Humans, computers, and the route to biological insights: Regaining our capacity for surprise

Jeff Elhai
Center for the Study of Biological Complexity, Virginia Commonwealth University, Richmond, VA, U.S.A

There's a person – a friend of mine – a microbiologist… I don't want to give his name, but I'm sure you know him. He's a really creative guy. He's able to look at problems that have stumped everyone else, important problems, and while the rest of us are plugging away, following general principles, he knows his system so well, the things that aren't in books, that it's like the system is speaking to him, and he can see what the critical principles are, in the same way that you'd recognize the voice of a friend.

But here's what I want to tell you: He doesn't work in the lab. Of course that's not remarkable. LOTS of researchers don't spend much time in the lab. But this guy, he's NEVER worked in the lab. He wouldn't know a pipette from a pipe cleaner. I wondered about this and asked him straight away what the story was. He told me he's always avoided going into the lab. It's too confusing, he said. He doesn't have the mind for that sort of thing. He's a biologist, not an engineer!

That was difficult for me to believe. You see, I've read his papers… masterful experiments! Who did them? I asked. He told me that he made a point of hiring someone who was a superb technician. Doesn't know a thing about biology, but that's OK. My friend tells him exactly what to do in the lab, and he does it.

It seemed rather strange to me, someone who knows nothing about experimental science directing someone who knows nothing about the subject of study. So I asked him, what inspired him to adopt this strange arrangement? He told me about his grandfather. His grandfather was an explorer. Discovered the South Pole. What inspired him was the fact that his grandfather was actually blind. He had a map in Braille and specially trained huskies. He'd point to the spot he wanted to go to, the dogs would sniff the map, and off they'd go!

But what if the map was wrong? …Wait, never mind that, wasn't he an <u>explorer</u>? Wasn't he supposed to look around and explore? No, my friend said. He was an adventurer, not a tourist! What made him successful was his single-minded determination. He wanted to go to the South Pole and he got there!

He had other talents too. He invented a remarkable machine for his wife, my friend's grandmother. She was a musicologist. His machine allowed her to feed in any musical manuscript, and it would tell her what composer's work it was most similar to.

"That's a remarkable machine!" I remarked. "But how could she tell if the machine was telling her the truth? "

"Well," my friend answered. "The answer came out with E-values. The lower the E-value…"

"…I know what E-values are, but that doesn't matter. Of course she could play the music herself to check up on the machine…

"Well, no she couldn't do that." My friend said. "She didn't know how to play any musical instrument. She was a musicologist, not a street musician! And anyway she'd been deaf since birth."

- - - - - -

…OK, I think I might have lost you with that one. You don't believe that there could be a musicologist who relies totally on a machine to understand music. Perhaps you also don't believe that there could be a South Pole explorer who can see nothing of the world around him. I get your point, but actually, it's the first case that I find the most unbelievable: an experimental researcher who is blind to the means by which experimental information is obtained and reliant on a black box for addressing questions of nature. Perhaps that's not so unusual. The unbelievable part is that I said this guy was creative, able to sense from a deep connection with his experimental system directions others couldn't see. How could he find the unexpected if the mistakes, the accidents, the intermediate results, were all hidden from him? When the world happens to be as he imagines, he may churn out the predicted findings, but how could he perceive the more interesting part, the world that he cannot imagine?

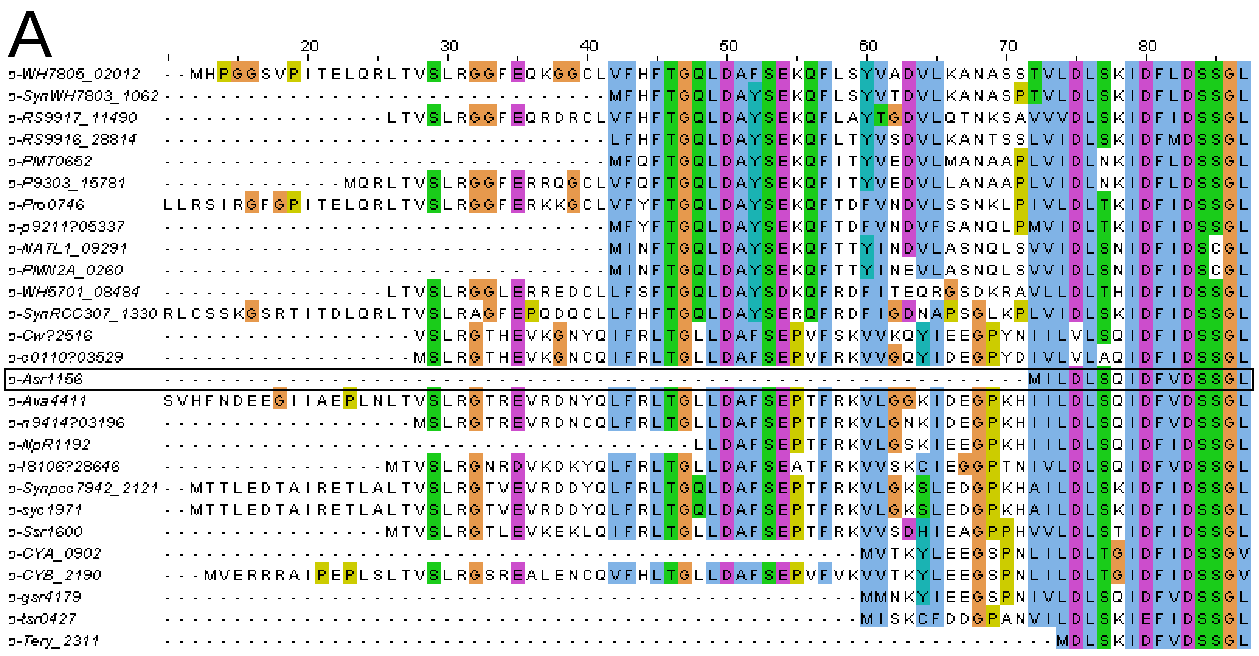
Let's examine a situation that may seem more familiar.

Scenario: Nagging doubts…
Suppose that your experiments have focused your attention momentarily on a protein, Asr1156 (GenBank NP_485199), from the cyanobacterium Anabaena PCC 7120, that seems to be important in differentiation. What could this protein do? You go to your favorite web site and look up the annotation of the gene: "Hypothetical", you're told. Oh, well. No help there. Back to your experiments.

Scenario (Part 2): …increased by a provocative Blast…
Suppose you go a step further, extracting the sequence of the predicted protein (perhaps this takes some irritating fiddling with the web site) and submitting it to Blast. Are there any similar proteins known? The result comes back: the protein is similar to the C-terminus of proteins from a large number of cyanobacteria. The protein is usually annotated as "an anti-sigma factor antagonist", a type of protein that might indeed be interesting with respect to differentiation! But you're concerned: Is the similarity incidental, perhaps just a conserved motif, or does it provide a sufficient basis to claim that your protein has the same activity as the rest? One web site says "hypothetical" and another suggests "anti-sigma factor antagonist"... how to decide?

Scenario (Part 3): … inflamed by a remarkable alignment…
On most days, that difficult question would remain unanswered, but suppose that today you expend the effort necessary to find the complete sequences of the similar proteins, download them, find a sequence alignment program, combine the sequences in a form acceptable to the program, and finally perform the alignment (Fig. 1A). It seems very peculiar that many proteins from organisms you know to be quite distant from each other have regions of very high similarity while other proteins appear truncated. For example, Anabaena variabilis (whose protein is just below Asr1156 in Fig. 1) and Prochlorococcus marinus MIT9303 (whose protein is sixth from the top) are as distant from each other as any two cyanobacteria can be, and yet their proteins have conserved residues not found in the truncated proteins of more closely related cyanobacteria. This strikes you as highly improbable and forces you to consider an alternative: Treachery! Perhaps the protein sequences are wrong! Perhaps what purports to be the start codons for the short open reading frames actually lie within the genes. Unfortunately, if you can't trust the information you're given, then what do you have to go on?

Upon further reflection, you see an answer: you have the DNA sequences to go on, which are unlikely to be in error. Those sequences are the direct result of experiments, while the protein sequences are mere inferences. But the work required toget all those sequences and then to translate them…



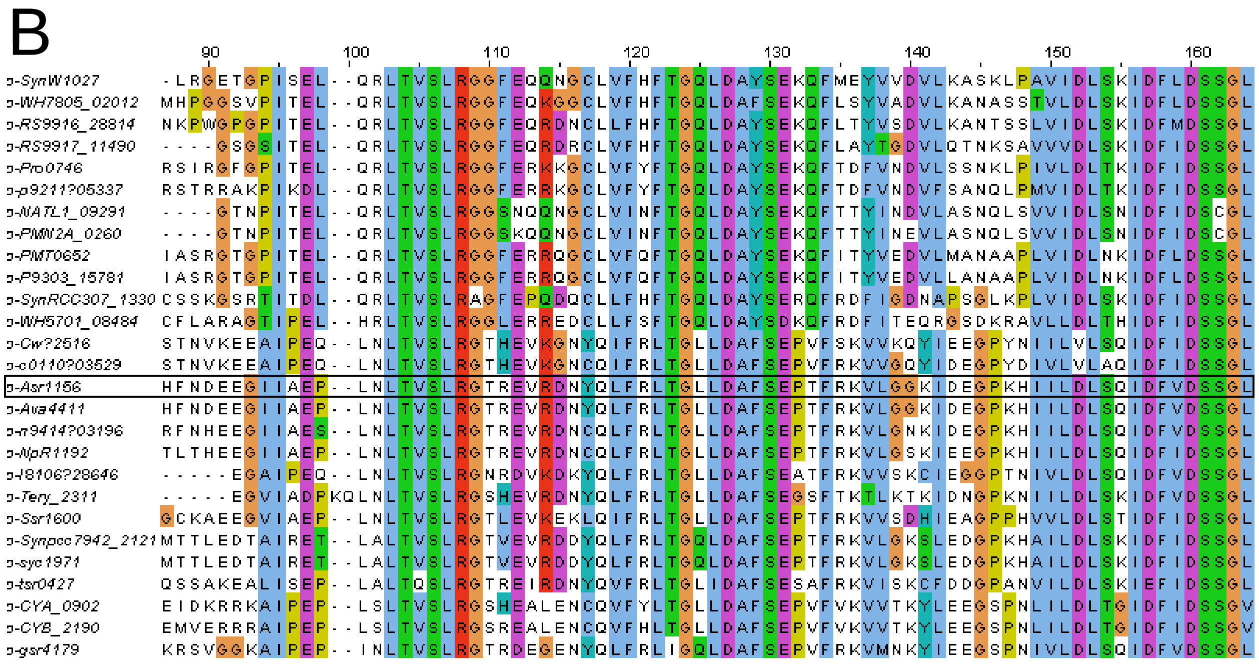


FIG. 1. Alignment of sequences of cyanobacterial orthologs of Asr1156. The alignment was produced as shown later in Fig. 6. To conserve space, only part of the alignment is shown. The line for the 70-amino acid protein p-Asr1156 is outlined. (**A**) Alignment of nominal protein sequences, according to annotated boundaries. (**B**) Alignment of protein sequences plus upstream potential amino acids, up to the proximal stop codon. Note that the same regions of the proteins are shown as in panel **A**.

Scenario (Part 4): …satisfied by an astonishing retranslation

Suppose that you are magically able to gather the upstream DNA sequences from the genes, virtually translate them in the genes' reading frames, and realign them, giving Fig. 1B. Clearly, there is overwhelming sequence similarity going backwards from the nominal beginning of each protein up to a conserved isoleucine at position 95 in the alignment. Beyond that point, the sequences become dissimilar. The annotated start codons indeed must be wrong, and not only for Asr1156 but for every protein in the list! No other explanation can reasonably account for such sequence similarity in proteins whose organisms you know diverged well over a billion years ago. But why did the gene callers do such an abysmal job? You examine the DNA sequences: In most cases there is no conventional start codon between the conserved isoleucine and the next upstream stop codon. You realize that the beginnings of these genes are determined by something apart from one of the conventional start codons, most likely the codon of the isoleucine itself.

- - - - - -

I have described a scenario where the end result is at odds with conventional wisdom, a result ordinarily hidden by the model of molecular biology built into our preexisting tools. In fact, in the case of one of the orthologs of Asr1156, there is strong experimental evidence that the protein begins with a conserved ATT (isoleucine) codon at the expected position (Binns and Masters, 2002). Though not widely appreciated, ATT has been previously shown to function in some genes as a start codon (Sazuka and Ohara, 1996).

All who work with their eyes open regularly encounter such situations where things seem to be profoundly wrong. Most of the time, there is a trivial explanation, but sometimes pursuing such anomalies leads to new and powerful ways of looking at the world. The history of scientific research is strewn with unsought accidental discoveries (Roberts, 1989). It is important to note that those in computer science generally hold a different view of serendipity (Saunders and Thagard, 2005). In the natural sciences, where the goal is understanding the sometimes mysterious ways of nature, surprise can be the impetus for basic change, but in computer science, where the goal may be to find the best process to reach a given end, surprise is generally an unwelcome irritant. One should not expect that a collaborator charged primarily with building a tool should view an accidental discovery in the same way as one charged primarily with using the tool to understand nature.

Whether you recognize a serendipitous finding depends on the preparedness of your mind. Machines can't do it, nor can technical specialists unfamiliar with the burning issues in your field. But there's more to it than that. Following up the intriguing observation depends in part on the time required for the pursuit or even the ability to pursue. At many different points in the journey described above, most would have put aside their nagging doubts, filing away the curious circumstances or forgetting about them entirely, because a meaningful pursuit would require abilities that practically speaking most do not have. The sense that something is wrong is one of our greatest friends in research, but without the means to follow up an observation as it is made, with as low an energy barrier as possible, chance discoveries are lost. The precepts of our time remain unchallenged because they cannot be effectively challenged.

Choices in the face of overwhelming information

In the past several years, biology has been showered with riches – genome sequences, gene expression information, huge positional data sets – literally beyond imagining, since the common thread is that the information is too much to fit in any human head. To extract its value, one must use computational tools that in some ways illuminate but in other ways obscure the fundamental data. That's true of any tool, and the greatest insight comes from combining the abstractions a tool provides with an appreciation of the process through which the abstractions were obtained (Pevzner and Shamir, 2009). However, few biologists understand the computational tools they use (and the assumptions hidden within those tools), and fewer still can meaningfully grapple with the underlying sequences and numbers in a creative way that requires the most basic and flexible of computational tools – computer programming.

Biologists seem to be faced with a list of unpalatable choices:

1. ***Ignore the wealth of information now available***
   This choice has the advantage of rooting one's research in tools one understands, thereby making more certain the connection between phenomena and conclusions. A principled approach, perhaps, but how frustrating!

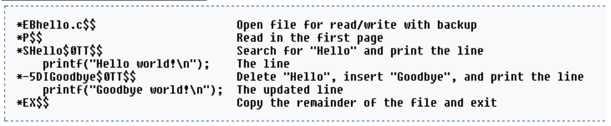
2. ***Muddle through as best one can with a limited number of tools***
   This strategy reduces the chance of attaining insights that require a broad understanding of a phenomenon, insights prompted by one of those annoying but precious results that wander outside the confines of our fixed tools. It promotes tool-directed rather than phenomenon-directed science.

3. ***Divide responsibilities amongst those with different expertise***
   Let the biologists think biological thoughts while working with computer types who think computational ones. This insidiously attractive option relies on the assumption that experimental results are produced by black boxes whose workings are of no scientific interest. The analogous situation would be directing a laboratory project from an office, with no knowledge of how experiments are being accomplished (or mushing blindly to the South Pole). We know well that a laboratory result has little scientific meaning without an appreciation of the experiment that produced it and the limitations of that experiment. Why should the experimental procedure be less critical for computational experiments? And if we give over the computational experiment to the programmer, then the unavoidable false turns and nonconformant intermediate results we are accustomed to see when we work with our own hands will be seen only by someone who is unprepared to grasp the biological implications and take the next step. By dividing responsibilities, we make rapid gains in predictable ways but slow the pace of insights that would radically change how we look at the world.

4. ***Entice microbiologists into learning computer programming***
   There has been ample incentive over the past 40 years for biologists to learn and use computer programming. They have voted overwhelmingly to this day not to do so.[*]

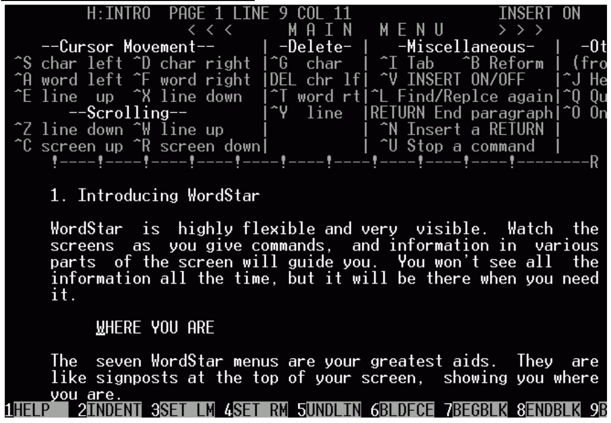
[*] In an informal survey of 6 departments of biology or microbiology in U.S. universities, only 4% of their members (out of 124) were reported able to write computer programs to the extent that they use them in their research.

TECO (1970's)



```
*EBhello.c$$                    Open file for read/write with backup
*P$$                            Read in the first page
*SHello$0TT$$                   Search for "Hello" and print the line
    printf("Hello world!\n");   The line
*-5DIGoodbye$0TT$$              Delete "Hello", insert "Goodbye", and print the line
    printf("Goodbye world!\n"); The updated line
*EX$$                           Copy the remainder of the file and exit
```

WordStar (1980's)



```
         H:INTRO  PAGE 1 LINE 9 COL 11                INSERT ON
                        < < <     M A I N   M E N U      > > >
   --Cursor Movement--      | -Delete- |   -Miscellaneous-  |  -Ot
^S char left ^D char right |^G  char   | ^I Tab    ^B Reform | (fro
^A word left ^F word right |DEL chr lf| ^V INSERT ON/OFF    |^J He
^E line  up  ^X line down  |^T word rt|^L Find/Replce again|^Q Qu
      --Scrolling--         |^Y  line  |RETURN End paragraph|^O On
^Z line down ^W line up     |           | ^N Insert a RETURN |
^C screen up ^R screen down|           | ^U Stop a command  |
   !----!----!----!----!----!----!----!----!----!----!--------R

    1. Introducing WordStar

    WordStar  is  highly flexible and very  visible.  Watch  the
    screens  as  you give commands,  and information in  various
    parts  of the screen will guide you.  You won't see all  the
    information all the time, but it will be there when you need
    it.

         WHERE YOU ARE

    The  seven WordStar menus are your greatest aids.  They  are
    like signposts at the top of your screen,  showing you where
    you are.
1HELP   2INDENT 3SET LM 4SET RM 5UNDLIN 6BLDFCE 7BEGBLK 8ENDBLK 9B
```

Word (2000's)

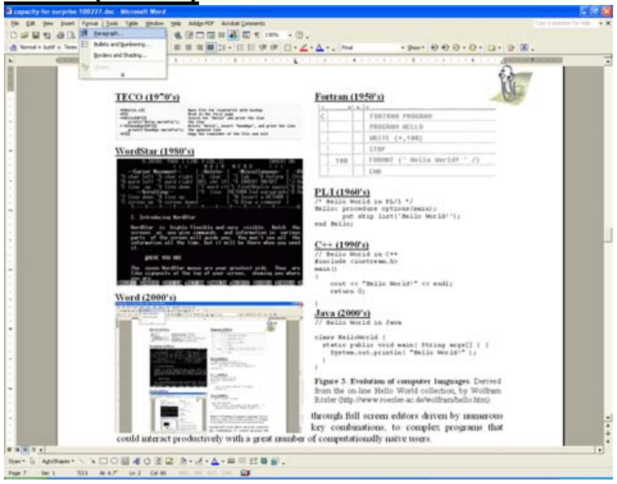
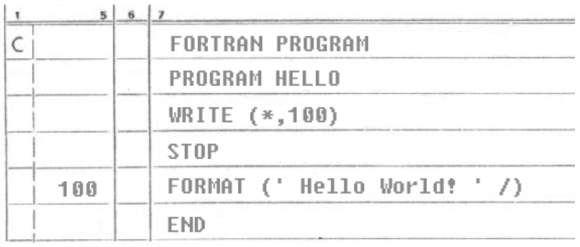


Figure 2. Evolution of word processors.

Fortran (1950's)



```
1   5 6 7
C         FORTRAN PROGRAM
          PROGRAM HELLO
          WRITE (*,100)
          STOP
  100     FORMAT (' Hello World! ' /)
          END
```

PL/I (1960's)

```
/* Hello World in PL/1 */

Hello: procedure options(main);
       put skip list('Hello World!');
end Hello;
```

C++ (1990's)

```
// Hello World in C++

#include <iostream.h>
main()
{
    cout << "Hello World!" << endl;
    return 0;

}
```

Java (2000's)

```
// Hello World in Java

class HelloWorld {
  static public void main( String args[]
) {
    System.out.println( "Hello World!" );
  }
}
```

Figure 3. Evolution of computer languages. Derived from the on-line Hello World collection, by Wolfram Rösler (http://www.roesler-ac.de/wolfram/hello.htm).

One might imagine an additional choice, one that does not yet exist but should: that computer programming become as accessible to typical researchers as word processing, so that anyone can use it to manipulate bioinformation in creative ways with minimal activation energy.

Computation vs word processing

The analogy of word processing is informative (Haigh, 2006). Just 35 years ago word processing was confined to those who worked with computers professionally. Originally commercial word processing programs were designed for pools of specialist within an automated office. This vision of the world did not last for long, as software companies discovered the more lucrative market of individual users. As a result, word processors evolved rapidly (Fig. 2) from clumsy line editors, through full screen editors driven by numerous key combinations, to complex programs that could interact productively with a great number of computationally naïve users.

Compare this progression of word processors with the succession of programming languages from the 1950's until today (Fig. 3). Computer languages have become far more powerful during that period, but they have not become significantly more usable by those new to the game. It is no easier today than it was 50 years ago for a novice to learn how to write simple computer programs. Probably less so.

Is it truly more difficult to encompass the operations necessary to program a computer to do what we wish than it is to encompass the operations necessary to run a full-featured word-processor? Computer programmers will generally respond with a resounding "Yes!", but is that really so? If we wish to write complex applications that are efficient, robust, and resilient, then programming a computer is indeed complicated, intrinsically so. But if we wish to do those things that are simple for researchers to conceive of, then programming should not be so complicated. Just because typesetting a newspaper is inherently difficult, that's no reason why we should not have the capability to compose our thoughts on a word processor. We gain enormously living in a society where almost all can read street signs and write shopping lists, even though very few amongst us are Shakespeares.

In the end, however, I do confess that computer languages by their nature must be more complicated. After all, a word processor has only a limited number of things it can do (if we exclude the computer languages embedded in some), while, in contrast, connect a creative human, a digital computer, and a general purpose language and what is there that they cannot do? How do we avoid being overwhelmed by the complexity that seems inevitable from a language that gives free rein to creativity? I'll return to this question in a moment.

Summary of the problem

Fundamental changes in our views of the world come about through anomalous observations that challenge and ultimately break the prevailing models through which we filter perception (Kuhn, 1962). They are made possible by the ability to loosen the hold of the old paradigm and rearrange perceptions in a new way. The most basic discoveries are necessarily surprises, requiring a cohabitation in one mind of the surprising observation and the data (minimally filtered) that must be rearranged.

That's the way science is supposed to work. Today, the enormous amount of biological information – genomes, microarrays, data from other high-throughput experiments – has allowed research to make breakneck progress in increasing our understanding of how cells and organisms function. However, this rapid progress has come at a cost: a reduced connection between researcher and unprocessed data and the consequent reduced possibility for the most fundamental surprises. We gain in the short term. In the long run, we're in for trouble.

Most biologists are incapable of asking new questions directly about raw genomic information or any other mass data, because that would require going outside of fixed computational tools and making new tools. In short, it would require the ability to program a computer. We are stuck in our offices, cut off from the experiments that feed us the results we chew on, and oblivious to the surprising accidents that would rock our foundations. We are blind explorers, fixated on the South Pole but oblivious to new phenomena we don't know exist and don't know how to look for.

But how can biologists become computationally adept? It takes considerable effort to learn a programming language sufficiently well so as to build a new computational tool, and it takes a

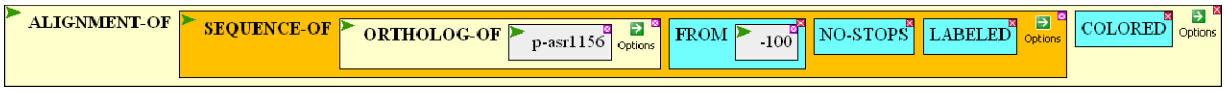


Figure 4. Example of a BioBIKE function. The function makes an alignment (shown in Fig. 1B) of the sequences of all orthologs of a protein, starting as many as 100 amino acids before the nominal beginning of each protein but going backwards only up to the first stop codon.

particular cast of mind to enjoy the process. Understandably, few biologists have made the investment. Human nature is unlikely to change. Biologists: It's not our fault!

How about machine nature? Machine interfaces ***can*** change. Word processors have evolved from simplistic and clunky to powerful and intuitive. In contrast, computer languages have evolved from simplistic and clunky to powerful and clunky, from the perspective of the naïve biologist. Most efforts today at interfaces connecting biologists with bioinformation focus on providing pushbutton access to filtered information and tools. These have been valuable to achieve their limited purposes, but it has been a serious mistake to go no further. The reconnection of biologists with unprocessed data requires a rapid evolution of computer languages to those that can be used at a simple level to facilitate the manipulation of raw data in new and creative ways.

Towards a solution: Human-centered programming

If the article stopped here, you might be excused for leaving discouraged. I hope, however, that you will instead take away the brighter note sounded at the end, that given sufficient incentive, powerful interfaces can be developed to engage humans with minimal computational experience. Our group has taken initial steps in this direction, developing an integrated knowledge and programming environment, called BioBIKE (**B**iological **I**ntegrated **K**nowledge **E**nvironments; Elhai et al., 2009), freely accessible through the web at http://biobike.csbc.vcu.edu/ (with links on the site to many tours of the resource). While BioBIKE offers a considerably less intimidating interface than conventional languages, it still poses more challenges to a new user than a modern word processor, more than most biologists would tolerate. It is not yet a solution, but future efforts may build upon the results of our experiment.

The main principles on which BioBIKE rests might be basic to any practical solution. First, knowledge is integrated with the means to manipulate it. Naïve users should not be asked to assume that most difficult and most tedious of programming chores: finding data and converting it from one form to another. Second, utterances must be comprehensible at sight to those in a targeted community. An example is provided in Fig. 4. Third, the language should cater to the needs of a specific clientele (in the case of BioBIKE, that would be those with questions in the domain of molecular biology). Questions or requests that are easy to pose in the language of the specified domain should be easy to express in the programming language. However, the language must still be capable of expressing anything within the realm of a general purpose language (perhaps with considerably more difficulty). Fourth, the language must be extensible by its practitioners, as a language can meet the needs of a community facing new phenomena only when that community is empowered to coin new tools and concepts. If a series of operations is found to be of general use, then a user should be able to package them into a chunk for use under a single name and that chunk incorporated into the language.

Unfortunately, these principles are not sufficient, even when assiduously followed, to make programming accessible, in a practical sense, to most biologists. A statement may make perfect sense when a user sees it but that same user may well be at a loss to construct it from scratch. We need the computer in Star Trek, one that can understand our intentions and figure out a way to satisfy them. Computer intelligence of that sort is far in the future. For now, the greatest progress may be made in developing a conversation between the language environment and the naïve user, where the interface does not need true intelligence but can nonetheless guide the user in the development of the needed tool.

Programming a computer will never be easy, because it is difficult to formulate questions that are both unambiguous and meaningful. However, formulating logical questions is what researchers do for a living. This is not the primary obstacle to enabling biologists to program the computer. That obstacle is the high activation energy that currently needs to be overcome to gain sufficient proficiency in a language to pose meaningful questions to a computer. So long as that obstacle remains in place, many biologists will continue to mush blindly forward, aided by easy-to-use programs built by many computer scientists.

I have tried in this article to address those two audiences. First, to those who study nature, if you are satisfied with your rate of progress, understand that traveling exclusively by highways offers speed but sacrifices most possible destinations. And if you despair at being able to depart from the smooth bioinformatics superhighway, trade that in for anger, and demand a language that enables you to analyze and manipulate mass biological information in a way that makes sense to you. Second, to those who are inventors of computational tools, consider the greater impact of devising a more general language that could potentially engage a vastly greater audience than those served by current computer languages, not one that aids a transition from a neophyte to a programmer like yourself, but a language that is an end point, one serving the needs of biological researchers and students. With this push and pull, a human-centered language might be developed sufficient to enable biologists to embrace computation as a basic tool, and we can thereby regain our historical connection with the raw data we study and regain the agility necessary to pursue surprising results to new biological insights.

ACKNOWLEDGMENTS

This work rests on the efforts of JP Massar, Jeff Shrager, and Mike Travers, who first conceived of and built BioBIKE and provided philosophical outlets and responses to many of the thoughts expressed in this article. Thanks also to John Myers, Peter Seibel, and Arnaud Taton, who provided much of the superstructure on which BioBIKE rests and to the many other participants in the project who sadly cannot be listed without going over the word limit. Thanks also to anonymous reviewers for valuable comments. The work was supported by grant DBI-0516378 and DBI-0850146 from the National Science Foundation.

REFERENCES

Binns, N., and Masters, M. 2002. Expression of the *Escherichia coli* pcnB gene is translationally limited using an inefficient start codon: a second chromosomal example of translation initiated at AUU. *Mol. Microbiol.* 44, 1287-1298.

Elhai, J., Taton, A., Massar, J.P., Myers, J.K., Travers, M., Casey, J., Slupesky, M., Shrager, J. 2009. BioBIKE: A web-based, programmable, integrated biological knowledge base. *Nucl. Acids Res.* 37, W28-W32.

Haigh, T. 2006. Remembering the Office of the Future: Word Processing and Office Automation before the Personal Computer. *IEEE Ann. History Computing* 28, 6-31.

Kuhn, T. 1962. *The Structure of Scientific Revolutions*. U. Chicago Press.

Pevzner, P., and Shamir, R. 2009. Computing has changed biology – biology education must catch up. *Science* 325, 541-542.

Roberts, R.M. 1989. *Serendipity: Accidental Discoveries in Science*. John Wiley & Sons, New York.

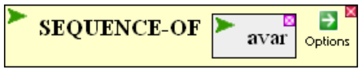
Saunders, D., and Thagard, P. 2005. Creativity in computer science, 153-167. *In* Kaufman, J.C., and Baer, J., eds., *Creativity Across Domains*. Lawrence Erlbaum Associates, London.

Sazuka, T., Ohara, O. 1996. Sequence features surrounding the translation initiation sites assigned on the genome sequence of *Synechocystis* sp. strain PCC6803 by amino-terminal protein sequencing. *DNA Res.* 3, 225-232.

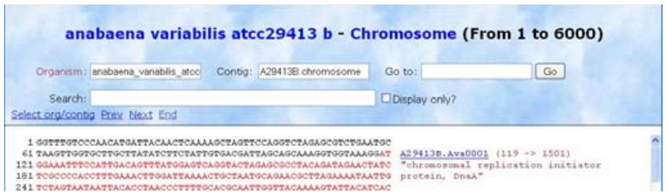
Appendix: Computer programming as a tool in biology education

The disconnect from tangible reality that I argue now confronts researchers has long been a way of life for students of molecular biology, much to their detriment. The advantages of learning by discovery rather than relying on lecture and textbooks are well documented (Wood, 2009), but molecular biology laboratories are expensive in both time and resources. Computational experiments offer an attractive alternative, and I have used BioBIKE to present university and high school students with a playground through which they can discover some concepts of molecular biology. At first, students must be led step by step through the process of asking useful questions and using BioBIKE to answer them, but it doesn't take long for them to gain increasing independence and the ability to strike out on their own.

I present here a highly abbreviated summary of an investigative tour that students without prior computational experience have used to discover concepts related to the nature of genes. You can find the complete tour by going to the BioBIKE portal (http://biobike.csbc.vcu.edu), clicking *Guided Tours*, and then *What is a Gene?*



```
SEQUENCE-OF  avar  Options
```



anabaena variabilis atcc29413 b - Chromosome (From 1 to 6000)

Organism: anabaena_variabilis_atcc   Contig: A29413B.chromosome   Go to: Go

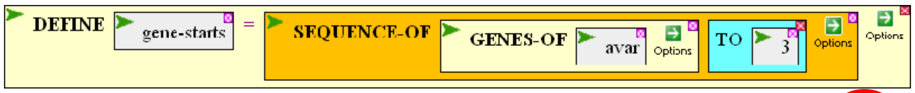
Search: Display only?

Select org/contig Prev Next End

```
  1 GGTTTGTCCCAACATGATTACAACTCAAAAGCTAGTTCCAGGTCTAGAGCGTCTGAATGC
 61 TAAGTTGGTGCTTGCTTATATCTTCTATTGTGACGATTAGCAGCAAAGGTGGTAAAGGAT  A29413B.Ava0001 (119 -> 1501)
121 GGAAATTTCCATTGACAGTTTATGGAGTCAGGTACTAGAGCGCCTACAGATAGAACTATC  "chromosomal replication initiator
181 TCGCCCCACCTTTGAAACTTGGATTAAAACTGCTAATGCAGAACGCTTAGAAAATAATTG  protein, DnaA"
241 TCTAGTAATAATTACACCTAACCCTTTTGCACGCAATTGGTTACAAAAGTATTACATCAC
```

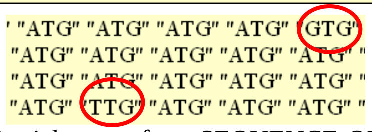
Students bring with them all sorts of fanciful notions concerning genes that connect verbally with textbook descriptions but bear little resemblance with reality. It is important to help them build a mental picture of a real genome. In this tour, they use the genome of the cyanobacterium *Anabaena variabilis* (nicknamed *Avar*) as their laboratory. BioBIKE allows them to explore the genome and realize the problem confronted by the cell: how is it possible to discern the beginning and end of a gene from a continuous sequence of nucleotides?

Drawing on the analogy of picking out sentences from English text, students consider two strategies: find internal signals in genes that mark their beginnings (analogous to capital letters), and find special sequences preceding genes (analogous to periods). To see if the internal signal hypothesis is true, the investigators experiment first with extracting the beginning of a single gene and then generalize the procedure over all genes of *Avar*. Seeing that there is a recurring pattern in the first three nucleotides, they focus on them, giving the set of opening triplets a name.



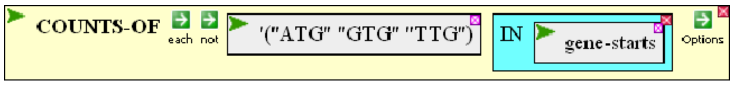
```
DEFINE  gene-starts  =  SEQUENCE-OF  GENES-OF  avar  Options  TO  3  Options  Options
```

Many investigators come to the tour with an expectation of finding ATG at the beginning of genes, as is typically stated in textbooks. They are surprised to find that some genes begin with other similar triplets. Do these triplets account for the beginnings of all genes? This can be tested by counting each one and comparing the sum to the number of all genes. As it happens, 55 genes are missing, evidently beginning with some other triplet.

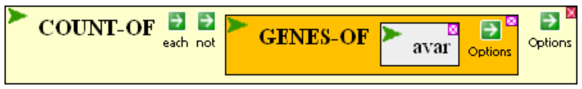


```
' "ATG" "ATG" "ATG" "ATG" "GTG"
 "ATG" "ATG" "ATG" "ATG" "ATG" "
 "ATG" "ATG" "ATG" "ATG" "ATG" "
 "ATG" "TTG" "ATG" "ATG" "ATG" "
```

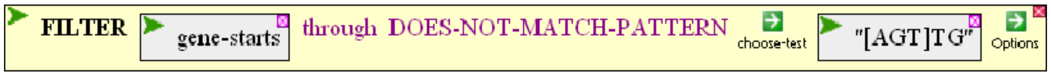
Partial output from SEQUENCE-OF



```
COUNTS-OF  each  not  '("ATG" "GTG" "TTG")  IN  gene-starts  Options
```



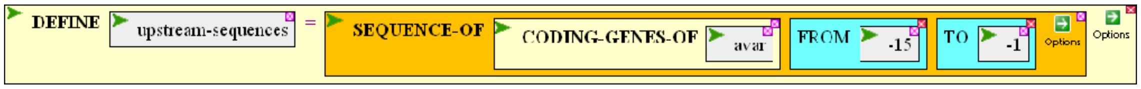
```
COUNT-OF  each  not  GENES-OF  avar  Options  Options
```



```
FILTER gene-starts through DOES-NOT-MATCH-PATTERN choose-test "[AGT]TG" Options
```

What are they? Filtering the list of triplets beginning genes, retaining only those that do not match the pattern of “A”, “G”, or “T” followed by “TG”, brings up a list of gene beginnings that are found to be associated with tRNA, rRNA, and other RNA-determining genes. In this way, investigators discover the concept of coding- and noncoding-genes. However, by counting the entire genome, they also find that there are many instances of “ATG” that do not begin genes. Some other feature must be important to signal the beginning of a gene.

The second strategy they employ is to examine sequences upstream from genes, focusing on those genes that encode protein.



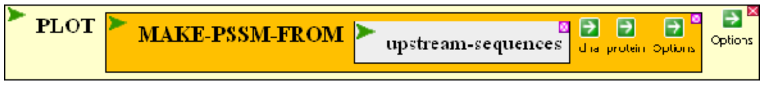
```
DEFINE upstream-sequences = SEQUENCE-OF CODING-GENES-OF avar FROM -15 TO -1 Options Options
```

At this stage, as in previous, investigators can go back to the genome sequence to make sure that the sequence extraction worked as expected. No obvious pattern emerges from inspection, so the upstream sequences are analyzed, counting the number of each nucleotide at each position, to obtain a Position-Specific Scoring Matrix (PSSM). The count is shown below, after the first gene has been tabulated:

-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	+1	+2	+3
C	A	A	A	G	G	T	G	G	T	A	A	A	G	G	A	T	G

	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
A		1	1	1							1	1	1		
C	1														
G					1	1		1	1					1	1
T							1			1					

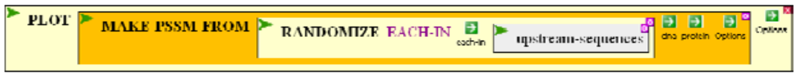
The complete count is done by the MAKE-PSSM-FROM function, and the results are plotted:



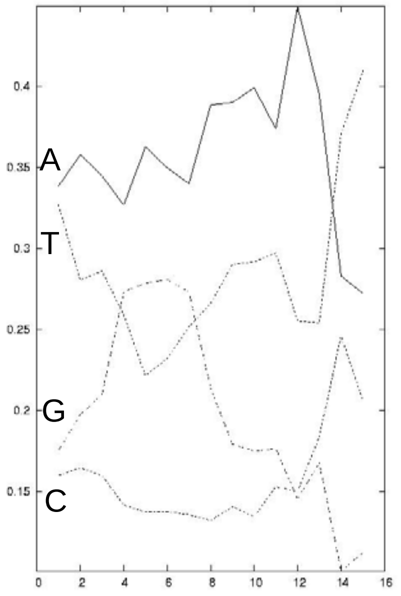
```
PLOT MAKE-PSSM-FROM upstream-sequences dna protein Options Options
```

It seems that there is a precipitous drop in A’s and G’s and corresponding rise in T’s and C’s before the beginning of genes. There’s also a rise and fall in G’s about 10 nucleotides before the beginning, and a strange up and down pattern in the A’s.

How much of this is real and how much imagined, just the result of random fluctuation? Investigators can readily get a feel for what random fluctuation there is by repeating the plot a few times with randomized sequences,
giving a typical plot below.



```
PLOT MAKE PSSM FROM RANDOMIZE EACH-IN each-in upstream-sequences dna protein Options Options
```

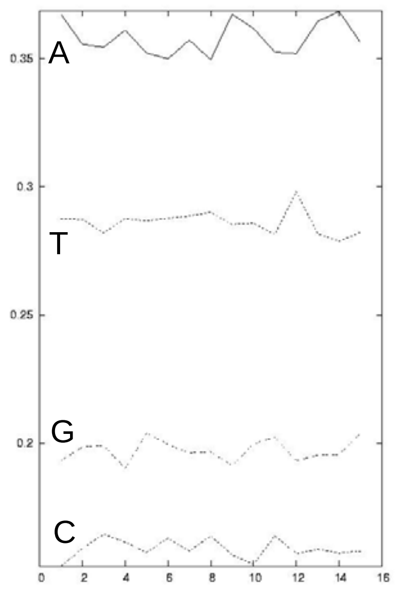


Positional frequencies of nucleotides upstream from the coding genes of *A. variabilis.* The positions 1 to 15 correspond to -15 to -1 relative to the beginnings of genes.

Randomization is used in other investigative tours to give students an intuitive feel for what statistical tests do and how to interpret them.

The tour ends with some hints but otherwise a free-form invitation for investigators to attempt to try out the tools they have used to find interesting features near the ***ends*** of genes.

Needless to say, it is not of critical importance that students discover the diversity of start codons or the presence of ribosome binding sites typical of prokaryotic genes. What is important is that they discover – anything – so that they learn to make a habit of it. It is also not critical that they become comfortable with the latest bioinformatics tools (otherwise known as the obsolete tools of a decade from now). What is of lasting importance is a degree of comfort with uncertainty, a skepticism towards ideas and information, and an urge to check both in any way possible. Students learn that computation is a powerful ally in this regard and that they are capable of making use of it without losing control.



Positional frequencies of nucleotides of randomized upstream sequences of *A. variabilis* genes.

Integrating computation and quantitative thinking into the biology curriculum can go a long way towards bridging the current gap felt by many to lie between biology and numbers. A computer programming language that meets students more than halfway is necessary to encompass the great majority of biology students who avoid computation. Students who gain computational skills may emerge as a new generation of biological researchers, one that is able to encounter the surprises that are hidden in bioinformation and to pursue them to new insights.

REFERENCES

Wood, W.B. 2009. Innovations in teaching undergraduate biology and why we need them. *Annu. Rev. Cell Dev. Biol.* 25, 93-112.