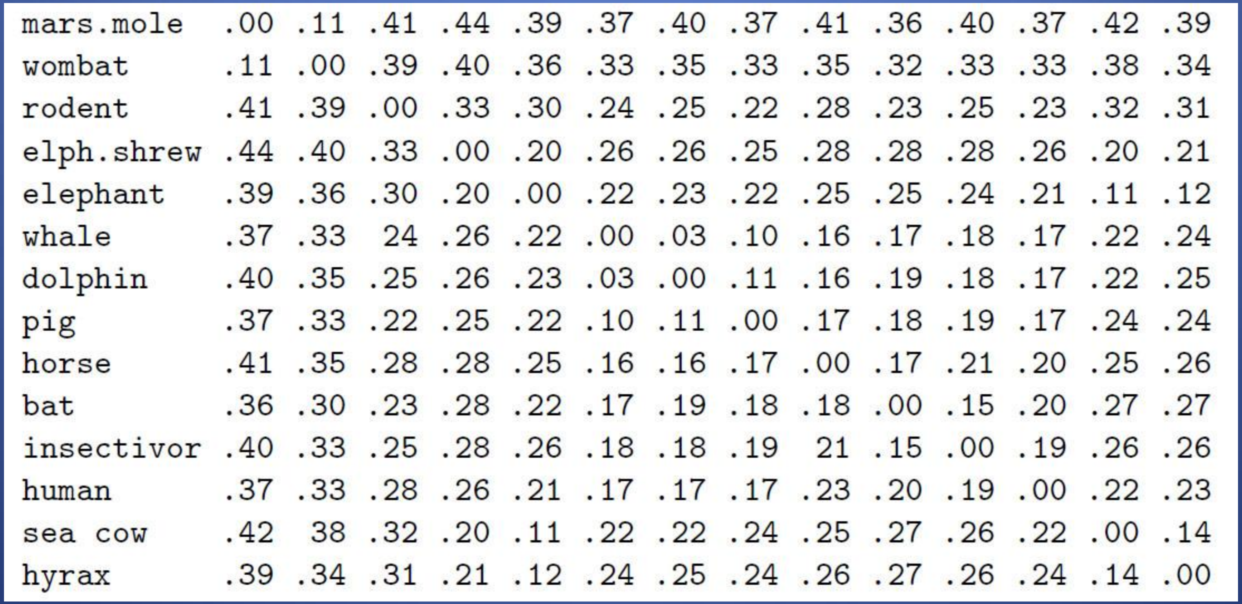
```
mars.mole
wombat
rodent
elph.shrew
elephant
whale
dolphin
pig
horse
bat
insectivor
human
sea cow
hyrax
```



```
gaggtagtagaaaaagtcggggagttcctggtgaatgacatctggaagaagctcatggggacatcctctctagtgctagatctccagcacagcacagggggtgaagtttcgggaatcccctttgtcatttcctatctacatcagggggatat cctgctccatgtagacacagtttatgaccggccatcaaacactaccacagagatctggacccagcctcaggtgctg
gaggtggtagagaaagtcggggagttcctggtgaatgatgtctggaagaagctcatggggacctcttctctggtgttggatctccagcacagcacgggaggcgaagtttcaggaatcccgtttgtcatttcctacctacaccagggggataa tctgctgcatgtagacacagtttatgaccggccatcaaacaccaccacagagatctggaccctgccccaggtgttg
gaggtactgagtaagctcgggggattcctggtggcccacatgtgggggcagctcatgaatacctctggcttggtgctagatctccggcactgtactgggggcatgtttctggtattccctatgtcatctcctacttgcaccccgggaacac aatcatgcatgtgaacaccatctatgatcggccctctaataccaccacagagatctggaccttggccaaggtcctg
gaggtactgagccgactaggggccttcctggtggcccatgtctggagacagctcatgggcacctctgctttggtgttggacctgcggcagtgcacaggaggccatgtttccagcatcccttaccttatttcctacctgcacccagcgggcac ggtcctgcacgttgacaccatttacaaccgtccctctaacacaaccactgagctctggactttgcctcaggtgctt
gaggtgctgaaccagctgggggccttcctggtgactcacgtctggaagcagcttatgggctcctctgccttagtgctggacctgcgacactgcacagggggccatgtctccagcatcccttacctcatttcctacctgcacccgggcggcac cgtgctgcacgtggacaccatttacaaccgcccctccaatacgactacggagctctggaccttgccccaggtgctg
gaggtgatgagcaagctgaggagcttcctggtggccaacgtctggaggaagctcatgggcacctctgccttggtgctggacctccgccattgcactgggggccacatttctggcatcccctatgtcatctcctacctgcacccggggaacac agtcctgcacgtggataccatctatgatcgcccctctaatacgaccactgagatctggaccctgcccgaagtccta
gaggtgatgagcaagctgaggagcttcctggcggccaacgtctggaggaagctcatgggcacctctgccttggtgctggacctccgccactgcactggcggccacatttccggcatcccctatgtcatctcctacctgcacccagggaacac agtcctgcatgtggataccatctacgatcgcccctctaatacgaccactgagatctggaccctccccgaagtccta
gaggtgatgaacaagctggggagcttcctggtagtcaacgtctgggaaaagctaatgggcacctctgccttggtgctagacctccggcactgcaccagggggccacgtttctggcatcccctatgtcatctcctacctgcacccagggaacac ggtcctgcacgtggacaccatctatgaccgtccctccaatacgaccactgagatctggaccctgcccgaagtcctg
gaggtggtgagcaagctggggggcttcctggtggacaatgtctggaggaagctcatgggcacctctgccttggtgctggacctccggcactgcactgggggccacgtttccggcatcccctatatcatctcctacctgcacccaggaaacac ggtcctgcacgtggacaccatctacgaccgcccctccaatacgaccactgagatctggaccctgcccgaggtcctg
gaggtggtgagcaatcttgggggcttcctcgtggacaatttctggaggaagctcctgggcacctctgccttggtgctagacctcccacactgcactgggggcacgtttctgggatctcctatgtcatctcctacttgcaccgagggaacac cgtcctgaatgtggacccactctatgaccccccctccaacacgaccacagagatctggaccctgccccaggtcctg
gaggtggtgaccatactgggggctctcctggtggccaatgtctgggggaagctcatagccacctctcccttggtgctggacctccgacactgcactgggggccatgtctctgggatcccctacgtcatctcctacctgtacccaggaaacac ggtcctgcatatggacaccatctatgaccgcccctccaatatcaccactgagctctggaccctgccccagctccag
gaggtgctgagcatgatgggggagttcctggtggcccacgtgtgggggaatctcatgggcacctccgccttagtgctggatctccggcactgcacaggaggccaggtctctggcattccctacatcatctcctacctgcacccagggaacac catcctgcacgtggacactatctacaaccgcccctccaacaccaccacggagatctggaccttgccccaggtcctg
gaggtgctgagccgtctggggggcttcctggtgactcacatctggaagcagctcatgggctcctctgccttagtcctggacctgcggcactgtatgggtggccatgtctccagcatcccttacatcatctcctacctacaccccggaggagc agtgctgcatgtggacaccatttacaaccgcccctccaatacgactactggggtctggaccttgccccaagtgctg
gatgtgctgaaccagctggggggcttcctggtgactcatgtgtggaagcagctcatgggctcctctgccttagtgctggacctaaggcactgcacggggggccatgtctccagtatcccttacctcatctcctacctgcatccagggagcac tgtgctgcacgtggacaccatttacaaccgcccctccaatacaactactgagctctggaccttgccccaggtgctg
```

Distance Matrix

- The distance matrix looks like this.
  - It has been normalized so all distances are between 0 and 1, but it’s essentially the same information as before.
  - Notice is is symmetric.



mars.mole	.00	.11	.41	.44	.39	.37	.40	.37	.41	.36	.40	.37	.42	.39
wombat	.11	.00	.39	.40	.36	.33	.35	.33	.35	.32	.33	.33	.38	.34
rodent	.41	.39	.00	.33	.30	.24	.25	.22	.28	.23	.25	.23	.32	.31
elph.shrew	.44	.40	.33	.00	.20	.26	.26	.25	.28	.28	.28	.26	.20	.21
elephant	.39	.36	.30	.20	.00	.22	.23	.22	.25	.25	.24	.21	.11	.12
whale	.37	.33	24	.26	.22	.00	.03	.10	.16	.17	.18	.17	.22	.24
dolphin	.40	.35	.25	.26	.23	.03	.00	.11	.16	.19	.18	.17	.22	.25
pig	.37	.33	.22	.25	.22	.10	.11	.00	.17	.18	.19	.17	.24	.24
horse	.41	.35	.28	.28	.25	.16	.16	.17	.00	.17	.21	.20	.25	.26
bat	.36	.30	.23	.28	.22	.17	.19	.18	.18	.00	.15	.20	.27	.27
insectivor	.40	.33	.25	.28	.26	.18	.18	.19	21	.15	.00	.19	.26	.26
human	.37	.33	.28	.26	.21	.17	.17	.17	.23	.20	.19	.00	.22	.23
sea cow	.42	38	.32	.20	.11	.22	.22	.24	.25	.27	.26	.22	.00	.14
hyrax	.39	.34	.31	.21	.12	.24	.25	.24	.26	.27	.26	.24	.14	.00

Four Algorithms, Four Trees

- This shows how you can get different trees depending on algorithm.
  - The tree also depends on the pairwise scoring function, because different ones can give different distance matrices.
- There are some consistencies between the trees however.

Neighbor Joining

```
                       +--insectivor
         +-------------+
         !             +--bat
         !
         !       +--------horse
       +-+ +-----+
       ! ! !     !  +-----pig
       ! ! !     +--+
       ! ! !        !  +--dolphin
       ! +-+        +--+
       !   !           +--whale
       !   !
       !   !  +-----------human
    +--+   !  !
    !  !   +--+     +-----sea cow
    !  !      !  +--+
    !  !      !  !  !  +--hyrax
    !  !      +--+  +--+
 +--+  !         !     +--elephant
 !  !  !         !
 !  !  !         +--------elph.shrew
 !  !  !
-+  !  +------------------rodent
 !  !
 !  +---------------------wombat
 !
 +------------------------mars.mole
```

Maximum Parsimony

```
     +--------rodent
     !
     !           +whale
     !        +--+
  +--+     +--+  +dolphin
  !  !     !  !
  !  !  +--+  +--pig
  !  !  !  !
  !  +--+  +-----horse
  !     !
  !     !  +---bat
  !     +--+
  !        +----insectivore
  !
--+----human
  !
  !             +----mars.mole
  !  +----------+
  !  !          +-wombat
  +--+
     !  +-------elph.shrew
     !  !
     +--+     +--elephant
        !  +--+
        +--+  +---sea cow
           !
           +---hyrax
```

Maximum Likelihood

```
+-wombat
!
!               +--------rodent
!            +--+
!            !  +----human
!            !
!         +--+       +---sea cow
!         !  !    +--+
!         !  !    !  !  +---hyrax
!         !  +----+  +--+
!      +--+       !     +--elephant
!      !  !       !
!      !  !       +------elph.shrew
!      !  !
!      !  !  +-----horse
!      !  +--+
!   +--+     !  +--pig
!   !  !     +--+
!   !  !        !  +-dolphin
!   !  !        +--+
+---+  !           +whale
!   !  !
!   !  +----insectivor
!   !
!   +---bat
!
+----mars.mole
```

UPGMA

```
          +--mars.mole
  +-------+
  !       +--wombat
  !
  !      +------rodent
  !      !
  !      !              +whale
  !      !           +--+
--+      !        +--+  +dolphin
  !   +--+        !  !
  !   !  !     +--+  +--pig
  !   !  !     !  !
  !   !  !  +--+  +----horse
  !   !  !  !  !
  !   !  +--+  +----human
  +---+     !
  !         !  +---bat
  !         +--+
  !            +---insectivore
  !
  !  +-----elph.shrew
  !  !
  +--+     +--elephant
     !  +--+
     +--+  +--sea cow
        !
        +---hyrax
```

Guide Tree

- However, we get the tree, it can now be used to determine the order in which the sequences are progressively aligned.
- First all neighbors are aligned in pairwise alignments.
- Then they can be merged into alignments of four sequences.
  - Etc., until all sequences are in one big multiple alignment.

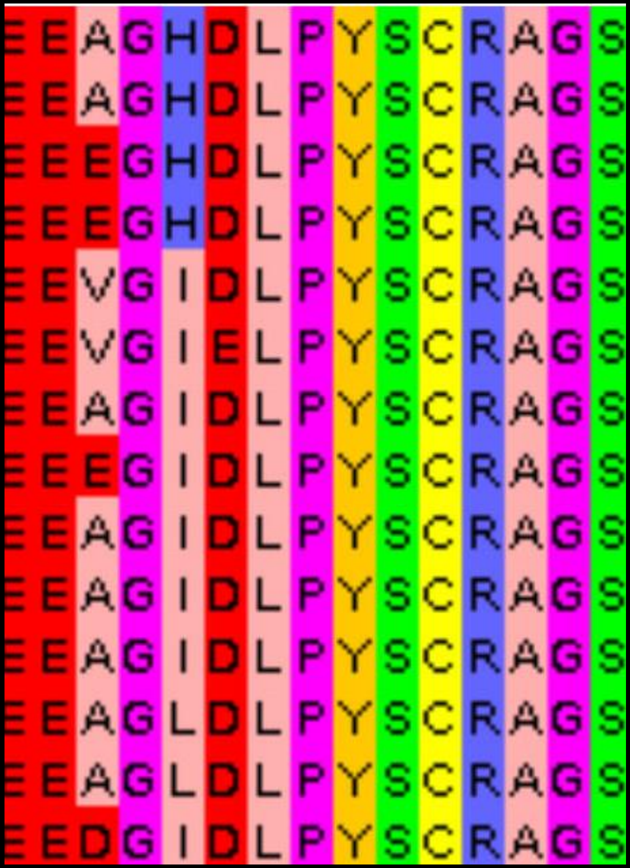
ClustalW

- The app clustalW uses this neighbor-joining/progressive alignment approach.
  - But the devil is in the details.
- Nothing is ever as simple as it seems in class.
  - We strip out a lot of the details to illustrate the main concepts.
  - But it always requires tweaking and fine-tuning to get it to work well.
- There are numerous online servers for ClustalW, for example this one out of the European Bioinformatics Institute:

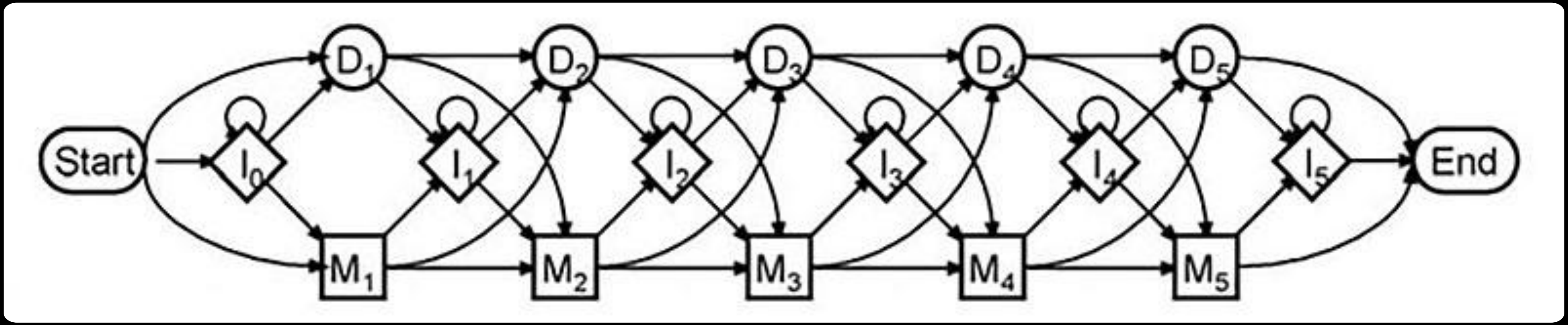
  `https://www.ebi.ac.uk/Tools/msa/clustalo/`

Genes, Proteins, Sequences - and Models -

- Consider a gene's sequence.
- Now consider that same gene across all Eukaryotes.
  - Say it's a basic gene found in all species.
- These sequences are all different but they're similar.
- Imagine if you had these sequence from enough species that you could make up your own novel ones that look like they belong to the family.
- Your brain is modeling the family.
- Now imagine writing a computer program to do it.
- You'd need some sort of "model" you can program.
- For some positions that might be easy, for example if it's the same nucleotide in all species.
- For others you might mimic the frequencies of the four bases.
- You'd also have to figure out how to model indels so they resemble the ones in the family.
- This can be done with a Hidden Markov Model.

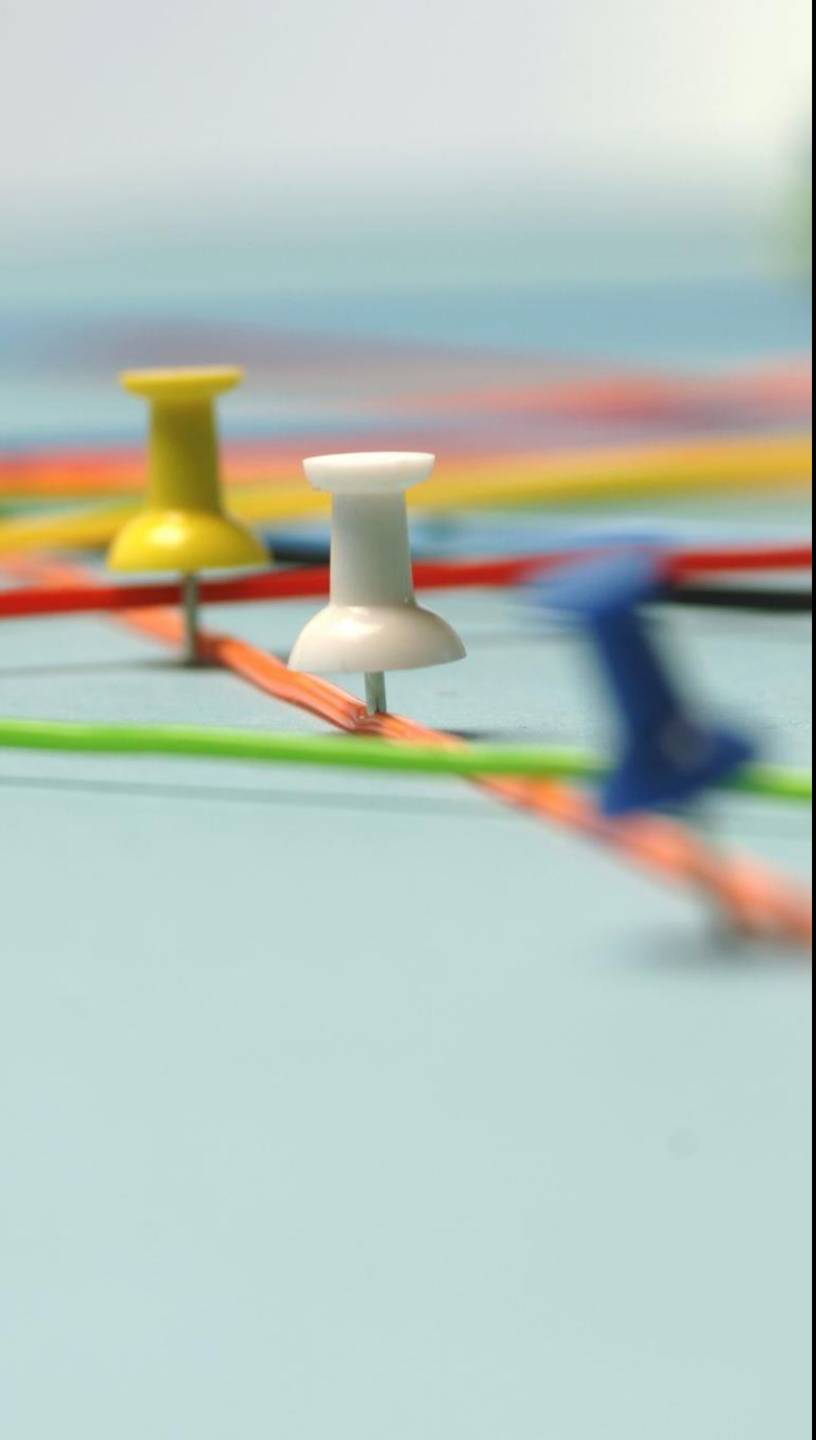


```
EEAGHDLPYSCRAGS
EEAGHDLPYSCRAGS
EEEGHDLPYSCRAGS
EEEGHDLPYSCRAGS
EEVGIDLPYSCRAGS
EEVGIELPYSCRAGS
EEAGIDLPYSCRAGS
EEEGIDLPYSCRAGS
EEAGIDLPYSCRAGS
EEAGIDLPYSCRAGS
EEAGIDLPYSCRAGS
EEAGLDLPYSCRAGS
EEAGLDLPYSCRAGS
EEDGIDLPYSCRAGS
```



Hidden Markov Models

- A completely different approach to multiple alignment is to built something called a Hidden Markov Model (HMM).
  - We will learn about Markov Models soon. But not HMMs.
- A Hidden Markov Model is just a bit more complex and therefore allows for modeling more complex biological problems.

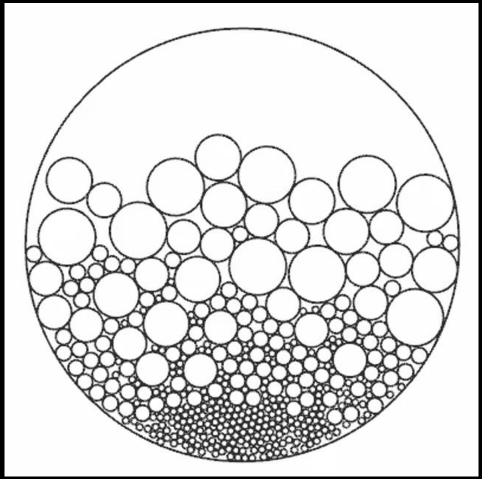


Hidden Markov Model Approach to Multiple Alignment

- You train the model parameters on the data.
  - The data just consists of the unaligned sequences.
- Then there's a way for each sequence to determine its shortest route through the model.
- Those shortest routes when put together determine a multiple alignment.
- We're skipping the details for the high-level concept.

The Brazil Nut Effect

- Suppose you are tasked with placing different sized objects in a container so that no object is higher than another object that weighs more.
  - To solve this problem exactly, you might have to weigh each object and place them methodically.
- But what if an approximate solution was acceptable.
- In this case you could just shake the container and let physics do its thing.
  - Most of the largest objects will rise to the top and the smallest to the bottom.
- Gibbs Sampling is sort of like this, you start with something random and you create the right force (alignment score instead of gravity) and then you shake and a decent alignment falls out.

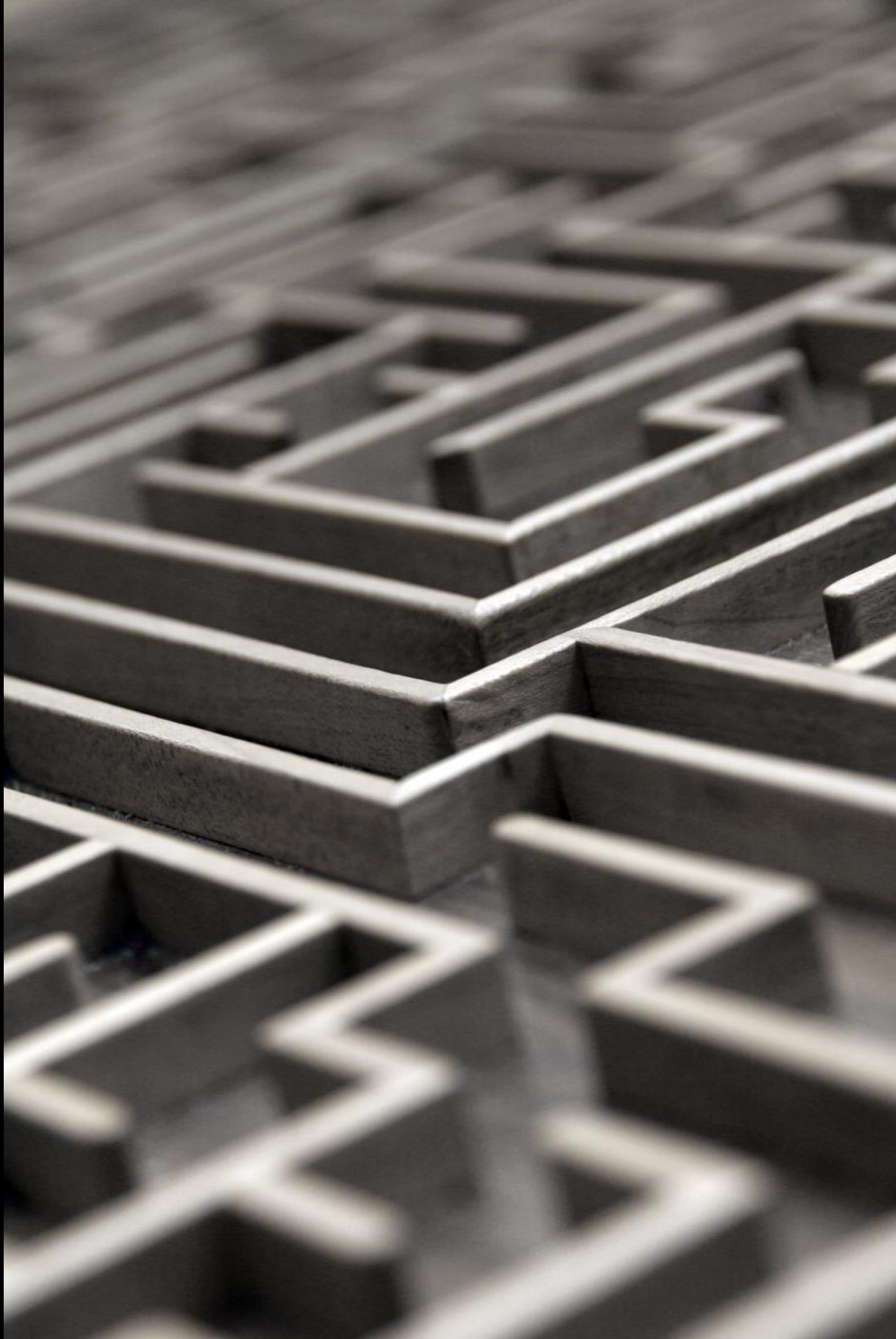


STOP HERE

- You are not responsible for the following material, pseudo-counts and the Lawrence method.
- But it is another multiple sequence alignment method using yet another completely different approach.

Gibbs Sampling

- There's a multiple alignment method that is basically an implementation of Gibbs Sampling, which uses Markov Chain theory.
  - The details are beyond our scope here.
- It's a (clever) heuristic approach to searching large spaces for points where a function of the space (into the real numbers) is optimized.
  - This is done by bouncing around the space somewhat randomly, but in a way the makes 'interesting' things more likely to be visited over time.



BLOCKS
(we will cover this time permitting)

- Next, we are going to look at a statistical method for finding the *ungapped local* alignments used to create the blocks for BLOSUM matrix construction.
  - Introduced by Lawrence et al. (1993).

Counts and Pseudocounts

- Suppose we have a large population of three types of things: *A, B* and *C.*
  - Say there are n_A things of type *A*, n_B of type *B*, and n_C of type *C.*
- The probability of choosing *A* at random is

$$p_A = \frac{n_A}{n_A + n_B + n_C}$$

- Suppose we don't know n_A, n_B or n_C and we want to estimate them from data.

Counts and Pseudocounts

- We sample the population a *N* times and count.
- Suppose there are m_A *As*, m_B *Bs* and m_C *Cs* in the sample.
- The natural estimates are

$$\widehat{p_A} = \frac{m_A}{m_A + m_B + m_C}$$

$$\widehat{p_B} = \frac{m_B}{m_A + m_B + m_C}$$

$$\widehat{p_C} = \frac{m_C}{m_A + m_B + m_C}$$

- Convince yourself that these sum to 1.

Rare Events

- But what if one of the things, say B is rare, so rare that we are unlikely to sample one in our random sample of size N.
- In that case, the count $m_b = 0$ and as a result the estimated probability $p_b = 0$.
  - This can be undesirable in some situations, including in the Lawrence algorithm, as we shall see.

Pseudocounts

- This situation is often rectified by adding a natural number to each count, known as a pseudocount.

$$\widehat{p_A} = \frac{m_A + b}{m_A + m_B + m_C + 3b}$$

$$\widehat{p_B} = \frac{m_B + b}{m_A + m_B + m_C + 3b}$$

$$\widehat{p_C} = \frac{m_C + b}{m_A + m_B + m_C + 3b}$$

- Convince yourself that these also sum to 1.
- And they are now all greater than 0.

Pseudocounts

- Pseudocounts have a greater impact on smaller numbers.
- Adding 1 to the numerator and denominator of 0.05 / 0.23
  - Before: 0.2174
  - After: 0.8536
- has a much greater effect than adding 1 to the numerator and denominator of 1247 / 8253
  - Before: 0.1511
  - After: 0.1512
- This will be important in RNA sequencing and gene quantification.

The Lawrence Method
- to create multiple alignments -

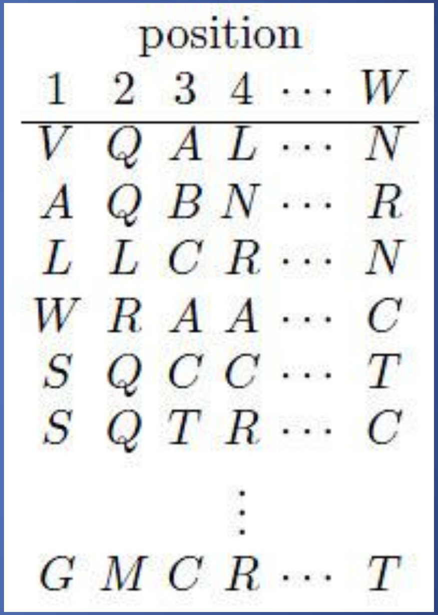
- Number the amino acids 1, 2, … , 20, and suppose that these have respective background frequencies p_1, p_2, … , p_{20}.
- Given N protein sequences, of respective lengths L_1, L_2, … , L_N, the aim is to find N segments of length W, one in each sequence, that are most similar to each other.
  - For now, W is a fixed constant

The Lawrence Method

- There are $S = \prod_{j=1}^{N}(L_j - W + 1)$ possible choices for the respective locations of *N* segments of length *W* in the *N* respective sequences.
- It is assumed that *N* and the L_j are so large that a purely algorithmic approach is not feasible.
- The procedure consists of repeated iteration of a basic step.

The Lawrence Method

- The initial alignment can be chosen arbitrarily or as an initial guess of the best alignment.
- This consists of choosing the starting (leftmost) location in each of the *N* sequences.
  - Which can be any position from 1 to $L_i - W + 1$
- So, we start with a block array of letters *W* wide and *N* high.

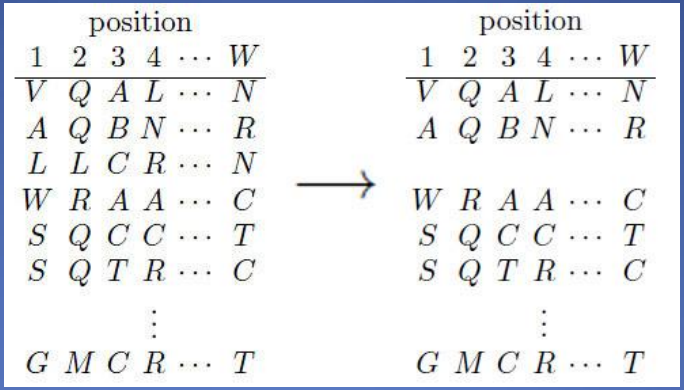


position

1	2	3	4	⋯	W
V	Q	A	L	⋯	N
A	Q	B	N	⋯	R
L	L	C	R	⋯	N
W	R	A	A	⋯	C
S	Q	C	C	⋯	T
S	Q	T	R	⋯	C
			⋮		
G	M	C	R	⋯	T

- In each step one of the various sequences is chosen at random to be changed.
  - We remove this sequence from the array

The Lawrence Method



$$\begin{array}{cccccc} \multicolumn{6}{c}{\text{position}} \\ 1 & 2 & 3 & 4 & \cdots & W \\ \hline V & Q & A & L & \cdots & N \\ A & Q & B & N & \cdots & R \\ L & L & C & R & \cdots & N \\ W & R & A & A & \cdots & C \\ S & Q & C & C & \cdots & T \\ S & Q & T & R & \cdots & C \\ & & & \vdots & & \\ G & M & C & R & \cdots & T \end{array} \longrightarrow \begin{array}{cccccc} \multicolumn{6}{c}{\text{position}} \\ 1 & 2 & 3 & 4 & \cdots & W \\ \hline V & Q & A & L & \cdots & N \\ A & Q & B & N & \cdots & R \\ & & & & & \\ W & R & A & A & \cdots & C \\ S & Q & C & C & \cdots & T \\ S & Q & T & R & \cdots & C \\ & & & \vdots & & \\ G & M & C & R & \cdots & T \end{array}$$

- We'll call the array after removal the 'reduced array'
- Suppose that in the reduced array, amino acid *j* occurs c_{ij} times in the *i*th column.
- Define

$$q_{ij} = \frac{c_{ij} + b_j}{N - 1 + B}$$

where b_j are pseudocounts and $B = \sum b_j$

- The q_{ij}'s sum to 1.

The Lawrence Method

- The aim of the step is to replace the original segment in the chosen row by a new segment in a way that tends to increase the overall quality of the alignment of the *N* resulting segments in the new array.
- Assume for concreteness that row 3 is chosen.
  - There are $L_3 - W + 1$ segments of length W in sequence 3.
- Call the segment starting in position *x* in sequence 3 “segment *x*.”
- Suppose the amino acids in the segment are $x_1, x_2, \ldots, x_W$.
- The probability of this ordered set of amino acids *under the population frequencies* is

$$P_x = p_{x_1} \cdot p_{x_2} \cdots p_{x_W}$$

Likelihood Ratio

- P_x is the null hypothesis probability of the segment.
- We want to compare it to the alternative hypothesis probability derived from the current aligned block, with the one chosen sequence removed.
- This is given by

$$Q_x = q_{1,x_1} \cdot q_{2,x_2} \cdots q_{W,x_W}$$

- The likelihood ratio LR(x) is defined as

$$Q_x / P_x$$

Likelihood Ratio

- The numerator may be thought of as the probability of the sequence under the model reflecting the sequences in the current alignment.
- While the denominator is the probability of the sequence under “background” frequencies.
- We replace the segment in the chosen row with the segment *x* which has maximum LR(*x*).

Iteration

- As one step follows another the *N* segments in the block tend to become more similar to each other.
  - Or in other words, to align better.
- This iterative procedure visits various possible alignments according to a random process, and thus does not systematically approach the "best" alignment.
  - However, after many steps the best alignments tend to arise more and more frequently and hence be recognized.

The Pseudocounts

- If pseudocounts were not used and the probability estimate q_{ij} replaced by c_{ij}/(N - 1), then q_{ij} could be equal to zero, causing possibly excellent alignments to be skipped over in the random search.
- If we make each $b_i > 0$ then the entire space of all possible blocks can be visited.
- The choice of the pseudocounts b_j must be made subjectively.
  - Lawrence et al. (1993) make the reasonable suggestion of making b_j proportional to q_j with proportionality constant $\sqrt{N}$.