TECHVIEW: GENOMICS

The Babel of Bioinformatics

Teresa K. Attwood

The sequencing of entire genomes is a major achievement, but the meaning of the mass of accumulated data is only just beginning to be unraveled. At first sight, the task appears straightforward: locate the genes and translate the coding regions to establish their protein products; perform similarity searches to establish relationships with previously characterized sequences and assign function by evolutionary inference; and rationalize the function in structural terms using known or model-derived structures. Given the quantity of data, the procedures should be automated as much as possible.

The reality, of course, is not so simple. Attempts to decipher the clues latent in genomic data are hampered because current methods to predict genes in uncharacterized DNA are unreliable (and it is not always clear what we mean by "gene"); it is presumptuous to make functional assignments merely on the basis of some degree of similarity between sequences (and it is not always clear what we mean by "function"); very few structures are known compared with the number of sequences, and structure prediction methods are unreliable (and knowing structure does not inherently tell us function); the degree of automation that has been used of necessity, with its imperfect tools and protocols, has led to the accumulation of much database misinformation; and the terminology has been imprecise, muddying perceptions of what can realistically be achieved. Given these problems, what is the state of the art in sequence-structure-function bioinformatics?

Gene prediction

Information used to predict genes includes signals in the sequence, content statistics, and similarity to known genes. In a recent test of gene detection tools on part of the *Drosophila* genome, the majority of these "gene finders" identified 95% of coding nucleotides, but intron/exon structures were correctly predicted for only about 40% of genes. The different methods failed to find between 5% and 95% of genes, and incorrectly identified up to 55% (*1*). But probably the most sobering evidence of the frailty of gene prediction methods is the uncertainty in the number of genes in the human genome, with current estimates ranging from 27,462 to 312,278. The methods used to arrive at these numbers each involve different approximations and extrapolations. Nevertheless, it is disturbing that the different analytical approaches should yield such disparate results.

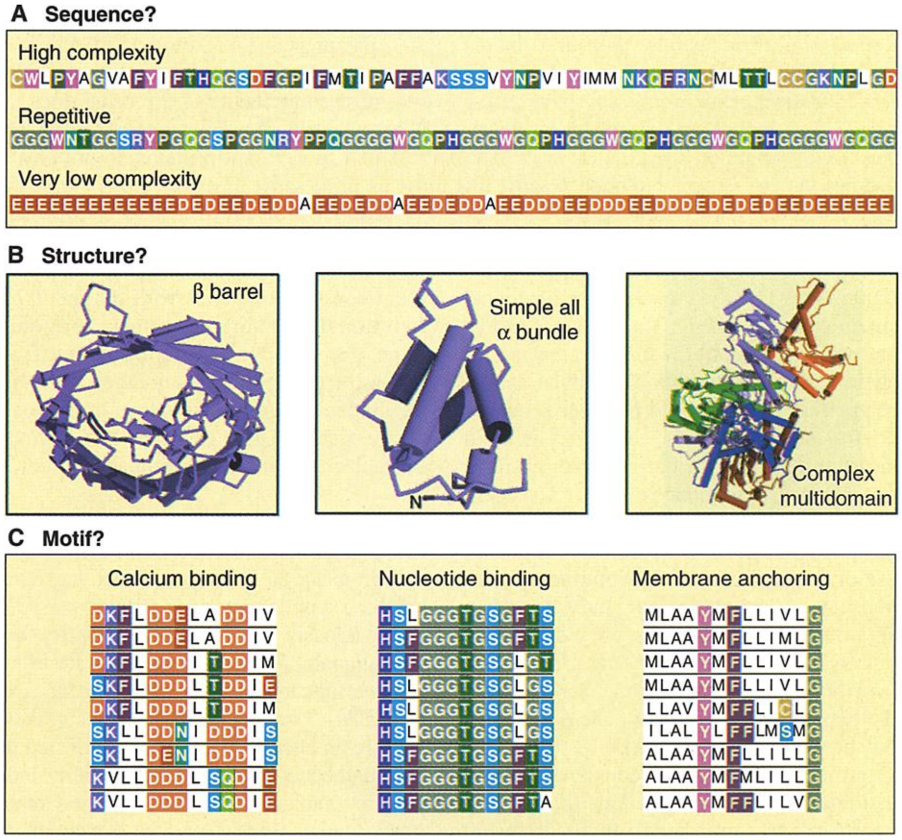
What is a gene?

Perhaps the biggest obstacle to accurate gene counting is that even the definition of a gene is unclear. Is it a heritable unit corresponding to an observable phenotype? Or is it a packet of genetic information that encodes a protein, or proteins? Or perhaps one that encodes RNA? Must it be translated? Are genes genes if they are not expressed? As definitions vary, inevitably so do estimates of the total number of genes in sequenced genomes.

The sequence-structure imbalance

To date, more than 540,000 protein sequences have been deposited in the nonredundant database maintained by the National Center for Biotechnology Information (NCBI), and millions of expressed sequence tags (ESTs), which are partial sequences of clones that are often error prone, are housed in public and proprietary repositories. These numbers will snowball with the fruition of further genome projects. By contrast, the number of unique protein structures is still less than 2000. Of course, we do not know how many unique sequences there are; nevertheless, it is clear that there is a dearth of structural information.

Given this sequence-structure imbalance, it is imperative that we focus on deciphering the structural, functional, and evolutionary clues encoded in the language of biological sequences. Two distinct analytical approaches have emerged. Pattern recognition methods aim to detect similarity between sequences and structures and infer related functions. Thus, they require some characteristic to have been observed and deposited in a reference database. In contrast, ab initio



Levels of complexity. Looking at a sequence (**A**) or a fold (**B**) in isolation, we can say little about its function. Only when we look at sequences or structures together do the patterns of conservation that emerge (motifs) begin to provide functional clues. For example, the above motifs (**C**) may suggest roles in calcium binding, nucleotide binding, and membrane anchoring. We can think of folds as providing different scaffolds, which can be decorated in different ways by different sequences to confer different functions. Knowing both the fold and the function allows us to rationalize the mechanism by which the structure effects its function at the molecular level.

The author is at the School of Biological Sciences, University of Manchester, Manchester, UK. E-mail: attwood@bioinf.man.ac.uk

prediction methods deduce structure directly from sequence. The approaches are quite different and should not be confused. Their levels of success also differ markedly.

Function prediction through pattern recognition

Tools for similarity searching are standard components of the sequence annotator's armory. Sequence similarity programs may seek pairwise similarities in large sequence repositories or search for conserved patterns in gene family databases (*2–5*). Gene family databases allow more specific functional diagnoses to be made than is possible by pairwise searching. They are based on the principle that related sequences can be aligned to find regions (motifs) that show little variation. These motifs usually reflect some vital structural or functional role (see the figure), and they can be used to derive diagnostic family signatures. Sequences can then be searched against databases of such signatures to see whether they can be assigned to known families. Gene family databases have recently been integrated to create a unified protein family resource (*6*), facilitating the inference of function by identifying homologous relationships.

The term "homology," a fundamental concept in bioinformatics, is often used incorrectly. Sequences are homologous if they are related by divergence from a common ancestor (*7*). Conversely, analogy relates to the acquisition of common structural or functional features via convergent evolution from unrelated ancestors. For example, β barrels occur in soluble serine proteases and integral membrane porins, but despite their common architecture, they share no sequence or functional similarity. Similarly, the enzymes chymotrypsin and subtilisin share groups of catalytic residues with almost identical spatial geometries, but they have no other sequence or structural similarities. Homology is not a measure of similarity, but rather an absolute statement that sequences have a divergent rather than a convergent relationship. This is not just a semantic issue because imprecise use of the term obscures evolutionary relationships. In comparing structures, the same arguments apply. Structures may be similar, but common evolutionary origin remains a hypothesis until supported by other evidence; the hypothesis may be correct or mistaken, but the similarity is a fact (*8*).

Among homologous sequences, we can distinguish orthologs (proteins that usually perform the same function in different species) and paralogs (proteins that perform different but related functions within one organism). Orthologs allow investigation of cross-species relationships, whereas paralogs, which arise via gene duplication events, shed light on underlying evolutionary mechanisms because the duplicated genes follow separate evolutionary pathways and new specificities evolve through variation and adaptation. Such complexity presents real challenges for bioinformatics. When analyzing a database search, it may be unclear how much functional annotation can be legitimately inherited by a query sequence, and whether the best match turned up by the search is the true ortholog or a paralog. This difficulty is the source of numerous annotation errors.

Further complications result from the domain and/or modular nature of many proteins. Modules are autonomous folding units that often function as protein building blocks, forming multiple combinations of the same module or mosaics of different modules. They can confer a variety of functions on the parent protein. If the best hit in a database search is a match to a single domain or module, it is unlikely that the function annotation can be propagated from the parent protein to the query sequence.

In using modules to confer different functionalities, Nature uses old material to create new systems. The complexity of such systems poses important problems for computational approaches because the properties of a system can be explained by but not deduced from those of its components (*9*, *10*). The presence of a module tells little of the function of the complete system; knowing most components of a mosaic does not allow us easily to predict a missing one, and modules in different proteins do not always perform the same function.

Many other factors also complicate function assignment: gene functions may be redundant, nonorthologous displacement can replace genes with unrelated but functionally analogous genes, horizontal gene transfer can introduce genes from different phylogenetic lineages, and lineage-specific gene loss can eliminate ancestral genes. Thus, genomes harbor many obstacles to reliable function assignment.

What is function?

Protein function is context-dependent. Vagueness in using the term has yielded confusing database annotations. It is currently used to refer variously to biochemical activities, biological goals, and cellular structure; for example, the function of actin might be described as "ATPase" or "constituent of the cytoskeleton." In an attempt to introduce rigor into the field and better reflect biological reality, independent ontologies such as the Gene Ontology (*11*) are under development that aim to define more explicitly the relationships between gene products and biological processes, molecular functions, and cellular components.

Structure prediction and fold recognition

We have seen that definitions of "genes" differ, making it difficult to count genes accurately, and that our concepts of "function" differ, making function assignment tricky. It would seem, however, that we can agree on what structures are. They are tangible, measurable things, so should we not be able to predict them reliably?

Structure prediction methods range from computationally intensive strategies that simulate the physical and chemical forces involved in protein folding to knowledge-based approaches that use information from structure databases to build models. Yet the problem of predicting protein structure remains unsolved: knowledge-based techniques typically produce low-resolution models, and no current method yields reliable predictions for remote homologs (*12*). For small proteins, ab initio methods generate models with substantial segments that resemble the correct fold, but results deteriorate beyond ~100 residues. Today, knowledge-based methods, especially those that combine information from different approaches, give best results (*13*). The most successful modeling and fold recognition studies have balanced better algorithms with appropriate levels of manual intervention (*14*).

Prediction methods do not work well because we do not fully understand how the primary structure of a protein determines its tertiary structure. Structural genomics projects will gradually lessen our reliance on prediction, because they aim to provide experimental structures or models for every protein in all completed genomes (although membrane protein structures will be difficult to obtain because they are difficult to crystallize). We must keep in mind, however, that structure alone will not inherently tell us function (see the figure). For example, determining the structure of a hypothetical protein and discovering that it binds ATP (*15*) may shed light on possible aspects of its functionality, but such information does not reveal its specific biological function.

What is structure?

In the context of fold recognition and prediction, it is important to be precise about what we mean by "structure." For example, is a prediction a "good" prediction if it correctly reproduces all atomic positions, the topology (connectivity of secondary structures), the architecture (gross arrangement of secondary structures), or merely the structural class (mainly α, mainly β, etc.)? Where does a "reasonably good" prediction fall in this hierarchy, and what level of structural detail does a "tough near miss" (*16*) reveal? Using such imprecise words hinders comprehension, making it difficult to evaluate what a good prediction really is.

Outlook

In "predicting" genes, protein functions, and structures, it is helpful to define our terms precisely and be honest about our achievements. Otherwise, we will continue to be baffled by paradoxical new prediction methods that yield >80% error rates. Gene identification, structure prediction, and functional inference are nontrivial computational tasks, but with the relentless accumulation of sequence data, improvements continue to be made in all areas.

Nature functions by integration, and the adoption of a more holistic view of complex biological systems is an essential next step for bioinformatics. To get the most from genomic data, we need to take account of information on the regulation of gene expression, metabolic pathways, and signaling cascades. Proteins do not work in isolation but are involved in interrelated networks. Unraveling these networks and their interactions will be vital to our understanding of normal and pathologic cell development, and will help us create an integrated mapping between genotype and phenotype.

Genomics-based drug discovery is heavily dependent on accurate functional annotation. Toward this end, bioinformatics will need to deliver highly integrated, interoperable databases (and data "warehouses") that allow the user to reason over disparate data sources and ultimately enable knowledge-based inference and innovation. The more genome annotation is automated, the greater will be the need for collaboration between software developers, annotators, and experimentalists. And the more data we have to handle, the more rigorous we must be in our thinking (and writing) if we are to make sense of the complexities. Sequence-structure-function bioinformatics does not yet yield all the answers, but a future holistic approach should help fuse today's glimmerings of knowledge into a new dawn of understanding.

References and Notes

1. M. G. Reese *et al.*, *Genome Res.* **10**, 483 (2000).
2. K. Hofmann *et al.*, *Nucleic Acids Res.* **27**, 215 (1999).
3. T. K. Attwood *et al.*, *Nucleic Acids Res.* **28**, 225 (2000).
4. A. Bateman *et al.*, *Nucleic Acids Res.* **28**, 263 (2000).
5. J. Henikoff *et al.*, *Nucleic Acids Res.* **28**, 228 (2000).
6. R. Apweiler *et al.*, *Bioinformatics*, in press.
7. W. M. Fitch, *Syst. Zool.* **19**, 99 (1970).
8. G. R. Reeck *et al.*, *Cell* **50**, 667 (1987).
9. F. Jacob, *Science* **196**, 1161 (1977).
10. L. Gold *et al.*, *Curr. Opin. Genet. Dev.* **7**, 848 (1997).
11. M. Ashburner *et al.*, *Nature Genet.* **25**, 25 (2000).
12. B. Rost and S. O'Donoghue, *Comput. Appl. Biosci.* **13**, 345 (1997).
13. A. R. Panchenko *et al.*, *J. Mol. Biol.* **296**, 1391 (2000).
14. M. J. E. Sternberg *et al.*, *Curr. Opin. Struc. Biol.* **9**, 368 (1999).
15. T. I. Zarembinski *et al.*, *Proc. Natl. Acad. Sci. U.S.A.* **95**, 15189 (1998).
16. K. A. Olszewski *et al.*, *Comput. Chem.* **24**, 499 (2000).
17. Single-letter abbreviations for the amino acid residues are as follows: A, Ala; C, Cys; D, Asp; E, Glu; F, Phe; G, Gly; H, His; I, Ile; K, Lys; L, Leu; M, Met; N, Asn; P, Pro; Q, Gln; R, Arg; S, Ser; T, Thr; V, Val; W, Trp; and Y, Tyr.

TECHSIGHTING

SOFTWARE

Conquering by Dividing

The average personal computer spends much less than half a day actually performing useful computations. Many users, concerned about the vulnerability of expensive electronic components to the constant cycling of the power on and off, leave their systems on continuously. It is staggering to imagine the enormous, unused computing resources of several million PCs left running unattended. One popular approach to tapping this computing power is the Search for Extra-Terrestrial Intelligence (SETI) project (*1*), which breaks giant computing problems into pieces that can be solved on personal computers in their spare time.

Popular Power Worker
Popular Power, Inc.
San Francisco, CA
Free
www.popularpower.com

Popular Power, Inc. is a company offering a new twist on this theme. Like SETI, a company computer feeds pieces of large computing problems to networked personal computers via their software program, Popular Power Worker, for idle-time operation. Popular Power's approach differs, however, in providing a variety of computing problems to work on. These include nonprofit projects with no financial incentive to the personal computer owner, as well as commercial jobs that will eventually pay users for tasks performed on their machines.

The current version of the Popular Power Worker runs only on Windows and Linux systems and is officially in pre-release form. The preliminary status of the software is readily apparent; numerous bugs, frequent crashes, and difficulties in installation plague the program currently. If information at the company Web site is accurate, personal computer owners interested in Popular Power's computing model may find dealing with the problems of the early release worth their while. Users of the pre-release software are promised priority of access to commercial computing jobs after the official version is released. Popular Power Worker can be downloaded for free from the company's Web site, and it installs as a screen saver, which starts the program running when it becomes active. Future versions of the program for Macintosh and Solaris systems are planned.

The benefits of the Popular Power scheme for distributed computing tasks do not accrue solely to the user whose computer is used. The flexible nature of Popular Power's design provides access for businesses, scientists, and anyone with massive computing projects to computing power that is potentially far greater than they would gain from a fixed piece of hardware. Personal computer users might be able to select which commercial job to run through Popular Power Worker depending on the return offered by the originating contractor. A key to the success of the computing model is likely to be the price Popular Power demands for acting as the interface between the computing project creators and the personal computer users.

In summary, the current version of Popular Power Worker is still in the testing phase and users may find the software unstable. Tech-savvy personal computer enthusiasts are best suited to test the current pre-release product. The remaining users are advised to wait at least for the official release of the software.

—Kevin Ahern

Department of Biochemistry and Biophysics, Oregon State University, Corvallis, OR 97331, USA. E-mail: ahernk@ucs.orst.edu

References

1. J. Kaiser, *Science* **282**, 839 (1998).

Tech.Sight is published in the third issue of each month. Contributing editor: Kevin Ahern, Department of Biochemistry and Biophysics, Oregon State University. Send your comments by e-mail to techsight@aaas.org

TECHSIGHTING

SOFTWARE

Eyes on the Skies

The orbital space above Earth contains an astonishing collection of man-made satellites. Tracking all of these objects is no small task. Liftoff is a NASA Web site that provides several software tools to locate, track, and identify Earth-orbiting satellites. At the Web site, three programs are available: J-Pass (identifies satellites passing overhead); J-Track (allows one to track orbiting objects); and J-Track 3D (allows one to view satellites orbiting Earth from a perspective far away in space). Each of these platform-independent applications is written in Java and is accessible from both Internet Explorer and Netscape

J-Track, J-Track 3D, and J-Pass
NASA
Free
http://liftoff.msfc.nasa.gov/realtime/JTrack/Spacecraft.html