

Bioinformatics Computing

Dr. Rahul Singh
Professor, Department of Computer Science

San Francisco State University

Email: rahul@sfsu.edu
URL: <http://tintin.sfsu.edu/RahulSingh.html>

Our Understanding of the Phenomenon Called “Life”

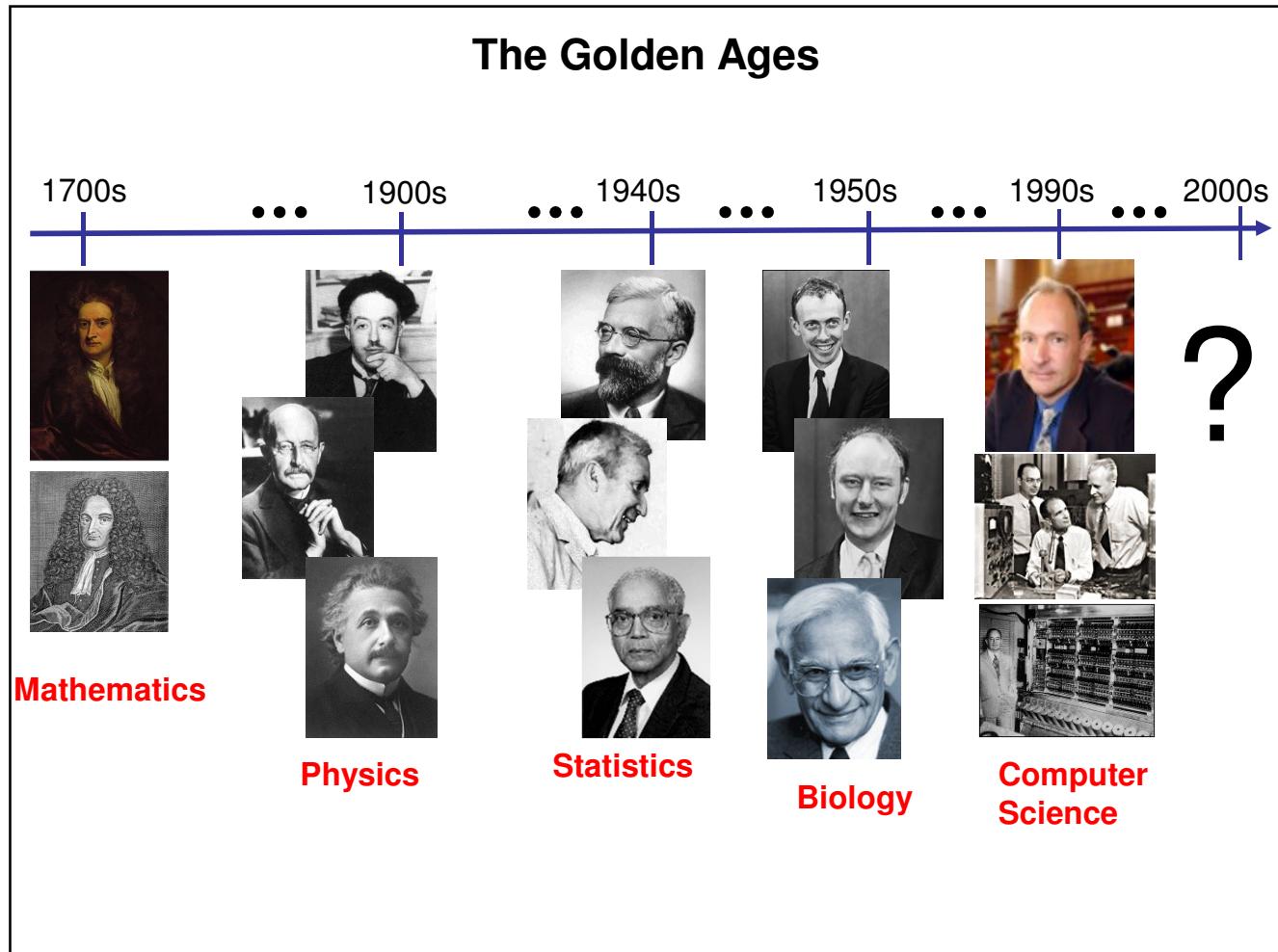
What is a “living thing”: Something which eats, grows, reproduces, dies

- Used mostly till the 80s

What is a “living thing”: Something which has a unique DNA sequence

- Typically used today

The Golden Ages

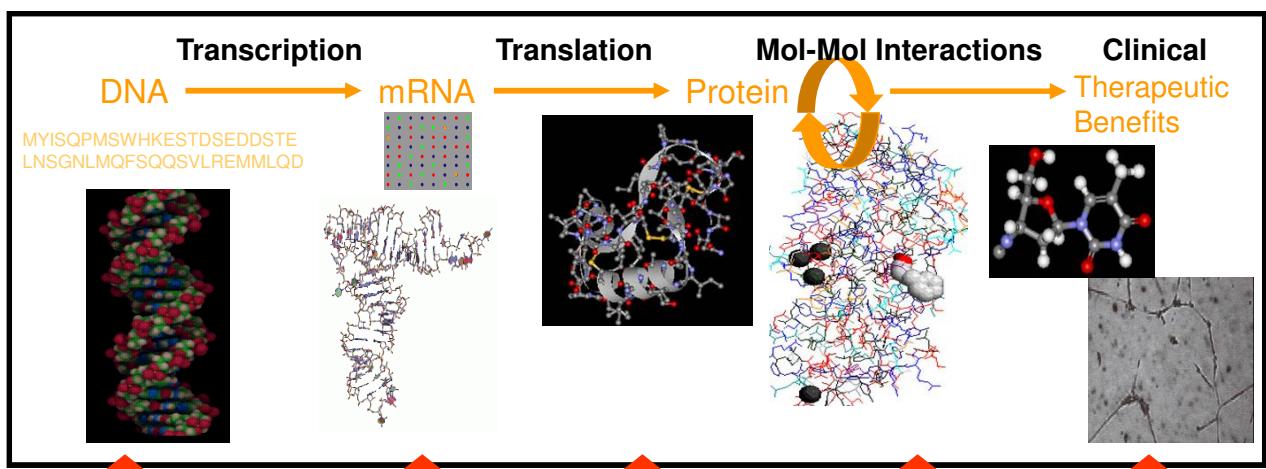


? =

Biology + Chemistry + Physics + Nanotechnology
+
Computer Science

i.e. an amalgam of biochemical, mathematical, engineering and computational sciences

What is the role of Computing?



Sequence Analysis,
Sequence to Structure,
Sequence to function,
Pathway analysis

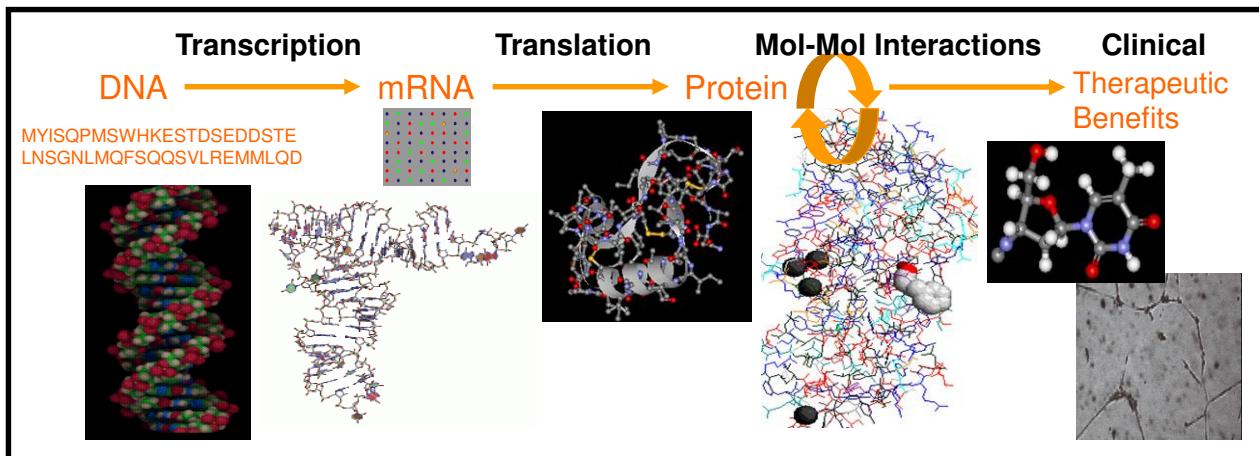
Transcriptional Analysis

Protein Folding,
Threading

Structural Analysis,
Docking,
Active Site Description,

Structure-Property
Modeling,
Structure-Class
Discovery,
Storage

Why Life-Sciences and Computing?



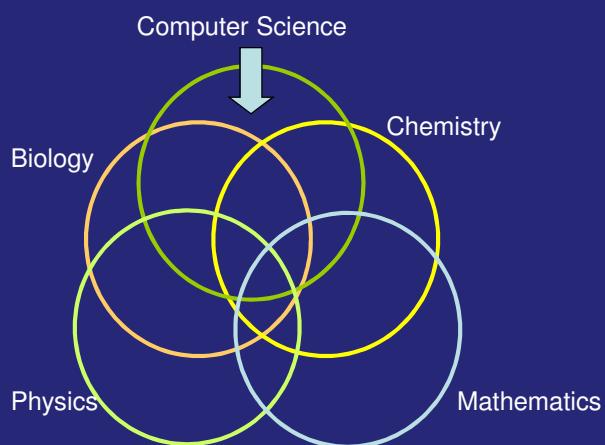
Needs induced by data volume: Traditional and high-throughput biology, combinatorial chemistry, automated pharmacology, ...

-The data management problem: storage, querying, visualization, remote interactions

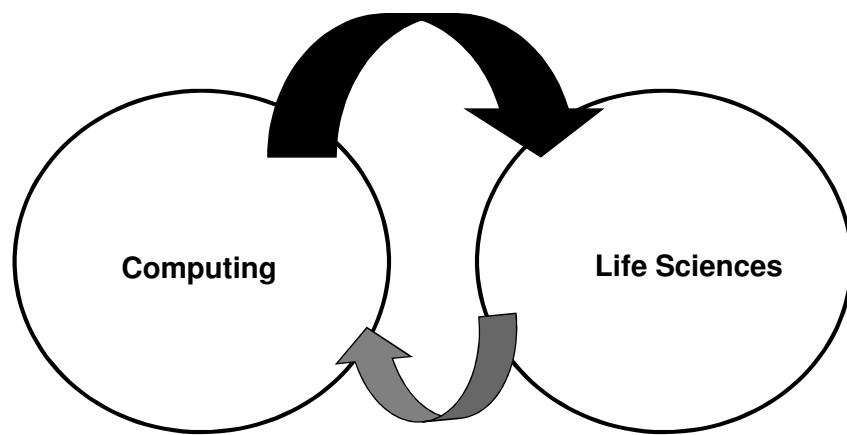
Needs induced by data complexity: Understanding large and small molecular structures, molecule-molecule interactions, metabolic pathways, ...

-The information modeling-analysis problem: new algorithms for systems biology, molecular modeling, structure-activity modeling

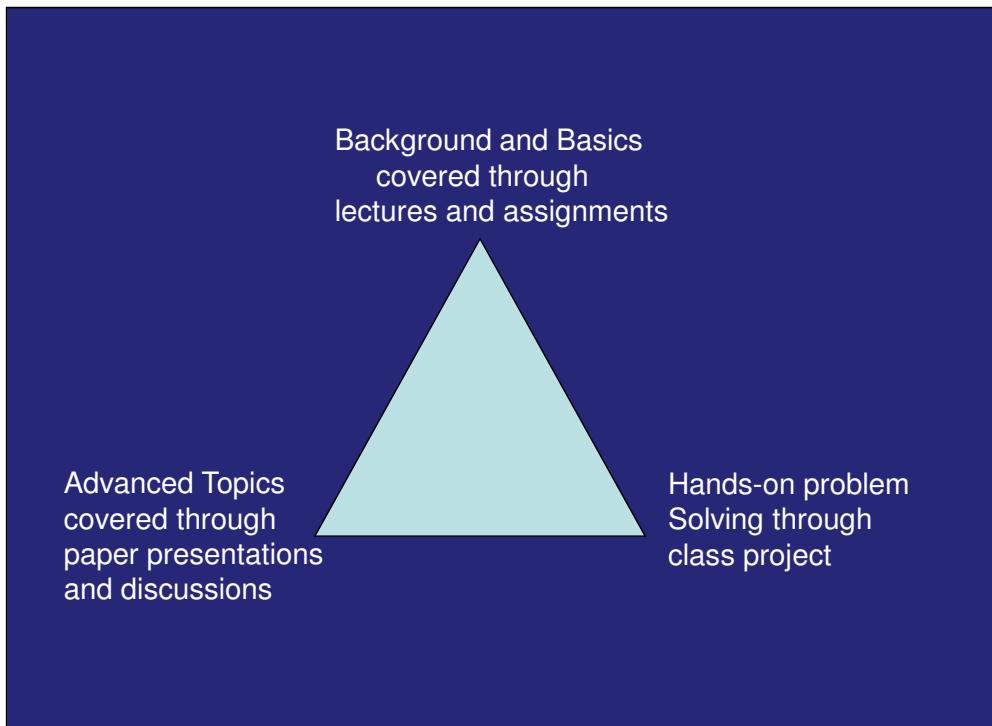
The Perspective of this Course



Impact on your career (If you are a computer scientist)



Course Design



Readings on Biological Background

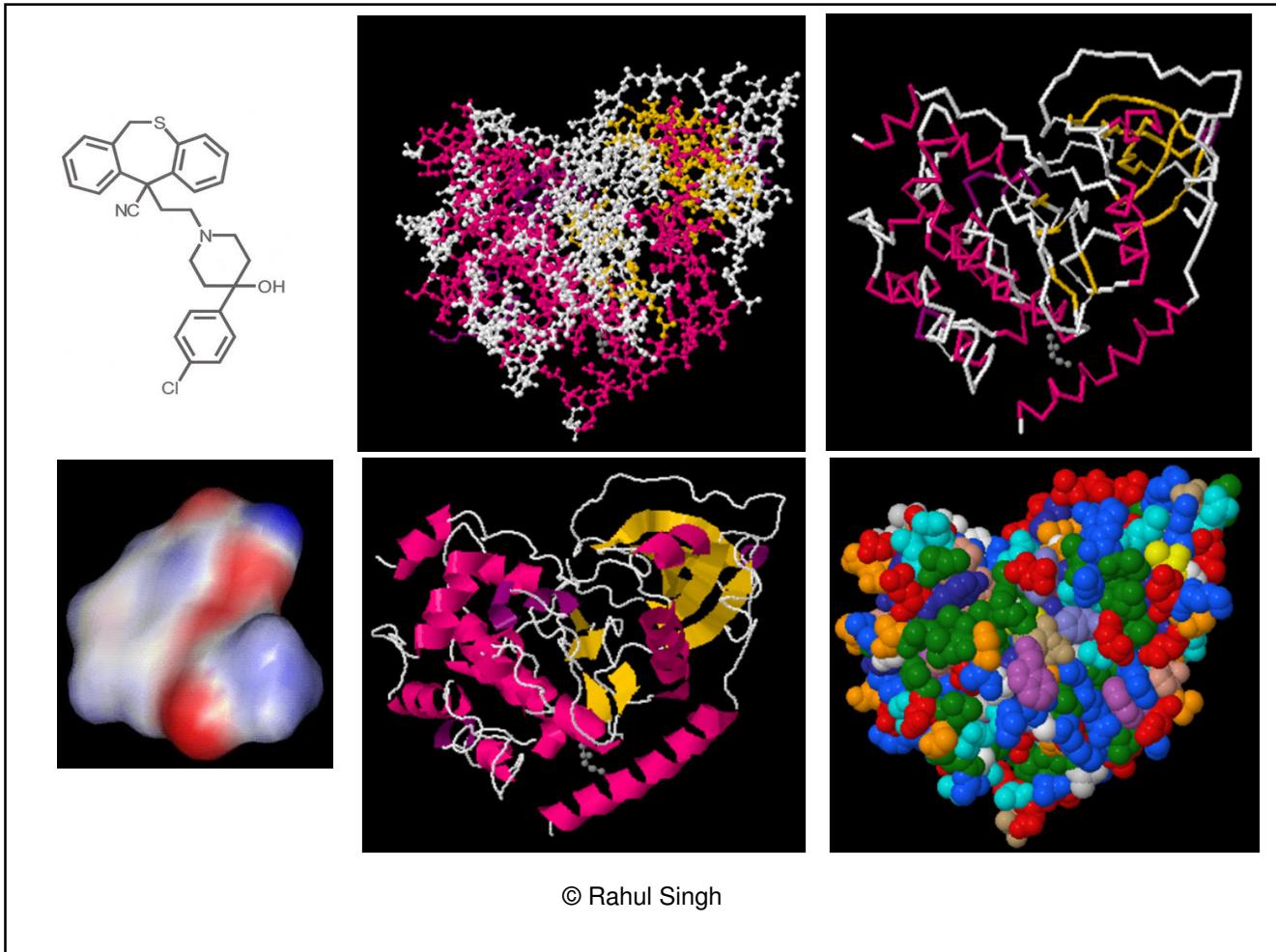
“Bioinformatics – An Introduction for Computer Scientists”, Jacques Cohen, ACM Computing Surveys, Vol. 36, No. 2, June 2004, pp. 122 – 158

“Our Secret Universe: The Hidden Life of the Cell , BBC Documentary”
<http://www.youtube.com/watch?v=Cz7agFql7iE>

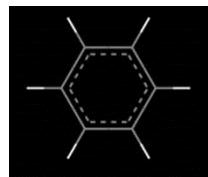
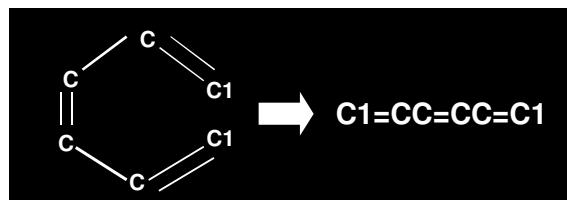
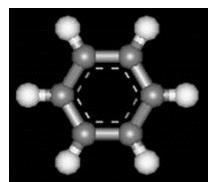
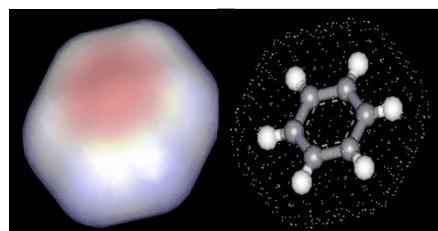
Course Organization and Plan

Lecture: The Molecular Basis of Life: DNA, RNA, and Proteins

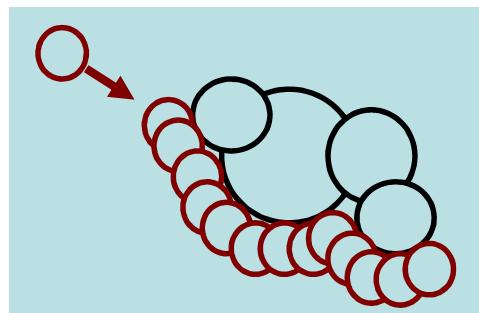
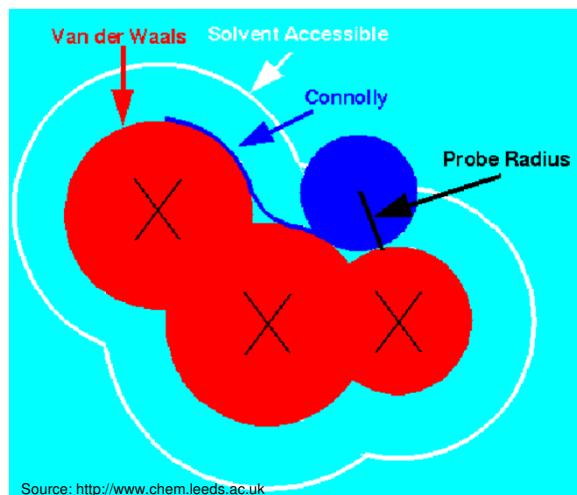
© Rahul Singh



Molecular Representations are Abstractions



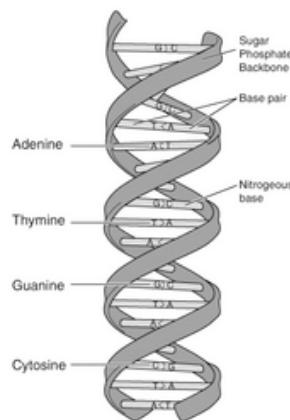
Complexity



© Rahul Singh

DNA and Genes

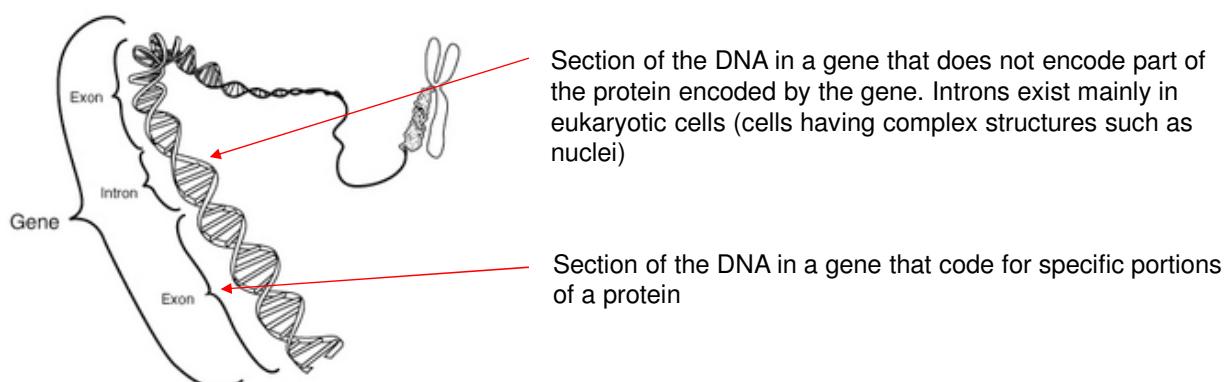
- DNA is the basis of inheriting biological characteristics. It is the genetic material
- Information stored in the DNA allows organization of molecules to form living organisms as well as inherit specific characteristics
- DNA is a Nucleic Acid: It is a high molecular weight macromolecule consisting of nucleotides.
- DNA consists of two strands of molecules that are entwined to form a double helix



© Rahul Singh

DNA and Genes

- A strand of DNA contains genes
- Additionally DNA contains areas that regulate genes and areas that don't seem to do anything (or we don't understand what these areas are doing)
- Genes can be thought of as material passed by parents to their offspring. Genes contain information for construction and regulation of proteins, polypeptides, and other molecules that control the growth and functioning of an organism

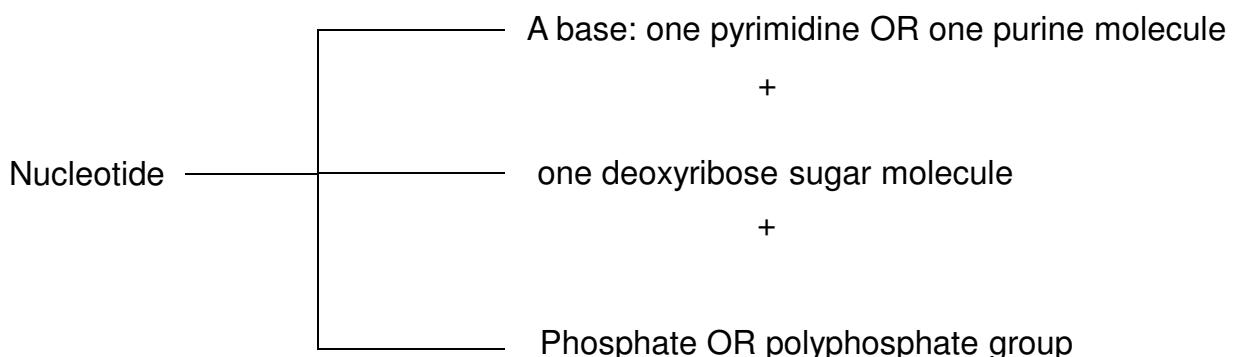


© Rahul Singh

Nucleotides

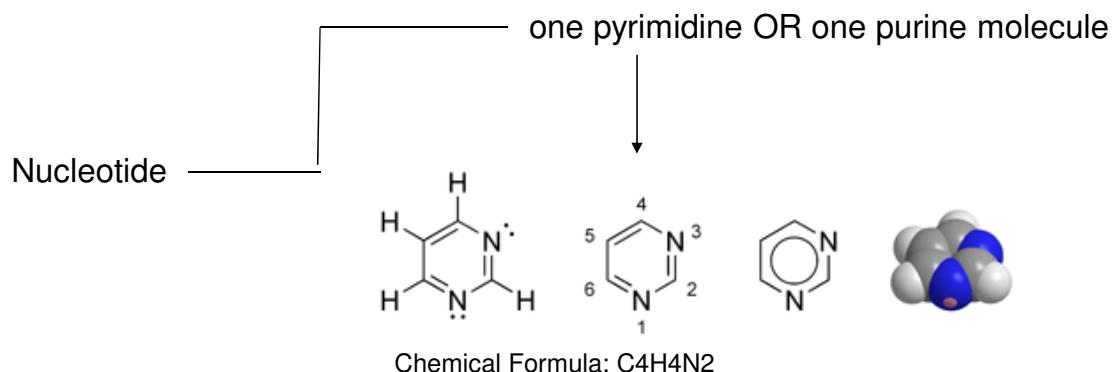
DNA is a Nucleic Acid: It is a high molecular weight macromolecule consisting of nucleotides.

A nucleotide is an organic molecule that has the following constitution:



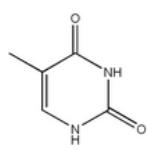
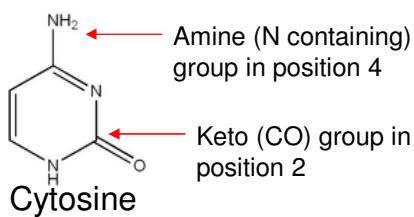
© Rahul Singh

Nucleotide Structure: Pyrimidine

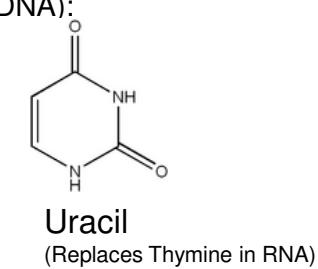


Pyrimidine is: Heterocyclic (Contains a ring structure, but has atoms other than C in the ring)
is: Aromatic (electrons are free to cycle around the circular arrangement of atoms)

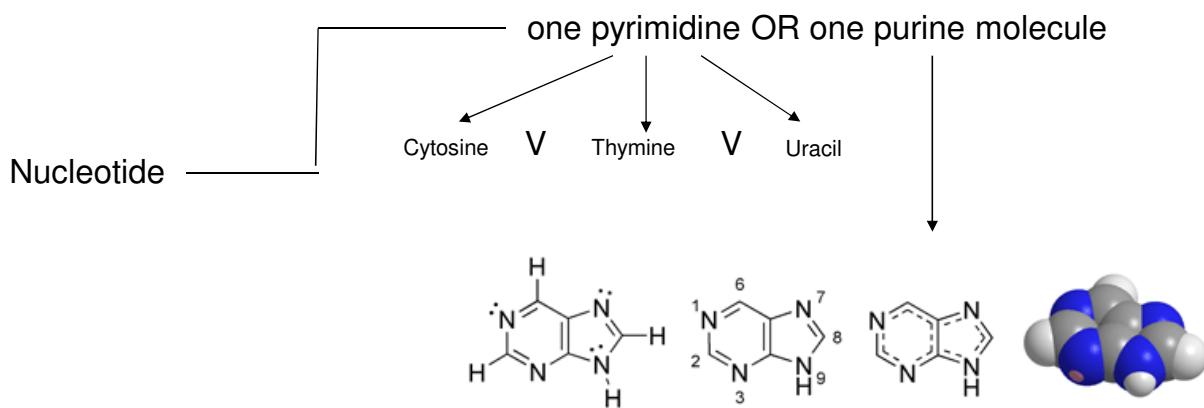
Three derivatives of Pyrimidine form bases in nucleic acids (such as in DNA):



© Rahul Singh

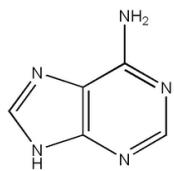


Nucleotide Structure: Purines

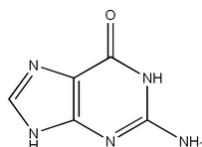


Purine is: Heterocyclic (Why?) and Aromatic. It has: a Pyrimidine ring + Imidazole

Two derivatives of Purine form bases in nucleic acids:



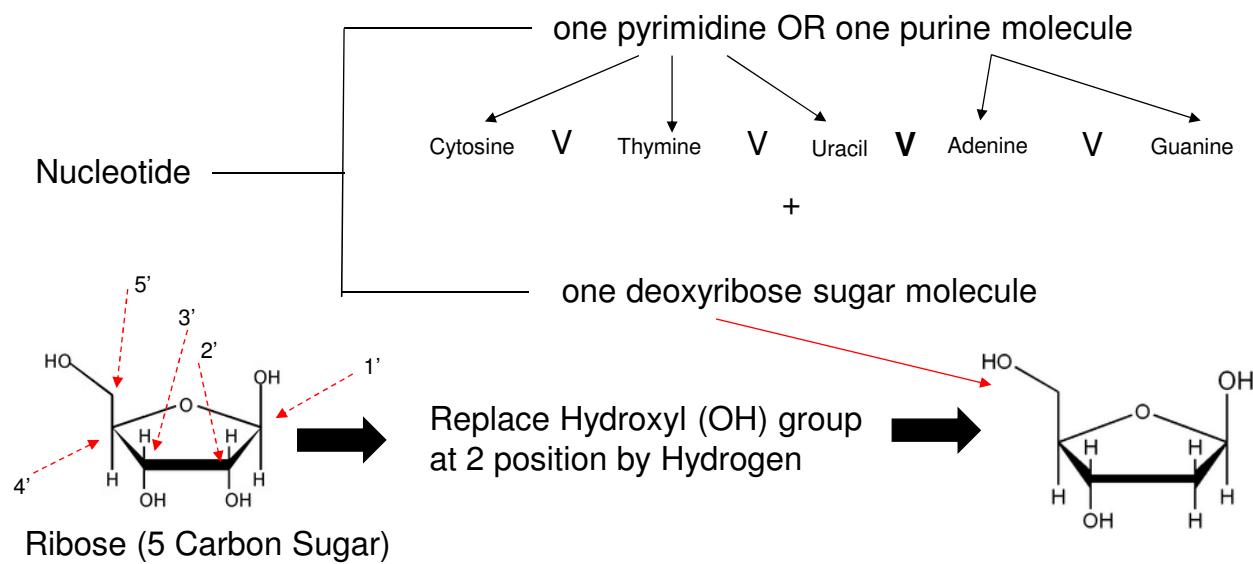
Adenine



Guanine

© Rahul Singh

Nucleotide Structure: The Bases



-The most interesting derivates of deoxyribose (or ribose) are those where a phosphate group ($[PO_4]^{3-}$) is attached to the 5 position

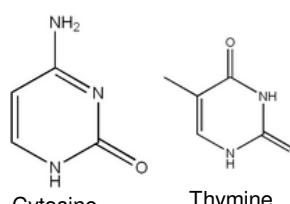
- Bases are attached to the deoxyribose (or ribose) at the 1 position

© Rahul Singh

So We Get

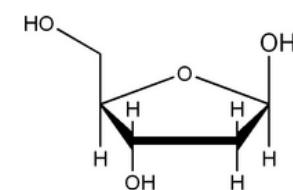
Nucleotide →

Base:



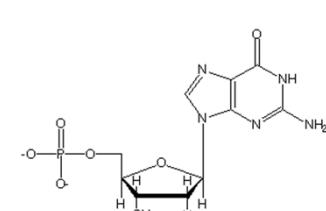
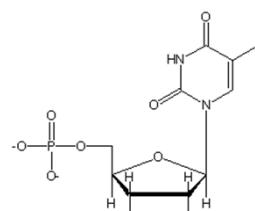
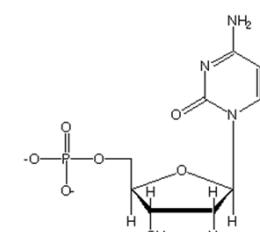
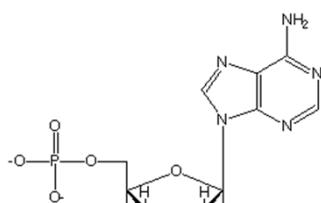
+

deoxyribose sugar molecule



+

Phosphate



Deoxyadenosine-5'-monophosphate

Deoxycytosine-5'-monophosphate

Deoxythymidine-5'-monophosphate

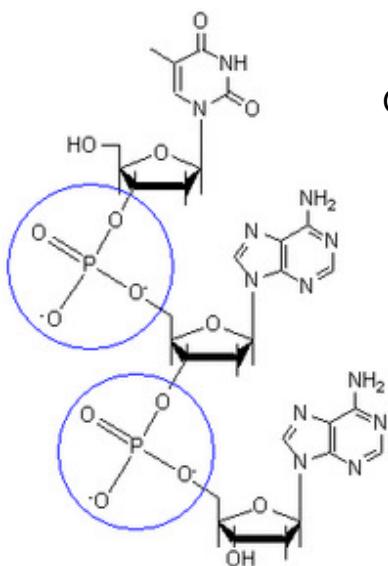
Deoxyguanosine-5'-monophosphate

-Essentially, the only thing that makes one nucleotide different from another is the base (A, C, T, or G) it contains.

© Rahul Singh

Building Larger Molecules with Nucleotides: Polynucleotide Chains

- Nucleotides can be attached to each other to make long chains of molecules. These chains are called *polynucleotide chains*
- Two Nucleotides are attached using a phosphodiester bond which connects the phosphate group of one nucleotide to the deoxyribose sugar of another nucleotide



Observations:

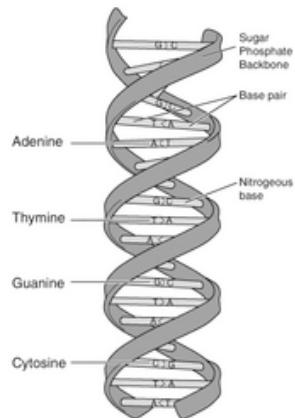
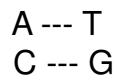
1. The phosphate group on the 5' Carbon bridges to the 3' Carbon on the deoxyribose sugar
2. Because of this, one end of the polypeptide chain has a 5' Carbon not attached to another nucleotide while the other end has a 3' Carbon that is unattached

© Rahul Singh

Base Pairing

-In the DNA double helix, the two polynucleotide chains are associated through a complimentary pairing

-This pairing is determined by each base forming a bond (specifically a H-bond) to another specific base:



-The two strands of DNA are anti-parallel to each other. That is the 5' end of one strand corresponds to the 3' end of the other strand

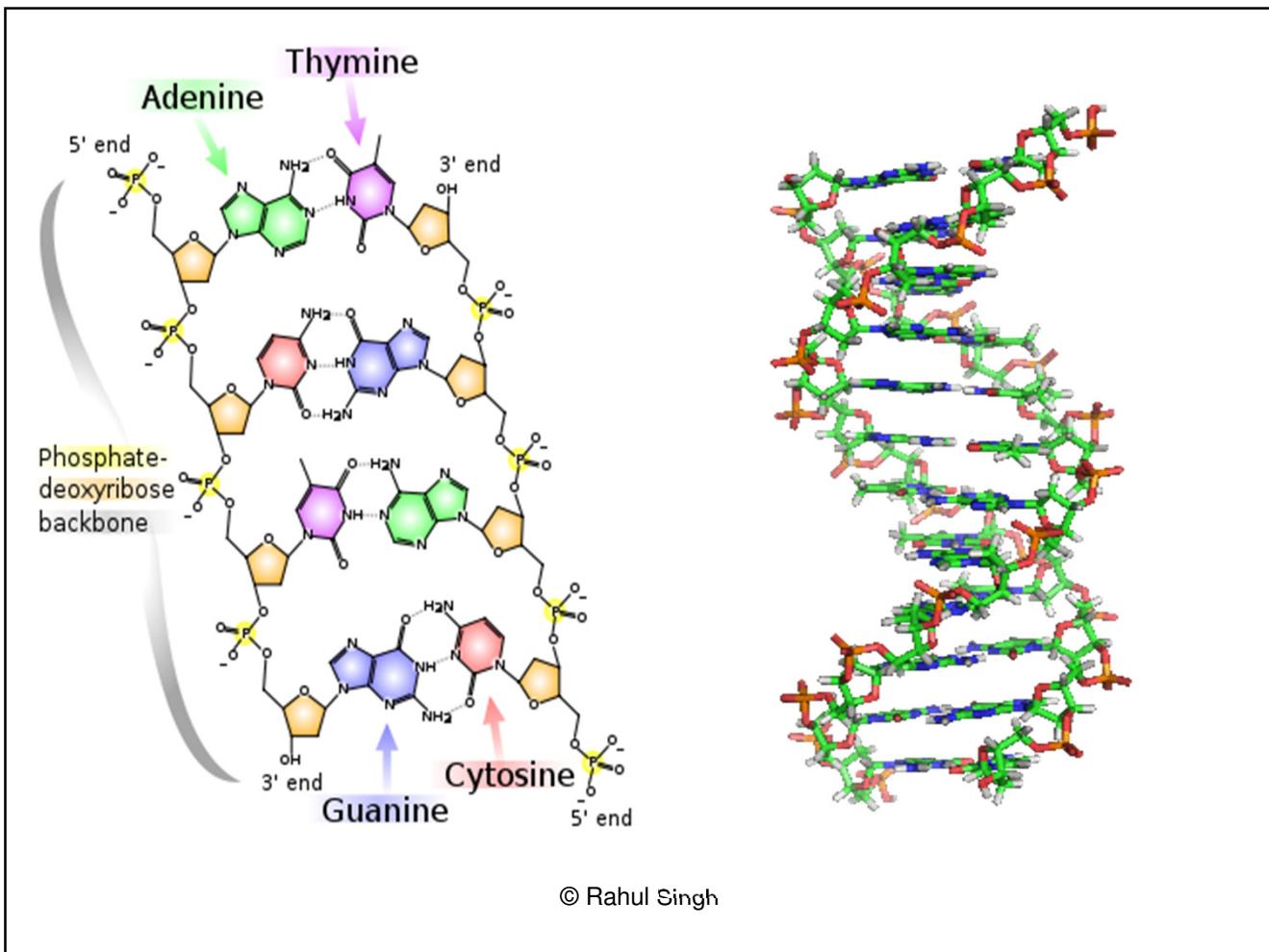
- 5' G T A T C C **A** T A A G G 3'
 ↑
 upstream downstream

-So if we have:

5' G T A T C C A T 3'

The complimentary strand will be?

© Rahul Singh



Central Dogma of Molecular Biology

-Proteins and enzymes (which are either a protein or a protein complex) are some of the most important molecules for sustaining the process of life.

e.g. Hemoglobin for carrying Oxygen

Rhodopsin for vision

Antibodies in the immune system

...

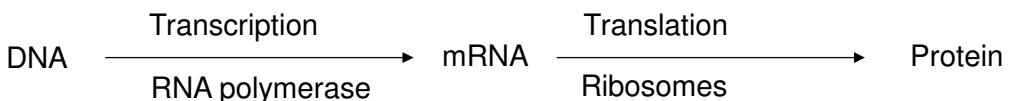
-Similarly enzymes are essential because they help speed up chemical reactions which otherwise would be too slow to sustain life.

-The process through which information is extracted from the nucleotide sequence of a gene and then used to synthesize a protein is called the central dogma of molecular biology. This process consists of the following steps:

Step-1: Information stored in the DNA is used to make a transient single stranded polynucleotide called mRNA (messenger Ribonucleic acid) – this step is called Transcription.
Transcription occurs thru the activity of RNA polymerase (which is an enzyme)

Step 2: The process of converting the information in the nucleotide sequence of the RNA to a protein is called Translation.
Translation is performed by ribosomes (a complex of proteins and rRNA)
© Rahul Singh

The Central Dogma



The transcription process happens in 3 stages:

1. Initiation
2. Elongation
3. Termination

It occurs in the 5' -> 3' direction

Key Concepts:

1. Promoter: A promoter is a DNA sequence that enables a gene to be transcribed. In RNA synthesis, promoters demarcate the genes used for mRNA creation. Therefore, they ultimately control which proteins the cell manufactures
2. RNA Polymerase (RNAP): Is an enzyme that polymerizes (forms long repeating chains) ribonucleotides, according to the information present in the DNA. Note, that ribonucleotides are nucleotides where a purine or pyrimidine base is linked to a ribose molecule. The base may be Adenine, Guanine, Cytosine, or Uracil

© Rahul Singh

Quiz

- Two main differences between RNA and DNA

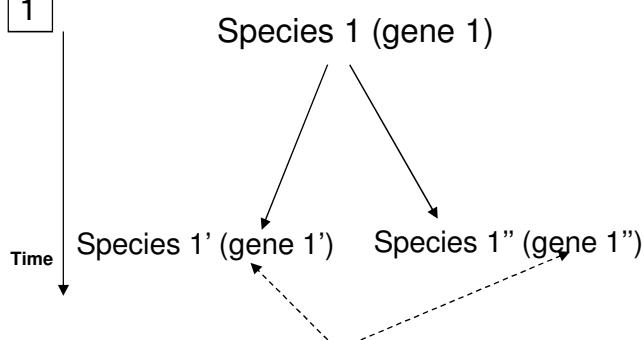
1

2

© Rahul Singh

Lets consider a problem (and a more complicated one)

1



How are these genes (1' and 1'') going to be related?

- Similar
- Not similar

2

Are two gene sequences similar if and only if they evolved from a common ancestor?

-No. For example, they can be similar by chance, or because the proteins they encode act similarly, or because of errors in DNA replication (gene duplication)

© Rahul Singh

Homology

- In the context of bioinformatics and molecular biology, *homology* typically refers to DNA or protein sequences that share ancestry
- Regions of a sequence that are homologous are also called conserved or consensus sequences
- Homology of sequences can be of two types:

Orthology: Two sequences are called orthologs if they are homologous and were separated by a speciation event (such as the case in Example 1, previous slide)

Paralogy: Two sequences are called paralogs, if they are homologous and were separated by gene duplication.

Gene duplication happens when an error occurs in DNA replication.
(DNA replication is the process of copying a double stranded DNA prior to cell division)

The error causes duplication of a region of DNA containing a gene.

© Rahul Singh

Finding Similarity Between Genes

AATCTATA
AAGATA

AATCTATA
ACTGGTAACGA

- The similarity between two sequences can be determined by obtaining a pair-wise match between the characters that constitute each sequence

- Such a match is called an alignment

- Note that, given two arbitrary sequences, there are many possible alignments

AATCTATA
AAGATA

AATCTATA
AAGATA

AATCTATA
AAGATA

- Two questions come up:

(1) What principles should direct the construction of possible alignments?

(2) How to decide which alignment is better?

© Rahul Singh

Lets address (1)

AATCTATA
AAGATA

Principles

1. A mutation occurred changing one character with another
2. An insertion occurred, adding one more (new) character to a sequence
3. A deletion occurred, removing an existing character from a sequence

Observations

1. Insertions/Deletions typically occur at a lower frequency than mutations
 2. There are no homologs of insertion/deletions. Therefore, we need to add gaps (denoted as "-"), to reflect such occurrences
- If there are no gaps, aligning two sequences only involves choosing a starting point

AATCTATA
AAGATA

AATCTATA
AAGATA

AATCTATA
AAGATA

© Rahul Singh

Which is a better alignment?

AATCTATA
AAGATA

AATCTATA
AAGATA

AATCTATA
AAGATA

-Lets say, every time there is a match between nucleotides at two corresponding positions, we give it a positive score (e.g. match= 1)

-Also, when there is no match at a position, we give it a zero

-Then, for each possible alignment, we score the match or mismatch at every position and sum the scores. This sum will tell us how good an alignment is.

AATCTATA
AAGATA
Sum = 4

AATCTATA
AAGATA
Sum = 1

AATCTATA
AAGATA
Sum = 3

© Rahul Singh

What about gaps?

AATCTATA
AAGATA- -

-We can add a “gap penalty”. For example,

gap penalty = -1 if Sequence #1 has a gap OR Sequence #2 has a gap

AATCTATA = 2
AAGATA- -

- But introducing gaps, brings up another problem (can you guess?):

© Rahul Singh

Are all gaps the same?

Consider these two possible alignments

AATCTATT	AATCTATT
ACG- AT- T	ACGATT- -
Score= 0	Score= 0

- We know, insertions and deletions are rare events
- Typically alignments that extend an existing sequence of gaps is considered to be more probable (and hence more reflective of the true possible homology) than an alignment that introduces new gaps
- To capture this idea an affine gap penalty is used. This penalty has the form:

$$\text{penalty} = G + Ln$$

G : denotes the cost of opening a new gap
 L : is the gap extension penalty
 n : is the length of the gap

By making $G > L$, lengthening existing gaps will be favored over starting new ones.

© Rahul Singh

An Example

AATCTATA
AAG- AT- A

AATCTATA
AA- G- ATA

AATCTATA
AA- - GATA

Origination penalty: -2
Length penalty: -1
Match score: +1
Mismatch score: 0

Score= -3

Score= -1

Score= 1

© Rahul Singh

The Real Problem

- If we have two sequences of 100 and 95 nucleotides each.
- How many possible alignments are there?
- To avoid exhaustively solving such large problems (and we do not need to do so), an algorithmic framework called *Dynamic Programming* is used
- This is the basis of Needleman-Wunsch and Smith-Waterman algorithms

© Rahul Singh

Sequence Alignment and Comparison

© Rahul Singh

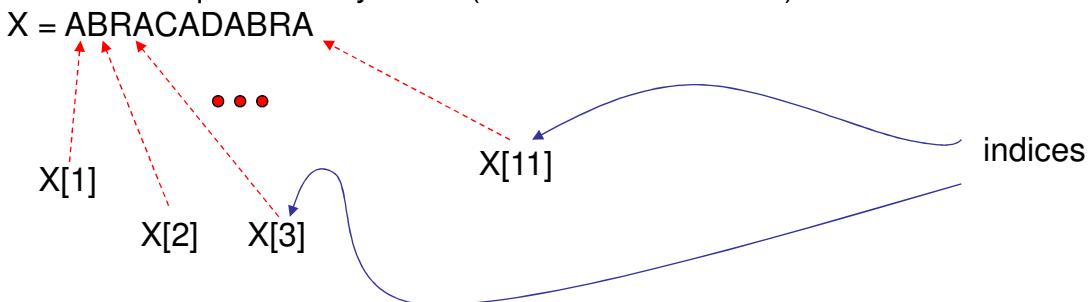
Dynamic Programming and Understanding Genes

- In 1955 R. Bellman began the systematic study of dynamic programming and was one of the key people who developed the mathematical basis of this approach
 - Typically, dynamic programming is applied to *optimization problems*. Such problems have many possible solutions and each solution can be attributed with a value. The goal is to find the solution which has an optimal (max or min) value
 - Note, that we use the term “an optimal solution” since many such solutions may be available
 - By the 1970s, the technique was well established and started to figure in textbooks (Since the 80s, commonly taught in undergrad/grad CS courses)
 - DP turned out to be fundamental to comparison of nucleotide sequences. Two of the main techniques, the Needleman-Wunsch algorithm (1970) and the Smith-Waterman algorithm (1981) are based on it.
- “Genomics pretty much would not happen without these algorithms”*

© Rahul Singh

Understanding DP: ABRACADABRA

-Consider a sequence of symbols (letters or nucleotides):



-Another sequence Z will be called a sub-sequence of X, if it has the following form:
 $Z = \text{AADA}$ or $Z = \text{ACAD}$ or ...

That is, the letters in Z correspond to a (strictly) increasing sequence of indices of X

For example, if $Z = \text{AADA}$, the sequence of indices is $\{1, 4, 7, 8\}$

if $Z = \text{ACAD}$, the sequence of indices is $\{4, 5, 6, 7\}$ or $\{1, 5, 6, 7\}$

- If $X = \text{ABRACADABRA}$ then ABRACADABR if the prefix of A
similarly, ABRACA is the prefix of the substring DABRA

© Rahul Singh

The Problem of Finding Common Sub-Sequences

X = ABCBDAB

Y = BDCABA

Z = BCA

```
graph LR; Z[\"Z = BCA\"] --> A["{2,3,6}"]; Z --> B["{1,3,4}"]
```

- Z is a common subsequence of X and Y, if Z is a subsequence of X AND Z is a subsequence of Y
- Is Z the longest common sub-sequence of X and Y ?

No, since BCBA is longer and common to both X and Y

So, given two sequences X and Y, how can we find the longest common subsequence respective to them?

© Rahul Singh

A simple but poor solution

Say X = ATGCGCTAATG

Step 1:

Generate a subsequence for X

Step 2:

Check if it's a subsequence of Y

If so, save it.

Repeat Step 1 and Step 2, till all subsequences of X are generated

Step 3:

Compare the stored sub-sequences to find the longest and that is our result.

So why is this bad?

© Rahul Singh

The Problem has Some Interesting Characteristics

Characteristic-1 (Optimal substructure of the LCS)

Let $X = x_1 x_2 \dots x_m$

$Y = y_1 y_2 \dots y_n$

Let $Z = z_1 z_2 \dots z_k$ be the LCS for X and Y

If $x_m = y_n$

Then $z_k = x_m = y_n$

AND Z_{k-1} – which is a subsequence of length $k-1$, is an LCS of X_{m-1} and Y_{n-1}

Why?

(1) $z_k = x_m = y_n$

If $z_k \neq x_m$, then we can append $x_m = y_n$ to Z and get a longer LCS (than Z), this would contradict our supposition that Z is an LCS

(2) We need to show that Z_{k-1} is an LCS of X_{m-1} and Y_{n-1} .

Lets say this was not the case – then there will exist another sequence W , which is LCS of X_{m-1} and Y_{n-1} and W is longer than Z (has length greater than $k-1$)

But then, we can append $x_m = y_n$ to W and get a LCS longer than Z .

This would contradict our assumption about Z being the LCS.

Therefore, no such W can exist!

© Rahul Singh

$X = ABCBDAB$ \rightarrow $Z = BCBA$ is the LCS
 $Y = BDCABA$

If $x_m \neq y_n$

Then $z_k \neq x_m$ implies Z is an LCS of X_{m-1} and Y

Then $z_k \neq y_n$ implies Z is an LCS of X and Y_{n-1}



$BCBA$ is LCS of $X_{m-1} = ABCBDA$ and $Y = BDCABA$

Of course!

If there were to be another sub-sequence say W , which was longer than Z and also was an LCS of X_{m-1} and Y , then it would also be an LCS of X and Y , thus contradicting our assumption that Z is LCS of X and Y

Why should this be interesting to us?

LCS of two sequences contains within it the LCS of the prefixes of these sequences

© Rahul Singh

Characteristic-2 (recursive nature of the solution)

Given $X = x_1 x_2 \dots x_m$

$Y = y_1 y_2 \dots y_n$

From the previous discussion, we know that there are either one or two sub-problems we would need to solve to get the LCS of X and Y

If $x_m = y_n$

We need to find the LCS of X_{m-1} and Y_{n-1} and append to it $x_m = y_n$ to get the LCS of X and Y

If $x_m \neq y_n$

We must solve two sub-problems: Finding the LCS of X_{m-1} and Y and finding the LCS of X and Y_{n-1} . The longer of these two is the LCS of X and Y

But each of these sub-problems require us to solve the sub-sub-problem of finding the LCS of X_{m-1} and Y_{n-1} .

So we can solve the overall problem by solving the sequence of the sub-sub-sub problems – this allows us to solve this class of problems effectively

© Rahul Singh

Example of DP: Solving the LCS Problem

Think of this as a three-course meal:

The appetizer (a formula):

Let $c[i, j]$ be the length of the LCS of the sequences X_i and Y_j

$$c[i, j] = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1, & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]), & \text{if } i, j > 0, \text{ and } x_i \neq y_j \end{cases}$$

The main course (the algorithm):

- The algorithm takes two sequences $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_m$ as inputs
- It stores the $c[i, j]$ values in a table

c_{00}	c_{01}	
c_{10}	c_{11}	
		• • •

b_{00}	b_{01}	
b_{10}	b_{11}	
		• • •

$b_{ij} = \uparrow \quad \leftarrow \quad \nwarrow$

- It also maintains another table b , where $b[i, j]$ “points” to the optimal sub-problem solution chosen while computing $c[i, j]$

© Rahul Singh

The LCS-Algorithm

```
m = length(X)  
n = length(Y)
```

} Initialization: Get the length of the two sequences

```
for i= 0 to m  
    c[i, 0] = 0  
for j= 0 to n  
    c[0, j] = 0
```

} Initialize the table c

0	0	•••
0		
⋮		

```
for i = 1 to m  
    do for j = 1 to n  
        do if  $x_i = y_j$   
            then c[i,j] = c[i-1, j-1] + 1  
                b[i,j] = ↘  
  
        else if c[i-1, j] >= c[i, j-1]  
            then c[i, j] = c[i-1, j]  
                b[i, j] = ↑  
  
        else c[i, j] = c[i, j-1]  
            b[i, j] = ←
```

} Using the optimal substructure to efficiently solve the problem

```
return(c and b)
```

© Rahul Singh

An Example (the dessert)

$m = \text{length}(X)$
 $n = \text{length}(Y)$

```

for i= 0 to m
  c[i, 0] = 0
for j= 0 to n
  c[0, j] = 0

for i = 1 to m
  do for j = 1 to n
    do if  $x_i = y_j$ 
      then  $c[i, j] = c[i-1, j-1] + 1$ 
          b[i, j] = ↘
    else if  $c[i-1, j] \geq c[i, j-1]$ 
      then  $c[i, j] = c[i-1, j]$ 
          b[i, j] = ↑
    else  $c[i, j] = c[i, j-1]$ 
          b[i, j] = ←
  return(c and b)

```

	$j=$	0	1	2	3	4	5	6
$i=$	y_j	B	D	C	A	B	A	
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

© Rahul Singh

And the LCS is . . .

$X = ABCBDAB$

$Y = BDCABA$

LCS = BCBA

A~~BCBDA~~B

~~BDCABA~~

And how to get this?

$j =$	0	1	2	3	4	5	6
$i =$	y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0
1	A	0	0	0	1	1	1
2	B	0	1	1	1	2	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3
5	D	0	1	2	2	2	3
6	A	0	1	2	2	3	3
7	B	0	1	2	2	3	4

Start and follow the arrows

© Rahul Singh

And How long did it take us?

	$j=$	0	1	2	3	4	5	6
$i=$	y_j	B	D	C	A	B	A	
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

- All we did was fill up the table and calculate the arrows: there were $6 \times 7 = 42$ entries.
- In general for two sequences of length m and n, there would be $m \times n$ entries.
- An estimate of the amount of work we did is called the (time) complexity of the algorithm

© Rahul Singh

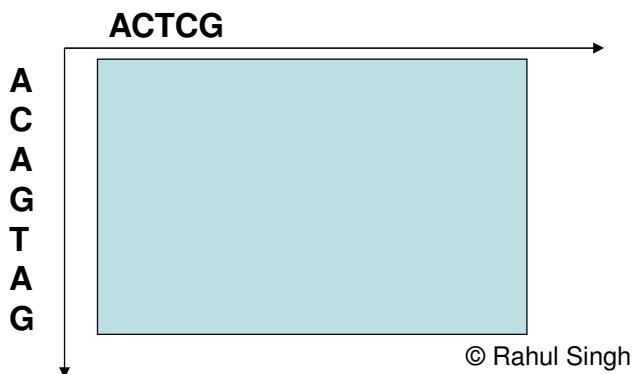
The Needleman-Wunsch Algorithm for Sequence Alignment

- Published in 1970 in Journal of Molecular Biology
- A straightforward application of dynamic programming

Basic Approach

Consider the problem of aligning the following two sequences: **ACAGTAG** and **ACTCG**

1. Begin by creating a table, where each sequence is aligned along a specific axis



© Rahul Singh

2. Determine a cost model

- For example, gap penalty = -1; match score = +1; mismatch score = 0;
The first row and column of the table are initialized with multiples of the gap penalty

	$j =$	0	1	2	3	4	5
$i =$	y_j	A	C	T	C	G	
0	x_i	0	-1	-2	-3	-4	-5
1	A	-1					
2	C	-2					
3	A	-3					
4	G	-4					
5	T	-5					
6	A	-6					
7	G	-7					

3. Start by filling in the table starting from position (1, 1)

© Rahul Singh

$j=$	0	1	2	3	4	5
$i=$	y_j	A	C	T	C	G
0	x_i	0	-1	-2	-3	-4
1	A	-1	X			
2	C	-2		X		
3	T	-3			X	
4	C	-4				X
5	G	-5				

$j=$	0	1	2	3	4	5
$i=$	y_j	A	G	C	C	G
0	x_i	0	-1	-2	-3	-4
1	A	-1	X			
2	G	-2		X	-	
3	G	-3				X

$j=$	0	1	2	3
$i=$	y_j	A	G	G
0	x_i	0	-1	-2
1	A	-1	X	
2	G	-2		X
3	C	-3		-
4	C	-4		-
5	G	-5		X

4. A diagonal move indicates an alignment of nucleotides from each sequence
5. A horizontal move indicates a gap in the sequence along the left axis, while a vertical move indicates a gap in the sequence along the top axis
6. The table is filled in a row-major order – that is each row is filled from left-to-right as we go from top to bottom of the table

© Rahul Singh

	$j =$	0	1	2	3	4	5
$i =$	y_j	A	C	T	C	G	
0	x_i	0	-1	-2	-3	-4	-5
1	A	-1					
2	C	-2					
3	A	-3					
4	G	-4					
5	T	-5					
6	A	-6					
7	G	-7					

7. For any position, we encounter three possible choices:
- A gap in the top sequence – which will induce a vertical move
 - A gap in the left sequence – which induces a horizontal move
 - An alignment (match or mismatch) of nucleotides, inducing a move along the diagonal
8. Just as in the LCS problem, we compute the costs appropriately and pick the choice that gives the highest value and employ an arrow to mark the previous correspondence

© Rahul Singh

Lets consider some examples. (a) What is the value at position (1, 1):

$$\begin{aligned} \text{Value at (1, 1)} &= \max \{ (-1 + \text{gap in the top sequence}), (-1 + \text{gap in the left sequence}), \\ &\quad (0 + \text{match/mismatch cost}) \} \\ &= \max \{ (-2), (-2), 1 \} = 1 \end{aligned}$$

	$j =$	0	1	2	3	4	5
$i =$	y_j	A	C	T	C	G	
0	x_i	0	-1	-2	-3	-4	-5
1	A	-1	1	0	-1	-2	-3
2	C	-2	0	2	1	0	-1
3	A	-3	-1	1	2	1	0
4	G	-4	-2	0	1	2	2
5	T	-5	-3	-1	1	1	2
6	A	-6	-4	-2	0	1	1
7	G	-7	-5	-3	-1	0	2

(b) Value at (1, 2):
 $= \max\{ (-2-1)\uparrow, (-1+0)\leftarrow, (1-1)\leftarrow \}$
 $= 0 \leftarrow$

(c) Value at (7, 5):
 $= \max\{ (1-1)\uparrow, (1+1)\leftarrow, (0-1)\leftarrow \} = 2$

The optimal alignment can be obtained by following the arrows from (m,n) to (0,0)

- **Exercise:** Calculate the “arrows” at each of the other positions in the matrix
- Can there be alternate optimal alignments for this example? Find them

© Rahul Singh

Converting a Path to an Alignment

	$j =$	0	1	2	3	4	5
$i =$	y_j	A	C	T	C	G	
0	x_i	0	-1	-2	-3	-4	-5
1	A	-1	1	0	-1	-2	-3
2	C	-2	0	2	1	0	-1
3	A	-3	-1	1	2	1	0
4	G	-4	-2	0	1	2	2
5	T	-5	-3	-1	1	1	2
6	A	-6	-4	-2	0	1	1
7	G	-7	-5	-3	-1	0	2

AC - TCG
ACAGTAG
C - TCG
CAGTAG
- TCG
AGTAG

1. The alignment is reconstructed right-to-left
2. If the arrow is diagonal, the two nucleotides are aligned
3. If the arrow is vertical, a gap is placed in the top sequence and it is then aligned with the nucleotide from the left sequence
4. If the arrow is leftwards, a gap is placed in the left sequence and aligned with the corresponding nucleotide in the top sequence

© Rahul Singh

Semi-Global Alignments

AC- - TCG
ACAGTAG

- In the previous approach, we looked at what is called “global alignment”
- In global alignment, the two sequences are compared in their entirety and the gap penalty is applied regardless of whether the gaps were internal (in a sequence) or at the end of one or both of the sequences

- What about the following case?

AAACACGTGTCT → AAACACGTGTCT
ACGT ----- ACGT -----

- Of the possible alignments, this is most interesting because it shows that the shorter sequence is completely contained in the longer sequence
- Such terminal gaps typically are the result of incomplete data acquisition and therefore should be treated differently

- How to do this?

© Rahul Singh

AAACACGTGTCT	\rightarrow	AAACACGTGTCT
ACGT		---- -ACGT-----

Goal: We would like to “adjust” our dynamic programming approach get such alignments

Step 1: We know: *A horizontal move indicates a gap in the sequence along the left axis, while a vertical move indicates a gap in the sequence along the top axis*

Consider the following sub-case of the above example:

ACACGT	
- - ACGT	

$j =$	0	1	2	3	4	
$i =$	y_j	A	C	G	T	
0	x_i	0	-1	-2	-3	-4
1	A	-1	1	0	-1	-2
2	C	-2	0	2	1	0
3	A	-3	-1	1	2	1
4	C	-4	-2	0	1	2
5	G	-5	-3	-1	1	1
6	T	-6	-4	-2	0	2

- To get the desired path, we need to: make the cost of (2,0) and (1,0) = 0
- We can do this by initializing the first column of the table by zeros. This means we do not penalize for gaps in the beginning of the (top) sequence.
- Similarly, we initialize the first row with zeros (to allow gaps in the left sequence)

© Rahul Singh

Step 2: What do we do for gaps at the end of a sequence?

		ACACTGATCG											
		ACACTG - - -											
		$j =$	0	1	2	3	4	5	6	7	8	9	10
$i =$	y_j	A	C	A	C	T	G	A	T	C	G		
0	x_i	0	0	0	0	0	0	0	0	0	0	0	0
1	A	0	1*	0	1	0	0	0	1	0	0	0	0
2	C	0	0	2*	1	2	1	0	0	1	1	0	0
3	A	0	1	1	3*	2	2	1	1	0	1	1	1
4	C	0	0	2	2	4*	3	2	1	1	1	1	1
5	T	0	0	1	2	3	5*	4	3	2	1	1	1
6	G	0	0	0	1	2	3	6*	6*	6*	6*	6*	6*

-Main point: We need to make the horizontal moves in the last row – and these are penalized. So to allow gaps at the end of the left sequence, we let horizontal moves in last row occur without penalty

- Similarly we allow vertical moves in the last column without penalty for gaps in the top sequence

© Rahul Singh

Local Alignments

Consider the following example:

AACCTATAGCT		AAC- CTATA GCGATATA
		Gap= -1
		Match= +1

Mismatch= -1

-Looks like a bad alignment: First five position-alignments are gaps/mismatches
Last three are gaps

BUT

The match reveals that the subsequence motif TATA is common to both the sequences

-This is different from semi-global alignment, where we only allow gaps at the beginning and end of the subsequences

-Here, we are interested in finding a matching subsequence between two sequences.
That is, we are ready to ignore mismatches and gaps before and after the matching region

-Semi-global alignment will not work since each alignment will be penalized for the non-matching positions

© Rahul Singh

The Smith-Waterman Algorithm

- In the dynamic programming approach, while filling in the table do the following:
 - If at any point during the partial alignment process, the best score possible at a point is negative, place a zero at that position
 - Complete the table
 - Find the maximum partial alignment score in the table and starting from it, work backwards till a zero is reached
 - The resulting alignment, represents the best subsequence match between the sequences.

© Rahul Singh

	$j =$	0	1	2	3	4	5	6	7	8	9	10	11
$i =$	y_j	A	A	C	C	T	A	T	A	G	C	T	
0	x_i	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	0	0	0	0	0	0	0	1	0	0	0
2	C	0	0	0	1	1	0	0	0	0	0	2	1
3	G	0	0	0	0	0	0	0	0	0	1	0	1
4	A	0	1	1	0	0	0	1	0	1	0	0	0
5	T	0	0	0	0	0	1	0	2	1	0	0	1
6	A	0	1	1	0	0	0	2	0	3	2	1	0
7	T	0	0	0	0	0	1	1	3	2	2	1	2
8	A	0	1	1	0	0	0	2	2	4	3	2	1

Gap = -1, Match = +1, mismatch = -1

The matching subsequence

$\max\{ 0 \vee -1 \vee -1 \vee -1 \} = 0$

$\max\{ 0 \vee 1 \vee -1 \vee -1 \} = 1$

$\max\{ 0 \vee 2 \vee 0 \vee -1 \} = 2$

© Rahul Singh

Can we do better than quadratic space-time for global alignment problems?

Time: To an extent. Non-trivial lower bounds remain unknown. Theoretically, there exists a sub-quadratic approach with complexity $O(n^2/\log n)$ called the Four Russian's (should be called the Four Soviet's) algorithm proposed by Vladimir Arlazarov, Efim Dinic, Mikhail Kronrod, and Igor Faradzev in a paper titled "On Economical Construction of the Transitive Closure of an Oriented Graph", *Soviet. Math. Dokl.*, 11:1209-1210, 1970

There are however practical considerations that can limit the applicability of this method.

Space: Yes. The Hirschberg Algorithm. Can be easily used in practice
D. Hirschberg, "A linear space algorithm for computing maximal common subsequences", *Comm. of the ACM*, 18:341-343, 1975

Linear Space Alignment: The Hirschberg Algorithm

The dynamic programming solution to the LCS problem requires:

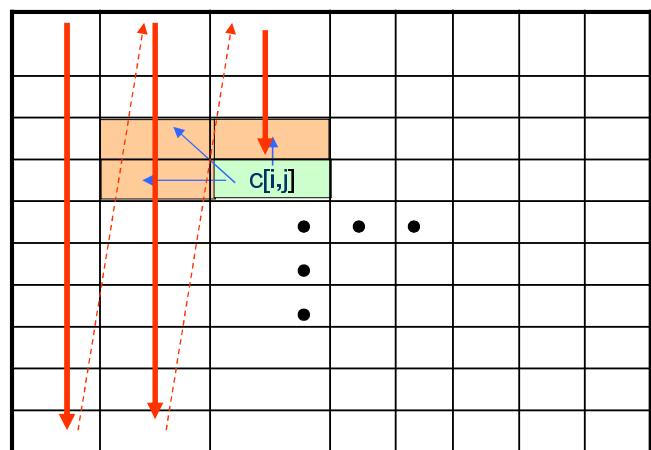
$O(n \times m)$ space

$O(n \times m)$ time

The Hirschberg algorithm can be used to solve this problem in linear space

Basic Idea (Part 1): $c[i, j]$ can be computed in linear space

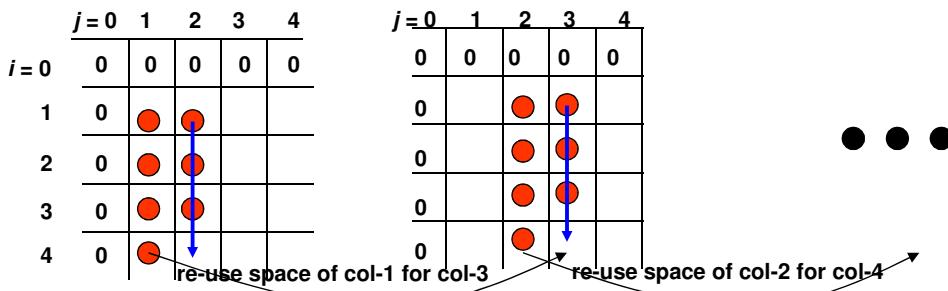
$$c[i, j] = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1, & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]), & \text{if } i, j > 0, \text{ and } x_i \neq y_j \end{cases}$$



© Rahul Singh

Linear Space Computation of the Optimal Score

We only need the previous column to compute the current column. So we only need the space for two columns and can “recycle” the $j-1^{\text{th}}$ column once the j^{th} column is computed



$m = \text{length}(X)$, $n = \text{length}(Y)$, initialize $c[i, j] = 0$

```

for i = 1 to m
  do for j = 1 to n
    do if  $x_i = y_j$ 
      then  $c[i, j] = c[i-1, j-1] + 1$ 
      b[i, j] = ↘
    else if  $c[i-1, j] \geq c[i, j-1]$ 
      then  $c[i, j] = c[i-1, j]$ 
      b[i, j] = ↑
    else  $c[i, j] = c[i, j-1]$ 
      b[i, j] = ←
return(c and b)
  
```

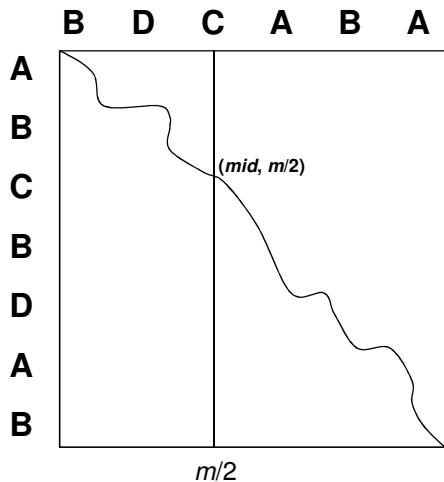
$m = \text{length}(X)$, $n = \text{length}(Y)$, Allocate (column[1]), Allocate (column[2])

```

for j = 1 to n
  if j > 2, then:
    Free( column[j - 2] )
    Allocate( column[j] )
  for i = 1 to m
    do if  $x_i = y_j$ 
      then  $c[i, j] = c[i-1, j-1] + 1$ 
      b[i, j] = ↘
    else if  $c[i-1, j] \geq c[i, j-1]$ 
      then  $c[i, j] = c[i-1, j]$ 
      b[i, j] = ↑
    else  $c[i, j] = c[i, j-1]$ 
      b[i, j] = ←
  return(c and b)
  
```

Computing the Optimal Score and Optimal Alignment

If $X = \text{aatgccta}$ then we shall denote by X_r the reverse string, such that, $X_r = \text{atccgtaa}$. Similarly, if c is the cost of matching X and Y , then c_r is that of matching X_r and Y_r .



Observation: The longest path from $(0, 0)$ to (n, m) passes through some point on the middle column.

Let us denote this (unknown) mid-point by $(\text{mid}, m/2)$

How can we find $(\text{mid}, m/2)$?

Let us denote by $L(i)$ the length of the path from $(0, 0)$ to (n, m) that passes through $(i, m/2)$

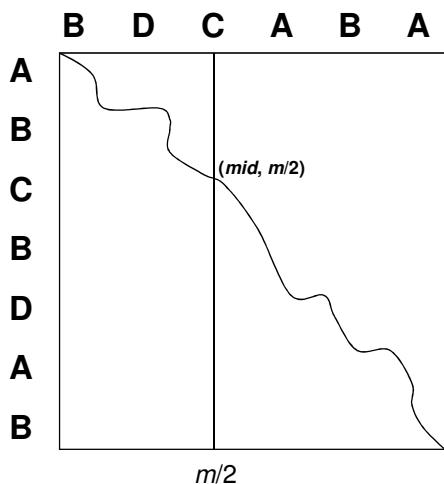
Next, Let the path $(0, 0), \dots, (i, m/2)$ be called the prefix subpath. Its length is $L(\text{prefix}(i))$. Similarly, let the path $(i, m/2), \dots, (n, m)$ be called the suffix, with length $L(\text{suffix}(i))$.

Then, $L(i) = L(\text{prefix}(i)) + L(\text{suffix}(i))$, further

$$L(\text{mid}) = \arg \max_i L(i) \quad \text{Let us denote this } i \text{ (corresponding to the largest } L(i) \text{ by } i^*$$

Thus $\text{mid} = i^*$

Computing the Optimal Score and Optimal Alignment



We have:

$$(0,0), \dots (i, m/2) : \text{prefix subpath} \longrightarrow L(\text{prefix}(i)) \quad (1)$$

$$(i, m/2), \dots (n, m) : \text{suffix subpath} \longrightarrow L(\text{suffix}(i)) \quad (2)$$

and

$$L(i) = L(\text{prefix}(i)) + L(\text{suffix}(i)) \quad (3)$$

Note, (2) is equivalent to saying that the suffix subpath is the longest subpath from (n, m) to $(i, m/2)$, that is we match: X_r and Y_r in the range (n, m) to $(i, (m/2))$

That is (assume m is even):

$$c[n, m] = \max_{i=0 \dots n} c[i, m/2] + c_r[n-i, m/2]$$

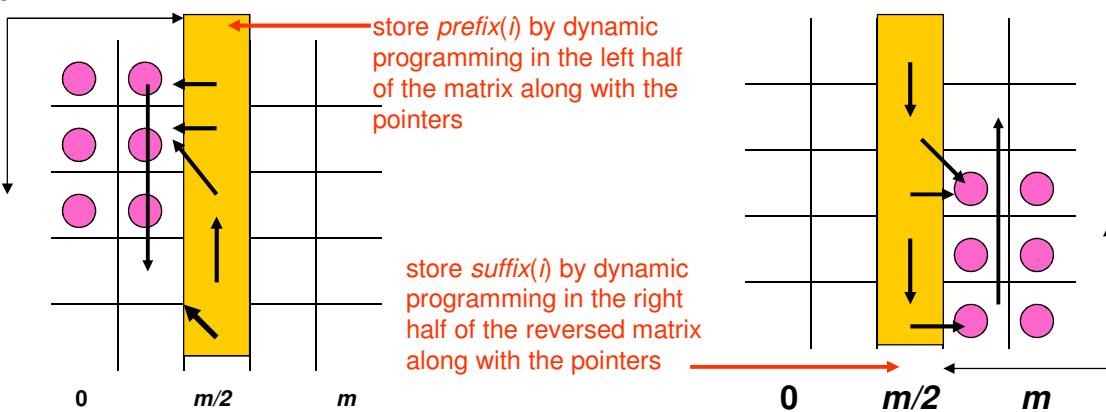
↑ ↑
prefix(i) suffix(i)

We know:

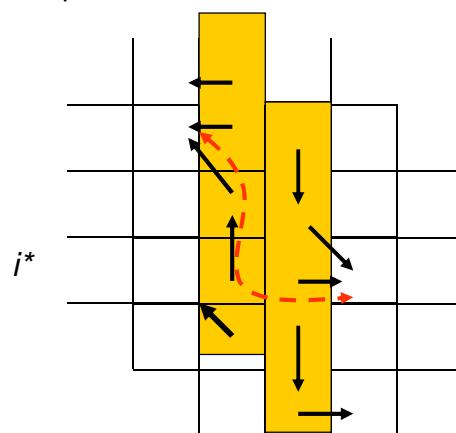
(1) $L(i^*) = L(\text{prefix}(i^*)) + L(\text{suffix}(i^*))$ - is the length of the longest path

(2) both $L(\text{prefix}(i^*))$ and $L(\text{suffix}(i^*))$ can be computed in linear space
(see the two columns pseudocode)

Now we can put together the entire algorithm

Step 1

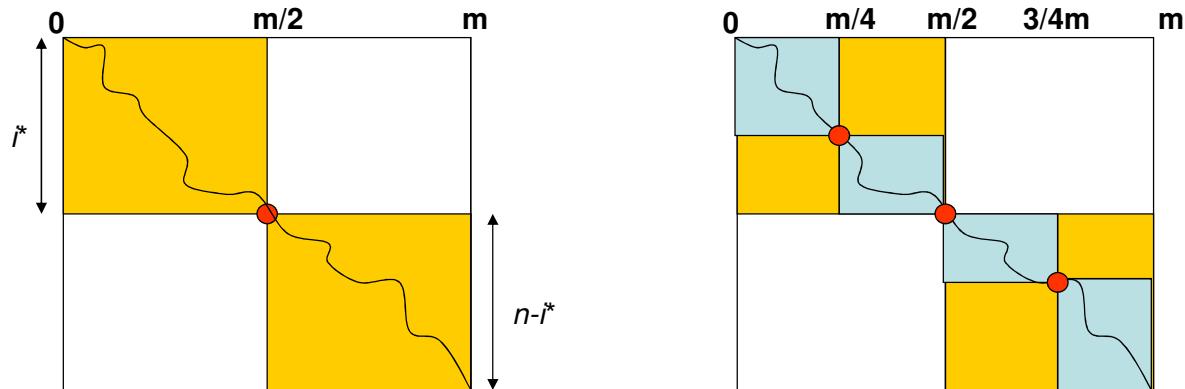
Step 2 $L(i) = L(\text{prefix}(i)) + L(\text{suffix}(i))$, that is, add $\text{prefix}(i)$ and $\text{suffix}(i)$ to get $L(i)$ and identify i^* . Further, identify the path that exits column $m/2$ from row i^* (red dotted line)



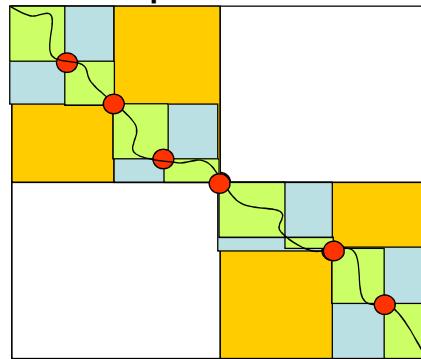
Step 3: Divide and conquer

The problem of finding the longest path from $(0, 0)$ to (n, m) can be broken up into two problems:

- finding the longest path from $(0, 0)$ to $(mid, m/2)$
- finding the longest path from $(mid, m/2)$ to (n, m)



Step 4: Recursion till the start and end points are in consecutive columns



The Hirschberg Linear Space Alignment Algorithm: Pseudocode

Hirschberg-LSA(k, k', r, r'): (align strings $x_k \dots x_{k'}$ with $y_r \dots y_{r'}$)

Let $mid = \lceil (k'-k)/2 \rceil$

Find

The optimal path, P_{mid} , entering column $mid-1$ and exiting column mid where the optimal path is the one with the longest length

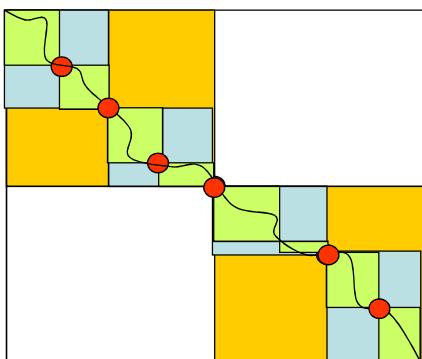
Let $M1$ be the row number at column $mid-2$ where P_{mid} enters
 $M2$ be the row number at column $mid+1$ where P_{mid} exits

Hirschberg-LSA($k, mid-2, r, M1$)

Output P_{mid}

Hirschberg-LSA($mid+1, k', M2, r'$)

Complexity?



Space = $2m$

Time = area = $m \times n + (m \times n)/2 + (m \times n)/4 + \dots$
 $\leq 2 \times (m \times n)$

Proteins, Protein Sequences and Alignment

© Rahul Singh

DNA and Protein (Amino Acid) Sequence Matching

- So far we have looked at matching of DNA sequences only
- The algorithms and techniques developed by us are equally suitable for matching protein (amino-acid) sequences
- So what are protein sequences? How are they created?

© Rahul Singh

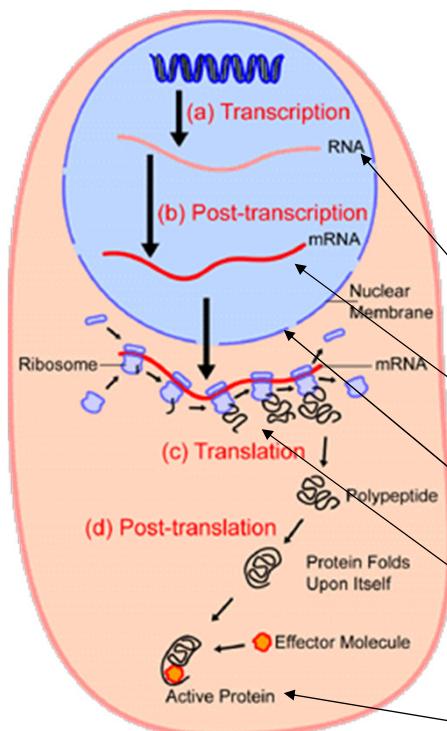
Protein Biosynthesis: The Big Picture

- The genetic information carried by an organism is inscribed in a DNA molecule
- Each functional portion of this molecule is called a gene
- For protein synthesis, each gene is *transcribed* into a molecule called RNA
- The RNA molecule is *translated* by mediation of ribosomes, transfer RNA, and associated enzymes into an amino acid chain
- The amino-acid chain then *folds* into a protein

© Rahul Singh

Protein Biosynthesis

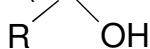
- Process through which cells build proteins
- Multi-step process starting with transcription and ending with translation
- Though similar, differs between eukaryotes and prokaryotes
- Illustration of the main steps:
 1. In the nucleus, genes are transcribed into RNA
 2. The RNA is subjected to modification and control, resulting in formation of mRNA
 3. mRNA is then transported out of the nucleus into the cytoplasm
 4. Ribosomes translate the mRNA into poly-peptide chains that fold to form proteins
 5. The proteins are often further modified (such as binding with an effector molecule) to become fully active



© Rahul Singh

Amino Acids

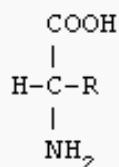
- An Amino Acid is a molecule that contains both an amino (-NH_2) and a carboxylic acid ($\text{C}=\text{O}$) functional groups



-Amino acids form polymers (long repeating chains of atoms) chains called polypeptides or peptides, which in turn form proteins

-Over 100 amino acids are found in nature. However, only 20 of them are genetically coded. These 20 are called proteinogenic or standard amino acids

- The general structure of a proteinogenic amino acid is:

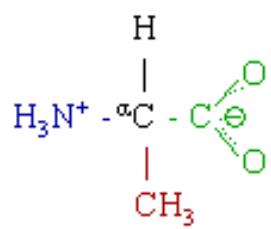
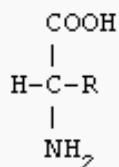


R – represents a side chain specific to each amino acid.

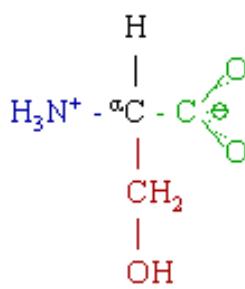
Note that the amino and the carboxylate functionalities are attached to the same C atom (referred to as the alpha carbon)

© Rahul Singh

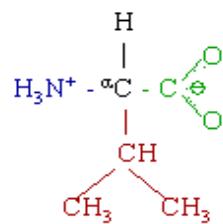
Structures of Some Amino Acids



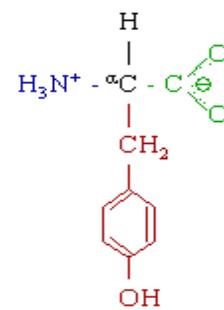
Alanine
(Ala / A)



Serine
(Ser / S)

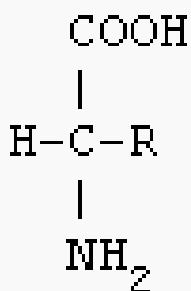


Valine
(Val / V)

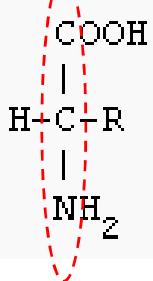


Tyrosine
(Tyr / Y)

© Rahul Singh



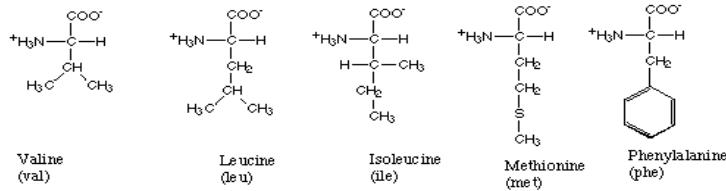
-The **backbone** of an amino acid is formed by the amino group, the alpha-carbon, and the carboxylic acid:



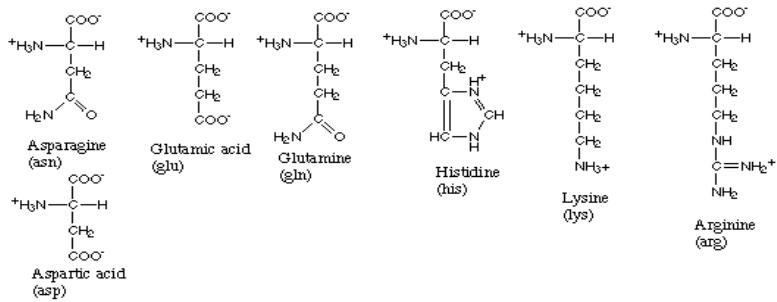
-Amino acids are grouped into three common categories: **hydrophobic** (side chain having C and H – do not form H-bonds with water molecules), **polar** (side chains with O or N which form H-bonds with water more easily), and **charged** (have a charge at biological pH)

© Rahul Singh

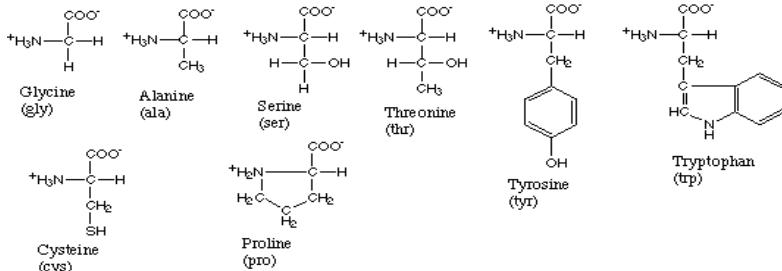
Amino acids with hydrophobic side groups



Amino acids with hydrophilic side groups



Amino acids that are in between



© Rahul Singh

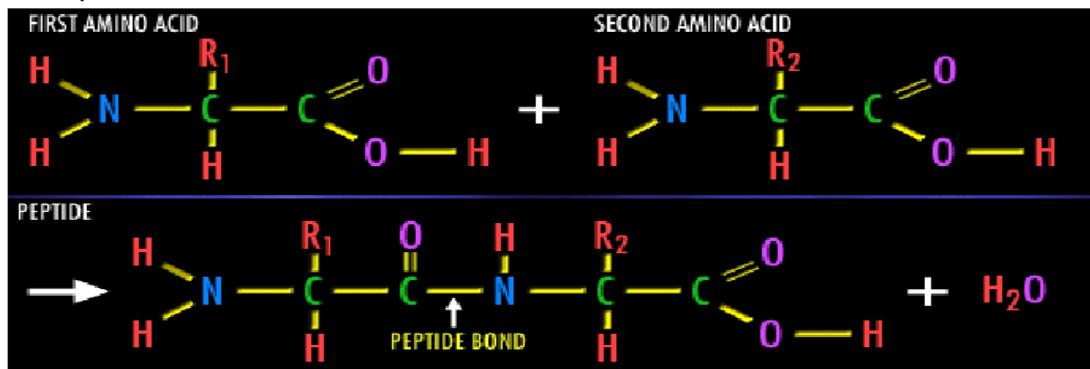
The Peptide Bond

-A chain of amino acids is called a peptide. Long chains are called polypeptides

- Two amino acids are joined by a **peptide bond**. A peptide bond is a chemical bond that is formed between two molecules when the carboxyl group of one molecule reacts with the amino group of another molecule. When this happens, a molecule of water is released (dehydration synthesis reaction)

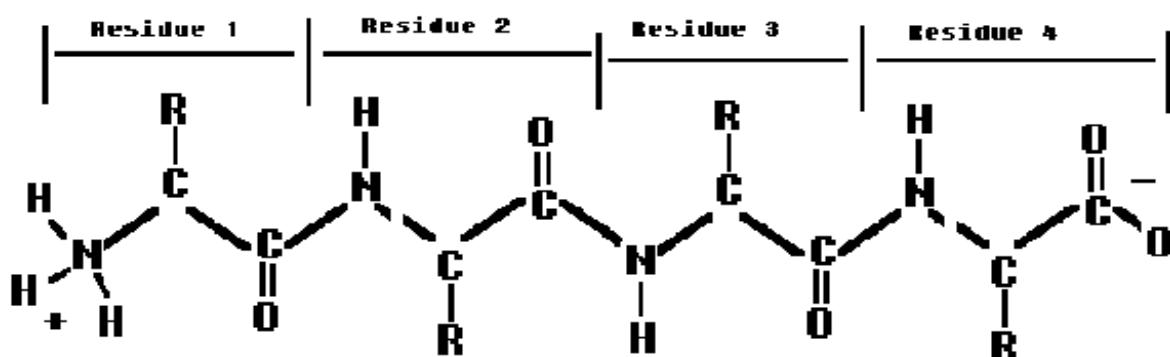


For example:



© Rahul Singh

Polypeptides



-Peptides are always referred from the amino-terminal (N-terminal) to the carboxy-terminal (C-terminal)

For example: Glycyl-alanyl-tyrosyl-glycine

Primary Structure of Proteins: The sequence of amino acids that comprise a protein. This is the basic information that determines the 3D shape of the protein, its physical, chemical, and biological properties

© Rahul Singh

Secondary Structure

To understand secondary structure, we have to consider the molecule as a discrete structure

-When we look at the structures of proteins that are known, we find a small set of local patterns that occur commonly. These structures are formed by

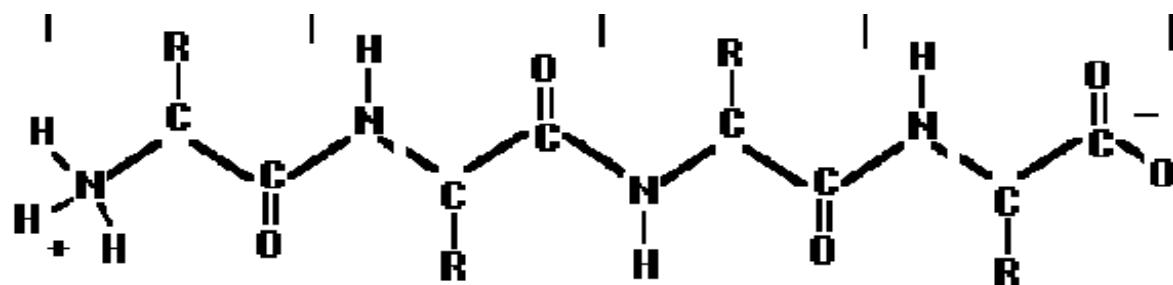
- (a) intramolecular hydrogen bonding
- (b) specific backbone torsion angles

- The location and direction of these pattern form the **secondary structure** of the protein. The two most common patterns we see are called: α - helix and β - sheet

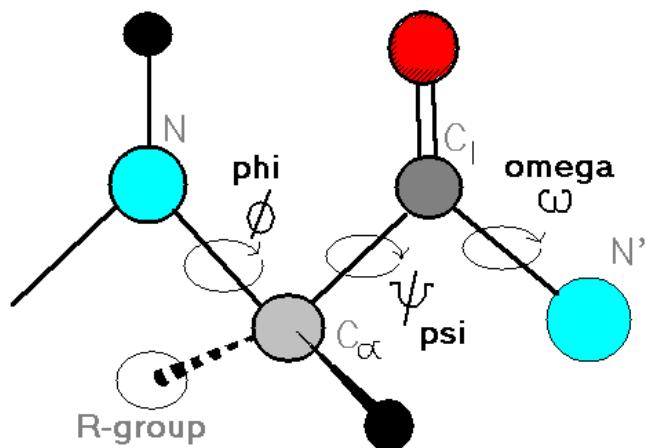
To understand these patterns, we have to understand the geometry of the protein molecule

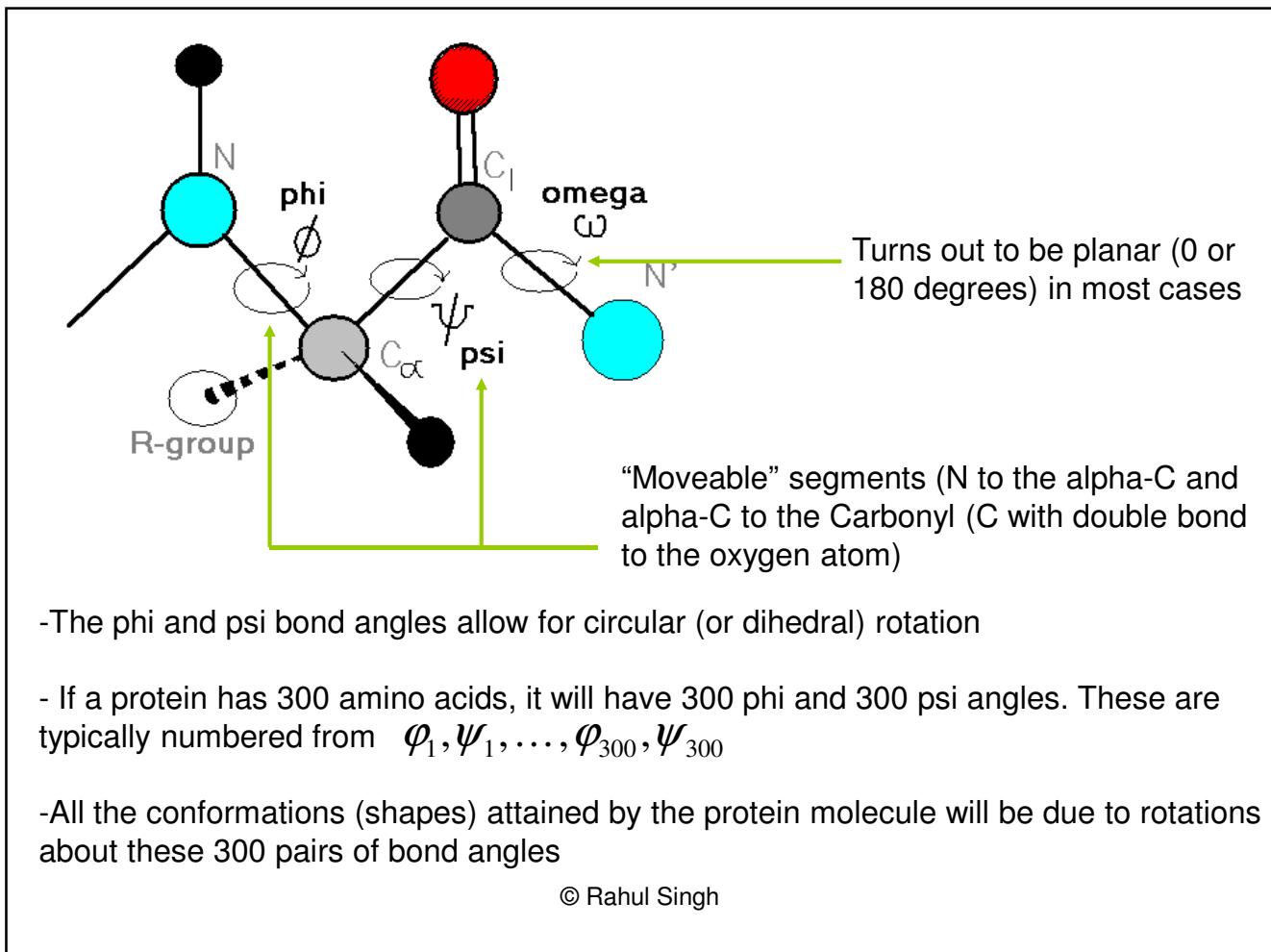
© Rahul Singh

Consider a polypeptide chain:



We are interested in three specific torsional angles





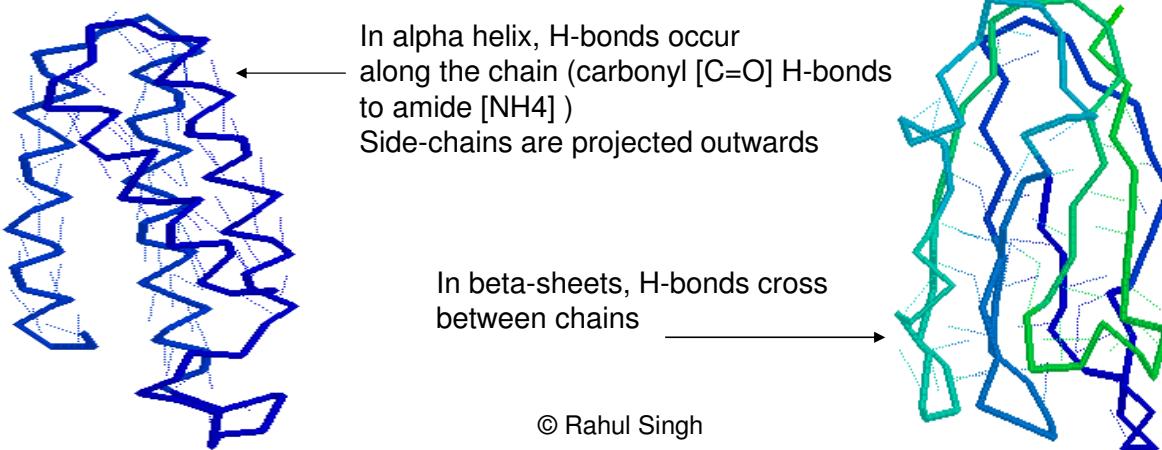
α - helix and β - strands

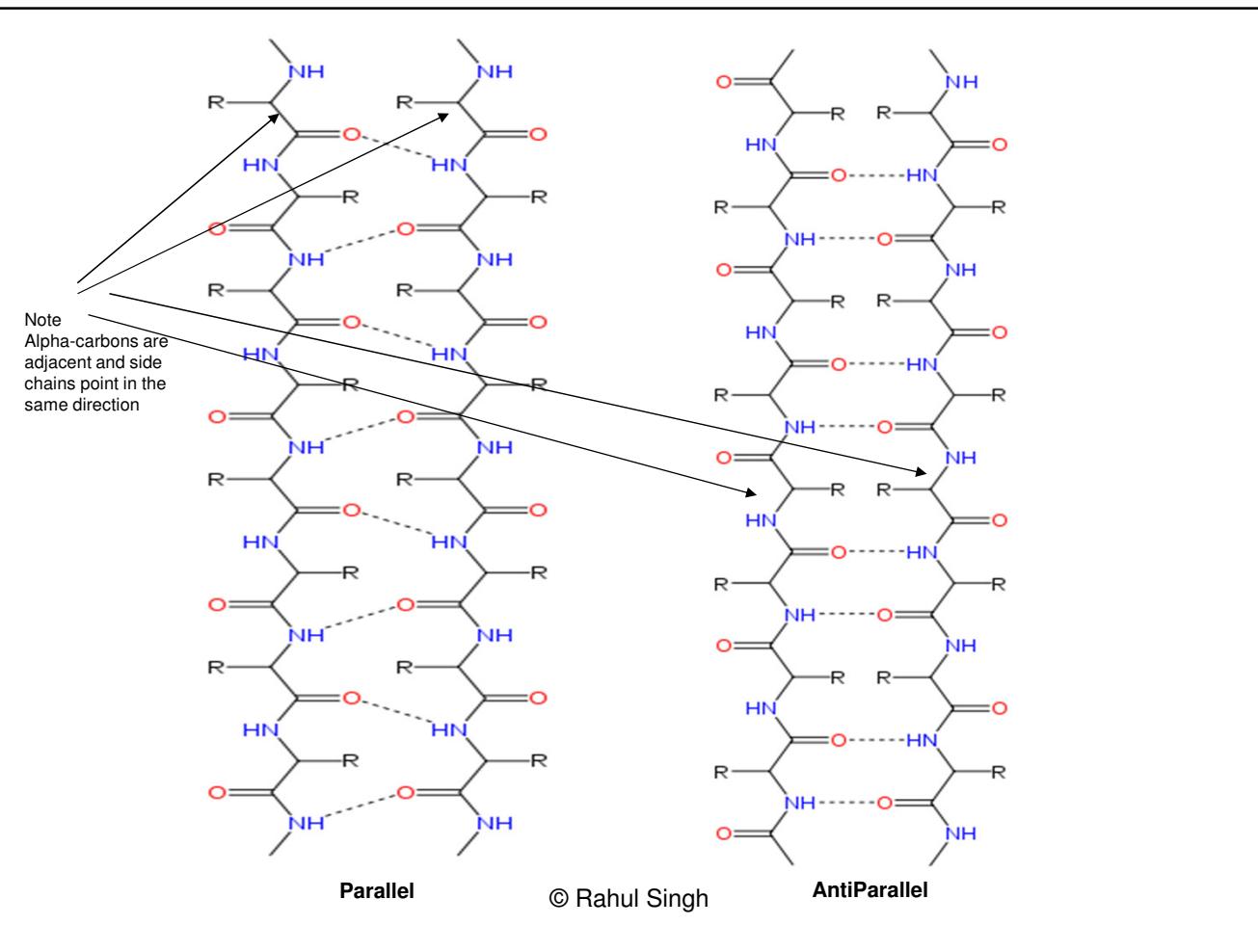
The α - helix : is characterized by phi and psi angles of roughly -60 degrees

The β - strands: are characterized by regions of extended, nearly linear, backbone conformations with phi roughly equal to -135 degrees and psi roughly equal to 135 degrees

The β - strands assemble into one of two types of β - sheets:

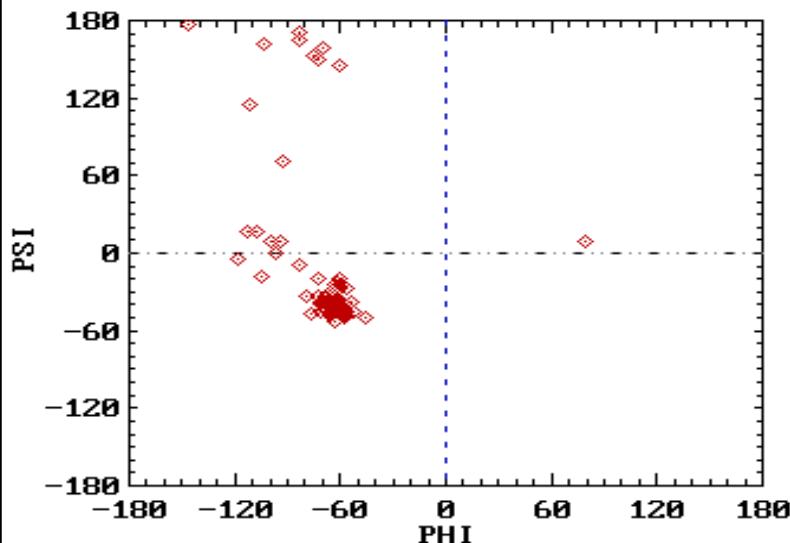
Parallel sheets (strands run in the same direction) and anti-parallel sheets, where adjacent strands run in opposite directions





The Ramachandran Plot

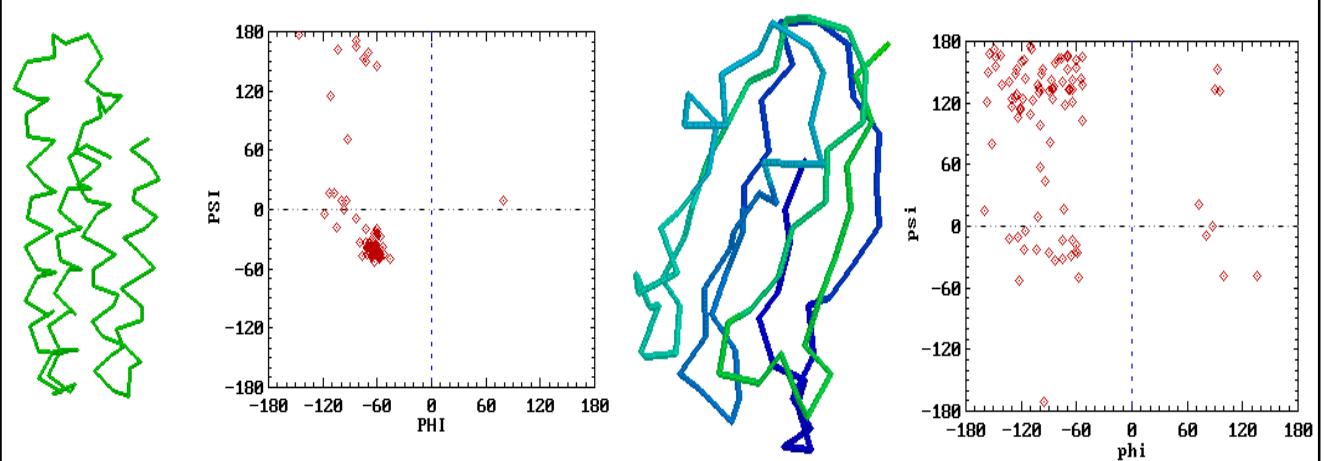
- Not all combinations of phi and psi angles are realizable; for example certain values for phi and psi may require different atoms to occupy the same space – which is physically impossible
- A plot of φ (phi) vs ψ (psi) is called the Ramachandran plot



For example, this plot shows repeating values of phi=-57 degrees and psi = -47 degrees respectively.

For example, the repeating values of phi and psi in this case give a right-handed helical fold (the alpha helix)

© Rahul Singh



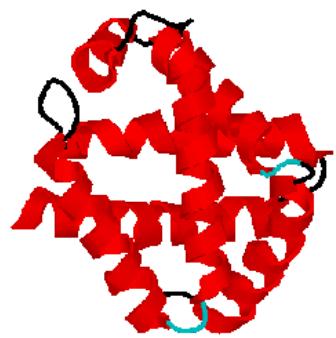
Indeed, the Cytochrome C-256 structure, from which this data is derived shows the presence of such helical structures (top left)

Similarly repetitive values in the region $\phi = -110$ to -140 degrees and $\psi = +110$ degrees give extended chains with conformations that allow interactions between closely folded parallel segments (beta sheet structures). Top right is the structure of Plastocyanin and its Ramachandran plot, which shows a concentration of angles around $\phi = -135$ degrees and $\psi = 135$ degrees thus indicating the presence of beta sheet-like structures which are observed in the molecule

© Rahul Singh

The secondary structure of a protein combined with less structured regions forms an overall 3D shape of the protein, which is called the tertiary structure

When protein chains come together in a single large complex (e.g. active enzymes consisting of two or more proteins), the overall structure formed by interacting proteins is called the quaternary structure



Marine bloodworm Hemoglobin
(mostly alpha-helix)



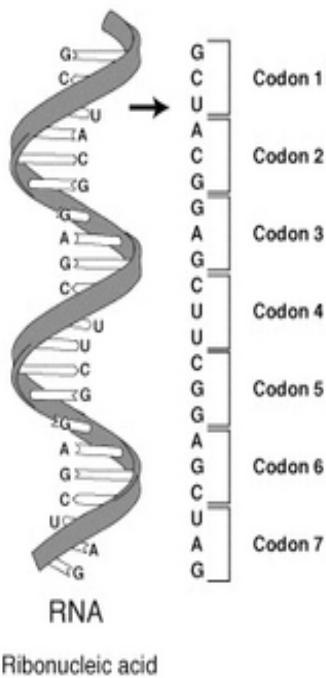
Sea-snake neurotoxin
(mostly beta-sheets)



Ribonuclease T1
(mix of alpha helices and beta-sheets)

© Rahul Singh

Returning to the Sequence: the Genetic Code



-Both DNA and RNA are composed of 4 nucleotide bases: DNA (A, G, C, T) and RNA (A, G, C, U)

-The gene sequence in DNA (and RNA) is composed of units called codons. Each codon corresponds to a single amino acid

-Codons are non-overlapping groups of 3 bases (see example on left)

- Example: RNA sequence: UUUAAACCC

codon: UUU codon: AAA codon: CCC

Since, each codon corresponds to one amino-acid, this sequence represents a protein sequence that is three amino-acid long

- There are $4^3 = 64$ possible codons

© Rahul Singh

Technical Details

Codon Tables

Two types of codon tables are typically employed. The forward codon table shows each of the 64 codons and the amino acids they correspond to. The reverse codon table shows each of the amino acids and the codons that code for them.

Stop codons

There are three codons that do not instruct the ribosomes to insert a specific amino acid. They are called stop codons (UAA, UGA, UAG), because they cause translation to be terminated

Reading Frame

A codon is completely defined by the start position. For example: GGGAAACCC (Read from start position): GGG, AAA, CCC (Read from second): GGA, AAC – ignoring partial codons. (Read from third): GAA, ACC.

The actual frame a protein sequence is translated in, is defined by a start codon.

Start Codons

In RNA sequences, this is typically AUG (also codes for Methionine)

© Rahul Singh

Amino Acid	SLC	DNA Codons
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Valine	V	GUU, GUC, GUA, GUG
Phenylalanine	F	UUU, UUC
Methionine	M	AUG
Cysteine	C	UGC, UGC
Alanine	A	GCU, GCC, GCA, GCG
Glycine	G	GGU, GGC, GGA, GGG
Proline	P	CCU, CCC, CCA, CCG
Threonine	T	ACU, ACC, ACA, ACG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Tryptophan	W	UGG
Tyrosine	Y	UAU, UAC
Glutamine	Q	CAA, CAG
Asparagine	N	AAU, AAC
Histidine	H	CAU, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAU, GAC
Lysine	K	AAA, AAG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Stop codons	Sto p	UAA, UAC, UGA

© Rahul Singh

Ala	A	GCU, GCC, GCA, GCG	Leu	L	UUA, UUG, CUU, CUC, CUA, CUG
Arg	R	CGU, CGC, CGA, CGG, AGA, AGG	Lys	K	AAA, AAG
Asn	N	AAU, AAC	Met	M	AUG
Asp	D	GAU, GAC	Phe	F	UUU, UUC
Cys	C	UGU, UGC	Pro	P	CCU, CCC, CCA, CCG
Gln	Q	CAA, CAG	Ser	S	UCU, UCC, UCA, UCG, AGU, AGC
Glu	E	GAA, GAG	Thr	T	ACU, ACC, ACA, ACG
Gly	G	GGU, GGC, GGA, GGG	Trp	W	UGG
His	H	CAU, CAC	Tyr	Y	UAU, UAC
Ile	I	AUU, AUC, AUA	Val	V	GUU, GUC, GUA, GUG
Start		AUG, GUG	Stop		UAG, UGA, UAA

Reverse codon table

© Rahul Singh

Degeneracy of the Genetic Code

- Most codons are degenerate: that is two or more codons code for the same amino acid
- Typically, degenerate codons differ in their third position:

glutamine → CAA
glutamine → CAG

-A codon is four fold degenerate, if any nucleotide at its third position specifies the same amino acid. Find an example in the forward codon table

-Similarly, it is two-fold degenerate if only two of the four possible nucleotides in the third position, specify the same amino acid

Glutamic Acid ([GAA, GAG]), Aspartic Acid ([GAU, GAC]), Glutamine ([CAA, CAG]), Histidine ([CAU, CAC]), ... Notice any pattern?

-These properties make the genetic code fault tolerant to mutations. For example, a four-fold degenerate codon can tolerate any mutation at the third position

© Rahul Singh

BLAST

- BLAST = A Basic Local Alignment Search Tool
- Published in Journal of Molecular Biology, 1990 (Altschul, Gish, Miller, Myers, and Lipman)
- Original algorithm was designed to search a sequence database for maximal ungapped local alignments
- Many variants now exist:

blastp	Amino-acid query against protein sequence database (Protein – Protein)
blastn	Nucleotide query against nucleotide sequence database (Nucleotide – Nucleotide)
blastx	nucleotide query translated in all reading frames against a protein sequence database (Nucleotide – Protein)
tblastn	protein query against a nucleotide sequence database that is dynamically translated in all reading frames (Protein – Nucleotide)
tblastx	6-frame translations of a nucleotide query sequence against 6-frame translations of a nucleotide sequence database (Nucleotide – Nucleotide)

© Rahul Singh

How BLAST works

- Blast is a local alignment method.
- It is capable of detecting the best region of local alignment between a query sequence and the target.
- It can also find other plausible alignments between the query and the target

- Consider the following example:

Query sequence: PFIETAERL RDQHKKD YPEYKY QP RRRK

- The search involves three main steps:

Step 1: The query sequence is broken into fixed length sub-sequences called words (default length(s) = 3 or 4) by sliding a window of size equal to the word length



© Rahul Singh

Step 1 (contd.) These subset of letters are then used to “seed” the sequence search. That is, BLAST tries to find sequences in the target database that contain either the seed or subsequences where conservative substitutions may have occurred

Why should we be interested in conservative substitutions?

Step 2: Search for word matches in the database sequences. The matches can be found, for example, by using the BLOSUM matrices

PFIETAERL**RDQHKKDYPEYKYQPRRRK**



RDQ 16	QDQ 12	• • •
RBQ 14	REQ 12	
RDZ 12	RDR 12	
KDQ 13	•	
RDE 13	•	

RDP 10
RDT 10
RDY 10
RDX 10
DDQ 9
• • •

© Rahul Singh

RDQ 16	QDQ 12	• • •	RDP 10
RBQ 14	REQ 12		RDT 10
RDZ 14	RDR 12		RDY 10
KDQ 13	•		RDX 10
RDE 13	•		DDQ 9



T = 12

{ (RDQ}, (RBQ), (RDZ), (KDQ), (RDE) }

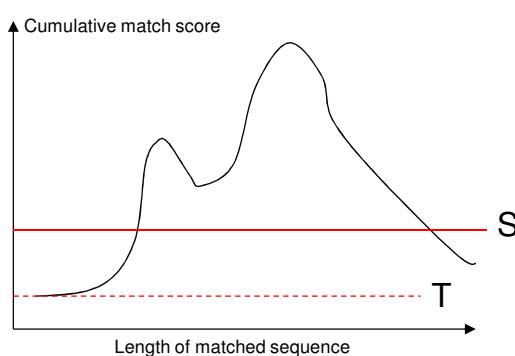
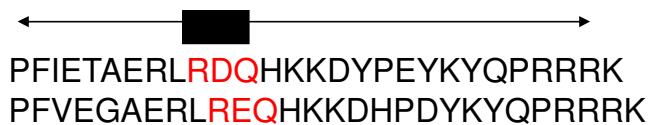
Step 3: Identify the words that are closely related to the original query sequence. This is done by using a threshold (T) on the similarity scores obtained from the previous step. The threshold is automatically determined by BLAST, but can be set manually.

Note: By increasing the threshold, we get more exact and fewer matches

By decreasing the threshold, we can detect more distant relationships between sequences

© Rahul Singh

Step 4: “Extend” the match in both directions from the matching word, till the alignment score falls below a threshold

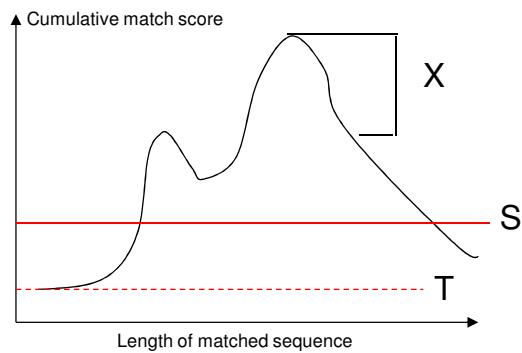


Consider the number of aligned residues and the cumulative match score they generate:

- In the beginning, the match score is greater than or equal to T , for the initial seed word match
- As the match proceeds, scores from matches and conservative substitutions add to the score while mismatches and gaps reduce it
- Once the cumulative score exceeds a threshold S , the alignment is reported in the BLAST output

- As the match extension continues, at some point, the cumulative score starts to decrease as the mismatches and gap penalties rise

© Rahul Singh



- As the match extension continues, at some point, the cumulative score starts to decrease as the mismatches and gap penalties rise
- As the cumulative score starts to decrease, BLAST measures the dropoff and compares it with a threshold X
- If the dropoff is greater than X, the extension is trimmed back to the length corresponding to the preceding maximum value from where the latest decrease in the cumulative score started
- The resulting alignment is called a high-scoring pair (HSP)

© Rahul Singh

Does a BLAST Alignment Always Mean Something?

-A variety of factors may influence the raw cumulative match score we get from BLAST. For example, the score would depend on the scoring matrix used, the length of the query and target sequences, the actual composition of these sequences etc.

-Therefore, once an HSP is obtained, it is important to determine if the resulting alignment is actually significant

-The Karlin-Altschul equation is used for this: This equation uses search-specific parameters to calculate an expectation value E. The value of E represents the number of high-scoring segment pairs that may be expected, purely by chance.

("Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes", PNAS, 87, 2264-2268)

$$E = k \times m \times N \times e^{-\lambda s}$$

Constant ↑
Number of letters in the query ↑
Total number of letters in the target database ↑

Score of the HSP ↑
Constant with value depending on the
scoring matrix
Used for normalizing the score of the HSP

© Rahul Singh

Heuristics for BLAST cutoffs

-BTW, what is better – a lower value of E or a higher value of E? Why?

- Heuristics-based guidelines are often proposed for establishing meaningful hits from non-relevant ones. For example,

- For nucleotide-based search, E should be no more than 10^{-6} and sequence similarity should be 70% or more

- For protein-based searches, E should be no more than 10^{-3} and sequence similarity should be 25% or more

- NEVER USE THE CUTOFFS BLINDLY, ESPECIALLY WHEN CLOSE TO THE THRESHOLD VALUES

- MAKE SURE TO USE THE CORRECT SCORING MATRIX

- MANUALLY INVESTIGATE THE ALIGNMENT

- INVESTIGATE THE BIOLOGY

© Rahul Singh

Related Algorithms

- FASTA: Perform gapped local alignment between sequences
- MEGABLAST: Variation of the BLASTN algorithm used for aligning either long or highly similar (> 95%) sequences
- PSI-BLAST: (Position Specific Iterated Blast) – used for identifying distantly related proteins. It uses position specific scoring matrices (hidden markov models)

© Rahul Singh

More Information

-An excellent set of on-line resources on sequence comparisons is supported by the NCBI (National Center for Biotechnology Information)

<http://www.ncbi.nlm.nih.gov/>

-Tutorials as well as a science primer are available at:

<http://www.ncbi.nlm.nih.gov/Education/index.html>

-Tutorials on using BLAST (and its various versions) can be found at:

<http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html>

Specifically, try the:

Query Tutorial: http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/query_tutorial.html

BLAST Tutorial: <http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/tut1.html>

© Rahul Singh

Percent Sequence Identity

Similarity between a pair of aligned biological sequences is represented as sequence identity: the number of aligned positions where the matching bases or amino acids (for proteins) are identical.

Since all sequences are not of the same length, sequence identity is corrected for length and expressed as percent identity (PID).

Four types of denominators are typically used:

1. Length of the shorter sequence (L1)
2. Alignment length including gaps, i.e., the number of aligned positions (L2)
3. Number of aligned residue pairs, i.e., identities and non-identities, excluding gaps (L3)
4. Mean sequence length (L4)

Note: PID is operationally defined. However you should be careful to indicate the choice of the denominator

© Rahul Singh

The behavior of these four denominators was studied in:

Percent sequence identity: the need to be explicit, A. May, *Structure* 12, 737-38, 2004

In this paper, the HOMSTRAD database of protein structure alignment for homologous families was used. (total 1032 families, Dec 2003 release)

For families consisting of more than 2 structures, all possible pair-wise sequence alignments were considered (as defined by the HOMSTRAD 3D structure-based multiple-sequence alignment. Matching gaps were ignored for a pair).

Linear correlation was used to describe the relationship between the four denominators for all the sequence pairs (N=9539). The table is reproduced below

	L1	L2	L3	L4
L1	1	0.978	0.99	0.996
L2	0.978	1	0.9494	0.99
L3	0.99	0.949	1	0.984
L4	0.996	0.99	0.984	1

- L4 is the most representative denominator
- Number of gaps is important in its own right

© Rahul Singh

Multiple Sequence Alignment

© Rahul Singh

Multiple Sequence Alignment

Consider the following sequences:

Sequence-1: M Q P I L L L M

Sequence-2: M L R L L

Sequence-3: M K I L L L

Sequence-4: M P P V L I L M

And an (arbitrary) cost model: For example, Match = 1, Mismatch = -1, Gap = -2

Definition: *Multiple sequence alignment (MSA): is obtained by inserting gaps in the strings so as to make them all of the same length, such that no column in the alignment is all gaps.*

Note: Intuitively, the gaps are inserted to optimize the match, given the cost model

For example:

M	Q	P	I	L	L	L	M
M	L	R	-	L	L	-	-
M	K	-	I	L	L	L	-
M	P	P	V	L	I	L	M

Note: No column is all gaps, but for any two sequences, there may be a column where both have gaps

© Rahul Singh

Why MSA?

1. Detect regions of similarity (conservation) or variance (variability) within family of proteins and/or across families of proteins
2. First step in Phylogenetic reconstruction
3. Used in secondary structure prediction
4. Used for building probabilistic models (profiles) of protein families

© Rahul Singh

Scoring an MSA

-Tricky!

Sum-of-pairs scheme (SP): Total score = sum of the scores of all-pair (unique pairs) matches in each column

M	Q	P	I	L	L	L	M
M	L	R	-	L	L	-	-
M	K	-	I	L	L	L	-
M	P	P	V	L	I	L	M

↓ ←

$$\text{score}[(I, -), (I, I), (I, V), (-, I), (-, V), (I, V)] = -2 + 1 -1 -2 -2 -1 = -7$$

$$\text{score}[(M, -), (M, -), (M, M), (-, -), (-, M), (-, M)] = \begin{matrix} & \\ & \leftarrow \end{matrix} \\ -2 -2 +1 +0 -2 -2 = -7$$

(Note: $(-, -)$ is scored as 0, to eliminate extra penalization for the use of gaps

© Rahul Singh

Optimum MSA and Optimum Pair-Wise Alignment

1: A T	A T	Score (Column 1) = Score[(A, A), (A, -), (A, A), (A, A),
2: A	A -	(A, -), (A, A), (A, A),
3: T	- T	(-, A), (-, A)
4: A T	A T	(A, A)] = -2
5: A T	A T	Score (Column 2) = -2

However, look at sequences 2 and 3. We have from MSA:

A -
- T

But the optimal pair-wise alignment would be: A
T

Why?

© Rahul Singh

See Any Problems with SP-scheme?

© Rahul Singh

Computing Optimal MSA

Let's remember the logic behind optimal pair-wise sequence alignment:

$i =$	$j =$	0	1	2	3	4	5
	y_j	A	C	T	C	G	
0	x_i	0	-1	-2	-3	-4	-5
1	A	-1					
2	C	-2					
3	A	-3					
4	G	-4					
5	T	-5					
6	A	-6					
7	G	-7					

- Given sequences of lengths n and m , We would construct a table of size $n+1 \times m+1$ and fill it using dynamic programming
- This is significantly better than enumerating all subsequences and matching them since such an naive approach has exponential complexity
- So can this approach work for MSA also?

© Rahul Singh

Assume, we have n sequences of length k each

To align 2 sequences we need to solve a dynamic programming problem with table size: $[k+1] \times [k+1]$

To align 3 sequences: $[k+1] \times [k+1] \times [k+1]$

To align 4: $[k+1] \times [k+1] \times [k+1] \times [k+1]$

...

To align n : $[k+1] \times [k+1] \times \dots \times [k+1] \times [k+1] = [k+1]^n$

Thus, the size of the table increases exponentially with the number of sequences

This is not an efficient (or realistic approach)

-Most approaches to MSA, do not compute optimal alignment. Rather they use heuristics to get “good” alignments

© Rahul Singh

Progressive Alignments

Intuition: Compute pair-wise alignments and merge these alignments consistently.

Observation-1: Given all pair-wise alignments, it is generally not possible to come up with a MSA consistent with all of them.

Example: (1) A C T, (2) C T A, (3) T A C

[1-2]: A C T -
- C T A

[1-3]: - A C T
T A C -

[2-3]: C T A -
- T A C

Merge by aligning with (1)

- A C T -
-- C T A
T A C - -

← MSA is not consistent with
[2-3]

Merge by aligning with (2)

A C T - -
- C T A -
- - T A C

← MSA is not consistent with
[1-3]

Merge by aligning with (3)

- - A C T
- T A C -
C T A - -

← MSA is not consistent with
[1-2]

© Rahul Singh

[1-2]: ACT-
- CTA

[1-3]: - ACT
TAC -

[2-3]: CTA-
- TAC

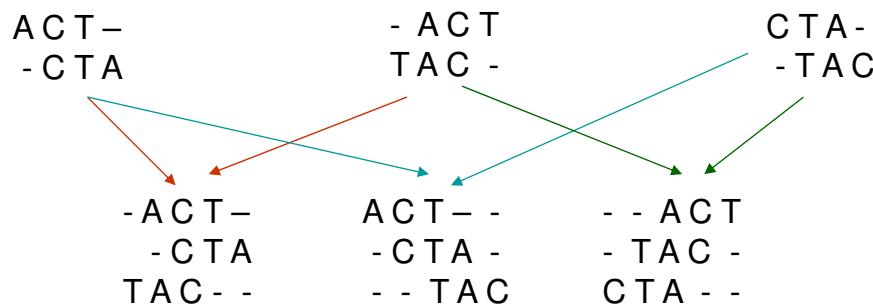
- ACT-
- CTA
TAC - -

ACT- -
- CTA -
- - TAC

- - ACT
- TAC -
CTA - -

Observation-2: However!

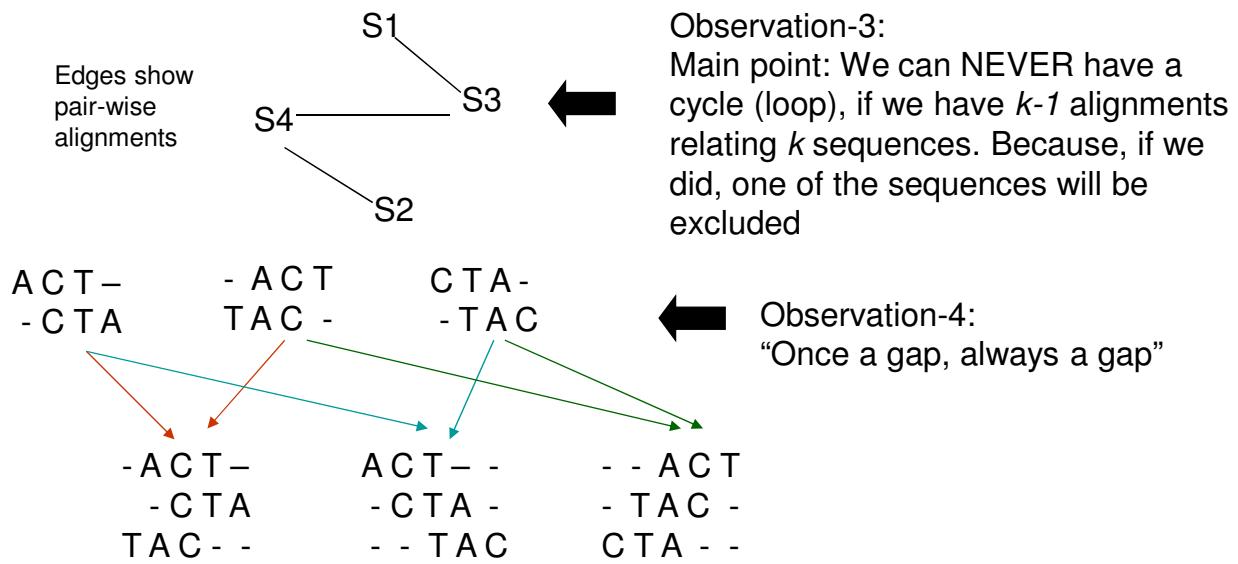
Given any $k-1$ alignments relating k sequences, there is always a consistent MSA – that is, the pair-wise alignments in the MSA include the given $k-1$ alignments



© Rahul Singh

Given any $k-1$ alignments relating k sequences, there is always a consistent MSA – that is, the pair-wise alignments in the MSA include the given $k-1$ alignments

For example, if $K = 4$, then we have 3 alignments relating all sequences



So, we can get a MSA by adding sequences to an alignment, ensuring there are no cycles and following the "Once a gap, always a gap" rule

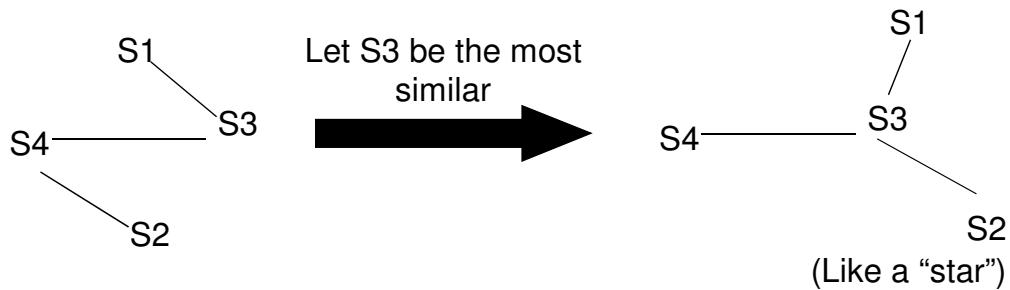
© Rahul Singh

Star-Alignment Approach

1. Find a sequence, most similar to all by using pair-wise alignment. That is find a sequence S_c , such that:

$$\text{maximize } \sum_{i \neq c} \text{similarity}(S_i, S_c)$$

2. Use pair-wise alignment with this sequence only, when building the MSA



© Rahul Singh

Example

S1: ATTGCCATT
S2: ATGGCCATT
S3: ATCCAATTT
S4: ATCTTCTT
S5: ACTGACC



	S1	S2	S3	S4	S5
S1	-	7	-2	0	-3
S2	7	-	-2	0	-4
S3	-2	-2	-	0	-7
S4	0	0	0	-	-3
S5	-3	-4	-7	-3	-

S1 is most similar to the rest. The alignments relative to S1 are

S1: ATTGCCATT
S2: ATGGCCATT

S1: ATTGCCATT - -
S3: ATC - C AATTTT

S1: ATTGCCATT
S4: ATCTTCTT

S1: ATTGCCATT
S5: ACTGACC - -

Let's build the MSA, by progressively aligning sequences to S1, following the "once a gap, always a gap" rule

© Rahul Singh

S1: ATTGCCATT
S2: ATGGCCATT

S1: ATTGCCATT - -
S2: ATGGCCATT - -
S3: ATC - C AATTTT

S1: ATTGCCATT - -
S2: ATGGCCATT - -
S3: ATC - C AATTTT
S4: ATCTTC - TT - -

S1: ATTGCCATT - -
S2: ATGGCCATT - -
S3: ATC - C AATTTT
S4: ATCTTC - TT - -
S5: ACTGACC - - - -

Complexity:

1. $k \times (k-1)/2$ pair-wise alignments
 2. Each alignment takes n^2 time
- So: $O(k^2 \times n^2)$ time

3. If the largest sequence is L long, then we can merge k^2 alignments in $k^2 \times L$ time
4. Overall: $O(K^2 \times n^2 + K^2 \times L)$

Most Important

The star alignment technique does not optimize the SP criterion. There is a trade-off between optimization and efficient computability

© Rahul Singh

ClustalW

<http://www.ebi.ac.uk/clustalw/#>

The screenshot shows the ClustalW Submission Form on the European Bioinformatics Institute (EBI) website. The form is titled "ClustalW Submission Form". It includes a brief description of the program, download links for Mac OS X, Windows, and Linux, and several configuration tables. The configuration tables include fields for "YOUR EMAIL", "ALIGNMENT TITLE", "RESULTS", "ALIGNMENT", "CPU MODE", "WINDOW LENGTH", "SCORE TYPE", "TOPDIAG", "PAIRGAP", "MATRIX", "GAP OPEN", "END GAPS", "GAP EXTENSION", "GAP DISTANCES", "OUTPUT FORMAT", "OUTPUT ORDER", "TREE TYPE", "CORRECT DIST.", and "IGNORE GAPS". At the bottom, there is a text area for pasting sequences and a "Help" button.

© Rahul Singh

General Algorithm of ClustalW

Step-1: Determine the pair-wise alignment between the participating sequences through global sequence alignment.

Step-2: Compute a distance between each pair of sequences: For example, for the non-gapped positions, compute the ratio of the sites having differences to the total number of sites.

$$\begin{array}{cccc} \text{K} & \text{Q} & - & \text{K} \\ & & - & \text{L} \\ & & - & \text{M} \\ & & - & \text{N} \\ & & \text{V} & \end{array}$$

$\frac{1}{4} = 0.25$

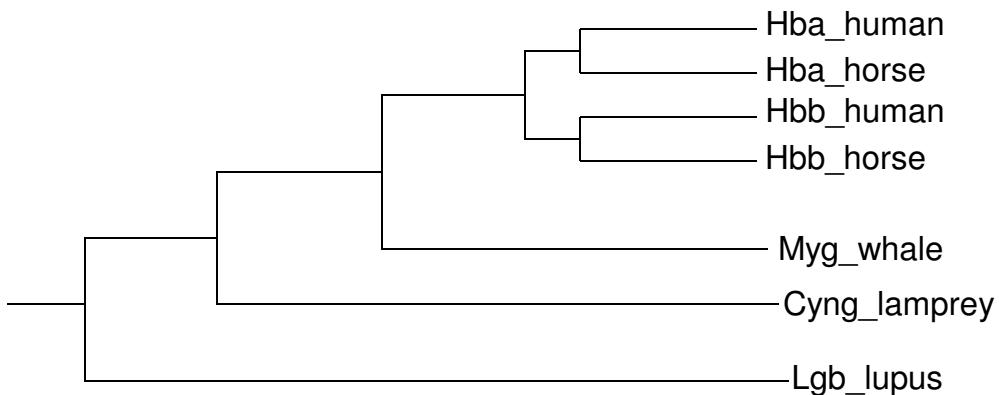
Basically, this is a measure of the number of observed substitutions – smaller the distance, more similar the sequences.

Step-3: Put the distances in a matrix (Matrix on next page for Globin protein data)

© Rahul Singh

Globin Type		1	2	3	4	5	6	7
Hbb_human	1	-						
Hbb_horse	2	0.17	-					
Hba_human	3	0.59	0.60	-				
Hba_horse	4	0.59	0.59	0.13	-			
Myg_whale	5	0.77	0.77	0.75	0.75	-		
Cyng_lamprey	6	0.81	0.82	0.73	0.74	0.80	-	
Lgb_lupus	7	0.87	0.86	0.86	0.88	0.93	0.90	-

Step 4: Construct a phylogenetic-tree using the neighbor-joining method.



© Rahul Singh

Step 5: Combine the alignments. Start with the most closely related group and progressively move to the most distantly related group. Remember, “*once a gap, always a gap*”.

For this example, align Hba_human and Hba_horse. Then align Hbb_human and Hbb_horse. Next, combine these alignments and so on ...

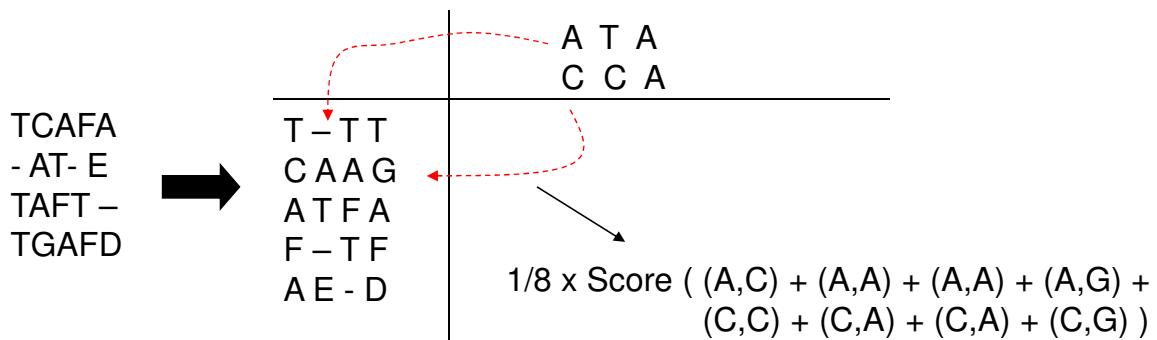
Few Important Points

1. Each pair of sequences is aligned with the Needleman-Wunsch algorithm using an affine gap penalty.
2. We also align *alignments* with other *alignments*. This is done using dynamic programming. However, the score in each cell of the matrix, uses the average of all pair-wise scores, from the two sets of sequences involved in the alignment.

For example: Alignment-1: ATA Alignment-2: TCAFA
 CCA - AT- E
 TAFT –
 TGAFD

© Rahul Singh

- To align alignments, we need to score and align columns against columns



Many Caveats Exist

1. No relation to optimal MSA with respect to sum-of-pairs measure
2. Possible to have more than one optimal pair-wise alignment. However, this is not considered.
3. The order in which sequences are added makes a difference
4. If pair-wise similarity is less than 25%, we can get bad overall alignments
5. Generally, one needs to correct some alignments manually.

© Rahul Singh

CSC-857 Bioinformatics Computing

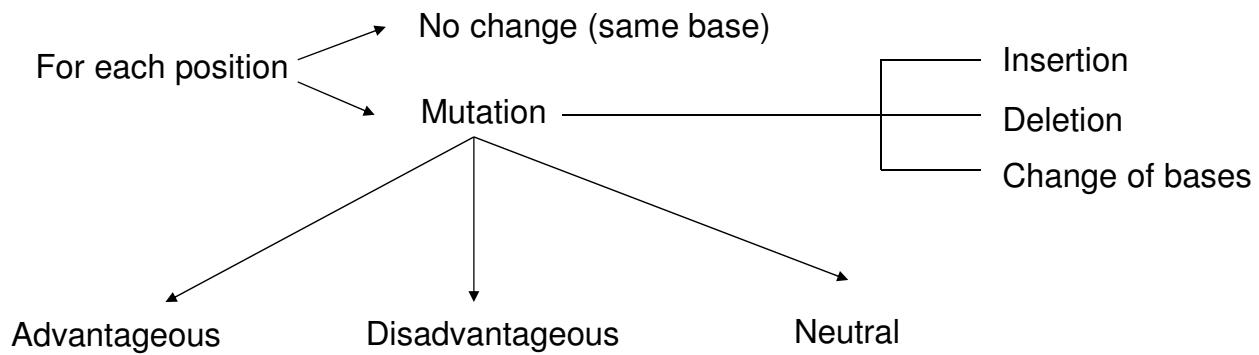
Lecture: Substitution Patterns and Substitution Matrices

© Rahul Singh

Substitution in Genes

-Consider two gene sequences in an alignment:

Sequence 1 ACTGGCATTGGGCCATAAA ...
Sequence 2 ATC – AAATGCGT - TATACC ...



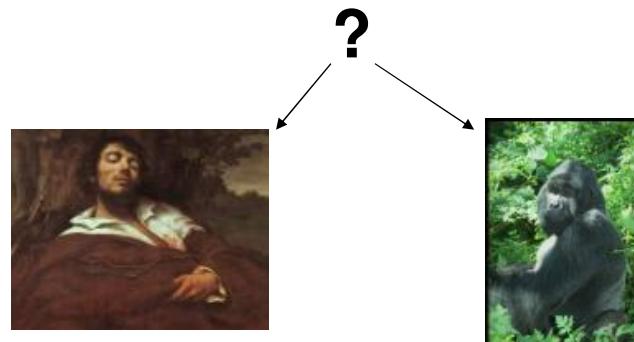
-Do we know the ratio of changes for each category? - No

-Advantageous changes are **few** - Why?

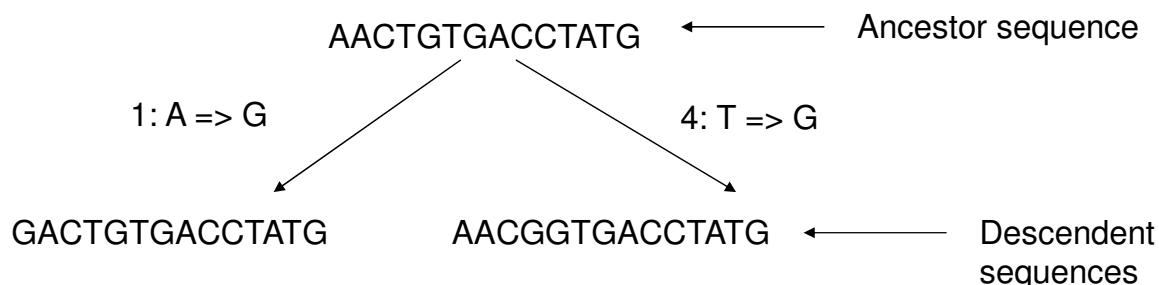
-Some changes in the sequence are more important than others

© Rahul Singh

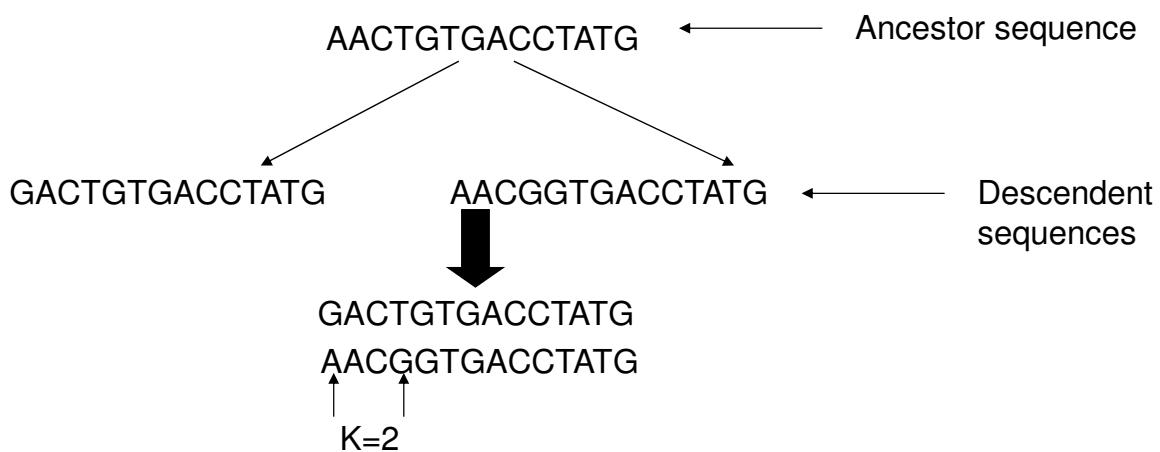
Number of Mutations and Mutation Rates



-How can we count the number of mutations (K), undergone by two gene sequences since they shared a common ancestor?



© Rahul Singh



From K, we can get the rate of substitution by considering the time over which the changes happened:

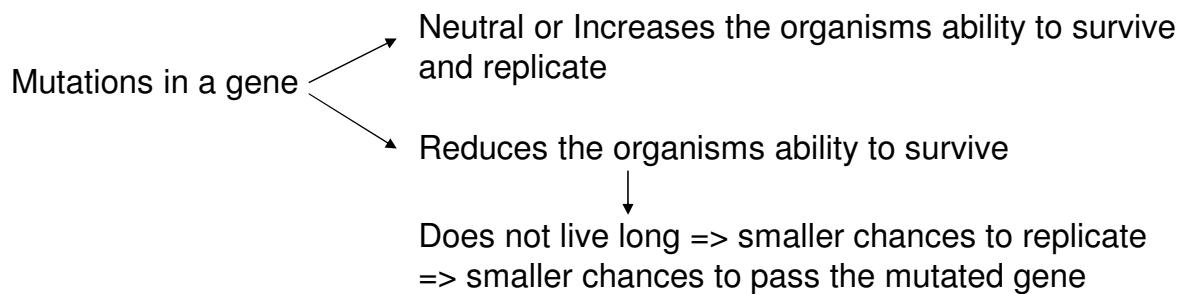
$$r = K / (2T)$$

-Note, we have $2T$ because we are assuming that the substitutions happened independently and simultaneously in both sequences

- This is a good but simplistic model (can you point out some of the reasons as to why?)

© Rahul Singh

Functional Constraints on Gene Substitution

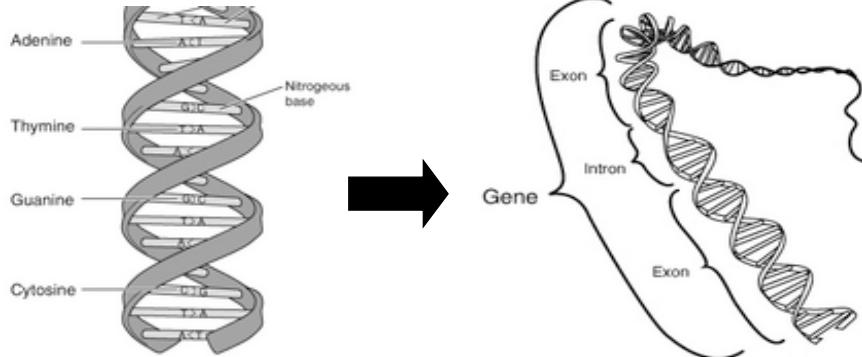


-This process, by which (the mutated) genes that diminish an organisms ability to survive, are removed from the gene pool is called natural selection

-We know by now, two important facts:

1. Genes are responsible for synthesizing proteins. And proteins are responsible for most important functions necessary for sustaining life
2. (As far as we know) some regions of a gene are more important than others

© Rahul Singh



- Indeed, it has been found that different portions of a gene accumulate changes at different rates
- Portions of a gene that are especially important, accumulate changes much more slowly – they are said to be under functional constraint

Consider the average pair-wise divergence between different regions of human, mouse rabbit and cow beta-like globin genes

<u>Region</u>	<u>Substitution Rate</u> (substitution/site/ 10^9 years)
---------------	---

Non-coding overall	3.33
Coding, overall	1.58
Intron-1	3.48
5' untranslated	1.86

© Rahul Singh

<u>Region</u>	<u>Substitution Rate</u> (substitution/site/ 10^9 years)	5' flanking	exon	3' flanking
Non-coding overall	3.33			
Coding, overall	1.58			
Intron-1	3.48			
5' untranslated	1.86			
5' flanking	3.39			
3' untranslated	3.00			
3' flanking	3.60			

Observations

- About two-times higher rate of change in the overall non-coding regions than in the overall coding regions
- The non-coding regions can be subdivided into different categories – introns, 5' and 3' flanking regions (not transcribed) and leader and trailer regions that are transcribed but not translated. *Each of these also change at different rates*
- The 5' untranslated sequence changes at a rather low rate (1.86) ! – This is because the nucleotides here are responsible for the translation of beta-like globin proteins.

© Rahul Singh

Types of Substitutions

Amino Acid	SLC	DNA Codons
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Valine	V	GUU, GUC, GUA, GUG
Phenylalanine	F	UUU, UUC
Methionine	M	AUG
Cysteine	C	UGC, UGC
Alanine	A	GCU, GCC, GCA, GCG
Glycine	G	GGU, GGC, GGA, GGG
Proline	P	CCU, CCC, CCA, CCG
Threonine	T	ACU, ACC, ACA, ACG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Tyrosine	Y	UAU, UAC
Tryptophan	W	UGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAU, AAC
Histidine	H	CAU, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAU, GAC
Lysine	K	AAA, AAG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	UAA, UAG, UGA

-Of the 20 genetically coded amino acids, 18 are coded by more than one codon

Glycine = { (GGG), (GGA), (GGC), (GGU) }

-Any change in the 3rd position, still leads to a codon that encodes for Glycine. Such changes are called **synonymous substitutions**

-In contrast, a change at the 2nd position will lead to a different amino acid. For example, Alanine = (GCG). Such changes are called **non-synonymous substitutions**

-Of the 47 substitutions that have occurred in the human and rabbit beta-like globin genes, 27 are synonymous substitutions and 20 are non-synonymous

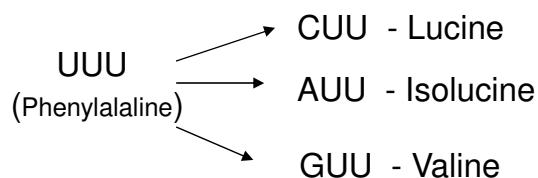
© Rahul Singh

Degeneracy

Amino Acid	SLC	DNA Codons
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Valine	V	GUU, GUC, GUA, GUG
Phenylalanine	F	UUU, UUC
Methionine	M	AUG
Cysteine	C	UGU, UGC
Alanine	A	GCU, GCC, GCA, GCG
Glycine	G	GGU, GGC, GGA, GGG
Proline	P	CCU, CCC, CCA, CCG
Threonine	T	ACU, ACC, ACA, ACG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Tyrosine	Y	UAU, UAC
Tryptophan	W	UGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAU, AAC
Histidine	H	CAU, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAU, GAC
Lysine	K	AAA, AAG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	UAA, UAG, UGA

Nucleotides in codons can be placed in one of three different categories based on whether a change leads to a synonymous or a non-synonymous substitutions:

Non-degenerate sites: Are codon positions where mutations will always result in amino acid substitutions

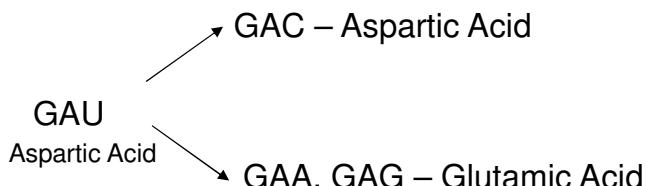


-That is, at this site, all possible changes are non-synonymous

© Rahul Singh

Amino Acid	SLC	DNA Codons
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Valine	V	GUU, GUC, GUA, GUG
Phenylalanine	F	UUU, UUC
Methionine	M	AUG
Cysteine	C	UGU, UGC
Alanine	A	GCU, GCC, GCA, GCG
Glycine	G	GGU, GGC, GGA, GGG
Proline	P	CCU, CCC, CCA, CCG
Threonine	T	ACU, ACC, ACA, ACG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Tyrosine	Y	UAU, UAC
Tryptophan	W	UGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAU, AAC
Histidine	H	CAU, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAU, GAC
Lysine	K	AAA, AAG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	UAA, UAG, UGA

Twofold degenerate sites: Are the codon positions, where two different nucleotides result in the translation of the same amino acid, while the two other nucleotides code for a different amino acid



-That is, at this site, one of three possible changes is synonymous

Amino Acid	SLC	DNA Codons
Isoleucine	I	AUU, AUC, AUA
Leucine	L	CUU, CUC, CUA, CUG, UUA, UUG
Valine	V	GUU, GUC, GUA, GUG
Phenylalanine	F	UUU, UUC
Methionine	M	AUG
Cysteine	C	UGU, UGC
Alanine	A	GCU, GCC, GCA, GCG
Glycine	G	GGU, GGC, GGA, GGG
Proline	P	CCU, CCC, CCA, CCG
Threonine	T	ACU, ACC, ACA, ACG
Serine	S	UCU, UCC, UCA, UCG, AGU, AGC
Tyrosine	Y	UAU, UAC
Tryptophan	W	UGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAU, AAC
Histidine	H	CAU, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAU, GAC
Lysine	K	AAA, AAG
Arginine	R	CGU, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	UAA, UAG, UGA

Fourfold degenerate sites: Are the codon positions, where placing any nucleotides results in the translation of the same amino acid

GGU → GGC, GGA, GGG - Glycine
Glycine

-That is, at this site, all possible changes are synonymous

© Rahul Singh

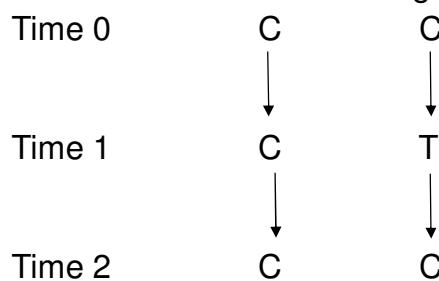
Mutations and Substitutions

- Mutations are changes in the DNA sequence (that may occur due to mistakes in the DNA replication or repair process)
- Substitutions are changes in the DNA (that is they **are mutations**) that **have passed** the filter of natural selection.
- This means that there are mutations that may have occurred, but did not survive natural selection. Note, that we can essentially observe only substitutions
- The functional constraints on a sequence can be best established by comparing the mutation and substitution rates
- Estimating substitution rates is possible. Estimating mutation rates is very difficult
- Generally, synonymous substitution rates are considered to be reflective of the actual mutation rates within a genome – because they are not subject to natural selection

© Rahul Singh

Estimating Substitution Numbers

- The number of substitutions K, between two sequences is the single most important data point in analyzing molecular evolution
- If we are looking at alignment between sequences having many differences, then simply counting the number of differences may underestimate the actual number of substitutions that may have occurred since the two genes last shared an ancestor



The observed distance between two nucleotide sequences is not equal to the actual evolutionary distance between them, except for recently diverged sequences

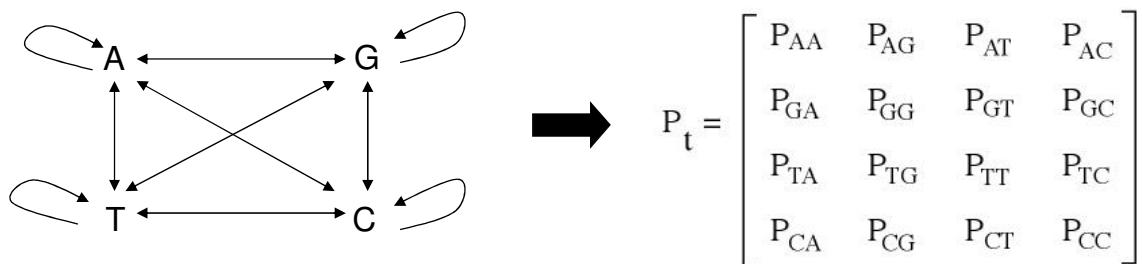
So, how to answer the following (simple sounding) question:
Given two sequences, how much evolutionary change has occurred?

- To do so, requires us to adopt a **model** of evolution

© Rahul Singh

Elements Needed For a Model

- Transition matrix



During evolution, substitution can change any base to any other. The transition matrix describes the rate of change between any pair of bases.

- Finite state discrete Markov process

- Consider a system, that can take any one of the following (finite set of) states over time: $E_1, E_2 \dots E_k$
- A Markov process is a probabilistic model of the progression of the system through these states over time

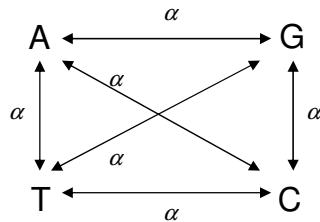
© Rahul Singh

Some important properties of the Markov chain are:

- The probability of being in state E_i at time t is designated as $p_i(t)$
- The state probability distribution at time t is given by a vector (list) of values:
 $p(t) = (p_1(t), p_2(t), \dots, p_n(t))$
Consequently, the initial distribution of states (at time $t=0$) is given as:
 $p(t) = (p_1(0), p_2(0), \dots, p_n(0))$
- The probability of transition from state i to state j is given by:
 $P_{ij} = \text{probability}(E_j \text{ at } t+1 \mid E_i \text{ at } t)$
- At any time, the probability of entering some state j from state i is 1: $\sum_j p_{ij} = 1$
- Markov chains are memory-less; P_{ij} only depends on the state of the system at time t but not the states at time $t-1, t-2, \dots$

© Rahul Singh

Jukes-Cantor Model



- The Jukes-Cantor model is the simplest discrete Markov model for describing gene substitutions
- In this model, there is only one parameter – namely the substitution rate from one base to another. Note, that this model assumes the same substitution rate for any pair of bases
- Based on this model, we can derive a formula for correcting observed distances for hidden substitutions

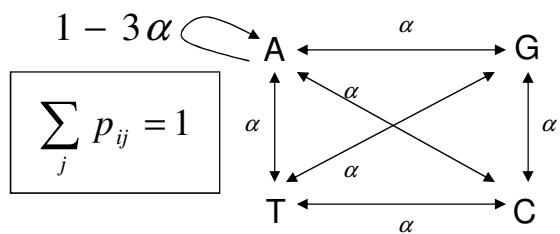
Consider the following situation:

$$\begin{array}{c} A \rightarrow A \rightarrow A \\ A \rightarrow \text{not } A \rightarrow A \\ t_1 \qquad \qquad \qquad t_2 \end{array}$$

© Rahul Singh

$$\begin{array}{c}
 A \rightarrow A \rightarrow A \\
 A \rightarrow \text{not } A \rightarrow A \\
 t_1 \qquad \qquad \qquad t_2
 \end{array}$$

-We note, that the probability that a site remains unchanged at two consecutive points in time is: $1 - 3\alpha$



- So, the probability that we see the base A at some time (say $t+2$) could be due to:
 - It was A at time $t+1$ and remained unchanged OR -- Case 1
 - It was one of {C, T, G} at time $t+1$ and mutated to A at time $t+2$ -- Case 2

The probability of case 1 happening is: $(1 - 3\alpha) P_A(t + 1)$

© Rahul Singh

The probability of case 2 happening is:

$$(1 - P_A(t+1)) \alpha$$

- That is, the probability of (not A) followed by the probability of a mutation (α)

Therefore, the overall probability of seeing an A at time t+2 will be:

$$P_A(t+2) = (P_A(t+1)(1 - 3\alpha) + (1 - P_A(t+1))\alpha)$$

-Let us now think about this issue in terms of expressing it as the change that occurs as we move from any time t to the next instance t+1

$$P_A(t+1) = P_A(t) - 3\alpha P_A(t) + \alpha - \alpha P_A(t)$$

From this, we can get:

$$P_A(t+1) - P_A(t) = -4\alpha P_A(t) + \alpha$$

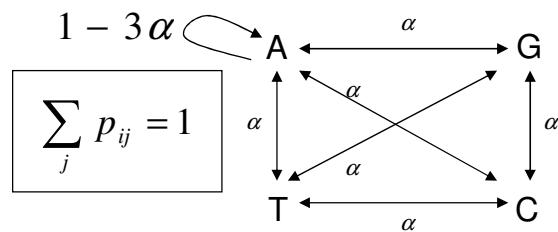
$$\frac{dP_A(t)}{dt} = \alpha(1 - 4P_A(t))$$

-This is a first order differential equation, which can be solved to get the probability of finding a particular base at a specific site

© Rahul Singh

Derivation of the Jukes-Cantor Model

Let us take another look at our basic transition model:



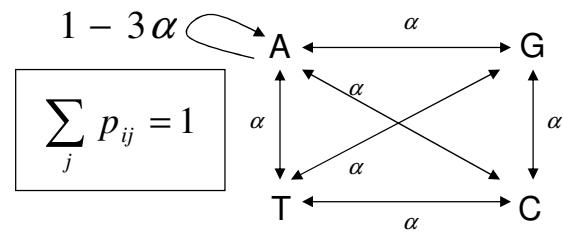
Rate of change from any base to any other base is: 3α

Further, the mean number of changes d is: Rate of change \times time = $3\alpha \times 2t = 6\alpha t$

However, we have a problem:

- Given two sequences, we can compute the fraction of sites that differ between them (let us henceforth represent this fraction by D)
- But, we do not know either α or t
- So we can not directly compute the mean number of changes

This is where our modeling helps us. Recall:



$$P_A(t+1) = (P_A(t)(1 - 3\alpha) + (1 - P_A(t))\alpha)$$

$$P_A(t+1) - P_A(t) = -4\alpha P_A(t) + \alpha$$

$$\frac{dP_A(t)}{dt} = \alpha(1 - 4P_A(t)) \quad (1)$$

$$dt = \frac{dP_A(t)}{\alpha(1 - 4P_A(t))} \Rightarrow dt = \frac{dP_A(t)}{-4\alpha P_A(t) + \alpha}$$

$$t = \int \frac{dP_A(t)}{-4\alpha P_A(t) + \alpha} \Rightarrow t = \frac{-1}{4\alpha} \ln(-4\alpha P_A(t) + \alpha) + C \quad (2)$$

Where in Eq.(2), C denotes the constant of integration

To determine the constant of integration, we have to consider the initial conditions:

If at $t = 0$, we assume that the base was an A, that is, the probability of the base being an A at time $t = 0$, $P_A(t=0) = 1$, we get from Eq. (2):

$$\begin{aligned}
 0 &= \frac{-1}{4\alpha} \ln(-4\alpha + \alpha) + C \\
 \Rightarrow C &= \frac{\ln(-3\alpha)}{4\alpha} \quad \text{Substituting this value of } C \text{ in Eq. (2), we have} \\
 t &= \frac{-\ln(-4\alpha P_A(t) + \alpha)}{4\alpha} + \frac{\ln(-3\alpha)}{4\alpha} \\
 \Rightarrow t &= \frac{-\ln(-4\alpha P_A(t) + \alpha) + \ln(-3\alpha)}{4\alpha} \quad (3) \\
 \Rightarrow -4\alpha t &= \ln(-4\alpha P_A(t) + \alpha) - \ln(-3\alpha) \\
 \Rightarrow -4\alpha t &= \ln\left(\frac{-4\alpha P_A(t) + \alpha}{-3\alpha}\right) \\
 \Rightarrow -4\alpha t &= \ln\left(\frac{-4 P_A(t) + 1}{-3}\right)
 \end{aligned}$$

Raising both sides of the equality to an exponent:

$$\begin{aligned}
 e^{-4\alpha t} &= \frac{-4P_A(t) + 1}{-3} \\
 \Rightarrow -3e^{-4\alpha t} &= -4P_A(t) + 1 \\
 \Rightarrow -3e^{-4\alpha t} - 1 &= -4P_A(t) \\
 \Rightarrow P_A(t) &= \frac{1}{4} + \frac{3}{4}e^{-4\alpha t} \quad (4)
 \end{aligned}$$

Note, that the expression in Eq. (4) is the probability of no change. Therefore, the probability of change is:

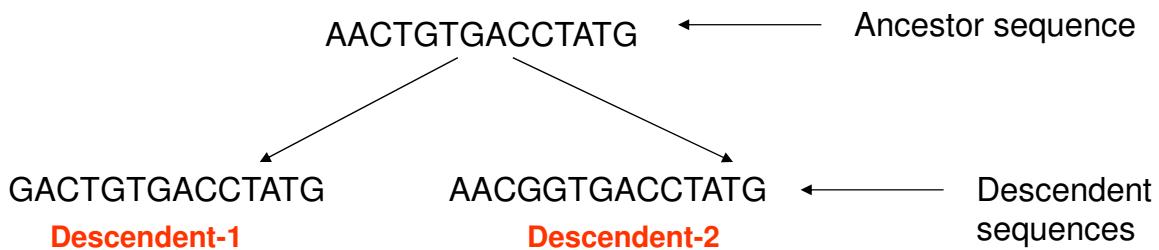
$$\begin{aligned}
 1 - P_{\text{no-change}} &= \\
 1 - P_A(t) &= 1 - \left(\frac{1}{4} + \frac{3}{4}e^{-4\alpha t}\right) = \frac{3}{4} - \frac{3}{4}e^{-4\alpha t} \quad (5)
 \end{aligned}$$

Note, that is most texts you will see:

$$P_{change} = P_{ij} = \frac{1}{4} - \frac{1}{4} e^{-4\alpha t}$$

This is a notational issue. Since, a base can change to 3 other bases, the probability of a specific change is 1/3rd that shown in Eq. (5)

Consider, now our initial motivation:



When we compare the two descendent species, which have changed independently over time t from the ancestor, the total time separating the two descendant species is $2t$. Thus, Eq. (5) becomes

$$D \equiv P_{change}(t) = \frac{3}{4} - \frac{3}{4} e^{-8\alpha t}$$

$$\Rightarrow -\frac{4}{3}(D - \frac{3}{4}) = e^{-8\alpha t}$$

$$\Rightarrow (1 - \frac{4}{3}D) = e^{-8\alpha t}$$

$$\Rightarrow \ln(1 - \frac{4}{3}D) = -8\alpha t$$

$$\Rightarrow \alpha t = \frac{-1}{8} \ln(1 - \frac{4}{3}D)$$

But we know that the mean number of changes d is: Rate of change \times time = $3\alpha \times 2t = 6\alpha t$
 From above now, we have the value for αt . Thus:

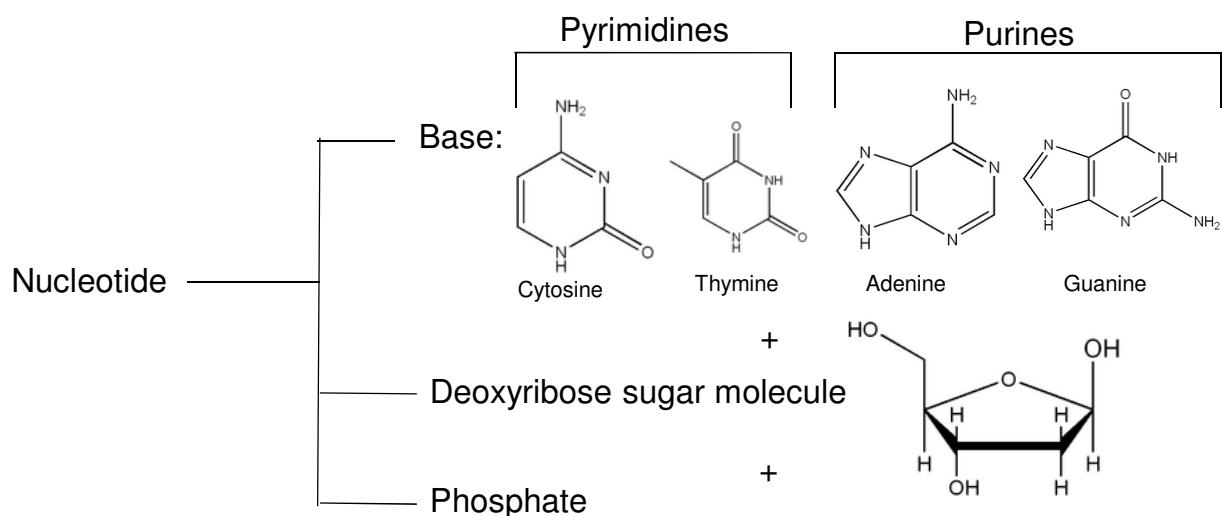
$$d = 6\alpha t = \frac{-6}{8} \ln(1 - \frac{4}{3}D) = \frac{-3}{4} \ln(1 - \frac{4}{3}D)$$

This is the Jukes-Cantor distance, thus completing our derivation

Problems with the Jukes-Cantor Model

-The fundamental assumption of the Jukes-Cantor model, that the probability (rate) of change between all nucleotides is the same, was found not to be true

Transitions and Transversions



Purines have a 2-ring structure
Pyrimidines have a single-ring structure

Therefore Purine-Purine exchanges as well as Pyrimidine-Pyrimidine exchanges can be expected to be easier than Purine-Pyrimidine exchanges

© Rahul Singh

Transitions: Are substitutions which are of the type purine-purine or pyrimidine-pyrimidine

Transversions: Are substitutions of the type purine-pyrimidine

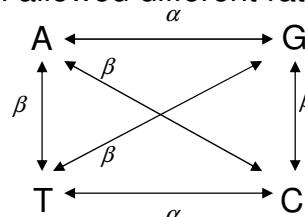
-Note that the Jukes-Cantor model was proposed before any experimental nucleotide sequence data was available

-The first sequence data sets showed that transitions are 3-times or more common than transversions

In 1980 Kimura developed a model which allowed different rates for transitions and transversions:

α : Uniform transition rate

β : Uniform transversion rate



Using this model, the formula for estimating the true number of substitutions is:

$$K = 1/2 \ln(1/(1 - 2p - q)) + 1/4 \ln(1/(1 - 2q))$$

p – fraction that are transitions, q – fractions that are transversions

© Rahul Singh

More Complex Models

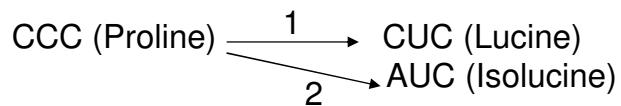
- Even Kimura's model was found to be simplistic
- Note, that 12 different types of substitutions are possible
 - G to {A, T, C}
 - A to {T, C, G}
 - C to {A, G, T}
 - T to {A, G, C}
- Each of these mutations can be assigned a probability (substitution rate)
- Further, parameters can be assigned to compensate for substitution biases in specific regions of the genome
- Such models end up making more assumptions than the simpler models and in most cases, the simple models are found to work sufficiently well

© Rahul Singh

Substitution Between Protein Sequences

Dealing with substitutions in protein sequence is complicated due to many factors:

- Most amino-acid substitutions do not have an equal effect on protein function
- There are different rates of substitutions
- The “edit distances” between amino acids does not necessarily correlate well with changes at the structural level
For example:



- Weight each amino-acid substitution differently by using empirical data
- Such weighting matrices can also be used during sequence comparison

© Rahul Singh

Substitution (Scoring) Matrices for DNA Sequences

	<i>A</i>	<i>T</i>	<i>C</i>	<i>G</i>
<i>A</i>	1	0	0	0
<i>T</i>	0	1	0	0
<i>C</i>	0	0	1	0
<i>G</i>	0	0	0	1

Identity Matrix

	<i>A</i>	<i>T</i>	<i>C</i>	<i>G</i>
<i>A</i>	5	-4	-4	-4
<i>T</i>	-4	5	-4	-4
<i>C</i>	-4	-4	5	-4
<i>G</i>	-4	-4	-4	5

BLAST Matrix

	<i>A</i>	<i>T</i>	<i>C</i>	<i>G</i>
<i>A</i>	1	-5	-5	-1
<i>T</i>	-5	1	-1	-5
<i>C</i>	-5	-1	1	-5
<i>G</i>	-1	-5	-5	1

Transition/Transversion Matrix

-Nucleotide substitution matrices are usually simple

- BLAST uses a very simple matrix
- The Transition/Transversion matrix: Small penalties for Purine-Purine (A or G) substitutions and for Pyrimidine-Pyrimidine (C or T) substitutions

© Rahul Singh

BLOSUM (BLOCKS of Amino Acid SUBstitution Matrix)

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

© Rahul Singh

Understanding the Anatomy of BLOSUM

Our goal: To have an alignment score that reflects whether two sequences are homologous.

Let's start by having the following two hypotheses:

- (1) The aligned pair of residues is reflective of homology
- (2) The aligned pair is not reflective of homology (appears by chance)

We can then use the log-odds score (logarithm of the ratio of the likelihoods of the two hypotheses).

$$score(a,b) = \frac{1}{\lambda} \log \frac{P(ab)}{f_a \times f_b}$$

$P(a,b)$: likelihood that the two residues co-occur at this position because they are homologous

f_a, f_b : background frequencies: probability that we can observe residues a, b on average in any sequence

λ : scaling factor (used to round off terms to the nearest integer)

© Rahul Singh

$$score(a, b) = \frac{1}{\lambda} \log \frac{P(ab)}{f_a \times f_b}$$

If a, b are aligned more often than what can be expected by chance, then $P(ab) > f_a \times f_b$

In such a case, the ratio is > 1 and the log(.) is positive.

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

Why ?

For the data BLOSUM62 was trained on:

$$P(\text{Leucine}, \text{Leucine}) = 0.0371$$

$$P(\text{Tryptophan}, \text{Tryptophan}) = 0.0065$$

However, typtophan is a much rarer amino-acid:

$$f_L = 0.099$$

$$f_W = 0.013$$

$$\text{score}(L, L) = \frac{1}{(\lambda =) 0.347} \log \frac{0.0371}{(0.099)^2} = 3.83$$

$$\text{score}(W, W) = \frac{1}{0.347} \log \frac{0.0065}{(0.013)^2} = 10.51$$

© Rahul Singh

How To Get $P(a,b)$?

- The basic idea is to take a large number of known (trusted) alignments and count the frequency at which each residue pair occurs.

In computing the BLOSUM matrix, the BLOCKS database of alignments was taken and pair-wise sequence alignments related by some threshold percentage identity were taken.

For example, a threshold identity of 62% or less resulted in the target frequencies for BLOSUM62

Similarly thresholds of 80% and 45% gave the more highly conserved and the more divergent BLOSUM80 and BLOSUM45 respectively.

© Rahul Singh

Making-up Values

For smaller 4 x 4 DNA score matrices, we can even make up the target frequencies based on some assumptions.

Assume: all mismatches are equi-probable

For each nucleotide, composition of alignment and background sequence is 25%

We want to make a score matrix optimized for finding 88% identity alignments

Let $\lambda = 1$

Thus:

$$f_A = f_T = f_G = f_C = 0.25$$

$$P(\text{match}) = 0.22$$

$$P(\text{mis-match}) = 0.01$$

$$\text{score}(A, A) = \frac{1}{\lambda} \log\left(\frac{0.22}{0.25 \times 0.25}\right) = 1.258$$

$$\text{score}(A, C) = \frac{1}{\lambda} \log\left(\frac{0.01}{0.25 \times 0.25}\right) = -1.83$$

What is the difference between making-up target frequencies and calculating scores versus just making up scores?

© Rahul Singh

PAM (Point Accepted Mutation) Matrix

-Scores in the PAM matrix are computed by observing the substitutions that occur in alignments between similar sequences (empirically derived (data-driven) matrix).

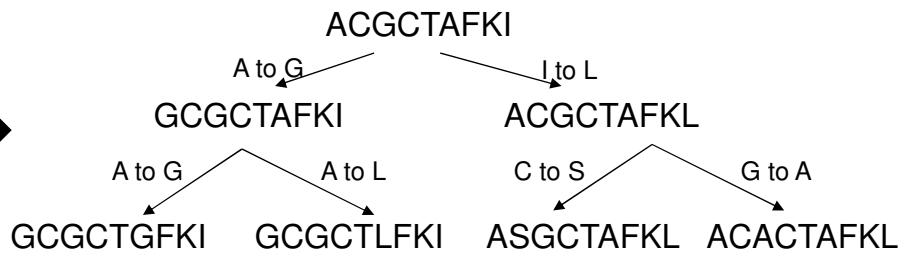
-This is done by aligning sequences having high (typically > 85%) identity. Consider the following example:

ACGCTAFKI, GCGCTAFKI, ACGCTAFKL, GCGCTGFKI, GCGCTLFKI,
ASGCTAFKL, ACAC TAFKL

Step 1: Construct the sequence alignment (multiple sequence alignment)

Step 2: Construct a phylogenetic tree that suggests which substitutions occurred in the sequences involved in the multiple alignment

ACGCTAFKI
GCGCTAFKI
ACGCTAFKL
GCGCTGFKI
GCGCTLFKI
ASGCTAFKL
ACACTAFKL



© Rahul Singh

Construction of the PAM Matrix (Contd.)

Step 3: For each amino acid, compute the frequency with which it is substituted. Substitutions are treated to be equally likely in each direction (i.e. A to G is also counted as G to A)

This can be done by counting all the X to Y and Y to X branches in the phylogenetic tree for amino acids X and Y. For example, $F_{GA} = 3$

Step 4: Compute the *relative mutability* of each amino acid

$$\text{Relative mutability } (m_{\text{amino-acid}}) = \frac{\text{No. of times an amino-acid is substituted}}{2 \times S_T \times F_{\text{amino-acid}} \times 100}$$

Where: S_T is the total number of substitutions across the entire tree

The use of the scaling factor 100 indicates 1 substitution across 100 residues

$F_{\text{amino-acid}}$: Is the relative frequency of a specific residue, given the data

For example, consider the A residues

$$\begin{aligned} \text{No. of times an amino-acid A is substituted} &= 4 \\ S_T &= 6 \\ F_A &= 10/63 = 0.159 \end{aligned} \quad \left. \right\} = 4/(12 \times 15.9) = 0.0209$$

© Rahul Singh

Construction of the PAM Matrix (Contd.)

Step 5: Compute the mutation probability (M_{ij}), for each pair of amino-acids. M_{ij} is given by the following formula:

$$M_{ij} = \frac{m_j F_{ij}}{\sum_i F_{ij}}$$

Where: m_j is the relative mutability of residue j
 F_{ij} is the frequency with which residue i is substituted by residue j
 $\sum_i F_{ij}$ denotes the total number of substitutions involving residue j in the phylogenetic tree

For example:

$$M_{GA} = \frac{m_A F_{GA}}{\sum_i F_{iA}} = \frac{0.0209 \times 3}{4} = 0.0156$$

© Rahul Singh

Construction of the PAM Matrix (Contd.)

Step 6: Let R denote the PAM matrix. Let R_{ij} denote the non-diagonal elements of R and let R_{ii} denote the diagonal elements of R

$$R_{ij} = \log\left(\frac{M_{ij}}{f_i}\right)$$

Here f_i denotes the frequency of occurrence of residue i in the multiple alignment. This can be obtained by dividing the number of occurrences of the residue i by the total number of residues.

For example, $f_G = 10 / 63 = 0.1587$, So

$$R_{GA} = \log\left(\frac{M_{GA}}{f_G}\right) = \log \frac{0.0156}{0.1587} = \log(0.0982) = -1.01$$

- For the diagonal entries, $M_{jj} = 1 - m_j$ (where m_j is the relative mutability). The steps enumerated above (Step 6) are then followed to obtain the diagonal elements R_{jj}

© Rahul Singh

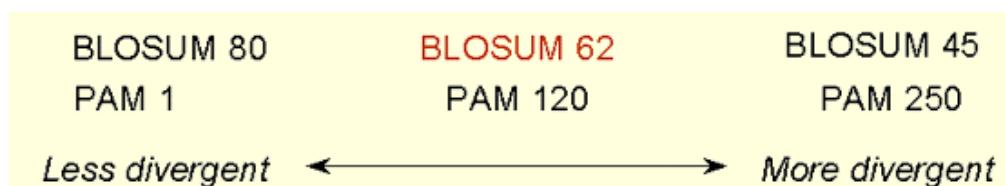
Semantics of the PAM Matrix

-The normalization of each entry in the PAM matrix is done such that the matrix represents substitution probabilities over a fixed unit of evolutionary change. For PAM-1, this is 1 substitution per 100 residues (or 1 PAM unit)

-So, the PAM-1 matrix contains values that denote the probabilities of the corresponding substitutions under the condition that: *We started with a sequence M at time t which then underwent changes till 1% of all residues underwent substitutions (assume this happened at time t+n) to get a new sequence M'*.

- PAM-1 is used for closely related sequences
- PAM-250 is commonly used in practice

Relationship Between BLOSUM and PAM



BLOSUM

1. Based on local alignment
2. Based on observed alignments. Not extrapolated from comparison of closely related proteins
3. BLOSUM62: calculated from sequences with identity less than 62% (divergence 62% or more)

PAM

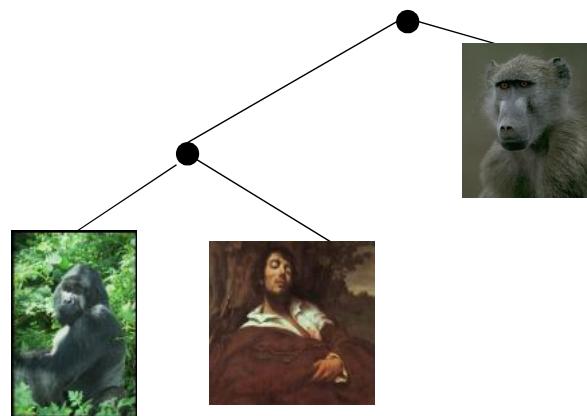
1. Based on global alignment
2. Based on closely related proteins (PAM1). Other PAM matrices are extrapolated from PAM1
3. PAM1: calculated from sequences with no more than 1% divergence

© Rahul Singh

Phylogenetics

© Rahul Singh

Phylogenetics: Study of evolutionary relationships amongst organisms or genes. Often a phylogenetic tree is constructed to illustrate the evolutionary relationships



Phenotype: The outward, physical manifestation of an organism. The phenotype can be used to classify an organism as determined by its physical or behavioral characteristics

Genotype: The internally coded, inheritable information carried by all living organisms

Earlier phylogenetic trees were built using morphological features (phenotypes). Today, typically DNA/protein data is used for this purpose

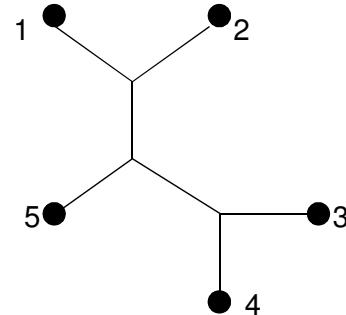
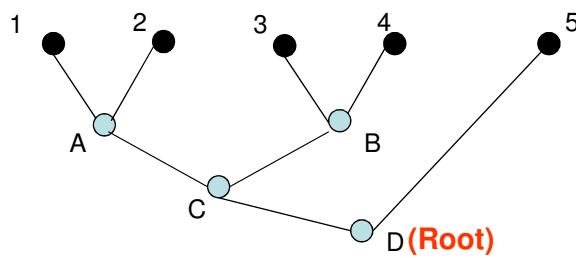
© Rahul Singh

Advantages to Molecular Phylogenetics

- Evolution is defined as genetic change – therefore genetic relationships are of primary importance in defining such relationships
- Prior to availability of molecular data, taxonomy was forced to work with comparisons of phenotypes to infer the genotypic relationships between organisms
 - Such characteristics could be, for example, anatomical or behavioral features
 - Traits like the above have been successfully used to construct evolutionary trees for many plants and animals
- However, phenotype-based approaches have their weaknesses
 - Similar phenotypes can develop in organisms that are actually distantly related (e.g. eyes in Humans and Fish)
 - Many organisms do not have easily expressible phenotypes that can be compared (e.g. easily observable traits between bacteria that relate to their genetic similarity are very difficult to find)
 - Phenotypes are often human-observed and qualitatively described. Very few rigorous (mathematical) attempts to detect and relate phenotypes

© Rahul Singh

Phylogenetic Trees and their Characteristics



-Phylogenetic trees are made by arranging nodes and branches.

- Each node represents a distinct taxonomical unit (called OTU: *operational taxonomic units*)

-There are two types of nodes:

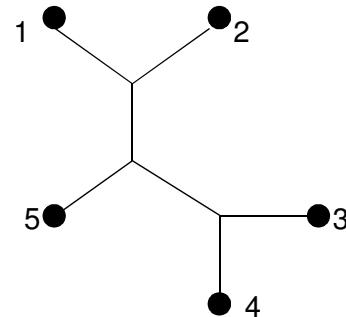
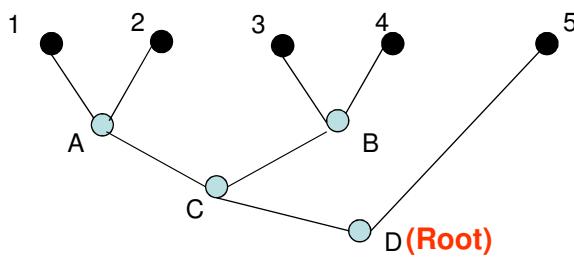
- *terminal nodes* are at the tip of the branches (dark filled circles)
- non-terminal nodes are called internal nodes (light filled circles)

-Terminal nodes correspond to a gene or an organism for which data has actually been collected

-Internal nodes correspond to an inferred common ancestor

© Rahul Singh

Rooted vs Unrooted Trees



- Some phylogenetic trees make an inference about a common ancestor and direction of evolution and some don't.

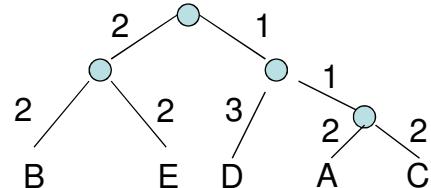
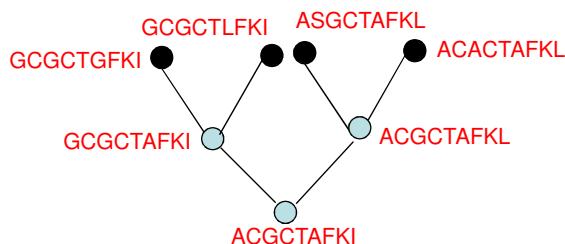
- Rooted Tree: A single node is designated as a common ancestor and for each descendant, there is a unique path leading from the root to it
- Unrooted Tree: Only show the relationship between nodes. No information is provided about the direction in which evolution occurred. An outgroup (an OTU which we have reason to think branched off earlier than other OTUs), can be used to root an unrooted tree. For example, we can make node 3 an outgroup to root the left tree.

© Rahul Singh

Un-Scaled vs Scaled Trees

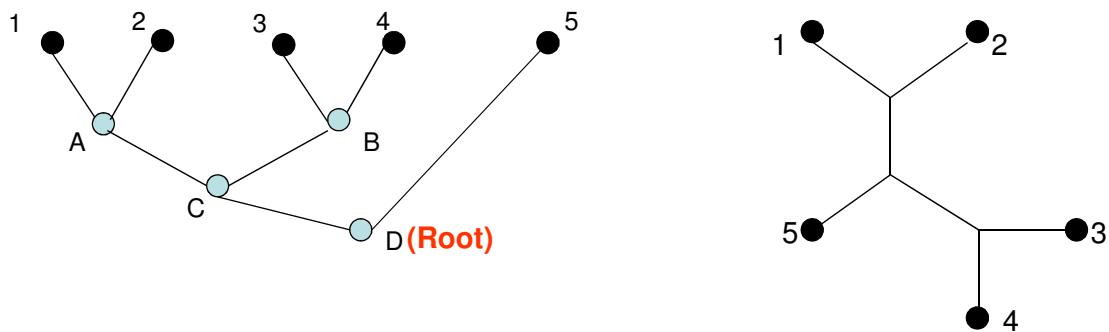
Unscaled Trees: Only convey the relative evolutionary relationships between participating nodes without attempting to convey any quantitative information about distinctions between nodes

Scaled Trees: If the phylogenetic tree is designed in a way that the length of the branches are proportional to the differences between neighboring nodes, it is called a scaled tree.



© Rahul Singh

Bifurcating and Multi-furcating Trees



All internal nodes of a bifurcating tree have 2 children if it is rooted and 3 neighbors Otherwise.

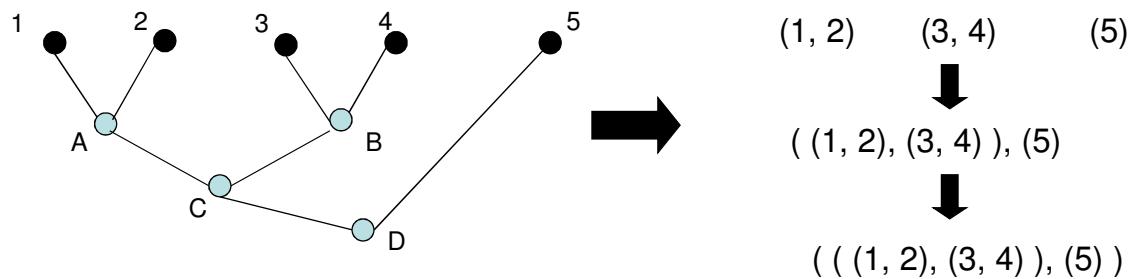
Multifurcating Trees: Trees in which internal nodes have three or more lineages are called multifurcating. This could be because (a) An ancestral population simultaneously gave rise to three lineages (b) two or more bifurcations happened at some point in time, but we can't distinguish the order in which they happened

We will focus only on bifurcating trees

© Rahul Singh

The Newick Format

The Newick format is a liner way of describing a phylogenetic tree. It uses a series of nested brackets (starting from the terminal nodes) to capture the structure of a specific tree



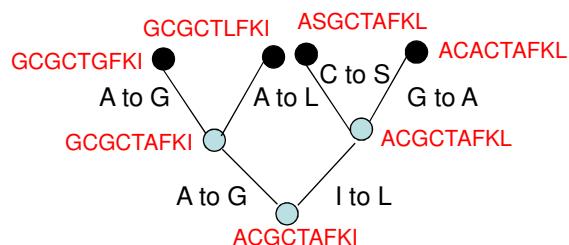
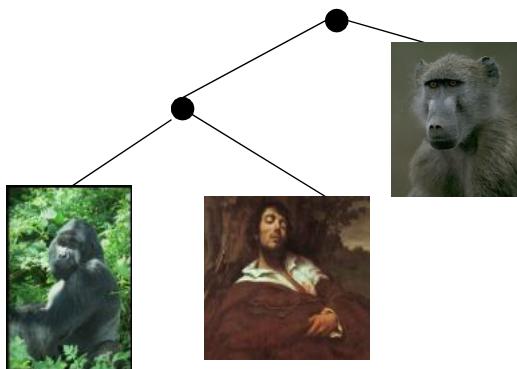
© Rahul Singh

Question

What is the phylogenetic tree for the Newick notation (((((A,B),C),(D,E)),F),G) ?

© Rahul Singh

Important Note: Species Trees and Gene Trees



Species trees are different from gene trees:

- A genes divergence may occur before speciation (due to genetic polymorphism in the population)
- In such cases, using the gene to infer the relationship amongst species can lead to overestimation of the branch length and/or incorrect topology
- Other issues that complicate the picture: gene duplication, horizontal gene transfer (transfer of genetic material from one genome to another – genes corresponding to such a case are called xenologs)
- Therefore to infer the species tree, one typically uses many orthologous genes

© Rahul Singh

Character and Distance Data

-Molecular data used in phylogenetic trees is of the two following types:

Character data: Consist of features that can take a limited number of values
For example, DNA sequence, Protein sequence, phenotypes (such as color), etc.

Distance data: Consist of overall pairwise difference between two data sets. Such information can easily be obtained from character data once a “measure” or “distance” between the character sets has been established

For example:

Gene 1: ASGCTAFLK, Gene 2: ACAC TAFKL

Measure: ASGCTAFLK
 ACAC TAFKL

Matches (M): 7

Sequence Size (S): 9

Measure (Distance) = M/S = 7/9

© Rahul Singh

Constructing Phylogenetic Trees

Three types of methods exist:

1. *Distance methods*: Given the “distances” between all OTUs, compute the phylogenetic tree
2. *Character-based methods* (maximum parsimony): Determine a phylogenetic tree that minimizes the changes required to explain the data
3. *Maximum likelihood methods*: Using a model of sequence evolution (e.g. the Jukes-Cantor model, Kimura’s model etc) determine the tree that has the highest probability of generating the observed data

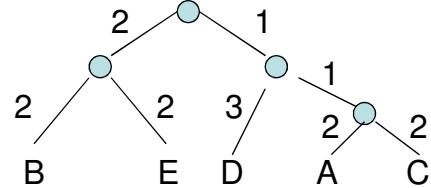
© Rahul Singh

Distance Methods

Given: n OTUs and an $n \times n$ distance matrix D , with D_{ij} (distance between OTU i and OTU j) ≥ 0

Goal: Construct an edge-weighted phylogenetic tree, such that the distance measured on the tree between leaves i and j equals the value D_{ij}

	A	B	C	D	E
A	0	8	4	6	8
B		0	8	8	4
C			0	6	8
D				0	8
E					0



When such a tree can be constructed, the distances in D are called **additive**

© Rahul Singh

Additive Metric Spaces

Definition 1: A metric space is a set of objects O , such that to every pair of objects $m, n \in O$ is associated a non-negative real number d_{mn} , such that:

$$d_{mn} > 0, \text{ for } m \neq n$$

$$d_{mn} = 0, \text{ iff } m = n$$

$$d_{mn} = d_{nm}, \forall m, n$$

$$d_{mn} \leq d_{mk} + d_{kn}, \forall m, n, k \text{ (triangle inequality)}$$

4-point condition: A metric space O is additive *iff* for any four objects of O , we can label them i, j, k, l , such that:

$$d_{ij} + d_{kl} = d_{ik} + d_{jl} \geq d_{il} + d_{jk}$$

© Rahul Singh

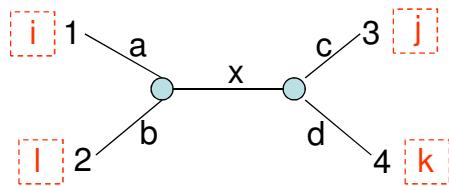
Intuition Behind the 4-point Condition

Consider the following tree for an additive matrix D with 4 species. The distances between the leaves are equal to the entries in the matrix D

We want to show that we can find a labeling i, j, k, l , such that:

$$d_{ij} + d_{kl} = d_{ik} + d_{jl} \geq d_{il} + d_{jk}$$

Consider the
Labeling:



Then:

$$\begin{array}{lll} d_{ij} = a + x + c & d_{ik} = a + x + d & d_{il} = a + b \\ d_{kl} = b + x + d & d_{jl} = b + x + c & d_{jk} = c + d \end{array}$$

And

$$d_{ij} + d_{kl} = a + b + c + d + 2x, \quad d_{ik} + d_{jl} = a + b + c + d + 2x$$

$$d_{il} + d_{jk} = a + b + c + d$$

© Rahul Singh

Are Distance Matrices Additive?

Unfortunately, distance matrices in general are ***not*** additive

So, constructing a phylogenetic tree, such that the distance measured on the tree between leaves i and j **equals** the value D_{ij} is difficult. However, we can try to build a tree that best fits the distance data.

There are several ways to define a “good fit” to the data. For example:

Fitch-Margoliash criterion: Choose a tree, where the distances d_{ij}

$$\text{minimize} \sum_{i,j} \frac{(D_{ij} - d_{ij})^2}{D_{ij}^2}$$

Cavalli-Sforza and Edwards criterion:

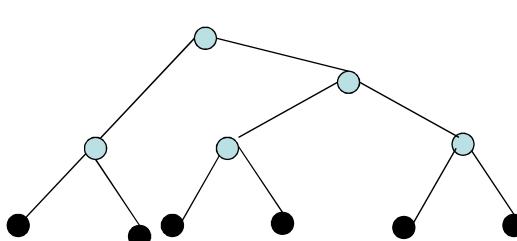
$$\text{minimize} \sum_{i,j} (D_{ij} - d_{ij})^2$$

© Rahul Singh

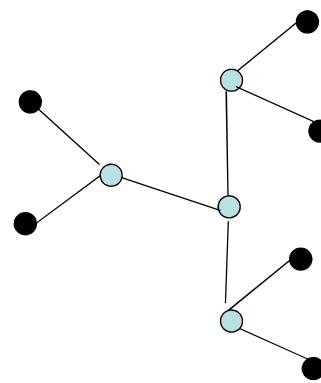
Can we Enumerate all the trees and pick the best one?

After all, we know:

Rooted Binary Tree



Unrooted Binary Tree



A rooted binary tree with n leaves has:

$2n - 2$ edges

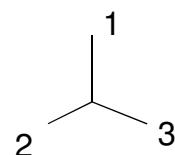
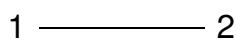
$n - 1$ internal nodes

An un-rooted binary tree with n leaves has:

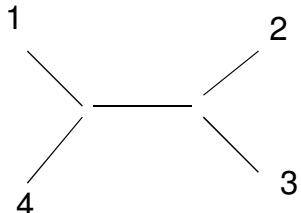
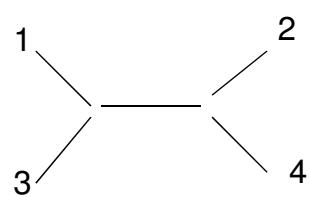
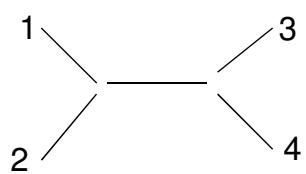
© Rahul Singh

Counting Trees

So, for 2 or 3 labeled leaves, there is only 1 un-rooted tree



- For 4 labeled leaves, there are three different un-rooted trees



© Rahul Singh

-Let us denote by $UN(n)$, the number of unrooted trees with n labeled leaves

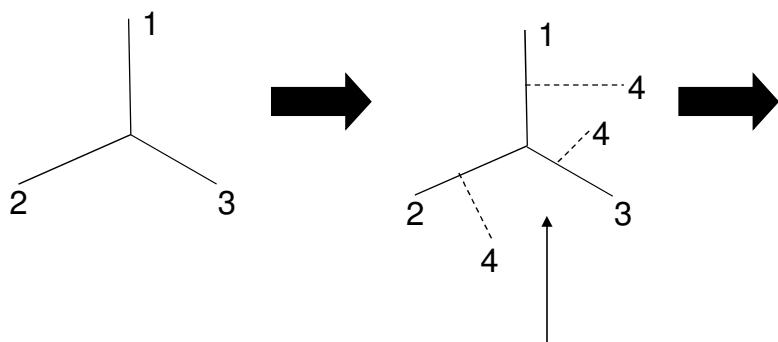
- From the previous examples, we have seen:

$$UN(2) = 1$$

$$UN(3) = 1$$

$$UN(4) = 3$$

-Note that for any unrooted tree with n leaves, an extra leaf can be added on any branch to get an unrooted tree with $n+1$ leaves



Given n leaves, there are:

$$UN(n) = \prod_{i=3}^n (2i - 5)$$

possible unrooted trees

1 unrooted binary tree with 3 leaves. Thus 3 possible ways to add a new leaf giving 3 possible trees with 4 leaves. For each of these 3 trees, there are 5 possible edges where a new leaf can be added, to get 15 trees with 5 leaves and so on...

© Rahul Singh

$n = 2$	$UN(n) = 1$
$n = 3$	$UN(n) = 1$
$n = 4$	$UN(n) = 3$
$n = 5$	$UN(n) = 15$
$n=10$	$UN(n)= 2,027,025$

$n = 2$	$R(n) = 1$
$n = 3$	$R(n) = 3$
$n = 4$	$R(n) = 15$
$n = 5$	$R(n) = 105$
$n=10$	$R(n)= 34,459,425$

Similarly, the number of rooted trees with n leaves can be shown to be: $\prod_{i=3}^n (2i - 3)$

-Bottom line: We can't devise algorithms that consider all possible trees to determine those that are optimal under a specific criteria

© Rahul Singh

Distance Method Heuristics: UPGMA

- UPGMA: Unweighted Pair Group Method with Arithmetic mean – developed in 1960s
- Statistically-based method
- Requires data that can be converted to a “distance” between all pairs of taxa being considered

Consider the set T consisting of the four taxa: $T = \{ A, B, C, D \}$

Let the pair-wise distances between them be represented as a distance matrix:

	A	B	C	D
A	0	d_{AB}	d_{AC}	d_{AD}
B	d_{AB}	0	d_{BC}	d_{BD}
C	d_{AC}	d_{BC}	0	d_{CD}
D	d_{AD}	d_{BD}	d_{CD}	0

-The distance between $(X, Y) = (Y, X) = d_{XY}$
for any $X, Y \in T = \{A, B, C, D\}$

-The distance between $(X, X) = d_{XX} = 0$
for any $X \in T = \{A, B, C, D\}$

-The distance matrix is symmetric about the diagonal

© Rahul Singh

The UPGMA Algorithm

Repeat the following step till all taxa are grouped

Step 1: Find the two species separated by the smallest distance in T. Cluster them into a single group (Note: we imply the smallest non-zero distance)

	A	B	C	D
A	0	d_{AB}	d_{AC}	d_{AD}
B	d_{AB}	0	d_{BC}	d_{BD}
C	d_{AC}	d_{BC}	0	d_{CD}
D	d_{AD}	d_{BD}	d_{CD}	0

→ Assuming that $\text{dist}(A, B) = d_{AB}$ is the smallest - group (A, B) together

Step 2: Remove the two species just clustered from T. Add the new cluster to the list of taxa T. Distance of the OTUs to the new OTU (cluster) is the mean distance.

Step 3: Compute a new distance matrix between the taxa. For the closest group (A,B) introduced in the taxa, this is done as follows: $d_{(AB)X} = \frac{1}{2}(d_{AX} + d_{BX}), \forall X \in T$

For our example: $d_{(AB)C} = \frac{1}{2}(d_{AC} + d_{BC})$ and $d_{(AB)D} = \frac{1}{2}(d_{AD} + d_{BD})$

© Rahul Singh

Example

- A: GTGCTGCACGG CTCAGTATA GCATTTACCC TTCCATCTTC AGATCCTGAA
- B: ACGCTGCACGG CTCAGTGCG GTGCTTACCC TCCCATCTTC AGATCCTGAA
- C: GTGCTGCACGG CTCGGCGCA GCATTTACCC TCCCATCTTC AGATCCTATC
- D: GTATCACACGA CTCAGCGCA GCATTTGCCC TCCCGTCTTC AGATCCTAAA
- E: GTATCACATAG CTCAGCGCA GCATTTGCCC TCCCGTCTTC AGATCTAAAA

$\text{dist}(X, Y)$ = number of non-matching bases in each pair

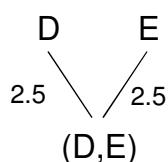


	A	B	C	D	E
A	0				
B	9	0			
C	8	11	0		
D	12	15	10	0	
E	15	18	13	5	0

© Rahul Singh

Step 1:

$$d_{\min} = d_{DE} = 5$$



Step 2:

Compute a new distance matrix

$$d_{(AB)X} = \frac{1}{2}(d_{AX} + d_{BX}), \forall X \in T, X \neq A, X \neq B$$

	A	B	C	D	E
A	0				
B	9	0			
C	8	11	0		
D	12	15	10	0	
E	15	18	13	5	0



	A	B	C	(D,E)
A	0			
B	9	0		
C	8	11	0	
(D,E)	13.5	16.5	11.5	0

© Rahul Singh

Iterating

Step 1:

$$d_{\min} = d_{AC} = 8$$



Step 2:

Compute a new distance matrix

	A	B	C	(D,E)
A	0			
B	9	0		
C	8	11	0	
(D,E)	13.5	16.5	11.5	0



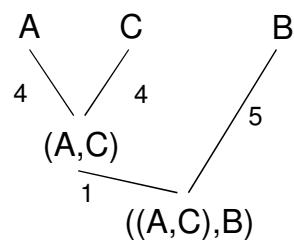
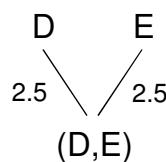
	(A,C)	B	(D,E)
(A,C)	0		
B	10	0	
(D,E)	12.5	16.5	0

© Rahul Singh

Iterating

Step 1:

$$d_{\min} = d_{(A,C),B} = 10$$



Step 2:

Compute a new distance matrix

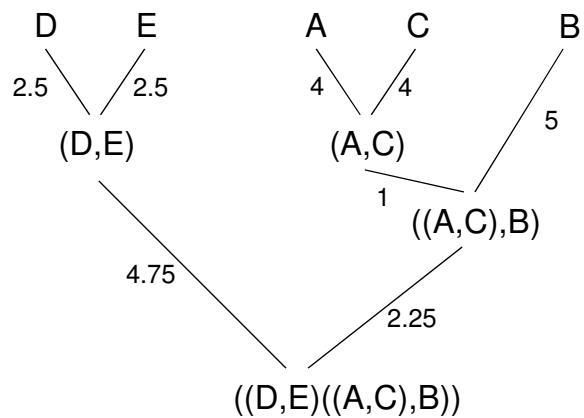
	(A,C)	B	(D,E)
(A,C)	0		
B	10	0	
(D,E)	12.5	16.5	0



	((A,C),B)	(D,E)
((A,C),B)	0	
(D,E)	14.5	0

© Rahul Singh

The Final Tree



© Rahul Singh

Compute the Tree

	A	B	C	D	E
A	0	8	4	6	8
B		0	8	8	4
C			0	6	8
D				0	8
E					0

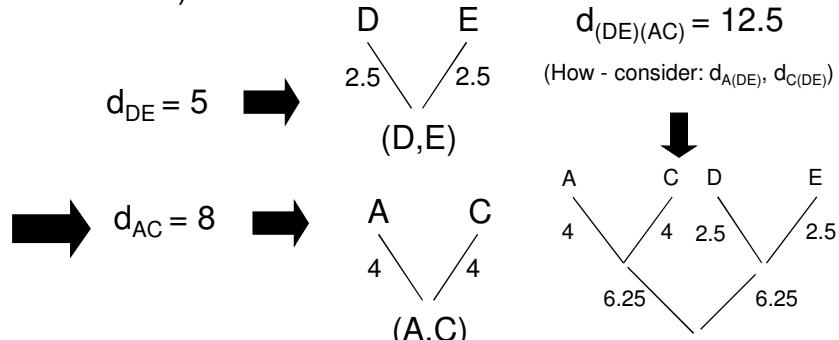
© Rahul Singh

Estimating Branch Lengths in UPGMA

- Topology of the phylogenetic tree can describe the relative degree to which the sequences have diverged
- This can be done by having the length of the branches correspond to the inferred amount of time that the sequences have been accumulating substitutions independently
- The relative length of each branch can be computed from the distance matrix
- If the rate of evolution is assumed to be constant in all lineages, then the internal nodes can be placed at equal distances from each of the species they give rise to, on a bifurcating part of the tree (as in UPGMA)

For Example:

	A	B	C	D	E
A	0				
B	9	0			
C	8	11	0		
D	12	15	10	0	
E	15	18	13	5	0



© Rahul Singh

Branch Lengths and Ultrametric Distances

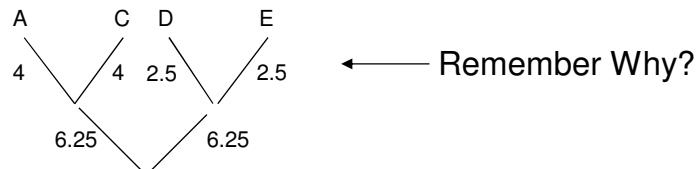
- If the rate of evolution among different OTUs is exactly the same, then we have ultrametric data
- The results of UPGMA are exact for such cases
- How to deal with anisotropic rates?

© Rahul Singh

Transformed Distance Method

-A drawback of UPGMA is that it assumes constant rate of evolution. Therefore, variations in the rate of substitution can lead to trees with incorrect topologies

	A	B	C	D	E
A	0				
B	9	0			
C	8	11	0		
D	12	15	10	0	
E	15	18	13	5	0



- Per the tree $d_{AE} = 4 + 6.25 + 6.25 + 2.5 = 19$
- Per the table $d_{AE} = 15$

The transformed distance method uses an alternate frame of reference to address this problem. It uses the following insight:

Consider a group of species under consideration. An **outgroup** is a species that is known to have diverged from the common ancestor of the other species earlier than the rest (of the species). These remaining species are called the **ingroup**

© Rahul Singh

Observation

The ingroups evolve separately from each other ONLY after they have diverged. Any differences in the number of substitutions they have accumulated can only be since that time

-Therefore, outgroups can be used as an objective frame of reference for comparing these substitutions

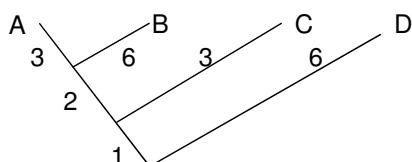
Example

A: GTGCTGCACGG CTCAGTATA GCATTTACCC TTCCATCTTC AGATCCTGAA
B: ACGCTGCACGG CTCAGTGCG GTGCTTACCC TCCCATCTTC AGATCCTGAA
C: GTGCTGCACGG CTCGGCGCA GCATTTACCC TCCCATCTTC AGATCCTATC
D: GTATCACACGA CTCAGCGCA GCATTTGCC C TCCCCTCTTC AGATCCTAAA

	A	B	C	D
A	0			
B	9	0		
C	8	11	0	
D	12	15	10	0

-Assume D to be an outgroup to A, B, and C

-Also assume the true relation to be:



© Rahul Singh

Using UPGMA

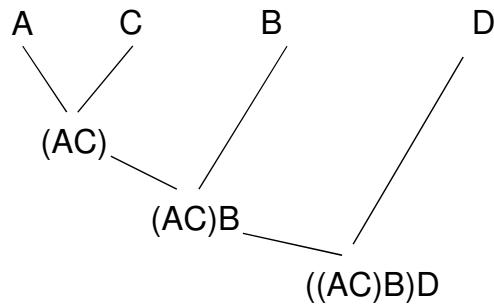
	A	B	C	D	
A	0				
B	9	0			
C	8	11	0		
D	12	15	10	0	

(AC) →

	(AC)	B	D	
(AC)	0			
B	10	0		
D	11	15	0	

(AC)B →

	(AC)B	D	
(AC)B	0		
D	13	0	



© Rahul Singh

Transformed Distance Method

Step 1: Assume an outgroup (e.g. D)

Step 2: Re-compute the distance matrix as follows:

$$d'_{ij} = \frac{(d_{ij} - d_{iD} - d_{jD})}{2} + \bar{d}_D$$

$$\bar{d}_D = \frac{d_{AD} + d_{BD} + d_{CD}}{3} = 37 / 3$$

$$d'_{ii} = 0$$

Step 3: Run the UPGMA algorithm

© Rahul Singh

Using the formulae, a new distance matrix is generated

	A	B	C	D
A	0			
B	9	0		
C	8	11	0	
D	12	15	10	0

$$d'_{AB} = \frac{(d_{AB} - d_{AD} - d_{BD})}{2} + \frac{37}{3} = -9 + \frac{37}{3} = \frac{10}{3}$$

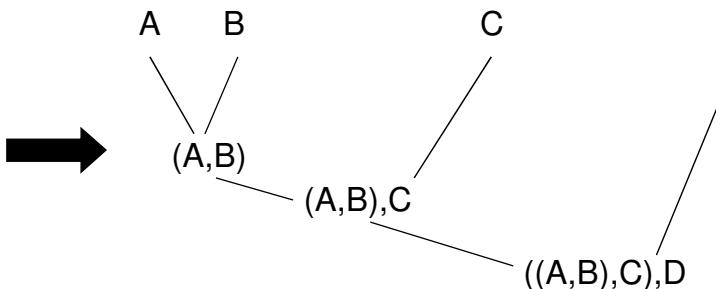
$$d'_{AC} = \frac{(d_{AC} - d_{AD} - d_{CD})}{2} + \frac{37}{3} = -7 + \frac{37}{3} = \frac{16}{3}$$

$$d'_{AD} = \frac{(d_{AD} - d_{AB} - d_{DD})}{2} + \frac{37}{3} = \frac{37}{3}$$

$$d'_{BC} = \frac{(d_{BC} - d_{BD} - d_{CD})}{2} + \frac{37}{3} = -7 + \frac{37}{3} = \frac{16}{3}$$



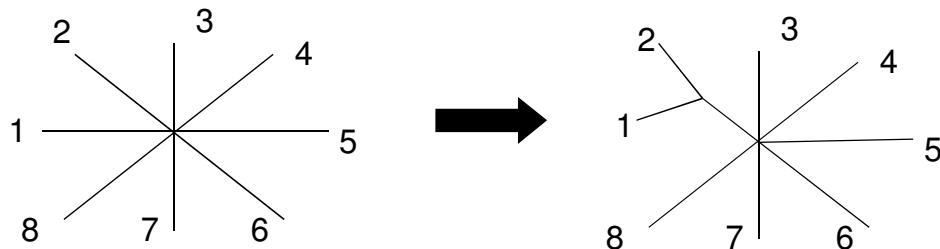
	A	B	C	D
A	0			
B	10/3	0		
C	16/3	16/3	0	
D	37/3	37/3	37/3	0



Question: How to select the outgroup?

© Rahul Singh

Neighbor-Joining Methods



Basic Idea

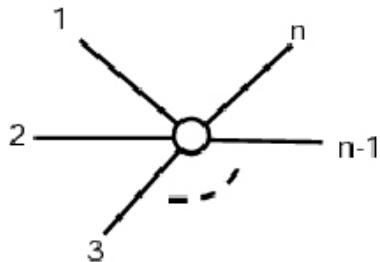
- Start with a star-like tree, with all species coming off a central node
- Neighbors are then found that minimize the total length of branches on the tree
 - The most similar terminals are joined and a branch is inserted between them and the remainder of the star
 - The new branch is consolidated by having its value equal the mean of the two original values
- Process is repeated till only one terminal remains

© Rahul Singh

Neighbor Joining Algorithm

Step 1: Initialize (Matrix D contains the distances between the species)

- Assign a (single) cluster to each OTU. Each initial cluster is a singleton
- Assign a leaf to each species/OTU in the tree



Iterate while more than two nodes (clusters) remain

Step 2: For a node i , its distance from the rest of the tree is denoted as u_i

$$u_i = \sum_{k \neq i} \frac{D_{i,k}}{n-2}$$

Compute u_i for each species i

© Rahul Singh

Step 3: Choose the clusters i and j such that

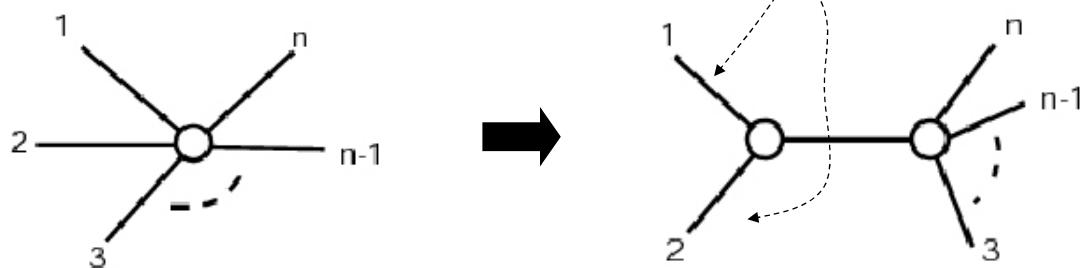
$$D_{i,j} - u_i - u_j \quad \text{is the smallest}$$

Note: this gives the two nodes with smallest sum of branch lengths

Step 4: Join the clusters i and j to form a new cluster (i,j) with a corresponding node in T. Compute the distances from i and j to (i,j) as follows:

$$d_{i,(i,j)} = \frac{1}{2} D_{i,j} + \frac{1}{2} (u_i - u_j),$$

$$d_{j,(i,j)} = \frac{1}{2} D_{i,j} + \frac{1}{2} (u_j - u_i)$$



© Rahul Singh

Step 5: Compute the distance between the new cluster (i,j) and every other cluster (excluding i and j) in the tree

$$D_{(i,j),k} = \frac{D_{i,k} + D_{j,k} - D_{i,j}}{2}$$

Step 6: Remove the clusters i and j from the distance table and replace them by (i,j)

Join the two remaining clusters by an edge of length equal to the distance between them

Note:

- For additive distance matrices, neighbor joining infers the correct tree connectivity
- For non-additive matrices, neighbor joining works well in practice

© Rahul Singh

Phylogenetics: Character-Based Methods

© Rahul Singh

Parsimony

Based on two ideas:

- Mutations/Substitutions are extremely rare events
- A model which invokes a larger number of unlikely events (mutations/substitutions) is *less likely* to be correct than one which invokes a fewer number of unlikely events

In other words, a model that requires the fewest number of mutations/substitutions to explain the current state of a sequence is most likely to be correct. This model (tree) is called the most parsimonious model (tree)

-Note: Not all sites within multiple sequence alignments have useful information for determining the most parsimonious tree. Consider the following example:

Site:	1	2	3	4	5	6
Sequence 1	G	G	G	G	G	G
2	G	G	G	A	G	T
3	G	G	A	T	A	G
4	G	A	T	C	A	T

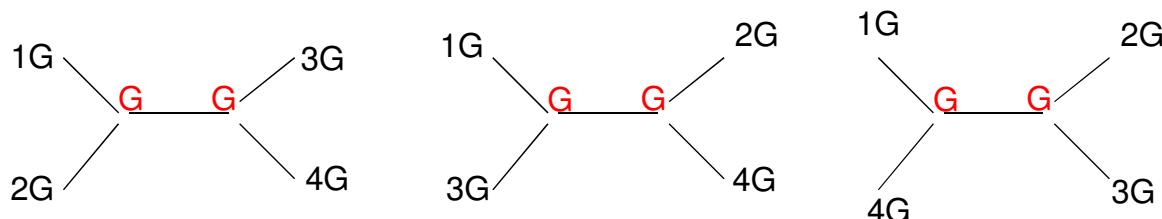
We shall see that only sites 5 and 6 are “informative” for parsimony

© Rahul Singh

Consider site 1

	Site: 1	2	3	4	5	6
Sequence 1	G	G	G	G	G	G
2	G	G	G	A	G	T
3	G	G	A	T	A	G
4	G	A	T	C	A	T

We know that for 4 labeled leaves, there are three different unrooted trees



How many substitutions for each tree?

Tree-1: 0

Tree-2: 0

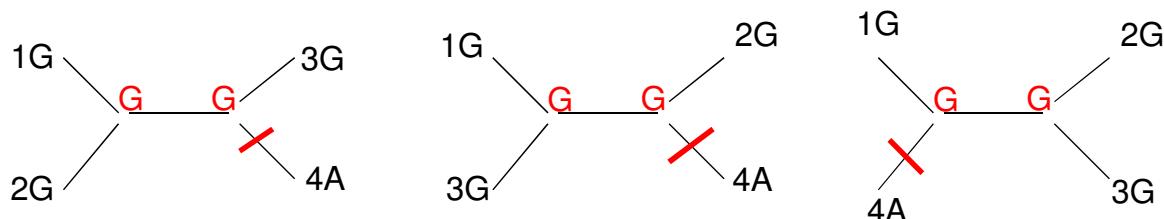
Tree-3: 0

© Rahul Singh

Consider site 2

	Site:	1	2	3	4	5	6
Sequence	1	G	G	G	G	G	G
2	G	G	G	A	G	T	
3	G	G	A	T	A	G	
4	G	A	T	C	A	T	

We know that for 4 labeled leaves, there are three different unrooted trees



How many substitutions for each tree?

Tree-1: 1

Tree-2: 1

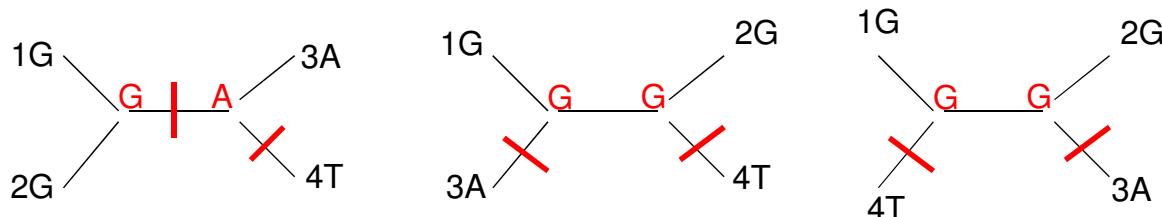
Tree-3: 1

© Rahul Singh

Consider site 3

Site:	1	2	3	4	5	6
Sequence 1	G	G	G	G	G	G
2	G	G	G	A	G	T
3	G	G	A	T	A	G
4	G	A	T	C	A	T

We know that for 4 labeled leaves, there are three different unrooted trees



How many substitutions for each tree?

Tree-1: 2

Tree-2: 2

Tree-3: 2

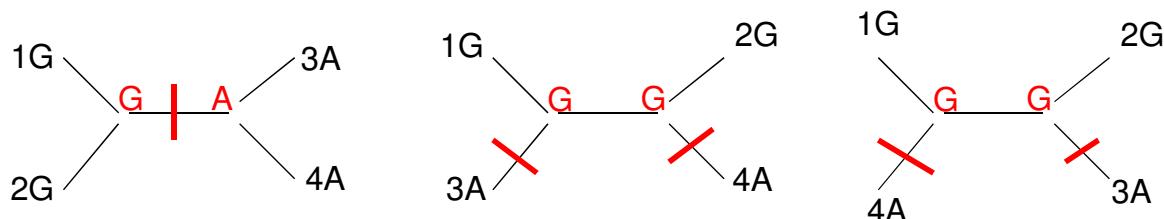
Similarly, all trees for site 4 require 3 mutations. However, sites 5 and 6 are different:

© Rahul Singh

Consider site 5

	Site:	1	2	3	4	5	6
Sequence	1	G	G	G	G	G	G
2	G	G	G	A	G	T	
3	G	G	A	T	A	G	
4	G	A	T	C	A	T	

We know that for 4 labeled leaves, there are three different unrooted trees



How many substitutions for each tree?

Tree-1: 1

Tree-2: 2

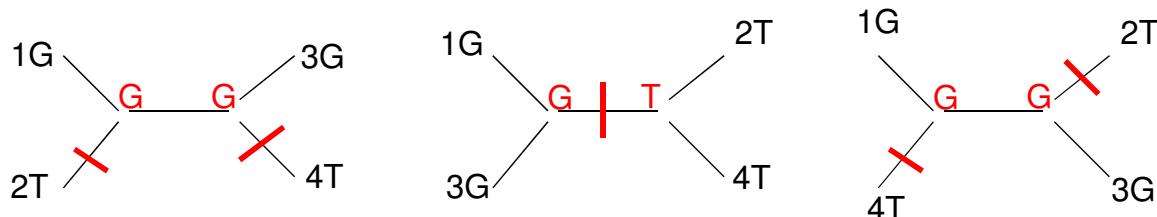
Tree-3: 2

© Rahul Singh

Consider site 6

Site:	1	2	3	4	5	6
Sequence 1	G	G	G	G	G	G
2	G	G	G	A	G	T
3	G	G	A	T	A	G
4	G	A	T	C	A	T

We know that for 4 labeled leaves, there are three different unrooted trees



How many substitutions for each tree?

Tree-1: 2

Tree-2: 1

Tree-3: 2

© Rahul Singh

Site:	1	2	3	4	5	6
Sequence 1	G	G	G	G	G	G
2	G	G	G	A	G	T
3	G	G	A	T	A	G
4	G	A	T	C	A	T

-For a position to be informative:

- (a) It has to have at least 2 different nucleotides (characters)
- (b) Each of the nucleotides have to be present at least twice

-All parsimony approaches begin by applying this simple rule to the data set

- After the non-informative sites are eliminated, every tree is considered for each site.
The tree that induces the minimum number of substitutions is identified
- After all informative sites have been considered, the tree (or trees) that induce the smallest number of changes are reported as the most parsimonious

© Rahul Singh

Site:	1	2	3	4	5	6
Sequence	1	G	G	G	G	G
	2	G	G	G	A	G
	3	G	G	A	T	A
	4	G	A	T	C	A
						T
		Tree-1	Tree-2	Tree-3		
Site 1	0	0	0	0		
Site 2	1	1	1	1		
•	2	2	2	2		
•	3	3	3	3		
	1	2	2	2		
Site 6	2	1	1	2		
		9	9	10		

How is this different from a distance-based approach? (point-out the issue)

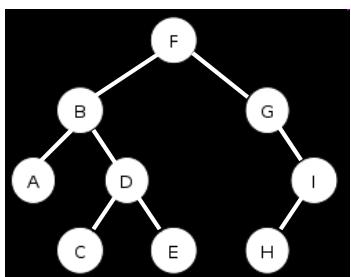
© Rahul Singh

Small and Large Parsimony

- Given a tree, how to find the most parsimonious labeling of the ancestral sequence (minimum number of changes required to explain the data?) That is, *we are given a set of character data and a tree, with leaves labeled by species. We need to label the internal nodes so as to minimize the number of changes.*
- How to search through all the possible trees? That is, *given a set of character data find the most parsimonious tree.*

Problem 1 is easy (the Fitch algorithm). Problem 2 is NP hard!

A note on tree traversals



Pre-order traversal: root – left subtree – right subtree
FBADCEGIH

In-order traversal: left subtree – root - right subtree
ABCDEFGHI

Post-order traversal: left subtree – right subtree – root
ACEDBHIGF

© Rahul Singh

The Fitch Algorithm

Step 1:

For each site

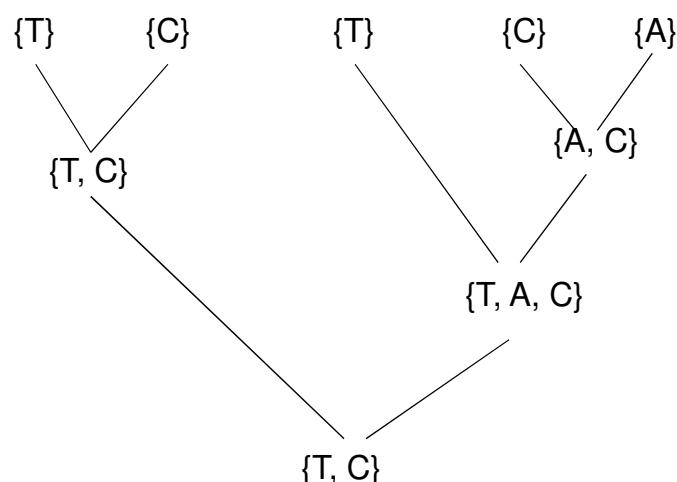
- Each leaf is labeled with a set containing observed nucleotide at that position
- For each internal node i having children j and k with corresponding labels S_j and S_k (Note: we are traversing the tree starting from the leaves to the root, i.e. a post-order traversal)

$$S_i = \begin{cases} S_j \cup S_k & \text{if } S_j \cap S_k = \emptyset \\ S_j \cap S_k & \text{otherwise} \end{cases}$$

Total number of changes necessary for a site is the number of union operations

© Rahul Singh

Example



Step 2

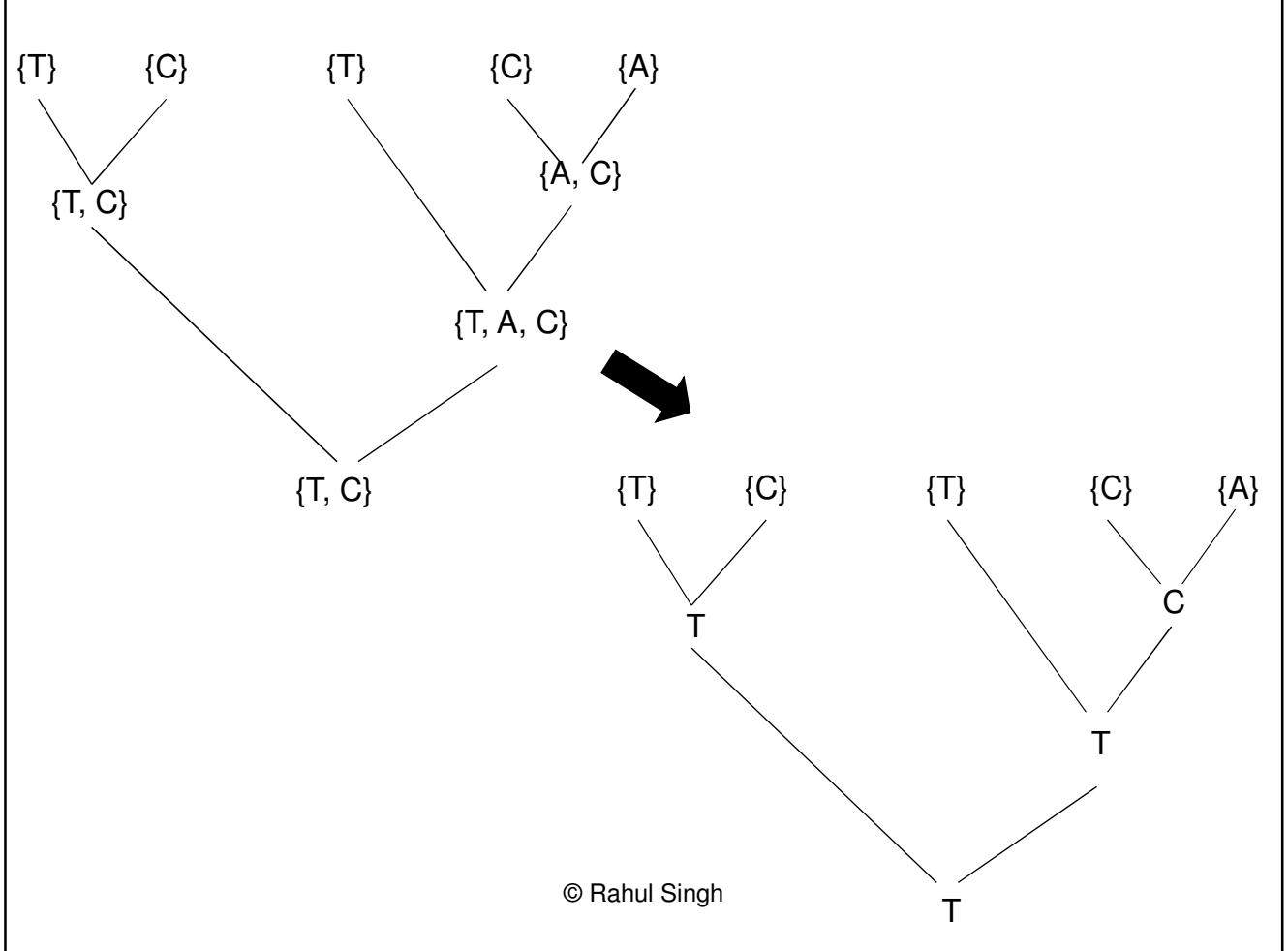
Given S_i , we need to determine the character i_c to be assigned to each internal node i

The tree is now traversed pre-order, that is from the root to the leaves

For each internal node i , if its parent u satisfies:

$u_c \in S_i$, then $i_c \leftarrow u_c$
else
 $i_c \leftarrow t : t \in S_i$ (including the root)

© Rahul Singh



A short digression: Inferred Ancestral Sequences

	Site:	1	2	3	4	5	6
Sequence	1	G	G	G	G	G	G
	2	G	G	G	A	G	T
	3	G	G	A	T	A	G
	4	G	A	T	C	A	T

-Consider the sequences 1 and 2 from the example we have seen earlier

- Even though the ancestor of 1 and 2 may be extinct, parsimony makes a strong prediction that the nucleotide found, for example, in the 5th position of the sequence of this ancestor was a G

- Considered over the entire genome, this can provide valuable insights about ancient organisms especially in the context of proteins whose structure and function are well understood

A Must Read: "Species Adaptation in a Protein Molecule", M. Perutz, Mol. Biol. Evol., Vol. 1, pp. 1- 28, 1983

© Rahul Singh

The Large Parsimony Problem

Input: A matrix M describing the characters for n species

Output: The most parsimonious phylogenetic tree

This is a difficult problem

-The number of possible unrooted trees increases rapidly with the number of sequences in the alignment

-Once a tree is given, we can evaluate it pretty rapidly but we don't know which tree to use

-This problem is known to be NP-Hard

What is NP-Hard?

Intuitively:

- A problem is NP, if a NDTM can solve it in polynomial time (SAT, TSP, etc)

- A problem is NP hard if it is at least as hard as a NP problem

© Rahul Singh

Strategies for Faster Searches

Branch and Bound

- Start by obtaining an upper bound (L) on the length of the most parsimonious tree for the given data set (initialization)
 - L can be obtained from a UPGMA-based tree derivation or
 - L can be obtained from a random tree that described the relations between the data
- Start with a tree that described the relationships between a few of the sequences involved
- Incrementally “grow” this tree by adding branches one at a time
- If you get a tree that requires more substitutions than L , discard that tree (because we already know another tree that is more parsimonious)
- If you find a tree that requires fewer substitutions than L , change the value of L accordingly
- Branch and bound guarantees that no trees more parsimonious than L have been missed

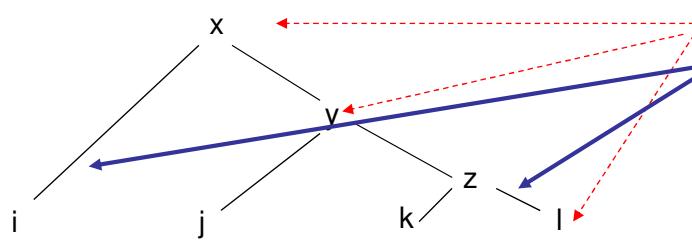
© Rahul Singh

Maximum Likelihood

- We have studied different probabilistic models for nucleotide/protein substitution
Jukes-Cantor
PAM
Kimura's model etc.

- Idea of ML approach: Given a model for nucleotide/protein substitution,
Given a data set
Pick the phylogenetic tree that maximizes the probability of generating the observed data

- What does it mean to pick a tree?



1. Each node in the tree has a label
2. Each branch in the tree has a length, which can denote the biological time or genetic distance between the species

Picking a tree would require us to solve both these problems

- We shall use the term *reconstruction* to denote **a labeling** of the internal nodes of a tree

© Rahul Singh

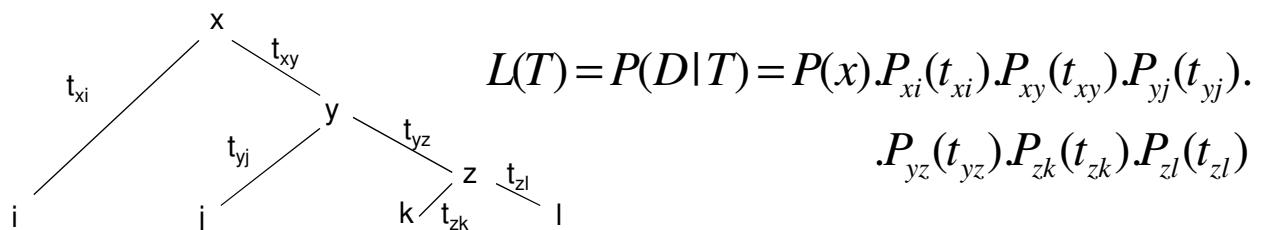
How to compute the likelihood of a tree?

- We shall start by making two assumptions

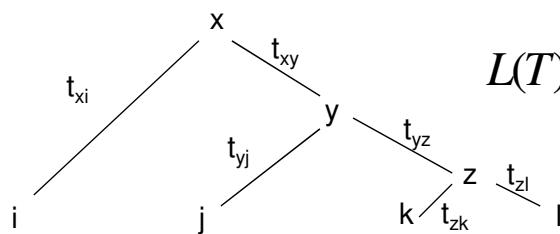
1. Characters are pair-wise independent
2. Branching is a Markov process: The probability of a node having a specific label is only dependent on its parent node and the branch length between them

Also note that the substitution model gives us a way to determine the probability that the label x will transform into label y in a time period T (denoted as $P_{xy}(T)$)

Let's start with an example



© Rahul Singh



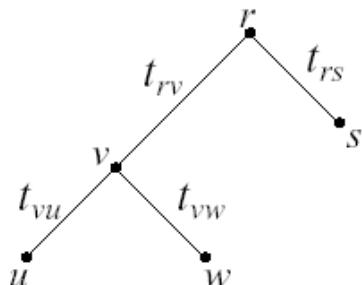
$$L(T) = P(D|T) = P(x)P_{xi}(t_{xi})P_{xy}(t_{xy})P_{yz}(t_{yz}).$$

$$P_{yz}(t_{yz})P_{zk}(t_{zk})P_{zl}(t_{zl})$$

Since the internal labels are unknown, we need to compute the above value for all possible values of the internal labels (reconstructions):

$$L(T) = P(D|T) = \sum_{x} \sum_{y} \sum_{z} P(x)P_{xi}(t_{xi})P_{xy}(t_{xy})P_{yz}(t_{yz})P_{zk}(t_{zk})P_{zl}(t_{zl})$$

Lets try another example:



$$L(T) = P(D|T) = \sum_{r} \sum_{v} P(r)P_{rs}(t_{rs})P_{rv}(t_{rv})P_{vu}(t_{vu})P_{vw}(t_{vw})$$

© Rahul Singh

BUT!

We made a simplifying assumption that only one character identifies each species

- How do we calculate the likelihood of a tree when there are multiple characters identifying each species?

Recall our first assumption: *Characters are pair-wise independent*

So:

- We repeat the previous calculations for each character
- And multiply the results

The general formula now is:

$$\begin{aligned} L(T) &= P(D | T) = \prod_{\text{character } j} P(D_j | T) \\ &= \prod_{\text{character } j} \left\{ \sum_{\text{reconstruction } R} P(D_j, R | T) \right\} \\ &= \prod_{\text{character } j} \left\{ \sum_{\text{reconstruction } R} \left(P(\text{root}) \cdot \prod_{\text{edge } uv} P_{uv}(t_{uv}) \right) \right\} \end{aligned}$$

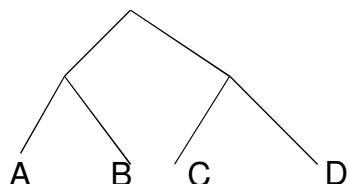
© Rahul Singh

Efficiently computing the likelihood

- Dynamic programming can be used to efficiently compute the likelihoods ($P(D|T, M)$)
- Finding the best ML-Tree is more complex and expensive:
 1. We need to consider all tree topologies
 2. We need to determine the lengths of each tree branch in every topology
(Optimization techniques such as EM or Newton-Raphson are typically used to do this)

Assessing confidence in a phylogenetic hypothesis

- Each phylogenetic tree represents a hypothesis about the evolutionary history of the sequences being studied
- How much confidence can be assigned to each such hypothesis (the overall tree and its component branches)?



What is the confidence level that A and B belong together? (C and D belong together?)

© Rahul Singh

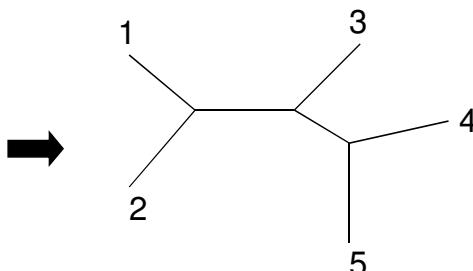
Bootstrapping: Assessing Reliability

Basic Idea

1. Draw a subset from the original data with replacement (a pseudosample)
2. Construct a tree using this pseudosample
3. Determine if the hypothesis is true
4. Repeat
5. The confidence of a hypothesis is the percentage of times it was true in the above iterations

Example

Site:	1	2	3	4	5	6	7	8	9	10
Sequence	G	G	G	G	G	G	A	T	C	A
1	G	G	G	A	G	T	A	T	C	A
2	G	G	A	T	A	G	A	C	A	T
3	G	G	A	T	A	G	A	C	A	T
4	G	A	T	C	A	T	G	T	A	T
5	G	T	T	C	A	T	A	T	C	T

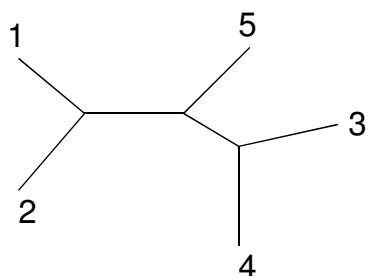


What is the confidence that (1, 2) and (4, 5) are together?

© Rahul Singh

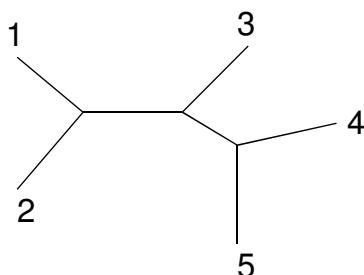
Iteration 1

Site:	1	1	3	5	5	5	6	9	9	9
Sequence										
1	G	G	G	G	G	G	G	C	C	C
2	G	G	G	G	G	G	T	C	C	C
3	G	G	A	A	A	A	G	A	A	A
4	G	G	T	A	A	A	T	A	A	A
5	G	G	T	A	A	A	T	C	C	C
	⋮									

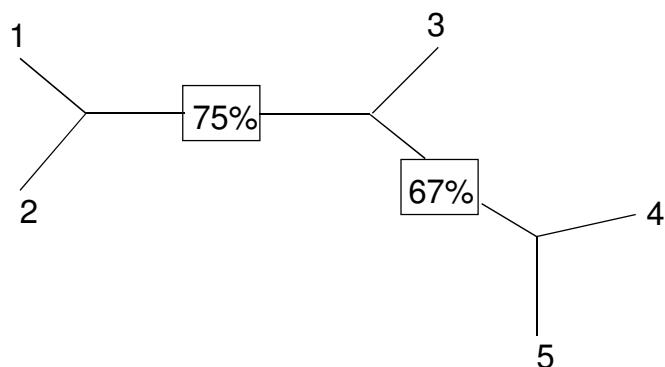


Iteration k

Site:	1	2	2	4	5	5	5	7	7	10
Sequence										
1	G	G	G	G	G	G	G	A	A	A
2	G	G	G	A	G	G	G	A	A	A
3	G	G	G	T	A	A	A	A	A	T
4	G	A	A	C	A	A	A	G	G	T
5	G	T	T	C	A	A	A	A	A	T



© Rahul Singh



- Commonly used technique
- Can be very time consuming depending on how the tree is obtained
- Does not give reliable results unless the number of iterations are large (hundreds)
- Tends to underestimate high confidence values and overestimate low confidence values
- Some results may appear to be statistically significant simply by chance since many groupings are being considered

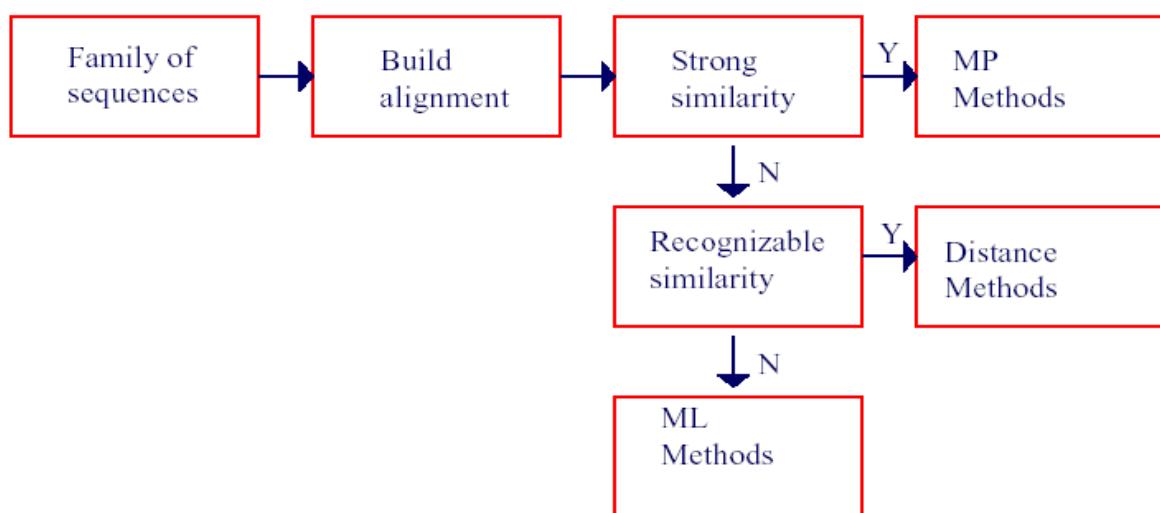
© Rahul Singh

Types of Phylogenetic Algorithms

		COMPUTATIONAL METHOD	
		Optimality criterion	Clustering algorithm
DATA TYPE	Characters	PARSIMONY MAXIMUM LIKELIHOOD	
	Distances	MINIMUM EVOLUTION LEAST SQUARES	UPGMA NEIGHBOR-JOINING

© Rahul Singh

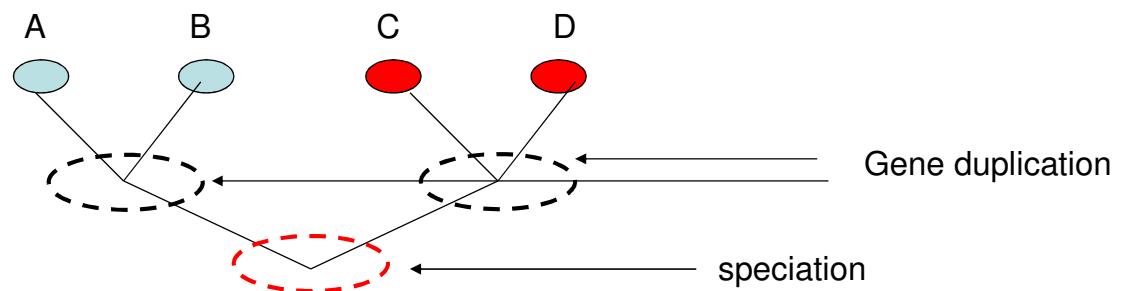
Which technique to use?



© Rahul Singh

Things Can Still Go Wrong!

- Distance tables may not match true evolutionary distances
- The most parsimonious tree may not be the true tree
- The most likely tree may not be the true tree



Orthologs: AC, AD, BC, BD

Paralogs: AB, CD

- Do not mix orthologs and paralogs in building phylogenetic trees!

© Rahul Singh



PHYLIP

PHYLIP is a *free* package of programs for inferring phylogenies. It is distributed as source code, documentation files, and a number of different types of executables. These Web pages, by [Joe Felsenstein](#) of the [Department of Genome Sciences](#) and the [Department of Biology](#) at the [University of Washington](#), contain information on PHYLIP and ways to transfer the executables, source code and documentation to your computer.

New as of 30 August 2005:

PHYLIP 3.65 is now available. For details on getting PHYLIP see the link below "Get Me PHYLIP".

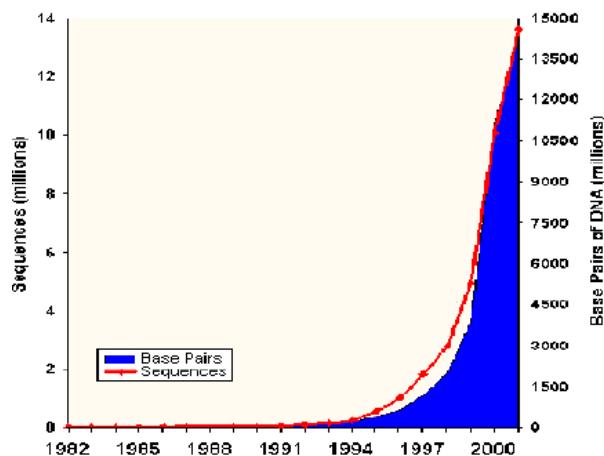
- ♦ A [general description](#) of PHYLIP.
- ♦ [Programs](#) in the PHYLIP package
- ♦ About the [Executables](#)
- ♦ About the [Source code](#) ... compiling it yourself
- ♦ The documentation web pages for PHYLIP can be read [here](#)
- ♦ [Get me PHYLIP](#)
- ♦ [How to install PHYLIP](#)
- ♦ [Frequently asked questions](#)
- ♦ An excellent guide to using PHYLIP with molecular data is available [here](#).
- ♦ [PHYLIP on the web](#) (HTML documentation, server services)
- ♦ [Current and future versions of PHYLIP \(including new features\)](#)
- ♦ [Older versions of PHYLIP, including version 3.5](#)
- ♦ [Bugs in the package, known or recently fixed](#)
- ♦ [Phylogeny programs](#) available elsewhere
- ♦ [Credits \(people, grants etc.\)](#)

© Rahul Singh

Introduction to Gene Finding

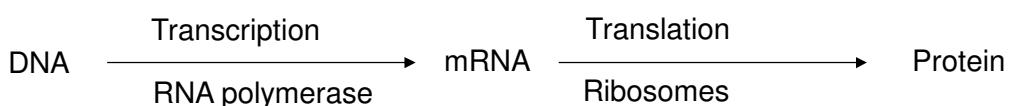
© Rahul Singh

The Problem



- There is a huge amount of data in sequence databases (like GenBank)
 - Not all sequences are coding, i.e. encode a protein
 - In the human genome for example less than 3% of the sequences are thought to be coding
 - Manual search of databases to find coding sequences is difficult
 - How to automatically find the genes?
- At the current state-of-the-art we can't do ab-initio gene identification for all genes
© Rahul Singh

Remembering the Biological Background



Central dogma of Molecular Biology

- The genetic information carried by an organism is inscribed in a DNA molecule
- Each functional portion of this molecule is called a gene
- For protein synthesis, each gene is *transcribed* into a molecule called RNA
- The RNA molecule is *translated* by mediation of ribosomes, transfer RNA, and associated enzymes into an amino acid chain
- The amino-acid chain then *folds* into a protein

© Rahul Singh

Remembering the Biological Background: Key Concepts

Promoter: A promoter is a DNA sequence that enables a gene to be transcribed.
In RNA synthesis, promoters demarcate the genes used for mRNA creation.

RNA Polymerase (RNAP): Enzyme that performs the transcription

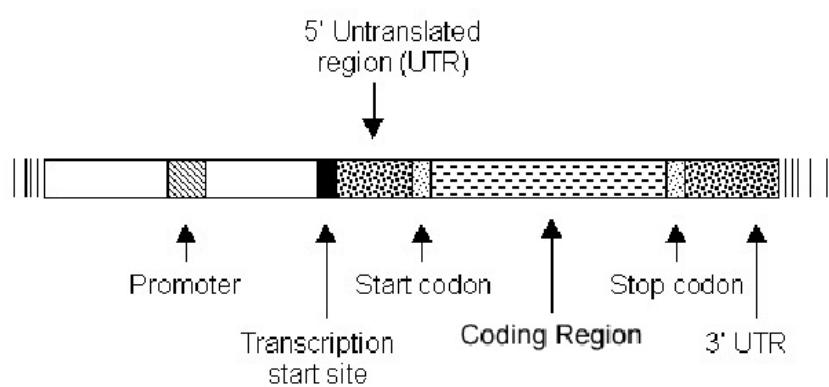
Direction of transcription: 5' end to 3' end

Codon: Are non-overlapping groups of 3 bases. The gene sequence in DNA (and RNA) is composed of codons. Each codon corresponds to a single amino acid

How many possible codons are there? Why?

- There are $4^3 = 64$ possible codons

© Rahul Singh



Start Codon: The codon AUG indicates the beginning of translation

Stop Codon: The three codons that do not instruct the ribosomes to insert a specific amino acid (UAA, UGA, UAG). They cause translation to be terminated

RNAP starts transcription a few bases upstream of the protein coding region by recognizing and binding to the promoter region. It stops a few bases after the end of the protein coding region. Therefore there are regions that are transcribed but not translated. They are called the untranslated regions (UTR)

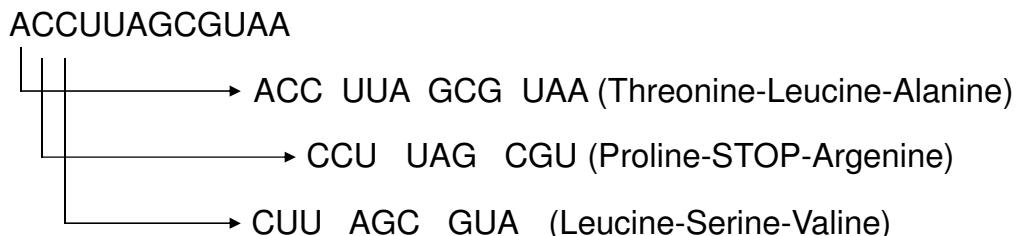
The ribosome scans the mRNA molecule from 5' to the 3' end. On detecting the start codon it starts generating the amino acid sequence as specified by the mRNA code. It stops when it detects a stop codon.

© Rahul Singh

Gene Finding in Prokaryotes

- Prokaryotic cells are cells that do not contain a nucleus
- In prokaryotic cells, most of the DNA sequences are used for coding proteins (e.g. in *H. Influenza* about 70% of the genome is protein coding)
- Open Reading Frames:

-- Reading Frames: Any nucleotide sequence can be interpreted in three possible ways depending on where the starting codon lies



An open reading frame (ORF) is a sequence of codons without a stop codon

© Rahul Singh

Detecting Coding Regions: Long ORFs

- Coding regions can be distinguished from non-coding ones by looking at the frequency of stop codons
- Since there are 3 stop codons out of 64 possible codons, assuming random distribution, we should see a stop codon every $64/3 = 21$ codons.
- However average proteins are much longer (1Kbp or more)

Idea

- Look for long sequences of codons without any stop codon: Look for long ORFs in all three reading frames
- When a stop codon is detected, scan backward and search for the corresponding start codon

Problems:

- Can miss short genes
- There are more ORFs than genes (6.5K ORFs in E. Coli but only 1.1K genes)

© Rahul Singh

Detecting Coding Regions: Codon Frequencies

-Can the frequencies at which codons are known to occur in coding regions be used to detect the coding regions?

Leucine: 6 codons (CUU, CUC, CUA, CUG, UUA, UUG)

Alanine: 4 codons (GCU, GCC, GCA, GCG)

Tryptophan: 1 codon (UGG)

-So, if a uniformly random DNA sequence is translated, these three amino acids should occur in the ratio of 6:4:1

BUT they occur in the ratio of 6.9:6.5:1

- So DNA coding is not random and therefore the codon frequencies could be of help

© Rahul Singh

Model With Successive Codon Independence

-Consider a model where the occurrence of successive codons is independent

- Let the frequency of occurrence of codon abc in the coding region be F_{abc}

- Consider a coding sequence:

$a_1 b_1 c_1 a_2 b_2 c_2 \dots$

$a_n b_n c_n a_{n+1} b_{n+1} c_{n+1}$

- Assume that we don't know the reading frame

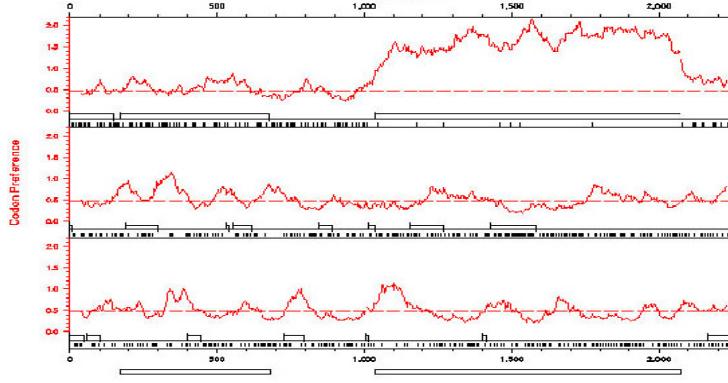
-What is the probability of observing a sequence of n codons in a reading frame starting with $a_1 b_1 c_1$?

$$p_1 = F_{a_1 b_1 c_1} \times F_{a_2 b_2 c_2} \times \dots \times F_{a_n b_n c_n}$$

- The probability of observing a sequence of n codons in the second reading frame is:

$$p_2 = F_{b_1 c_1 a_2} \times F_{b_2 c_2 a_3} \times \dots \times F_{b_n c_n a_{n+1}}$$

© Rahul Singh



Similarly $p_3 = F_{c_1a_2b_2} \times F_{c_2a_3b_3} \times \dots \times F_{c_n a_{n+1} b_{n+1}}$ For the third reading frame

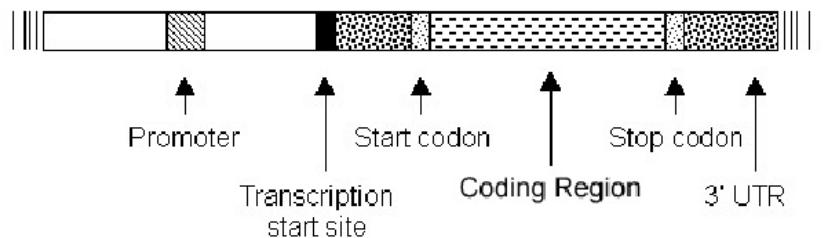
Let P_i be the probability that the i^{th} reading frame is the true coding reading frame.
We can calculate P_i as follows:

- Slide a window of size n along the sequence
- For each start position of the window compute P_i as follows:

$$P_i = \frac{p_i}{p_1 + p_2 + p_3}$$

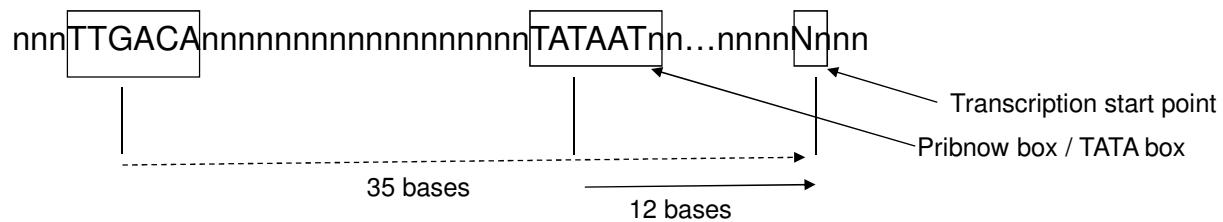
-The above figure shows the log likelihood ($\log(P/(1-P))$) for the three reading frames. Each point is a score for a 25 codon window at it. Actual genes are shown as boxes below. Note the reading frame in the upper plot recognizes the genes
© Rahul Singh

Detecting Promoter Regions



-During transcription, RNAP binds to the promoter and at the stop signal it releases the RNA and detaches itself from the DNA. Thus the promoter acts like an anchor point

Consider the following example from E. Coli around the RNA transcription start point:



- This can be used as a feature to recognize the promoter region

© Rahul Singh

- Unfortunately, there is too much variability in the binding site sequences to use exact strings for identifying promoter regions

- However a pattern search method using frequencies can be devised:

- Construct a table of $f_{B,i}$, where $f_{B,i}$ denotes the frequency of base b in position i of known promoter region suffixes

Position:	1	2	3	4	5	6
A	2	95	26	59	51	1
C	9	2	14	13	20	3
G	10	1	16	15	13	0
T	79	3	44	13	17	96

- Let f_B denote the (background) frequency of base b in the genome

- Our goal is to calculate the likelihood of a given sequence being a TATA-box

© Rahul Singh

Consider a sequence: $S = B_1B_2B_3B_4B_5B_6$

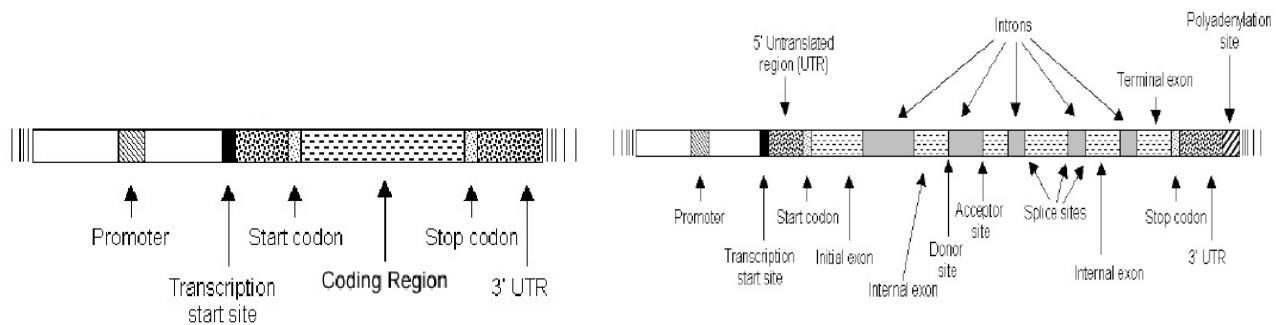
$$P(S \mid S \rightarrow TATAbox) = \prod_{i=1}^6 f_{B_i,i} \quad \text{Similarly, } P(S \mid S \text{ is not a TATAbox}) = \prod_{i=1}^6 f_{B_i}$$

The log-likelihood ratio is: $\log\left(\frac{P(S \mid S \text{ is a promoter})}{P(S \mid S \text{ is not a promoter})}\right)$

$$\log\left(\frac{P(S \mid S \text{ is a promoter})}{P(S \mid S \text{ is not a promoter})}\right) = \log\left(\frac{\prod_{i=1}^6 f_{B_i,i}}{\prod_{i=1}^6 f_{B_i}}\right) = \sum_{i=1}^6 \log\left(\frac{f_{B_i,i}}{f_{B_i}}\right)$$

© Rahul Singh

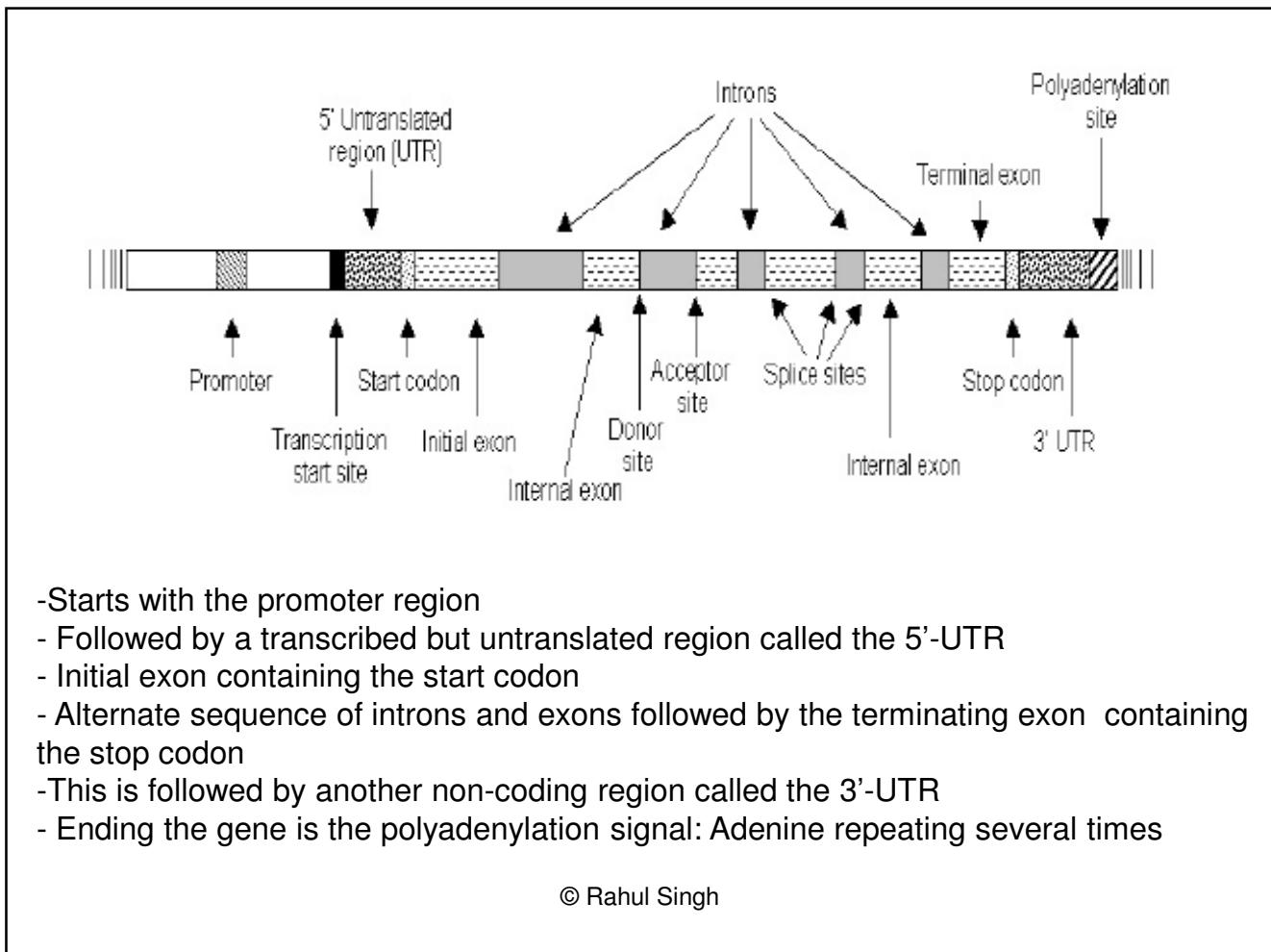
Gene Finding in Eukaryotes



-Gene structure in Eukaryotes is much more complex

- For instance, the protein coding region of the DNA is not continuous. It is composed of alternate stretches of introns and exons.
- During transcription, both introns and exons are transcribed onto the RNA
- The introns are then spliced-out and discarded from the RNA

© Rahul Singh



Alternate Splicing

- One of the most important characteristics of eukaryotic genes is that a single gene can code for a number of different proteins
- The splicing of exons allows for this flexibility
- At least 8 different types of introns have been found in eukaryotic genes however only one of these (conforming to the GU-AG rule) is associated with protein coding genes
- The 5' end of introns are known as 5' splice sites or donor sites and (almost) always begin with the nucleotides GT (GU)
- Similarly the 3' end of the introns are called the 3' splice sites or acceptor sites and end in AG
- The alternating sequence of introns and exons or equivalently the alternating sequence of 5' and 3' splice sites determine how the mRNA is spliced together

— GU ----- AG ----- GU ----- AG -----

© Rahul Singh

Mathematical Formalization of Alternate Splicing

From *Algebraic Statistics for Computational Biology*, Eds. L Pachter and B. Sturmfels

Coding Gene: Is a subsequence of DNA that is transcribed and which has the property that a subset of the transcribed sequence is translated

A gene structure: Is an alternating sequence of exons and introns within a gene. Thus a gene may contain multiple gene structures. These are called alternate splicings

Let a coding gene be represented by a sequence of parentheses:

$()()((())())()$

Here a “(“ represents a donor splice site and
a “)” represents an acceptor splice site

A gene structure is a sequence of open and close parentheses $()()()$

For example, one way to achieve this is

$(())((\textcolor{red}{0})(\textcolor{red}{0}))()$

© Rahul Singh

Markov Sequence Models

- Coding and non-coding regions can be modeled as Markov chains
- Based on known data, two sets of tables are defined: one for coding regions and the other for non-coding regions
- Such models do not take into account any reading frame information (unlike the previous techniques which explicitly modeled the three possible reading frames) and are called homogeneous models
- Consider the problem of modeling splice-sites as an example:
 - Splicing is the process of removing introns and is performed by complexes called spliceosomes (enzymes containing snRNA proteins)
 - Recognition of splice sites has to be precise, otherwise there would be serious consequences for protein consistency
 - So the splice sites must exhibit strong features which allows the spliceosomes to recognize them

© Rahul Singh

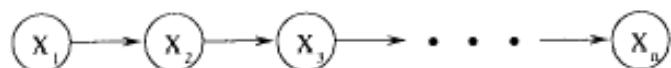
Markov Modeling of Sequences

-Consider a DNA sequence of length N: ACTGGTAGC...

- Associate a discrete random variable X_i with each position i in the sequence

$$X_i \in \{A, C, T, G\}$$

-A first order Markov model assumes that the probability distribution of bases in position i depend only on the distribution in the previous base $i-1$

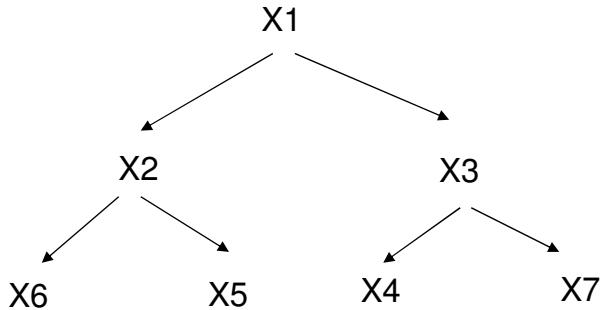


$$P(X_1, X_2, \dots, X_n) = P(X_1).P(X_2 | X_1) \dots P(X_n | X_{n-1})$$

-What if the probability of occurrence at position j depended not on j-1 but on any position i?

- This can be represented as a tree

© Rahul Singh



$$P(X_1, X_2, X_3, X_4, X_5, X_6, X_7) = P(X_1) \cdot P(X_2|X_1) \cdot P(X_3|X_1) \cdot P(X_6|X_2) \cdot P(X_5|X_2) \cdot P(X_4|X_3) \cdot P(X_7|X_3)$$

Building the (tree) model [Cai Et. al, Bioinformatics 2000]

- Let x and y be the set of values taken by the variables i and j . For every i and j compute their mutual information $M(i,j)$

$$M(i, j) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

© Rahul Singh

- Construct a graph G where for every variable X_i there is a corresponding node and the weight of an edge between X_i and X_j equals their mutual information
- Construct a maximum spanning tree on the graph G
- Orient the tree by choosing the variable X_1 (variable associated with the first position) as the root
- For each pair of variables X_i and X_j such that X_i is a parent of X_j , compute the conditional probability $P(X_j|X_i)$ – from emperical frequencies in the data
- Select a data set representing true splice sites and one representing non-splice sites. Using each build a corresponding model (M-true and M-false)
- Given an unknown sequence U , determine the probability that it was generated using M-true and the probability that it was generated using M-false

© Rahul Singh

Other Models

-Generalized HMM

- In a HMM the states of the Markov chain are not directly observable, rather the output of the current state is observable.
- This output is randomly chosen from a finite alphabet according to some probability distribution
- In a GHMM the output of a state may be a string of finite length.
- For a specific state, the length of the output string as well as the output string itself can be randomly chosen per some probability distribution which can change from state to state

GHMM are used in GenScan model for gene identification
(Burge and Karlin, Finding Genes in Genomic DNA, J. Mol. Bio. 1997)

© Rahul Singh

Motif Discovery

© Rahul Singh

Understanding the Problem

Consider the following 8 random sequences of length 32 (*32-mers*)

```
AGTGAGCGACAGGCTAGACTAACTGACTATT  
TATACACGTAGTACCACTCAAGATTGGCTCA  
ATCGGCTAACTAGCTAAGGTCCCTCACGGTCAC  
CGGCGAGTTCCAATGGAGATCCTTAAGGCCAT  
GCGCAAGACCCTAGCTAGAACTTAAGCGACTT  
GACTAACTTCACGGTCCGGCACTTCGGCGAGT  
TACGACTCAAGGCCATACTTCACGGTCAGTTC  
GAATCCACGCTTGTCCCTAGATTCCATATGAC
```

I am going to take a “secret” pattern of length 8 (**ATGCATAG**) and randomly embed it in each of the above sequences

```
AGTGAGCGACAGG ATGCATAG CTAGACTAACTGACTATT  
TATACACGTAGTACCACTCAAGATT ATGCATAG GGCTCA  
ATCGGC ATGCATAG TAACTAGCTAAGGTCCCTCACGGTCAC  
CGGCGAGTTCCA ATGCATAG TGGAGATCCTTAAGGCCAT  
GCGCAAGAC ATGCATAG CCTAGCTAGAACTTAAGCGACTT  
GACTAACTTCACGGTCCGGCACTTCGGCGAGT ATGCATAG  
ATGCATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC  
GAATCCACGCTTGTCC ATGCATAG TTAGATTCCATATGAC
```

© Rahul Singh

```
AGTGAGCGACAGGATGCATAG CTAGACTAACTGACTATT  
TATACACGTAGTACCACTCAAGATTGCATAG GGCTCA  
ATCGGCATGCATAGTAACTAGCTAAGGTCCTCACGGTCAC  
CGGCGAGTTCCAAATGCATAGTGGAGATCCTTAAGGCCAT  
GCGCAAGACATGCATAGCCTAGCTAGAACCTAACGCGACTT  
GACTAACCTCACGGTCCGGCACTTCGGCGAGTATGCATAG  
TATGCATAGACGACTCAAGGCCATACTTCACGGTCAGTTC  
GAATCCACGCTTGTCCATGCATAGTTAGATTCCATATGAC
```



Not knowing what the pattern is and where I have embedded it
in the data, find me this pattern

IDEAS?

© Rahul Singh

One Way to the Solution

AGTGAGCGACAGGATGCATAG CTAGACTAACTGACTATT
TATACACGTAGTACCACTCAAGATTGCATGCATAG GGCTCA
ATCGGCATGCATAGTAACTAGCTAAGGTCCACGGTCAC
CGCGAGTTCCAAATGCATAGTGGAGATCCTTAAGGCCAT
GCGCAAGACATGCATAGCCTAGCTAGAACTTAAGCGACTT
GACTAACTTCACGGTCCGGCACTTCGGCGAGTATGCATAG
TATGCATAGACGACTCAAGGCCATACTTCACGGTCAGTTC
GAATCCACGCTTGTCCATGCATAGTTAGATTCCATATGAC

© Rahul Singh

AGTGAGCGACAGGATGCATAGCTAGACTAACTGACTATT

 ● ● ●
 ↑ ↑

Similarly

TATACACGTAGTACCACTCAAGATTGATGGCTCA

 ● ● ●
 ↑ ↑

Based on the nature of the data I can expect

- Most 8-mers will occur only once or at most twice
- Only the “secret” 8-mer pattern **ATGCATAG** will occur at least 8 times (once for each sequence) and possibly more
- In all probability, this will be the only sequence that will occur with the highest frequency in the data set

© Rahul Singh

What if, my secret pattern was allowed to randomly mutate?

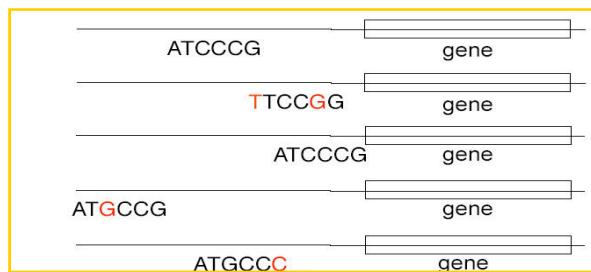
AGTGAGCGACAGG **AcGCATAG** CTAGACTAACTGACTATT
TATACACGTAGTACCACTCAAGATT **tTGCATAG** GGCTCA
ATCGGC **ATGCATgG** TAACTAGCTAAGGTCCTCACGGTCAC
CGGCGAGTTCCAA **ATaCATAG** TGGAGATCCTTAAGGCCAT
GCGCAAGAC **ATGCcTAG** CCTAGCTAGAACTTAAGCGACTT
GACTAACTTCACGGTCCGGCACTTCGGCGAGT **AcGCATAG**
TATGtATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC
GAATCCACGCTTGTCC **cTGCATAG** TTAGATTCCATATGAC

Now, the previous strategy does not work, since the 8-mer frequency will not reveal the pattern.

In fact the string **ATGCATAG** does not even occur in the data set above!

© Rahul Singh

The Biological Relevance of Our Pattern Finding Problem



- Every gene contains a regulatory region stretching 100 – 1000 bp upstream from the transcription start site
- Located within the regulatory regions are *transcription factor binding sites* also known as **motifs**, specific for a given transcription factor (a regulatory protein)
- Transcription factors influence gene expression by binding to a specific location in the regulatory region of the corresponding gene
 - So finding the same motif in the regulatory regions of multiple genes would suggest a regulatory relationship amongst those genes
 - A motif can be located anywhere in the regulatory region
 - Motifs can vary slightly across different regulatory regions, since non-essential bases can mutate

© Rahul Singh

The Challenge

AGTGAGCGACAGG **AcGCATAG** CTAGACTAACTGACTATT
TATACACGTAGTACCACTCAAGATT **tTGCATAG** GGCTCA
ATCGGC **ATGCATgG** TAACTAGCTAAGGTCCTCACGGTCAC
CGGCGAGTTCCAA **ATaCATAG** TGGAGATCCTTAAGGCCAT
GCGCAAGAC **ATGCcTAG** CCTAGCTAGAACTTAAGCGACTT
GACTAACTTCACGGTCCGGCACTTCAGGTAGT **AcGCATAG**
TATGtATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC
GAATCCACGCTTGTCC **cTGCATAG** TTAGATTCCATATGAC

- We do not know the motif sequence
- We do not know where it is located relative to the starting point of the gene
- Motifs can have slight variability
- How to distinguish “true” motifs from “random” motifs?

© Rahul Singh

Formalizing the Notion of a Motif (l -mer)

$n = 40$ →

ACTGAGCGAGG **AcGCATAG** CTAGACTAACTGACTATT
TATAACCGTAGTACCA ACTCAAGATT **tTGCATAG** GGCTCA
ATCCGGC **ATG**CATgG TAACTAGCTAAGGTCCCTCACGGTCAC
CGGCGAGTTCAA **AtaCATAG** TGGAGATCCCTTAAGGCCAT
GCGCAAGAC **ATGCcTAG** CCTAGCTAGAACTT **AAGCGACTT**
GACTAACTTACGGTCCGGC **ACTTCACGGTCAGTTC** **AcGCATAG**
TATGtATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC
GAATCCACCGCTTGTC **cTGCATAG** TTAGATTCCATATGAC

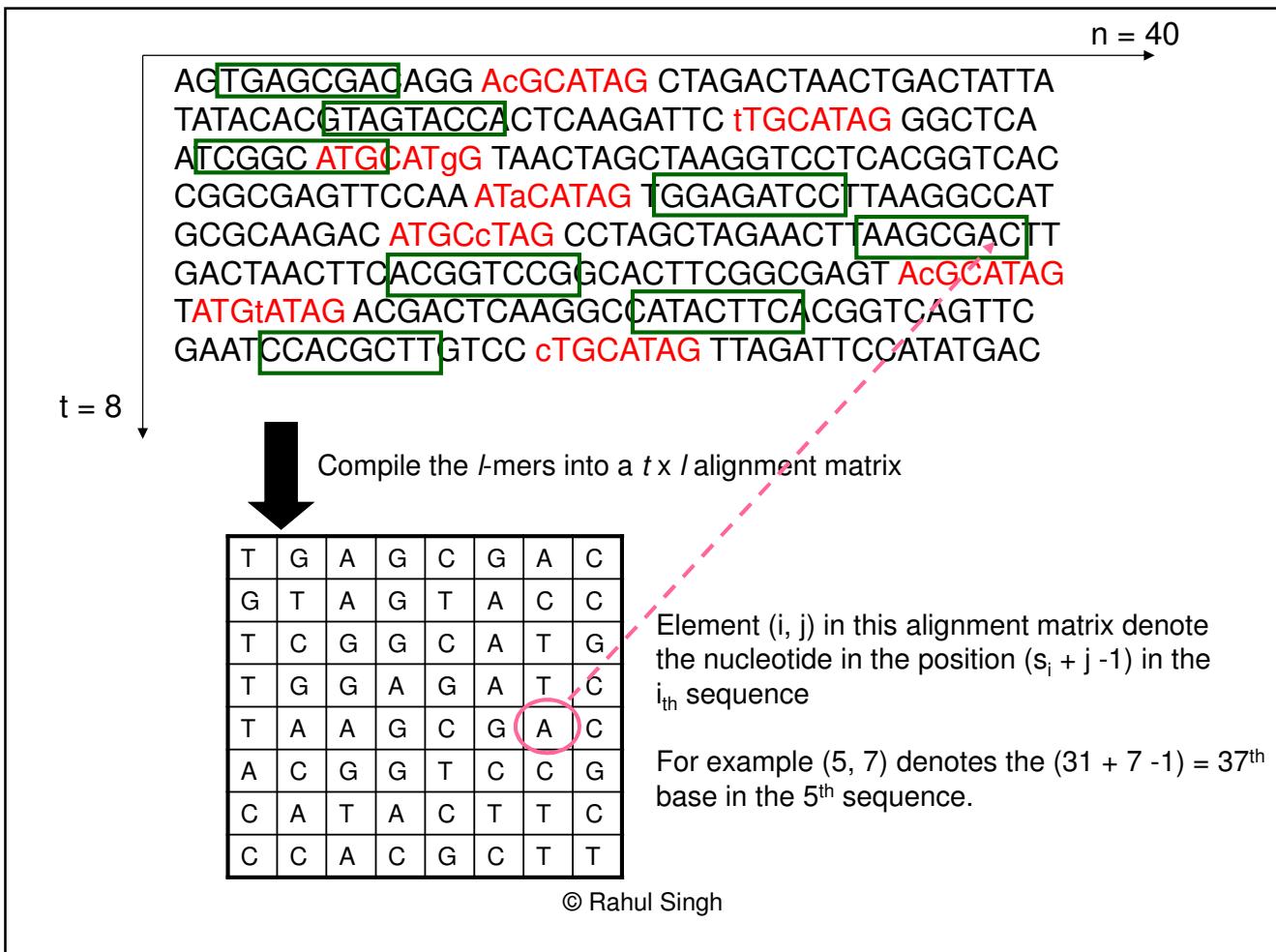
↓
 $t = 8$

Consider a set of t DNA sequences, each having n nucleotides. We will consider 8-mers

Select one position s_i in each of these sequences forming an array $s = (s_1, s_2, \dots, s_t)$, such that $1 \leq s_i \leq n - l + 1$. The positions s_i will denote the start of our l -mers

For example, in the above case we have: $s = (3, 8, 2, 22, 31, 11, 22, 5)$

© Rahul Singh



The Profile Matrix

Starting positions

$s = (3, 8, 2, 22, 31, 11, 22, 5)$

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

Profile Matrix ($4 \times l$)

$(i, j)^{\text{th}}$ element is the frequency of base i in column j of the alignment matrix

A	1	2	4	2	0	3	2	0
T	4	1	1	0	2	1	4	1
G	1	2	3	5	2	2	0	2
C	2	3	0	1	4	2	2	5

Consensus string

T C A G C A T C

1. The profile matrix shows the variability of nucleotide composition at each position, for a specific choice of l -mers. For example, positions 4 and 8 are well conserved, while position 2 is not

2. The most popular base in each position of the alignment matrix is put at the corresponding position in the consensus string

By varying the starting positions in s , different profile matrices can be constructed for a given data set

© Rahul Singh

The Consensus Score

Let X be a set of sequences and $P(s)$ denote a profile matrix, corresponding to the starting position set s . Let $M_{P(s)}(j)$ be the largest count in column j of $P(s)$.

The consensus score, $Score(s, X)$, is defined as: $\sum_{j=1}^l M_{P(s)}(j)$

A	1	2	4	2	0	3	2	0
T	4	1	1	0	2	1	4	1
G	1	2	3	5	2	2	0	2
C	2	3	0	1	4	2	2	5

$$Score(s) = 4 + 3 + 4 + 5 + 4 + 3 + 4 + 5 = 32$$

For the best possible consensus, $Score(s, X) =$

For the worst possible consensus, $Score(s, X) =$

Motif-finding problem: Given a set of sequences X and the length l of the pattern, find s (the set of starting positions), such that $Score(s, X)$ is maximized

© Rahul Singh

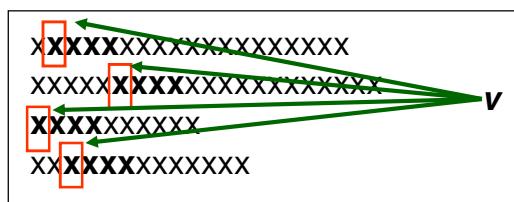
Another Perspective: The Median String Problem

Consider two strings (l -mers) v and w :

$$v = A T T G T C$$

$$w = A C T C T C$$

The *Hamming distance* $d_H(v, w)$, between v and w , is the number of positions at which the two strings differ. In this case the hamming distance is 2



Let v be an l -mer. Let $s = (s_1, s_2, \dots, s_t)$ be a set of starting positions

$$d_H(v, s) = \sum_{i=1}^t d_H(v, s_i)$$

Now, let's drop the assumption that we know s and introduce the idea of the *TotalDistance*(v, X), where X is a set of sequences

$$\text{TotalDistance}(v, X) = \min_s (d_H(v, s)) \quad \text{Where, } s \text{ is a set of starting positions of } l\text{-mers on } X$$

© Rahul Singh

Note: Finding $\text{TotalDistance}(v, X)$ is very easy ! Why?

Median string problem: Given a set of DNA sequences X and l the length of the pattern to be found, find a l -mer v such that $\text{TotalDistance}(v, X)$ is minimized.

Note: The task is to *find* the l -mer v . Since there are 4-bases (A, C, T, G), there are 4^l possible strings of size l , which we would need to consider and align with the strings in X .

This is an NP-Hard problem !

© Rahul Singh

Relationship Between Motif Finding and Median String

Given a set of sequences X. Let s be a set of starting positions, with the consensus score, $Score(s, X)$ and let w be the consensus string of the corresponding profile.

Then

$$d_H(w, s) = l \times t - Score(s)$$

For our example

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

Consensus string T C A G C A T C

From the formula: $d_H(w, s) = 8 \times 8 - 32 = 32$

Computing the Hamming distance for the data and the consensus string (positions differing from the consensus string are shown in red) = 32

Go over this at home and convince yourself of the intuition behind this observation

© Rahul Singh

Search Trees: Enumerating the Possibilities

In the motif finding version of the problem we need to consider all the starting positions s . That is for t sequences, we have to look at:

$(1, 1, \dots, 1), (1, 1, \dots, 2), \dots, (n - l + 1, n - l + 1, \dots, n - l + 1)$

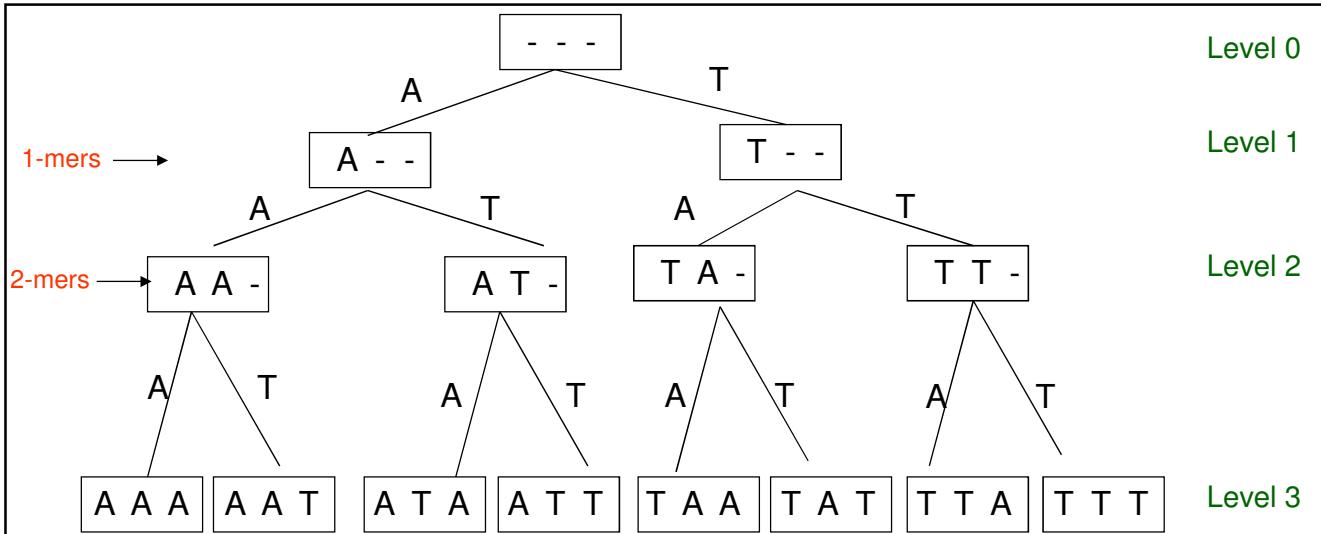
- There are $(n - l + 1)^t$ such positions.

In the median string version of the problem, we have to consider all the 4^l l -mers:

$(A, A, \dots, A, A), (A, A, \dots, A, T), (A, A, \dots, A, G), \dots, (C, C, \dots, C, C)$

Such exhaustive enumeration of all patterns can be done using a tree-structure.

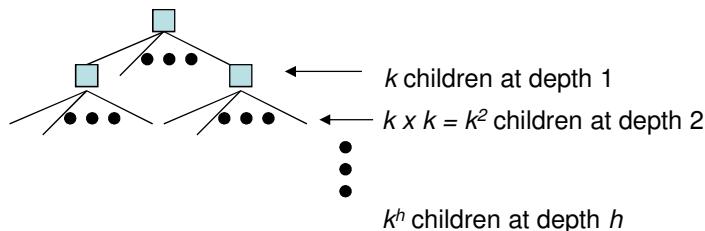
Consider a 2 letter alphabet $\{A, T\}$. Let us enumerate all 3-mers using a tree



- In general we will be dealing with k -ary trees, where all internal nodes have degree k
- For the motif finding problem, we would construct a tree with t levels (there are t sequences) and $n - l + 1$ children per internal node of the tree
- For the median string problem, we would construct a tree with l levels (l is the length of the consensus string) and 4 (A, T, G, C) children per internal node of the tree

© Rahul Singh

The Tree Sizes



Motif finding problem

AGTGAGCGACAGG **AcGCATAG** CTAGACTAACTGACTATT
 TATACACGTAGTACCACTCAAGATT **cTGCATAG** GGCTCA
 ATCGGC **ATGCATgg** TAACTAGCTAAGGTCTCACGGTCAC
 CGCGCAGTTCAA **AtaCATAG** TGGAGATCCTAACGCCAT
 GCGCAAGAC **ATGCcTAG** CCTAGCTAGAACTTAAGCGACTT
 GACTAACTTCACGGTCCGGCACTTCGGCGAGT **AcGCATAG**
TATGiATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC
 GAATCCACGCTTGCC **cTGCATAG** TTAGATTCATATGAC

The k -ary tree will have t (=8) levels

Further, $k = (n - t + 1) = (40 - 8 + 1 = 33)$

Therefore, the number of leaves of the tree at depth t will be: $k^t = 33^8 = 1.4 \times 10^{12}$

Median String problem

$v \in \{A, T, G, C\}^l$

AGTGAGCGACAGG **AcGCATAG** CTAGACTAACTGACTATT
 TATACACGTAGTACCACTCAAGATT **cTGCATAG** GGCTCA
 ATCGGC **ATGCATgG** TAACTAGCTAAGGTCTCACGGTCAC
 CGGCAGTTCAA **AtaCATAG** TGGAGATCCTAACGCCAT
 GCGCAAGAC **ATGCcTAG** CCTAGCTAGAACTTAAGCGACTT
 GACTAACTTCACGGTCCGGCACTTCGGCGAGT **AcGCATAG**
TATGiATAG ACGACTCAAGGCCATACTTCACGGTCAGTTC
 GAATCCACGCTTGCC **cTGCATAG** TTAGATTCATATGAC

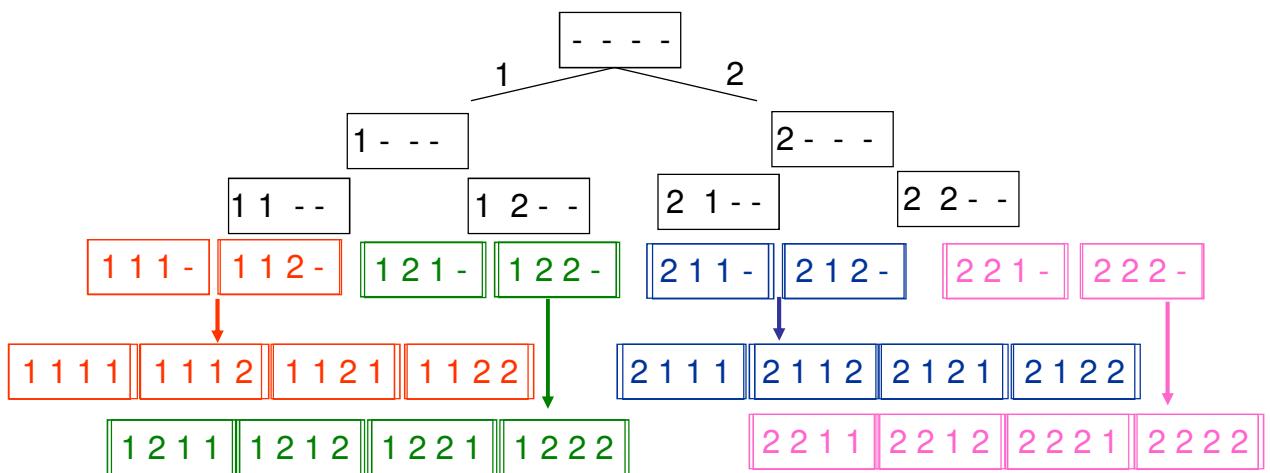
The k -ary tree will have l (=8) levels

Further, $k = 4$ ($|\{A, T, G, C\}|$)

Therefore, the number of leaves of the tree at depth l will be: $k^l = 4^8 = 65536$

© Rahul Singh

Motif Finding: Searching the Tree



Consider motifs of length 3 ($l = 3$), 4 sequences ($t = 4$), each having 4 nucleotides ($n = 4$)

BruteForceSearch (T, X)

```

BestScore = 0; BestMotif = Nil // Initialization
For s = (1 1 1 1) to (2 2 2 2) // Enumerate all possible starting positions; list the leaves
  do if Score(s, X) > BestScore
    BestScore = Score(s, X)
    BestMotif = s
  return(BestMotif);

```

© Rahul Singh

Partial Consensus Score

Consider a specific set of starting positions $s = (s_1, s_2, \dots, s_t)$

We define the partial consensus $s_p = (s_1, s_2, \dots, s_i, -, -, \dots, -)$ to be the alignment obtained for the first i sequences, corresponding to the starting positions s_1, s_2, \dots, s_i . Here a “-” indicates that no value has been chosen for that position.

Corresponding to the partial consensus, is the partial consensus score $Score_p(s_p)$, which is the score obtained from the $i \times l$ / alignment matrix.

Example

$n = 40$

ACTGAGCGADAGG	AcGCATAG	CTAGACTAACTGACTATT		
TATACACCGTAGTACCA	CTCAAGATT	tTGCATAG	GGCTCA	
ATCGGC ATG	ATG	TAACTAGCTAAGGTCCTCACGGTCAC		
CGGCGAGTTCCAA	ATaCATAG	TGGAGATCC	TTAAGGCCAT	
GCGCAAGAC	ATGCcTAG	CCTAGCTAGAACTT	AAGCGACTT	
GACTAACTT	OACGGTCCGG	GCAC	TTCGGCGAGT	AcGCATAG
TATGtATAG	ACGACTCAAGGCC	CATACTTC	ACGGTCAGTTC	
GAATCCACGCTT	GTCC	cTGCATAG	TTAGATTCCATATGAC	

$t = 8$

© Rahul Singh

The Partial Consensus Score and the Best Score

BestScore | s_p ?

That is, what is the best possible score given a specific partial consensus score?

$$\text{Best Score} = \text{Partial score} + (t - i) \times l$$

This provides us with the means to improve BruteForceSearch(T, X) as follows:

- During the enumeration of the starting positions in s , we compute the partial score $\text{Score}_p(s_p)$ and the estimate of the best possible score, given the specific partial consensus s_p
- If our estimate of the best possible score turns out to be less than the current best score, we do not need to explore the remaining part of the search space

This is the classical Branch-and-Bound formulation we had looked at earlier

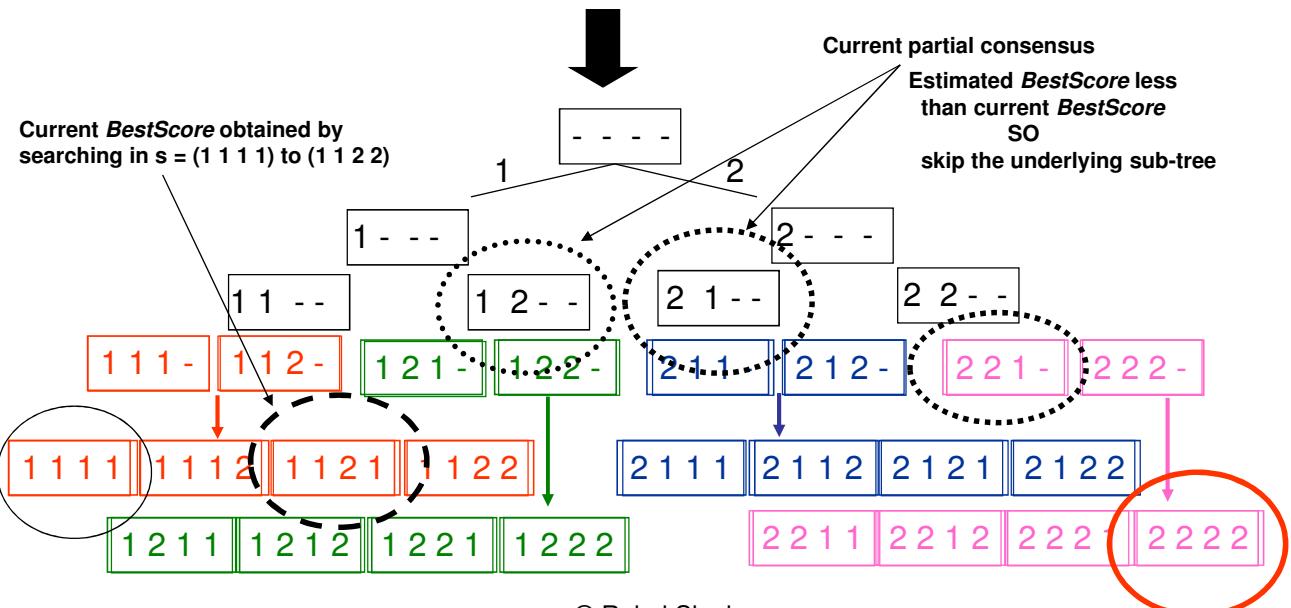
© Rahul Singh

Example

```

BruteForceSearch (T, X)
    BestScore = 0; BestMotif = Nil
    For s = (1 1 1 1) to (2 2 2 2)
        do if Score(s, X) > BestScore
            BestScore = Score(s, X)
            BestMotif = s
    return(BestMotif);

```



More on Branch-and-Bound

It is easily possible to construct scenarios where we end up traversing the entire search space

- For example, the current *BestScore* is always improved during the estimation

Therefore, the worst case complexity is still exponential

You can analogously construct a branch-and-bound strategy for the median string problem
(try this at home)

Recent examples of branch-and-bound strategies can be found in Eskin and Pevzner (2002) and Marsan and Sagot, 2000

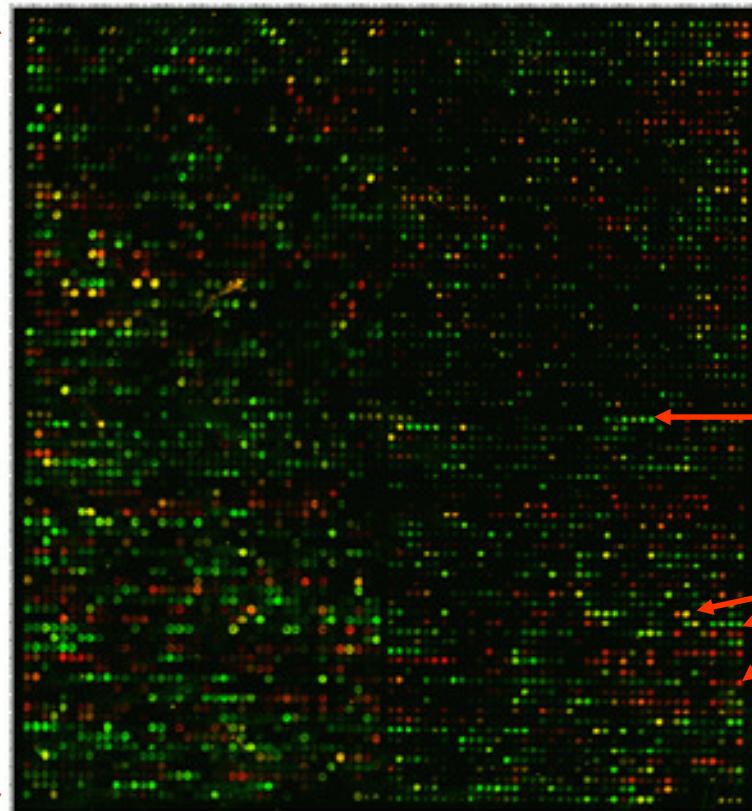
© Rahul Singh

Microarray Expression Analysis

© Rahul Singh

Microarray

Can be
1-inch
or less



© Rahul Singh

History

Idea from a technique called “Southern blotting” where fragments are attached to a surface and then probed with a known gene/fragment.

Use of miniature microarrays for gene expression profiling presented in 1995
Schena M., Shalon D., Davis RW, and Brown P, “Quantitative Monitoring of Gene Expression Pattern with a Complementary DNA Microarray”, Science 270 (5235), pp. 467-470

S. cerevisiae genome on a microarray came out in 1997

© Rahul Singh

The Central Dogma and the Measurement point



-What makes cells differ?

- Which genes are expressed
- How much of each gene is expressed

-Genes can be expressed at different levels (e.g. mRNA, Proteins)

- One way to measure gene expression is to look at the amount (number) of mRNA/cell

© Rahul Singh

Traditional Biology Vs Microarrays (Systems Biology)

Traditional

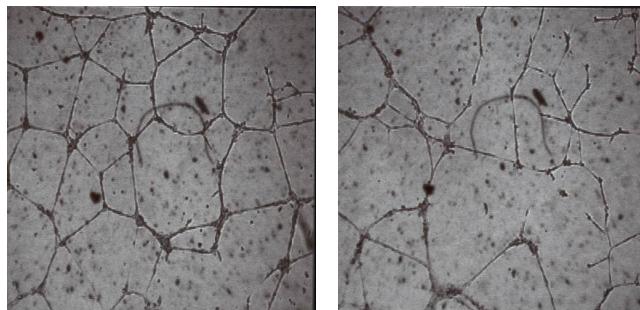
- Look at the genes that are differentially expressed
 - Study the function of these genes
 - Find out which of these genes interact with the genes you are interested in or already know about
- ONE GENE AT A TIME
- Slow
 - Misses systemic effects

Systems Biology with Microarrays

- Microarrays allow us to measure the concentration of mRNA (Microarray is about gene expression) and can screen an entire genome at a time
- Can help us find single genes as well as groups of genes that are differentially expressed

© Rahul Singh

Applications of Microarrays



Tumor

Tumor in regression/non-tumor



1. Can find all the genes that are differentially expressed
2. Functional analysis of genes
3. Identification of regulatory processes
4. Discovery of new drugs
5. Genotype (population) targeted drugs

•
•
•

© Rahul Singh



© Rahul Singh

Relation of Gene Expression to PPI

- How is gene expression related to protein-protein interactions

In Yeast **weak** – Jensen et al. Genome Res, 2002

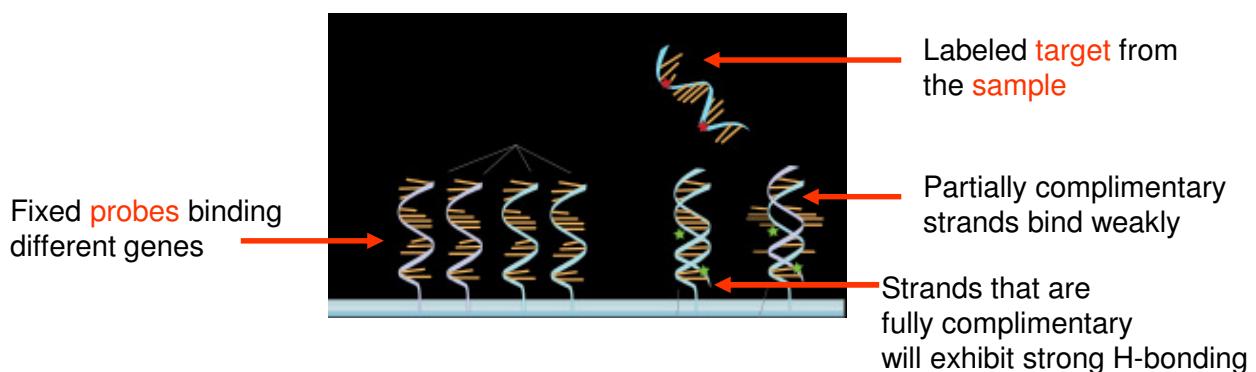
In *E. coli* **strong** – Bhardwaj & Lu, Bioinformatics 2005

In Human, Mouse, and Yeast – **weak** (slightly better than that of random pairs)

- Bhardwaj & Lu, Bioinformatics 2005

© Rahul Singh

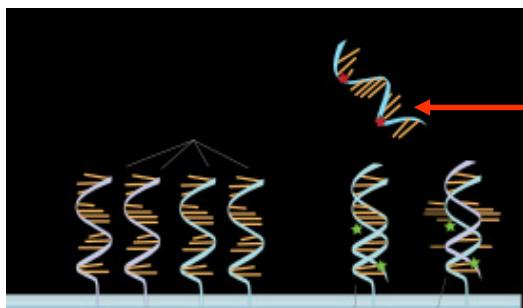
Principles of the Method



- Complimentary hybridization is the basis of mRNA measurement
 - A - - T
 - G - - C
- DNA complementary to genes of interest can be generated and accurately placed on solid surfaces at predefined positions
- High number of complimentary base pairs implies tighter non-covalent bonding between the two nucleotide sequences (strands). After non-specific bonding sequences are washed off, only the strongly paired strands will remain hybridized.

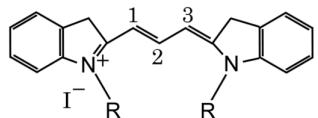
© Rahul Singh

Principles underlying observation/quantification of the binding

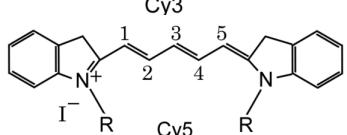


The target can be fluorescently labeled

Two commonly used labels are Cy3 and Cy5 of the Cyanine dye family.



Cy3 fluoresces in the green region (~550-570 nm)



Cy5 is fluorescent in the red region (~650-670 nm)

The dyes (Cy3/Cy5) are designed to link to both nucleic acids or proteins. A label every 60 bases is considered good labeling.

© Rahul Singh

The Fluorescent signal depends on the strength of the hybridization as given by the number of paired bases, the hybridization conditions, and washing.

After washing and keeping other conditions same, the signal strength from a spot (feature), depends upon the amount of target sample binding to the probes present on that spot.

Overall Structure of the technology

- DNA complementary to genes of interest is generated and accurately placed on solid surfaces at predefined positions
- DNA from samples is eluted over the surface. The complimentary bases bind
- Presence of bound DNA is detected by fluorescence following excitation by laser

The Technology

Technological differences between different ways of implementing this idea lie in

1. How the DNA sequences are laid down: Spotting or Photolithography
2. Length of DNA sequences that are laid down: Complete sequences or series of fragments

For example:

-Stanford/Synteni Chips use a robot to spot slides at precise locations with complete gene or EST sequences

-Affymetrix chips use a photo-lithography technique similar to what is used in semiconductor design

© Rahul Singh

The Technology

- Two variants of chips are common: **cDNA arrays** and **Oligonucleotide arrays**

→ Relatively long DNA molecules are immobilized by high-speed robots on glass or silicon chips. Such arrays are used for large scale screening and expression studies

cDNA (complimentary DNA) is a DNA strand synthesized using reverse transcriptase enzyme. This makes a DNA sequence that is complimentary to a RNA template. The cDNA is next amplified using PCR. Each cDNA can be 500 – 2000 bases long. It thus contains a significant fraction of a gene sequence (but not the complete sequence in most cases). Hybridization is therefore much more specific. Typically one spot on a cDNA array corresponds to one gene.

→ Fabricated by in-situ light directed chemical synthesis. Those with short nucleic acid sequences (oligonucleotides) are useful for gene discovery and expression monitoring. (Short = around 25 (25-mer) or fewer nuclueotides long)

Sequence in each spot is carefully chosen in advance with reference to the genome. Each oligo should hybridize to a specific gene sequence. However, cross hybridization is possible in genes with related sequences. Therefore, several separate oligo-sequences are chosen for each gene and expression of a gene in a sample is inferred only if hybridization occurs with most of them.

Both types of microarrays are exposed to labeled samples, hybridized, and complementary sequences determined

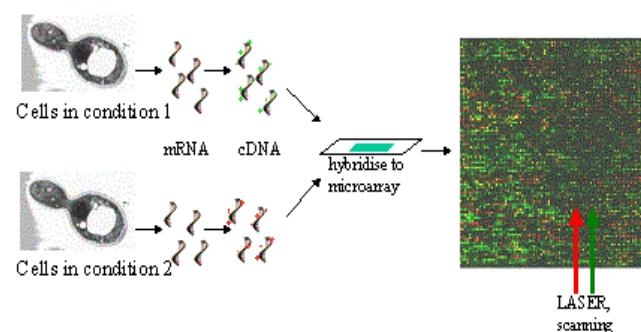
© Rahul Singh

Steps in a Microarrays Experiment

Goal: Compare gene expression levels in two different samples, e.g. same cell type in healthy and diseased state

Step 1: Extract the mRNA from the cells in two different conditions and label with different fluorescent dyes; e.g. green dye for cells in condition 1 and red dye for cells in condition 2

Step 2: Both extracts are washed over the microarray. Labeled cDNA from the extracts hybridize to their complementary sequences on the spots: longer the complementary sequences, stronger the attraction



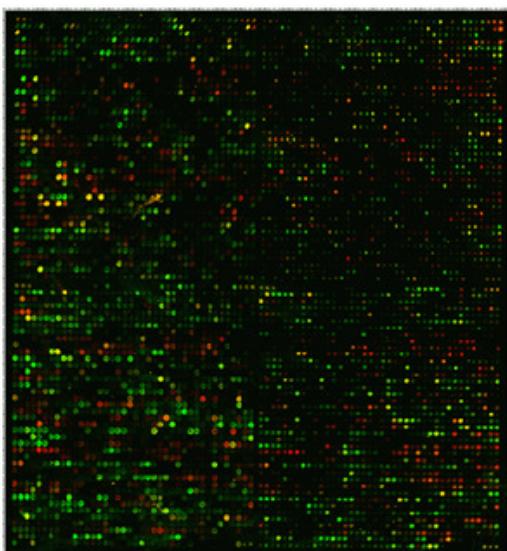
Step 3: The dyes enable the amount of samples bound to be measured by level of fluorescence emitted when excited by a laser. Note, typically two 16-bit image files are produced corresponding to each of the two conditions.

- If RNA from condition 1 is more, the spot will be green.
- If condition RNA from 2 is more, the spot will be red.
- If both are equal, spot will be yellow
- If none, then black

© Rahul Singh

Genetic Information Extraction

Given

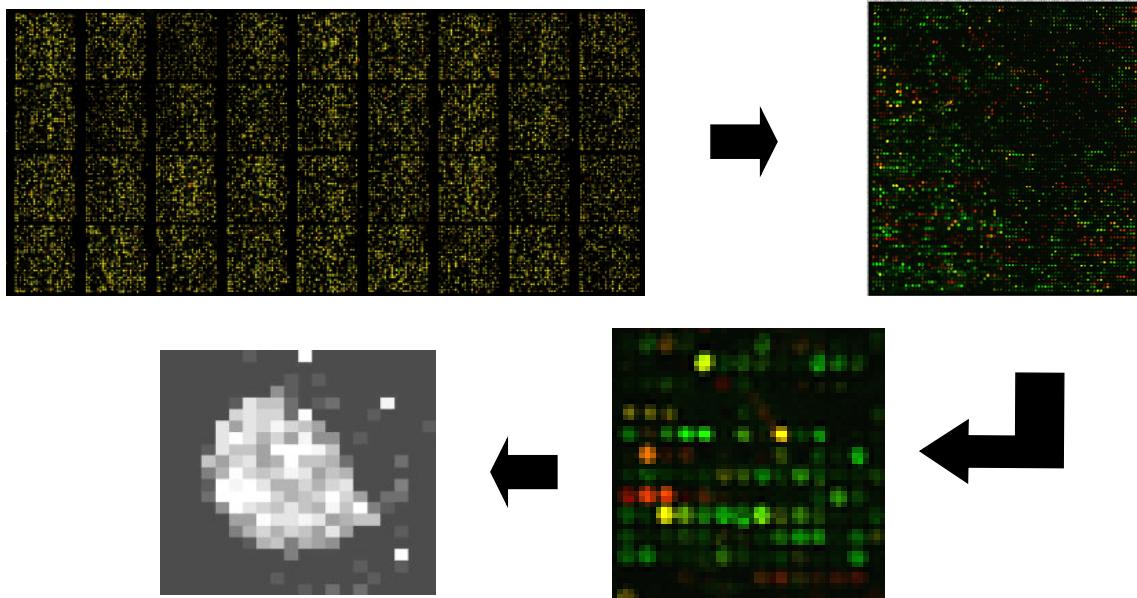


Required

1. Locate the high-intensity spots in the image as these represent the differentially expressed genes. This step is called *microarray image registration and segmentation*
2. Estimate and remove the local background noise. This is called *background correction*. It can/is often done as part of image segmentation
3. Quantification of the differential expression

© Rahul Singh

Image Registration and Segmentation



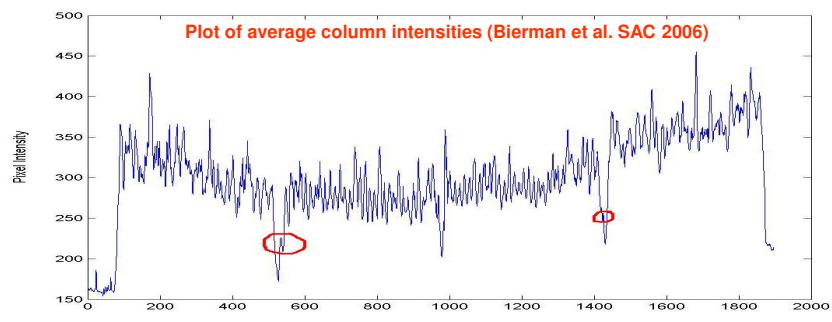
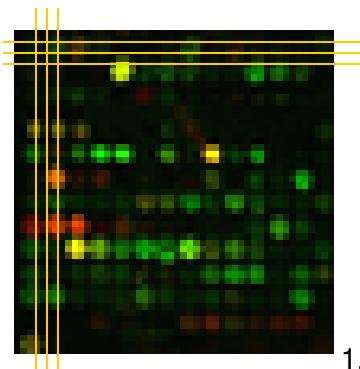
- Registration is done to identify the approximate centers or the grid location of the spots due to possible shifting or skewing
- The grid-like geometry of the spotting process can help in this task. In the first step, a grid is overlaid on the microarray image

© Rahul Singh

Gridding

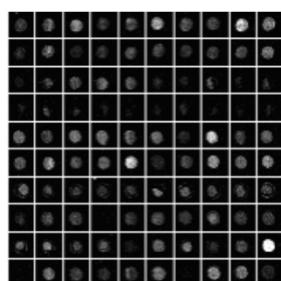
- Can be done manually, semi-automatically, and automatically

Automatic gridding (MicroZip, Lonardi and Luo, CSB 2004)



1. Compute the average intensities row-by-row and column-by-column
2. Use low-pass filter to remove noise. For sub-grids, the smoothing window is 25 pixels, for spots it is 4 pixels
3. The minima of the intensities are treated as first approximation of the cell boundaries P
4. If the pixel intensities in a window of 25 pixels around a minimum show little variation, the minimum is declared a point on the grid line
5. The first approximation of the cell boundaries can have errors due to
 - (a) extra lines caused by high-intensity noise
 - (b) missing lines due to low intensity spots in a row/column

© Rahul Singh



Automatic Gridding in MicroZip

6. Improving the first approximation grid positions

Parameters: $\alpha = 0.9$, $\beta = 1.2$, and $\gamma = 0.01$

real REFINE_GRID(P)

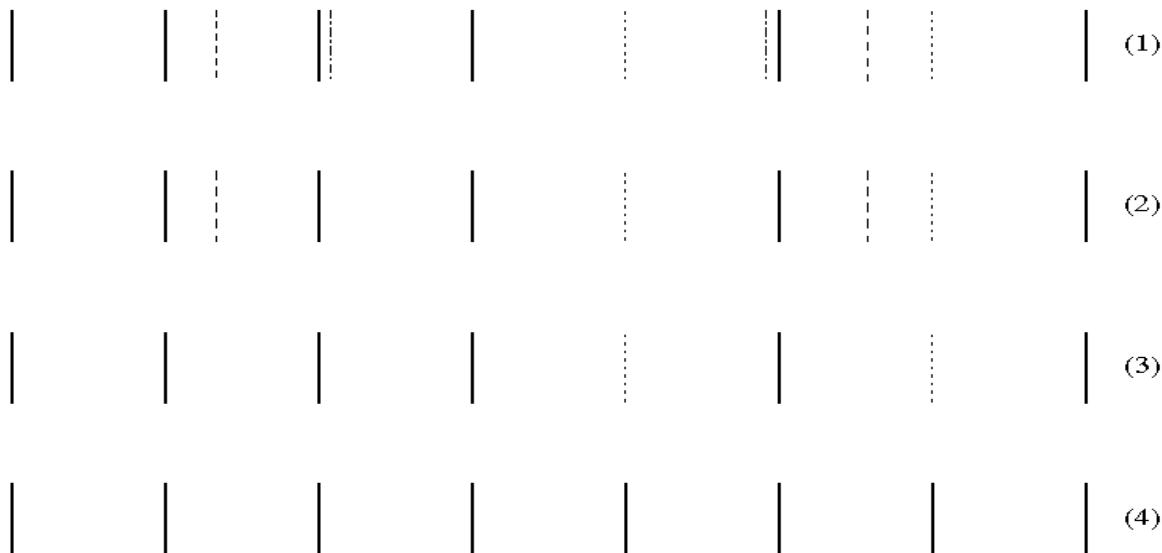
- 1 $R_{new} \leftarrow$ average distance between adjacent cells on P
- 2 **repeat**
- 3 $R \leftarrow R_{new}$
- 4 $P' \leftarrow P$
- 5 **repeat**
- 6 $d \leftarrow$ distance between a pair of adjacent lines
- 7 **if** $d < \alpha R$ **then** remove a line from P'
- 8 **if** $d > \beta R$ **then** add a line to P'
- 9 **until** all lines have been considered
- 10 $R_{new} \leftarrow$ average distance between adjacent cells on P'
- 11 **until** $|R_{new} - R| \leq \gamma R$
- 12 **return** R

© Rahul Singh

Example of Grid Refinement in MicroZip

— Correct line
- - - Noisy line
- - - Extra line
- - - - Missing line

- 1: Input
- 2: Lines which are very close are merged
- 3: Incorrect lines are removed
- 4: Missing lines are added back

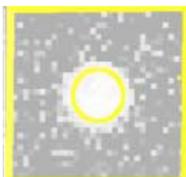


© Rahul Singh

Methods for Spot Detection



- Once the gridding is done, the center of each grid can be considered to be the idealized spot center
- Various techniques can be used to find the actual spot:



Fixed Circle: inscribe a circle of fixed radius to each grid region and consider all pixels inside the circle to belong to the spot



Iterative Circle growing (used, e.g. in MicroZip): Begin by obtaining an average background intensity value B (e.g. average of the intensity minima found during gridding). Next, inside each cell, classify pixels as spot (intensity greater than B) or background. Compute the spot center as the average x and y coordinates of all the pixels classified to be inside the spot. Using this center, fit circles of increasing radii to the spot, checking at each radius if the average intensity of the pixels in the purported spot remains greater than φB , where, $\varphi=0.01$. Note, in this method the spot may be found off-center within the rectangular grid region

More refined methods, such as seeded region growing, which uses the intrinsic connectivity of pixels are also possible. An advantage of simpler methods, such as various types of circle fitting, is that the spot can be encoded by using only 3 values

© Rahul Singh

Quantification of the Spots

Common statistic: log ratio of spot intensities. Note that the experiment produces two intensity Images (green = control and red = treatment)

Let R_i denote the red scan pixels
 G_i denote the green scan pixels

The spot differential expression level, $\log_2 \mu_R/\mu_G$ is defined as:

$$\log_2 \frac{\mu_R}{\mu_G} = \log_2 \frac{\frac{1}{s} \sum_{R_i \in \text{spot}} R_i - \sigma_R}{\frac{1}{s} \sum_{G_i \in \text{spot}} G_i - \sigma_G}$$

where S is the number of pixels in the spot and σ_R and σ_G are the estimates of the local background

Other measurements include: spot variances, spot shape, and product intensities $\log_2 (\text{Sqrt}(\mu_R * \mu_G))$

© Rahul Singh

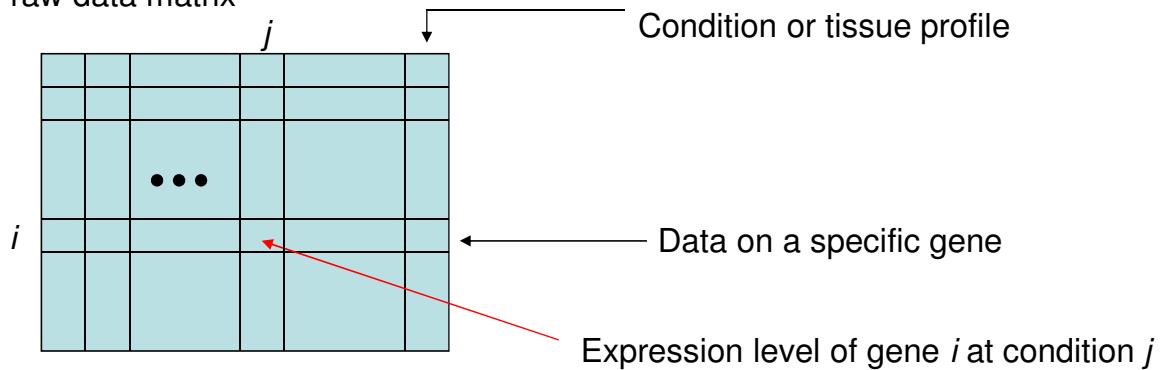
Observations on Spot Quantification

- The intensities of spots can vary due to biological factors or experimental conditions. Normalization of data is important to ensure that systematic variation in the data does not dominate over chance variation
- Use of intensity ratios: R_i/G_i (for spot i) eliminates a lot of spot-to-spot variability
- If the ratio is > 1 : gene is up-regulated, if < 1 : then gene is down-regulated
- Typically the ratios are dealt on a \log_2 scale: $M_i = \log(R_i/G_i)$. This means that the expression doubles if $M= 1$ and halves if $M = -1$

© Rahul Singh

Vector Space Representation of Expression Data for Analysis

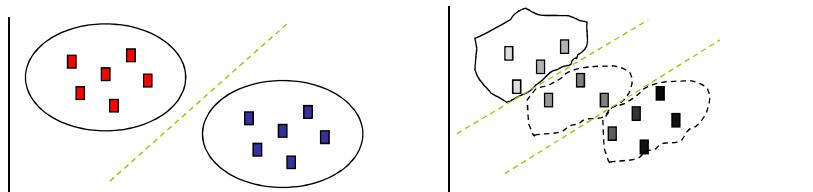
-The expression data can be represented as a matrix of real-valued data called the raw data matrix



- Typically, the number of genes is much more than the number of conditions
- The expression pattern of gene i is the i^{th} row of the above matrix
- The expression pattern of condition j is the j^{th} column of the matrix
- Often the information in the raw data matrix is used to generate a gene similarity matrix

© Rahul Singh

Goals of Analyzing Gene Expression Profiles



- If an unknown gene has an expression pattern similar to that of a known gene, then it may have a similar function (**Identification of gene function**)
- Tissues with similar disease should have similar expression patterns (**Identification of disease profiles**)
- Co-expressed genes are likely co-regulated (**Researching regulatory mechanisms**)

Basic Idea

- Partition the genes into distinct sets such that:
 - a) Genes assigned to each set are similar
 - b) Genes assigned to different sets are dissimilar

- Criteria (a) is reflected as the homogeneity of a set (cluster): Higher the homogeneity, greater the similarity between the genes
- Criteria (b) is reflected as the separation between the sets (clusters)

All this requires a notion of a distance between the genes

© Rahul Singh

From Vector Space Representation to Gene Similarity

- Since each row in the expression table (matrix) represents a gene, this data may be used to compute similarities between them.
- Similarity computations may be used to cluster genes, classify new genes, etc.
- How to compute the similarity amongst genes?

Idea: Define a “distance” between the genes. The smaller the distance, the more similar the genes

The Euclidean Measure of Gene Similarity

-Use the Euclidean measure to determine the similarity of genes.

For example: if we have two points P(x₁, x₂) and Q (y₁, y₂), the Euclidean distance between them is:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Note: You can think of genes as points in a multi-dimensional “condition space” (Vector Space Representation)

Cond	1	2	3	4	5	6	
gene-1	1	1	0	0	0	0	→ (1, 1, 0, 0, 0, 0)
gene-2	0	1	1	0	0	0	→ (0, 1, 1, 0, 0, 0)
gene-3	1	1	0	1	1	0	
gene-4	0	0	0	1	0	1	→ (0, 0, 0, 1, 0, 1)

In the general case, denoting the genes as D1, D2, ... and the expression values as F_{ij} (for gene i in condition j):

$$\left(\begin{array}{cccc} & C_1 & C_2 & \dots & C_n \\ D_1 & F_{1,1} & F_{1,2} & \dots & F_{1,n} \\ D_2 & F_{2,1} & F_{2,2} & \dots & F_{2,n} \\ \vdots & & & & \\ D_m & F_{m,1} & F_{m,2} & \dots & F_{m,n} \end{array} \right) \rightarrow \begin{array}{l} D_i = (F_{i,1} \ F_{i,2} \ \dots \ F_{i,n}) \\ D_j = (F_{j,1} \ F_{j,2} \ \dots \ F_{j,n}) \end{array}$$

$$Dist_{Euclidean}(d_i, d_j) = \sqrt{(F_{i,1} - F_{j,1})^2 + \dots + (F_{i,n} - F_{j,n})^2}$$

Example (Note, we use simple values for F_{ij} for illustration purposes)

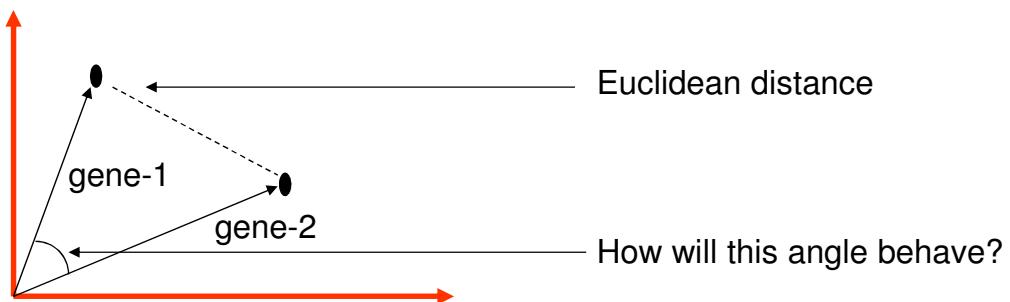
$$\left(\begin{array}{cccc} & C_1 & C_2 & C_3 \\ D_1 & 6 & 3 & 1 \\ D_2 & 2 & 8 & 0 \\ D_3 & 5 & 3 & 2 \end{array} \right) \quad \begin{array}{l} Dist(d_1, d_3) = \sqrt{(6-5)^2 + (3-3)^2 + (1-2)^2} = \sqrt{2} = 1.4 \\ Dist(d_2, d_3) = \sqrt{(2-5)^2 + (8-3)^2 + (0-2)^2} = \sqrt{38} = 6.16 \end{array}$$

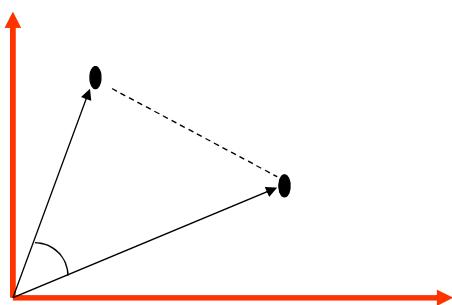
The Cosine Measure of Vector Similarity

Idea: We thought of genes as points in multidimensional condition-space

Cond	1	2	3	4	5	6
gene-1	1	1	0	0	0	0
gene-2	0	1	1	0	0	0
gene-3	1	1	0	1	1	0
gene-4	0	0	0	1	0	1

$(1, 1, 0, 0, 0, 0)$ ← gene vectors
 $(0, 1, 1, 0, 0, 0)$
 $(0, 0, 0, 1, 0, 1)$





-If gene-1 and gene-2, have similar conditions in similar proportions, then the angle between them will be smaller.

-This is equivalent to saying that the respective gene vectors will point in similar directions in the condition-space

Cosine similarity captures this intuition

Let gene-1 and gene-2 be two genes represented by their gene vectors
(Note we are only thinking of genes as rows in an expression matrix)

The Cosine similarity between these two genes is given by:

$$\text{Cosine-similarity(gene-1, gene-2)} = \cos(\theta)$$

Where θ defines the angle between the two respective gene vectors

How to compute the cosine similarity?

Recall:

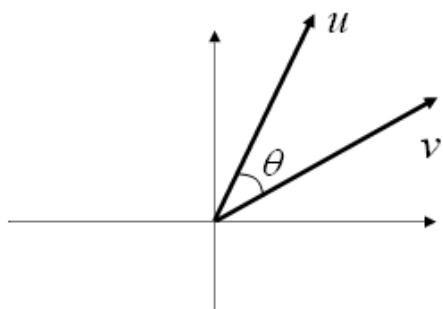
Let \mathbf{x} be a vector: $\mathbf{x} = (x_1, x_2, \dots, x_n)$

The length of the vector \mathbf{x} is given by:

$$\|\mathbf{x}\| = \sqrt{(x_1^2 + x_2^2 + \dots + x_n^2)}$$

Let $u=(x_1, y_1)$ and $v=(x_2, y_2)$ be vectors in 2 dimensions, then

$$\cos(\theta) = \frac{x_1 x_2 + y_1 y_2}{\|u\| \|v\|} = \frac{u \cdot v}{\|u\| \|v\|}$$



In the general case:

$$\left(\begin{array}{cccc} D1 & C1 & C2 & \dots & Cn \\ & F_{1,1} & F_{1,2} & \dots & F_{1,n} \\ D2 & F_{2,1} & F_{2,2} & \dots & F_{2,n} \\ \vdots & & & & \\ Dm & F_{m,1} & F_{m,2} & \dots & F_{m,n} \end{array} \right) \rightarrow \begin{array}{l} D_i = (F_{i,1} \ F_{i,2} \ \dots \ F_{i,n}) \\ D_j = (F_{j,1} \ F_{j,2} \ \dots \ F_{j,n}) \end{array}$$

$$Dist_{Cosine}(d_i, d_j) = \frac{F_{i,1} \times F_{j,1} + \dots + F_{i,n} \times F_{j,n}}{\sqrt{(F_{i,1})^2 + \dots + (F_{i,n})^2} \times \sqrt{(F_{j,1})^2 + \dots + (F_{j,n})^2}}$$

Example:

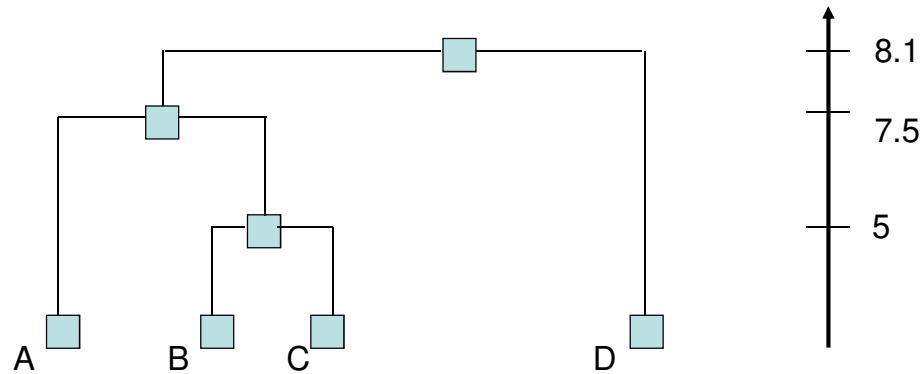
	C1	C2	C3
D1	6	3	1
D2	2	8	0
D3	5	3	2

$$Dist_{Cosine}(D1, D3) = \frac{6 \times 5 + 3 \times 3 + 1 \times 2}{\sqrt{(6)^2 + (3)^2 + (1)^2} \times \sqrt{(5)^2 + (3)^2 + (2)^2}}$$
$$= \frac{41}{\sqrt{46} \times \sqrt{38}} = 0.98$$

$$Dist_{Cosine}(D2, D3) = \frac{5 \times 2 + 8 \times 3 + 0 \times 2}{\sqrt{(2)^2 + (8)^2 + (0)^2} \times \sqrt{(5)^2 + (3)^2 + (2)^2}}$$
$$= \frac{34}{\sqrt{68} \times \sqrt{38}} = 0.669$$

Clustering Gene Expression Data: Agglomerative Clustering

- Data points are clustered using a tree structure. The data is located on the tree leaves
- Distances within the tree reflect similarity of elements (closer the elements, more similar they are)



Distance(B, C)= 5, Distance(A, B)= 7.5, Distance(A, C) = 7.5, Distance(C, D)= 8.1

© Rahul Singh

Neighbor Joining Clustering

Input: Distance matrix D_{ij}

Repeat until a single element is left

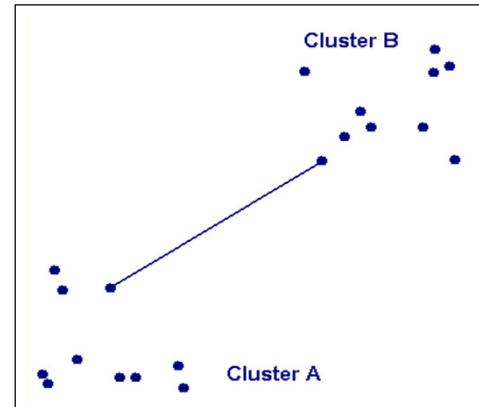
Find elements r and s such that: $D_{rs} = \min_{i,j}(D_{ij})$

Join clusters r and s into a new cluster t (inner node of the tree)

In the distance matrix, delete element r and s and replace them by t such that:

$$D_{it} = D_{ti} = \frac{D_{ir} + D_{is} - D_{rs}}{2}$$

Present the hierarchy as a tree



© Rahul Singh

Average Linkage Clustering

-Similar to the idea of NJ-algorithm. The distance between the clusters is defined as the average distance between all-pairs of data points between the clusters.

In this method, the distance between two clusters r and s is defined as:

$$D_{rs} = \frac{T_{rs}}{n_r \times n_s}$$

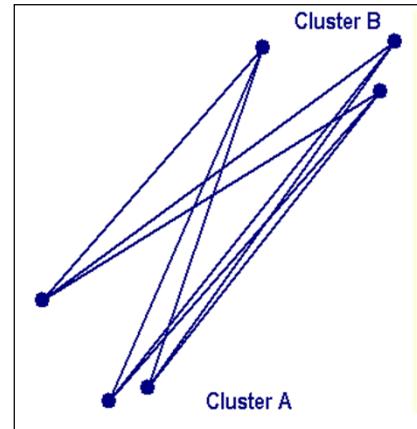
T_{rs} : Sum of distance between all pairs of points in clusters r and s

n_r, n_s : Size of clusters r and s respectively

Algorithm

Iterate till only one cluster is left

At each iteration, merge clusters for which the inter-cluster distance (as defined above) is minimal



© Rahul Singh

Application of Hierarchical Clustering to Gene Expression Data

M. Eisen, P. Spellman, P. Brown, D. Botstein, "Cluster Analysis and Display of Genome-Wide Expression Patterns", PNAS, Vol. 95, 14863-14868, 1998

Goal of the study: Check the growth response of starved human fibroblast cells

Study details: 8600 genes are monitored over 13 time points (approximately 2 cell cycles)

Data Pre-processing:

t_{ij} - fluorescence levels of gene i in condition j

r_{ij} - reference fluorescence levels of gene i in condition j

$$R_{ij} = \log\left(\frac{t_{ij}}{r_{ij}}\right); \quad N_{ij} = \frac{R_{ij} - \bar{R}_i}{std(R_i)} \text{ :denotes the normalized expression levels for (i, j)}$$

© Rahul Singh

The distance matrix was defined from N_{ij} as:

$$D_{kl} = \frac{\sum_j N_{kj} \cdot N_{lj}}{N_{cond}}$$

Where, the number of conditions checked is: N_{cond}

-Average linkage hierarchical clustering was then applied to the data

Results

- Genes in five groups serve similar functions:

A: Cholesterol biosynthesis

B: Cell cycle

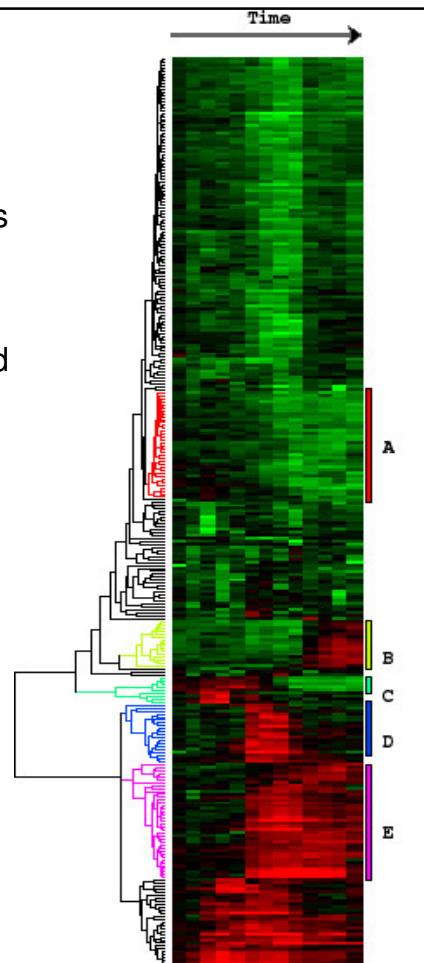
C: Intermediate-early response

D: Signaling and angiogenesis

E: Wound healing and tissue remodeling

Data available at:

<http://genome-www.stanford.edu/clustering/>



© Rahul Singh

Pros and Cons of Hierarchical Clustering

Advantages

- A single coherent global hierarchy
- Representation (tree) is historically known in biology (phylogenetics)

Disadvantages

- No real reason to assume that a binary-phylogenetic relationship is the true underlying relationship amongst the data
- Difficult to read the tree for large data sets
- Partition of the data into clusters is less explicit

© Rahul Singh

Non-Hierarchical Clustering

K-Means Clustering

Basic idea: Partition the data into K clusters based on some function that measures the quality of the partition. Perform the partition iteratively. At each iteration, modify the clusters if it improves the partition quality. Stop when no further improvement is possible

Algorithm

Step 1: Initiate an arbitrary partition of the data into k clusters

Step 2: For each cluster compute the cluster centroids (there are K centroids)

For a cluster having n point masses m_i located at positions x_i , the centroid is:

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^n m_i \mathbf{x}_i}{\sum_{i=1}^n m_i},$$

If all the m_i are equal, this simplifies to: $\bar{\mathbf{x}} = \frac{\sum_{i=1}^n \mathbf{x}_i}{n}$

© Rahul Singh

Step 3: Tessellate the data set such that every data point is associated with its closets centroid

Step 4: Redefine the clusters: A cluster is re-defined as a set of points that are all closer to a given centroid than any other centroid

Step 5: Re-compute the cluster centroids using the formula in Step 2 and repeat steps 3 through 5 till convergence

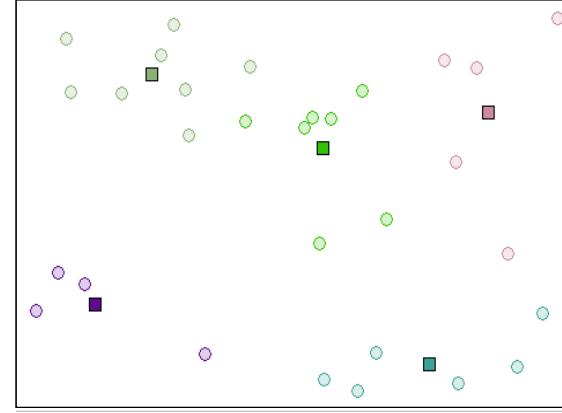
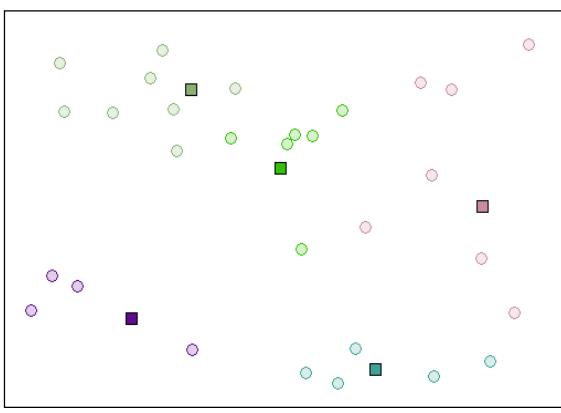
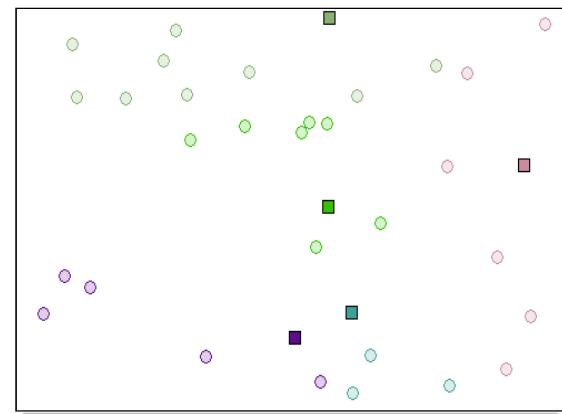
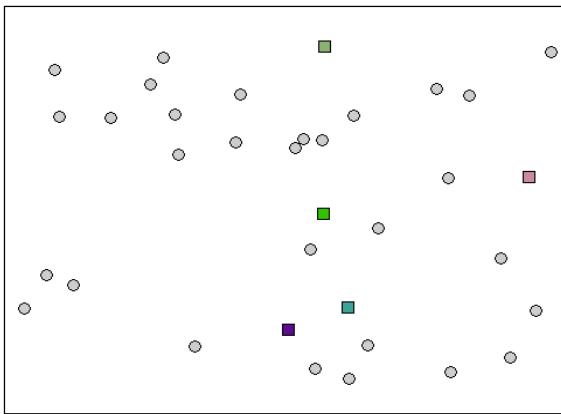
If S_j denotes the j^{th} cluster, x_n is a vector representing the n^{th} data point, and the centroid of the j^{th} cluster is: μ_j

Then, the K-means algorithm minimizes the sum-of-squares criteria:

$$J = \sum_{j=1}^K \sum_{n \in S_j} |x_n - \mu_j|^2$$

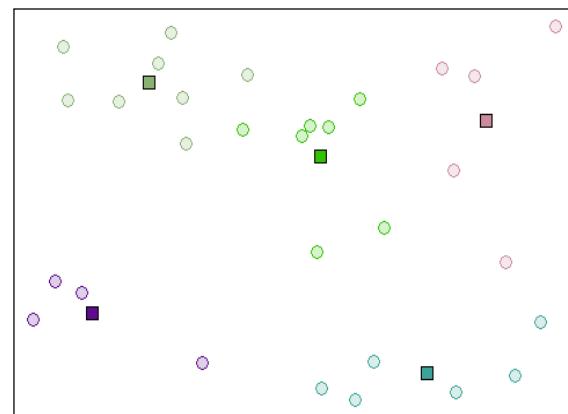
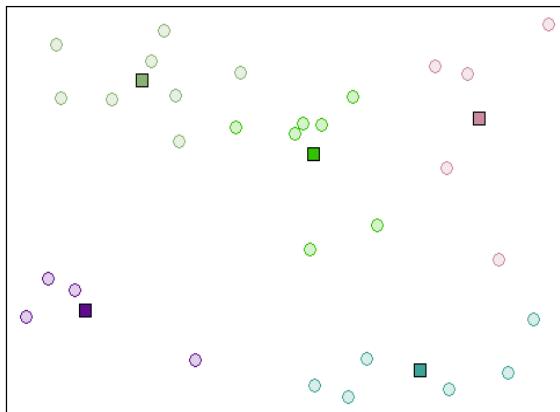
© Rahul Singh

K-Means Example



© Rahul Singh

K-Means Example

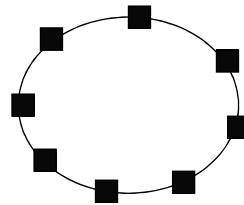
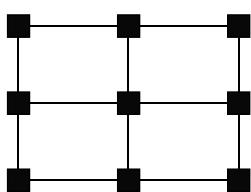


Convergence: Centroids do not move

© Rahul Singh

Self-Organizing Maps (SOM)

- Introduced by Kohonen
 - Used in many applications. Main claim when used in clustering is that SOM provide a topologically consistent mapping from the data space to the clusters
- Construction of SOM
- Choose a map size K
 - Choose a map topology (e.g. linear, grid, circular etc)



© Rahul Singh

-The SOM is mapped on the data space (nodes mapped on the space at random)
(Note, the gene vectors also exist in this space)

-The nodes of the SOM are then iteratively adjusted

- There are two versions of the SOM evolution: a *flow-through* version and a *batch* version

Flow-Through SOM

Idea

-Randomly select a data point p

-Find the closest SOM node n_p

- Move the SOM nodes towards the data point. The closest node is moved the most and the other nodes (that are neighbors of n_p) are moved by a smaller amount

-It is as if the map is being *smoothly stretched*

- Neighboring points in the data space are thus mapped to neighboring points in the map (cluster) space

Note: The data may need to be presented many times before the map converges
© Rahul Singh

Flow-Through SOM Algorithm

Input: N-dimensional vector for each data point (gene)

Step 1: Select a map size and map topology. For expression analysis, a grid topology has been used: $k = l \times m$

Step 2: Randomly initialize the SOM nodes and iterate:

Iteration i

Pick a data point p . Find the closest SOM node to p : n_p

Update the SOM nodes as follows:

$$f_{i+1}(v) = f_i(v) + H(D(n_p, v), i) \times (p - f_i(v))$$

Where:

$f_i(v)$: is the position of node v at iteration i in the sample (data) space

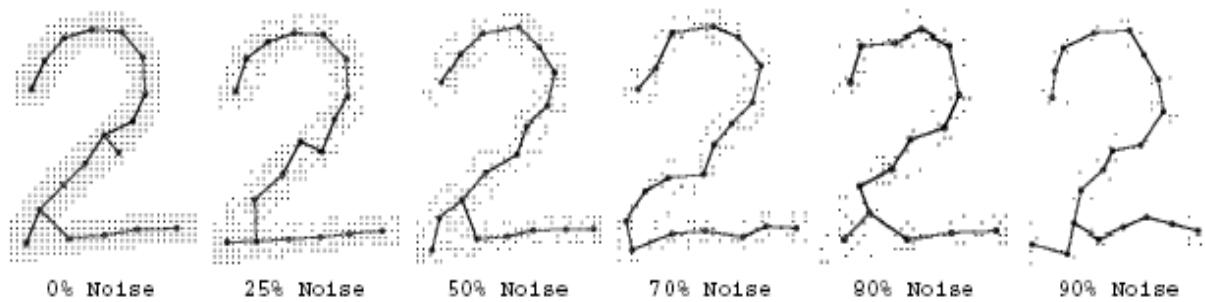
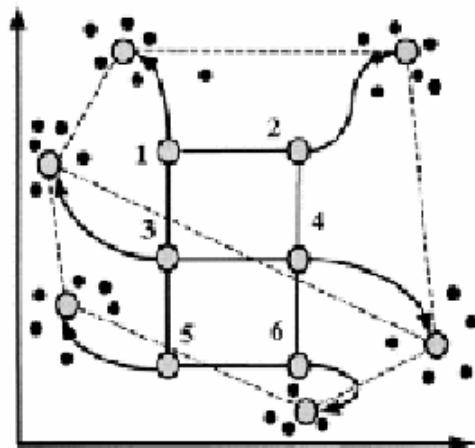
$D(x, y)$: is a distance function between nodes defined on the map topology. Example: Minimum number of edges between nodes

$H(D(.,.), i)$: A function that decreases with the distance between SOM nodes as well as with increasing number of iterations

Step 3: Iterate till convergence

© Rahul Singh

Visualizing SOM Evolution and Properties



© Rahul Singh

-Flow-through SOMs have been used for clustering gene expression data

P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub, "Interpreting Patterns of Gene Expression with Self-Organizing Maps: Methods and Application to Hematopoietic Differentiation", PNAS, Vol. 96, 2907-2912, 1999

-Program GENECLUSTER available at:

<http://www-genome.wi.mit.edu/MPR/GeneCluster/GeneCluster2.html>

Open Questions:

- How to choose the best map size?
- How to choose the best map topology?
- Can the map topology be optimally generated based on the data?
- Flow-through SOM has an inherent problem: final node positions depend on the order of data presentation
- Flow-through SOM takes a long time to converge

-Many other approaches have been tried: graph-theoretic clustering, correspondence analysis, temporal analysis (time-series), visualizations, ...

- Many open problems: Area of much activity, especially in academic research

© Rahul Singh

Lecture: Introduction to Modern Pharmaceutical Drug Discovery

© Rahul Singh

Importance

1. Exceedingly important for everyone
2. Highly complex interplay of techniques from Biology and Chemistry
3. Huge scope for engineers, computer-scientists, statisticians, mathematicians
4. Most industrial biology, organic chemistry, and physical and computational chemistry is done in this sector
5. Most of industrial computing in life sciences is done in this sector
6. Increasing interest in academia towards drug discovery

Key Attribute: Multi-stage process. Stages can be highly dissimilar, yet need to interface effectively

© Rahul Singh

Key Stages of Drug Discovery



Stage-1: Target Discovery

Big-Picture: This stage deals with the selection of the disease, against which a drug will be developed.

Two major ways of doing this:

1. Target-based drug discovery: Using molecular biology, identify the function of a possible therapeutic target and its role in a disease. Such targets could be: DNA, proteins and enzymes, RNA and ribosomal targets etc.

Typically, target discovery is done using *functional genomics*, where gene activity is systematically studied in healthy Vs diseased organism/tissue/cell. This is done among others, through the analysis of regulatory-networks, pathways, protein-protein interactions, and gene knockouts

© Rahul Singh

Target Discovery

Techniques such as microarrays, gene-knockouts in model organisms are typically used in this stage.

2. Physiology-based drug discovery: Older approach and was used before the advent of molecular biology and genomics. In this paradigm, the disease phenotype is directly targeted as part of the drug discovery process. Often, such an approach uses existing knowledge about chemical entities (natural or synthetic) than can ameliorate the disease-phenotype (e.g. discovery of Aspirin (from Willow bark))

Note that the two above approaches are not mutually exclusive

© Rahul Singh

Target Validation



Stage-2: Target Validation

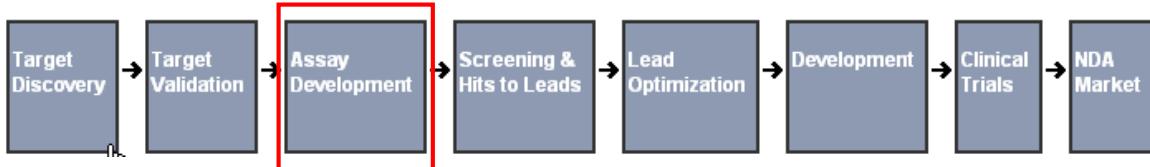
Big Picture: Demonstrate that a molecular target is critically involved in a disease process and that modulation of this target will have therapeutic value.

This stage involves techniques like gene knockouts, transgenic models, RNAi, and antisense DNA/RNA

Systems Biology can be expected to become one of the important new ways to improve the efficacy of this step

© Rahul Singh

Assay Development



Stage-3: Assay Development

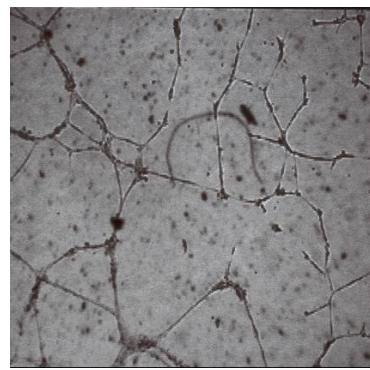
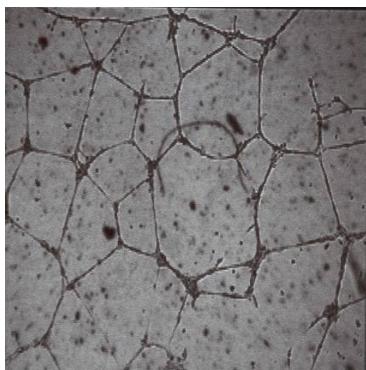
Big Picture: Development of experimental techniques/protocols that allow derivation of quantitative data about the target and/or interactions of putative drugs with the target.

Assays are of two types: In-vitro (outside a living body) or in-vivo. In vitro assays are typically easier to design and interpret, and cheaper. However, they simplify the problem and essentially deal with it in isolation.

© Rahul Singh

Assay Development

Example of an assay: quantify the blood capillary growth/reduction to determine effectiveness of a anti-tumor drug



A key technology in modern drug discovery is “High-Throughput Screening” (HTS) which allows assessing the activity of a large number of compounds on a given target in parallel.

© Rahul Singh

High-Throughput Screening



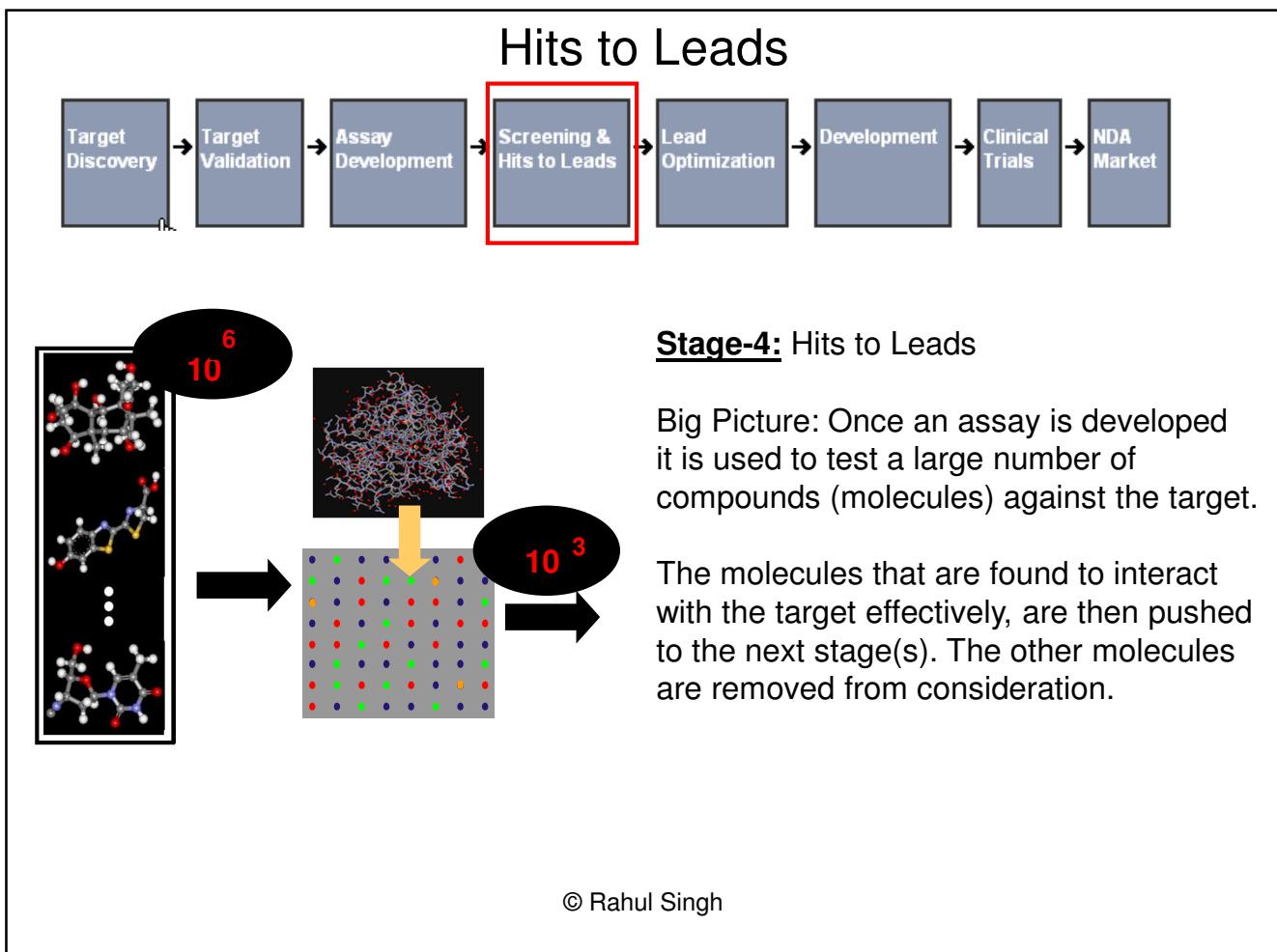
	1	2	3	4	5	6	7	8	9	10	11	12
A	●	●	●	●	●	●	●	●	●	●	●	●
B	●	●	●	●	●	●	●	●	●	●	●	●
C	●	●	●	●	●	●	●	●	●	●	●	●
D	●	●	●	●	●	●	●	●	●	●	●	●
E	●	●	●	●	●	●	●	●	●	●	●	●
F	●	●	●	●	●	●	●	●	●	●	●	●
G	●	●	●	●	●	●	●	●	●	●	●	●
H	●	●	●	●	●	●	●	●	●	●	●	●

Well Properties
Events
Drug ID
Drug Conc (nm)
Strain
Events
Parasitemia
Cells
21,580
22,295
22,425
22,620
22,885
23,075
23,335
23,985
31,135

High-throughput screens are run using multi-well (96/384/1536) plates, with each well containing the target being studied (e.g. a protein). Then (typically) different molecules are added to each well, and their interaction with the target measured.

Developing assays that can be run in high-throughput mode is a key task in modern pharmaceutical biochemistry. Also, storage, management, and visualization of such data is a key area for applying computer science.

© Rahul Singh

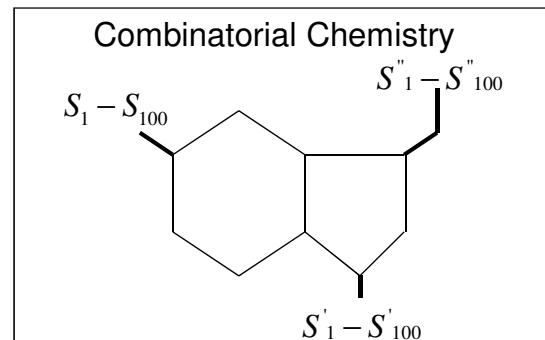
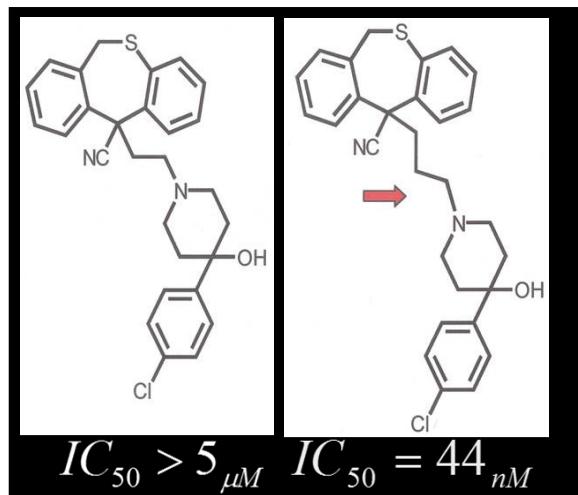


Combinatorial Chemistry

Combinatorial chemistry is one of the key techniques used for synthesizing large number of highly similar compounds. These compounds can then be tested against targets to find the hits.

Two questions:

1. Why do we need to test a large number of highly similar compounds?
2. How does combinatorial chemistry work?



© Rahul Singh

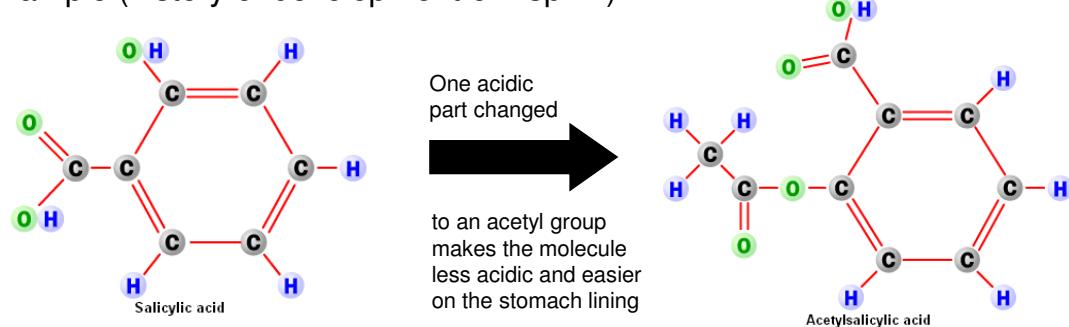
Lead Optimization



Stage-5: Lead optimization

Big Picture: A combination of techniques from medicinal chemistry, computational molecular modeling, and computational chemistry is used to “improve” how a hit interacts with the target and how it affects the rest of the body.

Example (history of development of Aspirin):



© Rahul Singh

PK, PD, and Toxicity

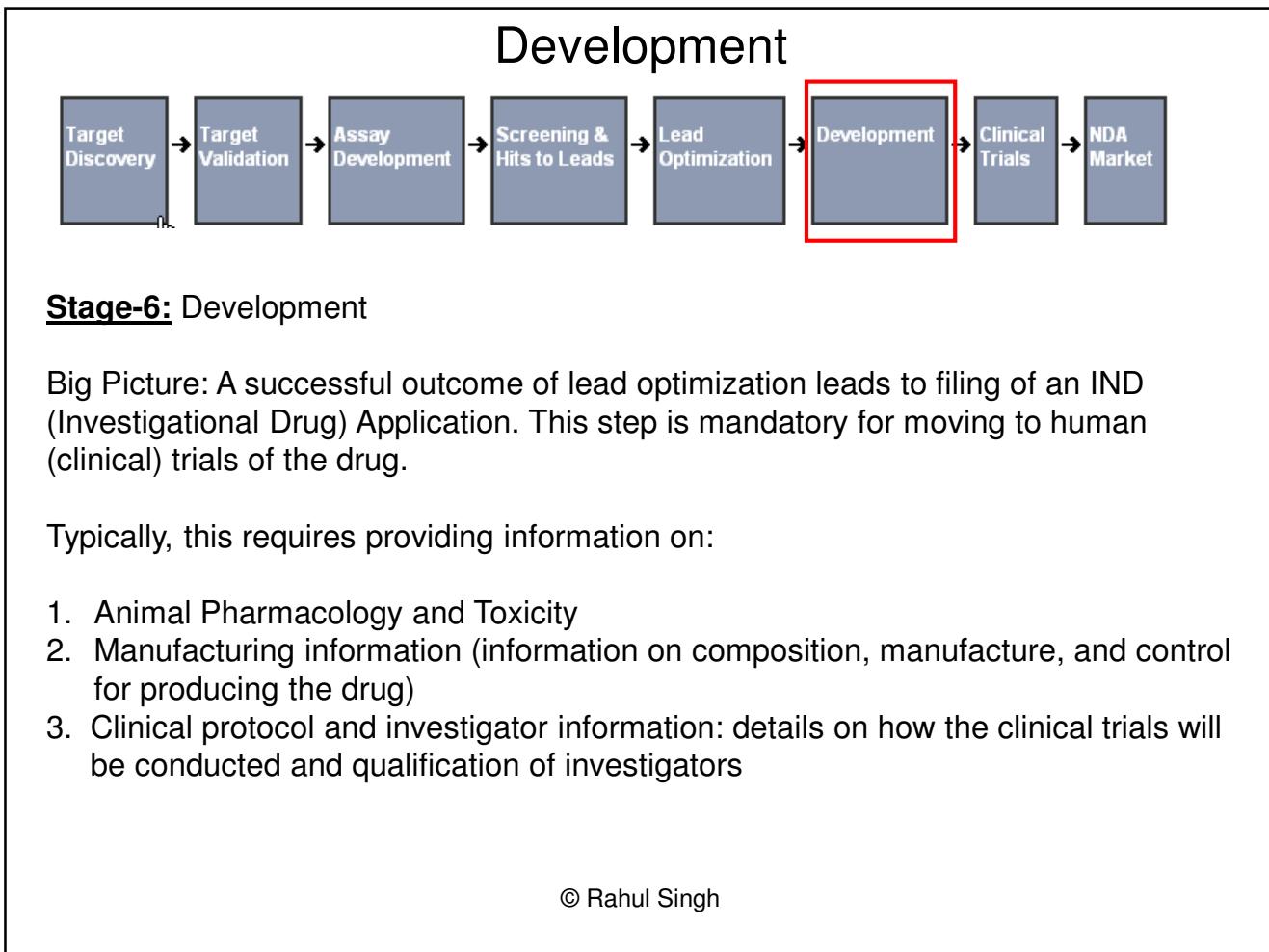
Pharmacokinetics (PK): The study of the action of an organism on a drug. This includes ADME (Absorption, Distribution, Metabolism, and Excretion).

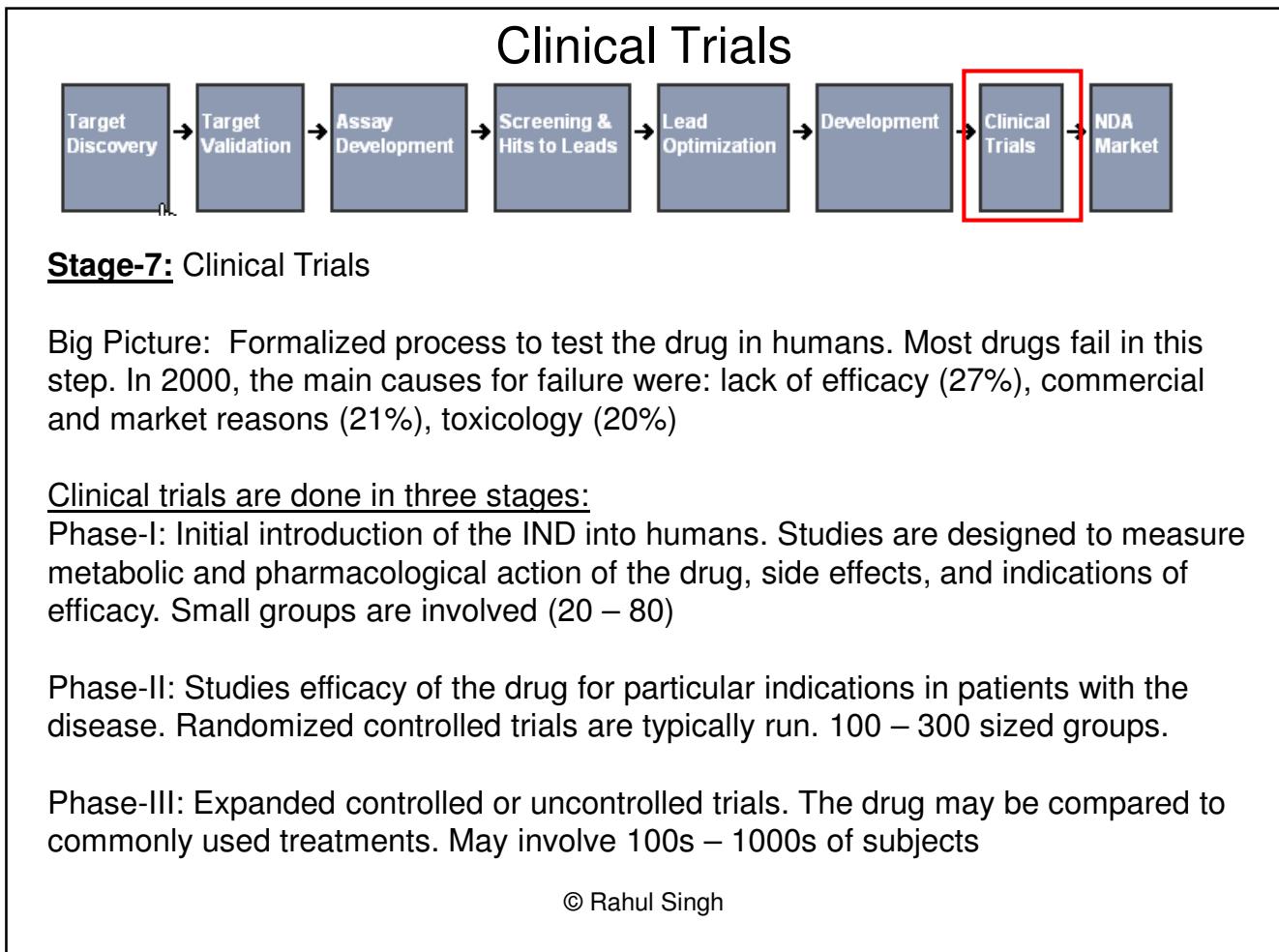
Pharmacodynamics (PD): The study of the action of a drug molecule on an organism.

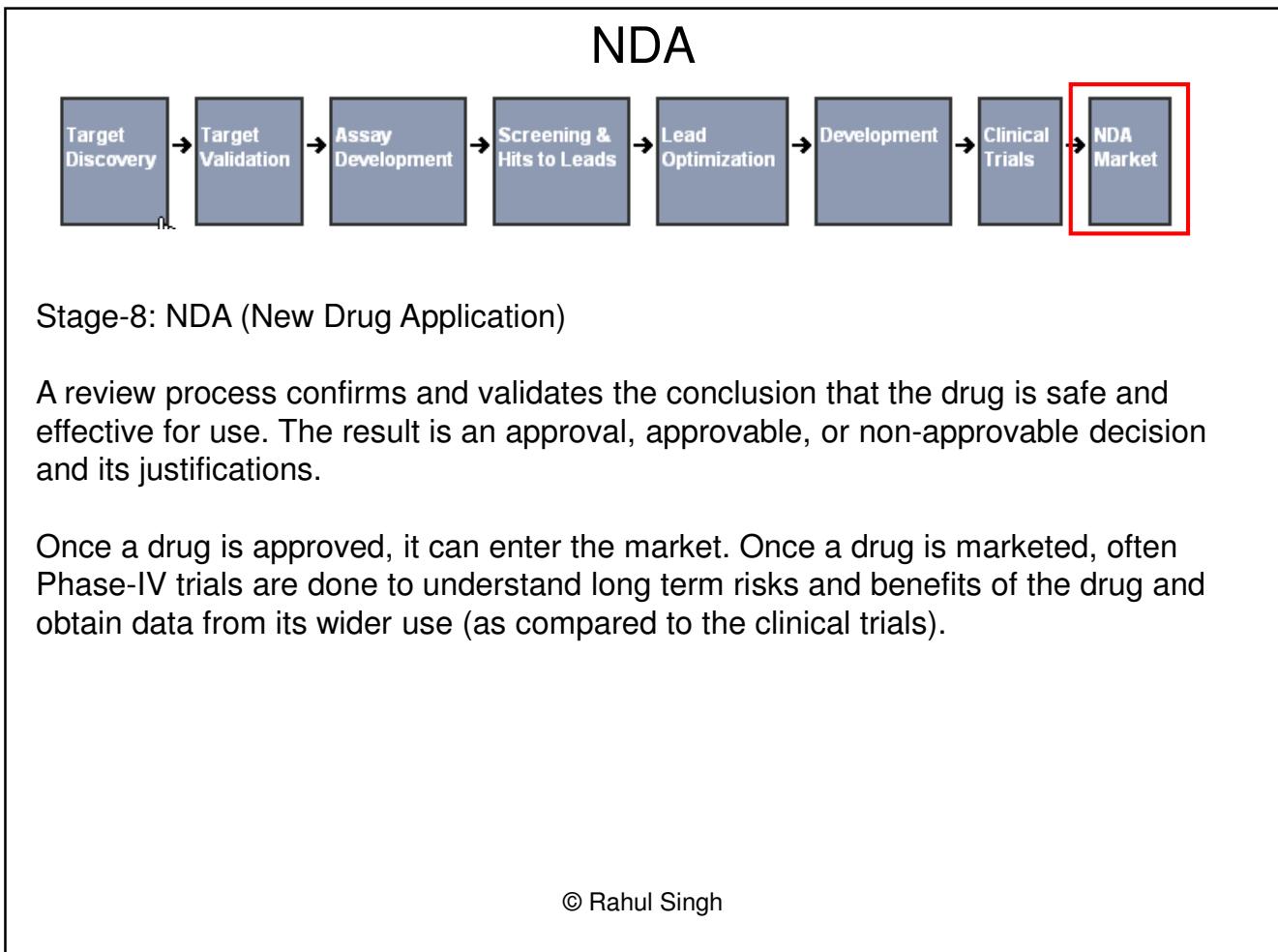
Toxicity: The degree to which a chemical substance can harm an organism. By definition all drugs are toxic. It is a matter of how significant is the toxicity. Typically it is hard to optimize a molecule against toxicity. So, early identification of toxicity can be very useful in limiting wastage of time/effort/resources.

Early assessment (or rigorous prediction) of ADMET is a key area of research in pharmaceutical sciences and computational drug discovery. Current efforts in this area (computational efforts) include application of data mining and machine learning techniques to this problem, to predict toxicity.

© Rahul Singh







Secondary Structure Prediction

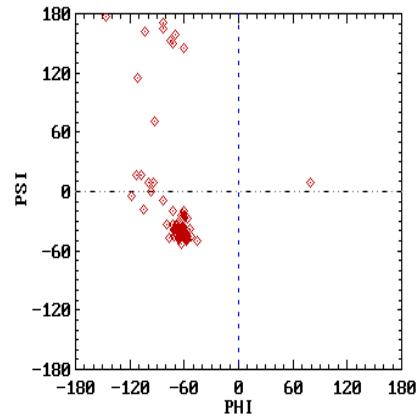
© Rahul Singh

Key SSE: α - helix and β - strands

The α - helix : is characterized by phi and psi angles of roughly -60 degrees

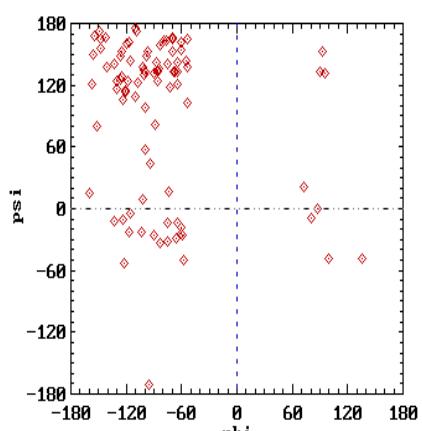
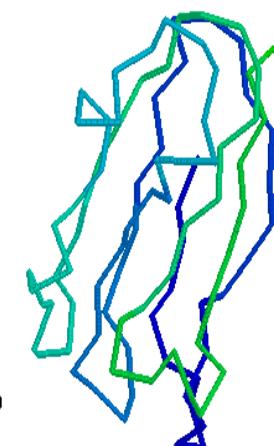
The β - strands: are characterized by regions of extended, nearly linear, backbone conformations with phi roughly equal to -135 degrees and psi roughly equal to 135 degrees

A plot of φ (phi) vs ψ (psi) is called the Ramachandran plot



Cytochrome C-256 structure

© Rahul Singh



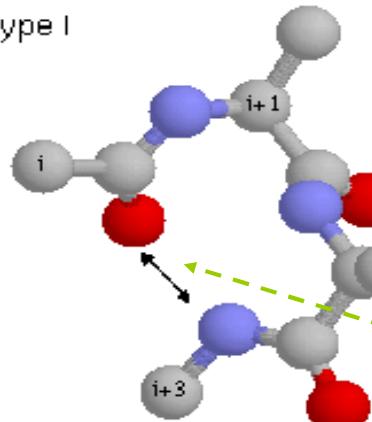
Plastocyanin structure

The β - Turn

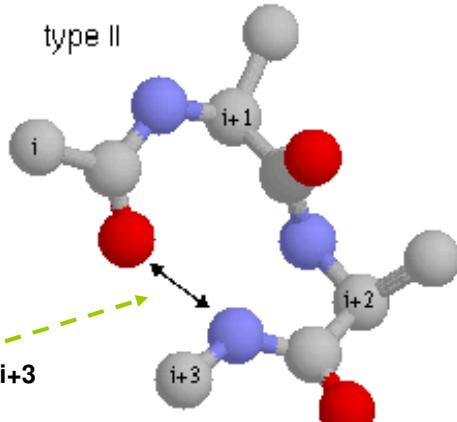
- The β - Turn allows the peptide chain to reverse direction
- Turns involving 4 residues are more common with H-bonding from the Carbonyl of residue 1 to the amino group of residue 4

Designation	Residue 2 Phi, Psi	Residue 3 Phi, Psi	Comments
I	-60, -30	-90, 0	
II	-60, 120	80, 0	Most common type.

type I



type II



H-bond between C=O of residue i to NH of residue i+3

© Rahul Singh

Predicting Secondary Structure: The Chou-Fasman Method (Observation-based Approach)

- Is a statistical approach (based on observed data) to predict secondary structure
- First widely used method with a 3-state prediction: alpha-helix, beta-strands, and beta turns
- Each amino acid is assigned several conformational parameters that represent its structural propensities (to participate in alpha-helices, beta sheets, and beta turns)
- Parameters are based on observed frequencies in a set of sample proteins of known structure

The parameters are of the type:

$$p_{\text{structure}} = \frac{f_{\text{struct}}}{N_{\text{struct}} / N_{\text{total}}}$$

Frequency of a particular residue
in the secondary structure

Average frequency of a structure

© Rahul Singh

Chou-Fasman Parameters

Name	P (a)	P (b)	P (turn)	f (i)	f (i+1)	f (i+2)	f (i+3)
Alanine	142	83	66	0.06	0.076	0.035	0.058
Arginine	98	93	95	0.070	0.106	0.099	0.085
Aspartic Acid	101	54	146	0.147	0.110	0.179	0.081
Asparagine	67	89	156	0.161	0.083	0.191	0.091
Cysteine	70	119	119	0.149	0.050	0.117	0.128
Glutamic Acid	151	037	74	0.056	0.060	0.077	0.064
Glutamine	111	110	98	0.074	0.098	0.037	0.098
Glycine	57	75	156	0.102	0.085	0.190	0.152
Histidine	100	87	95	0.140	0.047	0.093	0.054
Isoleucine	108	160	47	0.043	0.034	0.013	0.056
Leucine	121	130	59	0.061	0.025	0.036	0.070
Lysine	114	74	101	0.055	0.115	0.072	0.095
Methionine	145	105	60	0.068	0.082	0.014	0.055
Phenylalanine	113	138	60	0.059	0.041	0.065	0.065
Proline	57	55	152	0.102	0.301	0.034	0.068
Serine	77	75	143	0.120	0.129	0.125	0.106
Threonine	83	119	96	0.086	0.108	0.065	0.079
Tryptophan	108	137	96	0.077	0.013	0.064	0.167
Tyrosine	69	147	114	0.082	0.065	0.114	0.125
Valine	106	170	50	0.062	0.048	0.028	0.053

-Additionally, each amino acid is assigned four turn parameters: $f(i)$, $f(i+1)$, $f(i+2)$, and $f(i+3)$ corresponding to the frequency with which the amino acid was observed in the first, second, third, or fourth position of a beta-turn

Each position of a turn has preferences, for example, Proline at turn position 2 is more than double that any other residue

© Rahul Singh

The Chou-Fasman Algorithm: Basic Structure

1. Find helical-initiation regions
2. Extend the helices until they reach tetra-peptide breakers (4 consecutive peptides that do not have helical properties)
3. Verification of the region as alpha helix
4. Find beta-initiation sites
5. Extend until they reach tetra-peptide breakers
6. Verify region as beta-strand
7. Resolve conflicts between alpha and beta
8. Find turns

Finding alpha-helices: (1, 2, 3), beta-strands: (4, 5, 6), resolving conflicts (7), and turns (8)

© Rahul Singh

Finding Alpha Helices

- (a) Find all regions where four of six contiguous amino-acids have $P(a) > 100$
- (b) For each region identified above, extend it in both directions till four contiguous residues with $P(a) < 100$ are encountered
- (c) For each region identified above compute:

$\sum P(a)$ - sum of $P(a)$ values for each residue in the region

$\sum P(b)$ - sum of $P(b)$ values for each residue in the region

- (d) If the region has more than 5 residues (> 5 residues in length) then

If $\sum P(a) > \sum P(b)$ then

The region is predicted to be an alpha helix

© Rahul Singh

Finding Beta-Strands

(Similar idea to that of finding alpha-helices)

- (a) Find all regions where four of six contiguous amino-acids have $P(b) > 100$
- (b) For each region identified above, extend it in both directions till four contiguous residues with $P(b) < 100$ are encountered
- (c) For each region identified above compute:

$\sum P(a)$ - sum of $P(a)$ values for each residue in the region

$\sum P(b)$ - sum of $P(b)$ values for each residue in the region

- (d) The region is predicted to be a beta strand:

If $\sum P(b) > \sum P(a)$ and

$average(P(b)) > 100$

© Rahul Singh

Resolving Conflicts and Finding Turns

Resolving conflicts

If a an alpha helix region overlaps with a region identified as a beta strand, than the overlapping part is predicted to be

an alpha helix if $\sum P(a) > \sum P(b)$

a beta strand if $\sum P(b) > \sum P(a)$

Predicting Beta Turns

For each residue j , calculate its turn propensity $P(t, j)$

where: $P(t, j) = f(i, j) \times f(i+1, j+1) \times f(i+2, j+2) \times f(i+3, j+3)$

$f(i, j)$: is the $f(i)$ value of residue j

$f(i+1, j+1)$: is the value of the residue subsequent to j at position $i+1$ and so on

Predict a beta turn starting at position j if

$P(t) > 0.000075$ AND (average value of $P(\text{turn})$ from position i thru $i+3$) > 100
AND

$\sum P(a) < \sum P(\text{turn}) > \sum P(b)$ over residues in position i thru $i+3$

© Rahul Singh

Example

Predict the turns in the sequence: CAENKLDHV**A**DCCILFMTWYNDGPCIFIYDNGP

Name	P(a)	P(b)	P(turn)	f(i)	f(i+1)	f(i+2)	f(i+3)
Alanine	142	83	66	0.06	0.076	0.035	0.058
Arginine	98	93	95	0.070	0.106	0.099	0.085
Aspartic Acid	101	54	146	0.147	0.110	0.179	0.081
Asparagine	67	89	156	0.161	0.083	0.191	0.091
Cysteine	70	119	119	0.149	0.050	0.117	0.128
Glutamic Acid	151	037	74	0.056	0.060	0.077	0.064
Glutamine	111	110	98	0.074	0.098	0.037	0.098
Glycine	57	75	156	0.102	0.085	0.190	0.152
Histidine	100	87	95	0.140	0.047	0.093	0.054
Isoleucine	108	160	47	0.043	0.034	0.013	0.056
Leucine	121	130	59	0.061	0.025	0.036	0.070
Lysine	114	74	101	0.055	0.115	0.072	0.095
Methionine	145	105	60	0.068	0.082	0.014	0.055
Phenylalanine	113	138	60	0.059	0.041	0.065	0.065
Proline	57	55	152	0.102	0.301	0.034	0.068
Serine	77	75	143	0.120	0.139	0.125	0.106
Threonine	83	119	96	0.086	0.108	0.065	0.079
Tryptophan	108	137	96	0.077	0.013	0.064	0.167
Tyrosine	69	147	114	0.082	0.065	0.114	0.125
Valine	106	170	50	0.062	0.048	0.028	0.053

Amino Acid	SLC
Isoleucine	I
Leucine	L
Valine	V
Phenylalanine	F
Methionine	M
Cysteine	C
Alanine	A
Glycine	G
Proline	P
Threonine	T
Serine	S
Tyrosine	Y
Tryptophan	W
Glutamine	Q
Asparagine	N
Histidine	H
Glutamic acid	E
Aspartic acid	D
Lysine	K
Arginine	R

(i) $P(t) = 0.06 \times 0.11 \times 0.117 \times 0.128 = 0.000098842 > 0.000075$

(ii) Average P(turn) = $(66+146+119+119)/4=112.5 > 100$

(iii) $\sum P(a) = 142+101+70+70=383$, average = $95.75 < P(\text{turn})$

(iv) $\sum P(b) = 83+54+119+119=375$, average = $93.75 < P(\text{turn})$

© Rahul Singh

→ Beta Turn

Predict the turns in the sequence: CAENKLDHVADCCILFMTWYNDGPCIFIYDNGP

Name	P(a)	P(b)	P(turn)	f(i)	f(i+1)	f(i+2)	f(i+3)
Alanine	142	83	66	0.06	0.076	0.035	0.058
Arginine	98	93	95	0.070	0.106	0.099	0.085
Aspartic Acid	101	54	146	0.147	0.110	0.179	0.081
Asparagine	67	89	156	0.161	0.083	0.191	0.091
Cysteine	70	119	119	0.149	0.050	0.117	0.128
Glutamic Acid	151	037	74	0.056	0.060	0.077	0.064
Glutamine	111	110	98	0.074	0.098	0.037	0.098
Glycine	57	75	156	0.102	0.085	0.190	0.152
Histidine	100	87	95	0.140	0.047	0.093	0.054
Isoleucine	108	160	47	0.043	0.034	0.013	0.056
Leucine	121	130	59	0.061	0.025	0.036	0.070
Lysine	114	74	101	0.055	0.115	0.072	0.095
Methionine	145	105	60	0.068	0.082	0.014	0.055
Phenylalanine	113	138	60	0.059	0.041	0.065	0.065
Proline	57	55	152	0.102	0.301	0.034	0.068
Serine	77	75	143	0.120	0.139	0.125	0.106
Threonine	83	119	96	0.086	0.108	0.065	0.079
Tryptophan	108	137	96	0.077	0.013	0.064	0.167
Tyrosine	69	147	114	0.082	0.065	0.114	0.125
Valine	106	170	50	0.062	0.048	0.028	0.053

Amino Acid	SLC
Isoleucine	I
Leucine	L
Valine	V
Phenylalanine	F
Methionine	M
Cysteine	C
Alanine	A
Glycine	G
Proline	P
Threonine	T
Serine	S
Tyrosine	Y
Tryptophan	W
Glutamine	Q
Asparagine	N
Histidine	H
Glutamic acid	E
Aspartic acid	D
Lysine	K
Arginine	R

(i) $P(t) = 0.147 \times 0.05 \times 0.117 \times 0.056 = 0.000048157 < 0.000075$

- Not a beta turn !

© Rahul Singh

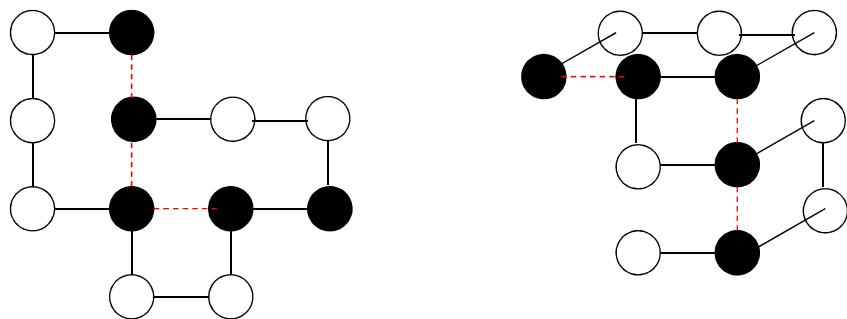
Simplified Physics-Based Algorithms for Protein Folding: Lattice Models

- Most algorithms for protein folding make simplifying assumptions to make the computations simpler
- One commonly used idea is to limit the degrees of conformational freedom given to the protein backbone. In such a scheme, the alpha C atoms are constrained to lie on a 2D or 3D grid (lattice)
- The most commonly used lattice model is the H-P model (Hydrophobic-Polar model) In it each residue is represented as a single atom of fixed radius. Further, each such atom is assigned one of two types: Hydrophobic or Polar

HOW?

hydrophobic (side chain having C and H – do not form H-bonds with water molecules)
polar (side chains with O or N which form H-bonds with water more easily)

© Rahul Singh

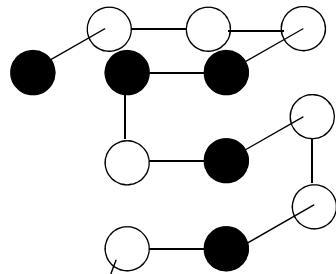
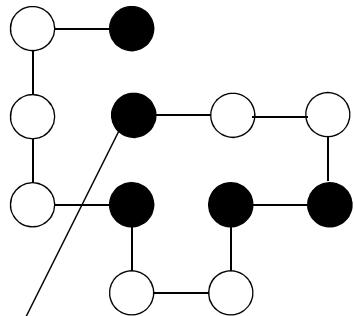


2D (left) and 3D (right) lattice H-P models of a polypeptide. Hydrophobic residues are shown as black filled circles, polar residues are white circles

- Each such model can be scored by the number of Hydrophobic-Hydrophobic contacts
- Each such contact is assumed to provide an energy contribution of -1, except those contacts that are adjacent in the primary structure
- The optimal conformation is the one with most H-H contacts (generally this can be achieved by a conformation where there is a hydrophobic-core consisting of many H-residues with the polar residues on the surface of the polypeptide)
- The scores for the 2D and 3D conformations above are -3

© Rahul Singh

Representing Lattice Configurations



- Place the first residue at (0, 0) and represent the direction moved for each residue.
In 2D these are (Up, Down, Right, Left) and in 3D (Up, Down, Right, Left, Back, Forward),
where B corresponds to increasing Z axis values and F to decreasing Z axis values

→ R R D L D L U L U U R

↓
R B U F L U R B L L F

© Rahul Singh

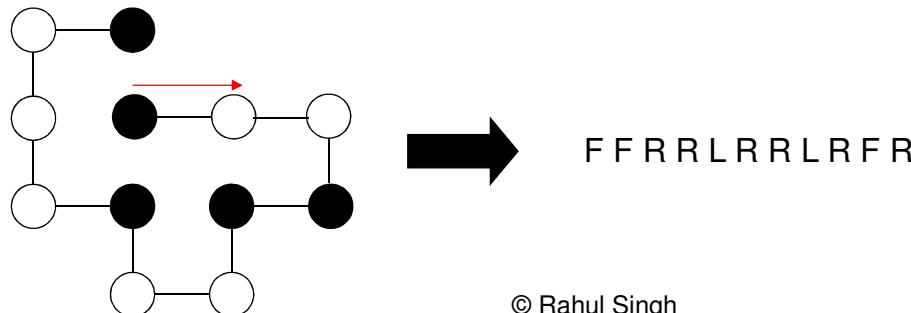
Relative Direction Representation

In the previous scheme there are

- 4 possible choices of direction at each point in a 2D lattice model
- 6 possible choices of direction at each point in the 3D lattice model

A **relative direction representation** can be used to reduce the number of choices at each step

- assume you are located at each residue and looking at the following residue
 - In 2D lattices you can go F (forward), L (left), or R (right)
 - In 3D lattices you can go F (forward), L (left), R (right), U (up), or D (down)



© Rahul Singh

Optimizing Lattice Models and Off-Lattice Modeling

-Certain configurations in Lattice models can lead to residues occupying the same space – these are called bumps or steric collisions. (e.g. L L L L in a relative representation). This can be resolved in multiple ways:

- Assign high-energy to conformations with bumps
- Avoid bumps using preference ordering. Each residue has a set of possible directions – e.g. L, F, R – it picks the one which does not lead to a steric collision

-The lattices thus obtained are scored and the best one selected

- **Off-Lattice Models** allow the protein any allowable values for the phi and psi angles
- A complex energy function is optimized to pick the best configuration. The energy function may have various terms accounting for Hydrogen bonding, electrostatic interactions, solvent-interactions etc.

-The resultant protein structure can be compared with an experimental one by overlaying them in such a way that the RMSD (root mean square deviation) between the alpha atoms is minimized

© Rahul Singh

Homology Modeling

Predicts the structure of a target protein by comparing it with the structures of related proteins

Main Steps:

1. Use sequence similarity (BLAST, FASTA) to identify related structures to the one of interest (these are called template structures)
2. Align the target sequence to the sequences of the related proteins using a multiple alignment approach (e.g. CLUSTALW). This allows identifying regions of the target that are conserved across all the template structures (and those that are not).
3. Construct a model of the structure where its backbone is aligned with the conserved regions in the template. Where the templates diverge, use secondary structure prediction/manual evaluation etc. for the target protein
4. Model the loops either by searching a database for the best loop conformation or by conformation search on the loop structure.
Note that the structure typically diverges in the loop region, so loops are modeled separately. Also note that modeling loops longer than 6 residues is difficult to do well today
5. Model the side chains
6. Evaluate: Check for steric collisions, values of phi/psi not in allowable regions of the Ramachandran plot, unfavorable bond lengths and angles etc. Often these anomalies are corrected by hand

© Rahul Singh
-

CSC-857 Bioinformatics Computing

**What Next?
(Models for Computing in Life Sciences)**

© Rahul Singh

Goals of the Lecture

- We have looked at many techniques in Bioinformatics: Sequence matching, sequence analysis, various problems in phylogenetics, transcriptional analysis, structural analysis, pharmaceutical drug discovery. . .
 - We have surveyed a series of advanced ideas through paper presentations
 - Each one of us is working at a detailed level on a significant specific technical problem
- How do I put all this together in a big picture, given
- I would like to make my next career step in the industry
 - I would like to make my next career step in academia

© Rahul Singh

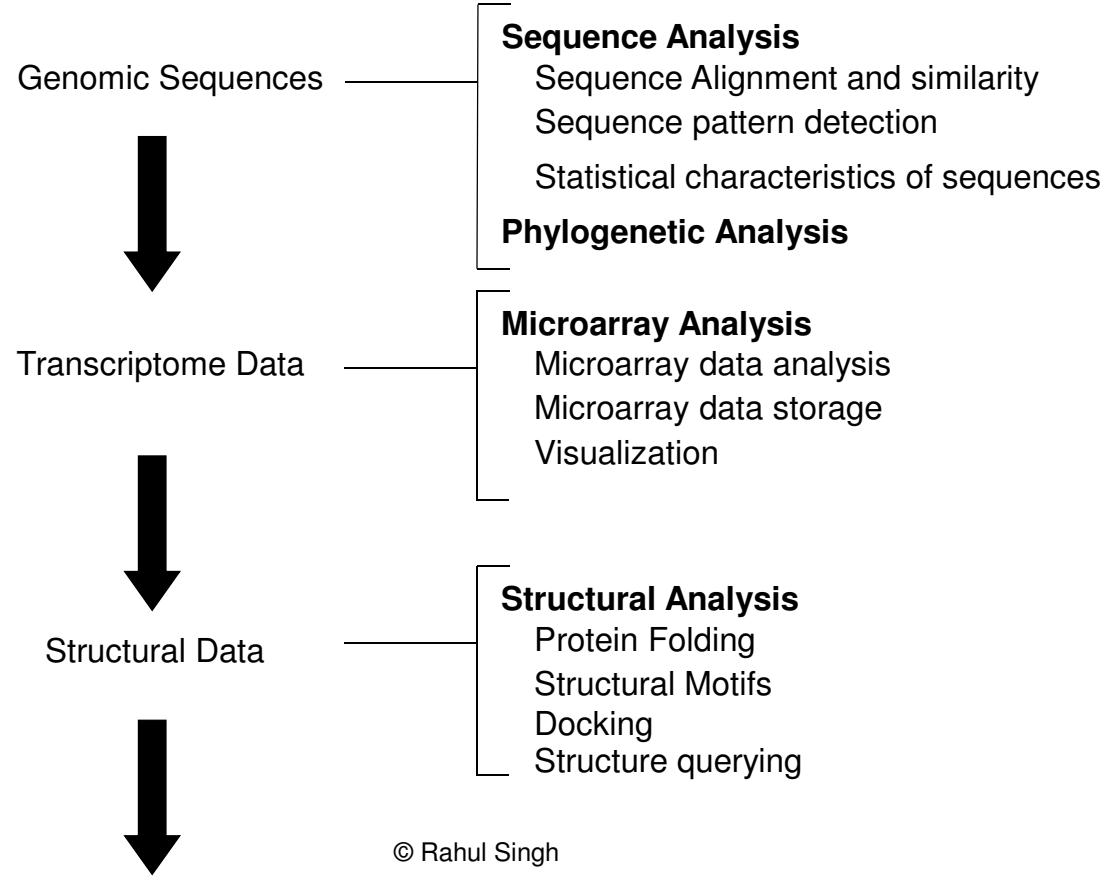
Models for Biological Information Generation

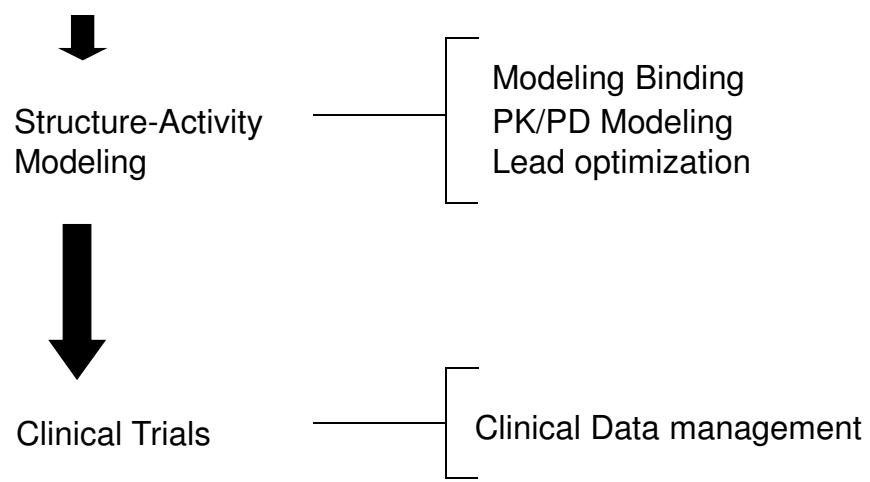
Generative models: Models (ways of understanding) of how information in life sciences is generated

1. Models reflecting the biochemical flow of living systems
 2. Models reflecting technical methodologies
 3. Process Management
 4. Processing/analysis/management of Data types
 5. Meta Data processing/analysis/management
- •
•

© Rahul Singh

Models reflecting the biochemical flow of living systems





(Examples: General Biotech and Pharmaceutical sector: SRI, Genentech, Celera Genomics, Exelixis, Incyte, Tularik/Amgen, Vertex, Rigel, Elan, Cytokinetics, Millennium, Roche, BMS, Pfizer, Novartis, GlaxoSmithKline, and the rest of the biotech industry ...)

© Rahul Singh

Models reflecting technical methodologies

Sequencing



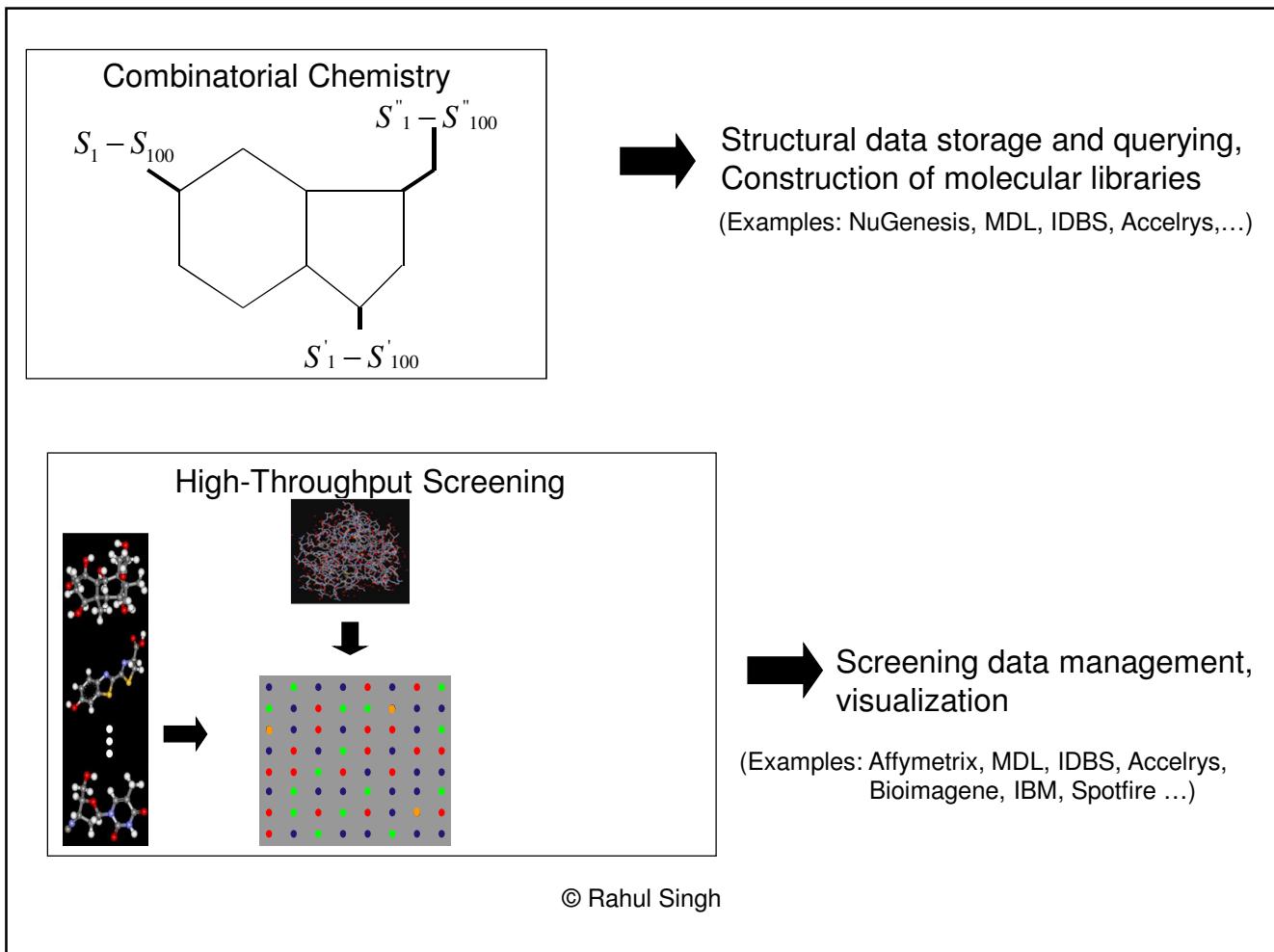
Sequence assembly, analysis, matching
(Examples: Sun, IBM, ...)

Mass Spectrometry



Enhancement and analysis of MS data
(ACDLabs, LabVantage, Applied Biosystems, NuGenesis, ...)

© Rahul Singh



Process Management

- Process centric and not data centric view
 - Essential in any big (people and/or data) organization
- (Examples: GeoSpiza, CDM, ...)

Combinatorial Chemistry
Library-1 synthesized
Library-2 partially synthesized
Target-3 synthesized
Library-4 not synthesized
.

Target Identification
Target-1
Target-2
Target-3
.
.



Target Validation
Target-1 validated
Target-2
Target-3 validated
.
.



Screening
Target-1 screened
Target-2
Target-3 screened
.
.

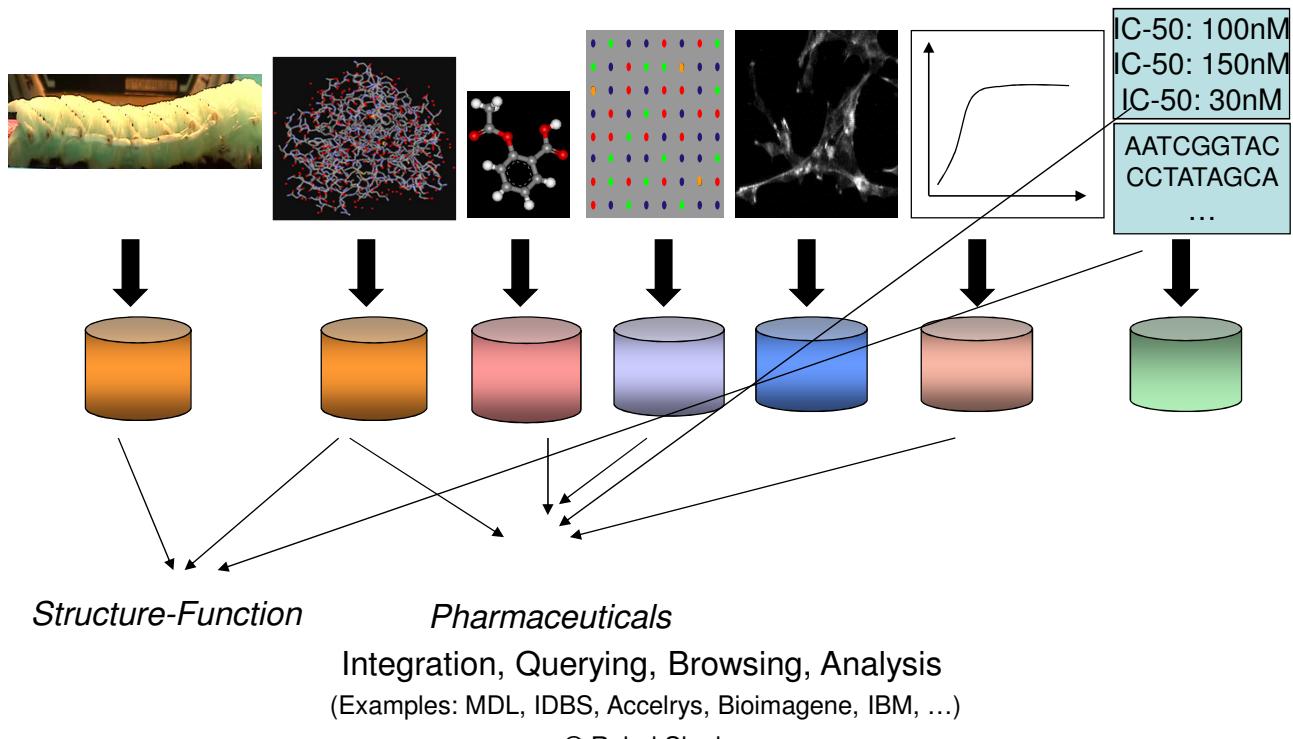
Lead Optimization
Target-1/Library-1 optimized
Target-2/Library-1 ongoing
Target-3/Library-3 optimized
.
.



© Rahul Singh

Processing/analysis/management of Data types

- Data in life sciences is inherently multi-modal/multimedia



Metadata Processing/analysis/management

There is a large amount of metadata that is generated and consequently has to be managed, processed, and presented. These include:

-Various types of “digital libraries” such as Publication repositories

For example:

MEDLINE has over 12 million papers and grows at the rate of approximately 10K publications a week

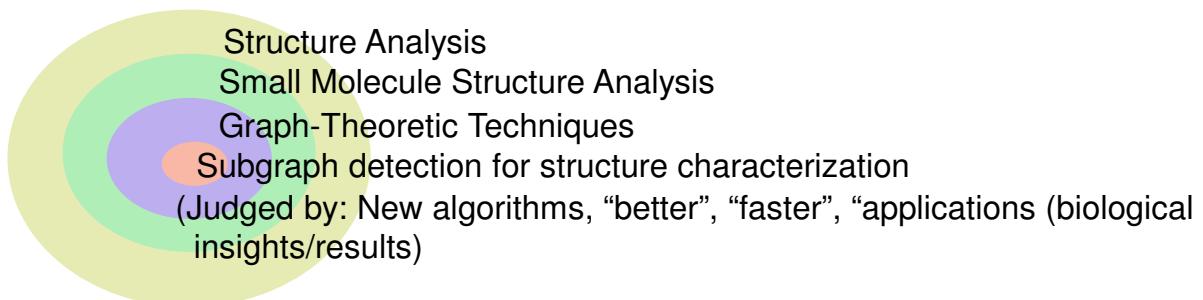
- Better ways to access/query/explore this data
- Data mining on literature (gene function, gene-disease correlation, protein function)
- Protein focused mining: function, binding/inhibitors, homologs

(Examples: CAS, IBM, ...)

© Rahul Singh

Computing and Life Sciences: Industry Vs Academic Models

General Model in Academia



General Model in Industry

Example Problem: Need to find new molecules to test as drugs

Constraints: Lot of data (need fast algorithm), limited development time and manpower, has to interface with other technologies (e.g. compound synthesis, high-throughput screening)

Judged by: Needs to work, may not be perfect, may or may not be novel, can be used in conjunction with other techniques (both computational and biological)

© Rahul Singh

Models: Academia Vs Industry

Academia: The Algorithm/software is the goal/product (in computing)
The biological insight

Industry:

Biology sector: The result (product) obtained by applying the algorithm is the goal

Computing/Software sector: The software is the goal/product

Also Industry:

Using algorithms/software to analyze/curate/discover biologically important data
(Biology, Biochemistry, Medicinal Chemistry background + some knowledge of computing
Generally, formal knowledge of CS can be helpful in career growth)

© Rahul Singh