A2
2.2 Aims of bioinformatics

The aims of bioinformatics are three-fold. First, at its simplest bioinformatics organises data in a way that allows researchers to access existing information and to submit new entries as they are produced, eg the Protein Data Bank for 3D macromolecular structures (6, 7). While data-curation is an essential task, the information stored in these databases is essentially useless until analysed. Thus the purpose of bioinformatics extends far beyond mere volume control. The second aim is to develop tools and resources that aid in the analysis of data. For example, having sequenced a particular protein, it is of interest to compare it with previously characterised sequences. This requires more than just a straightforward database search. As such, programs such as FASTA (8) and PSI-BLAST (9) must consider what constitutes a biologically significant resemblance. Development of such resources requires extensive knowledge of computational theory, as well as a thorough understanding of biology. The third aim is to use these tools to analyse the data and interpret the results in a biologically meaningful manner. Traditionally, biological studies examined individual systems in detail, and frequently compared them with a few that are related. In bioinformatics, we can also conduct global analyses of all the available data with the aim of uncovering common principles that apply across many systems and highlight features that are unique to some.

In this review, we provide an introduction to bioinformatics. We focus on the first and third aims just described, with particular reference to the keywords underlined in the definition: ***information***,***informatics***, ***organisation***, ***understanding***, ***large-scale*** and ***practical applications***. Specifically, we discuss the range of data that are currently being examined, the databases into which they are organised, the types of analyses that are being conducted using transcription regulatory systems as an example, and finally discuss some of the major practical applications of bioinformatics.
Bioinformatics: wave of the future Roby Ajith Monday, June 09, 2003

Bioinformatics: wave of the future

Everyone is now talking about bioinformatics. This is the new segment created by the merger of two hot areas: information technology and biotechnology. So this has become one of the most promising field for job hunters. What are the opportunities available in this field?

Bioinformatics is the application of computer technology to the management of biological information. It combines computer science with biology and genetics with a good-sized dollop of mathematics, statistics and other medical specialties thrown into the mix. Computers are used to gather, store, analyze and integrate biological and genetic information which can then be applied to gene-based drug discovery and development. Bioinformatics is not just a useful tool in biological research or drug development. It is an indispensable ally of researchers.

"The technology is versatile and can be applied whenever gene, protein and cell research are used for the discovery of a new drug or a new herbicide/herbicide-resistant crop combination. Drug toxicology, pharmacogenetics and clinical trial studies can also benefit from this technology which can even be used to genetically engineer crops and livestock that have enhanced nutritional qualities and the ability to produce pharmaceuticals," said Anuradha Acharya, CEO, Ocimum Biosolutions, Hyderabad.

Recent years have seen an explosive growth in biological data. It should be managed and stored for various purposes. Also the managed data should be in tune with the current times. Here comes the relevance of bioinformatics. Large sequencing projects are producing increasing quantities of nucleotide sequences. The contents of nucleotide databases are doubling in size approximately every 14 months.

Without bioinformatics, new research in most fields of medicine and biology would come to a standstill. The explosion of publicly available genomic information resulting from the Human Genome Project has precipitated the need for bioinformatics capabilities. The enormous growth of biological data led to the development of several things. First, all these data need to be stored. The second requirement is the need for radical new methods for analyzing these huge databases. Thirdly, powerful hardware is required to carry out the task of analyzing these databases. For instance, IBM is giving away free Web services technology to help scientists to track down DNA as the company continues its push itself into the promising life sciences arena.

The latest release of GenBank (V.102) exceeded one billion base pairs. Not only the size of sequence data is rapidly increasing but also the number of characterized genes from many organisms and protein structures doubles every two years. To cope with this great quantity of data, a new scientific discipline has emerged: bioinformatics, biocomputing or computational
biology.

How to become a bioinformatics expert?

Bioinformatics combines the tools and techniques of mathematics, computer science and biology in order to understand the biological significance of a variety of data. So if you like to get into this new scientific field you should be fond of these ‘classic’ disciplines. Because the field is so new, almost everyone in it did something else before. Some biologist went into bioinformatics by picking up programming but others entered via the reverse route.

Eligibility: B.Sc/M.Sc
(Microbiology/Biochem/biotechnology/Agriculture/Horticulture/Seri/Food Technology/Organic Chemistry/botany/zoology/statistics/bio science & other divisions of life sciences, BE/B.Tech/B.VSc/M.VSc/B.Pharma, MBBS/BDS. ·

As a biologist, what skills do I need to make the transition?

National biotech institutes	
Chandigarh	The Institute of Microbial Technology, Chandigarh
New Delhi	The National Institute of Immunology, New Delhi IGIB, New Delhi
Hyderabad	The Center for Cellular and Molecular Biology, Hyderabad Centre for DNA Fingerprinti- ng Technology, Hyderabad
Bangalore	The National Center for Biological Sciences, Bangalore The Jawaharlal Nehru Center for Advanced Scientific Research, Bangalore

In addition the extensive knowledge of the run-of-the mill molecular biology packages (GCG, BLAST etc.), you will need to learn web and programming skills including HTML, Perl, JAVA and C++ and be familiar with a variety of operating systems (especially UNIX). Relational database skills are very much sought after. So knowledge of SQL and a major database application such as Sybase or Oracle will be highly advantageous. One area of bioinformatics that is set to expand is the determination of relationships between structures and sequence. If you wish to enter this field, you will need to learn all you can about structural biology and modeling, mathematical optimization, computer graphics theory and linear algebra.

Is it easier to move from biology to computers or the reverse?

The answer depends on whether you are talking to a computer scientist who 'does' biology or a molecular biologist who 'does' computing. Most of what you will read in the popular press is that the importance of interdisciplinary scientists cannot be over-stressed and that the young people getting the top jobs in the next few years will be those graduating from truly interdisciplinary programs.

However, there are many types of bioinformatics jobs available, so no one background is ideal for all of them. The fact is that many of the jobs available currently involve the design and implementation of programs and systems for the storage, management and analysis of vast amounts of DNA sequence data. Such positions require in-depth programming and relational database skills which very few biologists possess and so it is largely the computational specialists who are filling these roles.

This is not to say the computer-savvy biologist doesn't play an important role. As the bioinformatics field matures there will be a huge demand for outreach to the biological community as well as the need for individuals with the in-depth biological background necessary to sift through gigabases of genomic sequence in search of novel targets. It will be in these areas that biologists with the necessary computational skills will find their niche.

GVK BioSciences Pvt. Ltd	Hyderabad

The Scope

As geneticists, microbiologists and other researchers continue to gather huge amounts of new information about the human genome and biological molecules, there is a growing need for sophisticated, computerized approaches for compiling and analyzing that data. The process by which that is done is called bioinformatics. Every major university in the world is trying to get its share in this field.

There is a great scope for Bioinformatics in India. Companies have to work hard to gain respect and credibility. Bioinformatics hasn't and cannot create a million jobs like IT as it is only a subset of IT. The numbers will increase but in small percentages.

Strand Genomics	Bangalore
AstraZeneca	Bangalore
Dr Reddy's Laboratories	Hyderabad
Ingenovis (division of I labs)	Hyderabad
Jubilant Biosys (subsidiary from Jubilant Organosys)	Noida, UP
Landsky Solutions	Secunderabad
Molecular Connections	Bangalore
Ocimum Biosolutions	Hyderabad
PrayogNET Computing	Chennai
Questar Bioinformatics	Hyderabad
Satyam Computers	Hyderabad
Spectramind Services	New Delhi
Total Consultancy Services	Hyderabad

"I wouldn't advice everyone to jump into this field as it would only dilute the market with excess supply of professionals. On the other hand, it might be good for companies as it would give us enough people to choose from", said Ocimum's Anuradha.

Another observation was that for a Bioinformatics company which hires 100 people, about 70 percent are people with core knowledge with some understanding of bioinformatics. The number of people with bioinformatics resumes have increased rapidly but the quality of these "professionals" hasn't.

"Companies like us are always looking for good people but it takes us, on an average, 100 shortlisted resumes to finally pick one qualified person," she added

According to Rajendran, Sr. Executive - Business Development, BrainWave Bioinformatics Ltd, a lot of universities and institutes are into bioinformatics. Almost

every university in Andhra Pradesh and Karnataka offer a diploma in Bioinformatics. The prominent ones are the University of Hyderabad, Osmania University, IICT, IIIT.

Many private institutions which started during the hype have shut down. There are very good universities like the University of Pune, Madurai Kamaraj, Bose Institute and Jawaharlal Nehru University. The IIT's at Kharagpur and Delhi also have a very good biotechnology department.

A lot of IT companies like TCS and Infosys have ventured in to this area but most of them do not have very large teams. Many large companies and research institutions are hiring hundreds of bioinformatics professionals.

"Bioinformatics as a career is very lucrative and has a great future. Requirement from an individual is the ability to contribute either in life sciences or in IT when working in a team comprising of professionals from both fields. Typical qualifications would be Masters and Ph.D. The salaries are benchmarked against industry standards and would be comparable with any other industry including IT. The sector is growing at an impressive rate and companies which understand the 'real issues' of the industry will only survive in the long run. Working with such companies will result in overall development for professionals in this sector," says Sowmya Narayan of Strand Genomics.

According to Dr GPS Raghava, Scientist & Co-ordinator of Bioinformatics Centre, Institute of Microbial Technology, Chandigarh, there is a big gap between the demand and expertise available. The gap is not only in India but in the US also. Despite the hype and the presence of large number of bioinformatics training centers in India our contribution is too limited.

Other useful areas

Bioinformatics is today seen as primarily applied to speeding up new drug discovery. But the other area that assumes increasingly higher significance is the application of IT to the entire life sciences sector- for the same purpose it is done in other industrial sectors- improving efficiency, reducing costs, wider access, etc. For example bio-diversity data management is an area that requires application of the best database design techniques and planning for data warehousing and data-mining. Knowledge management as applied to corporations will also become relevant in the scientific context to ensure that Indian scientists get relevant and timely information related to their research to help them network and collaborate to create new intellectual property.

"There may be around 200-300 employed in this sector every year. There are a lot of private institutions getting into the foray, but then quality is indeterminate," said Rajiv Vasudevan, who is an expert both in IT and biotechnology.

Bioinformatics in India is at an early stage of development. But at 4 to 5 centers in the country, one sees mature understanding of the needs of this sector and world class development of tools and applications. These centers will ensure that India's traditional strengths in IT are leveraged to place us on par with the developed countries

Bioinformatics : Introduction

Bioinformatics or computational biology is the use of information technology in the field of molecular biology, or the application of computer technology to the management and analysis of biological data. Here computers are used to gather, store, analyze and merge biological data.

It is an emerging interdisciplinary research area and is increasingly being used to improve the quality of life. The ultimate goal of bioinformatics is to uncover the wealth of biological information hidden in the mass of sequence, structure, literature and other biological data and obtain a clearer insight into the fundamental biology of organisms and to use this information to enhance the standard of life for mankind.

It's importance is unlimited in the genomic era (genome is the complex molecular chains that constitute each organism's unique genetic heritage).
The hereditary information of an organism is encoded in the DNA. Genomics, the study of an organisms entire genome is developed to understand the basic molecule of life which is known as the code of life. DNA directly controls the biological makeup of humans or any living organism. Studying the structure of DNA has made amazing success over many genetically transferring diseases. It is variations and errors in the genomic DNA which ultimately define the likelihood of developing diseases or resistance to the same disorders.

Bioinformatics could have profound impact in fields as varied as human health, agriculture, the environment, energy and biotechnology to advance biomedical research and development. It is being used now in the areas of molecular medicine to help produce better and more customized medicines to prevent or cure diseases, it has environmental benefits in identifying waste cleanup bacteria and in agriculture it can be used for producing high yield low maintenance crops. These are just a few of the many benefits bioinformatics can offer.

Related Career Options

» Political Science

The career prospects in the field has been steadily increasing with more and more use of information technology in the field of molecular biology. Job prospects are in all sectors of biotechnology, pharmaceutical and biomedical sciences, in research institutions, hospital and industry. Some of the specific career areas that fall within the scope of bioinformatics include Sequence assembly, Database design and maintenance, Sequence analysis, Proteomics (the study of protein, particularly their structures and functions), Pharmacogenomics, Pharma-cology, Clinical pharmacologist, Informatics developer, Computational chemist, Bio-analytics and Analytics etc.

A3

Genomes of prokaryotes

Prokaryotic genomes are very different from eukaryotic ones. There is some overlap in size between the largest prokaryotic and smallest eukaryotic genomes, but on the whole prokaryotic genomes are much smaller. For example, the *E. coli* K12 genome is just 4639 kb, two-fifths the size of the yeast genome, and has only 4405 genes. The physical organization of the genome is also different in eukaryotes and prokaryotes. The traditional view has been that an entire prokaryotic genome is contained in a single circular DNA molecule. As well as this single 'chromosome', prokaryotes may also have additional genes on independent smaller, circular or linear DNA molecules called plasmids (Figure 2.3). Genes carried by plasmids are useful, coding for properties such as antibiotic resistance or the ability to utilize complex compounds such as toluene as a carbon source, but plasmids appear to be dispensable - a prokaryote can exist quite effectively without them. We now know that this traditional view of the prokaryotic genome has been biased by the extensive research on *E. coli*, which has been accompanied by the mistaken assumption that *E. coli* is a typical prokaryote. In fact, prokaryotes display a considerable diversity in genome organization, some having a unipartite genome, like *E. coli,* but others being more complex. *Borrelia burgdorferi* B31, for example, has a linear chromosome of 911 kb, carrying 853 genes, accompanied by 17 or 18 linear and circular molecules, which together contribute another 533 kb and at least 430 genes (Fraser *et al*., 1997). Multipartite genomes are now known in many other bacteria and archaea.

Figure 2.3

Plasmids are small circular DNA molecules that are found inside some prokaryotic cells.

In one respect, *E. coli* is fairly typical of other prokaryotes. After our discussion of eukaryotic gene organization, it will probably come as no surprise to learn that prokaryotic genomes are even more compact than those of yeast and other lower eukaryotes. We can see this fact illustrated in Figure 2.2E, which shows a 50-kb segment of the *E. coli* K12 genome. It is immediately obvious that there are more genes and less space between them, with 43 genes taking up 85.9% of the segment. Some genes have virtually no space between them: *thrA* and *thrB*, for example, are separated by a single nucleotide, and *thrC* begins at the nucleotide immediately following the last nucleotide of *thrB*. These three genes are an example of an operon, a group of genes involved in a single biochemical pathway (in this case, synthesis of the amino acid threonine) and expressed in conjunction with one another. Operonshave been used as model systems for understanding how gene expression is regulated (Section 9.3.1). In general, prokaryotic genes are shorter than their eukaryotic counterparts, the average length of a bacterial gene being about two-thirds that of a eukaryotic gene, even after the introns have been removed from the latter (Zhang, 2000). Bacterial genes appear to be slightly longer than archaeal ones.

Two other features of prokaryotic genomes can be deduced from Figure 2.2E. First, there are no introns in the genes present in this segment of the *E. coli* genome. In fact *E. coli* has no discontinuous genes at all, and it is generally believed that this type of gene structure is virtually absent in prokaryotes, the few exceptions occurring mainly among the archaea. The second feature is the infrequency of repetitive sequences. Most prokaryotic genomes do not have anything equivalent to the high-copy-number genome-wide repeat families found in eukaryotic genomes. They do, however, possess certain sequences that might be repeated elsewhere in the genome, examples being the insertion sequences IS1 and IS186 that can be seen in the 50-kb segment shown in Figure 2.2E. These are further examples of transposable elements, sequences that have the ability to move around the genome and, in the case of insertion elements, to transfer from one organism to another, even sometimes between two different species (see page 64). The positions of the IS1 and IS186 elements shown in Figure 2.2E refer only to the particular *E. coli* isolate from which this sequence was obtained: if a different isolate is examined then the IS sequences could well be in different positions or might be entirely absent from the genome. Most other prokaryotic genomes have very few repeat sequences - there are virtually none in the 1.64 Mb genome of*Campylobacter jejuni* NCTC11168 (Parkhill *et al.*, 2000b) - but there are exceptions, notably the meningitis bacterium *Neisseria meningitidis* Z2491, which has over 3700 copies of 15 different types of repeat sequence, collectively making up almost 11% of the 2.18 Mb genome (Parkhill *et al.*, 2000a).

E. coli: A Model Prokaryote

Much of what is known about prokaryotic chromosome structure was derived from studies of *Escherichia coli*, a bacterium that lives in the human colon and is commonly used in laboratory cloning experiments. In the 1950s and 1960s, this bacterium became the model organism of choice for prokaryotic research when a group of scientists used phase-contrast microscopy and autoradiography to show that the essential genes of *E. coli* are encoded on a single circular chromosome packaged within the cell nucleoid (Mason & Powelson, 1956; Cairns, 1963). Prokaryotic cells do not contain nuclei or other membrane-bound organelles. In fact, the word "prokaryote" literally means "before the nucleus." Thenucleoid is simply the area of a prokaryotic cell in which the chromosomal DNA is located. This arrangement is not as simple as it sounds, however, especially considering that the *E. coli* chromosome is several orders of magnitude larger than the cell itself. So, if bacterial chromosomes are so huge, how can they fit comfortably inside a cell—much less in one small corner of the cell?

DNA Supercoiling

The answer to this question lies in DNA packaging. Whereas eukaryotes wrap their DNA around proteins called histones to help package the DNA into smaller spaces, most prokaryotes do not have histones (with the exception of those species in the domain Archaea). Thus, one way prokaryotes compress their DNA into smaller spaces is through supercoiling (Figure 1). Imagine twisting a rubber band so that it forms tiny coils. Now twist it even further, so that the original coils fold over one another and form a condensed ball. When this type of twisting happens to a bacterial genome, it is known as supercoiling. Genomes can be negatively supercoiled, meaning that the DNA is twisted in the opposite direction of the double helix, or positively supercoiled, meaning that the DNA is twisted in the same direction as the double helix. Most bacterial genomes are negatively supercoiled during normal growth.

Proteins Involved in Supercoiling

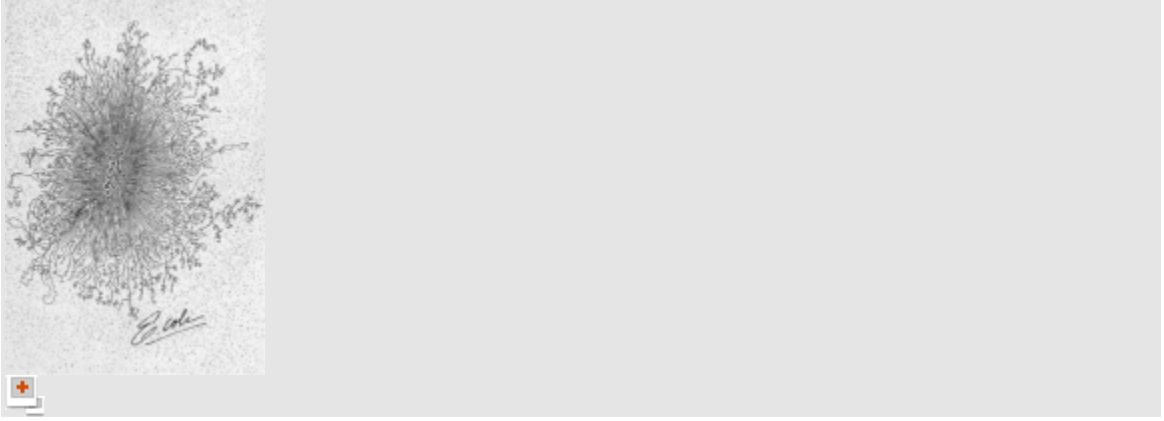


Figure 1

During the 1980s and 1990s, researchers discovered that multiple proteins act together to fold and condense prokaryoticDNA. In particular, one protein called HU, which is the most abundant protein in the nucleoid, works with an enzymecalled topoisomerase I to bind DNA and introduce sharp bends in the chromosome, generating the tension necessary for negative supercoiling. Recent studies have also shown that other proteins, including integration host factor (IHF), can bind to specific sequences within the genome and introduce additional bends (Rice *et al*., 1996). The folded DNA is then organized into a variety of conformations (Sinden & Pettijohn, 1981) that are supercoiled and wound around tetramers of the HU protein, much like eukaryotic chromosomes are wrapped around histones (Murphy & Zimmerman, 1997).

Once the prokaryotic genome has been condensed, DNA topoisomerase I, DNA gyrase, and other proteins help maintain the supercoils. One of these maintenance proteins, H-NS, plays an active role in transcription by modulating the expression of the genes involved in the response to environmental stimuli. Another maintenance protein, factor forinversion stimulation (FIS), is abundant during exponential growth and regulates the expression of more than 231 genes, including DNA topoisomerase I (Bradley *et al*., 2007).

Accessing Supercoiled Genes

Supercoiling explains how chromosomes fit into a small corner of the cell, but how do the proteins involved in replication and transcription access the thousands of genes in prokaryotic chromosomes when everything is packaged together so tightly? It has been determined that prokaryotic DNAreplication occurs at a rate of 1,000 nucleotides per second, and prokaryotic transcription occurs at a rate of about 40 nucleotides per second (Lewin, 2007), so bacteria must have highly efficient methods of accessing their DNA strands. But how?

Researchers have noted that the nucleoid usually appears as an irregularly shaped mass within the prokaryotic cell, but it becomes spherical when thecell is treated with chemicals to inhibit transcription or translation. Moreover, during transcription, small regions of the chromosome can be seen to project from the nucleoid into the cytoplasm (i.e., the interior of the cell), where they unwind and associate with ribosomes, thus allowing easy access by various transcriptional proteins (Dürrenberger *et al*., 1988). These projections are thought to explain the mysterious shape of nucleoids during active growth. When transcription is inhibited, however, the projections retreat into the nucleoid, forming the aforementioned spherical shape.

Because there is no nuclear membrane to separate prokaryotic DNA from the ribosomes within the cytoplasm, transcription and translation occur simultaneously in these organisms. This is strikingly different from eukaryotic chromosomes, which are confined to the membrane-bound nucleus during most of the cell cycle. In eukaryotes, transcription must be completed in the nucleus before the newly synthesized mRNA molecules can be transported to the cytoplasm to undergo translation into proteins.

Variations in Prokaryotic Genome Structure

Figure 2: Deer tick (Ixodes sp.) transmits the bacteria (Borrelia sp.) that causes Lyme disease.

Recently, it has become apparent that one size does not fit all when it comes to prokaryotic chromosome structure. While most prokaryotes, like *E. coli*, contain a single circular DNA molecule that makes up their entire genome, recent studies have indicated that some prokaryotes contain as many as fourlinear or circular chromosomes. For example, *Vibrio cholerae*, the bacteriathat causes cholera, contains two circular chromosomes. One of these chromosomes contains the genes involved in metabolism and virulence, while the other contains the remaining essential genes (Trucksis *et al.*, 1998). An even more extreme example is provided by *Borrelia burgdorferi*, the bacterium that causes Lyme disease. This organism is transmitted through the bite of deer ticks (Figure 2), and it contains up to 11 copies of a single linearchromosome (Ferdows & Barbour, 1989). Unlike *E. coli*, *Borrelia* cannot supercoil its linear chromosomes into a tight ball within the nucleoid; rather, these strands are diffused throughout the cell (Hinnebusch & Bendich, 1997).

Other organisms, such as *Bacillus subtilis*, form nucleoids that closely resemble those of *E. coli*, but they use different architectural proteins to do so. Furthermore, the DNA molecules of Archaea, a taxonomic domaincomposed of single-celled, nonbacterial prokaryotes that share many similarities with eukaryotes, can be negatively supercoiled, positively supercoiled, or not supercoiled at all. It is important to note that archaeans are the only group of prokaryotes that use eukaryote-like histones, rather than the architectural proteins described above, to condense their DNA molecules (Sandman *et al*., 1990). The acquisition of histones by archaeans is thought to have paved the way for the evolution of larger and more complex eukaryotic cells (Minsky *et al.*, 1997).

Other DNA Differences Between Prokaryotes and Eukaryotes

Most prokaryotes reproduce asexually and are haploid, meaning that only a single copy of each gene is present. This makes it relatively easy to generate mutations in the lab and study the resulting phenotypes. By contrast, eukaryotes that reproduce sexually generally contain multiple chromosomes and are said to be diploid, because two copies of each gene exist—with one copy coming from each of an organism's parents.

Yet another difference between prokaryotes and eukaryotes is that prokaryotic cells often contain one or more plasmids (i.e., extrachromosomal DNAmolecules that are either linear or circular). These pieces of DNA differ from chromosomes in that they are typically smaller and encode nonessential genes, such as those that aid growth in specific conditions or encode antibiotic resistance. *Borrelia*, for instance, contains more than 20 circular andlinear plasmids that encode genes responsible for infecting ticks and humans (Fraser *et al*., 1997). Plasmids are often much smaller than chromosomes (i.e., less than 1,500 kilobases), and they replicate independently of the rest of the genome. However, some plasmids are capable of integrating into chromosomes or moving from cell to cell.

Perhaps due to the space constraints of packing so many essential genes onto a single chromosome, prokaryotes can be highly efficient in terms of genomic organization. Very little space is left between prokaryotic genes. As a result, noncoding sequences account for an average of 12% of the prokaryotic genome, as opposed to upwards of 98% of the genetic material in eukaryotes (Ahnert *et al.*, 2008). Furthermore, unlike eukaryoticchromosomes, most prokaryotic genomes are organized into polycistronic operons, or clusters of more than one coding region attached to a singlepromoter, separated by only a few base pairs. The proteins encoded by each operon often collaborate on a single task, such as the metabolism of a sugar into by-products that can be used for energy (Figure 3).

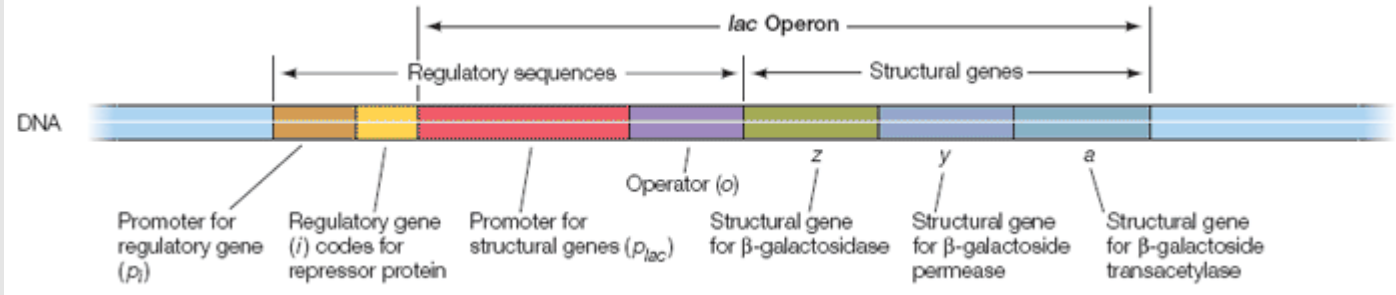


Figure 3: The lac operon of E. coli.
The lac operon of E. coli is a segment of DNA that includes a promoter, an operator, and the three structural genes that code for lactose-metabolizing enzymes.

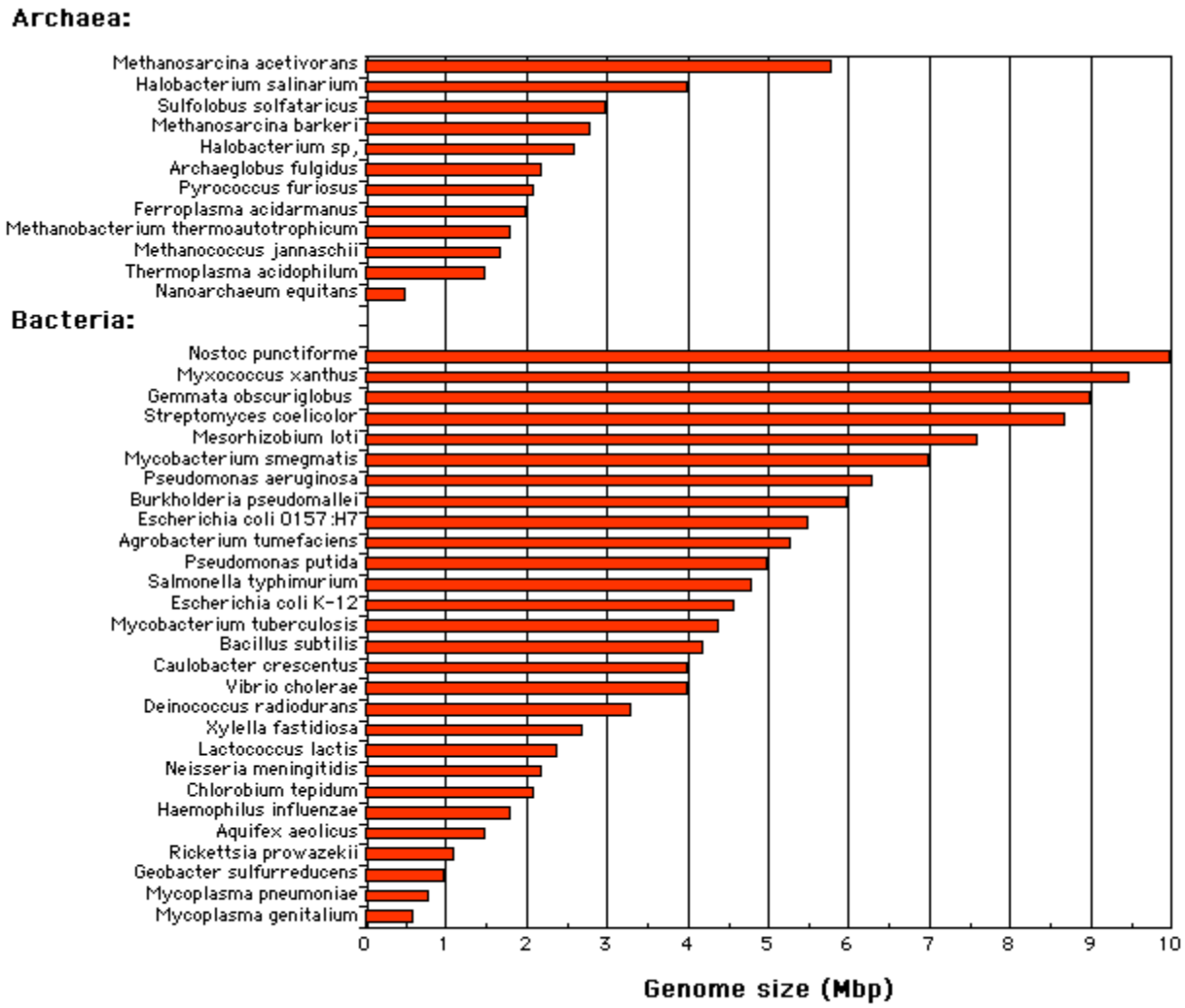
The organization of prokaryotic DNA therefore differs from that of eukaryotes in several important ways. The most notable difference is the condensation process that prokaryotic DNA molecules undergo in order to fit inside relatively small cells. Other differences, while not as dramatic, are summarized in Table 1.

Table 1: Prokaryotic versus** Eukaryotic **Chromosomes

Prokaryotic Chromosomes	Eukaryotic **Chromosomes**
Many prokaryotes contain a single circularchromosome.	Eukaryotes contain multiple linearchromosomes.
Prokaryotic chromosomes are condensed in the nucleoid via DNA supercoiling and the binding of various architectural proteins.	Eukaryotic chromosomes are condensed in a membrane-bound nucleus via histones.
Because prokaryotic DNA can interact with the cytoplasm, transcription and translationoccur simultaneously.	In eukaryotes, transcription occurs in thenucleus, and translation occurs in the cytoplasm.
Most prokaryotes contain only one copy of each gene (i.e., they are haploid).	Most eukaryotes contain two copies of eachgene (i.e., they are diploid).
Nonessential prokaryotic genes are commonly encoded on extrachromosomal plasmids.	Some eukaryotic genomes are organized into operons, but most are not.
Prokaryotic genomes are efficient and compact, containing little repetitive DNA.	Extrachromosomal plasmids are not commonly present in eukaryotes.
	Eukaryotes contain large amounts of noncoding and repetitive DNA.

How BIG are Prokaryotic Genomes?

Not long ago it was thought that all prokaryotic genomes (both Bacteria and Archae) were much smaller than eukaryotic genomes. However, the application of new techniques for constructing physical maps and whole genome sequencing has demonstrated that there is tremendous diversity in the size and organization of prokaryotic genomes. The following figure shows some examples of genome sizes of Bacteria and Archae. The size of Bacterial chromosomes ranges from 0.6 Mbp to over 10 Mbp, and the size of Archael chromosomes range from 0.5 Mbp to 5.8 Mbp. (For comparison, Eukaryotic chromosomes range from 2.9 Mbp (Microsporidia) to well over 4,000 Mbp, although the largest genomes are littered with a tremendous amount of repetitive "junk" DNA.)



The smallest Archae genome identified thus far is from Nanoarchaeum equtans, a tiny obligate symbiont with a genome size of 0.491 Mbp (491 Kbp). This organism lacks genes required for synthesis of lipids, amino acids, nucleotides, and vitamins, and hence must grow in close association with another organism which provides these nutrients.

The smallest Bacterial genome identified thus far is from Mycoplasma genitalium, an obligate intracellular pathogen with a genome size of 0.58 Mbp (580 Kbp). M. genitalium is restricted to the intracellular niche because it lacks genes encoding enzymes required for amino acid biosynthesis and the peptidoglycan cell wall, genes encoding TCA cycle enzymes, and many other biosynthetic genes.

In contrast to such obligate intracellular bacteria, free-living bacteria must dedicate many genes toward the biosynthesis and transport of nutrients and

building blocks. The smallest free-living organisms have a genome size over 1 Mbp.

A feature often used to distinguish prokaryotic and eukaryotic genomes is the amount on "junk" DNA (noncoding DNA that is probably largely remnants of genes that have been lost over the course of evolution). As a general rule, prokaryotes tend to have very little junk DNA (typically less than 15% of the genome) and eukaryotes have substantial amounts of junk DNA. However, genome sequences of certian bacteria with very limited ecological niches (e.g., the human pathogen Mycobacterium leprae and the Aphid endosymbiont Buchnera) indicates that, like in eukaryotes, a substantial amount of bacterial genomes can consist of junk DNA

A4

History of Bioinformatics

<-Prior Index Next->

The Modern bioinformatics is can be classified into two broad categories, Biological Science and computational Science. Here is the data of historical events for both biology and computer science.

Introduction:

The history of biology in general, B.C. and before the discovery of genetic inheritance by G. Mendel in 1865, is extremely sketch and inaccurate. This was the start of Bioinformatics history. Gregor Mendel. is known as the "Father of Genetics". He did experiment on the cross-fertilization of different colors of the same species. He carefully recorded the data and analyzed the data. Mendel illustrated that the inheritance of traits could be more easily explained if it was controlled by factors passed down from generation to generation.

The understanding of genetics has advanced remarkably in the last thirty years. In 1972, Paul berg made the first recombinant DNA molecule using ligase. In that same year, Stanley Cohen, Annie Chang and Herbert Boyer produced the first recombinant DNA organism. In 1973, two important things happened in the field of genomics. The advancement of computing in 1960-70s resulted in the basic methodology of bioinformatics. However, it is the 1990s when the INTERNET arrived when the full fledged bioinformatics field was born.

Here are some of the major events in bioinformatics over the last several decades. The events listed in the list occurred long before the term, "bioinformatics", was coined.

BioInformatics Events

1665	Robert Hooke published Micrographia, described the cellular structure of cork. He also described microscopic examinations of fossilized plants and animals, comparing their microscopic structure to that of the living organisms they resembled. He argued for an organic origin of fossils, and suggested a plausible mechanism for their formation.
1683	Antoni van Leeuwenhoek discovered bacteria.
1686	John Ray, John Ray's in his book "Historia Plantarum" catalogued and described 18,600 kinds of plants. His book gave the first definition of species based upon common descent.
1843	Richard Owen elaborated the distinction of **homology** and **analogy.**
1864	Ernst Haeckel (Häckel) outlined the essential elements of modern zoological classification.
1865	Gregory Mendel (1823-1884), Austria, established the theory of genetic inheritance.
1902	The chromosome theory of heredity is proposed by Sutton and Boveri, working independently.
1962	Pauling's theory of molecular evolution
1905	The word "genetics" is coined by William Bateson.
1913	First ever linkage map created by Columbia undergraduate Alfred Sturtevant (working with T.H. Morgan).
1930	Tiselius, Uppsala University, Sweden, A new technique, electrophoresis, is introduced by Tiselius for separating proteins in solution. "The moving-boundary method of studying the electrophoresis of proteins" (published in *Nova Acta Regiae Societatis Scientiarum Upsaliensis*, Ser. IV, Vol. 7, No. 4)
1946	Genetic material can be transferred laterally between bacterial cells, as shown by Lederberg and Tatum.
1952	Alfred Day Hershey and Martha Chase proved that the DNA alone carries genetic information. This was proved on the basis of their bacteriophage research.
1961	Sidney Brenner, François Jacob, Matthew Meselson, identify messenger RNA,
1965	Margaret Dayhoff's Atlas of Protein Sequences
1970	Needleman-Wunsch algorithm
1977	DNA sequencing and software to analyze it (Staden)
1981	Smith-Waterman algorithm developed
1981	The concept of a sequence motif (Doolittle)
1982	GenBank Release 3 made public
1982	Phage lambda genome sequenced
1983	Sequence database searching algorithm (Wilbur-Lipman)
1985	FASTP/FASTN: fast sequence similarity searching
1988	National Center for Biotechnology Information (NCBI) created at NIH/NLM
1988	EMBnet network for database distribution
1990	BLAST: fast sequence similarity searching

1991	EST: expressed sequence tag sequencing
1993	Sanger Centre, Hinxton, UK
1994	EMBL European Bioinformatics Institute, Hinxton, UK
1995	First bacterial genomes completely sequenced
1996	Yeast genome completely sequenced
1997	PSI-BLAST
1998	Worm (multicellular) genome completely sequenced
1999	Fly genome completely sequenced
2000	Jeong H, Tombor B, Albert R, Oltvai ZN, Barabasi AL. **The large-scale organization of metabolic networks.** Nature 2000 Oct 5;407(6804):651-4, PubMed
2000	The genome for *Pseudomonas aeruginosa* (6.3 Mbp) is published.
2000	The A. thaliana genome (100 Mb) is secquenced.
2001	The human genome (3 Giga base pairs) is published.

ort History of Bioinformatics

Allen B. Richon

Network Science

E-mail: TheEditors@netsci.org

http://www.netsci.org/Science/Bioinform/feature06.html

One of the fundamental principles of biology is that within each cell, the DNA that comprises the genes encodes RNA which in turn produces the proteins that regulate all of the biological processes within the organism. The human body is made up of an estimated 10^{12} cells, each of which contains 23 pairs of chromosomes that are comprised of approximately 30,000 genes which in turn contain some 3 billion pairs of DNA bases. While we have a basic understanding of how gene sequences code specific proteins, we lack the information necessary to completely understand role of DNA in specific diseases or the functions of the thousands of proteins that are produced. The methods that are used to collect, store, retrieve, analyze, and correlate this mountain of complex information are grouped into a discipline called bioinformatics. The goal of bioinformatics thus is to provide scientists with a means to explain:

- Normal biological processes
- Malfunctions in these processes which lead to diseases
- Approaches to improving drug discovery.

The use of these techniques has grown explosively in the past five years and shows no sign of slowing down. The result of this growth is that the number of sources of products, services, and information has increased to the point that keeping track of (or locating) the numerous providers has become extremely time consuming.

The intent of this article, and others that will appear in the near future, is to compile as comprehensive a list as possible of resources used by bioinformatics scientists. While the author has endeavored to provide a complete listing for this field, it is impossible to obtain profiles for all of the companies involved in this dynamic field.

Bioinformatics encompases the use of tools and techniques from three separate disciplines; molecular biology (the source of the data to be analyzed), computer science (supplies the hardware for running analysis and the networks to communicate the results), and the data analysis algorithms which strictly define bioinformatics. For this

reson, the editors have decided to incorporate events from these areas into a brief history of the field.

1933

A new technique, electrophoresis, is introduced by Tiselius for separating proteins in solution.

1951

Pauling and Corey propose the structure for the alpha-helix and beta-sheet (*Proc. Natl. Acad. Sci. USA*, **27**: 205-211, 1951; *Proc. Natl. Acad. Sci. USA*, **37**: 729-740, 1951).

1953

Watson and Crick propose the double helix model for DNA based on x-ray data obtained by Franklin and Wilkins (*Nature*, **171**: 737-738, 1953).

1954

Perutz's group develop heavy atom methods to solve the phase problem in protein crystallography.

1955

The sequence of the first protein to be analyzed, bovine insulin, is announced by F. Sanger.

1958

The first integrated circuit is constructed by Jack Kilby at Texas Instruments.

The Advanced Research Projects Agency (ARPA) is formed in the US

1968

Packet-switching network protocols are presented to ARPA

1969

The ARPANET is created by linking computers at Stanford, UCSB, The University of Utah and UCLA.

1970

The details of the Needleman-Wunsch algorithm for sequence comparison are published.

The maiden voyage of the Boeing 747 is made on January 12.

1971

Ray Tomlinson (BBN) invents the email program.

1972

The first recombinant DNA molecule is created by Paul Berg and his group.

1973

The Brookhaven Protein Data Bank is announced (*Acta. Cryst.* **B**, **1973**, 29: 1746).

Robert Metcalfe receives his Ph.D. from Harvard University. His thesis describes Ethernet.

1974

Vint Cerf and Robert Kahn develop the concept of connecting networks of computers into an "internet" and develop the Transmission Control Protocol (TCP).

Charles Goldfarb invents SGML (Standardized General Markup Language).

1975

Microsoft Corporation is founded by Bill Gates and Paul Allen.

Two-dimensional electrophoresis, where separation of proteins on SDS polyacrylamide gel is combined with separation according to isoelectric points, is announced by P. H. O'Farrell (*J. Biol. Chem.*, **250**: 4007-4021, 1975).
E. M. Southern published the experimental details for the Southern Blot technique of specific sequences of DNA (*J. Mol. Biol.*, **98**: 503-517, 1975).

1976

The Unix-To-Unix Copy Protocol (UUCP) is developed at Bell Labs.

1977

The full description of the Brookhaven PDB (http://www.pdb.bnl.gov) is published (Bernstein, F.C.; Koetzle, T.F.; Williams, G.J.B.; Meyer, E.F.; Brice, M.D.; Rodgers, J.R.; Kennard, O.; Shimanouchi, T.; Tasumi, M.J.; *J. Mol. Biol.*, **1977**, *112*:, 535).

Allan Maxam and Walter Gilbert (Harvard) and Frederick Sanger (U.K. Medical Research Council), report methods for sequencing DNA.

1978

The first Usenet connection is established between Duke and the University of North Carolina at Chapel Hill by Tom Truscott, Jim Ellis and Steve Bellovin.

1980

The first complete gene sequence for an organism (FX174) is published. The gene consists of 5,386 base pairs which code nine proteins.

Wüthrich et. al. publish paper detailing the use of multi-dimensional NMR for protein structure determination (Kumar, A.; Ernst, R.R.; Wüthrich, K.; *Biochem. Biophys. Res. Comm.*, **1980**, *95*:, 1).

IntelliGenetics, Inc. founded in California. Their primary product is the IntelliGenetics Suite of programs for DNA and protein sequence analysis.

1981

The Smith-Waterman algorithm for sequence alignment is published.

IBM introduces its Personal Computer to the market.

1982

Genetics Computer Group (GCG) created as a part of the University of Wisconsin of Wisconsin Biotechnology Center. The company's primary product is The Wisconsin Suite of molecular biology tools.

1983

The Compact Disk (CD) is launched.

Name servers are developed at the University of Wisconsin.

1984

Jon Postel's Domain Name System (DNS) is placed on-line.

The Macintosh is announced by Apple Computer.

1985

The FASTP algorithm is published.

The PCR reaction is described by Kary Mullis and co-workers.

1986

The term "Genomics" appeared for the first time to describe the scientific discipline of mapping, sequencing, and analyzing genes. The term was coined by Thomas Roderick as a name for the new journal.

Amoco Technology Corporation acquires IntelliGenetics.

NSFnet debuts.

The SWISS-PROT database is created by the Department of Medical Biochemistry of the University of Geneva and the European Molecular Biology Laboratory (EMBL).

1987

The use of yeast artifical chromosomes (YAC) is described (David T. Burke, et. al., *Science*, **236**: 806-812).
The physical map of *e. coli* is published (Y. Kohara, et. al., *Cell* **51:** 319-337).

Perl (Practical Extraction Report Language) is released by Larry Wall.

1988

The National Center for Biotechnology Information (NCBI) is established at the National Cancer Institute.

The Human Genome Initiative is started (Commission on Life Sciences, National Research Council. *Mapping and Sequencing the Human Genome*, National Academy Press: Washington, D.C.), 1988.

The FASTA algorithm for sequence comparison is published by Pearson and Lupman.

Des Higgins and Paul Sharpe announce the development of CLUSTAL (Higgins, D.G.; Sharp, P.M. Fast and sensitive multiple sequence alignments on a microcomputer. *Comput. Appl. Biosci.* **1989**, *5*, 151-153; Higgins, D.G.; Sharp, P.M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene***1988**, *73*, 237-244.)

A new program, an Internet computer virus designed by a student, infects 6,000 military computers in the US.

1989

The Genetics Computer Group (GCG) becomes a private company.

Oxford Molecular Group, Ltd. (OMG) founded in Oxford, UK by Anthony Marchington, David Ricketts, James Hiddleston, Anthony Rees, and W. Graham Richards. Primary products: Anaconda, Asp, Cameleon and others (molecular modeling, drug design, protein design).

1990

The BLAST program (Altschul, *et. al.*) is implemented.

Molecular Applications Group is founded in California by Michael Levitt and Chris Lee. Their primary products are Look and SegMod which are used for molecular modeling and protein design.

InforMax is founded in Bethesda, MD. The company's products address sequence analysis, database and data management, searching, publication graphics, clone construction, mapping and primer design.

The HTTP 1.0 specification is published. Tim Berners-Lee publishes the first HTML document.

1991

The research institute in Geneva (CERN) announces the creation of the protocols which make-up the World Wide Web.

Linus Torvalds announces a Unix-Like operating system which later becomes Linux.

The creation and use of expressed sequence tags (ESTs) is described (J. Craig Venter, *et. al.*, *Science*, **252**: 1651-1656).

Incyte Pharmaceuticals, a genomics company headquartered in Palo Alto California, is formed.

Myriad Genetics, Inc. is founded in Utah. The company's goal is to lead in the discovery of major common human disease genes and their related pathways. The Company has discovered and sequenced, with its academic collaborators, the following major genes: BRCA1, BRCA2, CHD1, MMAC1, MMSC1, MMSC2, CtIP, p16, p19, and MTS2.

1992

Human Genome Systems, Gaithersburg Maryland, is formed by William Haseltine.

The Institute for Genomic Research (TIGR) is established by Craig Venter.

Genome Therapeutics announces its incorporation.

Mel Simon and coworkers announce the use of BACs for cloning.

1993

CuraGen Corporation is formed in New Haven, CT.

Affymetrix begins independent operations in Santa Clara, California

Compugen begins operations in Israel.

InterNIC is created by the National Science Foundation.

1994

Netscape Comminications Corporation founded and releases Navigator, the commercial version of NCSA's Mozilla.

Gene Logic is formed in Maryland.

The PRINTS database of protein motifs is published by Attwood and Beck.

Oxford Molecular Group acquires IntelliGenetics.

1995

Microsoft releases version 1.0 of Internet Explorer.

Sun releases version 1.0 of Java. Sun and Netscape release version 1.0 of JavaScript

Version 1.0 of Apache is released.

The *Haemophilus influenzea* genome (1.8 Mb) is sequenced.
The *Mycoplasma genitalium* genome is sequenced.

1996

The working draft for XML is released by W3C.

Oxford Molecular Group acquires the MacVector product from Eastman Kodak.

The genome for *Saccharomyces cerevisiae* (baker's yeast, 12.1 Mb) is sequenced.
The Prosite database is reported by Bairoch, *et.al.*

Affymetrix produces the first commercial DNA chips.

Structural Bioinformatics, Inc. founded in San Diego, CA.

1997

The genome for *E. coli* (4.7 Mbp) is published.

Oxford Molecular Group acquires the Genetics Computer Group.

LION bioscience AG founded as an integrated genomics company with strong focus on bioinformatics. The company is built from IP out of the European Molecular Biology Laboratory (EMBL), the European Bioinformatics Institute (EBI), the German Cancer Research Center (DKFZ), and the University of Heidelberg.

Paradigm Genetics Inc., a company focussed on the application of genomic technologies to enhance worldwide food and fiber production, is founded in Research Triangle Park, NC.

deCode genetics publishes a paper that described the location of the FET1 gene, which is responsible for familial essential tremor, on chromosome 13 (Nature Genetics).

1998

The genomes for *Caenorhabditis elegans* and baker's yeast are published.

The Swiss Institute of Bioinformatics is established as a non-profit foundation.

Craig Venter forms Celera in Rockville, Maryland.

PE Informatics was formed as a Center of Excellence within PE Biosystems. This center brings together and leverages the complementary expertise of PE Nelson and Molecular Informatics, to further complement the genetic instrumentation expertise of Applied Biosystems.

Inpharmatica, a new Genomics and Bioinformatics company, is established by University College London, the Wolfson Institute for Biomedical Research, five leading scientists from major British academic centers and Unibio Limited.

GeneFormatics, a company dedicated to the analysis and prediction of protein structure and function, is formed in San Diego.

Molecular Simulations Inc. is acquired by Pharmacopeia

1999

deCode genetics maps the gene linked to pre-eclampsia as a locus on chromosome 2p13.

2000

The genome for *Pseudomonas aeruginosa* (6.3 Mbp) is published.

The A. thaliana genome (100 Mb) is secquenced.

The D. melanogaster genome (180Mb) is secquenced.

Pharmacopeia acquires Oxford Molecular Group.

2001

The human genome (3,000 Mbp) is published.

2002

Structural Bioinformatics and GeneFormatics merge

An international sequencing consortium published the full genome sequence of the common house mouse (2.5 Gb). Whitehead Institute researcher Kerstin Lindblad-Toh is the lead author on the paper; her institution lead the project and contributed about half of the sequence. Washington University School of Medicine delivered about 30 percent of the sequence, and created the mouse BAC-based physical map. The Wellcome Trust Sanger Institute in the UK was the third major partner. Other institutes in the International Mouse Genome Sequencing Consortium included the University of California at Santa Cruz, the Institute for Systems Biology, and the University of Geneva.

2004

The draft genome sequence of the brown Norway laboratory rat, Rattus norvegicus, was completed by the Rat Genome Sequencing project Consortium. The paper appears in the April 1 edition of Nature.

A5
Refer to 2 pdfs

A6
A

What is pharmacogenomics?

Pharmacogenomics is the study of how an individual's genetic inheritance affects the body's response to drugs. The term comes from the words pharmacology and genomics and is thus the intersection of pharmaceuticals and genetics.

Pharmacogenomics holds the promise that drugs might one day be tailor-made for individuals and adapted to each person's own genetic makeup. Environment, diet, age, lifestyle, and state of health all can influence a person's response to medicines, but understanding an individual's genetic makeup is thought to be the key to creating personalized drugs with greater efficacy and safety.

Pharmacogenomics combines traditional pharmaceutical sciences such as biochemistry with annotated knowledge of genes, proteins, and single nucleotide polymorphisms.

What are the anticipated benefits of pharmacogenomics?

- **More Powerful Medicines**
  Pharmaceutical companies will be able to create drugs based on the proteins, enzymes, and RNA molecules associated with genes and diseases. This will facilitate drug discovery and allow drug makers to produce a therapy more targeted to specific diseases. This accuracy not only will maximize therapeutic effects but also decrease damage to nearby healthy cells.
- **Better, Safer Drugs the First Time**
  Instead of the standard trial-and-error method of matching patients with the right drugs, doctors will be able to analyze a patient's genetic profile and prescribe the best available drug therapy from the beginning. Not only will this take the guesswork out of finding the right drug, it will speed recovery time and increase safety as the likelihood of adverse reactions is eliminated. Pharmacogenomics has the potential to dramatically reduce the the estimated 100,000 deaths and 2 million hospitalizations that occur each year in the United States as the result of adverse drug response (1).
- **More Accurate Methods of Determining Appropriate Drug Dosages**
  Current methods of basing dosages on weight and age will be replaced with dosages based on a person's genetics --how well the body processes the medicine and the time it takes to metabolize it. This will maximize the therapy's value and decrease the likelihood of overdose.
- **Advanced Screening for Disease**
  Knowing one's genetic code will allow a person to make adequate lifestyle and environmental changes at an early age so as to avoid or lessen the severity of a genetic disease. Likewise, advance knowledge of a particular disease susceptibility will allow careful monitoring, and treatments can be introduced at the most appropriate stage to maximize their therapy.
- **Better Vaccines**
  Vaccines made of genetic material, either DNA or RNA, promise all the benefits of existing vaccines without all the risks. They will activate the immune system but will be unable to cause infections. They will be inexpensive, stable, easy to store, and capable of being engineered to carry several strains of a pathogen at once.
- **Improvements in the Drug Discovery and Approval Process**
  Pharmaceutical companies will be able to discover potential therapies more easily using genome targets. Previously failed drug candidates may

be revived as they are matched with the niche population they serve. The drug approval process should be facilitated as trials are targeted for specific genetic population groups --providing greater degrees of success. The cost and risk of clinical trials will be reduced by targeting only those persons capable of responding to a drug.

- **Decrease in the Overall Cost of Health Care**
  Decreases in the number of adverse drug reactions, the number of failed drug trials, the time it takes to get a drug approved, the length of time patients are on medication, the number of medications patients must take to find an effective therapy, the effects of a disease on the body (through early detection), and an increase in the range of possible drug targets will promote a net decrease in the cost of health care.

Is pharmacogenomics in use today?

To a limited degree. The cytochrome P450 (CYP) family of liver enzymes is responsible for breaking down more than 30 different classes of drugs. DNA variations in genes that code for these enzymes can influence their ability to metabolize certain drugs. Less active or inactive forms of CYP enzymes that are unable to break down and efficiently eliminate drugs from the body can cause drug overdose in patients. Today, clinical trials researchers use genetic tests for variations in cytochrome P450 genes to screen and monitor patients. In addition, many pharmaceutical companies screen their chemical compounds to see how well they are broken down by variant forms of CYP enzymes (2).

Another enzyme called TPMT (thiopurine methyltransferase) plays an important role in the chemotherapy treatment of a common childhood leukemia by breaking down a class of therapeutic compounds called thiopurines. A small percentage of Caucasians have genetic variants that prevent them from producing an active form of this protein. As a result, thiopurines elevate to toxic levels in the patient because the inactive form of TMPT is unable to break down the drug. Today, doctors can use a genetic test to screen patients for this deficiency, and the TMPT activity is monitored to determine appropriate thiopurine dosage levels (3).

References

1. J. Lazarou, B. H. Pomeranz, and P. N. Corey. Incidence of adverse drug reactions in hospitalized patients: a meta-analysis of prospective studies. *JAMA.* Apr 15, 1998. **279**(15):1200-5.
2. J. Hodgson, and A. Marshall. Pharmacogenomics: will the regulators approve? *Nature Biotechnolgy*. **16**: 243-246. 1998
3. S. Pistoi. Facing your genetic destiny, part II. *Scientific American*. February 25, 2002.

What are some of the barriers to pharmacogenomics progress?

Pharmacogenomics is a developing research field that is still in its infancy. Several of the following barriers will have to be overcome before many pharmacogenomics benefits can be realized.

- **Complexity of finding gene variations that affect drug response** - Single nucleotide polymorphisms (SNPs) are DNA sequence variations that occur when a single nucleotide (A,T,C,or G) in the genome sequence is altered. SNPs occur every 100 to 300 bases along the 3-billion-base human genome, therefore millions of SNPs must be identified and analyzed to determine their involvement (if any) in drug response. Further complicating the process is our limited knowledge of which genes are involved with each drug response. Since many genes are likely to influence responses, obtaining the big picture on the impact of gene variations is highly time-consuming and complicated.

- **Limited drug alternatives** - Only one or two approved drugs may be available for treatment of a particular condition. If patients have gene variations that prevent them using these drugs, they may be left without any alternatives for treatment.

- **Disincentives for drug companies to make multiple pharmacogenomic products** - Most pharmaceutical companies have been successful with their "one size fits all" approach to drug development. Since it costs hundreds of millions of dollars to bring a drug to market, will these companies be willing to develop alternative drugs that serve only a small portion of the population?

- **Educating healthcare providers** - Introducing multiple pharmacogenomic products to treat the same condition for different population subsets undoubtedly will complicate the process of prescribing and dispensing drugs. Physicians must execute an extra diagnostic step to determine which drug is best suited to each patient. To interpret the diagnostic accurately and recommend the best course of treatment for each patient, all prescribing physicians, regardless of specialty, will need a better understanding of genetics.

Pharmacogenomics

From Wikipedia, the free encyclopedia

Pharmacogenomics is the branch of pharmacology which deals with the influence of genetic variation on drug response in patients by correlating gene expression or single-nucleotide polymorphisms with a drug's efficacy or toxicity. By doing so, pharmacogenomics aims to develop rational means to optimise drug therapy, with respect to the patients' genotype, to ensure maximum efficacy with minimal adverse effects. Such approaches promise the advent of "personalized medicine"; in which drugs and drug combinations are optimized for each individual's unique genetic makeup.[1]

Pharmacogenomics is the whole genome application of pharmacogenetics, which examines the single gene interactions with drugs.

Pharmacogenomics is being used for all critical illnesses like cancer, cardio vascular disorders, HIV, tuberculosis, asthma, and diabetes.

In cancer treatment, pharmacogenomics tests are used to identify which patient will have toxicity from commonly used cancer drugs and identify which patient will not respond to commonly used cancer drug. Over the last couple of years, pharmacogenomics is also known as companion diagnostics, meaning tests being bundled with drugs. Two good examples are K-ras test with cituximab and EGFR test with Gefitinib.

B

Cheminformatics

From Wikipedia, the free encyclopedia

Cheminformatics (also known as **chemoinformatics** and **chemical informatics**) is the use of computer and informational techniques, applied to a range of problems in the field ofchemistry. These *in silico* techniques are used in pharmaceutical companies in the process of drug discovery. These methods can also be used in chemical and allied industries in various other forms.

History

The term chemoinformatics was defined by F.K. Brown [1][2] in 1998:

Chemoinformatics is the mixing of those information resources to transform data into information and information into knowledge for the intended purpose of making better decisions faster in the area of drug lead identification and optimization.

Since then, both spellings have been used, and some have evolved to be established as Cheminformatics,[3] while European Academia settled in 2006 for Chemoinformatics.[4] The recent

establishment of the Journal of Cheminformatics is a strong push towards the shorter variant.

[edit]Basics

Cheminformatics combines the scientific working fields of chemistry and computer science for example in the area of topology and chemical graph theory and mining the chemical space.[5][6] Cheminformatics can also be applied to data analysis for various industries like paper and pulp, dyes and such allied industries.

[edit]Applications

[edit]Storage and retrieval

Main article: Chemical database

The primary application of cheminformatics is in the storage of information relating to compounds. The efficient search of such stored information includes topics that are dealt with in computer science as data mining and machine learning. Related research topics include:

- Unstructured data
- Structured Data Mining and mining of Structured data
  - Database mining
  - Graph mining
  - Molecule mining
  - Sequence mining
  - Tree mining

[edit]File formats

Main article: Chemical file format

The *in silico* representation of chemical structures uses specialized formats such as the XML-based Chemical Markup Language or SMILES. These representations are often used for storage in large chemical databases. While some formats are suited for visual representations in 2 or 3 dimensions, others are

more suited for studying physical interactions, modeling and docking studies.

[edit]**Virtual libraries**

Chemical data can pertain to real or virtual molecules. Virtual libraries of compounds may be generated in various ways to explore chemical space and hypothesize novel compounds with desired properties.

Virtual libraries of classes of compounds (drugs, natural products, diversity-oriented synthetic products) were recently generated using the FOG (fragment optimized growth) algorithm. [7]This was done by using cheminformatic tools to train transition probabilities of a Markov chain on authentic classes of compounds, and then using the Markov chain to generate novel compounds that were similar to the training database.

[edit]**Virtual screening**

Main article: Virtual screening

In contrast to high-throughput screening, virtual screening involves computationally screening *in silico* libraries of compounds, by means of various methods such as docking, to identify members likely to possess desired properties such as biological activity against a given target. In some cases, combinatorial chemistry is used in the development of the library to increase the efficiency in mining the chemical space. More commonly, a diverse library of small molecules or natural products is screened.

[edit]**Quantitative structure-activity relationship (QSAR)**

Main article: Quantitative structure-activity relationship

This is the calculation of quantitative structure-activity relationship and quantitative structure property relationship values, used to predict the activity of compounds from their structures. In this context there is also a strong relationship to Chemometrics. Chemical expert systems are also relevant, since they represent parts of chemical knowledge as an *in silico*representation.

[**edit**]

A7

Integrative bioinformatics

From Wikipedia, the free encyclopedia

This article **is an orphan, as few or no other articles link to it**.
Please introduce links to this page from related articles;suggestions may be available. *(July 2010)*

Integrative bioinformatics is a discipline of bioinformatics that focuses on problems of data integration for the life sciences.

With the rise of high-throughput (HTP) technologies in the life sciences, particularly in molecular biology, the amount of collected data has grown in an exponential fashion. Furthermore, the data is scattered over a plethora of both public and private repositories, and is stored using a large number of different formats. This situation makes the extraction of new knowledge from the complete set of available data very difficult.

Integrative bioinformatics attempts to tackle this problem by providing unified access to life science data.

Contents

[hide]

[edit]Approaches

[edit]Semantic web approaches

[edit]Data warehousing approaches

In the data warehousing strategy, the data from different sources are extracted and integrated in a single database. For example, various 'omics' datasets may be integrated to provide biological insights into biological systems. Examples include data from genomics, transcriptomics, proteomics, interactomics, metabolomics. Ideally, changes in these sources are regularly synchronized to the

integrated database. The data is presented to the users in a common format. One advantage of this approach is that data is available for analysis at a single site, using a uniform schema. Some disadvantages are that the datasets are often huge and are difficult to keep up to date.

Refer to pdf

A8

10 Useful Bioinformatics Skills to Have

Looking for useful bioinformatics skills to acquire, here are ten that might look good on you resume.

Bioinformatics is all about the way biological information is acquired, managed, structured and communicated. It is data management to the nth degree, combined with communications technology and scientific knowledge, with a dash of computer science thrown in.

Bioinformatics is a highly specialized, technical field. You need a background in both biology and in information management, as well as specialized skills specific to the field of bioinformatics. Here are ten of those skills that it's good to have.

1. **Good communications skills.** As a bioinformatics specialist, you will be communicating complex data to people with a variety of backgrounds. It's very much like being a translator. You have to know the languages involved, and you have to be an expert communicator in order to help people understand each other.

2. **Good teamwork skills.** Bioinformatics is not for lone rangers. Researchers can sometimes work independently, but bioinformatics is about information and communication. You will be working on a team with people who have diverse backgrounds and differing areas of expertise. Good teamwork skills are essential.

3. **The ability to multitask.** You will need to be able to handle several complex tasks at a time. This can be a high pressure job with deadlines that have to be met. The ability to multitask will help you manage your job with less stress.

4. **Flexibility.** You may be moved from one project to another as your skills are needed. You may have to put aside a project you are working on to help someone with an urgent request. You may need to stop what you are doing and explain a computer model to a scientist. Flexibility is a key skill to have in bioinformatics.

5. **A working knowledge of biology and its applications.** You don't have to be an expert in biology, but you do need to know what kind of information you are working with. It is especially useful to know about molecular biology and genetics and to understand recent genetic research.

6. **Proficiency in computer languages.** You need to know basic programming languages like JAVA and HTML, SQL and PERL. Most bioinformatics programming utilizes PERL.

7. **Skill in data mining.** Being able to extract data from multiple resources is invaluable.

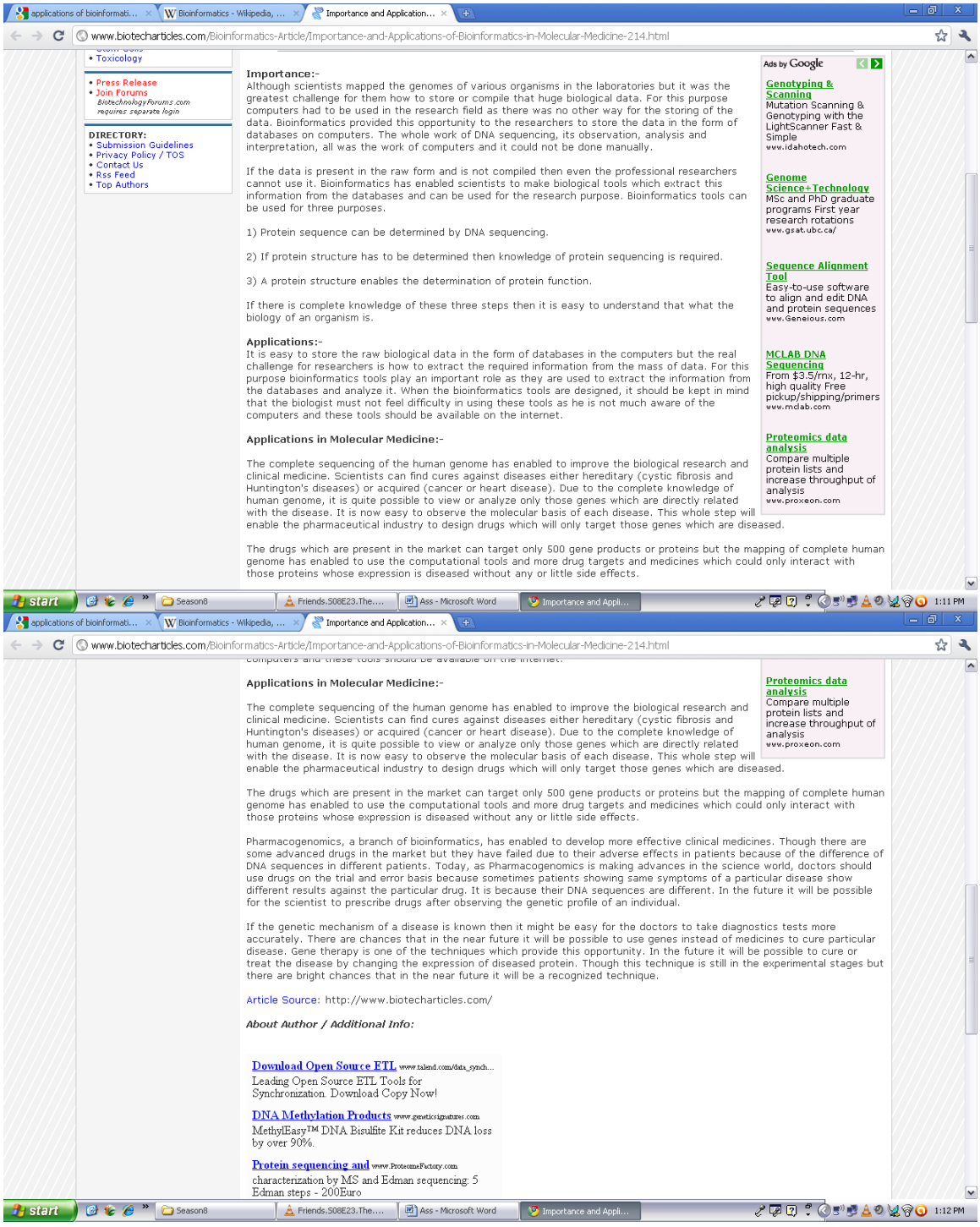
8. **Good data visualization skills.** You'll need to be able to take complex data and interpret it into models and other ways that make it understandable for biologists and other team members.

9. **Experience with bioinformatics tools,** such as Blast, BLAT, sequence analysis algorithms and clustering tools.

10. **Experience in using bioinformatics resources,** such as the UCSC genome browser and Entrez. You'll need to be familiar with the National Center for Bioinformatics (NCBI) and the database and analysis tools available on their website.

If you have these ten skills, you will have no trouble finding the bioinformatics job you want. You'll be a valuable member of a team that is doing life-changing research in molecular biology and its applications.

A9



Refer to pdf

A10

Bioinformatics

Bioinformatics is an extensive research subject which spans from biomedicine via computer science to mathematics. Bioinformatics ties together the life sciences and makes major contributions to, among other subjects, functional, comparative and structural genomics, proteomics and pharmacogenomics. At the same time, research in bioinformatics gives rise to many fascinating abstract problems in mathematics and computer science, in a similar way as physics has in previous centuries.

Bioinformatics is a multi-disciplinary subject, with research questions taken from biology, medicine, pharmacology and agricultural studies. In bioinformatics research, theoretical models, algorithms and methods are developed for the computational modeling of complex biological systems. Computer science and mathematics form the theoretical foundations for a major part of bioinformatics, using methods from subdisciplines including discrete mathematics, theoretical computer science, logic, artificial intelligence, information theory, probability theory, mathematical statistics, data base design, programming, optimization theory, analysis, geometry, simulation methods and visualization paradigms. In addition, scientific knowledge from a multitude of other disciplines, among which can be found automatic control, numerical analysis, computational science and linguistics, is used and augmented. By bridging the gaps between these disciplines, bioinformatics makes it possible for biological research to make the transition from an area low in data intensity, to one which is highly data-intense. To date, this research has resulted in such progress as the mapping of the human genome, projects in functional and comparative genomics, and new methods for drug development. Bioinformatics is constantly opening up new possibilities for scientific research within the life sciences. At the same time, research in biology continuously contributes new types of problems which the informatics community has not previously encountered, helping to drive these areas forward as well.

Bioinformatics encompasses development and application of information science for gathering, storage, processing and analysis of the large amounts of heterogeneous biological, medical and pharmacological data which are produced in modern biology. These data which are to be processed can, for example, be the results of large-scale projects for mapping of genomes, transcriptomes, proteomes and metabolomes, but can also come from follow-up projects which strive to utilize the information from the earlier infra-structure related projects to study the functions of biological systems. Bioinformatics contributes to the progress of a large number of research areas within the life sciences. There are many examples of the importance of bioinformatics research to, for instance, the construction of the large databases. These databases have made it possible to systematically store and search biologically relevant information among the enormous amounts of raw data generated within the genome projects; and further within the development of theory, algorithms and software for the identification of functional elements such as genes, regulatory sequences and three-dimensional protein structure from one-dimensional DNA sequence data. Moreover, this research drives the development of methods for identifying and elucidating function of genetic networks for the regulation of traits which are of interest in medical, evolutionary, agricultural and general biological research. By contributing to the development of better methods for data management within the biological sciences, it will become possible to construct more realistic models of biological and physiological systems by making use of and bringing together knowledge which has been attained within a wide range of biological disciplines

Bioinformatics is a multidisciplinary field and requires people from different working areas. It is the combination of biology and computer science and is a new emerging field that helps in collecting, linking, and manipulating different types of biological information to discover new biological insight. Before the emergence of bioinformatics, all scientists working in different biological fields, such as human science, ecological science and many other fields, feel a necessity of some tool that helps them to work together. They knew they are all interlinked and had important information for each other, but they did not know how to integrate. In such circumstances, bioinformatics emerges to help these scientists or researchers in fast research and leads to quick inventions by providing readily available information with the help of computer technology.

Scientist and researchers spend their whole life in inventing things for human benefits. After so many years of development, they have collected huge amount of valuable data from

their experiments all over the world and still this collection is continue and will always continue for the better development of human being. Sometimes, they need to repeat the old research because either it is hard to obtain old data or they do not know whether it exist or not; this wastes their valuable time. Let us take an example of DNA identification. Every species or human beings have particular DNA strands that contain the genetic instructions used in the development and functioning of all known living organisms. By identifying DNA information one can trace generations' links and can find the root of different disease. Earlier it was hard to manage this information. In order to collect and link DNA information from all over the world and to solve many medical complications, bioinformatics is a very helpful hand for them.

In addition, scientists also need a tool that can interlink information from different areas like biology, statistics, genomics etc to make their research faster. For instance, they may need some data regarding effects of particular gene on human being and its effect on animal or on other species, so that they can interlink and generate some beneficial results or antidote that helps in human development. Eventually, bioinformatics provides that help in interlinking information from different fields and leads to quick results.

Finally, bioinformatics also helps in digitizing the information available on paper or in the form of specimen, so that with the help of internet it could be easily available to everyone everywhere. These days, computer is an important part of every research without which so fast development cannot be imagine. Moreover, these days it is important to keep everyone aware of current developments. This will enable everyone to enhance their academic and research skills in their fields at minimum expanse of time, money, and matters. In this scenario, bioinformatics makes information readily available by collecting, linking, and manipulating.

Briefly, bioinformatics is playing a vital role in development of society by providing quick information and making research fast. It is using today's computer technology and biological research together very efficiently. This field is going to generate more opportunities in future for all people working in different areas.

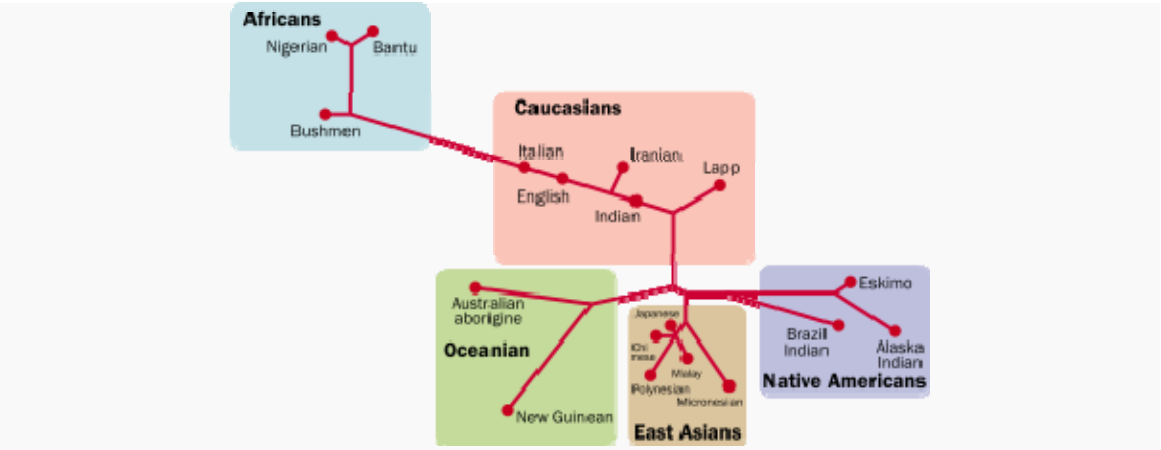
Comp bio

A1

Neighbor-joining

From Wikipedia, the free encyclopedia

In bioinformatics, **neighbor-joining** is a bottom-up clustering method for the creation of phenetic trees (phenograms), created by Naruya Saitou and Masatoshi Nei[1]. Usually used for trees based on DNA or protein sequence data, the algorithm requires knowledge of the distance between each pair of taxa (e.g., species or sequences) in the tree.



This genetic distance map made in 2002 is an estimate of 18 world human groups by a neighbour-joining method based on 23 kinds of genetic information.[2]

Contents

[hide]

[edit]The algorithm

Neighbor-joining starts with a completely unresolved tree, whose topology corresponds to that of a star network, and iterates over the following steps until the tree is completely resolved and all branch lengths are known:

1. Based on the current distance matrix calculate the matrix Q (defined below).
2. Find the pair of taxa in Q with the lowest value. Create a node on the tree that joins these two taxa (i.e., join the closest neighbors, as the algorithm name implies).
3. Calculate the distance of each of the taxa in the pair to this new node.
4. Calculate the distance of all taxa outside of this pair to the new node.
5. Start the algorithm again, considering the pair of joined neighbors as a single taxon and using the distances calculated in the previous step.

[edit]The Q-matrix

Based on a distance matrix relating the r taxa, calculate Q as follows:

$$Q(i,j) = (r-2)d(i,j) - \sum_{k=1}^{r} d(i,k) - \sum_{k=1}^{r} d(j,k)$$

where $d(i,j)$ is the distance between taxa i and j.

[edit]Distance of the pair members to the new node

For each neighbor in the pair just joined, use the following formula to calculate the distance to the new node. (Taxa f and g are the paired taxa and u is the newly generated node.):

$$d(f,u) = \frac{1}{2}d(f,g) + \frac{1}{2(r-2)}\left[\sum_{k=1}^{r} d(f,k) - \sum_{k=1}^{r} d(g,k)\right]$$

and, by reflection:

$$d(g,u) = d(f,g) - d(f,u)$$

[edit]Distance of the other taxa to the new node

For each taxon not considered in the previous step, we calculate the distance to the new node as follows:

$$d(u,k) = \frac{1}{2}[d(g,k) + d(f,k) - d(f,g)]$$

where u is the new node, k is the node for which we want to calculate the distance and f and g are the members of the pair just joined.

[edit]Complexity

Since neighbor-joining on a set of r taxa takes requires $r - 3$ iterations, and since at each step one has to build and search $r \times r$ matrices, the algorithm can be implemented so as to obtain a time complexity of $O(r^3)$.

[edit]Example

Let us assume that we have four taxa (A, B, C, D) and the following distance matrix:

	A	**B**	**C**	**D**
A	0	7	11	14
B	7	0	6	9
C	11	6	0	7
D	14	9	7	0

We obtain the following values for the Q matrix:

	A	**B**	**C**	**D**
A	0	−40	−34	−34
B	−40	0	−34	−34
C	−34	−34	0	−40
D	−34	−34	−40	0

In the example above, two pairs of taxa have the lowest value, namely −40. We can select either of them for the second step of the algorithm. We follow the example assuming that we joined taxa A and B together. If *u* denotes the new node, then the branch lengths of {A, u} and {B, u} are respectively 6 and 1, by using the above formula.

We then proceed to updating the distance matrix, by computing $d(u,k)$ according to the above formula for every node *k*. In this case, we obtain $d(u,C) = 5$ and $d(u,D) = 8$. The resulting distance matrix is:

	AB	**C**	**D**
AB	0	5	8
C	5	0	7
D	8	7	0

We can start the procedure anew taking this matrix as the original distance matrix. In our example, it suffices to do one more step of the recursion to obtain the complete tree.

[edit]Advantages and disadvantages

Neighbor-joining is based on the minimum-evolution criterion, i.e. the topology that gives the least total branch length is preferred at each step of the algorithm. However, neighbor-joining may not find the true tree topology with least total branch length because it is a greedy algorithm that constructs the tree in a step-wise fashion. Even though it is sub-optimal in this sense, it has been extensively tested and usually finds a tree that is quite close to the optimal tree. Nevertheless, it has been largely superseded by phylogenetic methods that do not rely on distance measures and offer superior accuracy under most conditions.

The main virtue of neighbor-joining relative to these other methods is its computational efficiency. That is, neighbor-joining is a polynomial-time algorithm. It can be used on very large data sets for which other means of

analysis (e.g. minimum evolution, maximum parsimony, maximum likelihood) are computationally prohibitive. Unlike the UPGMA algorithm for tree reconstruction, neighbor-joining does not assume that all lineages evolve at the same rate (molecular clock hypothesis) and produces an unrooted tree. Rooted trees can be created by using an outgroup and the root can then effectively be placed on the point in the tree where the edge from the outgroup connects.

Furthermore, neighbor-joining is statistically consistent under many models of evolution. Hence, given data of sufficient length, neighbor-joining will reconstruct the true tree with high probability.

Atteson[3] proved that if each entry in the distance matrix differs from the true distance by less than half of the shortest branch length in the tree, then neighbor joining will construct the correct tree.

RapidNJ and NINJA are fast implementations of the neighbor joining algorithm

The Neighbor-Joining Method

Neighbor-joining (Saitou and Nei, 1987) is a method that is related to the cluster method but does not require the data to be ultrametric. In other words it does not require that all lineages have diverged by eaqual amounts. The method is especially suited for datasets comprising lineages with largely varying rates of evolution. It can be used in combination with methods that allow correction for superimposed substitutions.

The following programs are available

- Neighbor of the Phylip package (Jo Felsentein, Univ. Washington),
- ClustalW (D. Higgins, EMBL) ,
- Distnj in the Protml package (Adachi and Hasegawa, Univ. Tokyo)

The neighbor-joining method is a special case of the star decomposition method. In contrast to cluster analysis neighbor-joining keeps track of nodes on a tree rather than taxa or clusters of taxa. The raw data are provided as a distance matrix and the initial tree is a star tree. Then a modified distance matrix is constructed in which the separation between each pair of nodes is adjusted on the basis of their average divergeance from all other nodes. The tree is constructed by linking the least-distant pair of nodes in this modified matrix. When two nodes are linked, their common ancestral node is added to

the tree and the terminal nodes with their respective branches are removed from the tree. This pruning process converts the newly added common ancestor into a terminal node on a tree of reduced size. At each stage in the process two terminal nodes are replaced by one new node. The process is complete when two nodes remain, separated by a single branch.

Negative branch lengths

As the neighbor-joining algorithm seeks to represent the data in the form of an additive tree, it can assign a negative length to the branch. Here the interpretation of branch lengths as an estimated number of substitutions gets into difficulties. When this occurs it is adviced to set the branch length to zero and transfer the difference to the adjacent branch length so that the total distance between an adjacent pair of terminal nodes remains unaffected. This does not alter the overall topology of the tree (Kuhner and Felsenstein, 1994).

Advantages and disadvantages of the neighbor-joining method

- Advantages
  - is fast and thus suited for large datasets and for bootstrap analysis
  - permist lineages with largely different branch lengths
  - permits correction for multiple substitutions
- Disadvantages
  - sequence information is reduced
  - gives only one possible tree
  - strongly dependent on the model of evolution used.

NB: *especially its suitability to handle large datasets has led to the fact that the method is widely used by molecular evolutionists. With the rapid growth of sequence databases it is still one of the few methods that allows the rapid inclusion of all homologous sequences present in the database in a single tree. A good example can be found in the Ribosomal Database Project that maintains a tree of life based on all available ribosomal RNA sequences.*

Example of the method

Suppose we have the following tree:

Since B and D have accumulated mutations at a higher rate than A. The Three-point criterion is violated and the UPGMA method cannot be used since this would group together A and C rather than A and B. In such a case the neighbor-joining method is one of the recommended methods.

The raw data of the tree are represented by the following distance matrix:

	A	B	C	D	E
B	5				
C	**4**	7			
D	7	10	7		
E	6	9	6	5	
F	8	11	8	9	8

We have in total 6 OTUs (N=6).

Step 1: We calculate the net divergence r (i) for each OTU from all other OTUs

```
r(A) =  5+4+7+6+8=30
r(B) = 42
r(C) = 32
r(D) = 38
r(E) = 34
r(F) = 44
```

Step 2: Now we calculate a new distance matrix using for each pair of OUTs the formula:

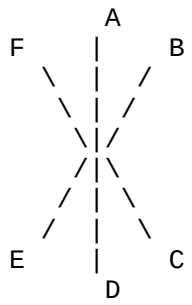
M(ij)=d(ij) - [r(i) + r(j)]/(N-2) or in the case of the pair A,B:

M(AB)=d(AB) -[(r(A) + r(B)]/(N-2) = -13

	A	B	C	D	E

B	-13				
C	-11.5	-11.5			
D	-10	-10	-10.5		
E	-10	-10	-10.5	-13	
F	-10.5	-10.5	-11	-11.5	-11.5

Now we start with a star tree:



Step 3: Now we choose as neighbors those two OTUs for which Mij is the smallest. These are A and B and D and E. Let's take A and B as neighbors and we form a new node called U. Now we calculate the branch length from the internal node U to the external OTUs A and B.

```
S(AU) =d(AB) / 2 + [r(A)-r(B)] / 2(N-2) = 1
S(BU) =d(AB) -S(AU) = 4
```

Step 4: Now we define new distances from U to each other terminal node:

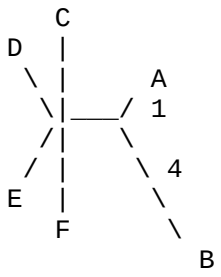
```
d(CU) = d(AC) + d(BC) - d(AB) / 2 = 3
d(DU) = d(AD) + d(BD) - d(AB) / 2 = 6
d(EU) = d(AE) + d(BE) - d(AB) / 2 = 5
d(FU) = d(AF) + d(BF) - d(AB) / 2 = 7
```

and we create a new matrix:

	U	C	D	E
C	3			
D	6	7		

E	5	6	5	
F	7	8	9	8

The resulting tree will be the following:



N= N-1 = 5
The entire prodcedure is repeated starting at step 1

Last updated: 8 August 1997.

---

created by :Fred Opperdoes

a2

Differences between PAM and BLOSUM

1. PAM matrices are based on an explicit evolutionary model (i.e. replacements are counted on the branches of a phylogenetic tree), whereas the BLOSUM matrices are based on an implicit model of evolution.
2. The PAM matrices are based on mutations observed throughout a global alignment, this includes both highly conserved and highly mutable regions. The BLOSUM matrices are based only on highly conserved regions in series of alignments forbidden to contain gaps.
3. The method used to count the replacements is different: unlike the PAM matrix, the BLOSUM procedure uses groups of sequences within which not all mutations are counted the same.
4. Higher numbers in the PAM matrix naming scheme denote larger evolutionary distance, while larger numbers in the BLOSUM matrix naming scheme denote higher sequence similarity and therefore smaller evolutionary distance. Example: PAM150 is used for

more distant sequences than PAM100; BLOSUM62 is used for closer sequences than Blosum50.

PAM stands for 'Point Accepted Mutations'

PAM matrices are developed based on the PAM model of evolution. This model assumes that the evolutionary changes occur according to Markov Model which states that residue mutations are independent of previous mutations.

PAM is a unit of evolutionary divergence in which 1% of the amino acids had changed. This doesn't mean that after 100 PAM's all the aminoacids are different.

The PAM (Point Accepted Mutation) matrices were developed by Margaret Dayhoff in the 1970s. The basi for their approach is to obtain substitution data from alignments between very similar proteins, allowing for the evolutionary relationships of the protein families, and then extrapolate this information to longer evolutionary distances. They constructed hypothetical hylogenetic trees using Parsiminy method.

The similarity scores are obtained by taking the natural logarithms of the frequencies. Thus they have constructed the PAM1 Matrix. The PAM1 matrix estimates what rate of substitution would be expected if 1% of the amino acids had changed. The PAM1 matrix is used as the basis for calculating other matrices by assuming that repeated mutations would follow the same pattern as those in the PAM1 matrix, and multiple substitutions can occur at the same site. Using this logic, Dayhoff derived matrices as high as PAM250. Usually the PAM 30 and the PAM70 are used.

A matrix for divergent sequences can be calculated from a matrix for closely related sequences by taking the second matrix to a power. For instance, we can roughly approximate the matrix2 from the matrix1 by saying 'Matrix2 Is Equal To The Square Of Matrix1'. This is how the PAM250 matrix is calculated.

Problems with PAM Matrices:

1) Evolutionary rates vary greatly with in a protein
2) Each position has its own 3-D environment
3) Environment changes over evolutionary time. BLOSUM matrices were constructed by Henikoff and Henikoff.

* They derived the BLOSUM matrices from a set of ungapped regions from protein database called the BLOCKS database. The procedure is as follows:
* Whenever two sequence were found to have sequence identity greater than some L%, they were put into the same CLuster. This way they clustered sequences from each BLOCK.
* Then the frequency Aab, of observing a in one cluster aligned against b in another cluster. Correction for the sizes of the clusters is done byweighting each occurence by 1/(n1n2), where n1 and n2 are the respective cluster sizes.
* From Aab they estimated qa and Pab. From these they derived the score matrix entries using the standard equation s(a,b) = log Pab/qa qb.
* The resulting log odds matrices were scaled and rounded to the nearest integer value.

The Matrices for L=62 and L=50 are widely used for pairwise alignment and Database searching.

BLOSUM 62 is the standard for ungapped alignments and BLOSUM50 is the standard for

gapped alignments.

(Lower L values correspond to longer evolutionary time and are applicable for more distant searches.) Difference Between PAM & BLOSUM

1. PAM matrices are based on an explicit evolutionary model (i.e. replacements are counted on the branches of a phylogenetic tree), whereas the BLOSUM matrices are based on an implicit model of evolution.
2. The PAM matrices are based on mutations observed throughout a global alignment, this includes both highly conserved and highly mutable regions. The BLOSUM matrices are based only on highly conserved regions in series of alignments forbidden to contain gaps.
3. The method used to count the replacements is different: unlike the PAM matrix, the BLOSUM procedure uses groups of sequences within which not all mutations are counted the same.
4. Higher numbers in the PAM matrix naming scheme denote larger evolutionary distance, while larger numbers in the BLOSUM matrix naming scheme denote higher sequence similarity and therefore smaller evolutionary distance. Example: PAM150 is used for more distant sequences than PAM100; BLOSUM62 is used for closer sequences than Blosum50.

PAM & BLOSUM Equivalents

PAM 100 => BLOSUM 90
PAM 120 => BLOSUM 80
PAM 160 => BLOSUM 60
PAM 200 => BLOSUM 52
PAM 250 => BLOSUM 40

A3

Phylogenetic tree

From Wikipedia, the free encyclopedia

"ptree" redirects here. For Patricia tree, see Radix tree.

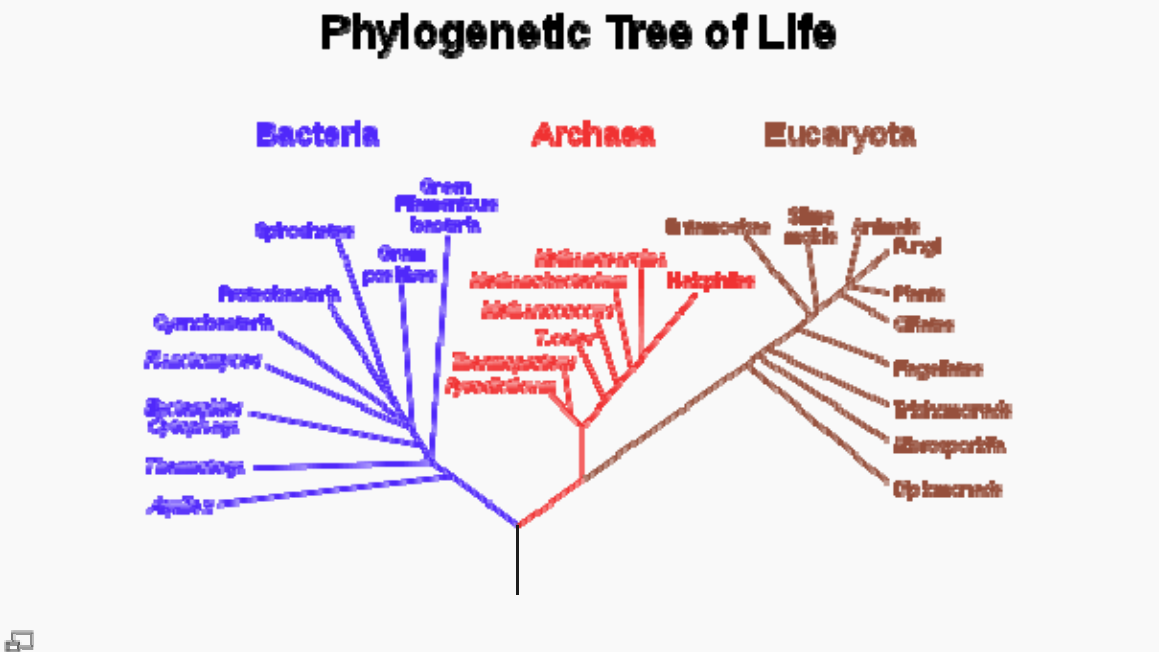


Fig. 1: A speculatively rooted tree for rRNA genes, showing major branches Bacteria,Archaea, and Eucaryote.

A **phylogenetic tree** or **evolutionary tree** is a branching diagram or "tree" showing the inferredevolutionary relationships among various biological species or other entities based upon similarities and differences in their physical and/or genetic characteristics. The taxa joined together in the tree are implied to have descended from a common ancestor. In a **rooted** phylogenetic tree, each node with descendants represents the inferred most recent common ancestor of the descendants, and the edge lengths in some trees may be interpreted as time estimates. Each node is called a taxonomic unit. Internal nodes are generally called hypothetical taxonomic units (HTUs) as they cannot be directly observed. Trees are useful in fields of biology such as bioinformatics, systematics and comparative phylogenetics.

Contents

[hide]

[edit]History

The idea of a "tree of life" arose from ancient notions of a ladder-like progression from lower to higher forms of life (such as in the Great Chain of Being). Early representations of *branching*phylogenetic trees include a "Paleontological chart" showing the geological relationships among plants and animals in the book Elementary Geology, by Edward Hitchcock (first edition: 1840).

Charles Darwin (1859) also produced one of the first illustrations and crucially popularized the notion of an evolutionary "tree" in his seminal book *The Origin of Species*. Over a century later, evolutionary

biologists still use tree diagrams to depict evolution because such diagrams effectively convey the concept that speciation occurs through the adaptive and randomsplitting of lineages. Over time, species classification has become less static and more dynamic.

[edit]Formal definition

In the analysis of genetic evolution, we are given a set X of k sequences, each stands for an extant species. Let Y be a set of hypothetical sequences, where $Y \cap X = \emptyset$ (usually, each sequence in Y could represent an extinct species). An evolutionary tree $T_{X,Y}$ for X is a weighted (sometimes rooted) tree of $|X \cup Y|$ nodes, where each node is associated with a unique sequence in $|X \cup Y|$. The cost of an edge is the edit distance between the two sequence associated with the ends of the edge. The cost $c(T_{X,Y})$ of the tree $T_{X,Y}$ is the total cost of all the edges in $T_{X,Y}$.[1]

[edit]Types

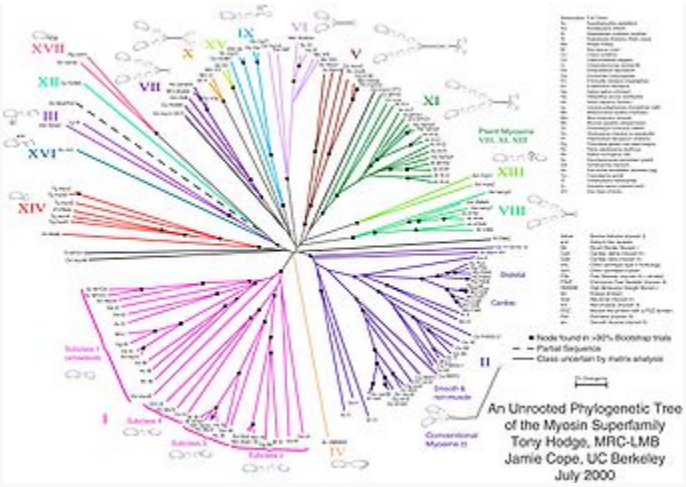


Fig. 1: Unrooted tree of the myosin supergene family[2]

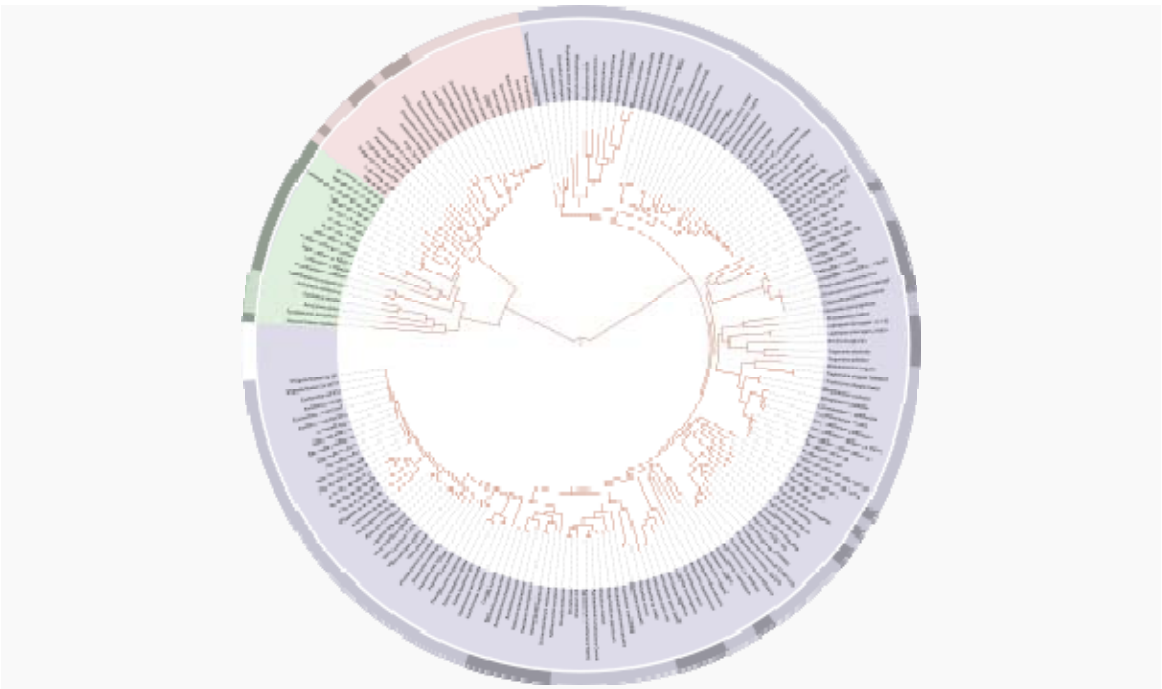


Fig. 2: A highly resolved, automatically generated Tree Of Life, based on completely sequenced genomes.[3][4]

A phylogenetic tree, showing how Eukaryota and Archaea are more closely related to each other than to Bacteria, based on Cavalier-Smith's theory of bacterial evolution. (Cf. LUCA, Neomura.)

A **rooted** phylogenetic tree is a directed tree with a unique node corresponding to the (usually imputed) most recent common ancestor of all the entities at the leaves of the tree. The most common method for rooting trees is the use of an uncontroversial outgroup — close enough to allow inference from sequence or trait data, but far enough to be a clear outgroup.

Unrooted trees illustrate the relatedness of the leaf nodes without making assumptions about ancestry at all. While unrooted trees can always be generated from rooted ones by simply omitting the root, a root cannot be inferred from an unrooted tree without some means of identifying ancestry; this is

normally done by including an outgroup in the input data or introducing additional assumptions about the relative rates of evolution on each branch, such as an application of themolecular clock hypothesis. Figure 1 depicts an unrooted phylogenetic tree for myosin, a superfamily of proteins.[5]

Both rooted and unrooted phylogenetic trees can be either **bifurcating** or **multifurcating**, and either **labeled** or**unlabeled**. A rooted bifurcating tree has exactly two descendants arising from each interior node (that is, it forms abinary tree), and an unrooted bifurcating tree takes the form of an unrooted binary tree, a free tree with exactly three neighbors at each internal node. In contrast, a rooted multifurcating tree may have more than two children at some nodes and an unrooted multifurcating tree may have more than three neighbors at some nodes. A labeled tree has specific values assigned to its leaves, while an unlabeled tree, sometimes called a **tree shape**, defines a topology only. The number of possible trees for a given number of leaf nodes depends on the specific type of tree, but there are always more multifurcating than bifurcating trees, more labeled than unlabeled trees, and more rooted than unrooted trees. The last distinction is the most biologically relevant; it arises because there are many places on an unrooted tree to put the root. For labeled bifurcating trees, there are

$$(2n-3)!! = \frac{(2n-3)!}{2^{n-2}(n-2)!}\text{, for}\, n \geq 2$$

total rooted trees and

$$(2n-5)!! = \frac{(2n-5)!}{2^{n-3}(n-3)!}\text{, for}\, n \geq 3$$

total unrooted trees, where n represents the number of leaf nodes. Among labeled bifurcating trees, the number of unrooted trees with n leaves is equal to the number of rooted trees with $n-1$ leaves.[6]

A **dendrogram** is a broad term for the diagrammatic representation of a phylogenetic tree.

A **cladogram** is a phylogenetic tree formed using cladistic methods. This type of tree only represents a branching pattern, i.e., its branch lengths do not represent time or relative amount of character change.

A **phylogram** is a phylogenetic tree that has branch lengths proportional to the amount of character change.

A **chronogram** is a phylogenetic tree that explicitly represents evolutionary time through its branch lengths.

[edit]Construction

Main article: Computational phylogenetics

Phylogenetic trees among a nontrivial number of input sequences are constructed using computational phylogenetics methods. Distance-matrix methods such as neighbor-joining or UPGMA, which calculate genetic distance from multiple sequence alignments, are simplest to implement, but do not invoke an evolutionary model. Many sequence alignment methods such as ClustalW also create trees by using the simpler algorithms (i.e. those based on distance) of tree construction. Maximum parsimony is another simple method of estimating phylogenetic trees, but implies an implicit model of evolution (i.e. parsimony). More advanced methods use the optimality criterion of maximum likelihood, often within a Bayesian Framework, and apply an explicit model of evolution to phylogenetic tree estimation.[6] Identifying the optimal tree using many of these techniques is NP-hard,[6] so heuristic search and optimization methods are used in combination with tree-scoring functions to identify a reasonably good tree that fits the data.

Tree-building methods can be assessed on the basis of several criteria:[7]

- efficiency (how long does it take to compute the answer, how much memory does it need?)
- power (does it make good use of the data, or is information being wasted?)
- consistency (will it converge on the same answer repeatedly, if each time given different data for the same model problem?)
- robustness (does it cope well with violations of the assumptions of the underlying model?)
- falsifiability (does it alert us when it is not good to use, i.e. when assumptions are violated?)

Tree-building techniques have also gained the attention of mathematicians. Trees can also be built using T-theory.[8]

[edit]Limitations

Although phylogenetic trees produced on the basis of sequenced genes or genomic data in different species can provide evolutionary insight, they have important limitations. They do not necessarily accurately represent the species evolutionary history. The data on which they are based is noisy; the analysis can be confounded by horizontal gene transfer,[9] hybridisationbetween species that were not nearest neighbors on the tree before hybridisation takes place, convergent evolution, and conserved sequences.

Also, there are problems in basing the analysis on a single type of character, such as a single gene or protein or only on morphological analysis, because such trees constructed from another unrelated data source often differ from the first, and therefore great care is needed in inferring phylogenetic relationships among species. This is most true of genetic material that is subject to lateral gene transfer and recombination, where different haplotype blocks can have different histories. In general, the output tree of a phylogenetic analysis is an estimate of the *character*'s phylogeny (i.e. a gene tree) and not the phylogeny of the taxa (i.e. species tree) from which these characters were sampled, though ideally, both should be very close. For this reason, serious phylogenetic studies generally use a combination of genes that come from different genomic sources (e.g., from mitochondrial or plastid vs. nuclear genomes), or genes that would be expected to evolve under different selective regimes, so that homoplasy (false homology) would be unlikely to result from natural selection.

When extinct species are included in a tree, they are terminal nodes, as it is unlikely that they are direct ancestors of any extant species. Scepticism might be applied when extinct species are included in trees that are wholly or partly based on DNA sequence data, due to the fact that little useful "ancient DNA" is preserved for longer than 100,000 years, and except in the most unusual circumstances no DNA sequences long enough for use in phylogenetic analyses have yet been recovered from material over 1 million years old.

In some organisms, endosymbionts have an independent genetic history from the host.

Phylogenetic networks are used when bifurcating trees are not suitable, due to these complications which suggest a more reticulate evolutionary history of the organisms sampled..

[edit]See also

Evolutionary biology portal

[edit]The "tree of life"

- Evolutionary history of life - An overview of the major time periods of life on earth
- Life - The top level for Wikipedia articles on living species, reflecting a diversity of classification systems.
- Three-domain system (cell types)
- Wikispecies - An external Wikimedia Foundation project to construct a "tree of life" appropriate for use by scientists

What Is Sequin

Sequin is a stand-alone software tool developed by the NCBI for submitting and updating entries to the GenBank sequence database. It is capable of handling simple submissions that contain a single short mRNA sequence, and complex submissions containing long sequences, multiple annotations, gapped sequences, or phylogenetic and population studies. Getting Started with Sequin is a brief overview of the program.

Overview of Sequin

General information about the submitters and the sequence is entered into Sequin on a pair of introductory forms. The user is prompted to import the nucleotide and any associated amino acid sequences into the program in FASTA format. From these basic data, Sequin prepares a window containing the initial database record. Many additional forms, that provide space for adding new or modifying existing annotations, are accessible from this window. Sequin also contains a built-in validation tool which checks the record for accuracy and consistency and suggests solutions for many problems.

Displaying the Record

Sequin can display the initial record in a number of different formats. The record can be seen as it would appear in the GenBank, EMBL, or DDBJ databases. Sequences and certain annotations can also be viewed in a graphical format, permitting, for example, a schematic display of the locations of mRNAs and coding sequences along a genomic DNA sequence. If you have submitted a set of aligned sequences, the alignments can be displayed as well.

Advantages of Sequin

Sequin automatically performs a number of functions necessary for submission. For example, Sequin obtains the proper genetic code from the name of the organism and automatically determines coding region intervals on the nucleotide sequence by back-translation of the protein sequence. Researchers who submit large numbers of related sequences can make use of the fact that Sequin can also interpret the name of the organism, strain, and other biological source information directly from a line of data entered along with each nucleotide sequence. Sequin also allows the designation of groups of sequences as population, phylogentic, mutant, or environmental sets for display in the

PopSet division of Entrez and the propagation of annotation from one member of the set to all others through an alignment.

Sequence Annotation Tools

A number of powerful sequence annotation tools have been integrated into Sequin. The ORF Finder identifies open reading frames within the sequence. The Sequence Editor allows basic editing and translation of nucleotide sequences. With the Update Sequence function, Sequin can import and align a replacement or overlapping sequence to the sequence in the record and propagate features between the two aligned sequences. In Network-Aware mode, Sequin integrates PubMed searching. Sequin also allows the propagation of features from one sequence in an aligned set to other sequences within the set.

A4

What Is Sequin

Sequin is a stand-alone software tool developed by the NCBI for submitting and updating entries to the GenBank sequence database. It is capable of handling simple submissions that contain a single short mRNA sequence, and complex submissions containing long sequences, multiple annotations, gapped sequences, or phylogenetic and population studies. Getting Started with Sequin is a brief overview of the program.

Overview of Sequin

General information about the submitters and the sequence is entered into Sequin on a pair of introductory forms. The user is prompted to import the nucleotide and any associated amino acid sequences into the program in FASTA format. From these basic data, Sequin prepares a window containing the initial database record. Many additional forms, that provide space for adding new or modifying existing annotations, are accessible from this window. Sequin also contains a built-in validation tool which checks the record for accuracy and consistency and suggests solutions for many problems.

Displaying the Record

Sequin can display the initial record in a number of different formats. The record can be seen as it would appear in the GenBank, EMBL, or DDBJ

databases. Sequences and certain annotations can also be viewed in a graphical format, permitting, for example, a schematic display of the locations of mRNAs and coding sequences along a genomic DNA sequence. If you have submitted a set of aligned sequences, the alignments can be displayed as well.

Advantages of Sequin

Sequin automatically performs a number of functions necessary for submission. For example, Sequin obtains the proper genetic code from the name of the organism and automatically determines coding region intervals on the nucleotide sequence by back-translation of the protein sequence. Researchers who submit large numbers of related sequences can make use of the fact that Sequin can also interpret the name of the organism, strain, and other biological source information directly from a line of data entered along with each nucleotide sequence. Sequin also allows the designation of groups of sequences as population, phylogentic, mutant, or environmental sets for display in the PopSet division of Entrez and the propagation of annotation from one member of the set to all others through an alignment.

Sequence Annotation Tools

A number of powerful sequence annotation tools have been integrated into Sequin. The ORF Finder identifies open reading frames within the sequence. The Sequence Editor allows basic editing and translation of nucleotide sequences. With the Update Sequence function, Sequin can import and align a replacement or overlapping sequence to the sequence in the record and propagate features between the two aligned sequences. In Network-Aware mode, Sequin integrates PubMed searching. Sequin also allows the propagation of features from one sequence in an aligned set to other sequences within the set.

A5

P-value

From Wikipedia, the free encyclopedia

In statistical significance testing, the **p-value** is the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesisis true. One often "rejects the null hypothesis" when the p-value is less than 0.05 or 0.01, corresponding respectively to a 5% or 1% chance of rejecting the null hypothesis when it is true (Type I error). When the null hypothesis is rejected, the result is said to be statistically significant.

A closely related concept is the **E-value**,[1] which is the average number of times in multiple testing that one expects to obtain a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. The E-value is the product of the number of tests and the p-value.

Interpretation

Traditionally, one rejects the null hypothesis if the p-value is smaller than or equal to the significance level,[2] often represented by the Greek letter α (alpha). A significance level of 0.05 would deem as extraordinary any results that are only 5% likely (or less), given that the null hypothesis is true. In this case a p-value less than 0.05 would be *rejected at the 5% (significance) level*.

When we ask whether a given coin is fair, often we are interested in the deviation of our result from the equality of numbers of heads and tails. In this case, the deviation can be in either direction, favoring either heads or tails. Thus, in this example of 14 heads and 6 tails, we may want to calculate the probability of getting a result deviating by at least 4 from parity in either direction (two-sided test). This is the probability of getting at least 14 heads or at least 14 tails. As the binomial distribution is symmetrical for a fair coin, the two-sided p-value is simply twice the above calculated single-sided p-value; *i.e.*, the two-sided p-value is 0.115.

In the above example we thus have:

- null hypothesis (H_0): fair coin; P(heads) = 0.5
- observation O: 14 heads out of 20 flips; and
- p-value of observation O given H_0 = Prob(≥ 14 heads or ≥ 14 tails) = 0.115.

The calculated p-value exceeds 0.05, so the observation is consistent with the null hypothesis — that the observed result of 14 heads out of 20 flips can be ascribed to chance alone — as it falls within the range of what would happen 95% of the time were the coin in fact fair. In our example, we fail to reject the null hypothesis at the 5% level. Although the coin did not fall evenly, the deviation from expected outcome is considered small enough to be consistent with chance.

However, had one more head been obtained, the resulting p-value (two-tailed) would have been 0.0414 (4.14%). This time the null hypothesis – that the observed result of 15 heads out of 20 flips can be ascribed to chance alone – is rejected when using a 5% cut-off.

To understand both the original purpose of the p-value *p* and the reasons *p* is so often misinterpreted, it helps to know that *p* constitutes the main result of statistical significance testing (not to be confused with hypothesis testing), popularized by Ronald A. Fisher. Fisher promoted

this testing as a method of statistical inference. To call this testing inferential is misleading, however, since inference makes statements about general hypotheses based on observed data, such as the post-experimental probability a hypothesis is true. As explained above, *p* is instead a statement about data *assuming* the null hypothesis; consequently, indiscriminately considering *p* as an inferential result can lead to confusion, including many of the misinterpretations noted in the next section.

On the other hand, Bayesian inference, the main alternative to significance testing, generates probabilistic statements about hypotheses based on data (and *a priori* estimates), and therefore truly constitutes inference. Bayesian methods can, for instance, calculate the probability that the null hypothesis H_0 above is true assuming an *a priori* estimate of the probability that a coin is unfair. Since *a priori* we would be quite surprised that a coin could consistently give 75% heads, a Bayesian analysis would find the null hypothesis (that the coin is fair) quite probable even if a test gave 15 heads out of 20 tries (which as we saw above is considered a "significant" result at the 5% level according to its p-value).

P-values

The number of random HSPs with score >= *S* is described by a Poisson distribution [10,11]. This means that the probability of finding exactly *a* HSPs with score >=*S* is given by

$$e^{-E}\frac{E^{a}}{a!} \qquad (4)$$

where *E* is the *E*-value of *S* given by equation (1) above. Specifically the chance of finding zero HSPs with score >=*S* is e^{-E}, so the probability of finding at least one such HSP is

$$P = 1 - e^{-E} \qquad (5)$$

This is the *P*-value associated with the score *S*. For example, if one expects to find three HSPs with score >= *S*, the probability of finding at least one is 0.95. The BLAST programs report *E*-value rather than *P*-values because it is easier to understand the difference between, for example, *E*-value of 5 and 10 than *P*-values of 0.993 and 0.99995. However, when $E < 0.01$, *P*-values and *E*-value are nearly identical.

A6

omputational Biology and Bioinformatics

POSTED *by* WALTER *on* JUN 23, 2010 · VIEWS: 718 VIEWS · COMMENTS: 2

I've been working as a Computational Biologist at Covance's Biomarker Center of Excellence since April. I'm the first computational scientist they've brought on and, perhaps not surprisingly, I've been asked several times over the past three months, "**What is a Computational Biologist?**"

Biology in the post-genome world has transformed from a largely laboratory-based science to one that integrates experimental and information science. Computational biology is an interdisciplinary field that uses the techniques of applied mathematics, informatics, statistics and computer science to address biological problems. A computational biologist analyzes, integrates and interprets large-scale biological data, including clinical data, literature and high-throughput genomic and/or proteomic data. Computational biology focuses on hypothesis testing and discovery in the biological domain; the goal is to learn something new about the biology of the system under investigation.

The large size of biological data sets, the inherent complexity of biological problems and the ability to deal with error-prone data all result in large run-time and memory requirements. As such, a powerful computing environment is essential for computational biology, and enables computational biologists to perform many complex, memory intensive tasks, including:

- Gene expression profiling: measuring the activity of tens of thousands of genes at once to create a global picture of cellular function. Statistics and other analytical approaches are used to identify changes between various biological states.
- Functional genomics: integrating genomic and proteomic knowledge to understand the relationship between an organism's genome and its phenotype. Statistics and the vast wealth of data produced by genomic projects are used to describe gene and protein functions and interactions.
- Comparative genomics: establishing a correspondence between genes (i.e. orthology analysis) or other genomic features in different organisms (i.e. an animal disease model and human patients).
- Analysis of protein expression: protein microarrays and high-throughput mass spectrometry can provide a snapshot of the proteins present in a biological sample. These methods involve the problem of matching large amounts of mass data against predicted masses from protein sequence databases, and the complicated statistical analysis of samples where multiple, but incomplete peptides from each protein are detected.
- Modeling of biological systems: involves the use of computer simulations of cellular subsystems (i.e. networks of metabolites and enzymes which comprise metabolism, signal transduction pathways and gene regulatory networks) to both analyze and visualize the complex connections of these cellular processes.

- Integrative biology: integrating multi-dimensional data from a variety of disparate sources, to provide a broader view of a biological system. Uses an integrative approach in which pathways and networks are studied.

As I work to develop the computational infrastructure in my new position, I find I'm also fulfilling the role of a bioinformatician. Although the terms bioinformatics and computational biology are often used interchangeably, *bioinformatics is different than computational biology* and focuses on the creation of tools (i.e. algorithms and specific computational methods) that work on biological data.

A bioinformatician enables the discovery of new biological insights through the creation and improvement of algorithms, databases, computational and statistical techniques and theory to solve formal and practical problems arising from the management and analysis of biological data. Bioinformatics focuses on developing and applying computationally intensive techniques such as pattern recognition, machine learning, data mining and visualization.

For a list of computational biology tools and resources that I've found useful, see the Computational Biology Resources page.

If you're in need of computational support for a research project or require computational analysis to inform and guide a study, **I'm interested in collaborating.** My CV is available here on the site along with a list of publications Feel free to send me a message using the Contact form on the homepage.

Computational biology vs. Bioinformatics

Main article: Bioinformatics

Bioinformatics and computational biology are rooted in life sciences as well as computer and information sciences and technologies. Both of these interdisciplinary approaches draw from specific disciplines such as mathematics, physics, computer science and engineering, biology, and behavioral science. Bioinformatics and computational biology each maintain close interactions with life sciences to realize their full potential. Bioinformatics applies principles of information sciences and technologies to make the vast, diverse, and complex life sciences data more understandable and useful. Computational biology uses mathematical and computational approaches to address theoretical and experimental questions in biology. Although bioinformatics and computational biology are distinct, there is also significant overlap and activity at their interface.[1]

[edit]

BLAST

From Wikipedia, the free encyclopedia

This article is about the bioinformatics software tool. For other uses, see Blast (disambiguation).

BLAST	
Developer(s)	Myers, E., Altschul S.F., Gish W., Miller E.W., Lipman D.J. NCBI
Stable release	2.2.25 / 31 March 2011; 20 days ago
Operating system	UNIX, Linux, Mac, MS-Windows
Type	Bioinformatics tool
License	Public Domain
Website	http://blast.ncbi.nlm.nih.gov/Blast.cgi

In bioinformatics, **B**asic **L**ocal **A**lignment **S**earch **T**ool, or **BLAST**, is an algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences. A BLAST search enables a researcher to compare a query sequence with a library or database of sequences, and identify library sequences that resemble the query sequence above a certain threshold. Different types of BLASTs are available according to the query sequences. For example, following the discovery of a previously unknown gene in the mouse, a scientist will typically perform a BLAST search of the human genome to see if humans carry a similar gene; BLAST will identify sequences in the human genome that resemble the mouse gene based on similarity of sequence. The BLAST program was designed by Eugene Myers, Stephen Altschul, Warren Gish, David J. Lipman, and Webb Miller at the NIH and was published in the *Journal of Molecular Biology* in 1990.[1]

Background

BLAST is one of the most widely used bioinformatics programs,[2] because it addresses a fundamental problem and the algorithm emphasizes speed over sensitivity. This emphasis on

speed is vital to making the algorithm practical on the huge genome databases currently available, although subsequent algorithms can be even faster.

Before fast algorithms such as BLAST and FASTA were developed, doing database searches for the protein or nucleic sequences was very time consuming by using a full alignment procedure like Smith-Waterman.

Indeed, BLAST is faster than Smith-Waterman, however, it cannot "guarantee the optimal alignments of the query and database sequences", as Smith-Waterman does, which "ensured the best performance on accuracy and the most precise results"[3] at the expense of time and computer power.

BLAST is more time efficient than FASTA by searching only for the more significant patterns in the sequences, but with comparative sensitivity. This could be further realized by knowing the algorithm of BLAST introduced below.

Examples of other questions that researchers use BLAST to answer are:

- Which bacterial species have a protein that is related in lineage to a certain protein with known amino-acid sequence?
- Where does a certain sequence of DNA originate?
- What other genes encode proteins that exhibit structures or motifs such as ones that have just been determined?

BLAST is also often used as part of other algorithms that require approximate sequence matching.

The BLAST algorithm and the computer program that implements it were developed by Stephen Altschul, Warren Gish, and David Lipman at the U.S. National Center for Biotechnology Information (NCBI), Webb Miller at the Pennsylvania State University, and Gene Myers at the University of Arizona. It is available on the web on the NCBI website. Alternative implementations include AB-BLAST (formerly known as WU-BLAST), FSA-BLAST (last updated in 2006), and ScalaBLAST.[4]

The original paper by Altschul, *et al.*[1] was the most highly cited paper published in the 1990s.[5]

CS-BLAST

From Wikipedia, the free encyclopedia

CS-BLAST

Developer(s)	Andreas Biegert and Johannes Soeding
Stable release	1.03 / April 14, 2009; 23 months ago
Preview release	1.1 / April 14, 2009; 23 months ago
Written in	C++
Available in	English
Type	Bioinformatics tool
License	Creative Commons Attribution-NonCommercial-3.0
Website	ftp://toolkit.lmb.uni-muenchen.de/csblast/

CS-BLAST (context-specific BLAST), an improved version[1] of BLAST (Basic Local Alignment Search Tool),[2] is a program for proteinsequence searching.

Sequence searches are frequently performed by biologists to infer the function of an unknown protein from its sequence. For this purpose, the protein's sequence is compared to the sequences of millions of other protein sequences in public databases. The functions of the protein can often be inferred from the functions of the most similar sequences found in such a search.[3]

Contents

[hide]

[edit]Implementation

CS-BLAST is implemented as a wrapper around BLAST. It finds up to twice as many distantly related protein sequences as BLAST at the same speed and error rate.[4] This is achieved by considering the

letters that make up protein sequences, the amino acids, in the context of the 12 neighboring amino acids (six on either side). Whereas the probabilities for amino acids to mutate into other types of amino acids in related proteins depend only on the single amino acid in BLAST and other sequence search programs, mutation probabilities in CS-BLAST depend on the local sequence contexts.

An extension of CS-BLAST for iterative search with position-specific scoring matrices is also available. This program (CSI-BLAST) is similar to the popular PSI-BLAST (position-specific iterative BLAST) program.[5] Two search iterations of CS-BLAST are more sensitive than five search iterations of PSI-BLAST.[4].

HMMER

From Wikipedia, the free encyclopedia

HMMER	
Developer(s)	Sean Eddy
Stable release	3.0 / 28 March 2010; 12 months ago
Written in	C
Available in	English
Type	Bioinformatics tool
License	GPL
Website	http://hmmer.janelia.org/

HMMER is a free and commonly used software package for sequence analysis written by Sean Eddy.[1] Its general usage is to identifyhomologous protein or nucleotide sequences. It does this by comparing a profile-HMM to either a single sequence or a database of sequences. Sequences that score significantly better to the profile-HMM compared to a null model are considered to be homologous to the sequences that were used to construct the profile-HMM. Profile-HMMs are constructed from a multiple sequence alignment in the HMMER package using the *hmmbuild* program. The profile-HMM implementation used in the HMMER software was based on the work of Krogh and

colleagues.[2] HMMER is a console utility ported to every major operating system, including different versions of Linux, Windows, and Mac OS.

HMMER is the core utility that protein family databases such as Pfam and InterPro are based upon. Some other bioinformatics tools such asUGENE also use HMMER.

HMMER3 is complete rewrite of the earlier HMMER2 implementation of the package, the aim of HMMER3 is to improve the speed of profile-HMM searches. The main performance gain is due to a heuristic filter that finds high-scoring un-gapped matches within database sequences to a query profile, this heuristic results in a computation time comparable to BLAST with little impact on accuracy. Further gains in performance are due to a log-likelihood model that means that the HMMs no longer need to be calibrated for estimating E-values and that more accurate forward scores can be used for computing the significance of a homologoussequence.[3]

HMMER3 also makes extensive use of vector instructions for increasing computational speed, this work is based upon earlier publication showing a significant acceleration of the Smith-Waterman algorithm for aligning two sequences.[4]

FASTA

From Wikipedia, the free encyclopedia

This article is about the FASTA software package. For the file format, see FASTA format.

FASTA	
Developer(s)	Pearson W.R.
Stable release	35
Operating system	UNIX, Linux, Mac, MS-Windows
Type	Bioinformatics tool
Licence	Free for academic users
Website	fasta.bioch.virginia.edu

FASTA is a DNA and protein sequence alignment software package first described (as FASTP) by David J. Lipman and William R. Pearsonin 1985.[1]

Contents

[hide]

[edit]History

The original FASTP program has designed for protein sequence similarity searching. FASTA added the ability to do DNA:DNA searches, translated protein:DNA searches, and also provided a more sophisticated shuffling program for evaluating statistical significance.[2] There are several programs in this package that allow the alignment of protein sequences and DNA sequences.

[edit]Usage

FASTA is pronounced "fast A", and stands for "FAST-All", because it works with any alphabet, an extension of "FAST-P" (protein) and "FAST-N" (nucleotide) alignment.

The current FASTA package contains programs for protein:protein, DNA:DNA, protein:translated DNA (with frameshifts), and ordered or unordered peptide searches. Recent versions of the FASTA package include special translated search algorithms that correctly handle frameshift errors (which six-frame-translated searches do not handle very well) when comparing nucleotide to protein sequence data.

In addition to rapid heuristic search methods, the FASTA package provides SSEARCH, an implementation of the optimal Smith-Waterman algorithm.

A major focus of the package is the calculation of accurate similarity statistics, so that biologists can judge whether an alignment is likely to have occurred by chance, or whether it can be used to infer homology. The FASTA package is available from fasta.bioch.virginia.edu.

The web-interface to submit sequences for running a search of the European Bioinformatics Institute (EBI)'s online databases is also available using the FASTA programs.

The FASTA file format used as input for this software is now largely used by other sequence database search tools (such as BLAST) and sequence alignment programs (Clustal, T-Coffee, etc).

[edit]Search method

FASTA takes a given nucleotide or amino-acid sequence and searches a corresponding sequence database by using local sequence alignment to find matches of similar database sequences.

The FASTA program follows a largely heuristic method which contributes to the high speed of its execution. It initially observes the pattern of word hits, word-to-word matches of a given length, and marks potential matches before performing a more time-consuming optimized search using a Smith-Waterman type of algorithm.

The size taken for a word, given by the parameter ktup, controls the sensitivity and speed of the program. Increasing the ktup value decreases number of background hits that are found. From the word hits that are returned the program looks for segments that contain a cluster of nearby hits. It then investigates these segments for a possible match.

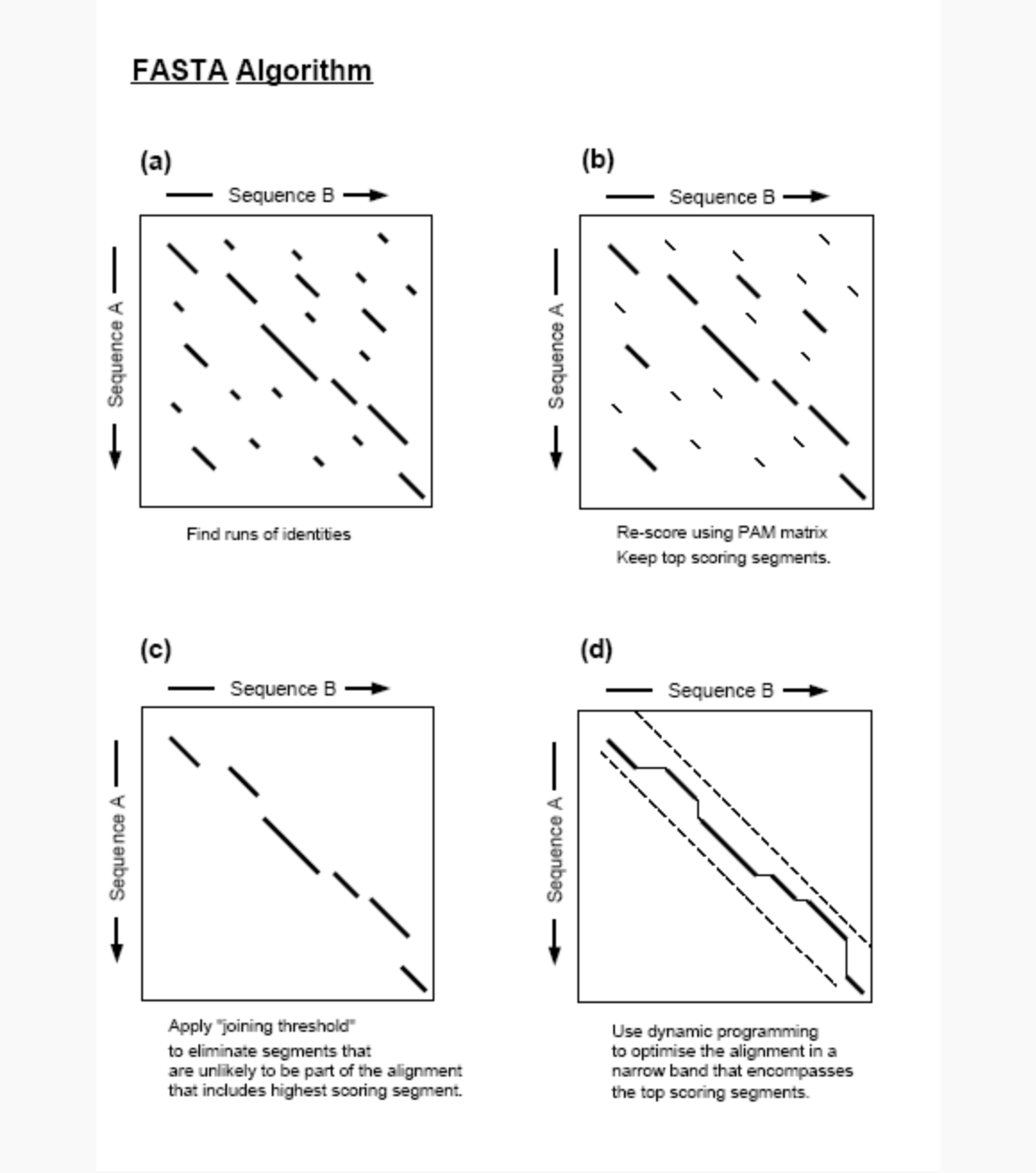


Diagram from Book *Protein Structure prediction - a practical approach* from chapter *Protein Sequence Alignment and Database Scanning*

There are some differences between fastn and fastp relating to the type of sequences used but both use four steps and calculate three scores to describe and format the sequence similarity results.

Position-Specific Iterated BLAST) is a tool that produces a position-specific scoring matrix constructed from a multiple alignment of the top-scoring BLAST responses to a given query sequence. This scoring matrix produces a profile designed to identify the key positions of conserved amino acids within a motif. When a profile is used to search a database, it can often detect subtle relationships between proteins that are

distant structural or functional homologues. These relationships are often not detected by a BLAST search with a sample sequence query.

For an oversimplified example of what a consensus sequence, or profile, looks like, consider that the EF-hand binding loop of the calmodulin family could be represented as follows:

```
Loop Position #          1   3 4  5  6   8       12
Profile                  D x D G D/N G x I x x x E
```

Here "x" stands for positions where there is variability in amino acid type, and therefore, that position is not heavily weighted in the alignment. Comparing the profile to some actual binding loop sequences from different calmodulins is the best way to illustrate the derivation of this profile.

```
POSITION #       1   3 4 5 6   8       12
CALM_HUMAN_1     D K D G D G T I T T K E
CALF_NAEGR_1     D K D G D G T I T T S E
CALM_SCHPO_1     D R D Q D G N I T S N E

CALM_HUMAN_2     D A D G N G T I D F P E
CALF_NAEGR_2     D A D G N G T I D F T E
CALM_SCHPO_2     D A D G N G T I D F T E

CALM_HUMAN_3     D K D G N G Y I S A A E
CALF_NAEGR_3     D K D G N G F I S A Q E
CALM_SCHPO_3     D K D G N G Y I T V E E

CALM_HUMAN_4     D I D G D G Q V N Y E E
CALF_NAEGR_4     D I D G D N Q I N Y T E
CALM_SCHPO_4     D T D G D G V I N Y E E
```

The rules for deriving this simple profile are: 1) any position with 90% amino acid identity or greater is considered conserved in the profile, and thus a higher score would be given when the conserved amino acid is found at that position in the sequence, and 2) any position that always contains one of only two types of amino acids would be up-weighted to give a higher score whenever either of those two amino acids appears at that position. A program such as PSI-BLAST will employ more sophisticated rules to create a profile than this example, of course. It is easy to see even with these sequences that amino acid similarity could be taken into consideration in addition to amino acid identity, and exploited in the profile.

There are three common categories of homologues that are studied in relation to biological molecules, sequence homology, structural homology, and functional homology. Sequence homology is the easiest to identify, and is therefore the primary target of many bioinformatics methods. Sequence homology yields direct implications about the relatedness of proteins and their potential pathways of derivation. However, to help understand how a protein is implicated in a certain disease state, or how to design a pharmaceutical that interacts with a given protein, functional and/or structural information is necessary. Functional homologues are relatively easy to define, as they are any two proteins, or protein domains, that perform similar functions. Structural homologues contain similar "folds", which are localized regions of a molecule that comprise a structural feature such as a "beta barrel" or "four helical bundle" motif. The fold can encompass the entire protein, or just one domain of the protein. A good introduction to the topic of protein folds can be found at the website for the Internet Course on The Principles of Protein Structure organized by **Birkbeck College** (2). When considering sequence, functional, or structural homology, it is important to

understand that one type of homology between proteins does not always infer another type of homology. Nevertheless, it is a reasonable assumption that proteins that are related through evolutionary pathways are likely to have some degree of all three types of homology. PSI-BLAST was engineered to identify distant relationships between sequences that are too subtle to discover with a regular BLAST search.

In the first round, PSI-BLAST is just like a normal BLAST; it finds sequence homologues. In the second round or "iteration" of PSI-BLAST, it figures out which residues tend to be conserved by creating a custom profile for each position of the sequence from a multiple alignment. Then another BLAST is performed, using the profile to produce a position-specific scoring matrix based on which positions evolution has conserved vs. which positions evolution has allowed to vary. The sequences found after the first round are added to the profile, allowing PSI-BLAST to detect more distant homologues in each iteration.

One of the known weaknesses of PSI-BLAST is that its ability to detect distant relationships between proteins is critically dependent on the choice of the query sequence. For this reason, a recommended strategy with PSI-BLAST is to query using individual functional domains. PSI-BLAST will then find other proteins that share this domain, even if they do not possess overall homology. To acquaint the new user with PSI-BLAST, this tutorial mimics an investigation performed by **Aravind and Koonin** (3) in 1999, wherein new members of the HSP70/actin protein family were identified, except the analysis in our tutorial will be on a much smaller scale than that presented in the paper. The HSP70/actin family members were originally recognized to have a common evolutionary origin as a result of a study performed by **Bork et al.** (4) in 1992. A structural superposition of the structures of actin, hexokinase, and the molecular chaperonin hsp70, and alignment of many sequences in each of the three families, uncovered a set of common conserved residues, distributed in five sequence motifs, that are involved in ATP binding and in a flexible interdomain hinge. Although each of these proteins performs very different functions, and their sequences are quite divergent, the similarity in the fold of the ATP-binding domain is visually recognizable. These are all ATP-dependent enzymes and the patterns discovered by Bork and associates could not be detected by traditional BLAST-type sequence searches. Therefore, Aravind and Koonin chose this family as a test of PSI-BLAST's ability to detect distant evolutionary relationships.

Aravind and Koonin chose actin from the PDB file with accession code 1atn as one of their query sequences. Begin the query by retrieving this sequence from the PDB. Check the box for searching the PDB archive that says "PDB ID", then enter the accession code 1atn as the query. Notice that the crystal structure deposited in this entry contained DNase I complexed with actin. There will be a link in the menu in the blue border on the left entitled "FASTA Sequence", select this link to download the sequence file. This file will contain two sequences 1ATN:A (actin) and 1ATN:D (DNase I). Copy the sequence for 1ATN:A and paste it into the BLAST query box that arises from choosing the Protein BLAST, then selecting "PSI-BLAST" under the algorithm section. Change the database from "nr" to "swissprot", but accept the default values for everything else. Click on BLAST, then view the results. NOTE THAT each time another iteration of PSI-BLAST runs, the results page will indicate the iteration number. This is very helpful for keeping track of the stage of the results.

A8

Dynamic programming

From Wikipedia, the free encyclopedia

For the programming paradigm, see Dynamic programming language.

This article **may require cleanup to meet Wikipedia's quality standards.** Please improve this article if you can. The talk pagemay contain suggestions. *(January 2010)*

In mathematics and computer science, **dynamic programming** is a method for solving complex problems by breaking them down into simpler subproblems. It is applicable to problems exhibiting the properties of overlapping subproblems which are only slightly smaller[1] and optimal substructure (described below). When applicable, the method takes far less time than naïve methods.

The key idea behind dynamic programming is quite simple. In general, to solve a given problem, we need to solve different parts of the problem (subproblems), then combine the solutions of the subproblems to reach an overall solution. Often, many of these subproblems are really the same. The dynamic programming approach seeks to solve each subproblem only once, thus reducing the number of computations. This is especially useful when the number of repeating subproblems is exponentially large.

Top-down dynamic programming simply means storing the results of certain calculations, which are later used again since the completed calculation is a sub-problem of a larger calculation. Bottom-up dynamic programming involves formulating a complex calculation as a recursive series of simpler calculations.

Introduction to Dynamic Programming

Dynamic programming is a method for efficiently solving a broad range of search and optimization problems which exhibit the characteristics of overlappling subproblemsand optimal substructure. I'll try to illustrate these characteristics through some simple examples and end with an exercise. Happy coding!

Contents

Overlapping Subproblems

A problem is said to have overlapping subproblems if it can be broken down into subproblems which are reused multiple times. This is closely related to recursion. To see the difference consider the factorial function, defined as follows (in Python):

```
def factorial(n):
    if n == 0: return 1
    return n*factorial(n-1)
```

Thus the problem of calculating `factorial(n)` depends on calculating the subproblem `factorial(n-1)`. This problem
does **not** exhibit *overlapping*subproblems since `factorial` is called exactly once for each positive integer less than n.

Fibonacci Numbers
The problem of calculating the n^{th} Fibonacci number does, however, exhibit overlapping subproblems. The naïve recursive implementation would be

```
def fib(n):
    if n == 0: return 0
    if n == 1: return 1
    return fib(n-1) + fib(n-2)
```

The problem of calculating `fib(n)` thus depends on both `fib(n-1)` and `fib(n-2)`. To see how these subproblems overlap look at how many times fib is called and with what arguments when we try to calculate `fib(5)`:

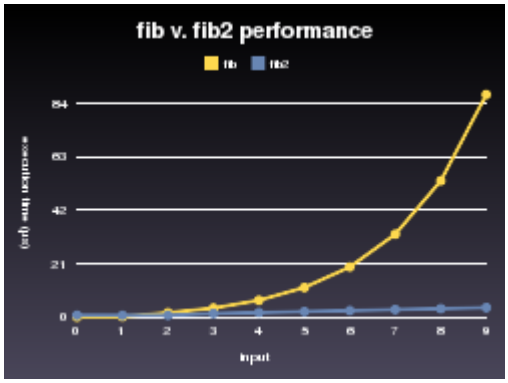
```
fib(5)
fib(4) + fib(3)
fib(3) + fib(2) + fib(2) + fib(1)
fib(2) + fib(1) + fib(1) + fib(0) + fib(1) + fib(0) + fib(1)
fib(1) + fib(0) + fib(1) + fib(1) + fib(0) + fib(1) + fib(0) + fib(1)
```

At the k^{th} stage we only need to know the values of `fib(k-1)` and `fib(k-2)`, but we wind up calling each multiple times. Starting from the bottom and going up we can calculate the numbers we need for the next step, removing the massive redundancy.

```
def fib2(n):
    n2, n1 = 0, 1
    for i in range(n-2):
        n2, n1 = n1, n1 + n2
    return n2+n1
```

In Big-O notation the `fib` function takes $O(c^n)$ time, i.e., exponential in n, while the`fib2` function takes O(n) time. If this is all too abstract take a look at this

graph comparing the runtime (in microseconds) of `fib` and `fib2` versus the input parameter.



The above problem is pretty easy and for most programmers is one of the first examples of the performance issues surrounding recursion versus iteration. In fact, I've seen many instances where the Fibonacci example leads people to believe that recursion is inherently slow. This is not true, but in cases where we can define a problem with overlapping subproblems recursively using the above technique will always reduce the execution time.

Now, for the second characteristic of dynamic programming: optimal substructure.

Optimal Substructure

A problem is said to have optimal substructure if the globally optimal solution can be constructed from locally optimal solutions to subproblems. The general form of problems in which optimal substructure plays a roll goes something like this. Let's say we have a collection of objects called A. For each object o in A we have a "cost," c(o). Now find the subset of A with the maximum (or minimum) cost, perhaps subject to certain constraints.

The brute-force method would be to generate every subset of A, calculate the cost, and then find the maximum (or minimum) among those values. But if A has n elements in it we are looking at a search space of size 2^n if there are no constraints on A. Oftentimes n is huge making a brute-force method computationally infeasible. Let's take a look at an example.

Maximum Subarry Sum

Let's say we're given an array of integers. What (contiguous) subarray has the largest sum? For example, if our array is [1,2,-5,4,7,-2] then the subarray with the largest sum is [4,7] with a sum of 11. One might think at first that this problem reduces to finding the subarray with all positive entries, if one exists, that maximizes the sum. But consider the array [1,5,-3,4,-2,1]. The subarray with the largest sum is [1, 5, -3, 4] with a sum of 7.

First, the brute-force solution. Because of the constraints on the problem, namely that the subsets under consideration are contiguous, we only have to check $O(n^2)$ subarrays (why?). Here it is, in Python:

```
def msum(a):

  return max([(sum(a[j:i]), (j,i)) for i in range(1,len(a)+1) for j inr
ange(i)])
```

This returns both the sum and the offsets of the subarray. Let's see if we can't find an optimal substructure to exploit.

We are given an input array `a`. I'm going to use Python notation so that `a[0:k]` is the subarray starting at 0 and including every element up to and including `k-1`. Let's say we know the subarray of `a[0:i]` with the largest sum (and that sum). Using just this information can we find the subarray of `a[0:i+1]` with the largest sum?

Let a[j:k+1] be the optimal subarray, t the sum of a[j:i], and s the optimal sum. If t+a[i] is greater than s then set a[j:i+1] as the optimal array and set s = t. If t + a[i] is negative, however, the contiguity constraint means that we cannot include a[j:i+1] in our subarray since any such subarray will have a smaller sum than a subarray without it. So, if t+a[i] is negative set t = 0 and set the left-hand bound of the optimal subarray to i+1.

To visualize consider the array [1,2,-5,4,7,-2].

```
Set s = -infinity, t = 0, j = 0, bounds = (0,0)
(1   2  -5   4   7  -2
(1)| 2  -5   4   7  -2  (set t=1.  Since t > s, set s=1 and bounds =
(0,1))
(1   2)|-5   4   7  -2  (set t=3.  Since t > s, set s=3, and bounds =
(0,2))
 1   2  -5(| 4   7  -2  (set t=-2. Since t < 0, set t=0 and j = 3 )
 1   2  -5  (4)| 7  -2  (set t=4.  Since t > s, set s=4 and bounds =
(3,4))
```

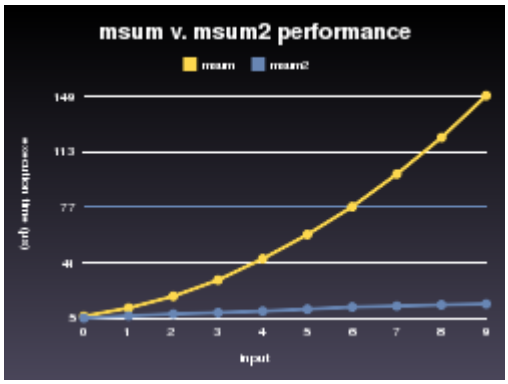
```
 1   2  -5  (4   7)|-2  (set t=11. Since t > s, set s=11 and bounds =
(3,5))
 1   2  -5  (4   7) -2| (set t=9.  Nothing happens since t < s)
```

This requires only one pass through the array and at each step we're only keeping track of three variables: the current sum from the left-hand edge of the bounds to the current point (t), the maximal sum (s), and the bounds of the current optimal subarray (bounds). In Python:

```
def msum2(a):
    bounds, s, t, j = (0,0), -float('infinity'), 0, 0

    for i in range(len(a)):
        t = t + a[i]
        if t > s: bounds, s = (j, i+1), t
        if t < 0: t, j = 0, i+1
    return (s, bounds)
```

In this problem the "globally optimal" solution corresponds to a subarray with a globally maximal sum, but at each each step we only make a decision relative to what we have already seen. That is, at each step we know the best solution *thus far*, but might change our decision later based on our previous information and the current information. This is the sense in the problem has optimal substructure. Because we can make decisions locally we only need to traverse the list once, reducing the run-time of the solution to O(n) from O(n^2). Again, a graph:



The Knapsack Problem

Let's apply what we're learned so far to a slightly more interesting problem. You are an art thief who has found a way to break into the impressionist wing at the Art Institute of Chicago. Obviously you can't take everything. In particular, you're constrained to take only what your knapsack can hold — let's say it can only hold W pounds. You also know the market value for each painting. Given that you can

only carry W pounds what paintings should you steal in order to maximize your profit?

First let's see how this problem exhibits both overlapping subproblems and optimal substructure. Say there are n paintings with weights w_1, ..., w_n and market values v_1, ..., v_n. Define A(i,j) as the maximum value that can be attained from considering only the first i items weighting at most j pounds as follows.

Obviously A(0,j) = 0 and A(i,0) = 0 for any i ≤ n and j ≤ W. If w_i > j then A(i,j) = A(i-1, j) since we cannot include the i^{th} item. If, however, w_i ≤ j then A(i,j) then we have a choice: include the i^{th} item or not. If we do not include it then the value will be A(i-1, j). If we do include it, however, the value will be v_i + A(i-1, j - w_i). Which choice should we make? Well, whichever is larger, i.e., the maximum of the two.

Expressed formally we have the following recursive definition

$$A(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ A(i-1,j) & \text{if } w_i > j \\ \max\{A(i-1,j), v_i + A(i-1, j-w_i)\} & \text{if } w_i \leq j \end{cases}$$

This problem exhibits both overlapping subproblems and optimal substructure and is therefore a good candidate for dynamic programming. The subproblems overlap because at any stage (i,j) we might need to calculate A(k,l) for several k < i and l < j. We have optimal substructure since at any point we only need information about the choices we have already made.

The recursive solution is not hard to write:

```
def A(w, v, i,j):
    if i == 0 or j == 0: return 0
    if w[i-1] > j:  return A(w, v, i-1, j)
    if w[i-1] <= j: return max(A(w,v, i-1, j), v[i-1] + A(w,v, i-1, j -
w[i-1]))
```

Remember we need to calculate A(n,W). To do so we're going to need to create an n-by-W table whose entry at (i,j) contains the value of A(i,j). The first time we calculate the value of A(i,j) we store it in the table at the appropriate location. This technique is called memoization and is one way to exploit overlapping subproblems. There's also a Ruby module called memoize which does it for Ruby.

To exploit the optimal substructure we iterate over all i <= n and j <= W, at each step applying the recursion formula to generate the A(i,j) entry by using the memoized table rather than calling A() again. This gives an algorithm which takes O(nW) time using O(nW) space and our desired result is stored in the A(n,W) entry in the table.

Everyday Dynamic Programming

The above examples might make dynamic programming look like a technique which only applies to a narrow range of problems, but many algorithms from a wide range of fields use dynamic programming. Here's a very partial list.

1. The Needleman-Wunsch algorithm, used in bioinformatics.
2. The CYK algorithm which is used the theory of formal languages and natural language processing.
3. The Viterbi algorithm used in relation to hidden Markov models.
4. Finding the string-edit distance between two strings, useful in writing spellcheckers.
5. The D/L method used in the sport of cricket.

That's all for today. Cheers!

A9

Smith–Waterman algorithm

From Wikipedia, the free encyclopedia

This article **needs references that appear in reliable third-party publications**. Primary sources or sources affiliated with the subject are generally not sufficient for a Wikipedia article. Please add more appropriate citations from reliable sources. *(August 2007)*

The **Smith–Waterman algorithm** is a well-known algorithm for performing **local sequence alignment**; that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

Algorithm Explanation

A matrix H is built as follows:

$$H(i,0) = 0,\ 0 \leq i \leq m$$

$$H(0,j) = 0,\ 0 \leq j \leq n$$

if $a_i = b_j$ $w(a_i,b_j) = w(\text{match})$ or if $a_i! = b_j$ $w(a_i,b_j) = w(\text{mismatch})$

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + & w(a_i,b_j) & \text{Match/Mismatch} \\ H(i-1,j) + & w(a_i,-) & \text{Deletion} \\ H(i,j-1) + & w(-,b_j) & \text{Insertion} \end{Bmatrix},\ 1 \leq i \leq m, 1 \leq$$

Where:

- a,b = Strings over the Alphabet Σ
- $m = \text{length}(a)$
- $n = \text{length}(b)$
- $H(i,j)$ - is the maximum Similarity-Score between a suffix of a[1...i] and a suffix of b[1...j]
- $w(c,d),\ c,d \in \Sigma \cup \{'-'\}$, '-' is the gap-scoring scheme

Example

- Sequence 1 = ACACACTA
- Sequence 2 = AGCACACA

- w(match) = +2
- $w(a,-) = w(-,b) = w(mismatch) = -1$

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix}$$

To obtain the optimum local alignment, we start with the highest value in the matrix (i,j). Then, we go backwards to one of positions (i-1,j), (i,j-1), and (i-1,j-1) depending on the direction of movement used to construct the matrix. We keep the process until we reach a matrix cell with zero value, or the value in position (0,0).

In the example, the highest value corresponds to the cell in position (8,8). The walk back corresponds to (8,8), (7,7), (7,6), (6,5), (5,4), (4,3), (3,2), (2,1), (1,1), and (0,0),

Once we've finished, we reconstruct the alignment as follows: Starting with the last value, we reach (i,j) using the previously-calculated path. A diagonal jump implies there is an alignment (either a match or a mismatch). A top-down jump implies there is a deletion. A left-right jump implies there is an insertion.

For the example, we get:

```
Sequence 1 = A-CACACTA
Sequence 2 = AGCACAC-A
```

Needleman–Wunsch algorithm

From Wikipedia, the free encyclopedia

The **Needleman–Wunsch algorithm** performs a global alignment on two sequences (called *A* and *B* here). It is commonly used in bioinformatics to align protein or nucleotidesequences. The algorithm was published in 1970 by Saul B. Needleman and Christian D. Wunsch.[1]

The Needleman–Wunsch algorithm is an example of dynamic programming, and was the first application of dynamic programming to biological sequence comparison.

A10

BLAST Search main parameters

HISTOGRAM
: Display a histogram of scores for each search; default is yes. (See parameter H in the BLAST Manual).

DESCRIPTIONS
: Restricts the number of short descriptions of matching sequences reported to the number specified; default limit is 100 descriptions. See also EXPECT.

ALIGNMENTS

Restricts database sequences to the number specified for which high-scoring segment pairs (HSPs) are reported; the default limit is 100. If more database sequences than this happen to satisfy the statistical significance threshold for reporting (see EXPECT below), only the matches ascribed the greatest statistical significance are reported.

EXPECT

The statistical significance threshold for reporting matches against database sequences; the default value is 10, such that 10 matches are expected to be found merely by chance, according to the stochastic model of Karlin and Altschul (1990). If the EXPECT value of a match is greater than the EXPECT threshold, the match will not be reported. Lower EXPECT thresholds are more stringent, leading to fewer chance matches being reported. Fractional values are acceptable.

CUTOFF

Cutoff score for reporting high-scoring segment pairs. The default value is calculated from the EXPECT value (see above). HSPs are reported for a database sequence only if the statistical significance ascribed to them is at least as high as would be ascribed to a lone HSP having a score equal to the CUTOFF value. Higher CUTOFF values are more stringent, leading to fewer chance matches being reported. (See parameter S in the BLAST Manual). Typically, significance thresholds can be more intuitively managed using EXPECT.

MATRIX

Specify an alternate scoring matrix for BLASTP, BLASTX, TBLASTN and TBLASTX. The default matrix is BLOSUM62 (Henikoff & Henikoff, 1992). The valid alternative choices include: PAM40, PAM120, PAM250 and IDENTITY. No alternate scoring matrices are available for BLASTN; specifying the MATRIX directive in BLASTN requests returns an error response.

STRAND

Restrict a TBLASTN search to just the top or bottom strand of the database sequences; or restrict a BLASTN, BLASTX or TBLASTX search to just reading frames on the top or bottom strand of the query sequence.

INCLUSION THRESHOLD

The statistical significance threshold for including a sequence in the model used by PSI-BLAST on the next iteration.

ORGANISM NAME

Enter the organism name in the form "Genus species" (e.g., "Homo sapiens"). A number of popular organism names are listed on a pull-down menu.

TAXONOMIC CLASSIFICATION

Enter any taxonomic group from the NCBI taxonomy (e.g. "Mammalia").

Some popular groups are:

```
Archaea
Bacteria
Eukaryota
Embryophyta (higher plants)
Fungi
Metazoa (multicellular animals)
Vertebrata
Mammalia
Rodentia
Primates
```

Explore the taxonomy database at NCBI

FILTER (Low-complexity)

Mask off segments of the query sequence that have low compositional complexity, as determined by the SEG program of Wootton & Federhen (Computers and Chemistry, 1993) or, for BLASTN, by the DUST program of Tatusov and Lipman (in preparation). Filtering can eliminate statistically significant but biologically uninteresting reports from the blast output (e.g., hits against common acidic-, basic- or proline-rich regions), leaving the more biologically interesting regions of the query sequence available for specific matching against database sequences.

Filtering is only applied to the query sequence (or its translation products), not to database sequences. Default filtering is DUST for BLASTN, SEG for other programs.

It is not unusual for nothing at all to be masked by SEG, when applied to sequences in SWISS-PROT, so filtering should not be expected to always yield an effect. Furthermore, in some cases, sequences are masked in their entirety, indicating that the statistical significance of any matches reported against the unfiltered query sequence should be suspect.

FILTER (Human repeats)

This option masks Human repeats (LINE's and SINE's) and is especially useful for human sequences that may contain these repeats. This option is still experimental and under development, so it may change in the near future.

FILTER (Mask for lookup table only)

This option masks only for purposes of constructing the lookup table used by BLAST. The BLAST extensions are performed without

masking. This option is still experimental and may change in the near future.

NCBI-gi

Causes NCBI gi identifiers to be shown in the output, in addition to the accession and/or locus name.

Query Genetic Code

Genetic code to be used in blastx translation of the query.

Graphical Overview

An overview of the database sequences aligned to the query sequence is shown. The score of each alignment is indicated by one of five different colors, which divides the range of scores into five groups. Multiple alignments on the same database sequence are connected by a striped line. Mousing over a hit sequence causes the definition and score to be shown in the window at the top, clicking on a hit sequence takes the user to the associated alignments.