Regaining our capacity for surprise

Jeff Elhai, Arnaud Taton, JP Massar, Jeff Shrager

The study of microbes has been tied to observation, through microscopy and the careful scrutiny of macroscopic characteristics. The study of the molecular biology of microbes is similarly connected to observation, for example electron micrographs of adenoviral RNA-DNA hybrids leading to the discovery of introns (1,2). The most basic discoveries in biology, regardless of discipline, are generally associated with the word "surprise" and a deep connection between the observer and the object of study, permitting a radical new interpretation of the surprising observation.

Why state the obvious?

Because the close connection between the biologist and the phenomenon under study is breaking down. We are paying for the rapid progress made possible by the new wealth of biological information by a reduced connection with unprocessed data and the consequent reduced possibility for productive surprises. We gain in the short term. In the long run, we're in for trouble.

The problem is that the information we have been gifted with – genomes, microarrays, data from other high-throughput experiments – demand computation in order to make sense of it, and few biologists are comfortable with computation. So what? Few biologists are comfortable with the physics of optics, yet we use microscopes. Few of us can build a spectrophotometer, yet we can still understand what a spectrum may tell us. The relationship of biologists with bioinformation is fundamentally different, however. Biologists using these older tools of biology could see the object of their study to the extent that it was technically possible to be seen. Perhaps a spectrophotometric scan is automated and provides intensities at preset wavelengths. If we're suspicious, we are fully capable of looking at the full spectrum or even painfully examining hundreds of specific wavelengths if we choose to. Few biologists are equally capable of examining raw genomic information or any other mass data, because that would require going outside of fixed computational tools and creating a new set of directions. That would require control over the process, i.e. programming the computer.

As a result, most of the growing number of biologists that use mass information confine themselves to the few computational tools, like Blast and Clustal, that have well developed user interfaces that cater to the noncomputational biologist. We gain comfort from this ease of use and the familiarity that comes from repetition, but that layer of comfort is thin. As soon as the tool begins to behave in ways we consider out of bounds, we are thrown back into confusion, with no means by which to steady our gaze and find new order.

The need for computational agility: A scenario

Suppose, for example, that you are interested in a protein, Asr1156 (GenBank NP_485199), from the cyanobacterium *Anabaena* PCC7120, that is induced during differentiation and annotated as "hypothetical". Blast returns the result that it is similar to the C-terminus of proteins from all or nearly all cyanobacteria with completed genomes. The protein is usually annotated as "an anti-sigma factor antagonist", a type of protein that might indeed be interesting with respect to differentiation. On most days, that would be the end of it, but suppose that today you expend the effort necessary to obtain the sequences of the similar protein and to align them, getting the

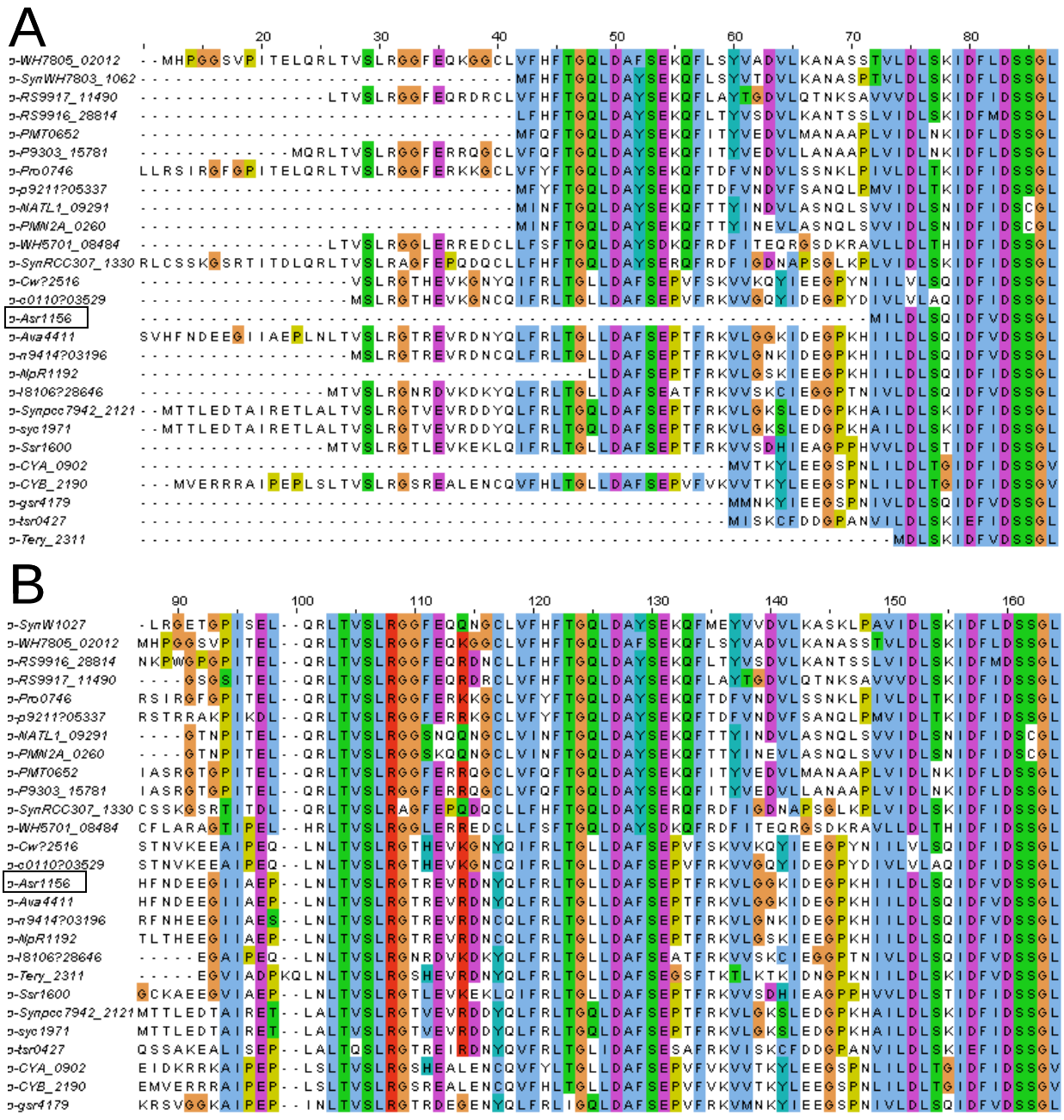


FIG. 1. Alignment of sequences of cyanobacterial orthologs of Asr1156. The alignment was produced as shown later in Fig. 6. To conserve space, only part of the alignment is shown. The line for Asr1156 is highlighted in pink. (**A**) Alignment of nominal protein sequences, according to annotated boundaries. (B) Alignment of protein sequences plus upstream potential amino acids, up to the proximal stop codon.

alignment shown in 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.

Suppose it occurs to you that perhaps the annotation is wrong, and what is purported to be the start codon for these short open reading frames are really within the gene. An interesting thought, but one that is very difficult to act on. But suppose further, that you are somehow able to gather

the upstream sequences from all these genes, virtually translate them, and realign them, giving Fig. 1B. There is overwhelming sequence similarity going backwards from the nominal beginning of proteins up to a conserved isoleucine at position 95 in the alignment. Beyond that point, the sequences become dissimilar. It is evident that indeed the start codons must be wrong, and not only for Asr1156 but for every gene in the list! No other explanation can account for such sequence similarity in genes that diverged well over a billion years ago. But why did the gene callers do such an abysmal job? Answer: In most cases there is no conventional start codon between the conserved isoleucine and the next upstream stop codon. The beginnings of the genes are determined by something apart from one of the conventional start codons.

We 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. At many different points in the journey, most would have filed away the curious circumstances because following up would require abilities that practically speaking we do not have. Without the means to follow up an observation as it is made, chance discovery is lost.

Choices in the face of overwhelming information

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.
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. We know this to be manifestly false in the case of physical experiments, whose results often depend markedly on exactly how the question is posed. 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 when we work with our own hands will be seen only by someone who does not grasp the biological implications and who does not recognize how a chance observation may attack some basic assumption. 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 biologists into learning computer programming***
   There has been ample incentive over the past 40 years for biologists to learn computer programming. They have voted overwhelmingly not to do so.

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.

TECO (1970's)

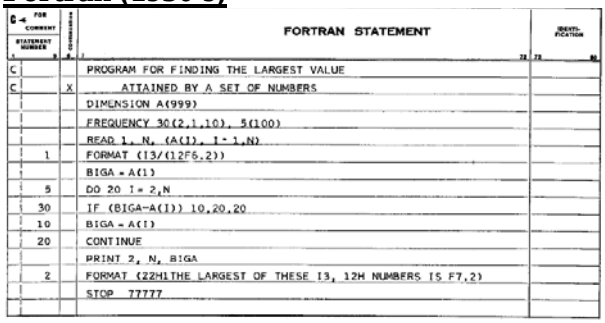
```
*EBhello.c$$                    Open file for read/write with backup
*P$$                            Read in the first page
*SHello$0IT$$                   Search for "Hello" and print the line
    printf("Hello world!\n");   The line
*-5D1Goodbye$HIT$$              Delete "Hello", insert "Goodbye", and print the line
    printf("Goodbye world!\n"); The updated line
*EX$$                           Copy the remainder of the file and exit
```

WordStar (1980's)

Word (2000's)

Figure 2. Evolution of word processors.

Fortran (1950's)



```
C     FOR COMMENT     CONTINUATION     FORTRAN STATEMENT     IDENTIFICATION
C     STATEMENT NUMBER
C                     X     ATTAINED BY A SET OF NUMBERS
      DIMENSION A(999)
      FREQUENCY 30(2,1,10), 5(100)
      READ 1, N, (A(I), I=1,N)
    1 FORMAT (I3/(12F6.2))
      BIGA = A(1)
    5 DO 20 I = 2,N
   30 IF (BIGA-A(I)) 10,20,20
   10 BIGA = A(I)
   20 CONTINUE
      PRINT 2, N, BIGA
    2 FORMAT (22H1THE LARGEST OF THESE I3, 12H NUMBERS IS F7.2)
      STOP 77777
```

Algol (1960's)

```
procedure Absmax(a) Size:(n, m)
    Result:(y) Subscripts:(i, k);
    value n, m; array a; integer n, m, i, k;
    real y;
begin integer p, q;
    y := 0; i := k := 1;
    for p:=1 step 1 until n do
    for q:=1 step 1 until m do
        if abs(a[p, q]) > y then
            begin y := abs(a[p, q]);
            i := p; k := q
            end
end Absmax
```

Java (2000's)

```
public void insertionSort()
  {
  int in, out;

  for(out=1; out<nElems; out++)
   {
   long temp = a[out];
   in = out;
   while(in>0 && a[in-1] >= temp)
     {
     a[in] = a[in-1];
     --in;
     }
   a[in] = temp;
   }
  }
```

Figure 3. Evolution of computer languages.

The analogy of word processing is informative (3). Just 35 years ago word processing was confined to those who worked with computers professionally. Originally commercial word processing programs were designed for specialist pools 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 handle a bewildering number of demands from users, many of whom would consider themselves noncomputational.

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 not significantly more usable by those who would consider themselves noncomputational. It is no easier today than it was 40 years ago for a novice to learn how to program a computer.

Is it truly more difficult to encompass all the operations necessary to program a computer than it is to encompass all the operations necessary to run a full-featured word-processor?

Table 1. Current BioBIKEs[a]

CyanoBIKE: Cyanobacteria (37 genomes)
ParaBIKE: Eukaryotic parasites (5 genomes)
PhotoBIKE: Photosynthetic bacteria (63 genomes)
StreptoBIKE: Streptococcus (27 genomes)
ViroBIKE: Viruses (1797 genomes and 16 metagenomes)

[a]All instances are available through http://ixion.csbc.vcu.edu/biobike

We have taken the lessons learned by word processor developers and applied them to an environment that enables biologists to access and manipulate bioinformation in ways limited only by their imaginations. The result is an integrated knowledge and programming environment that dispenses as much as possible with the arbitrary syntactical contrivances – the semicolons and parentheses – that so often dissuade biologists from learning the tools they need to make creative use of the knowledge available to them.

BioBIKE Knowledge Bases

We have defined an instance of BioBIKE (**B**iological **I**ntegrated **K**nowledge **E**nvironments) as a body of all available information of specific use to a coherent research community. It may seem retrograde to define packets of knowledge in this way, given the trend towards increasingly distributed data linked through the web (4). Doing so, however, brings significant benefits: (a) It is easier to find the data you want, (b) Data can be housed on a single server, greatly reducing access time and permitting calculations that would not otherwise be practical. Users of BioBIKEs have access to two levels of information: rapid access to community-specific knowledge and normal access to outside databases.

At the time of writing there have been five BioBIKEs established (Table 1), all freely available through the web. Each BioBIKE contains genomic sequences, annotation, metabolic function and links to related pathways, and orthologies amongst encoded proteins. In addition, experimental data (e.g. microarrays and results from proteomic experiments) may be available. Similarity scores for all proteins in the database are predetermined, so results of queries regarding protein similarity are available virtually instantaneously. Orthologs (based on bidirectional best hit, per Blast (5) at a threshold of $E = 10^{-6}$) are also precomputed, making it possible to find orthologs common to any subset of organisms in the database within seconds. It is a simple matter, for example, to find in CyanoBIKE all proteins common to, say, nitrogen-fixing cyanobacteria that are not found in those that don't fix nitrogen. Protein similarities and orthologs amongst proteins of organisms outside the database are also available, more slowly, through the NCBI implementation of Blast (5) and through KEGG (6). Orthologs may also be determined, more slowly, through any definition provided by the user.

Users are thus spared what is sometimes the grueling task of locating data and converting it to the formats required by different software application.

BioBIKE Programming Environment

The goal of the BioBIKE programming environment is to free the biologist from the burden of arbitrary syntax to take on what should be the more difficult task of formulating biologically meaningful questions. As much as possible, questions that are easy to conceive are also easy to ask. This requires that the language is cognizant of many concepts of molecular biology, such as

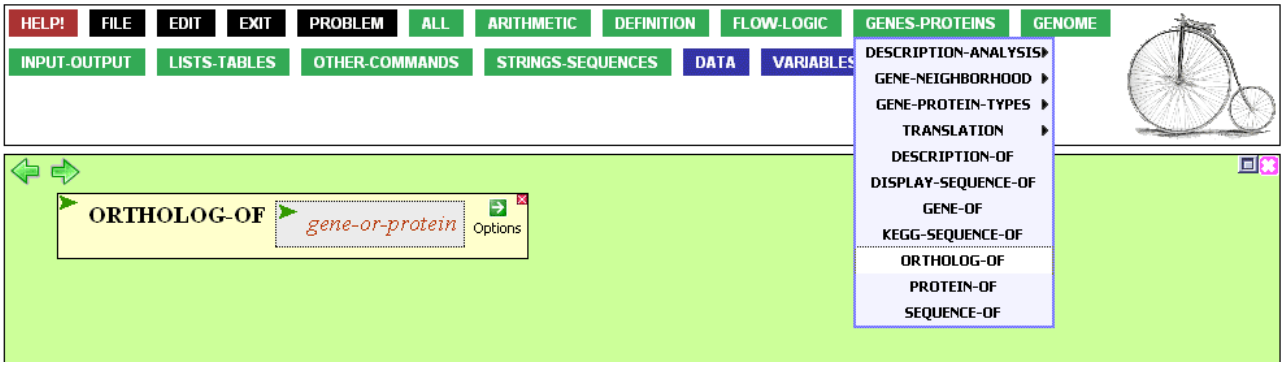


Fig. 4: BioBIKE palette and workspace.

"codon" and "ortholog". The environment adopts conventions common to word processors and web applications, e.g. menus and cut/paste operations.

The fundamental unit of the language is the function, represented as a box. Functions may be selected from one of several menus available from a function palette (Figure 4). The function specifies what input it requires of the user, and the user may supply the information by selecting the entry box and then typing it in or choosing an item from a data menu. The data menu provides (amongst other things) all available organisms and several convenient subsets (Figure 5).

The environment adheres to several design principles:

Intelligibility

The goal of the language is that the meaning of a function should be immediately intelligible to a molecular biologist with no prior experience with BioBIKE.

Limited vocabulary

To reduce the weight of language on the user, a single function may be employed for many different purposes, as directed by options chosen from function-specific menus. For example, the function SEQUENCE-SIMILAR-TO may be used to blast one protein sequence against another protein, against all proteins of one or more organisms in the database, or against all proteins in GenBank. The same function is used to perform nucleotide comparisons or comparisons of translated nucleotides against a protein database, and so forth. The function is also used to find sequences differing from a reference by a given number of mismatches.

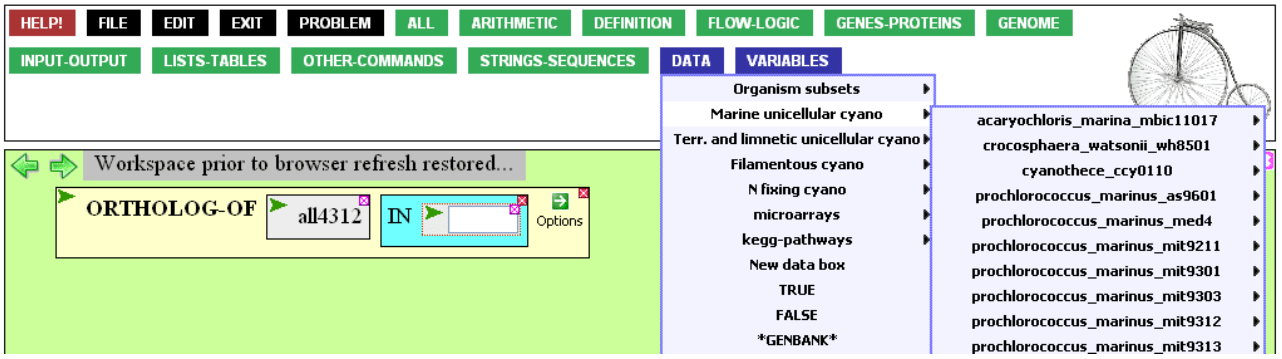


Fig. 5: Choosing organisms from DATA menu.

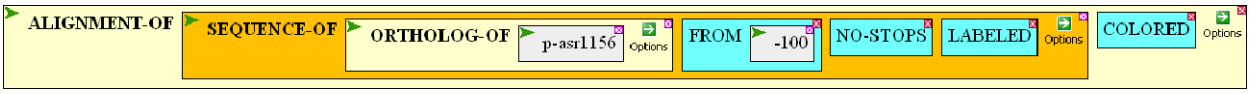


Fig. 6. Example of nested functions. The nested function makes an alignment of the sequences of all orthologs of the protein Asr1156, starting as many as 100 amino acids before the nominal beginning of the protein but going backwards only up to the first stop codon. The sequences are labeled with the name of the protein and aligned, using Clustal (7) and visualized using JalView (11).

Computability of results

A function may display results in a format designed for ready comprehension, but it will also return the results in a form available for further computation. For example, the results of a Blast performed through SEQUENCE-SIMILAR-TO can be fed directly into a function that performs alignments.

Nesting

Function boxes may serve as input for other function boxes, creating logical phrases (Fig. 6).

Reusability

Functions may be executed repeatedly, perhaps after modifying the contents of an input box. In Fig. 6, the user may re-execute the nested function after replacing the original protein with another.

Multiple arguments

Most BioBIKE functions that make sense to work on both individual items and lists of items do in fact work on both, returning a list of results when provided with a list of arguments. In Fig. 5, for example, the object of the IN option could be a single organism or a list of organisms. This feature greatly reduces the need for loops, the bane of beginning students of computer languages.

Single interface to multiple applications

BioBIKE provides access to several programs that are commonly used: Blast (ref 5; for sequence searches), Clustal (ref 7; for multiple sequence alignments), Meme (ref 8; for motif discovery), RNAz (ref 9; for discovery of conserved RNA sequences), and Phylip (ref 10; for construction of phylogenetic trees). Fig. 7 shows an example of the general interface through BioBIKE function boxes, where the specific behavior of the function is controlled by options chosen from a menu. Figuring out the requirements of free-standing Phylip would be a much more arduous task.

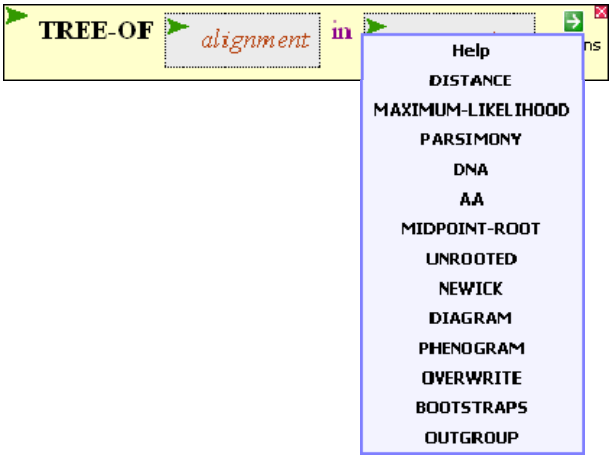


Fig. 7: BioBIKE access to Phylip

Integration of different data types

The examples thus far have focused on genomic information, but BioBIKE a great deal else of interest to a research community. Fig. 8 illustrates how the metabolic knowledge built into BioBIKE, interaction with KEGG, and microarray data can work together to allow a user to find expression values for a specific class of genes. The community can stock the site with microarray

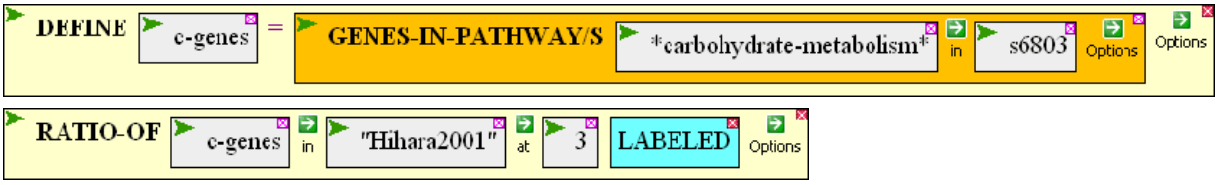


Fig. 8: Integration of metabolic and microarray information. Genes from the cyanobacterium *Synechocystis* PCC 6803 (nicknamed S6803) that encode proteins related to carbohydrate metabolism are gathered into a set called "c-genes". A list is then generated of the expression ratios of these genes in the microarray experiment of Hihara et al (12), using the third experimental condition, six hours after a shift from normal to high light intensity. The ratios are labeled with the names of the genes.

data, using a built-in function that can be adapted to a wide variety of different formats. Any data that can be put in the form of a table can be integrated into the knowledge-base.

Porous workspace divisions

All results and functions are confined to the user's personal workspace. However, users are able to share interesting results and functions by placing them in a common workspace. In this way, useful information can be propagated through the community in a form that maximizes its utility. As a body of knowledge of interest to a coherent research community, a BioBIKE may serve individuals of that community but also may help bring the community together.

Styles of BioBIKE Usage

Those new to BioBIKE often begin by using it as a simple query language (e.g., "*What is the sequence of my favorite gene?*"). From there, the next step is often to construct series of queries, each one dependent on the result of the previous (e.g., "*What are the orthologs of my favorite gene?*" "*What are the upstream sequences of those orthologs?*" "*What common sequence motifs are found in those upstream sequences?*"). This style of usage is made easy by the ability to copy and paste results into the argument boxes of functions and the ability to nest functions.

Many users continue within this style, but the language makes it possible to go much further, automating processes to the same extent as one can with any general purpose computer language. For example, suppose you find that the genome of *Anabaena* PCC 7120 (A7120) has only 8 instances of the sequence CCCGGG, using the COUNT-OF function (Fig. 9A). This seems intuitively to be too low, but you're not sure, so you repeat the count using a random DNA sequence the same length and nucleotide composition as A7120, and for statistical purposes, you repeat the simulation 3 times (Fig. 9B). Indeed, you find that the expected number of instances of CCCGGG is not 8 but 570. Is this oligomeric sequence uniquely special, or are there others that are also underrepresented? To answer that question, you decide to count every possible 6-nt oligomer, keeping only those that have fewer than 10 counts (Fig. 9C), and find that there are only 12 such hexamers (Fig. 9D). Sorting them leads to the remarkable finding that the 12 can all be fit into one of three restriction enzyme recognition sequences: *Ava*I (CyCGrG) from *Anabaena* PCC 7120 itself, *Aoc*II (G[AGT]GC[ACT]C) from *Anabaena* CCAP 1403/9, and *Nsp*I (rCATGy) from *Nostoc* ATCC29411 (a related cyanobacterium). Needless to say, it is highly suspicious that the 12 rarest hexamers (out of $4^6 = 4096$) are recognition sites for enzymes from a close set of relatives!

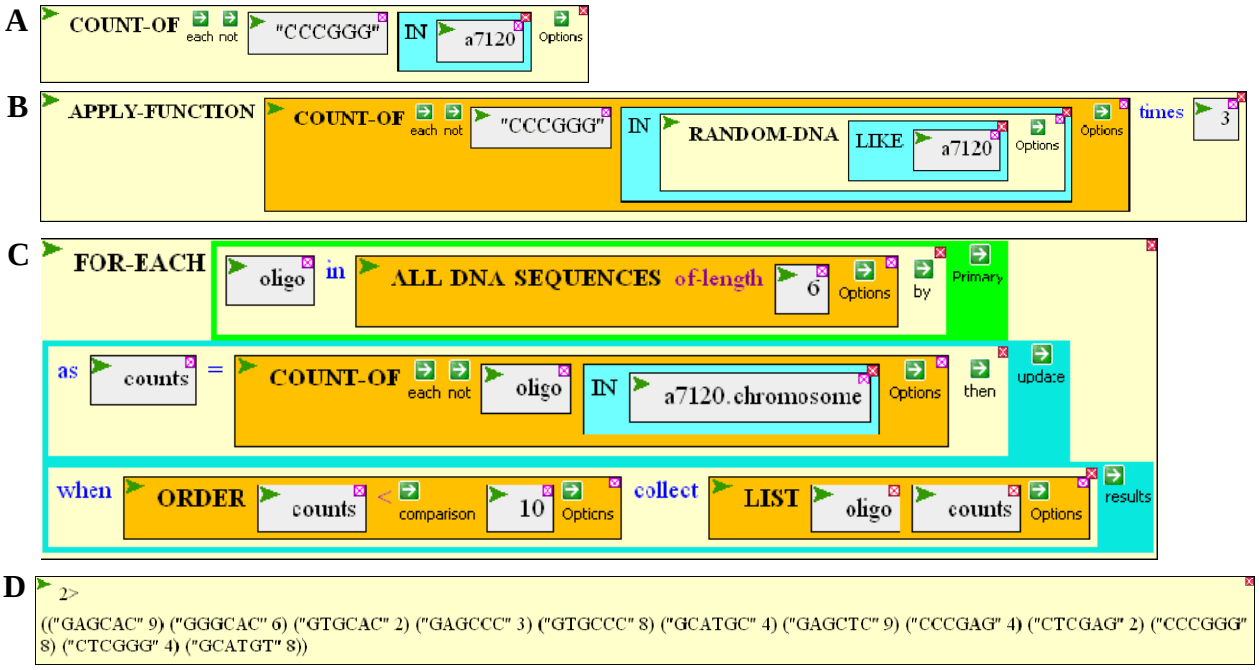


Fig. 9: Automated re-execution of functions

It is very common to run across a finding that seems intuitively peculiar. Usually these findings remain mere curiosities owing to the difficulty of following up the observations with meaningful tests. Chance discoveries from genomic information, like the one just described, are made possible by the ease with which computational tests can be implemented in BioBIKE.

Whenever a user has devised a set of operations that might be of general utility, the operations can be given a name and packaged into a function, accessible from a menu no differently from any other BioBIKE function. In this way, complicated operations can be broken up into logical chunks, thought of as named items, and used as distinct functions. Complicated chunks can also be collapsed visually into single boxes, making it easier to grasp the larger picture.

General Applicability

BioBIKE is the first attempt we know of to engage biologists without computational experience in the creative analysis of mass bioinformation through an accessible language tailored for their needs. Instances of BioBIKE have been used by researchers, university undergraduates, and high school students, very few of whom have had any prior experience with computer programming. There is no doubt that those without experience can become productive in a very short period of time.

As a first attempt, however, BioBIKE suffers from a number of deficiencies, for example, the sometimes crude representation of results, the incompleteness of documentation for the naïve user, and the deficiency of tools to work with microarrays and other mass data. These deficiencies are being addressed, and we hope that others will also build on our efforts. Any user can add a new function and can share it with others. If deemed useful, it can be integrated into the base language with little difficulty. Those who wish to make deeper changes in BioBIKE can

readily do so, as it is open source, freely available through SourceForge (sourceforge.net). Members of different research communities are invited to build new instances devoted to their favorite groups of organisms, either working with us or on their own.

Programming a computer will never be easy, because it is difficult to formulate questions that are both unambiguous and meaningful. However, formulating such 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 needs to be overcome to gain sufficient proficiency in a language to ask any meaningful question. BioBIKE reduces that activation energy to a minimum. If we biologists can overcome that barrier and embrace computation as a basic tool, we can regain our historical connection with the raw data we study and regain the agility necessary to pursue chance results to new biological insights.

REFERENCES

1. **Berget, S. M., C. Moore, and P. A. Sharp.** 1977. Spliced Segments at the 5′ Terminus of Adenovirus 2 Late mRNA. Proc. Natl. Acad. Sci. USA **74:**3171-3175.
2. **Chow, L. T., R. E. Gelinas, T. R. Broker, and R. J. Roberts.** 1977. An amazing sequence arrangement at the 5′ends of adenovirus 2 messenger RNA. Cell **12:**1-12.
3. **Haigh, T.** 2006. Remembering the Office of the Future: Word Processing and Office Automation before the Personal Computer. IEEE Ann. History Computing **28:4:**6-31
4. **Dowell, R. D., R. M. Jokerst, A. Day, S. R. Eddy, L. Stein.** 2001. The Distributed Annotation System. BMC BioInformatics **2**:7.
5. **Altschul, S. F., T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman.** 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucl. Acids Res. **25:**3389-3402.
6. **Kanehisa M., S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, M. Hirakawa.** 2006. From genomics to chemical genomics: new developments in KEGG. Nucl. Acids Res. **34**:D354-357.
7. **Thompson, J. D., D. G. Higgins, T. J. Gibson.** 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucl. Acids Res. **22:**4673-4680.
8. **Bailey, T.L., C. Elkan.** 1995. Unsupervised learning of multiple motifs in biopolymers using EM. machine learning **21:**51-80.
9. **Washietl, S., I. L. Hofacker, P. F. Stadler.** 2005. Fast and reliable prediction of noncoding RNAs. Proc. Natl. Acad. Sci. U.S.A. **102:**2454-2459.
10. **Felsenstein, J.** 2005. PHYLIP (Phylogeny Inference Package) version 3.6. *Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.*
11. **Clamp, M., J. Cuff, S. M. Searle, G. J. Barton.** 2004. The Jalview Java alignment editor. Bioinf. **20**:426-427.
12. **Hihara, Y., A. Kamei, M. Kanehisa, A. Kaplan, M. Ikeuchi.** 2001. DNA Microarray Analysis of Cyanobacterial Gene Expression during Acclimation to High Light. Plant Cell **13**:793-806.