



BHARATHIDASAN UNIVERSITY

Tiruchirappalli- 620024

Tamil Nadu, India.

Programme: M.Sc. Statistics

Course Title: Statistical Methods for Bioinformatics

Course Code: 23ST08DEC

Unit-II

Bio Computing

Dr. T. Jai Sankar
Associate Professor and Head
Department of Statistics

Ms. I. Angel Agnes Mary
Guest Faculty
Department of Statistics

UNIT – II

BIO COMPUTING

String Matching Algorithms

String matching algorithms have greatly influenced computer science and play an essential role in various real-world problems. It helps in performing time-efficient tasks in multiple domains. These algorithms are useful in the case of searching a string within another string. String matching is also used in the Database schema, Network systems. Let us look at a few string matching algorithms before proceeding to their applications in real world. String Matching Algorithms can broadly be classified into two types of algorithms

- Exact String Matching Algorithms
- Approximate String Matching Algorithms

Exact String Matching Algorithms:

Exact string matching algorithms is to find one, several, or all occurrences of a defined string (pattern) in a large string (text or sequences) such that each matching is perfect. All alphabets of patterns must be matched to corresponding matched subsequence.

These are further classified into four categories:

1. Algorithms based on character comparison

- **Naive Algorithm:** It slides the pattern over text one by one and check for a match. If a match is found, then slides by 1 again to check for subsequent matches.
- **KMP (Knuth Morris Pratt) Algorithm:** The idea is whenever a mismatch is detected, we already know some of the characters in the text of the next window. So, we take advantage of this information to avoid matching the characters that we know will anyway match.
- **Boyer Moore Algorithm:** This algorithm uses best heuristics of Naive and KMP algorithm and starts matching from the last character of the pattern.
- **Using the Trie data structure:** It is used as an efficient information retrieval data structure. It stores the keys in form of a balanced BST.

2. Deterministic Finite Automaton (DFA) method

Automaton Matcher Algorithm: It starts from the first state of the automata and the first character of the text. At every step, it considers next character of text, and looks for the next state in the built finite automata and move to a new state.

3. Algorithms based on Bit (parallelism method)

Aho-Corasick Algorithm: It finds all words in $O(n + m + z)$ time where n is the length of text and m be the total number characters in all words and z is total number of occurrences of words in text. This algorithm forms the basis of the original Unix command `fgrep`.

4. Hashing-string matching algorithms

Rabin Karp Algorithm: It matches the hash value of the pattern with the hash value of current substring of text, and if the hash values match then only it starts matching individual characters.

Approximate String Matching Algorithms:

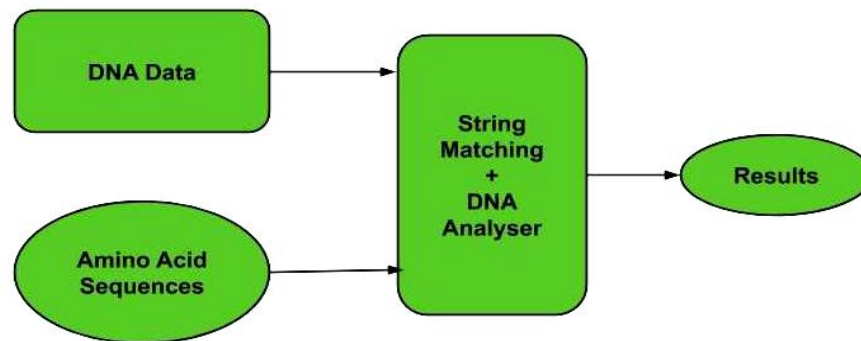
Approximate String Matching Algorithms (also known as Fuzzy String Searching) searches for substrings of the input string. More specifically, the approximate string matching approach is stated as follows: Suppose that we are given two strings, text $T[1 \dots n]$ and pattern $P[1 \dots m]$. The task is to find all the occurrences of patterns in the text whose edit distance to the pattern is at most k . Some well known edit distances are – Levenshtein edit distance and Hamming edit distance.

These techniques are used when the quality of the text is low, there are spelling errors in the pattern or text, finding DNA subsequences after mutation, heterogeneous databases, etc. Some approximate string matching algorithms are:

- **Naive Approach:** It slides the pattern over text one by one and checks for approximate matches. If they are found, then slides by 1 again to check for subsequent approximate matches.
- Sellers Algorithm (Dynamic Programming)
- Shift or Algorithm (Bitmap Algorithm)

Bioinformatics and DNA Sequencing:

Bioinformatics involves applying information technology and computer science to problems involving genetic sequences to find DNA patterns. String matching algorithms and DNA analysis are both collectively used for finding the occurrence of the pattern set.



Database Search Techniques

There are many methods for sequence searching. By far the most well known are the BLAST suite of programs. One can easily obtain versions to run locally (either at NCBI or Washington University), and there are many web pages that permit one to compare a protein or DNA sequence against a multitude of gene and protein sequence databases. To name just a few:

- National Center for Biotechnology Information (USA) Searches
- European Bioinformatics Institute (UK) Searches
- BLAST search through SBASE (domain database; ICGEB, Trieste)
- Others too numerous to mention.

One of the most important advances in sequence comparison recently has been the development of both gapped BLAST and PSI-BLAST (position specific iterated BLAST). Both of these have made BLAST much more sensitive, and the latter is able to detect very remote homologues by taking the results of one search, constructing a *profile* and then using this to search the database again to find other homologues (the process can be repeated until no new sequences are found). It is essential that one compares any new protein sequence to the database with PSI-BLAST to see if known structures can be found prior to doing any of the other methods discussed in the next sections.

Other methods for comparing a single sequence to a database include:

- The FASTA suite (William Pearson, University of Virginia, USA)
- SCANPS (Geoff Barton, European Bioinformatics Institute, UK)
- BLITZ (Compugen's fast Smith Waterman search)

It is also possible to use multiple sequence information to perform more sensitive searches. Essentially this involves building a *profile* from some kind of multiple sequence alignment. A profile essentially gives a score for each type of amino acid at each position in the sequence, and generally makes searches more sensitive. Tools for doing this include:

- PSI-BLAST (NCBI, Washington)
- ProfileScan Server (ISREC, Geneva)
- HMMER Hidden Markov Model searching (Sean Eddy, Washington University)
- Wise package (Ewan Birney, Sanger Centre; this is for protein versus DNA comparisons)
- Several others.

A different approach for incorporating multiple sequence information into a database search is to use a MOTIF. Instead of giving every amino acid some kind of score at every position in an alignment, a motif ignores all but the most invariant positions in an alignment, and just describes the key residues that are conserved and define the family. Sometimes this is called a "signature". For example, "H-[FW]-x-[LIVM]-x-G-x(5)-[LV]-H-x(3)-[DE]" describes a family of DNA binding proteins. It can be translated as "histidine, followed by either a phenylalanine or tryptophan, followed by an amino acid (x), followed by leucine, isoleucine, valine or methionine, followed by any amino acid (x), followed by glycine,... [etc.]".

PROSITE (ExpASy Geneva) contains a huge number of such patterns, and several sites allow you to search these data:

- ExpASy
- EBI

It is best to search a few different databases in order to find as many homologues as possible. A very important thing to do, and one which is sometimes overlooked, is to compare any new sequence to a database of sequences for which 3D structure information is available. Whether or not your sequence is homologous to a protein of known 3D structure is not obvious in the output from many searches of large sequence databases. Moreover, if the homology is weak, the similarity may not be apparent at all during the search through a larger database.

One last thing to remember is that one can save a lot of time by making use of pre-prepared protein alignments. Many of these alignments are hand edited by experts on the particular protein families, and thus represent probably the best alignment one can get given the data they contain (i.e. they are not always as up to date as the most recent sequence databases). These databases include:

- SMART (Oxford/EMBL)
- PFAM (Sanger Centre/Wash-U/Karolinska Intitutet)
- COGS (NCBI)
- PRINTS (UCL/Manchester)
- BLOCKS (Fred Hutchinson Cancer Research Centre, Seattle)
- SBASE (ICGEB, Trieste)

Sequence Comparison and Alignment Techniques

Sequence alignment is the process of comparing and detecting similarities between biological sequences. What “similarities” are being detected will depend on the goals of the particular alignment process. Sequence alignment appears to be extremely useful in a number of bioinformatics applications.

For example, the simplest way to compare two sequences of the same length is to calculate the number of matching symbols. The value that measures the degree of sequence similarity is called the alignment score of two sequences. The opposite value, corresponding to the level of dissimilarity between sequences is usually referred to as the distance between sequences. The number of non-matching characters is called the Hamming distance. Fig. 1 shows an example of two sequences with Hamming distance equal to 3.

ATTCGGATCTCA
ATTCGGCACTGA

Fig. 1. Example of two sequences with Hamming distances equal to 3

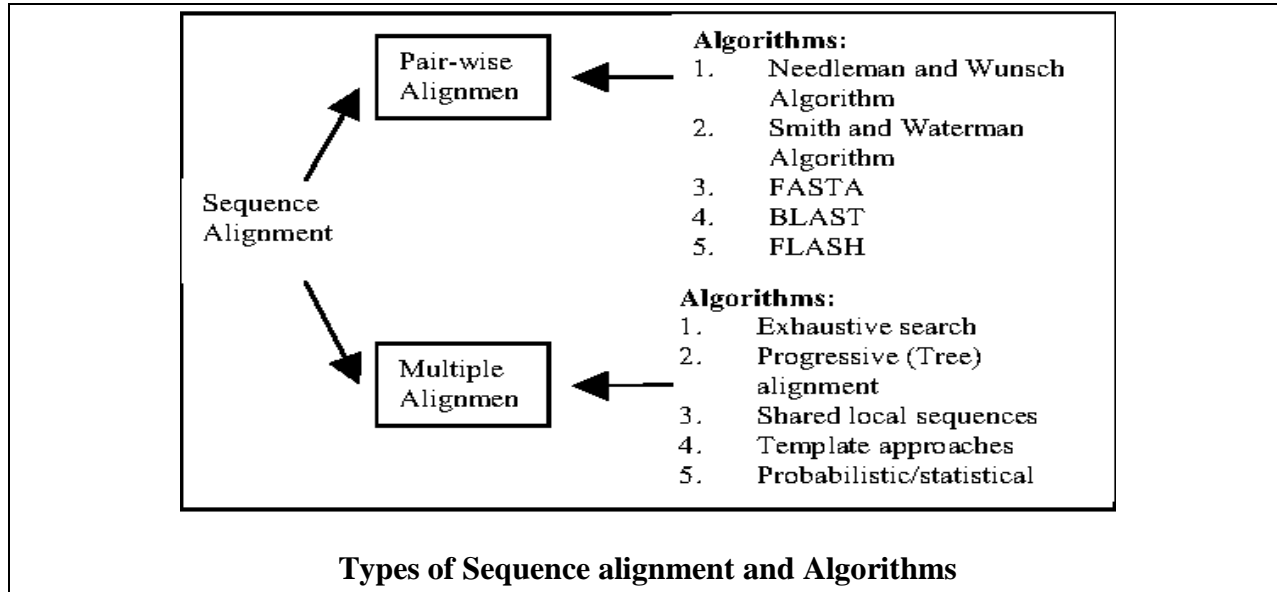
It is however, worth noting that comparing sequence characters position by position as described above can barely be referred to as alignment process; since it does not take into account such typical biological events as deletions and insertions. The classical notion of sequence alignment includes calculating the so called edit distance, which generally corresponds to the minimal number of substitution, insertions and deletions needed to turn one sequence into another. Fig. 2 demonstrates an example of two sequences with edit distance equal to 3.

ATTC-GATCTCA
ATTCGGAACT-A

Fig. 2. Example of two sequences with edit distances equal to 3

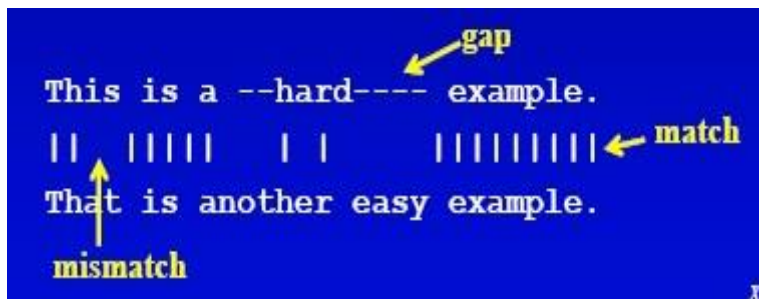
The problems of computing edit distance and various types of sequence alignment have exact solutions, e.g., (Smith and Waterman, 1981) and (Needleman and Wunsch, 1970) algorithms. Since these algorithms were initially developed for protein-protein alignment and later adapted for DNA sequence alignment, they are described in the section 'Protein-protein alignment'. In most real-life cases, however, these algorithms appear to be impractical for DNA alignment due to their running time and memory requirements.

Types of Sequence alignment Techniques



Pair-wise sequence alignment

Pair-wise sequence alignment is one form of sequence alignment technique, where we compare only **two sequences**. This process involves finding the optimal alignment between the two sequences, scoring based on their similarity (how similar they are) or distance (how different they are), and then assessing the significance of this score.



Scoring

Before moving on to the pair-wise sequence alignment techniques, let's go through the process of scoring.

The basis of sequence alignment lies with the scoring process, where the two sequences are given a score on how similar (or different) they are to each other. The pair-wise sequence aligning algorithms require a **scoring matrix** to keep track of the scores assigned. The scoring matrix assigns a positive score for a match, and a penalty for a mismatch.

Three basic aspects are considered when assigning scores. They are,

1. **Match value** — Value assigned for matching characters
2. **Mismatch value** — Value assigned for mismatching characters
3. **Gap penalty** — Value assigned for spaces

The Figure 3 given below shows how you can identify a match, mismatch and a gap among two sequences.

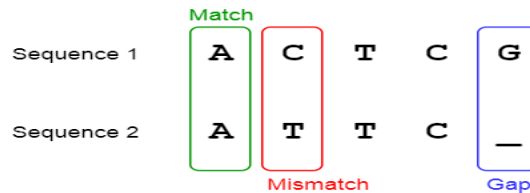


Figure 3: Match, mismatch and gap

The first character of the two sequences is a **match**, as both are letter **A**. Second character of the first sequence is **C** and that of the second sequence is **T**. So, it is **mismatch**.

A space is introduced at the end of the second sequence to match with G. This space is known as a **gap**. A gap is the maximal contiguous run of spaces in a single sequence within a given alignment.

Affined gap penalties impose an 'opening' penalty for a gap and an 'extension' penalty that decreases the relative penalty for each additional position in an already opened gap. In general, a gap is expressed as a **gap penalty function** which is a function that measures the cost of a gap as a (possibly nonlinear) function of its length.

Commonly used substitution Score matrices are:

(i) Point Accepted Mutation matrix (PAM)

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

(ii) BLOcks SUBstitution Matrix (BLOSUM)

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

Types of Pair-wise Sequence Alignment

Consider that you are given two sequences as below.

$$X = X_1 \dots X_i \dots X_n$$

$$Y = y_1 \dots y_j \dots y_m$$

1. **Global alignment:** This method finds the best alignment over the entire lengths of the 2 sequences. What is the maximum similarity between sequence **X** and **Y**?
2. **Local alignment:** This method finds the most similar subsequences among the 2 sequences. What is the maximum similarity between a subsequence of **X** and a subsequence of **Y**?

Global Alignment

In this method, we consider the entire length of the 2 sequences and try to match them to obtain the best alignment. It is obtained by inserting **gaps** (spaces) to X and Y until the length of the two sequences will be the same so that the two sequences are matched.

For example, consider the sequences X = **ACGCTGAT** and Y = **CAGCTAT**. One possible global alignment is,

```
AC-GCTGAT
 |  | |  |
-CAGC-TAT
```

Not that we have included **gaps** so that the strings are aligned.

If we set a scoring scheme as match score = 1, mismatch score = 0 and gap penalty = 0, then the overall score for the above alignment will be,

$$\begin{aligned} \text{Score} &= n\text{Match} \times 1 + n\text{Mismatch} \times 0 + n\text{Gap} \times 0 \\ &= 6 \times 1 + 1 \times 0 + 2 \times 0 \\ &= 6 \end{aligned}$$

Needleman-Wunsch Algorithm

One of the algorithms that uses dynamic programming to obtain global alignment is the Needleman-Wunsch algorithm. This algorithm was published by Needleman and Wunsch in 1970 for alignment of two protein sequences and it was the first application of dynamic programming to biological sequence analysis. The Needleman-Wunsch algorithm finds the best-scoring global alignment between two sequences.

Local Alignment

In this method, we consider subsequences within each of the 2 sequences and try to match them to obtain the best alignment.

For example, consider 2 sequences as X=**GGTCTGATG** and Y=**AAACGATC**. Characters in bold are the subsequences to be considered. The best local alignment is,

```
CTGAT (in X)
 |  | |
C-GAT (in Y)
```

Here, one **gap** is introduced in order to match the 2 subsequences.

If we set a scoring scheme as match score = 1, mismatch score = 0 and gap penalty = 0, then the overall score for the above alignment will be,

$$\begin{aligned}
\text{Score} &= n\text{Match} \times 1 + n\text{Mismatch} \times 0 + n\text{Gap} \times 0 \\
&= 4 \times 1 + 0 \times 0 + 1 \times 0 \\
&= 4
\end{aligned}$$

Smith-Waterman Algorithm

One of the algorithms that uses dynamic programming to obtain local alignments within two given sequences is the **Smith-Waterman algorithm**. **Smith** and **Waterman** published an application of dynamic programming to find the optimal local alignments in 1981. This algorithm is similar to Needleman-Wunsch algorithm, but there are slight differences with the scoring process.

Multiple Sequence Alignment

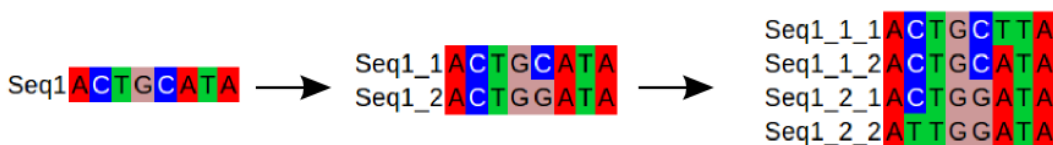
A Multiple Sequence Alignment is an alignment of more than two sequences. We could align several DNA or protein sequences.



Some of the most usual uses of the multiple alignments are:

- Phylogenetic analysis
- Conserved domains
- Protein structure comparison and prediction
- Conserved regions in promoters

The multiple sequence alignment assumes that the sequences are homologous, they descend from a common ancestor. The algorithms will try to align homologous positions or regions with the same structure or function.



Multiple alignment algorithms

Multiple alignments are computationally much more difficult than pair-wise alignments. It would be ideal to use an analog of the Smith & Waterman algorithm capable of looking for optimal alignments in the diagonals of a multidimensional matrix given a scoring schema. This algorithm would have to create a multidimensional matrix with one dimension for each sequence. The memory and time required for solving the problem would increase geometrically with the length of every sequence. Given the number of sequences usually involved no algorithm is capable of doing that. Every algorithm available reverts to a heuristic capable of solving the problem in a much faster time. The drawback is that the result might not be optimal.

Usually the multiple sequence algorithms assume that the sequences are similar in all its length and they behave like global alignment algorithms. They also assume that there are not many long insertions and deletions. Thus the algorithms will work for some sequences, but not for others.

These algorithms can deal with sequences that are quite different, but, as in the pair-wise case, when the sequences are very different they might have problems creating good alignment. A good algorithm should align the homologous positions or the positions with the same structure or function.

If we are trying to align two homologous proteins from two species that are phylogenetically very distant we might align quite easily the more conserved regions, like the conserved domains, but we will have problems aligning the more different regions. This was also the case in the pair-wise case, but remember that the multiple alignment algorithms are not guaranteed to give back the best possible alignment.

These algorithms are not designed to align sequences that do not cover the whole region, like the reads from a sequencing project. There are other algorithms to assemble sequencing projects.

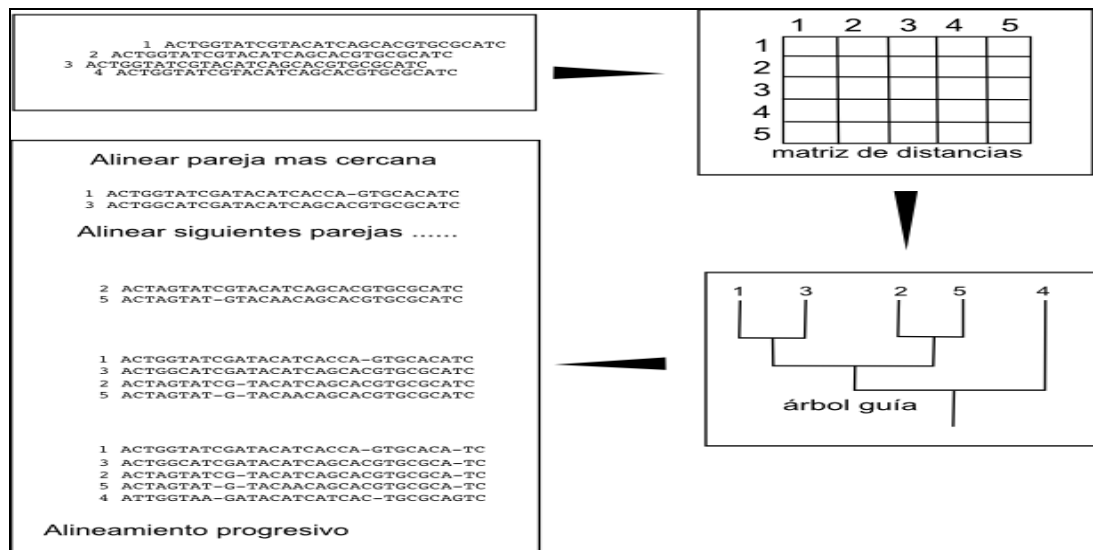
Progressive contraction algorithms

In Multiple Sequence Alignment it is quite common that the algorithms use a progressive alignment strategy. These methods are fast and allow aligning thousands of sequences.

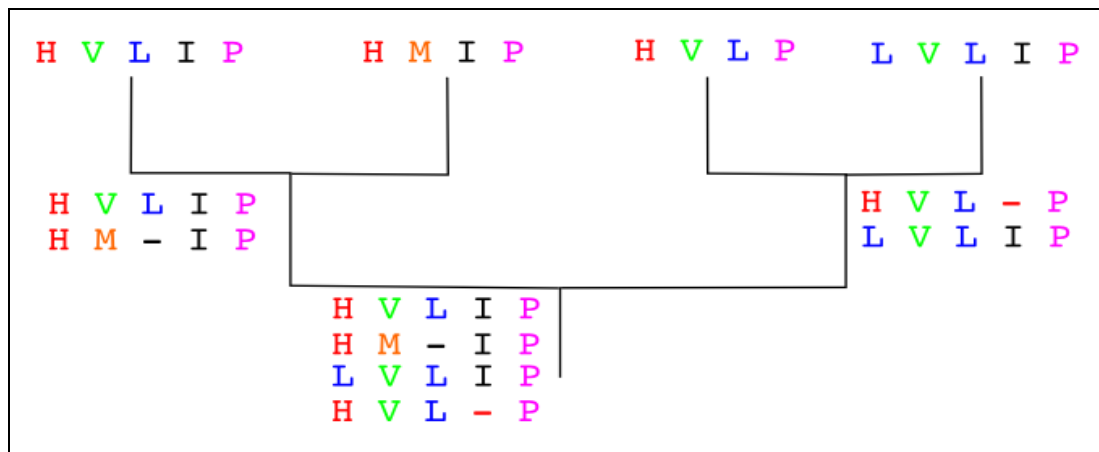
Before starting the alignment, as in the pair-wise case, we have to decide which is the scoring schema that we are going to use for the matches, gaps and gap extensions. The aim of the alignment would be to get the multiple sequence alignment with the highest score possible. In the multiple alignment case we do not have any practical algorithm that guarantees that it is going to get the optimal solution, but we hope that the solution will be close enough if the sequences comply with the restrictions assumed by the algorithm.

The idea behind the progressive construction algorithm is to build the pair-wise alignments of the more closely related sequences, which should be easier to build, and to align progressively these alignments once we have them. To do it we need first to determine which the closest sequence pairs are. One rough and fast way of determining which the closest sequence pairs are is to align all the possible pairs and look at the scores of those alignments. The pair-wise alignments with the highest scores should be the ones between the more similar sequences. So the first step in the algorithm is to create all the pair-wise alignments and to create a matrix with the scores between the pairs. These matrixes will include the similarity relations between all sequences.

Once we have this matrix we can determine the hierarchical relation between the sequences, which are the closest pairs and how those pairs are related and so on, by creating a hierarchical clustering, a tree. We can create these trees by using different fast algorithms like UPGMA or Neighbor joining. These trees are usually known as guide trees.



An example:



Example:

Sequences:

IMPRESIONANTE

INCUESTIONABLE

IMPRESO

Scores:

IMPRESIONANTE X IMPRESO 7/13

IMPRESIONANTE X INCUESTIONABLE 10/14

INCUESTIONABLE X IMPRESO 4/14

Scoring pair-wise matrix:

	IMPRESIONANTE	INCUESTIONABLE	IMPRESO
IMPRESIONANTE	1	10/14	7/13
INCUESTIONABLE	10/14	1	4/14
IMPRESO	7/13	4/14	1

Guide Tree:

```
    |--- IMPRESIONANTE
    |---|--- INCUESTIONABLE
    |
    |----- IMPRESO
```

The first alignment would be: IMPRESIONANTE x INCUESTIONABLE

IMPRES-IONABLE

INCUESTIANABLE

Now we align IMPRESO to the previous alignment.

IMPRES-IONANTE

INCUESTIONABLE

IMPRES--O-----

We have no guarantee that the final is the one with the highest score.

The main problem of these progressive alignment algorithms is that the errors introduced at any point in the process are not revised in the following phases to speed up the process. For instance, if we introduce one gap in the first pair-wise alignment this gap will be propagated to all the following alignments. If the gap was correct that is fine, but if it was not optimal it won't be fixed. These methods are especially prone to fail when the sequences are very different or phylogenetically distant.

Sequences to align already in the order given by a guide tree:

Seq A GARFIELD THE LAST FAT CAT

Seq B GARFIELD THE FAST CAT

Seq C GARFIELD THE VERY FAST CAT

Seq D THE FAT CAT

Step 1

Seq A GARFIELD THE LAST FAT CAT

Seq B GARFIELD THE FAST CAT

Step 2

Seq A GARFIELD THE LAST FA-T CAT

Seq B GARFIELD THE FAST CA-T

Seq C GARFIELD THE VERY FAST CAT

Step 3

Seq A GARFIELD THE LAST FA-T CAT

Seq B GARFIELD THE FAST CA-T

Seq C GARFIELD THE VERY FAST CAT

Seq D ----- THE ---- FA-T CAT

Historically the most used of the progressive multiple alignment algorithms was CLUSTALW. Nowadays CLUSTALW is not one of the recommended algorithms anymore because there are other algorithms that create better alignments like Clustal Omega or MAFFT. MAFFT was one of the best contenders in a multiple alignment software comparison.

T-Coffee is another progressive algorithm. T-Coffee tries to solve the errors introduced by the progressive methods by taking into account the pair-wise alignments. First it creates a library of all the possible pair-wise alignments plus a multiple alignment using an algorithm similar to the CLUSTALW one. To this library we can add more alignments based on extra information like the protein structure or the protein domain composition. Then it creates a progressive alignment, but it takes into account all the alignments in the library that relate to the sequences aligned at that step to avoid errors. The T-Coffee algorithm follows the steps:

- Create the pair-wise alignments
- Calculate the similarity matrix
- Create the guide tree
- Build the multiple progressive alignment following the tree, but taking into account the information from the pair-wise alignments.

T-Coffee is usually better than CLUSTALW and performs well even with very different sequences, especially if we feed it more information, like: domains, structures or secondary structure. T-Coffee is slower than CLUSTALW and that is one of its main limitations, it can not work with more than few hundred sequences.

Iterative algorithms

These methods are similar to the progressive ones, but in each step the previous alignments are reevaluated. Some of the most popular iterative methods are: Muscle and MAFFT are two popular examples of these algorithms.

Hidden Markov models

The most advanced algorithms to date are based on Hidden Markov Models and they have improvements in the guide tree construction, like the sequence embedding that reduce the computation time.

Clustal Omega is one of these algorithms and can create alignments as accurate of the T-Coffee, but with many thousands of sequences.

Alignment evaluation

Once we have created our Multiple Sequence Alignment we should check that the result is OK. We could open the multiple alignments in a viewer to assess the quality of the different regions of the alignment or we could automate this assessment. Usually not all the regions have an alignment of the same quality. The more conserved regions will be more easily aligned than the more variable ones.

Introduction to Graph matching algorithm

Matching algorithms are algorithms used to solve graph matching problems in graph theory. A matching problem arises when a set of edges must be drawn that do not share any vertices.

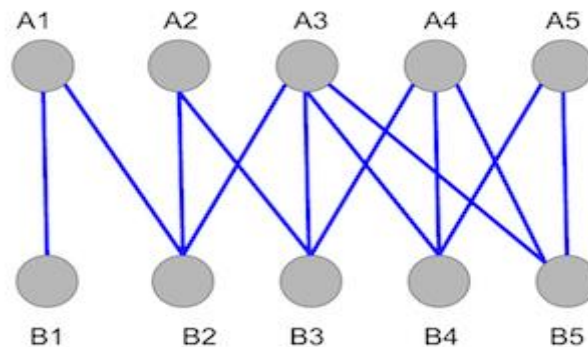
Graph matching problems are very common in daily activities. From online matchmaking and dating sites, to medical residency placement programs, matching algorithms are used in areas spanning scheduling, planning, pairing of vertices and network flows. More specifically, matching strategies are very useful in flow network algorithms such as the Ford-Fulkerson algorithm and the Edmonds-Karp algorithm.

Graph matching problems generally consist of making connections within graphs using edges that do not share common vertices, such as pairing students in a class according to their respective qualifications; or it may consist of creating a **bipartite matching**, where two subsets of vertices are distinguished and each vertex in one subgroup must be matched to a vertex in another subgroup.

Alternating and Augmenting Paths

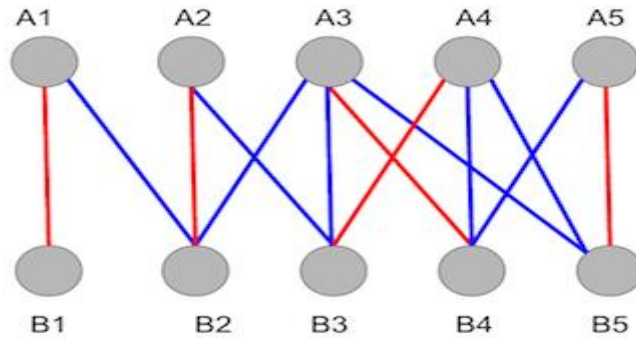
Graph matching algorithms often use specific properties in order to identify sub-optimal areas in a matching, where improvements can be made to reach a desired goal. Two famous properties are called augmenting paths and alternating paths, which are used to quickly determine whether a graph contains a maximum, or minimum, matching, or the matching can be further improved.

Graph 1 shows all the edges, in blue, that connect the bipartite graph. The goal of a matching algorithm, in this and all bipartite graph cases, is to maximize the number of connections between vertices in subset *A*, above, to the vertices in subset *B*, below.



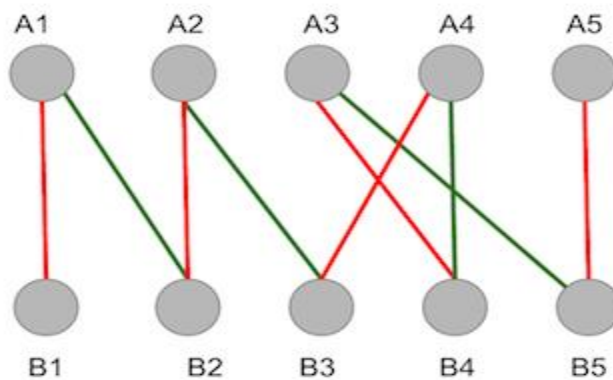
Graph 1. Unmatched bipartite graph

Most algorithms begin by randomly creating a matching within a graph, and further refining the matching in order to attain the desired objective.



Random initial matching, M , of Graph 1 represented by the red edges

Graph 1, with the matching, M , is said to have an alternating path if there is a path whose edges are in the matching, M , and not in the matching, in an alternating fashion. An alternating path usually starts with an unmatched vertex and terminates once it cannot append another edge to the tail of the path while maintaining the alternating sequence.



An alternating path in Graph 1 is represented by red edges, in M , joined with green edges, not in M .

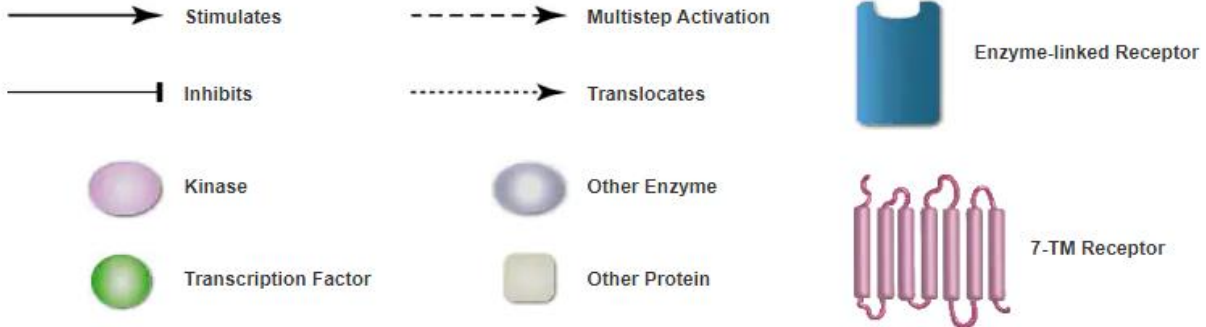
An augmenting path, then, builds up on the definition of an *alternating path* to describe a path whose endpoints, the vertices at the start and the end of the path, are free, or *unmatched*, vertices; vertices not included in the matching. Finding augmenting paths in a graph signals the lack of a maximum matching.

Signaling Pathways

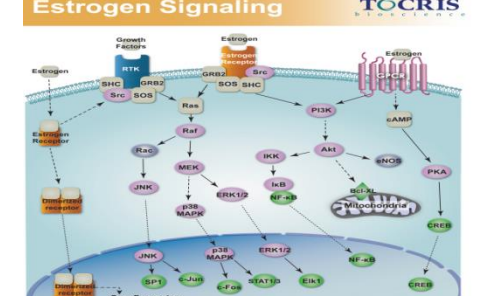
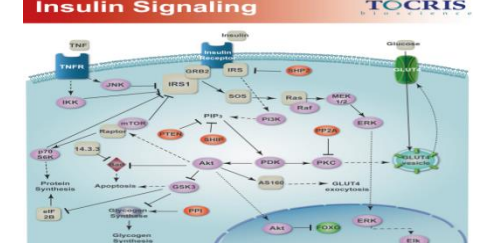
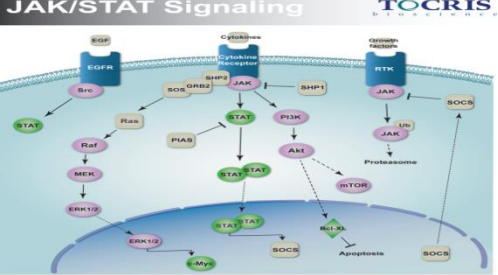
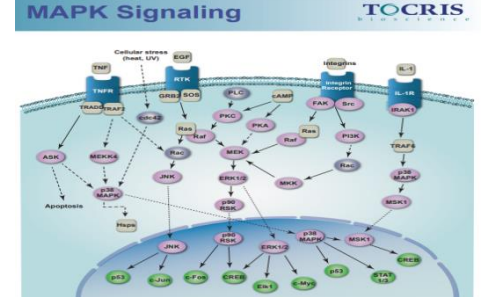
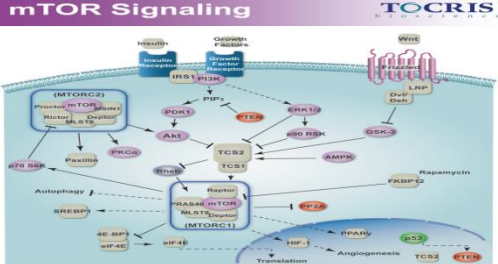
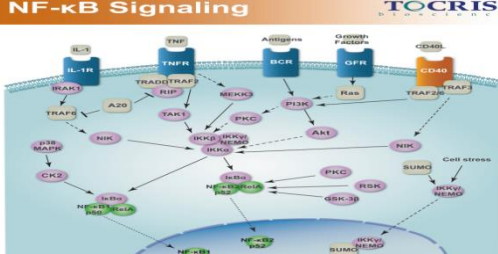
Interactive signal transduction pathways

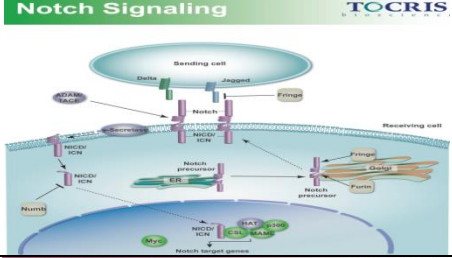
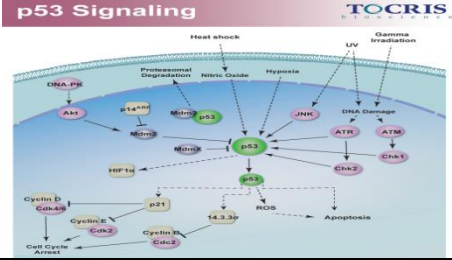
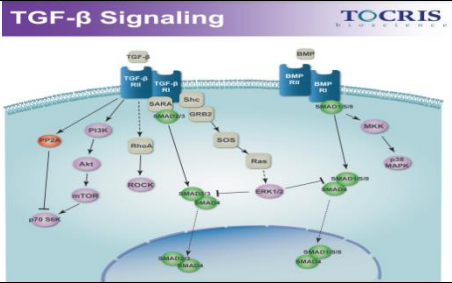
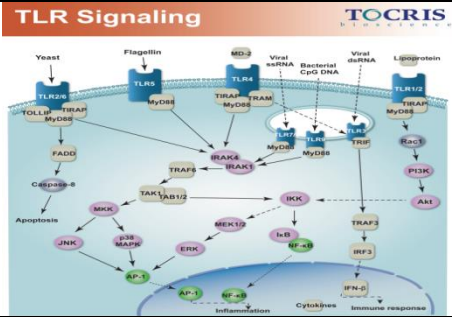
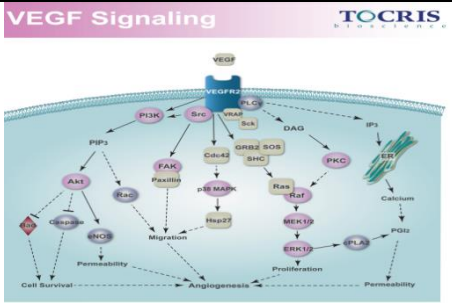
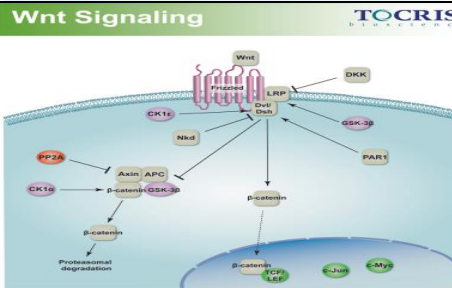
Our interactive pathways give an overview of some major signal transduction processes, and provide an alternative way to browse the Tocris website. Click on the individual proteins and receptors to locate key research tools and view the range of products Tocris has to offer.

Signaling Pathway Key



Signaling Pathway	Meaning	Graph
Akt Signaling Pathway	The Akt signaling pathway plays a key role in the mediation of protein synthesis, metabolism, proliferation and cell cycle progression. It may be referred to as a 'pro-survival' pathway.	
AMPK signaling pathway	AMPK signaling pathway plays an important role in the cellular response to low levels of available ATP, often caused by stresses such as heat shock, ischemia or hypoxia.	
Apoptosis Signaling Pathway	Apoptosis is a physiological process for cell death that is critical during aging and development. It may also be referred to as cell "suicide". Apoptosis can be triggered by events both inside and outside of the cell.	

<p>Estrogen Signaling Pathway</p>	<p>Estrogen is a steroid hormone that is responsible for the regulation of growth, differentiation and function of the reproductive system. Estrogen signaling is often dysregulated in breast cancer and osteoporosis.</p>	
<p>Insulin Signaling Pathway</p>	<p>Signaling through the insulin pathway is fundamental for the regulation of intracellular glucose levels. This pathway can become dysregulated in diabetes.</p>	
<p>JAK-STAT Signaling Pathway</p>	<p>The JAK-STAT signaling pathway has several roles, including the control of cell proliferation and hematopoiesis. It is the main signal transduction cascade from cytokine receptors.</p>	
<p>MAPK Signaling Pathway</p>	<p>The mitogen-activated protein kinase pathway evokes an intracellular signaling cascade in response to extracellular stimuli such as heat and stress. It can influence cell division, metabolism and survival.</p>	
<p>mTOR Signaling Pathway</p>	<p>mTOR is a serine/threonine kinase that nucleates at multiprotein complexes mTORC1 and mTORC2. Signaling by these complexes regulates cell growth, proliferation and survival.</p>	
<p>NF-kB Signaling Pathway</p>	<p>NF-kB signaling plays an important role in inflammation, the innate and adaptive immune response and stress. Dysregulated signaling can occur in inflammatory and autoimmune diseases.</p>	

<p>Notch Signaling Pathway</p>	<p>The Notch pathway is involved in determination of cell fate, regulation of pattern formation and other developmental settings. Disrupted signaling can cause developmental defects and a range of adult pathologies.</p>	
<p>p53 Signaling Pathway</p>	<p>p53 signaling plays an important role in the co-ordination of the cellular response different types of stress such as DNA damage and hypoxia. The downstream signals lead to apoptosis, senescence and cell cycle arrest.</p>	
<p>TGF-β Signaling Pathway</p>	<p>The TGF-β signaling pathway is involved in the regulation of growth and proliferation of cells along with migration, differentiation and apoptosis.</p>	
<p>Toll-like Receptor Signaling Pathway</p>	<p>TLR signaling is involved in the cellular response to threatening molecules such as bacteria and viruses. It results in an inflammatory and immunological response.</p>	
<p>VEGF Signaling Pathway</p>	<p>VEGF signaling pathway is involved in embryonic vascular development (vasculogenesis) and in the formation of new blood vessel (angiogenesis). It also induces cell migration, proliferation and survival.</p>	
<p>Wnt Signaling Pathway</p>	<p>The Wnt pathway is involved in cellular differentiation and proliferation in adult tissues and also during embryogenesis. Disturbances within the pathway may lead to the formation of tumors and promote metastasis.</p>	

Gene array

Gene arrays are solid supports upon which a collection of gene-specific nucleic acids have been placed at defined locations, either by spotting or direct synthesis. In array analysis, a nucleic acid-containing sample is labeled and then allowed to hybridize with the gene-specific targets on the array. Based on the amount of probe hybridized to each target spot, information is gained about the specific nucleic acid composition of the sample. The major advantage of gene arrays is that they can provide information on thousands of targets in a single experiment.

Currently the solid supports upon which nucleic acids are arrayed are either glass slides or nylon membranes. Typically, fluorescently labeled probes are used with glass arrays, while radio labeled probes are used with membranes. Depending on the type of array, the arrayed nucleic acids may be composed of oligonucleotides, PCR products or cDNA vectors or purified inserts. The sequences may represent entire genomes and may include both known and unknown sequences or may be collections of sequences such as apoptosis-related genes or cytokines. Many pre-made and custom arrays are available from commercial manufacturers although many labs prepare their own arrays with the help of robotic arrayers. The methods of probe labeling, hybridization, and detection depend on the solid support to which the sequences are bound.

Gene Arrays for Expression Analysis

Gene arrays have become a powerful approach for comparing complex sample RNA populations. Using array analysis, the expression profiles of normal and tumor tissues, treated and untreated cell cultures, developmental stages of an organism or tissue, and different tissues can be compared. A typical gene array experiment involves:

1. Isolating RNA from the samples to be compared
2. Converting the RNA samples to labeled cDNA via reverse transcription; this step may be combined with aRNA amplification
3. Hybridizing the labeled cDNA to identical membrane or glass slide arrays
4. Removing the unhybridized cDNA
5. Detecting and quantitating the hybridized cDNA, and
6. Comparing the quantitative data from the various samples

Data Analysis

Where once the bottleneck in gene expression analysis was the bench work, with array analysis, it is the computer work. Because a single array experiment can generate thousands of data points, the primary challenge of the technique is making sense of the data. Many commercial companies provide image analysis software, including Bio Discovery (ImaGene) and Imaging Research (Array Vision). Furthermore, many array manufacturers offer software specifically for the analysis of their arrays and offer the analysis as a service.

For membrane array analysis, a file of the data is generated by phosphoimaging and that file is then analyzed using software. The software will correlate spots to genes and can compare spot intensities for differential expression studies.

Glass array data is treated in much the same way, but the image's fluorescence is scanned and the software allows detection of each samples' fluorescence individually or simultaneously for analysis. Most software packages can analyze several arrays simultaneously.