BCH 5425 Molecular Biology Dr. Hong Li

Lecture 25 (Iodish section 7.4) **Bioinformatics Survey** lecture notes
Guest Lecturer: Steve Thompson (CSIT)

Introduction

It's a brand new field in the last twenty years or so, called various, often misunderstood names, that are largely subsets of one another — computational molecular biology, biocomputing, sequence analysis, and now "bioinformatics," "genomics" and "proteomics." But what does it mean? One way to think about computational biology is the reverse biochemistry analogy — biochemists no longer have to begin a research project by isolating and purifying massive amounts of a protein from its native organism in order to characterize a particular gene product. Rather, now scientists can amplify a section of some genome based on its similarity to other genomes, sequence that piece of DNA, and, **using sequence analysis tools, infer all sorts of functional, evolutionary, and, perhaps, structural insight into a gene** within it, and then, perhaps, go on to cloning that gene, expressing the gene product, and finally purifying the protein. The process has come full circle. The computer has become an important tool to be used at the beginning and throughout a research project in assisting experimental design, not just a number-cruncher used at the end of the process.

Definitions — Much confusion abounds in the area, even concerning the names of the disciplines themselves. The terms computational biology, biocomputing, bioinformatics, sequence analysis, molecular modelling, genomics, and proteomics are often bantered about with little regard to what they really mean. All are interdisciplinary by nature, combining elements of computer and information science, mathematics and statistics, and chemistry and biology. Each has elements of one another. **Biocomputing and computational biology** are the most encompassing terms and can be considered synonyms. They both **describe using computers and computational techniques to analyze a biological system**, whether that is a biomolecular primary sequence or tertiary structure, or a metabolic pathway, or even a complex system such as the interactions of populations within an ecological niche.

Bioinformatics necessarily intersects with this concept in that it **describes using computational techniques to access, analyze, and interpret the biological information in databases.** However, these databases can be the traditionally considered nucleic and amino acid sequence databases as well as three-dimensional molecular structure databases, but can even include such disparate data collections as medical records or population statistics. Therefore, bioinformatics is a type of biocomputing but also includes topics such as medical informatics that is not usually considered a part of computational biology.

Within bioinformatics the subdiscipline of **sequence analysis** has a clearly defined scope. It is the study of biological molecular sequence data for the purpose of **inferring the function, interactions, evolution, and perhaps structure of biological molecules**. Molecular modelling can also be considered a type of bioinformatics, though it often isn't. It is necessarily a subdiscipline of computational structural biology, but uses the methodology and techniques of that discipline as well sequence analysis' similarity searching and alignment algorithms. That is why it is often referred to as "homology modelling." Genomics can be considered the subdiscipline of bioinformatics that is concerned not with individual molecular sequences, but rather with sequences on a genomic scale. That is, **genomics analyzes the context of genes or complete genomes (the total DNA content of an organism) within and across genomes**. **Proteomics can be considered the subdivision of genomics concerned with analyzing the complete protein complement, i.e. the proteome, of organisms, both within and between different organisms**. The comparative method is crucial to all these methods but, perhaps most obvious and key to genomics and proteomics.

A good example of the type of knowledge available with these techniques are the two organisms *Mycoplasma genitalium*, *Methanococcus jannaschi*. These two organisms had not been studied much, yet upon publication of their complete genome most of their open reading frames were assigned putative functions based on comparisons to *Escherichia coli* which has been extensively studied. A great deal can

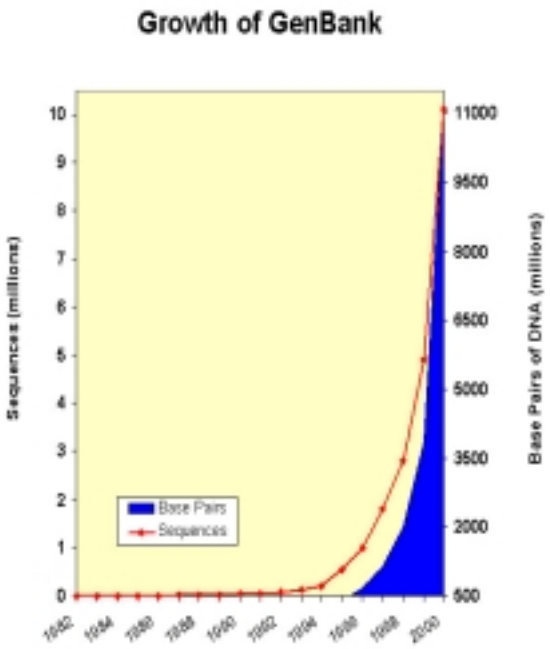
be learned about an organism just from studying its genome. For instance, *M. genitalium* does not code for any of the enzymes involved in the tricarboxylic acid cycle (TCA) nor the cytochromes required for oxidative phosphorylation; it does encode the enzymes required for glycolysis of glucose to lactate and acetate. This implies that this organism most likely produces ATP principally from the comparatively simple and low-yield glycolytic pathway. Eight Archae[bacteria] and twenty nine [Eu]Bacterial complete, annotated genomes are currently in GenBank (GenBank version 121, Dec. 15, 2000). Three eukaryotic genomes are completely finished and publicly available, *Saccharomyces cerevisiae*, *Caenorhabditis elegans*, and *Drosophila melanogaster*. *Arabidopsis thaliana* and *Homo sapiens* have been completed but are not yet entirely publicly available through GenBank. Over half of the genes in many of these organisms have predicted functions based on previously studied bacterial genes. Eukaryotes have many more genes that code for intracellular targeting and secretion than prokaryotes do because of the complex organelles in eukaryotic cells. In order to understand genes of unpredicted function, efforts are made to inactivate each individual gene (knock out) in yeast and learn the functional consequences. These data or strains of knock-outs are also made available to public.

Structural genomics is the acquisition and analysis of the complete set of three-dimensional structure coordinate data for an organism's entire proteome (or a representative set thereof). Through analyses such as these, it may eventually be possible to predict a completely unknown protein's structure and function just based on its deduced molecular sequence. Obviously this could be an incredible boost to the drug-design process and could go a long way toward curing many disease processes. We have come a long way in structural prediction but are still a long way from this goal.

The tremendous growth of biocomputing is reflected in and largely a result of the **exponential growth of the molecular sequence databases** along with a concurrent increase in the level of computational processing power available. The Human Genome Project and numerous smaller genome projects have kept the data coming at alarming rates. As of December 2000 (GenBank version 121.0) **40 complete, finished genomes** are publicly available for analysis, not counting all the virus and viroid genomes available. The International Human Genome Sequencing Consortium announced the completion of a "**Working Draft**" of the **human genome** in June 2000; independently that same month, the private company Celera Genomics announced that it had completed the first assembly of the human genome.

The GenBank database roughly doubles every year; the growth statistics found through the URL **http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html** are staggering:

Year	BasePairs	Sequences
1982	680338	606
1983	2274029	2427
1984	3368765	4175
1985	5204420	5700
1986	9615371	9978
1987	15514776	14584
1988	23800000	20579
1989	34762585	28791
1990	49179285	39533
1991	71947426	55627
1992	101008486	78608
1993	157152442	143492
1994	217102462	215273
1995	384939485	555694
1996	651972984	1021211
1997	1160300687	1765847
1998	2008761784	2837897
1999	3841163011	4864570
2000	**11101066288**	**10106023**



So how does one do Bioinformatics?

On the Internet through the World Wide Web — possible and easy and fun, but . . . can not readily handle large datasets or large multiple alignments, quickly becomes intractable. In spite of that . . .

some of my favorite World Wide Web sites for molecular biology, sequence analysis, and bioinformatics:

Site	URL (Uniform Resource Locator)	Content
National Center Biotech' Info'	http://www.ncbi.nlm.nih.gov/	databases/analysis/software
PIR/NBRF	http://www-nbrf.georgetown.edu/	protein sequence database
Johns Hopkins BioInfo'	http://www.bis.med.jhmi.edu/bioInformatics.html	databases/analysis/software
Harvard Bio' Laboratories	http://golgi.harvard.edu/	databases/analysis/software
IUBIO Biology Archive	http://iubio.bio.indiana.edu/	database/software archive
Univ. of Montreal MegaSun	http://megasun.bch.umontreal.ca/	database/software archive
Japan's GenomeNet Server	http://www.genome.ad.jp/	databases/analysis/software
European Mol' Bio' Lab'	http://www.embl-heidelberg.de/	databases/analysis/software
European Bioinformatics Lab'	http://www.ebi.ac.uk/	databases/analysis/software
The Sanger Institute	http://www.sanger.ac.uk/	databases/analysis/software
Univ. of Geneva BioWeb	http://www.expasy.ch/	databases/analysis/software
ProteinDataBank	http://www.rcsb.org/pdb/	3D mol' structure database
Molecules R Us	http://molbio.info.nih.gov/cgi-bin/pdb	3D protein/nuc' visualization
The Genome DataBase	http://www.gdb.org/	The Human Genome Project
Stanford Genomic Resource	http://genome-www.stanford.edu/	various genome projects
Inst. for Genomic Research	http://www.tigr.org/	microbial genome projects
HIV Sequence Database	http://hiv-web.lanl.gov/	HIV epidemeology seq' DB
The Baylor Search Launcher	http://searchlauncher.bcm.tmc.edu/	sequence search launcher
Pedro's BioMol Res' Tools	http://www.public.iastate.edu/~pedro/research_tools.html	extensive bookmark list
BioToolKit	http://www.biosupplynet.com/cfdocs/btk/btk.cfm	annotated molbio tool links
Felsenstein's PHYLIP site	http://evolution.genetics.washington.edu/phylip.html	phylogenetic inference
The Tree of Life	http://phylogeny.arizona.edu/tree/phylogeny.html	overview of all phylogeny
Ribosomal Database Project	http://www.cme.msu.edu/RDP/	databases/analysis/software
WIT Metabolism	http://wit.mcs.anl.gov/WIT2	metabolic reconstructions
BIOSCI/BIONET	http://net.bio.net	biologists' news groups
Access Excellence	http://www.accessexcellence.org/	biology teaching and learning
CELLS alive!	http://www.cellsalive.com/	animated microphotography
Genetics Computer Group	http://www.gcg.com/	Wisconsin S.A. Package

So what are the alternatives . . .

Other client-server protocols such as **Network Entrez from NCBI**; use and restrictions.

Desktop software solutions — public domain programs are available, but . . . complicated to install, configure, and maintain. User must be pretty computer savvy. So,

commercial software packages are available, e.g. Omiga, MacVector, DNAsis, DNAStar, etc.,

but . . . license hassles, expense per machine, and database access issues all complicate matters!

Therefore, still other server-based solutions, either public domain (with need for a very cooperative system administrator) or commercial (e.g. the **Wisconsin Package**) — usually UNIX server computers.

One license fee for an entire institution with commercial software and very fast, convenient database access on local server disks. Connections from any networked terminal or workstation anywhere!

Operating system: command line operation hassles, communications software — telnet and terminal emulation, X graphics for GUI's, file transfer — ftp and Mac Fetch, and editors — vi, emacs, pico (or desktop word processing followed by file transfer [save as "text only!"]).

I. Databases: Content and Organization.

What are primary sequences?

Primary refers to one dimensional — all of the "symbol" information written in sequential order necessary to specify a particular biological molecular entity, be it polypeptide or nucleotide. The symbols are the one letter alphabetic codes for all of the biological nitrogenous bases and amino acid residues and their ambiguity codes. Biological carbohydrates, lipids and structural information are not included within this sequence, however, much of this type of information is available in the reference documentation sections associated with primary sequences in the databases.

What are sequence databases?

These databases are an organized way to store the tremendous amount of sequence information that accumulates from laboratories worldwide. This data is piling up at exponential rates, as seen above. Each database has its own specific formats and access to this information is most easily handled through various software packages and interfaces, either on the World Wide Web or otherwise. Three major database organizations worldwide are responsible for maintaining most of this data.

In the United States the **National Center for Biotechnology Information** (**NCBI**), a division of the National Library of Medicine (NLM), at the National Institute of Health (NIH), supports and distributes the **GenBank** nucleic acid sequence database and **GenPept** CDS (Coding Sequence) translations database. The National Biomedical Research Foundation, an affiliate of Georgetown University Medical Center, maintains the **Protein Identification Resource** (**PIR**) database of polypeptide sequences.

The **European Molecular Biology Laboratory** maintains the **EMBL** nucleic acid sequence database and the excellently annotated **Swiss-Prot** protein sequence database (also supported by the Swiss Institute of Bioinformatics: SIB), as well as the minimally annotated **TrEMBL** (Translations from EMBL — those EMBL translations not in Swiss-Prot) protein sequence databases in Cambridge, UK; Heidelberg, Germany; and Geneva, Switzerland. Additional, lesser-known, sequence databases include sites with the military, with private industry and in Japan (the DNA Data Bank of Japan: DDBJ). In most cases data is openly exchanged between the databases so that many sites "mirror" one another.

One special sequence database deserves separate mention. **NRL_3D** is a database of the peptide sequences from entries in the three-dimensional structural **Protein Data Bank** (**PDB**) and as such serves as a "bridge" between primary and tertiary information.

What information do they contain and how is it organized?

Most sequence databases are examples of complex ASCII/Binary databases. They often contain several very long text files containing different types of information all related to particular sequences, such as all of the sequences themselves, versus all of the title lines, or all of the reference sections. Binary files often help "hold together" all of these other files by providing indexing functions. Software is usually required to successfully interact with these databases although systems level commands can be used if one understands the data's structure. Nucleic acid databases (and TrEMBL) are split into subdivisions based on taxonomy (historical).

Types of information in a sequence database:

Reference headers include much extremely valuable information—author and journal citations, organism and organ of origin, and the FEATURES table. The features table annotation lists all sorts of important regulatory, transcriptional and translational (CDS coding sequence), catalytic, and structural sites, depending on the database. Actual sequence data follows the annotation. Becoming familiar with the general format of sequence files for the type of software you use can save a lot of grief later.

How is this information accessed? As mentioned above, software specific routines are the most convenient method in which to access these databases. However, most methods require that you know the proper sequence name and/or database identifier and those are usually discovered with some sort of text searching program, again either on the World Wide Web or not. Which brings up two points, locus names versus accession numbers:

The LOCUS, ID, and ENTRY names category in the databases are different than the Accession number. Each sequence is given a unique accession number upon submission. This number allows tracking of the data when entries are merged or split; it will always be associated with its particular data. Entry names may change; accession numbers are forever; they just pile up.

What changes have occurred that affect software used to gather desired information and the end users — history and development?

The first well recognized sequence database was Dayhoff's Atlas of Protein Sequence and Structure begun in the late sixties. GenBank began in 1982, EMBL in 1980. They have all been attempts at establishing an organized, reliable, comprehensive and openly available library of genetic sequence. Databases have long-since outgrown a hardbound atlas. They have become huge and have evolved through many changes. Changes in format over the years are a major source of grief for software designers and program users. Each program needs to be able to recognize particular aspects of the sequence files; whenever they change, it's liable to throw a wrench in the works. NCBI's ASN.1 format and its **Entrez** interface, available on the Web or as a dedicated client-server program, attempt to circumvent these frustrations somewhat. EMBL's **SRS** (Sequence Retrieval System) found on the World Wide Web at all EMBL OutStations and the Wisconsin Package's **LookUp** derivative of SRS also help people perform text searches in, interact with, and browse in the sequence databases. Both SRS and Entrez provide 'links' to associated databases so that you can jump from, for instance, a chromosomal map location, to a DNA sequence, to its translated protein sequence, to a corresponding structure, and then to a MedLine reference, and so on. They are very helpful!

Specialized versions of sequence databases include sequence pattern databases such as restriction and protease sites, promoter regions, and protein motifs and profiles; and organism or system specific databases such as the sequence portion of ACeDb (A *C. elegans* Database), FlyBase (*Drosophila* dataBase), and SGD (*Saccharomyces* Genome Database) and the Ribosomal Database Project.

What **other types of databases** is bioinformatics concerned with?

In general two other general types of databases are accessed in bioinformatics — reference and three-dimensional structure.

Reference databases run the gamut from **OMIM** (Online Mendelian Inheritance In Man) to PubMed access to **MedLine** (NCBI's access to NLM' bibliographic database of over 4,000 biomedical journals). Other databases that could be put in this class include things like proprietary medical records databases and population studies databases.

Finally, the Research Collaboratory for Structural Bioinformatics (a consortium consisting of three institutions: Rutgers University, San Diego Supercomputer Center at University of California, San Diego, and the National Institute of Standards and Technology) support the three-dimensional structural Protein Data

Bank (**PDB**). Other three-dimensional structure databases include the Nucleic Acid Databank at Rutgers (NDB) and the Cambridge small molecule crystallographic Structural Database (CSD).

II. What about Homology?

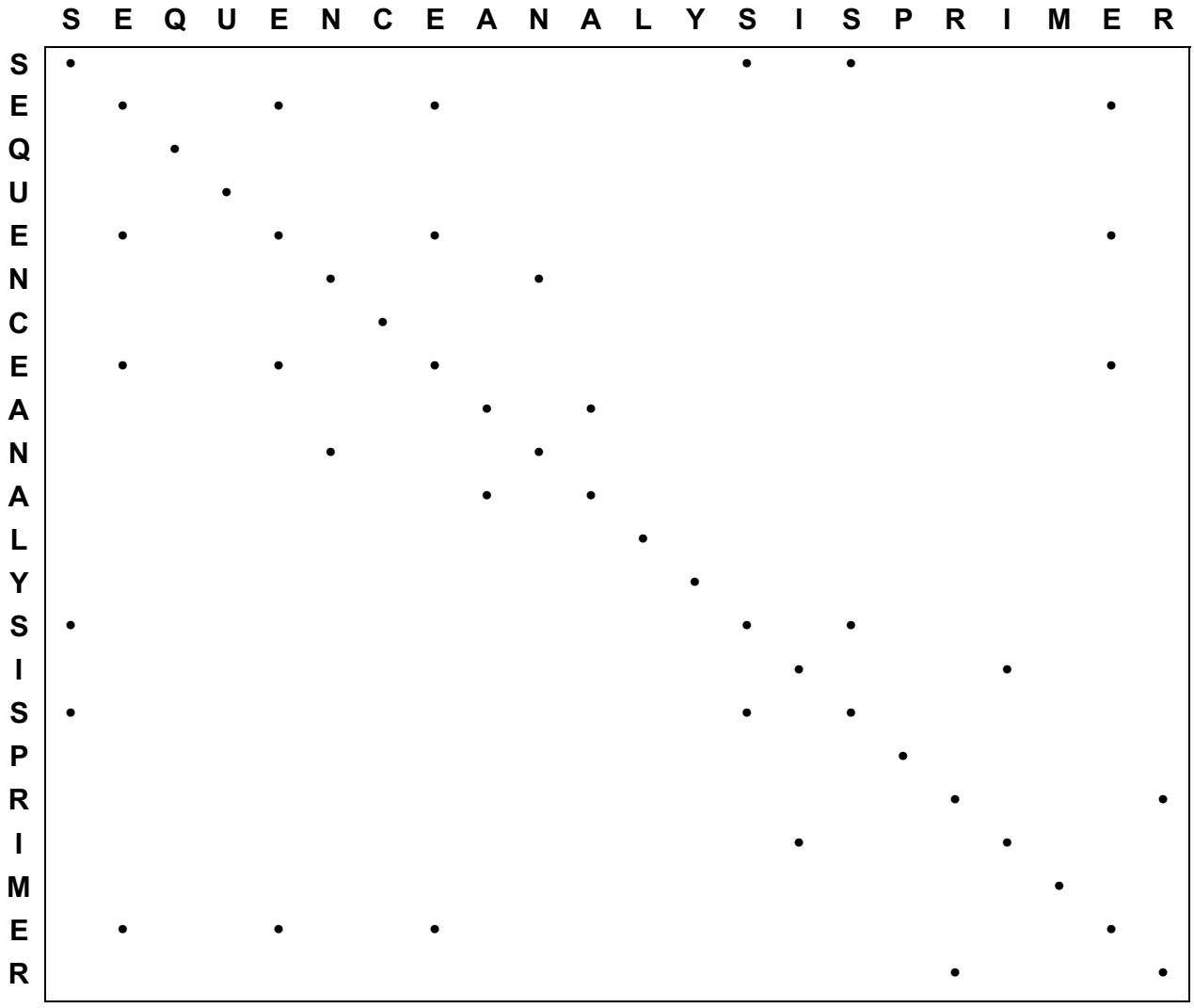
Inference through homology is a fundamental principle of biology!

A. Pairwise Comparisons: the Dot Matrix Method.

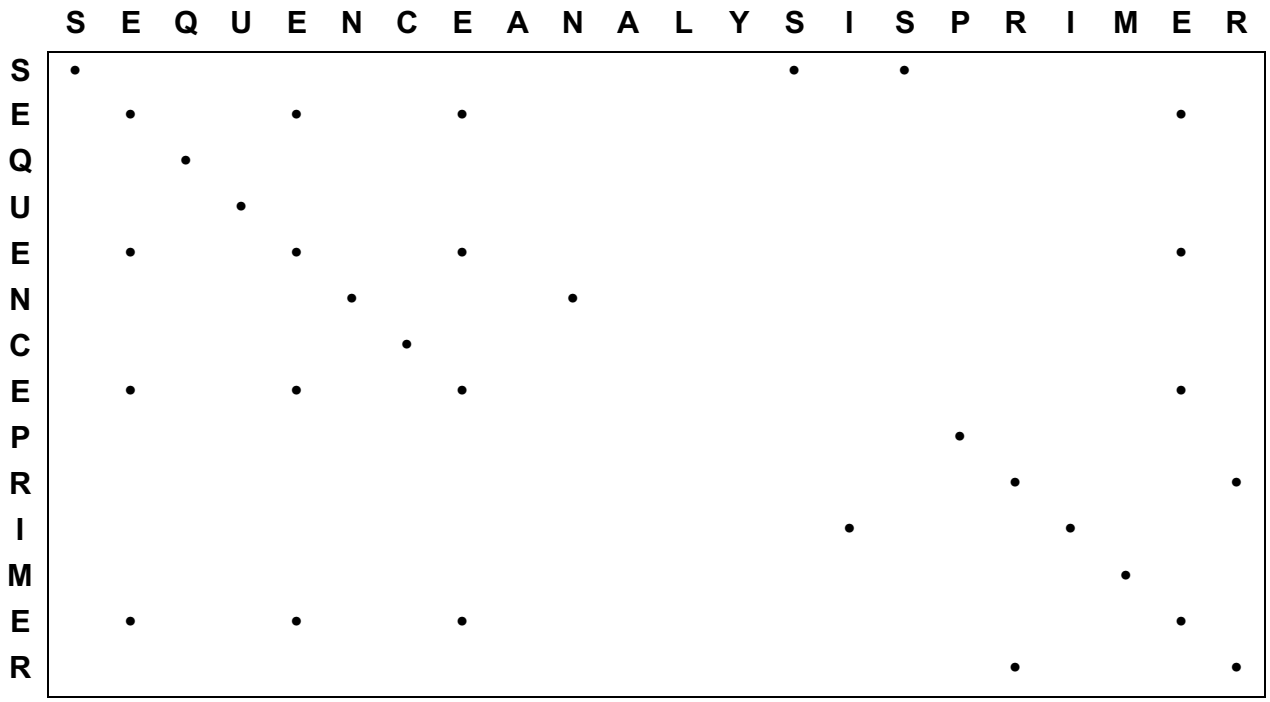
Why use dot matrix analysis? Because your own mind and eyes are still better than computers at discerning complex visual patterns, especially when more than one pattern is being considered.

What approaches are used?

To illustrate, I will use a very simple 0, 1 (match, nomatch) identity scoring function. As you will see later, more complex scoring functions will normally be used in sequence analysis (especially with amino acid sequences). This example is based on the illustration in*Sequence Analysis Primer* (Gribskov and Devereux, editors, 1991). The sequences to be compared are written out along the x and y axes of a matrix and then a dot is placed wherever the two squences' symbols match:

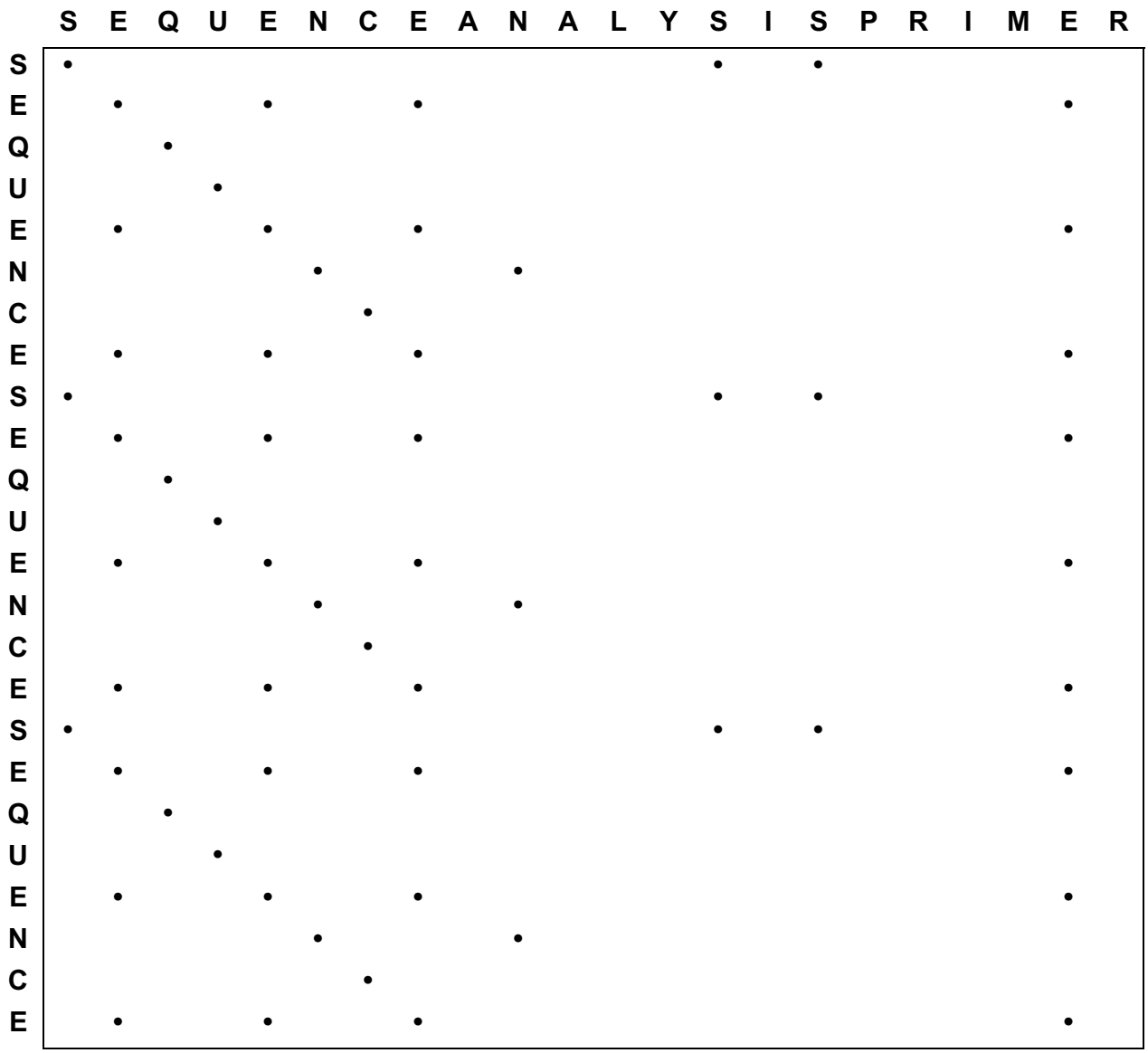


Since this is a comparison between two of the same sequences, an *intra*sequence comparison, the most obvious feature is the main identity diagonal. Two short perfect palindromes can be seen as crosses directly off the main diagonal; they are “ANA” and “SIS.” If this were a double-stranded DNA or RNA sequence self comparison, these inverted repeat regions would be indicative of potential cruciform pseudoknots at that point. Direct internal repeats will show up as parallel diagonals off of the main diagonal. The biggest asset of dot matrix analysis is it allows you to visualize the entire comparison at once, not concentrating on any one ‘optimal’ region, but rather giving you the ‘Gestalt’ of the whole thing. You can see the ‘less than best’ comparisons as well as the main one and then ‘zoom-in’ on those regions of interest using more detailed procedures. Check out the ‘mutated’ *inter*sequence comparison below:



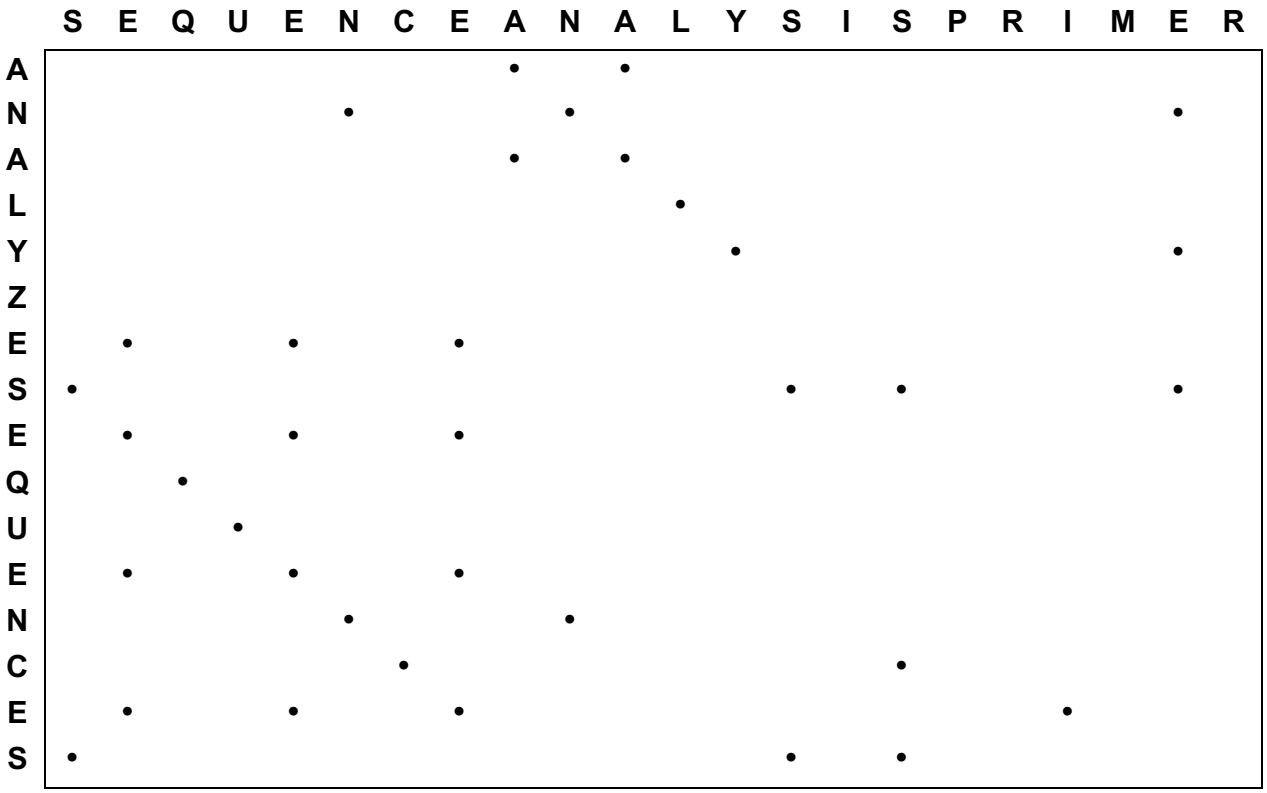
Here you can easily see the effect of a sequence ‘insertion’ or ‘deletion.’ It is impossible to tell whether the evolutionary event that caused the discrepancy between the two sequences was an insertion or a deletion and hence this phenomena is called an ‘indel.’ A jump or shift in the register of the main diagonal on a dotplot clearly points out the existence of a indel.

Other phenomena that are easy to visualize with dot matrix analysis are duplications and direct repeats. These are shown in the following example:



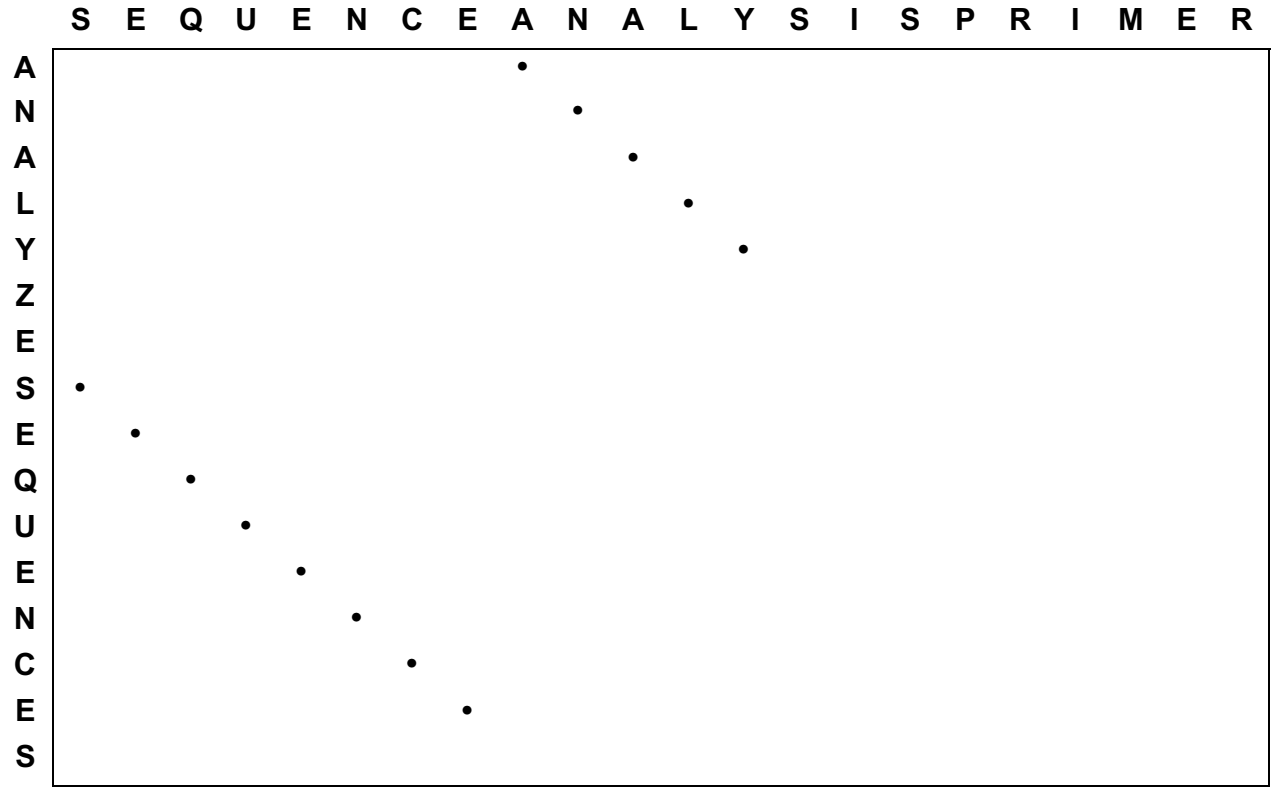
The ‘duplication’ here is seen as a distinct column of diagonals; whenever you see either a row or column of diagonals in a dotplot, you are looking at direct repeats.

Now consider the more complicated mutation in the following comparison:



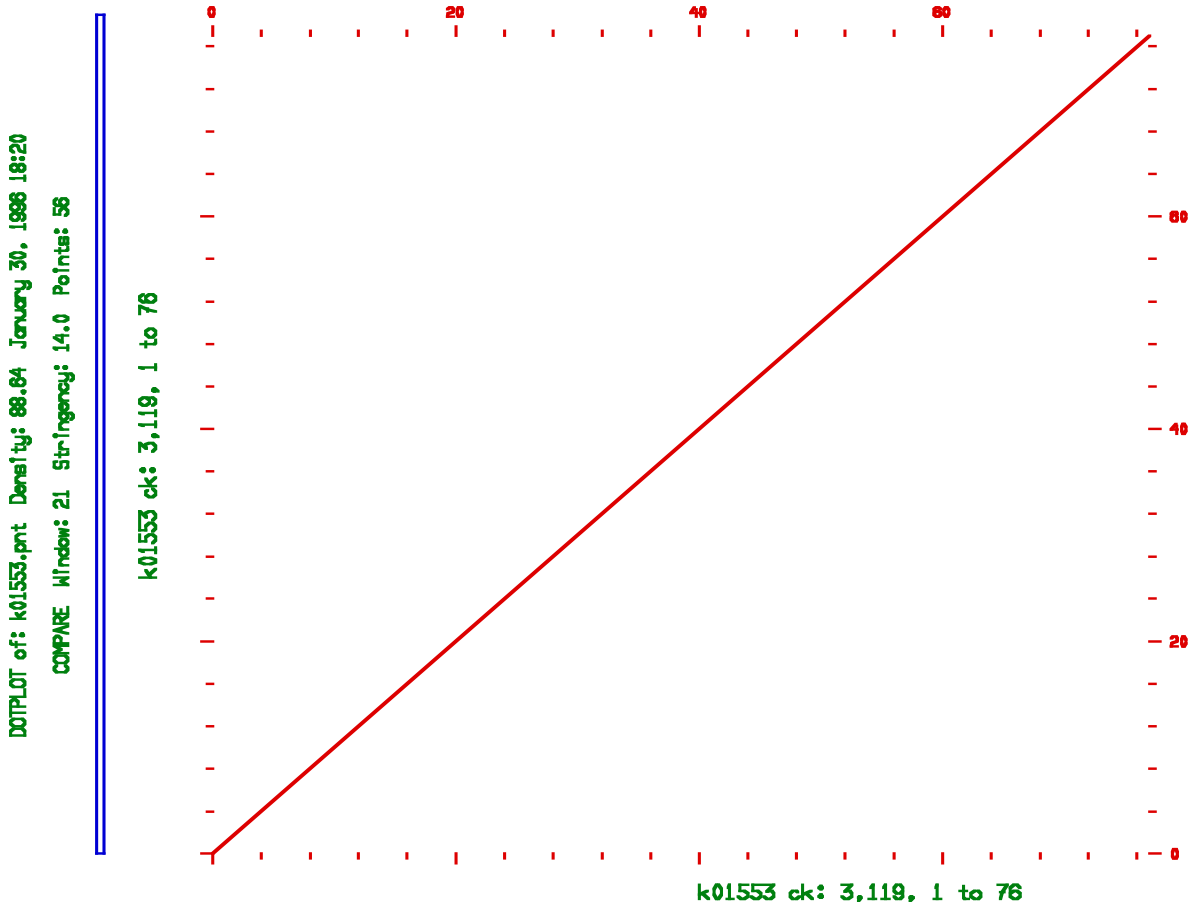
Again, notice the diagonals. However, they have now been displaced off of the center diagonal of the plot and, in fact, in this example, show the occurrence of a 'transposition.' Dot matrix analysis is the only sensible way to locate such transpositions in sequences. Inverted repeats still show up as perpendicular lines to the diagonals, they are just now not on the center of the plot. The 'deletion' of 'PRIMER' is shown by the lack of a corresponding diagonal.

Reconsider the same plot. Notice the extraneous dots that neither indicate runs of identity between the two sequences nor inverted repeats. These merely contribute '**noise**' to the plot and are due to the '**random**' occurrence of the letters in the sequences, the **composition** of the sequences themselves. How can we 'clean up' the plots so that this noise does not detract from our interpretations? Sequence analysis is all about balancing signal to noise, and sometimes it can be quite the balancing act! Consider the implementation of a **filtered windowing approach**; a dot will only be placed if some 'stringency' is met. What is meant by this is that if within some defined window size, and when some defined criteria is met, then and only then, will a dot be placed at the middle of that window. Then the window is shifted one position and the entire process is repeated. This very successfully rids the plot of unwanted noise. In the plot below a window of size **three** and a stringency of **two out of three** was used to considerably improve the signal to noise ratio:

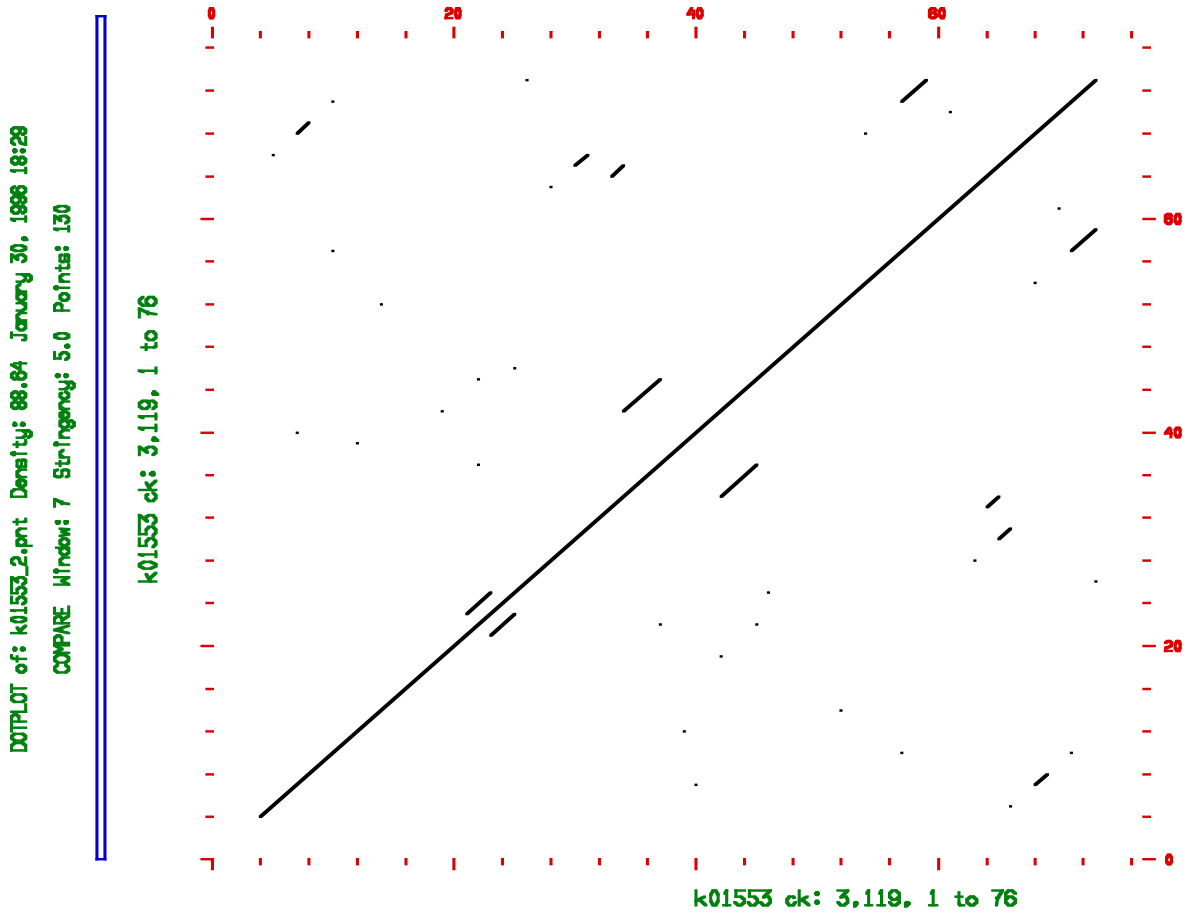


The only remaining dots indicate the two runs of identity between the two sequences, however, any indication of the palindrome, "ANA" has been lost. This is because our filtering approach was too stringent to catch such a short element. In general you need to make your window about the same size as the element you are attempting to locate. In the case of our palindrome, "AN" and "NA"' are the inverted repeat sequences and since our window was set to three, we will not be able to see an element only two letters long. Had we set our filter to one out of two, then these still would be visible.

You need to be careful with window/stringency dot matrix methods. Default window sizes and stringencies may not be appropriate for the analysis at hand. Consider the following set of examples from the phenylalanine transfer RNA molecule from yeast. The sequence and structure are both known for this molecule and this illustration will show how simple dot-matrix procedures can quickly lead to functional and structural insights (**even without complex folding algorithms**). If run with all default settings the dotplot from a comparison of this sequence with itself is quite noninformative, only showing the main identity diagonal:

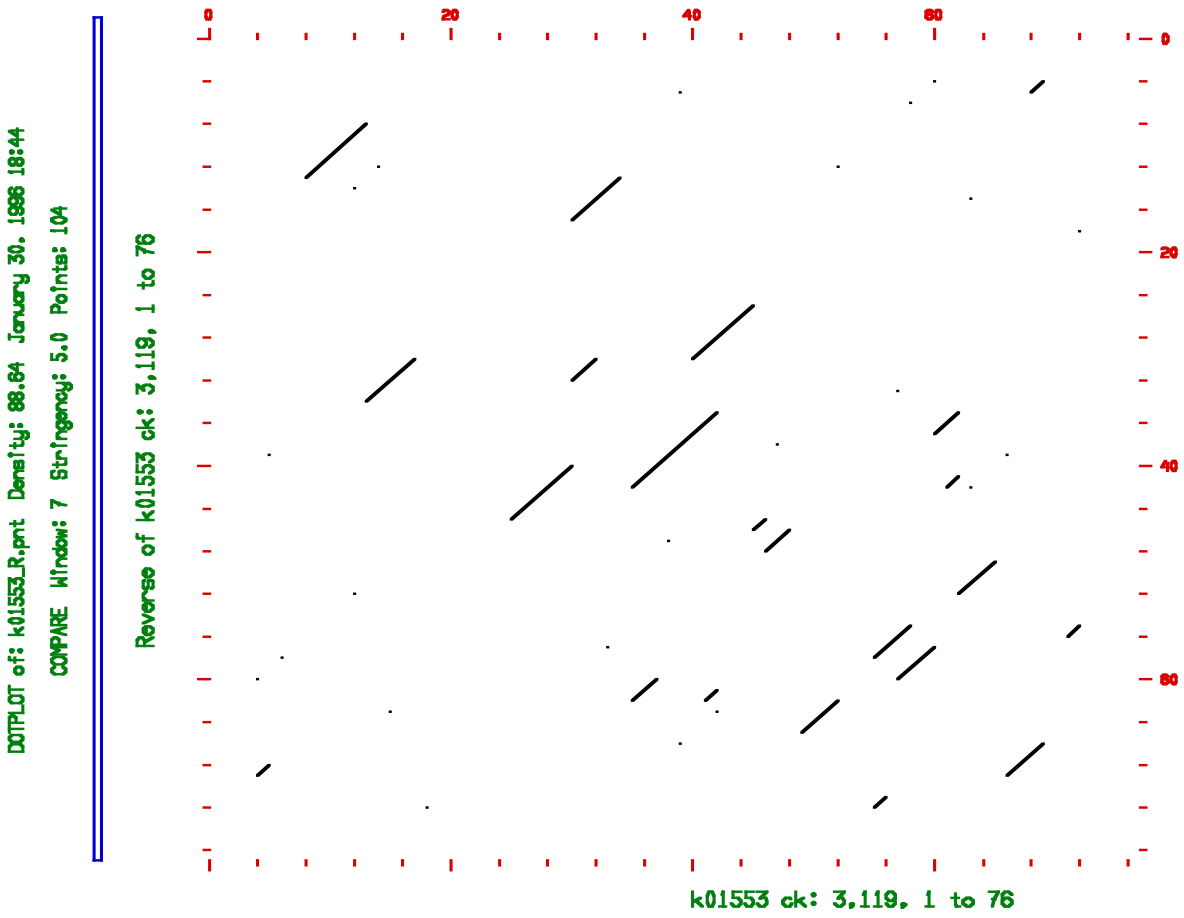


However, if you adjust the window size down to find finer features some elements of symmetry become apparent. Here I have changed the window size to 7 and the stringency value to 5. As a general guide pick a window size about the same size as the feature that you are trying to recognize and a stringency such that unwanted background noise is just filtered away enough to enable you to see that desired feature.



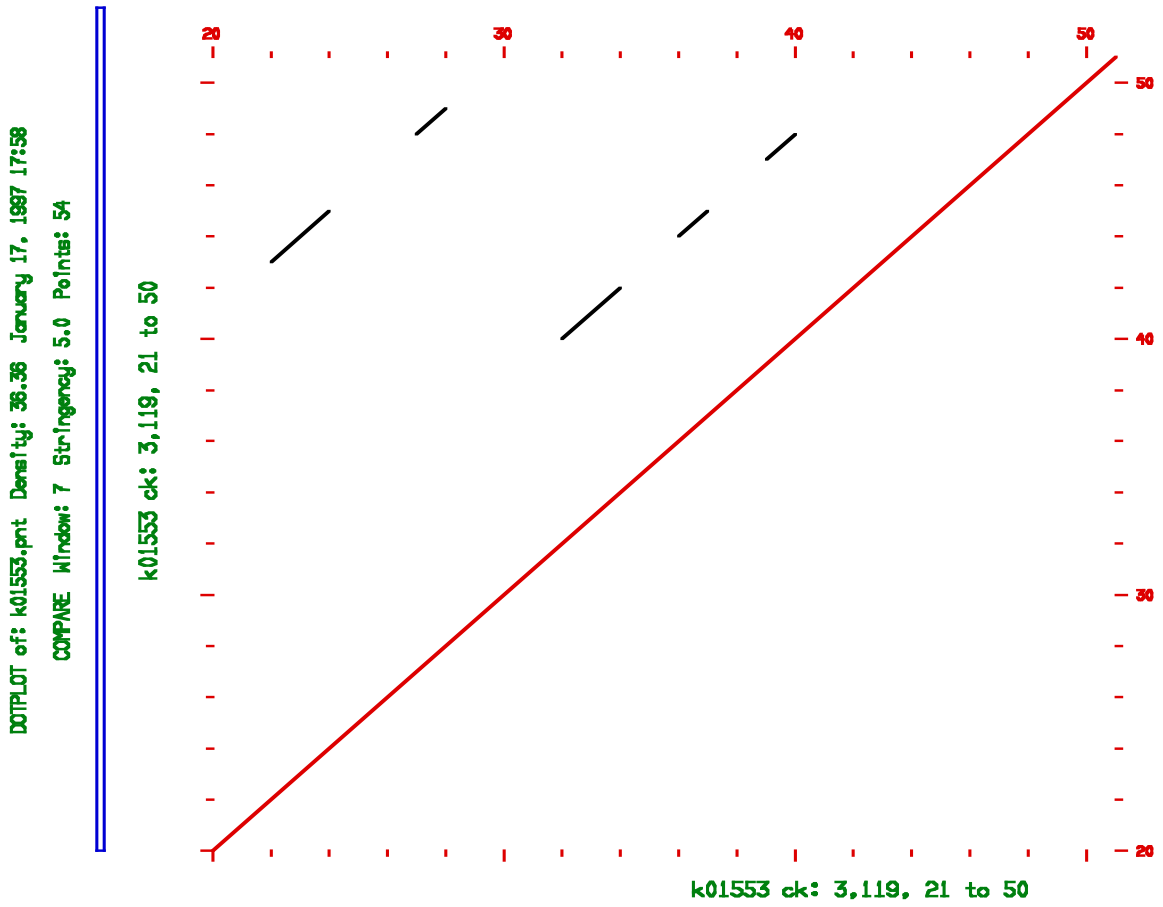
Several direct repeats are now obvious that remained obscured in the previous analysis.

When dealing with RNA/DNA, even more insight can be gained by comparing the reverse, complement of a sequence to itself. Compare the following dotplot to the previous ones; here the yeast tRNA sequence is compared to its reverse, complement using the same 5 out of 7 stringency setting:

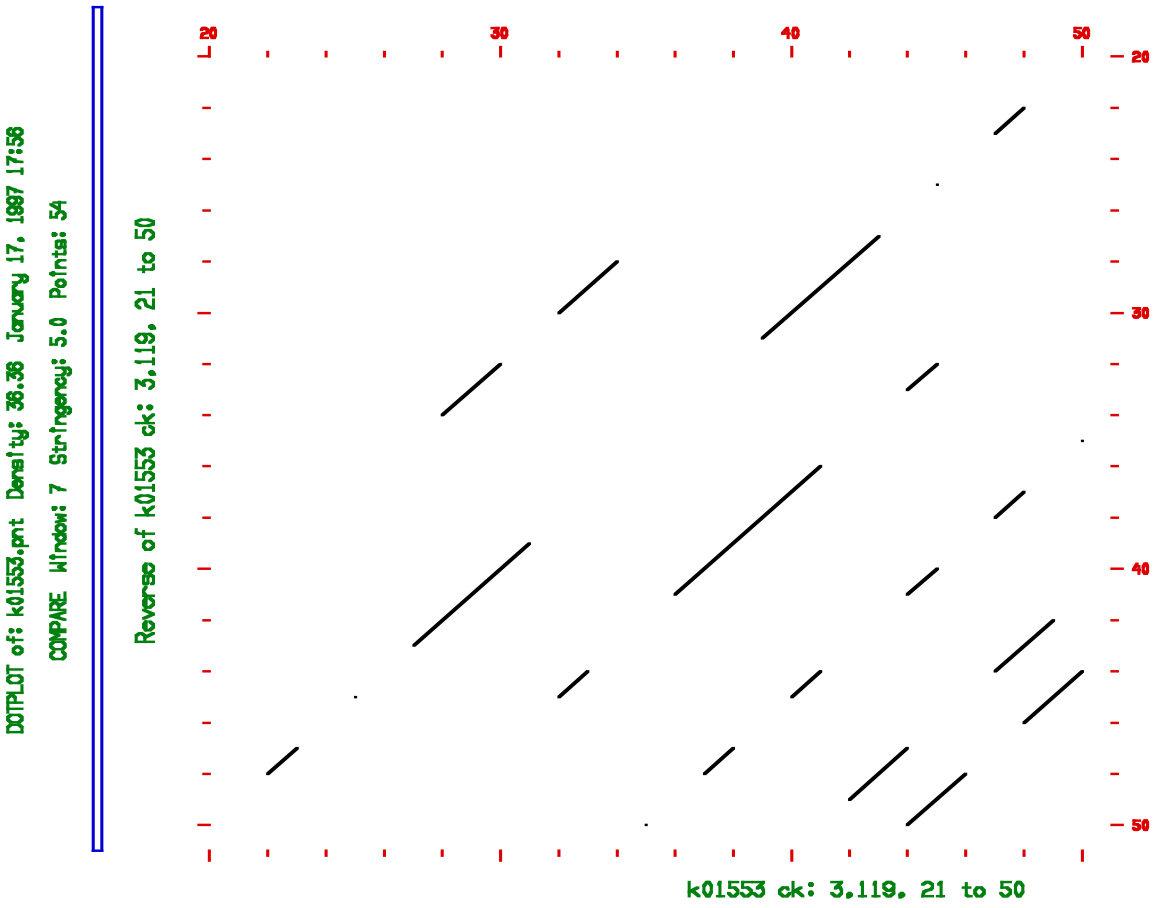


Now the potential for inverted repeats become obvious; these are the well characterized stem-loop structures of the tRNA clover-leaf molecular shape. They appear as clearly delineated diagonals. These diagonals are now perpendicular to an imaginary main diagonal running oppositely than in the previous cases since we reversed the orientation of the second sequence. Take for instance the middle stem; the region of the molecule centered at approximately base number 38 has a clear propensity to base pair with itself without creating a loop since it crosses the main diagonal and then just after a small unpaired gap another stem is formed between the region from about base number 24 through 30 with approximately 46 through 40.

That same region ‘zoomed in on’ has some small direct repeats seen by comparing the sequence against itself without reversal:



But looking at the same region of the sequence against its reverse-complement shows a wealth of potential stem-loop structure in the transfer RNA:

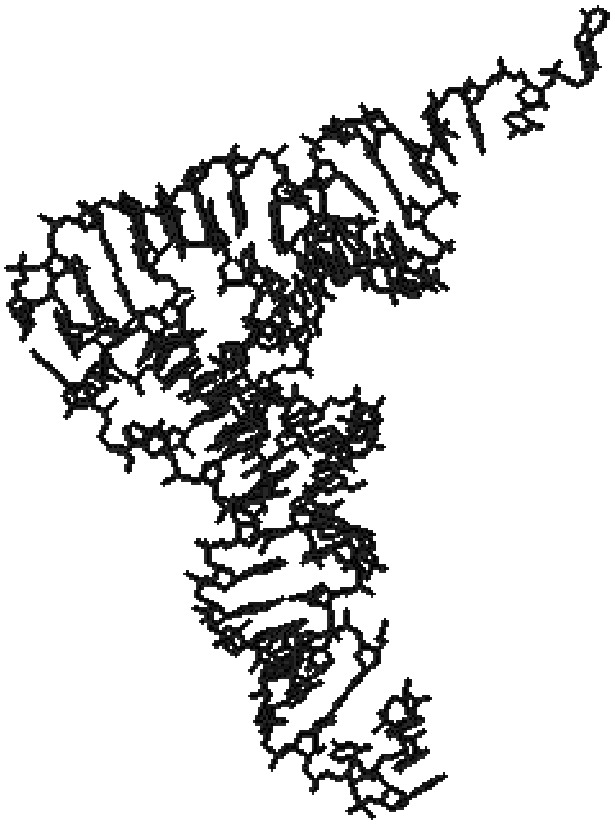


Another program can predict some of these matchups base by base. Depending on what parameters you use, this is one of them:

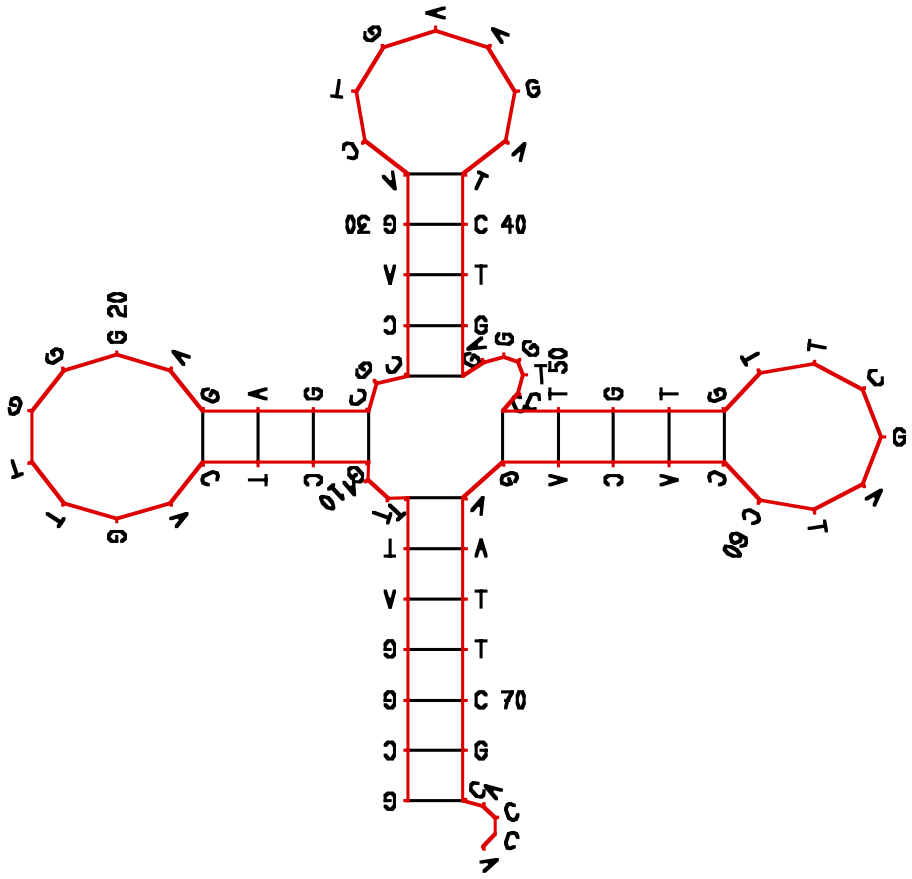
```
22 GAGCGCCAGACT  G    12, 22
   || | |||||  |   A
48 CTGGAGGTCTAG  A    3
```

So, as noted above, the region of K01553 from base position 22 through position 33 base pairs with (think — is quite similar to the reverse-complement of) itself from base position 37 through position 48. Got it?

This stem probably corresponds to the bottom-most stem representation in the standard model of tRNA. Here is a view in an orientation that allows you to visualize most of the stem structure of the yeast phenylalanine tRNA model as solved by Sundaralingam et al. (1976) deposited in PDB under access code 1TRA:



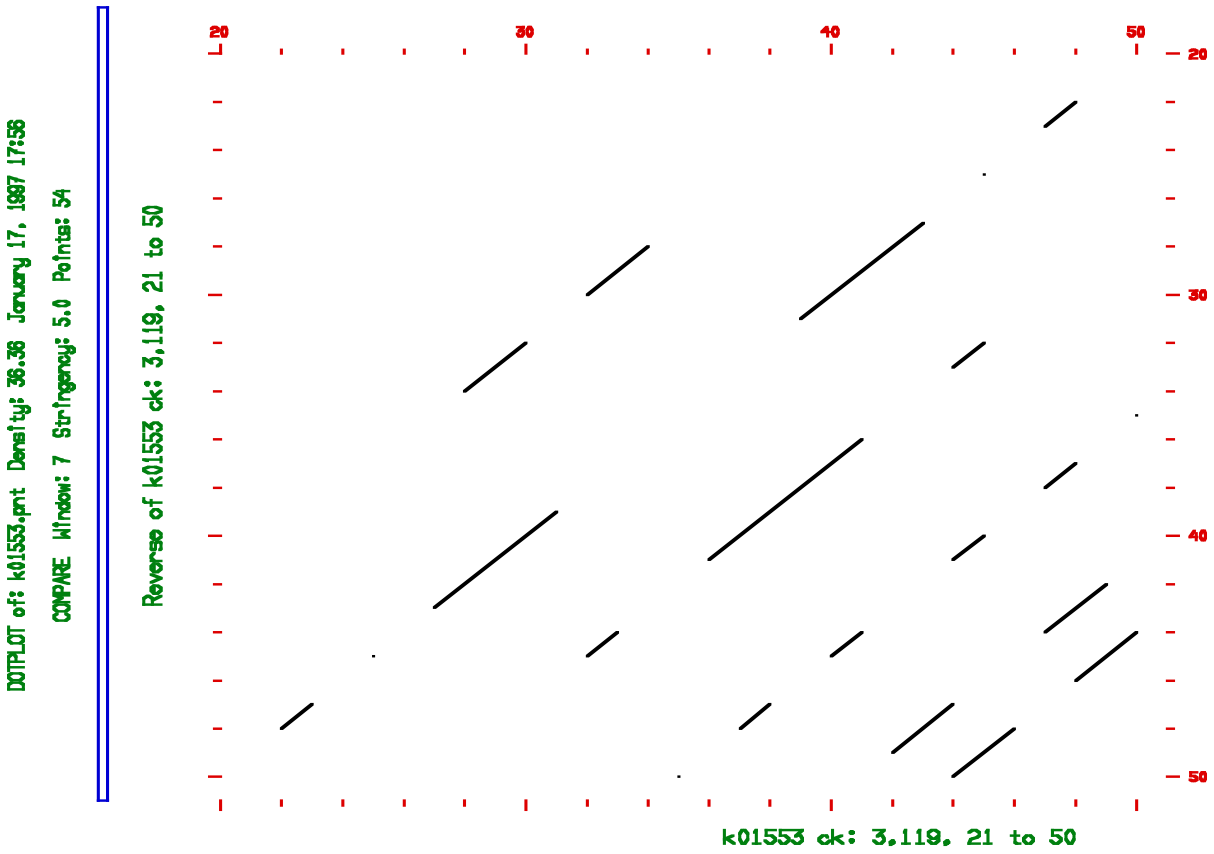
A ‘Squiggle’ plot of this same molecule from the output of MFold, Zuker’s (1989) RNA folding algorithm which uses base pairing energy minimization to find the family of most optimal and suboptimal structures, the most stable structure found is shown to possess a stem at positions 27 to 31 with 39 to 43. However the region around position 38 is represented as a loop. Note that the molecule is upside down here as compared to the previous model. Reality, as modeled above, is seen to lie somewhere in between these two interpretations, but the simple dotplot analysis did quickly provide some valuable insights.



B. Pairwise Comparisons: Dynamic Programming.

Given two Sequences — what can we know? Zooming-in on 'exact' alignments with dynamic programming — a method for finding one mathematically 'optimal' alignment.

As we've seen — we can use dot matrix procedures to gain the 'Gestalt' of a comparison between two sequences in order to learn about all the areas between them that are alike and maybe even gain some structural or functional insights into them. For instance, the above Phe tRNA example:



But what about these alike areas? What's the best 'path' through the dot matrix? How long do I extend it? How can I 'zoom-in' on it to see exactly what's happening? Where, specifically, is this alignment; how can I see the 'best' ones? And, what can I learn from these alignments?

This brings up the alignment problem. It is easy to see that two sequences are aligned when they have identical symbols at identical positions, but what happens when symbols are not identical or the sequences are not the same length? How can we know that the most alike portions of our sequences are aligned, when is an alignment optimal, and does optimal mean biologically correct?

But, how to do all of this?

A '**brute force**' approach just won't work. Even without considering the introduction of gaps, the computation required to compare all possible alignments between two sequences requires time proportional to the product of the lengths of the two sequences. Therefore, if the two sequences are approximately the same length (N), this is a N^2 problem. To include gaps, we would have to repeat the calculation 2N times to examine the possibility of gaps at each possible position within the sequences, now a N^{4N} problem.

Waterman illustrated the problem in 1989 stating that to align two sequences 300 symbols long, 10^{88} comparisons would be required, about the same number of elementary particles estimated to exist in the universe!

Part of a better solution . . . enter:

the dynamic programming algorithm.

To really understand the dynamic programming algorithm, you need to do it manually a few times, even though computers can do it much, much more easily. These examples will use DNA so that a unitary match matrix with 1 point for matching and 0 points for mismatching is still appropriate. We'll talk about proteins and 'similarity' and the concept of substitution matrices next.

In its simplest implementation we will not consider gapping at all. The solution occurs in two stages. The first begins very much like the dot matrix methods described previously; the second is totally different. I will start with the same example as that in text with one simplification. Instead of calculating the 'score matrix' on the fly as we proceed through the graph, I like to completely fill in an original 'match matrix,' then add points to those positions which produce favorable alignments.

a) A completed match matrix using one point for matching and zero points for mismatching:

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

b) Now begin to add points based on the best path through the matrix, always working diagonally, left to right and top to bottom (the 'over-your-left-shoulder' rule). The second row is completed here:

		A	**A**	**T**	**G**	**C**
	A	1	1	0	0	0
	G	0	0+1	0+1	1+1	0+1
	G	0	0	0	1	0
	C	0	0	0	0	1

c) Continue adding points based on the best previous path through the matrix; the third row is completed here:

		A	**A**	**T**	**G**	**C**
	A	1	1	0	0	0
	G	0	0+1= 1	0+1= 1	1+1= 2	0+1= 1
	G	0	0+1	0+1	1+1	0+2
	C	0	0	0	0	1

d) The score matrix is now complete:

		A	A	T	G	C
		1	1	0	0	0
G	0	0+1=1	0+1=1	1+1=2	0+1=1
G	0	0+1=1	0+1=1	1+1=1	0+2=2
C	0	0+1	0+1	0+1	1+2

e) Now pick the bottom, right-most, highest scores in the matrix and work your way back through it always choosing the most favorable path. This is called the traceback stage and the matrix is now referred to as the path graph. In this case that highest score is in the right-hand corner, but it need not be:

		A	A	T	G	C
		1	1	0	0	0
G	0	0+1=1	0+1=1	1+1=2	0+1=1
G	0	0+1=1	0+1=1	1+1=2	0+2=2
C	0	0+1=1	0+1=1	0+1=1	1+2=3

f) The completed traceback is shown with outline characters; these are all optimal alignments:

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

The following alignments are all generated from the above path graph (f). All five have three matches. Gap penalties would have eliminated the last two of them; however, that still leaves three:

```
AG.GC   A.GGC      .AGGC      A..GGC     .A.GGC
|  ||   |  ||       | ||      |  | |      | | |
AATGC   AATGC      AATGC      AATG.C     AATG.C
```

Be careful. Software will arbitrarily choose only one of these to report as the optimal!

The next example will be slightly more difficult. Unlike the previous example without gap penalties, I will now impose a very simple gap penalty function. I am going to penalize the scoring scheme by subtracting 1 point for every gap inserted unless they are at the beginning or end of the sequence. In other words, end gaps will not be penalized; both sequences do not have to begin or end at the same point in the alignment. This end-weighting scheme is the default for most alignment programs but can often be changed with a program option. However, this function is a much simpler gap penalty function than normally used in the alignment programs. Normally an 'affine,' i.e. a linear, function is used:

total penalty = gap opening penalty + ([length of gap][gap extension penalty]).

[To run most alignment programs with the type of simple gap penalty used in the following example, you would have to designate a 'gap creation' penalty of zero and a 'gap extension' penalty of one for DNA sequences.]

The following example uses two randomly generated sequences that happen to fit the tata region consensus in eukaryotes and prokaryotes. The sample eukaryote promoter sequence is along the X-axis, the prokaryote along the Y-axis.

a) First complete a match matrix using one point for matching and zero points for mismatching between bases, just like before:

	c	**T**	**A**	**T**	**A**	**t**	**A**	**a**	**g**	**g**
c	1	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	1	1
T	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
t	0	1	0	1	0	1	0	0	0	0
A	0	0	1	0	1	0	1	1	0	0
a	0	0	1	0	1	0	1	1	0	0
T	0	1	0	1	0	1	0	0	0	0

b) Now add and subtract points based on the best path through the matrix, working diagonally, left to right and top to bottom. When you have to jump a box to make the path, subtract one point per box jumped, except at the beginning or end of the alignment. Fill in all additions and subtractions and calculate the sums and differences as you go:

	c	**T**	**A**	**T**	**A**	**t**	**A**	**a**	**g**	**g**
c	**1**	0	0	0	0	0	0	0	0	0
g	**0**	0**+1=1**	0**+1-1=0**	0**+0-0=0**	0**+0-0=0**	0**+0-0=0**	0**+0-0=0**	0**+0-0=0**	1**+0-0=1**	1**+0=1**
T	**0**	1**+1-1=1**	0**+1=1**	1**+1-1=1**	0**+0-0=0**	1**+0-0=1**	0**+0-0=0**	0**+0-0=0**	0**+0-0=0**	0**+1-0=1**
A	**0**	0**+0-0=0**	1**+1=2**	0**+1=1**	1**+1=2**	0**+1-1=0**	1**+1=2**	1**+1-1=1**	0**+0-0=0**	0**+0-0=0**
t	**0**	1**+0-0=1**	0**+1-1=0**	1**+2=3**	0**+1=1**	1**+2=3**	0**+2-1=1**	0**+2=2**	0**+1=1**	0**+0-0=0**
A	**0**	0**+0-0=0**	1**+1=2**	0**+2-1=1**	1**+3=4**	0**+3-1=2**	1**+3=4**	1**+3-1=3**	0**+2=2**	0**+1=1**
a	**0**	0**+0-0=0**	1**+0-0=1**	0**+2=2**	1**+3-1=3**	0**+4=4**	1**+4-1=4**	1**+4=5**	0**+3=3**	0**+2=2**
T	**0**	1**+0-0=1**	0**+0-0=0**	1**+1=2**	0**+2=2**	1**+3=4**	0**+4=4**	0**+4=4**	0**+5=5**	0**+5-1=4**

c) I'll now clean up the score matrix, only showing the totals in each cell:

	c	**T**	**A**	**T**	**A**	**t**	**A**	**a**	**g**	**g**
c	1	0	0	0	0	0	0	0	0	0
g	0	1	0	0	0	0	0	0	1	1
T	0	1	1	1	0	1	0	0	0	1
A	0	0	2	1	2	0	2	1	0	0
t	0	1	0	3	1	3	1	2	1	0
A	0	0	2	1	4	2	4	3	2	1
a	0	0	1	2	3	4	4	5	3	2
T	0	1	0	2	2	4	4	4	5	4

d) Now convert the score matrix into a traceback path graph by picking the bottom-most, furthest right and highest scoring coordinates, always choosing the highest scoring traceback route, to connect them all the way back to the beginning:

	c	**T**	**A**	**T**	**A**	**t**	**A**	**a**	**g**	**g**
c	1	0	0	0	0	0	0	0	0	0
g	0	1	0	0	0	0	0	0	1	1
T	0	1	1	1	0	1	0	0	0	1
A	0	0	2	1	2	0	2	1	0	0
t	0	1	0	3	1	3	1	2	1	0
A	0	0	2	1	4	2	4	3	2	1
a	0	0	1	2	3	4	4	5	3	2
T	0	1	0	2	2	4	4	4	5	4

There will probably be more than one best path through the matrix. This time, starting at the top and working down as we did, then tracing back, I found two optimum alignments:

```
cTATAtAagg        cTATAtAagg
|  |||||          |   ||||
cg.TAtAaT.        cgT.AtAaT.
```

Each of these solutions yields a traceback total score of 22; this is the number optimized by the algorithm, not any type of a similarity or identity score! Even though one of these alignments has 6 exact matches and the other has 5, they are both optimal according to the relatively strange criteria by which we solved the algorithm. Software will report only one of these solutions. Do you have any ideas about how others could be discovered? Answer — Often if you reverse the solution of the entire dynamic programming process, other solutions can be found!

These examples bring up an important point. Just because dynamic programming is guaranteed to find an optimal alignment, it is not necessarily the only optimal alignment. Furthermore, the **optimal alignment found is not necessarily the 'right' or biologically relevant alignment**. As always, question the results of any computerized solution based on what you know about the biology of the system.

Now that we have worked through some examples of dynamic programming, let's attempt to formalize the steps. An optimal alignment is defined as an arrangement of two sequences, 1 of length *i* and 2 of length *j*, such that:

1) **you maximize the number of matching symbols between 1 and 2;**
2) **you minimize the number of indels within 1 and 2; and**
3) **you minimize the number of mismatched symbols between 1 and 2.**

Therefore, the actual solution can be represented by:

$$S_{ij} = s_{ij} + \max \begin{cases} S_{i-1\ j-1} & \text{or} \\ \max S_{i-x\ j-1} + w_{x-1} & \text{or} \\ 2 < x < i & \\ \max S_{i-1\ j-y} + w_{y-1} & \\ 2 < y < i & \end{cases}$$

where S_{ij} is the score for the alignment ending at *i* in sequence 1 and *j* in sequence 2,
s_{ij} is the score for aligning *i* with *j*,
w_x is the score for making a *x* long gap in sequence 1,
w_y is the score for making a *y* long gap in sequence 2,
allowing gaps to be any length in either sequence.

. . . yeah, sure !!!!

The above examples illustrate the Needleman and Wunsch (1970) **global** solution. These global solutions can be restricted to a range within sequences; however, they are still global processes. Later refinements (Smith and Waterman, 1981) demonstrated how dynamic programming can also be used to find optimal **local** alignments. To solve dynamic programming using local alignment (without going into all the gory details) programs use the following tricks:

1) An identity match matrix that uses **negative numbers** for mismatches is incorporated. Therefore, bad paths quickly become **very bad**. This leads to a traceback path matrix with many alternative paths, most of which do not extend the full length of the graph.

2) The best traceback within the graph is chosen. This does not have to begin or end at the edges of the graph — it is looking for the best segment of alignment!

What about proteins — conservative replacements and similarity as opposed to identity, and similarity versus homology!

Similarity is not automatically homology. **Homology means related by descent from a common ancestor** — there's no such thing as percent homology, either something is homologous or it is not. The famous evolutionist Walter Fitch likes to relate the joke — "homology is like pregnancy, you either are or you're not! There's no in between."

And what the heck is a PAM or BLOSUM table?

Based on very carefully prepared multiple sequence alignments and fancy mathematics the relative probabilities of any one amino acid residue changing into another can be empirically calculated as appropriate within some level of divergence between the sequences considered.

Hence, protein amino acid symbol substitution tables, also known as scoring matrices.

The most famous of all time . . . the original **Dayhoff PAM 250 Matrix**:

```
!!AA_SCORING_MATRIX_RECT 1.0

PAM250 amino acid substitution matrix.

Dayhoff table (Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C. [1979] in Atlas of Protein
Sequence and Structure,
Dayhoff, M. O. Ed, pp. 345-352 (Figure 84), National Biomedical Research Foundation, Washington (D.C.)

{
GAP_CREATE 12
GAP_EXTEND 4
}
```

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

The new kid on the block . . . the **BLOSUM62 Matrix**: Now the default in most alignment programs.

```
!!AA_SCORING_MATRIX_RECT 1.0

BLOSUM62 amino acid substitution matrix.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks.
Proc. Natl. Acad. Sci. USA 89: 10915-10919.

{
GAP_CREATE 12
GAP_EXTEND 4
}

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

Values whose magnitude is $\geq \pm 4$ are drawn in outline characters to make them easier to recognize. Notice that positive values for identity range from 4 to 11 and negative values for those substitutions that rarely occur go as low as –4. The most conserved residue is tryptophan with a score of 11; cysteine is next with a score of 9; both proline and tyrosine get scores of 7 for identity.

Combine the concept of amino acid substitution matrices with dynamic programming to align proteins! However, this is much more robust because proteins have twenty symbols versus DNA's four — a much better signal to noise ratio, so all alignment programs work better.

C. Significance: When is an Alignment Worth Anything Biologically?

Percent similarity and identity, Qualities, Init's, Opt's, Z scores, Expectation Functions, and Poisson Probabilities. What do they all mean? Many of the programs generate percent similarity scores, however these really don't mean a whole lot. Do not use percent similarities or identities to compare sequences except in the roughest manner. They are not optimized or normalized in any manner by the programs. The 'quality' score means a lot more since it is the metric maximized by the program, but it is hard to interpret. They are only relevant within the context of a particular comparison or search. At least they take the length of similarity, all of the necessary gaps introduced, and the matching of symbols all into account. The Quality ratio is the metric optimized by dynamic programming divided by the length of the shorter sequence. As such it represents a even fairer comparison metric, but it is relative to the particular symbol substitution matrix used in the procedure. To get a better handle on what these various scores mean, read up on the algorithms — statistics can be confusing, but it helps.

So, how do we tell if an alignment discovered by a program means anything? Is it statistically significant, and even more importantly, does it have anything to do with real biology?!

The traditional way of deciding alignment significance relies on an old statistics trick — **Monte Carlo simulations**. This type of significance estimation has many implicit statistical problems; however, few practical alternatives exist for just two sequences. Database searching algorithms, like BLAST and FastA, handle significance in a more meaningful way by comparing an alignment to a distribution of the rest of the database alignments and are discussed below. However, Monte Carlo techniques continue to be used because of their ease and speed; they will remain important in the field.

Monte Carlo techniques compare an actual score, in this case the quality score of an alignment, against the distribution of scores of alignments of a randomized sequence. Most programs do this by jumbling the second sequence of a comparison 100 times after an initial alignment is produced and then generating scores and a standard deviation based on the jumbled matches. Comparing the quality scores of the randomized alignments to the initial alignment can help give a feeling for the relative meaning of the scores. You can compare the mean of the random scores to the unjumbled score using a 'Z score' calculation to help decide significance. An old rule-of-thumb that people often use is, if the actual score is much more than three standard deviations above the mean of the randomized scores, the analysis may be significant; if it is much more than five, than it probably is significant; and if it is above nine, than it definitely is significant.

Many Z scores measure the distance from a mean using this simplistic Monte Carlo model assuming a normal distribution, in spite of the fact that 'sequence-space' actually follows what is know as an 'extreme value distribution;' however, this method does approximate significance estimates quite well and is calculated with the following formula:

$$\text{Z score} = \left[\frac{(\text{actual score}) - (\text{mean of randomized scores})}{(\text{standard deviation of randomized score distribution})} \right]$$

Rough, conservative guidelines to Z scores and Expectation values from typical searches:

Z score	BLAST/Fast E Value	Inference
≤ 3	≥ 0.05	little, if any evidence for homology
$\cong 6$	$\cong 10^{-9}$	probably homologous, but may be due to convergent evolution
≥ 9	$\leq 10^{-19}$	definitely homologous

When the two tata sequences from the previous dynamic programming example are compared to one another using the same scoring parameters as before, but incorporating a Monte Carlo Z score calculation, their similarity is found, surprisingly, not to be at all significant. It is merely a reflection of the compositional bias of the two sequences to contain lots of T's and A's. Those results follow:

Average quality based on 100 randomizations:

4.4 +/- 0.8. Plugged into the formula: (5.0 - 4.4) / 0.8 = 0.75, i.e. no significance!

D. Pairwise Comparisons: Database Searching.

Combine all concepts above and the idea of '**hashing**' to come up with **heuristic** style database searching. Hashing is the process of breaking your sequence into small '**words**' or 'ktuples' of a set size and creating a '**look-up**' table with those words keyed to numbers. Then when any of the words match part of an entry in the database, that match is saved. Compile all the results at the end and try to recreate the longest alignment up to the program's restrictions. In general, hashing reduces the complexity of the search problem from N^2 for dynamic programming to N, the length of all the sequences in the database. Approximation techniques are collectively known as 'heuristics.' Webster's defines heuristic as "serving to guide, discover, or reveal; . . . but unproved or incapable of proof." In database searching techniques

the heuristic usually restricts the necessary search space by calculating some sort of a statistic that allows the program to decide whether further scrutiny of a particular match should be pursued. This statistic may miss things depending on the parameters set — that's what makes it heuristic.

The exact implementation varies, but the basic idea follows this in most all of the programs.

Two predominant versions: the BLAST and Fast programs. Both return **local alignments**. Both are not a single program, but rather a **family of programs** with implementations designed to compare a sequence to a database in about every which way imaginable. These include: a DNA sequence against a DNA database (not recommended unless forced to do so because you are dealing with a nontranslated region of the genome), a translated (where the translation is done 'on-the-fly' in all six frames) version of a DNA sequence against a translated ('on-the-fly') version of the DNA database, a translated ('on-the-fly') version of a DNA sequence against a protein database, a protein sequence against a translated ('on-the-fly') version of the DNA database, or a protein sequence against a protein database. Many implementations allow the recognition of frame shifts in translated comparisons.

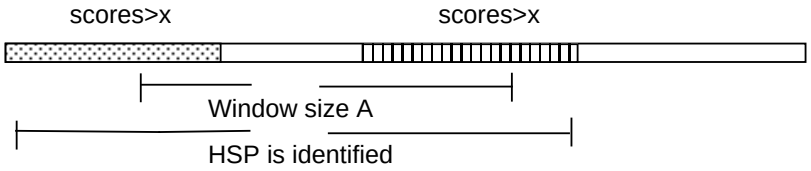
In more detail:

BLAST — Basic Local Alignment Search Tool, developed at NCBI (Altschul et al. 1990 and 1997)**.**

1) Normally **NOT** a good idea to use for DNA against DNA searches (not optimized);
2) **Prefilters** repeat and "low complexity" sequence regions;
4) Can find **more than one region** of gapped similarity;
5) **Very fast** heuristic and parallel implementation;
6) **Restricted** to precompiled, specially formatted databases;

The algorithm:

After BLAST has sorted its lookup table, it tries to find all double word hits along the same diagonal (remember the dot matrix graphs) within some specified distance using what NCBI calls a Discrete Finite Automaton (DFA). These word hits of size *W* do not have to be identical; rather, they have to be better than some threshold value *T*. To identify these double word hits, the DFA scans through all strings of words (typically *W*=3 for peptides) that score at least *T* (usually 11 for peptides). Each double word hit that passes this step then triggers a process called ungapped extension in both directions, such that each diagonal is extended as far as it can, until the running score starts to drop below a pre-defined value *X* within a certain range *A*. The result of this pass is called a High-Scoring segment Pair or HSP:



Those HSPs that pass this step with a score better than *S* then begin a gapped extension step utilizing dynamic programming. Those gapped alignments with Expectation values better than the user specified cutoff are reported. The extreme value distribution of BLAST Expectation values is precomputed against each precompiled database — this is one area that speeds up the algorithm considerably.

The math can be generalized thus: for any two sequences of length *m* and *n*, local, best alignments are identified as HSPs. HSPs are stretches of sequence pairs that can not be further improved by extension or trimming, as described above. For ungapped alignments, the number of expected HSPs with a score of at least *S* is given by the formula:

$$E = Kmne^{-\lambda S}$$

This is called an *E*-value for the score *S*. In a database search *n* is the size of the database in residues, so *N=mn* is the search space size. *K* and λ are be supplied by statistical theory, and, as mentioned above, can be calculated by comparison to precomputed, simulated distributions. These two parameters define the statistical significance of an *E*-value. The *E*-value defines the significance of the search. The smaller an *E*-value is, the more likely it is significant. A value of 0.001 is a good starting point for significance in most typical searches. In other words, in order to assess whether a given alignment constitutes evidence for homology, it helps to know how strong an alignment can be expected from chance alone.

FastA — and its family of relatives, developed by Bill Pearson at the University of Virginia (Pearson and Lipman, 1988 and Pearson, 1998)**.**

1) Works well for DNA against DNA searches (within limits of possible sensitivity);
2) Can find only one gapped region of similarity;
3) Relatively slow, should usually be run in the background;
4) Does not require specially prepared, preformatted databases.

FastA is a much older algorithm than BLAST. It was the first widely used, powerful sequence database searching program. Dr. Pearson continually refines the algorithm such that it remains a viable alternative to BLAST, especially if one is restricted to searching DNA against DNA without translation. It is also very helpful in situations where BLAST searches find no significant alignments — arguably, FastA may be more sensitive than BLAST in these situations.

The algorithm:

FastA also builds words of a set ktuple size, by default two for peptides. It then identifies all **exact** word matches between the sequence and the database members. Scores are assigned to each continuous, ungapped, diagonal by adding all of the exact match BLOSUM values. The ten highest scoring diagonals for each query-database pair are then rescored using BLOSUM similarities as well as identities and ends are trimmed to maximize the score. The best of each of these is called the *Init1* score. Next the program 'looks' around to see if nearby off-diagonal *Init1* alignments can be combined by incorporating gaps. If so, a new score, *Initn*, is calculated by summing up all the contributing *Init1* scores, penalizing gaps with a penalty for each. The program then constructs an optimal local alignment for all *Initn* pairs with scores better than some set threshold using a variation of dynamic programming "in a band." A sixteen residue band centered at the highest *Init1* region is used by default with peptides. A score is generated from this step known as the *opt* score. Next, FastA uses a simple linear regression against the natural log of the search set sequence length to calculate a normalized z-score for the sequence pair. Finally, it compares the distribution of these z-scores to an extreme-value distribution. Using this distribution, the program estimates the number of sequences that would be expected to have, purely by chance, a z-score greater than or equal to the z-score obtained in the search. This is reported as the Expectation score. Unfortunately, the z-score used in FastA and the previously discussed Monte Carlo style Z score are quite different and can not be directly compared. If the user requests pair-wise alignments in the output, then the program uses full Smith-Waterman local dynamic programming, not 'restricted to a band,' to produce its final alignments.

In review, both the BLAST and FastA family of programs base their Expectation "*E*" values on a more realistic 'extreme value distribution,' based on either real or simulated 'not significantly similar' database alignments, than Monte Carlo style Z scores do. Regardless, they follow Monte Carlo style Z scores fairly well. The higher the *E* value is, the more probable that the observed match is due to chance in a search of the same size database and the lower its Z score will be, i.e. is NOT significant. Therefore, **the smaller the *E* value, i.e. the closer it is to zero, the more significant it is and the higher its Z score will be**! The *E* value is the number that really matters.

III. Multiple Sequence Analysis.

A. Multiple Sequence Alignment.

More data yields stronger analyses — if done carefully! Otherwise, **it can confound the issue**. Patterns of conservation should become more apparent.

But, it is a very hard problem. **Dynamic programming's complexity increases exponentially with the number of sequences being compared** (complexity = [sequence length]$^{\text{number of sequences}}$). For instance, in pairwise dynamic programming you are dealing with a two dimensional matrix and the complexity of the solution is equal to the length of the longest sequence squared. Well, a three member comparison would be a matrix with three axes, the length of the longest sequence cubed, and so forth. It quickly becomes mind-boggling!

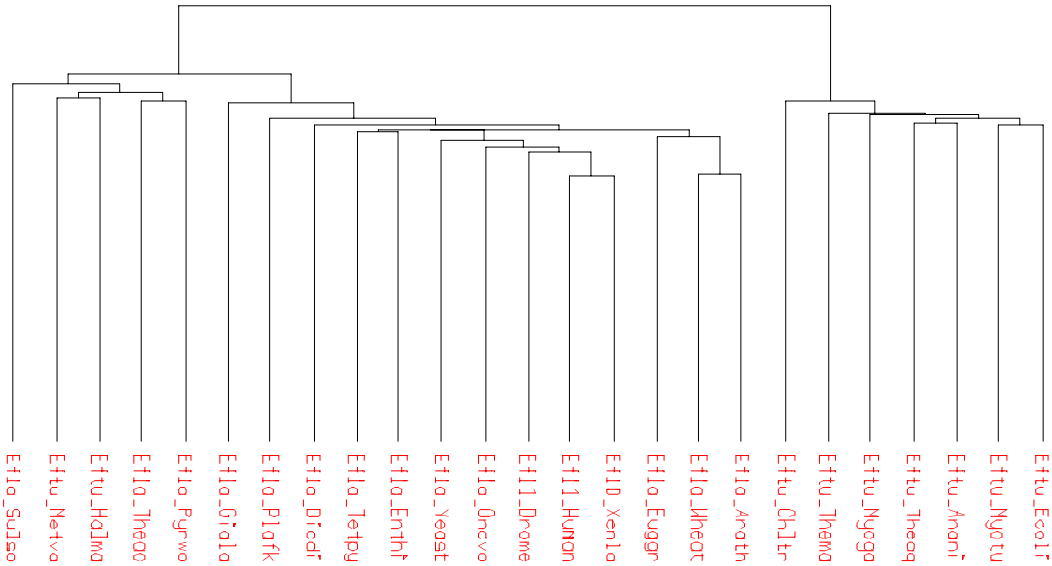
(one can draw a 3D matrix, but any more than that is impossible to even visualize)

Therefore, the most commonly used implementations of automated multiple alignment modify dynamic programming by establishing a pairwise order in which to build the alignment. This modification is known as **pairwise, progressive dynamic programming** and is generally attributed to Feng and Doolittle (1987). Search space is restricted at any one time to the local neighborhood of the full length of two sequences. It is implemented in the two most popular multiple alignment programs PileUp (GCG) and Clustal W (Thompson et al. 1994). Consider a group of six sequences. First all are compared to each other, pairwise, using normal dynamic programming. This establishes an order for the set, most to least similar. Then take the top two most similar sequences and align them using normal dynamic programming. Now create a consensus of the two and align that consensus to the third sequence using standard dynamic programming. Now create a consensus of the first three sequences and align that to the forth most similar. This process continues until you've worked your way through all six sequences. If two or more clusters are found in the process, they are zipped together along the way.

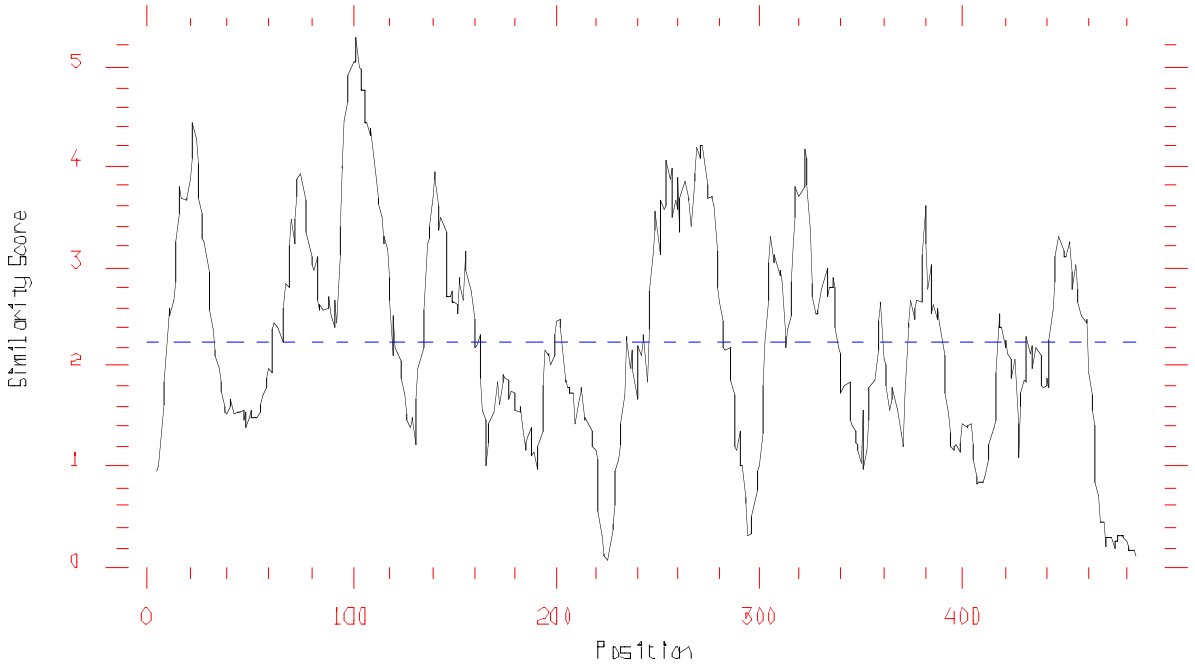
One liability of this method is it is entirely dependent on the order in which the sequences are aligned. Fortunately ordering them from most similar to least usually makes biological sense and empirically it works very well. However, the technique is very sensitive to the substitution matrix and gap penalties specified. Advanced programs let you vary the substitution matrix and gap penalties across the length of the alignment. And, just like with database searching, multiple sequence alignment should always be done on a protein level if at all possible; it all gets much more difficult with DNA. Twenty symbols are just much easier to align then only four; the signal to noise ratio is so much better. Therefore, as it is in database searching, translate nucleotide sequences to their protein counterparts if you are dealing with coding sequences before performing further analyses including multiple sequence alignment. If you need to align nucleotides because the region does not code for a protein, then automated methods may be able to help as a starting point, but they are certainly not guaranteed to come up with a biologically correct alignment. The resulting nucleotide alignment will probably have to be extensively edited, if it works at all. Furthermore, be sure that you are aligning things that make biological sense to align!

Check out the following example.

The protein: The Elongation Factors are a vital protein family crucial to protein biosynthesis. They are ubiquitous in all cellular life and must have been one of the very earliest enzymatic factories in life. Three GTP binding domains are essential for the activity of Elongation Factor Tu/1α. This protein promotes the GTP-dependent binding of aminoacyl-tRNA to the A-site of the ribosome during protein biosynthesis. A dendrogram shows the ordering process used by the multiple alignment algorithm:

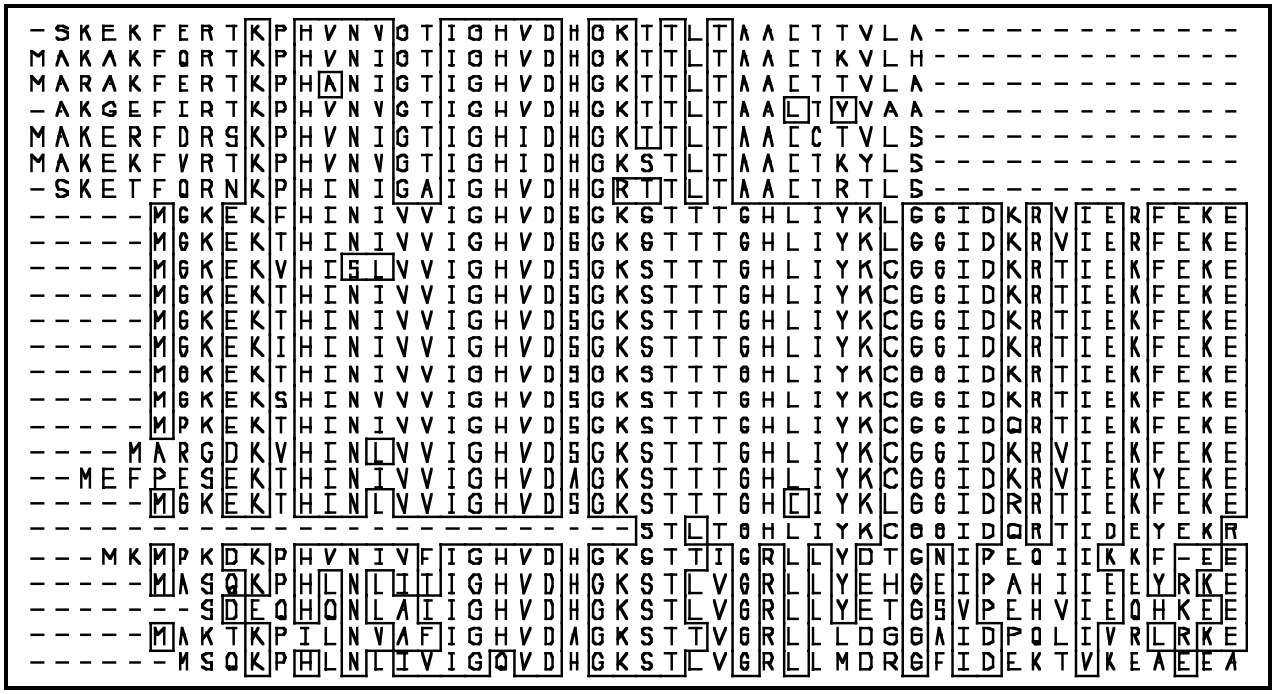


You can generate a plot that shows the running similarity along the length of a multiple sequence alignment using a sliding window approach. This qualitatively illustrates the positional conservation in the alignment. The following plot uses a window of ten and a BLOSUM62 scoring matrix; peaks correspond to the most conserved areas in the alignment, valleys to the least conserved areas. Let's concentrate on the first peak, centered at about residue twenty:



B. Motifs.

One very simplistic approach is to look at an alignment, see that certain regions are conserved, and create a consensus of that region. A multiple sequence alignment of elongation factor Tu/1α from many different organisms illustrates the conservation of the first of these GTP-binding domains, that area around position twenty:



G H V D H G K S

Based on experimental evidence, we know that the indicated region bounded by the Glycine and Serine above is essential. So we merely count up the various residues in those locations and assign the most common one to the consensus. Simple. But what about the fact that the middle Histidine isn't always a Histidine; in this data set, just as often it's a Serine and sometimes it's an Alanine. Other positions are also seen not be invariant. **A consensus isn't necessarily the biologically "correct" combination**. How do we include this other information? A simple consensus throws much of it away. Therefore, we need to adopt some sort of standardized ambiguity notation. The trick is to define a motif such that it minimizes false positives and maximizes true positives; i.e. it needs to be just discriminatory enough. The development of the exact motif is largely empirical; a pattern is made, tested against the database, then refined, over and over, although when experimental evidence is available, it is always incorporated. This approach is known as motif definition and fortunately a scientist in Switzerland, Amos Bairoch, has done it for tons of sequences!

Dr. Bairoch's compilation is known as the "**PROSITE** Database of protein families and domains" and contains 1079 documentation entries that describe 1459 different patterns, rules, and profiles/matrices (Release 16.33, of 25-Jan-2001). This type of pattern description for these characteristic local sequence areas are variously and confusingly known as motifs, templates, signatures, patterns, and even fingerprints; don't let the terminology bewilder you. They can all be thought of as a **one-dimensional** description of some sort of functional or otherwise constrained consensus region of a sequence alignment (e.g. glycosylation and phosphorylationsites, SH3-binding sites, nuclear localization sequence). Common motifs may or may not represent sequence homology — they do not all signify known function nor common origin.

In the above case for EF-Tu/1α, the corresponding PROSITE entry shows the motif's name and pattern, its location in the sequence, its biological activity, and it lists several representative references. Two PROSITE GTP binding motifs can be found in this alignment. The following describes the P-Loop GTP

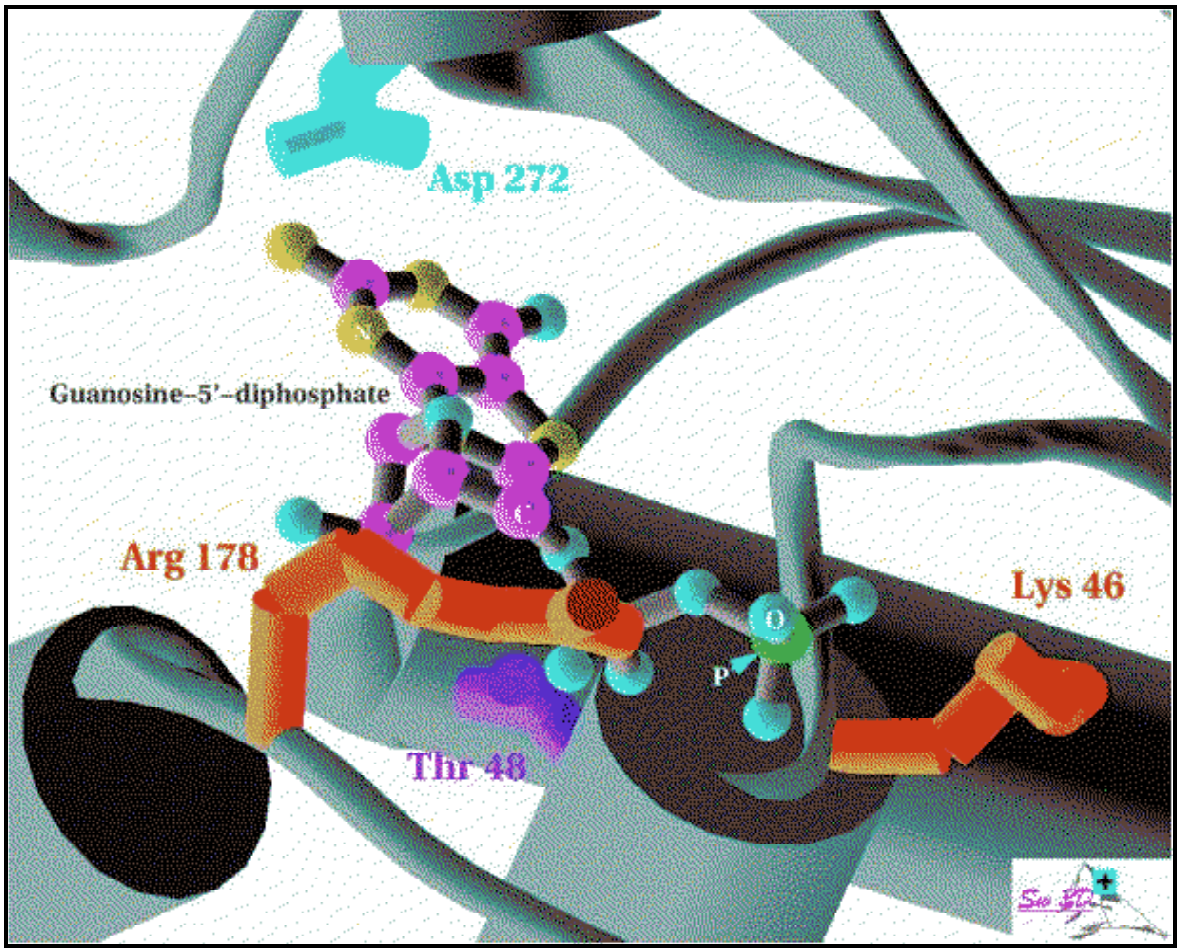
binding domain that we will concentrate on; the other is another GTP-binding domain, a bit downstream from this one, found in all Elongation Factors:

```
Atp_Gtp_A             (A,G)x4GK(S,T)
                       (G)x{4}GK(S)
            19: NIVVI    GHVDSGKS    TTTGH

****************************************
* ATP/GTP-binding site motif A (P-loop) *
****************************************

From sequence comparisons and crystallographic data analysis it has been shown
[1,2,3,4,5] that an appreciable proportion of proteins  that  bind ATP or GTP
share a number of more or less conserved sequence motifs.   The best conserved
of these motifs is a glycine-rich region, which probably forms a flexible loop
between a beta-strand and an alpha-helix.  This loop interacts with one of the
phosphate groups of the nucleotide.  This sequence motif is generally referred
to as the 'A' consensus sequence [1] or the 'P-loop' [5].

There are numerous ATP- or GTP-binding proteins in which the P-loop is found.
We list below a number of protein families for which the relevance of the
presence of such motif has been noted:

 - ATP synthase alpha and beta subunits.
 - Myosin heavy chains.
 - Kinesin heavy chains and kinesin-like proteins.
 - Guanylate kinase.
 - Thymidine kinase.
 - Thymidylate kinase.
 - Shikimate kinase.
 - Nitrogenase iron protein family (nifH/frxC).
 - ATP-binding proteins involved in 'active transport' (ABC transporters) [6].
 - ATP-dependent helicases [7,8].
 - GTP-binding elongation factors (EF-Tu, EF-1alpha, EF-G, EF-2, etc.).
 - Ras family of proteins (Ras, Rho, Rab, Ral, Ypt1, SEC4, etc.).
 - ADP-ribosylation factors family.
 - Bacterial dnaA protein.
 - Bacterial recA protein.
 - Bacterial recF protein.
 - Guanine nucleotide-binding proteins alpha subunits (Gi, Gs, Gt, G0, etc.).
 - DNA mismatch repair proteins mutS family.
 - Bacterial type II secretion system protein E.

Not all ATP- or GTP-binding proteins are picked-up by this motif.  A number of
proteins escape detection because the structure of their ATP-binding site is
completely different from that of the P-loop.  Examples of such proteins are
the E1-E2 ATPases or  the  glycolytic kinases.  In other ATP- or GTP-binding
proteins the flexible loop exists in a slightly different form; this is  the
case for tubulins or protein kinases.  A special mention must be reserved for
adenylate  kinase,  in  which  there is a  single  deviation  from the P-loop
pattern: in the last position Gly is found instead of Ser or Thr.

-Consensus pattern: [AG]-x(4)-G-K-[ST]
-Sequences known to belong to this class detected by the pattern: a majority.
-Other sequence(s) detected in SWISS-PROT: in addition to the proteins  listed
 above, the 'A' motif is also found in a number  of other proteins.  Most of
 these proteins probably bind a nucleotide, but others are definitively not
 ATP- or GTP-binding (as for example  chymotrypsin, or human ferritin light
 chain).

-Expert(s) to contact by email: Koonin E.V.
                                koonin@ncbi.nlm.nih.gov

-Last update: June 1994 / Text revised.

[ 1] Walker J.E., Saraste M., Runswick M.J., Gay N.J.
     EMBO J. 1:945-951(1982).
[ 2] Moller W., Amons R.
     FEBS Lett. 186:1-7(1985).
[ 3] Fry D.C., Kuby S.A., Mildvan A.S.
     Proc. Natl. Acad. Sci. U.S.A. 83:907-911(1986).
[ 4] Dever T.E., Glynias M.J., Merrick W.C.
     Proc. Natl. Acad. Sci. U.S.A. 84:1814-1818(1987).
[ 5] Saraste M., Sibbald P.R., Wittinghofer A.
     Trends Biochem. Sci. 15:430-434(1990).
[ 6] Higgins C.F., Hyde S.C., Mimmack M.M., Gileadi U., Gill D.R.,
```

Gallagher M.P.
J. Bioenerg. Biomembr. 22:571-592(1990).
[7] Hodgman T.C.
Nature 333:22-23(1988) and Nature 333:578-578(1988) (Errata).
[8] Linder P., Lasko P., Ashburner M., Leroy P., Nielsen P.J., Nishi K.,
Schnier J., Slonimski P.P.
Nature 337:121-122(1989).

The characteristic P-Loop is defined as (A,G)x4GK(S,T), i.e. either an Alanine or a Glycine, followed by four of anything, followed by an invariant Glycine-Lysine pair, followed by either a Serine or a Threonine. Exceptions are noted in the documentation.

This particular site has been very well researched and many three-dimensional structures are available for it. It always has a beta/alpha/beta secondary structure conformation and is sometimes known as the "Rossman Fold." Here the site is shown in the Guanine Nucleotide-Binding Protein G(I), Alpha-1 Subunit (Adenylate Cyclase-Inhibiting) from *Rattus norvegicus* (common rat):



OK, so motifs are one way to 'capture' the information of an important portion of an alignment. However, **motifs can not convey any degree of the 'importance' of the residues**. For instance, in the above P-Loop, is it better to have an Alanine or a Glycine in that first position or doesn't it matter? This lack of a sense of importance causes a loss of sensitivity. In order to convey the importance of each residue, a more 'robust' method must be used.

So, **given a multiple sequence alignment**, how can we use the information contained in it to find ever more remotely similar sequences? How do we **search and explore** Russell Doolittle's "**Twilight Zone**," i.e. those similarities below ~20% identity, those Z scores below ~5, those BLAST/Fast *E* values above ~10^{-5}

or so? Just because a similarity score between two sequences is quite low, we do not automatically know that the two structures do not fold in a similar manner and perform a similar function, nor do we know whether they are truly homologous!

Obviously much of the information in a multiple sequence alignment is 'noise' at this similarity level. Searching with the full-length of any of its members would not gain us anything. Too much evolution has happened over its full length — the 'history' of most of it has been lost. However, certain regions of the alignment have been constrained throughout evolutionary history. They are somehow very 'important' to the sequence — functionally, structurally, or whatever — can we use them to find other sequences with similarly constrained regions? Yes, we can.

C. The Profile Method.

Enter **two-dimensional** consensus techniques. The basic idea is to tabulate how often every possible residue occurs at each position. This information is stored in a matrix twenty residues wide by the length of your pattern. Does this remind you of anything? We're talking about the same concept as a symbol substitution table or scoring matrix, in other words a very special PAM style table — a matrix custom built based on a specific pattern in a collection of related sequences.

However, Michael Gribskov and David Eisenberg (1987) had an even better idea — Why not, in addition to building a matrix specific to an alignment, build a matrix where the most conserved areas of the alignment receive the highest values and those areas of the alignment with less conservation receive smaller values. They have implemented these ideas in a method known as Profile Analysis, that consists of several programs:

ProfileMake —	creates the profile from a multiple sequence alignment.
ProfileSearch —	searches other sequences (the database) with a profile.
ProfileSegments —	aligns the output list of a ProfileSearch.
ProfileGap —	aligns individual sequences to a profile.
ProfileScan —	searches sequences against library of over 600 validated profiles. These were built by Michael Gribskov and are based on PROSITE.

Profile analysis allows all the information content of the specified portion of a multiple sequence alignment to be fully used. (It also works on the full length of an alignment but is much more powerful on local domains.) All other methods of describing a region within an alignment such as consensus or pattern descriptors either through away too much information or become too ambiguous. Two-dimensional weight matrix type searching is way more powerful than simple one-dimensional string searching. Profiles are a special type of weight matrix where —

conserved areas of the alignment receive the most importance and variable regions hardly matter!

By virtue of the position specific, variable weighting of a profile, the creation of gaps is highly discouraged in conserved areas but occurs easily in more variable regions in subsequent profile alignments and searches. This is what gives profiles so much power. Furthermore, just as in BLAST and FastA searches, a realistic significance parameter is estimated by ProfileSearch. In the case of profile searching that parameter is a Z score based on the distance, the number of standard deviations away, from the rest of the insignificant database matches; rather than a simplistic, non-biological, Monte Carlo jumbled statistics model.

The greatest amount of conservation of the P-Loop region in the above alignment is centered about residue twenty or so. What happens if we prepare a profile around this region? What does this profile look like? It's a big table of numbers that doesn't mean a whole lot on first inspection to us mere mortals, but let's check it out (I apologize for the tiny text; I had to use it to fit it on the page.):

Cons	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y	Z	Gap	Len..
E	11	20	-11	27	33	-21	16	10	-4	10	-9	-6	16	6	18	0	8	17	-3	-29	-15	26	12	12
K	0	27	-40	21	22	-47	-6	7	-13	100	-20	13	27	7	27	53	14	13	-13	5	-40	28	12	12
! 11																								
P	13	3	4	3	3	-13	9	2	3	3	-2	-1	1	28	4	3	11	20	9	-21	-16	4	12	12
H	-7	26	-6	26	26	-6	-14	99	-18	6	-12	-19	33	13	46	33	-13	-6	-19	-7	20	33	12	12
I	3	-7	2	-7	-6	19	-6	-9	43	-7	29	22	-10	-4	-6	-10	-4	6	38	-17	1	-5	12	12
N	14	73	-19	47	33	-34	27	33	-20	27	-27	-20	100	0	26	7	22	14	-20	-20	-7	27	12	12
I	1	-10	-1	-10	-8	26	-9	-10	46	-8	34	27	-12	-6	-8	-12	-6	5	40	-12	4	-7	12	12
V	**15**	2	7	3	1	-1	**20**	-9	**24**	-6	**14**	11	-3	6	-3	-11	4	10	**37**	-30	-9	-1	12	12
V	9	-4	7	-5	-4	5	7	-8	29	-4	20	15	-6	4	-7	-9	0	19	36	-21	-2	-5	12	12
I	0	-16	16	-16	-16	55	-24	-24	118	-16	63	47	-24	-16	-24	-24	-8	16	87	-39	8	-16	12	12
G	55	47	16	55	39	-47	118	-16	-24	-8	-39	-24	31	24	16	-24	47	31	16	-79	-55	24	12	12
H	-6	27	-7	27	27	-8	-13	100	-20	7	-13	-20	34	14	48	34	-13	-7	-20	-7	19	34	12	12
! 21																								
V	11	-12	12	-12	-12	13	11	-18	67	-12	48	36	-18	5	-12	-18	-6	12	89	-47	-6	-12	12	12
D	24	87	-39	118	79	-79	55	31	-16	24	-39	-31	55	8	55	0	16	16	-16	**-87**	-39	71	12	12
S	9	12	11	11	11	-8	8	22	-7	5	-10	-10	14	11	11	9	23	4	-6	1	-2	9	12	12
G	55	47	16	55	39	-47	118	-16	-24	-8	-39	-24	31	24	16	-24	47	31	16	-79	-55	24	12	12
K	0	27	-40	20	20	-47	-7	7	-14	100	-20	13	27	7	27	55	13	13	-14	8	-40	27	12	12
S	19	14	30	10	10	-14	27	-9	-2	10	-17	-12	14	19	-5	3	63	24	-2	7	-19	1	100	100
T	40	20	20	20	20	-30	40	-10	20	20	-10	0	20	30	-10	-10	30	**150**	20	-60	-30	10	100	100
T	8	-4	-9	-4	0	13	1	-6	18	0	23	22	-2	2	-4	-9	0	34	18	-6	-2	-1	100	100
T	19	8	10	8	8	-12	19	-6	16	8	1	4	7	14	-6	-6	13	69	18	-32	-14	3	100	100
G	40	24	10	28	21	-27	61	-8	-11	-4	-19	-11	16	16	9	-14	26	18	9	-44	-28	13	100	100
! 31																								
H	10	11	-1	11	11	-10	1	34	-8	7	-8	-5	13	11	19	18	0	1	-6	-1	0	14	100	100
L	-4	-20	-27	-20	-13	50	-21	-10	43	-13	62	53	-17	-13	-7	-17	-15	-2	40	13	12	-9	100	100
*	20	0	0	27	12	3	73	70	65	46	38	0	24	11	5	6	33	85	65	0	0	0		

On closer inspection, the matrix begins to make some sense. Across the top are all possible residues. The first column is that residue that received the highest score in the program — the consensus. But notice the interior of the matrix. Numbers bounce all over the place, from 150 to -87. What's that all about? Well, without going into all the fancy mathematics involved, based on the alignment we fed it and the initial scoring matrix used (by default the BLOSUM62 matrix but you can specify others) the program has scaled those positions which are most important up and those positions least important down. For instance the Threonine at position 27 in our alignment is the only residue absolutely conserved throughout — it gets the highest score! The Aspartate at position 22 substituted with a Tryptophan would never happen, hence the -87 score. Tryptophan is the most conserved residue on both the PAM and BLOSUM matrix series and the Aspartate is conserved at all positions in our alignment that have residues at that position — the negative matrix score of any substitution to Tryptophan times the high conservation at that position for Aspartate equals the most negative score in the profile. How about those positions where the conservation is not as striking? Position 16 is a good one to pick on. Valine is the assigned consensus residue because it has the highest score, 37, but Glycine also occurs several times, a score of 20. However, other residues are ranked in the substitution matrices as being quite similar to Valine; therefore Isoleucine and Leucine also get similar scores, 24 and 14, and Alanine occurs some of the time in the alignment so it gets a comparable score, 15. But realize that all of these numbers are way less than the highest numbers in the matrix — because the position is not well conserved all the values are fairly mediocre at that position.

OK, but what about the last two columns in the matrix, and the last row? The last row is the composition of the whole profile. Our alignment has twenty Alanines overall and no Cysteines — big deal. However, the last two columns are very important! They relate to gap penalties in any subsequent analysis with this particular profile. Remember that I stated that gaps are more easily introduced into variable regions than conserved regions in profile analysis. Well, this is where that comes from. The first column is the gap opening penalty and the second is the gap extension penalty for that particular spot in any subsequent analysis (both as a percentage). Unlike all other implementations of dynamic programming, **the penalties are not constant throughout the length of the profile**. Those regions where conservation is highest, receive 100% of the assigned gap penalty. Those regions with less conservation, receive less gap penalty. Here, everywhere else only gets 12% of the assigned gap penalty!

One very important consideration that I have not mentioned yet is sequence weighting. Each sequence, by default, contributes an equal importance to the profile. This may or may not be appropriate for your situation. Consider a multiple sequence alignment with several very similar sequences and a few more divergent ones. In this case, the contribution of the more divergent sequences would be "lost" among the overpowering signal of all the similar ones. It may be appropriate to increase the weight of the more divergent sequence to even out all the sequences' contribution. This is often done in quite an "ad-hoc" manner, although a similarity dendrogram, as seen earlier, can aid the decision. Those clusters with less than their "fair share" of contribution, have their weights increased. To figure out the appropriate weighting factors, choose the largest cluster, assign each member a weight of one and then propagate that up throughout the clusters. (If you're interested, I can explain further personally.) The process of weighting your sequences appropriately and repeatedly searching the database with your profile and then adjusting the weights and including or excluding subsequent members of the profile is known as "validating" your profile.

Submitting this profile to ProfileSearch reflects the robustness of the method. Several entries in Swiss-Prot with a P-Loop, beyond just the elongation factors, are found, even by this very simple, unrefined profile based on just a few of the elongation factor Tu entries. Naturally, a P-Loop profile that includes all known representatives would be much more powerful. I'll list some of the more interesting entries:

```
            Normalization equation:

                    Calc_Score = 7.97 * ( 1.0 - exp(-0.0029*SeqLen - 1.1983) )

            Correlation for curve fit: 0.959

            Z score calculation:
            Average and standard deviation calculated using 41033 scores
            121 of 41154 scores were rejected

                    Z_Score = ( Score/Calc_Score - 1.001 ) / 0.101

             Sequence  Strd ZScore   Orig Length ! Documentation  ..

SW:EF1A_ENTHI         +   10.92  15.30    430 ! P31018 entamoeba histolytica. el
ongation factor 1-alpha (ef-1-alpha). 7/93
SW:EF1A_WHEAT         +   10.79  15.28    447 ! Q03033 triticum aestivum (wheat)
. elongation factor 1-alpha (ef-1-alpha). 7/93
SW:EF12_HUMAN         +   10.70  15.27    463 ! Q05639 homo sapiens (human). elo
ngation factor 1-alpha 2 (ef-1-alpha-2) (statin s1)
SW:EF1A_MOUSE         +   10.68  15.25    461 ! P10126 mus musculus (mouse). elo
ngation factor 1-alpha (ef-1-alpha). 7/93
////////////////////////////////////////////////////////////////////////////////
SW:IF2G_HUMAN         +    4.69  10.85    472 ! P41091 homo sapiens (human). tra
nslational initiation factor 2 gamma subunit (eif-2
SW:EFG2_YEAST         +    4.42  11.21    819 ! P39677 saccharomyces cerevisiae
(baker's yeast). elongation factor g, mitochondrial
SW:YIHK_ECOLI         +    4.41  10.90    591 ! P32132 escherichia coli. 65.4 kd
 protein in glna-fdhe intergenic region (o591). 2/9
SW:EF2_SULSO          +    4.41  11.11    735 ! P30925 sulfolobus solfataricus.
elongation factor 2 (ef-2). 2/94
SW:THI_CYACA          +    4.41   8.97    107 ! P37395 cyanidium caldarium. thio
redoxin. 10/94
SW:HYBG_ECOLI         +    4.38   8.77     82 ! P37185 escherichia coli. hydroge
nase-2 operon protein hybg. 2/95
SW:TET5_ENTFA         +    4.26  10.86    639 ! P11131 enterococcus faecalis (st
reptococcus faecalis). tetracycline resistance prot
SW:TETM_UREUR         +    4.24  10.85    639 ! P09757 ureaplasma urealyticum. t
etracycline resistance protein tetm. 10/93
SW:IF2M_YEAST         +    4.19  10.86    676 ! P25038 saccharomyces cerevisiae
(baker's yeast). initiation factor if-2, mitochondr
////////////////////////////////////////////////////////////////////////////////
SW:TSX_ENTAE          +    3.03   9.08    294 ! P40785 enterobacter aerogenes (a
erobacter aerogenes). nucleoside-specific channel-f
SW:IF2_BACST          +    3.01  10.04    741 ! P04766 bacillus stearothermophil
us. initiation factor if-2. 2/94
SW:YP12_STAAU         +    3.01   8.06    102 ! P03859 staphylococcus aureus. hy
pothetical 12.4 kd protein (reading frame c). 11/90
SW:PKD1_DICDI         +    3.00   9.45    416 ! P34100 dictyostelium discoideum
(slime mold). developmentally regulated protein kin
```

A sample alignment of one of the quite low scoring segments shows the type of importance that the algorithm places on those most conserved residues:

```
                Quality:   9.25            Length:     48
                  Ratio:   0.39              Gaps:      2

       epiz_staep x ef-p-loop.prf

                      .         .         .         .
       S    192 QKMEINDLNFTKDNKQIINSISLDLHKGYIYNIIGESGCGKSTLLNIL 239
               .| :||          |.             :||: | ||||  | |
       P      1 EKPHIN.........IV.............VIGHVDSGKSTTTGHL 24
```

Check out the P-Loop; even if this hypothetical protein has no known function, it definitely binds GTP with a Rossman Fold motif! In spite of a relatively terrible Z score of 3.39, the alignment was discovered by profile analysis. Another thing to notice in all profile alignments is well illustrated here. Notice the 'gappy,' clustered residue, appearance of the alignment. This is due to the variable nature of the values in the profile — important areas will not be gapped — yet gaps are easily inserted into the remainder of the sequence!

ProfileScan in the Wisconsin Package includes over 600 profiles, over half of the motifs in PROSITES! Therefore, you can scan your sequence with Motifs first to see if it has any of Dr. Bairoch's pattern descriptions and then if you don't find anything that way, rescan it with the profile libary. As you've now seen, this is much more sensitive. When I scanned the human Elongation Factor 1α from my alignment with ProfileScan, this is what I found:

```
 Profile: profiledir:efactor_gtp.prf

   Gap weight:  4.50     Gap Length weight:   0.05   Quality:  14.97       Gaps: 0
   Ave match:   0.14     Ave mismatch     :  -0.11     Ratio:   0.28     Length: 53

   Normalized quality:  3.35
                .         .         .         .         .
S     46 KFEKEAAEMGKGSFKYAWVLDKLKAERERGITIDISLWKFETSKYYVTIIDAP 98
         .|.::|::.|:::.: :.:||::::|||||||||::. .|:|.|  .::||:|
P      3 AFEKEAAEKGKGSAKEAFVLDKMKEERERGITIDIAHVEFETEKWHFTIIDAP 55

*****************************************
* GTP-binding elongation factors signature *
*****************************************

Elongation factors  [1,2]  are proteins  catalyzing the  elongation of peptide
chains in protein biosynthesis. In both prokaryotes and  eukaryotes, there are
three  distinct types  of elongation  factors,  as  described in the following
table:

 ------------------------------------------------------------------------------
 Eukaryotes   Prokaryotes   Function
 ------------------------------------------------------------------------------
 EF-1alpha    EF-Tu         Binds GTP and  an aminoacyl-tRNA;  delivers  the
                            latter to the A site of ribosomes.
 EF-1beta     EF-Ts         Interacts with EF-1a/EF-Tu  to  displace GDP and
                            thus allows the regeneration of GTP-EF-1a.
 EF-2         EF-G          Binds GTP and peptidyl-tRNA and translocates the
                            latter from the A site to the P site.
 ------------------------------------------------------------------------------
The GTP-binding elongation factor family also includes the following proteins:

 - Eukaryotic  peptide  chain  release  factor GTP-binding subunits [3]. These
   proteins interact  with  release  factors  that bind to ribosomes that have
   encountered a  stop  codon  at  their decoding site and help them to induce
   release of  the  nascent  polypeptide.  The yeast protein was known as SUP2
   (and also as SUP35, SUF12 or GST1) and the human homolog as GST1-Hs.
 - Prokaryotic  peptide  chain release factor 3 (RF-3) (gene prfC).  RF-3 is a
   class-II RF, a GTP-binding  protein  that  interacts  with class I RFs (see
   <PDOC00607>) and enhance their activity [4].
 - Prokaryotic GTP-binding protein lepA and its  homolog  in yeast (gene GUF1)
   and in Caenorhabditis elegans (ZK1236.1).
 - Yeast HBS1 [5].
 - Rat statin S1 [6], a protein of unknown function which is highly similar to
```

```
   EF-1alpha.
 - Prokaryotic selenocysteine-specific elongation factor selB [7], which seems
   to replace  EF-Tu  for  the insertion of selenocysteine directed by the UGA
   codon.
 - The tetracycline resistance proteins tetM/tetO [8,9] from various  bacteria
   such as  Campylobacter jejuni, Enterococcus faecalis,  Streptococcus mutans
   and Ureaplasma urealyticum. Tetracycline binds to the prokaryotic ribosomal
   30S subunit and inhibits binding of aminoacyl-tRNAs. These proteins abolish
   the inhibitory effect of tetracycline on protein synthesis.
 - Rhizobium nodulation protein nodQ [10].
 - Escherichia coli hypothetical protein yihK [11].

In EF-1-alpha, a specific region  has  been  shown [12] to  be  involved  in a
conformational change mediated by the hydrolysis of GTP to GDP. This region is
conserved in  both EF-1alpha/EF-Tu as well as EF-2/EF-G and thus seems typical
for GTP-dependent proteins which bind non-initiator tRNAs to the ribosome. The
pattern we  developed  for  this  family  of  proteins  include that conserved
region.

-Consensus pattern: D-[KRSTGANQFYW]-x(3)-E-[KRAQ]-x-[RKQD]-[GC]-[IVMK]-[ST]-
                    [IV]-x(2)-[GSTACKRNQ]
-Sequences known to belong to this class detected by the pattern: ALL, except
 for 11 sequences.
-Other sequence(s) detected in SWISS-PROT: NONE.
-Last update: November 1997 / Text revised.

[ 1] Concise Encyclopedia Biochemistry, Second Edition, Walter de Gruyter,
     Berlin New-York (1988).
[ 2] Moldave K.
     Annu. Rev. Biochem. 54:1109-1149(1985).
[ 3] Stansfield I., Jones K.M., Kushnirov V.V., Dagkesamanskaya A.R.,
     Poznyakovski A.I., Paushkin S.V., Nierras C.R., Cox B.S.,
     Ter-Avanesyan M.D., Tuite M.F.
     EMBO J. 14:4365-4373(1995).
[ 4] Grentzmann G., Brechemier-Baey D., Heurgue-Hamard V., Buckingham R.H.
     J. Biol. Chem. 270:10595-10600(1995).
[ 5] Nelson R.J., Ziegelhoffer T., Nicolet C., Werner-Washburne M., Craig E.A.
     Cell 71:97-105(1992).
[ 6] Ann D.K., Moutsatsos I.K., Nakamura T., Lin H.H., Mao P.-L., Lee M.-J.,
     Chin S., Liem R.K.H., Wang E.
     J. Biol. Chem. 266:10429-10437(1991).
[ 7] Forchammer K., Leinfeldr W., Bock A.
     Nature 342:453-456(1989).
[ 8] Manavathu E.K., Hiratsuka K., Taylor D.E.
     Gene 62:17-26(1988).
[ 9] Leblanc D.J., Lee L.N., Titmas B.M., Smith C.J., Tenover F.C.
     J. Bacteriol. 170:3618-3626(1988).
[10] Cervantes E., Sharma S.B., Maillet F., Vasse J., Truchet G., Rosenberg C.
     Mol. Microbiol. 3:745-755(1989).
[11] Plunkett G. III, Burland V.D., Daniels D.L., Blattner F.R.
     Nucleic Acids Res. 21:3391-3398(1993).
[12] Moller W., Schipper A., Amons R.
     Biochimie 69:983-989(1987).
```

Unfortunately and quite surprisingly, the P-Loop signature was not incorporated into Dr. Gribskov's profile library; it was not found. However, human Elongation Factor 1α is seen to contain the Elongation Factor specific GTP-binding domain, the other PROSITE pattern found by Motifs that we have not discussed. As always, do not place all of your confidence in any one analysis — you need to combine all available sources of analyses and information in order to ascertain the function of any unknown sequence!

D. Advanced Methodologies.

One can achieve many wild and wonderful objectives using various combinations of the techniques discussed here. They all take advantage of the information content of a multiple sequence alignment.

Profile technology is also used in NCBI's PSI-BLAST implementation to automate iterative database searching. See NCBI's BLAST Web pages for more information on this procedure.

Expectation Maximization and Hidden Markov Chain Models are two other statistics techniques used for profile discovery. A neat feature of these two methods is the input sequences are not aligned beforehand

— the programs find the conserved features of unaligned sequences! Just as in all profile techniques, the result is a weighted matrix that can then be used for further analyses, such as database searching and sequence alignment.

Another huge area that we don't have much time to discuss is protein structure analysis. Protein structural prediction is fraught with difficulties. However, using comparative multiple sequence approaches is by far the most reliable strategy. In fact, in my opinion, the best predictor of secondary structure around, available on the Web at http://www.embl-heidelberg.de/predictprotein/predictprotein.html, uses multiple sequence alignment profile techniques along with neural net technology. PredictProtein is a service offered by the Protein Design Group at the European Molecular Biology Laboratory, Heidelberg, Germany. A multiple sequence alignment is created and a secondary structure prediction is produced by the profile network method (PHD). PHD is rated at an expected 70.2% average accuracy for the three states helix, strand, and loop (Rost and Sander, 1994). In fact, even three-dimensional modeling without crystal coordinates is possible. This is "homology modeling." It will often lead to remarkably accurate representations if the similarity is great enough between your protein and one in which the structure has been solved through experimental means. Automated homology modeling is even available through the Web as SwissModel at Amos Bairoch's Expasy server in Switzerland (see e.g. Peitsch, 1996) (http://www.expasy.ch/swissmod/SWISS-MODEL.html).

And finally, an area that is my passion, the evolutionary relationships of the same gene between different organisms (orthologous homologs) and even of duplicated genes within an organism (paralogous homologs) can be ascertained using the techniques of molecular phylogenetic inference and multiple sequence alignment. The software packages PAUP* (Swofford, 1997) and PHYLIP (Felsenstein, 1993) have many programs for performing these types of analyses. If this area really interests you, I would recommend looking into participating in one of the many fine courses offered worldwide in phylogenetic inference. I am personally involved in the Molecular Evolution Course offered every August at the Woods Hole Marine Biological Laboratory. Check it out: http://newfish.mbl.edu/Course/.

Gunnar von Heijne in his quite readable but well dated treatise, *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit* (1987), provides a very appropriate conclusion:

> "Think about what you're doing; use your knowledge of the molecular system involved to guide both your interpretation of results and your direction of inquiry; use as much information as possible; and do not blindly accept everything the computer offers you."

He continues:

> ". . . if any lesson is to be drawn . . . it surely is that to be able to make a useful contribution one must first and foremost be a biologist, and only second a theoretician . . . . We have to develop better algorithms, we have to find ways to cope with the massive amounts of data, and above all we have to become better biologists. But that's all it takes."

References.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990) Basic Local Alignment Tool. *Journal of Molecular Biology* **215**, 403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a New Generation of Protein Database Search Programs. *Nucleic Acids Research* **25**, 3389-3402.

Bairoch A. (1992) PROSITE: A Dictionary of Sites and Patterns in Proteins. *Nucleic Acids Research* **20**, 2013-2018.

Felsenstein, J. (1993) PHYLIP (Phylogeny Inference Package) version 3.5c. Distributed by the author. Dept. of Genetics, University of Washington, Seattle, Washington, U.S.A.

Feng, D.F. and Doolittle, R.F. (1987) Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees. *Journal of Molecular Evolution*. **25**, 351-360.

Genetics Computer Group (GCG), Inc. (Copyright 1982-2000) *Program Manual for the Wisconsin Package*, Version 10.1, Madison, Wisconsin, USA 53711.

Gribskov, M. and Devereux, J., editors (1992) *Sequence Analysis Primer*. W.H. Freeman and Company, New York, N.Y., U.S.A.

Gribskov M., McLachlan M., Eisenberg D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. U.S.A.* **84**, 4355-4358.

Henikoff, S. and Henikoff, J.G. (1992) Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences U.S.A.* **89**, 10915-10919.

Needleman, S.B. and Wunsch, C.D. (1970) A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* **48**, 443-453.

Pearson, P., Francomano, C., Foster, P., Bocchini, C., Li, P., and McKusick, V. (1994) The Status of Online Mendelian Inheritance in Man (OMIM) medio 1994. *Nucleic Acids Research* **22**, 3470-3473.

Pearson, W.R. (1998) Empirical Statistical Estimates for Sequence Similarity Searches. *Journal of Molecular Biology*. **276**, 71-84.

Pearson, W.R. and Lipman, D.J. (1988) Improved Tools for Biological Sequence Analysis. *Proceedings of the National Academy of Sciences U.S.A.* **85**, 2444-2448.

Peitsch, M.C. (1996) ProMod and Swiss-Model: Internet-based tools for automated comparative protein modelling. *Biochemical Society Transaction s*. **24**, 274-279.

Rost, B. and Sander, C. (1993) Prediction of Protein Secondary Structure at Better than 70% Accuracy. *Journal of Molecular Biology* **232**, 584-599.

Smith, S.W., Overbeek, R., Woese, C.R., Gilbert, W., and Gillevet, P.M. (1994) The Genetic Data Environment, an Expandable GUI for Multiple Sequence Analysis. *CABIOS*, **10**, 671-675.

Schwartz, R.M. and Dayhoff, M.O. (1979) Matrices for Detecting Distant Relationships. In *Atlas of Protein Sequences and Structure*, (M.O. Dayhoff editor) **5**, Suppl. **3**, 353-358, National Biomedical Research Foundation, Washington D.C., U.S.A.

Smith, T.F. and Waterman, M.S. (1981) Comparison of Bio-Sequences. *Advances in Applied Mathematics* **2**, 482-489.

Sundaralingam, M., Mizuno, H., Stout, C.D., Rao, S.T., Liedman, M., and Yathindra, N. (1976) Mechanisms of Chain Folding in Nucleic Acids. The Omega Plot and its Correlation to the Nucleotide Geometry in Yeast tRNAPhe1. *Nucleic Acids Research* **10**, 2471-2484.

Swofford, D.L., PAUP (Phylogenetic Analysis Using Parsimony) (1989-1993) Illinois Natural History Survey, (1994) personal copyright, and (1997) Smithsonian Institution, Washington D.C., U.S.A.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, **22**, 4673-4680.

von Heijne, G. (1987) *Sequence Analysis in Molecular Biology; Treasure Trove or Trivial Pursuit.* Academic Press, Inc., San Diego, California, U.S.A.

Wilbur, W.J. and Lipman, D.J. (1983) Rapid Similarity Searches of Nucleic Acid and Protein Data Banks. *Proceedings of the National Academy of Sciences U.S.A.* **80**, 726-730.

Zuker, M. (1989) On Finding All Suboptimal Foldings of an RNA Molecule. *Science* **244**, 48-52.