W28–W32 *Nucleic Acids Research, 2009, Vol. 37, Web Server issue* *Published online 11 May 2009*
doi:10.1093/nar/gkp354

BioBIKE: A Web-based, programmable, integrated biological knowledge base

Jeff Elhai[1], Arnaud Taton[1], JP Massar[2], John K. Myers[3], Mike Travers[4], Johnny Casey[3], Mark Slupesky[2] and Jeff Shrager[4,5,*]

[1]Center for the Study of Biological Complexity, Virginia Commonwealth University, Richmond VA, [2]Berkeley CA, USA, [3]Sequoia Consulting, North Hills, [4]CollabRx, Inc., Palo Alto and [5]Symbolic Systems Program (consulting), Stanford University, Stanford, CA, USA

Received January 31, 2009; Revised April 15, 2009; Accepted April 23, 2009

ABSTRACT

BioBIKE (biobike.csbc.vcu.edu) is a web-based environment enabling biologists with little programming expertise to combine tools, data, and knowledge in novel and possibly complex ways, as demanded by the biological problem at hand. BioBIKE is composed of three integrated components: a biological knowledge base, a graphical programming interface and an extensible set of tools. Each of the five current BioBIKE instances provides all available information (genomic, metabolic, experimental) appropriate to a given research community. The BioBIKE programming language and graphical programming interface employ familiar operations to help users combine functions and information to conduct biologically meaningful analyses. Many commonly used tools, such as Blast and PHYLIP, are built-in, allowing users to access them within the same interface and to pass results from one to another. Users may also invent their own tools, packaging complex expressions under a single name, which is immediately made accessible through the graphical interface. BioBIKE represents a partial solution to the difficult question of how to enable those with no background in computer programming to work directly and creatively with mass biological information. BioBIKE is distributed under the MIT Open Source license. A description of the underlying language and other technical matters is available at www.Biobike.org.

INTRODUCTION

Research in all areas of biology has come increasingly to rely upon massive sets of digital data and knowledge, the manipulation of which places most researchers outside their area of comfort. Despite a spectacular range of resources available to analyze biological information (witness this issue of NAR), biological problems still often require the development of novel methods. Existing tools may display results that are easy for humans to read, but they generally do not deliver them in a form that is useful for subsequent computations. Biologists without programming expertise (no doubt the majority) muddle through as best they can, using isolated tools and spreadsheets, or seeking the help of programmers. In the latter case, the resulting division of knowledge is far from ideal, obscuring the process from the biologist's view and making it difficult to understand the meaning of the results. Moreover, the biologist loses easy access to surprising intermediate results, which are at the heart of fundamental accidental discoveries (Elhai *et al.*, manuscript submitted for publication).

BioBIKE (the Biological Integrated Knowledge Environment; formerly BioLingua, 1) has been developed to allow researchers without programming expertise to combine tools, data and knowledge in ways demanded by the biological problem at hand. BioBIKE is composed of three integrated components: (i) a biological knowledge base, (ii) a graphical programming interface and (iii) an extensible set of tools that can be combined in novel ways.

BioBIKE INSTANCES AND THEIR KNOWLEDGE AND DATA BASES

A BioBIKE instance provides a framework for all available information needed by a given research community (Table 1), including sets of genomic sequences, gene annotations, functional descriptions, formal categories (e.g. COG), hierarchical groupings of metabolic reactions linked with genes (from KEGG, 2) and internal tables of Blast scores to support rapid protein comparisons. In addition, an instance may be stocked with experimental data, such as results from microarray or proteomic experiments. Indeed, any data that can be put into a standardized form, such as a table or XML structure, can be

*To whom correspondence should be addressed. Tel: +1 650 380 6306; Email: jshrager@stanford.edu

integrated into the knowledge-base (in simple cases through built-in resources, otherwise with the help of BioBIKE engineers). All of this knowledge and data are represented in an integrated manner within the BioBIKE frame system.

The availability of integrated data and knowledge on the same server makes possible certain operations that are not practical with data that is distributed across the web. For example, in CyanoBIKE it is a simple matter to find all proteins common to one set of organisms (perhaps user-defined) but not in another, for example those in N_2-fixing cyanobacteria that are not found in non-N_2-fixing cyanobacteria. Protein similarities and orthologs amongst proteins of organisms outside the database are also available using the same interface, albeit more slowly, through services such as NCBI's Blast (3).

THE BioBIKE GRAPHICAL PROGRAMMING INTERFACE

Checking the VPL (visual programming language) box at the login screen of a BioBIKE instance brings the user to the graphical programming interface. (Users may also access BioBIKE with scripts through a web-based command line interface described in ref. 1.) An example of the function palette and workspace is shown in Figure 1. BioBIKE functions and other constructs are represented by boxes obtained from pull down menus. These may be moved around by familiar actions such as drag-and-drop and copy-paste to form complex expressions. When completed, expressions may be executed by double-clicking them. Results are returned (and sometimes displayed in a human-readable format) so that the user can assess the effect of each step. Data, and whole sessions, may be saved to the BioBIKE server so that incomplete work can be continued later.

The design of the BioBIKE language adheres to these principles:

Intelligibility: an expression should be intelligible to someone with requisite biological knowledge but no prior experience with BioBIKE. Many concepts of molecular biology, such as codon and ortholog, are incorporated into the language.

Computability of results and nesting: BioBIKE functions often display results formatted for human comprehension. In addition to this, functions generally return their results in a form that can serve as input for further analysis. This allows users to compose expressions by taking the result of one function and feeding it into another, producing new results at each turn. This process can be abbreviated by nesting expressions together, as shown in Figure 2.

Small working vocabulary: expressions that are related to each other have been brought together within a single function, to reduce the burden on the memory of a new user. For example, the function SEQUENCE-SIMILAR-TO performs all flavors of Blast, or finds sequences differing from a reference by a given number of mismatches, depending on options specified by the user.

Implied iteration: the size of biological databases often makes it necessary to perform iterative operations (i.e. loops). Such operations in conventional languages are the bane of those new to programming. Most BioBIKE functions iterate automatically. In Figure 1, for example, a specific gene could be given to the ORTHOLOG-OF function or a list of genes could be given instead. In the latter case, the function returns a list of results, one for each gene.

Table 1. Current BioBIKEs[a]

CyanoBIKE: Cyanobacteria (42 genomes)
ParaBIKE: Eukaryotic parasites (5 genomes)
StaphyloBIKE: Staphylococcus (45 genomes)
StreptoBIKE: Streptococcus (25 genomes)
ViroBIKE: Viruses (1797 genomes, 20 metagenomes)
BIKE: Used for education (0 genomes)

[a]All instances are available through biobike.csbc.vcu.edu

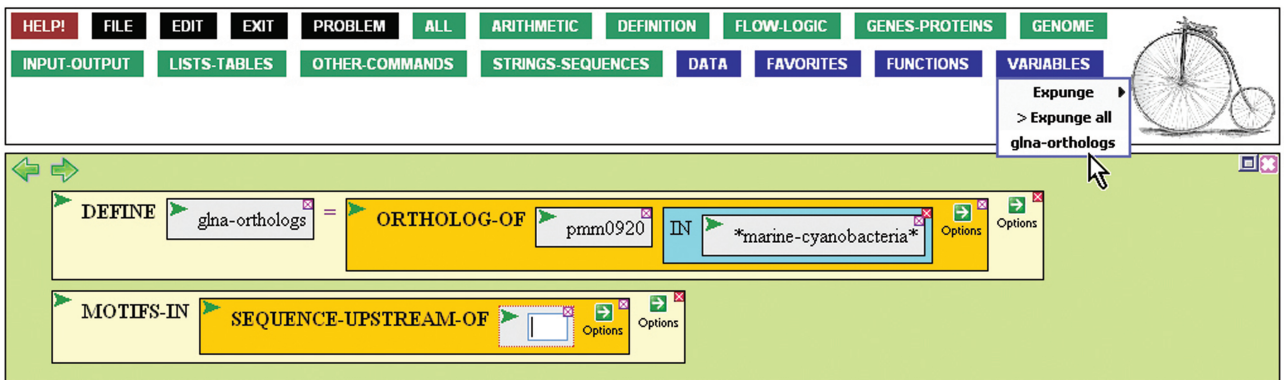

Figure 1. BioBIKE function palette and workspace. The green workspace shows the work of a user looking for a regulatory sequence upstream from a gene, by focusing on sequences common amongst upstream sequences of orthologous genes in related organisms. The first function defines the variable gln-orthologs as the set of orthologs in marine cyanobacteria of a gene the user knows to encode glutamine synthetase. The second function is in the midst of being completed. The user is choosing the newly defined variable from the VARIABLES menu to be inserted into a function that will extract the sequences upstream from all the orthologs and then find statistically overrepresented sequences within the set of sequences, using MEME (6).

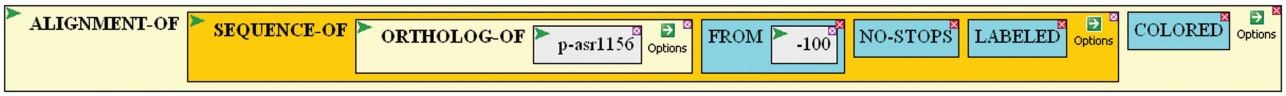

Figure 2. Example of a nested function. The function makes an alignment of the sequences of all orthologs of the protein Asr1156, starting as many as 100 amino acids before the nominal beginning of the protein but going backwards only up to the first stop codon. The sequences are labeled with the name of the protein and aligned, using Clustal (5) and visualized using JalView (19). This is the code used to generate an alignment (discussed in Elhai, Taton, Massar, and Shrager, manuscript submitted for publication) that provides evidence against existing annotations of a family of conserved genes and for the use of nonstandard start codons in cyanobacteria.

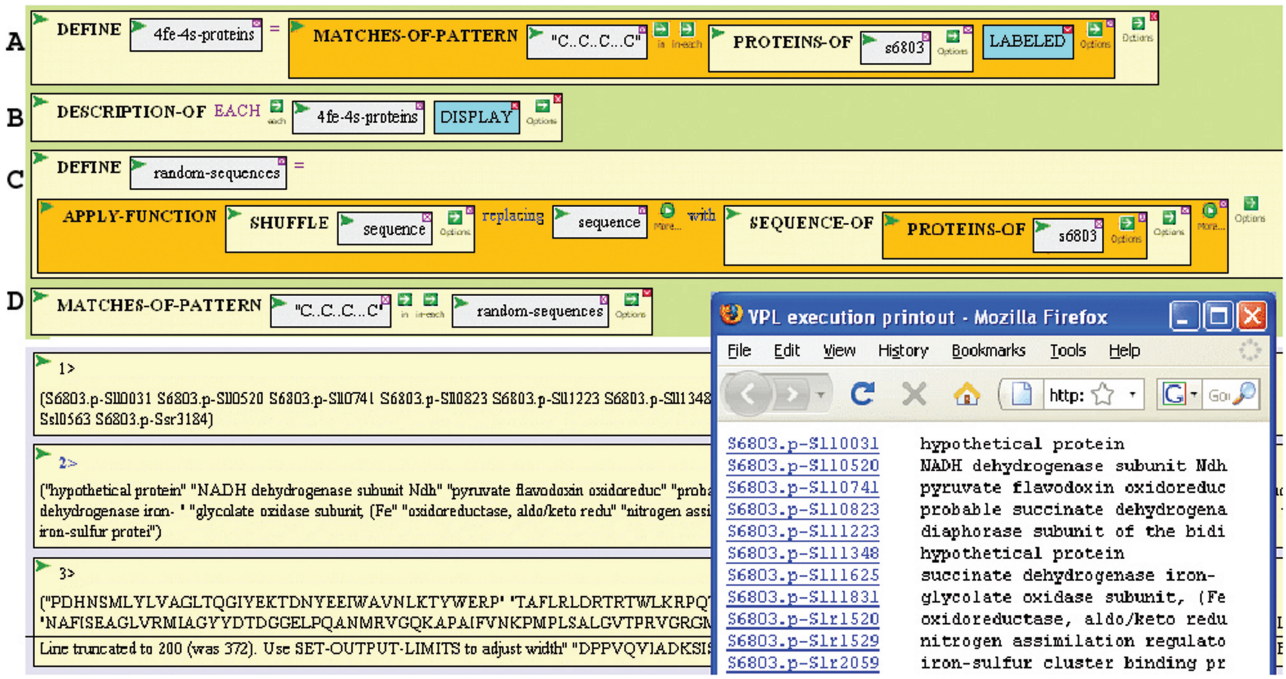

Figure 3. Example of progressive evaluation and iteration in BioBIKE. The pattern of four cysteine residues separated by 2, 2 and 3 amino acids is often found in proteins with 4Fe-4S clusters (20). (A) The first function finds the pattern of cysteines amongst the sequences of all proteins in the cyanobacterium Synechocystis PCC 6803 and assigns the names of the proteins bearing the motif (Result 1) to a user-defined variable called 4fe-4s-proteins. (**B**) The annotation for each of the proteins is displayed in a separate window (see inset), and the annotations are also returned as result #2. (**C**) The user is concerned that this motif might well arise by chance on some proteins of *Synechocystis*. To test this, a set of random protein sequences is generated, each element being a random shuffling of a real protein sequence. The set is assigned to the variable random-sequences, and the random sequences are returned as result #3. (**D**) This set of random sequences is searched for the characteristic motif, and none are found (no result), lending some confidence to the belief that the presence of the motif in proteins of *Synechocystis* is of biological significance.

Extensibility: users can define new data and functions which immediately enter the language, becoming instantly accessible through the same sort of menus as built-in objects. In addition to serving as a memory aid, this affords the modular addition of concepts into the language itself. Users not satisfied with the names of concepts built into the language can readily build a private vocabulary if desired.

Although specialized for bioinformatics, BioBIKE is built on top of the standard computer language Lisp, and is therefore capable of all operations typical of a general purpose programming language. Behind the scenes, BioBIKE expressions are translated into Lisp and compiled, yielding code that runs at a speed comparable to that of C code. Lisp is a uniquely powerful language, often used to create new specialized languages, as we have done here. R, for example, is written on top of Scheme, a dialect of Lisp (4). Lisp is also the language of choice for artificial intelligence, which continues to inform BioBIKE's development.

BioBIKE TOOLSET AND ADVANCED FACILITIES

BioBIKE provides access to several programs that are commonly used: Blast (3), for sequence searches); Clustal (5), for multiple sequence alignments); Meme (6), for motif discovery; RNAz (7), for discovery of conserved RNA sequences; and Phylip (8), for construction of phylogenetic trees. All are accessed through the same interface, greatly reducing the need to figure out the idiosyncrasies of each resource. Useful tools not already in the language that have Application Programming Interfaces (APIs), or that are capable of running within a Linux

environment can generally be added to BioBIKE on request with little difficulty, and thus be made accessible to BioBIKE users through the standard graphical programming interface.

LEARNING BioBIKE AND STYLES OF BioBIKE USAGE

Online tours of BiobIKE are accessible through the BioBIKE portal (biobike.csbc.vcu.edu), and a tour of the resources of the interface and the basic conventions of the language is available through the HELP button. A tour that describes how BioBIKE can be used in motif discovery is included in the Supplementary Material.

BioBIKE expressions are often intelligible when read, but new users do not find them easy to write. Those new to BioBIKE often begin by using it as a simple query language, asking, for example: '*What is the sequence of my favorite gene?'* From there, one might construct a progressive series of queries, each one utilizing on the result of the previous, for example: '*What are the orthologs of the sequence of my favorite gene?*' '*What are the upstream sequences of those orthologs?*' '*What common sequence motifs are found in those upstream sequences?*' This progression of questions might have led to Figure 1.

This progressive evaluation style is critical for programming novices (9), and one may continue indefinitely within this style, obtaining useful results. However, it is also possible to create more complex structures from simple elements, facilitated by drag-and-drop and copy-paste operations. Figure 3 provides an example of iteration mixed in with sequential evaluation. Since each simple element may be executed independently by double-clicking on it, users may still examine the intermediate results, even within complex expressions.

Complex expressions or sub-expressions can also be collapsed visually into single boxes, making it easier to grasp the larger picture. Moreover, as mentioned above, BioBIKE itself is extensible: if a user should devise a complex expression that might be of continued utility, the expression can be packaged, given a unique name, and made accessible via a menu, no differently from any other BioBIKE function. In this way, complicated operations can be broken up into logical chunks and subsequently offered as distinct functions.

CONCLUSION

BioBIKE represents a novel paradigm regarding the interaction of biologists with information of interest to them. Its goal is to put the analysis of large amounts of information directly into the hands of biologists themselves—to enable them to manipulate biological knowledge and data in an interactive computational environment. This offers extraordinary power to biologists with little computational background. BioBIKE has already made possible a deep analysis of proteomic data (10), a cross-genomic analysis of repeated sequences (11), and the introduction of many dozens undergraduates and high school students to biological analysis on the computer (Elhai, unpublished results).

Some excellent web-based resources, such as Entrez (12), provide convenient access to sequences and other information. BioBIKE does the same, but the information is returned in a form that may be used immediately for further analysis. Still other resources, such as IMG (13) and the NCBI implementation of Blast (14) provide a good interface for the analysis of sequences with a fixed set of tools. Some, e.g. Taverna (15) and Galaxy (galaxy.psu.edu), go a step further and facilitate the creation of a work flow using a fixed set of tools (fixed to those unfamiliar with computer programming). BioBIKE does these things as well, but does not confine the user to follow predetermined channels. The user new to programming may use existing tools or combine basic functions to create ways to answer questions for which tools do not exist. Such flexibility has previously required one to employ conventional programming languages, sometimes supplemented with bioinformatic add-ons, such as BioRuby (bioruby.org) or BioPerl (16). BioBIKE is equally powerful but does not require the user to learn the underlying language. BioBIKE is aimed at biologists who do not wish to expend the effort required to learn a conventional programming language but who wish to have the same hands-on relationship with informational objects of study as they do with objects in the laboratory.

BioBIKE is a first step in a new direction, and although even in its present state it is a powerful tool, it must be stressed that the goal of intuitive use remains unmet. Users should not expect to figure out BioBIKE as they would a simple web-based tool that offers a small number of functions. A greater set of tours and help pages may increase the ability of naïve users to exploit the resource independently. However, our current direction is more ambitious. We plan to extend BioDeducta (17), which combines BioBIKE with an automated reasoning system, to enable users to present BioBIKE with a natural language question (e.g. 'Is there a common sequence motif found upstream of orthologs of *glnA?'*), and through a series of natural language interactions, arrive at a BioBIKE expression that answers the question.

SOFTWARE AVAILABILITY AND COMPATIBILITY

At the time of writing, there are five BioBIKE instances freely available through the web (Table 1). BioBIKE is written in Common Lisp, operating within the KnowOS paradigm (18), and is distributed under the MIT Open Source license. Although it is freely available for anyone to download and install (see www.BioBIKE.org for instructions), we encourage users to use already-existing servers. The authors are happy to discuss collaboration with communities of biologists who would like to create BioBIKE instances particular to sets of model organisms. At present, the graphical interface is only operational within Firefox 1.5 and above.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We thank Andrew Pohorille of NASA's Division of Astrobiology and Fundamental Biology who provided seed support for BioLingua/BioBIKE. Others who have contributed significantly to this work include Pat Langley, Marc Santoro, Michiko Kato, James Mastro, Bogdan Mihai, Emily Niman, Craig Noe, Peter Seibel, Hien Truong, Andy Whittam, Richard Waldinger, Carolyn Talcott and Merill Knapp. Richard Waldinger and several anonymous reviewers provided valuable comments on various drafts of this article.

FUNDING

National Science Foundation (DBI-0516378, DBI-0850146 to J.E.); the National Aeronautics and Space Administration (JRIs: NCC2-5555, NCC2-5462, NCC2-5471 to J.S.); software grants from Franz, Inc. and LispWorks, Inc. Funding for open access charge: Jeff Shrager.

Conflict of interest statement. None declared.

REFERENCES

1. Massar,J.P., Travers,M., Elhai,J. and Shrager,J. (2005) BioLingua: a programmable knowledge environment for biologists. *Bioinformatics*, **21**, 199–207.
2. Kanehisa,M., Goto,S., Hattori,M., Aoki-Kinoshita,K.F., Itoh,M., Kawashima,S., Katayama,T., Araki,M. and Hirakawa,M. (2006) From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, **34**, D354–D357.
3. Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
4. Ihaka,R. and Gentleman,R. (1996) R: a language for data analysis and graphics. *J. Comput. Graphical Stat.*, **5**, 299–314.
5. Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.
6. Bailey,T.L., Williams,N., Misleh,C. and Li,W.W. (2006) MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Res.*, **34**, W369–W373.
7. Washietl,S., Hofacker,I.L. and Stadler,P.F. (2005) Fast and reliable prediction of noncoding RNAs. *Proc. Natl Acad. Sci. USA*, **102**, 2454–2459.
8. Felsenstein,J. (2005) *PHYLIP (Phylogeny Inference Package) version 3.6*. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.
9. Green,T.R.G. and Petre,M. (1996) Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *J. Visual. Lang. Comput.*, **7**, 131–174.
10. Ow,S.Y., Cardona,L., Taton,A., Magnuson,A., Lindblad,P., Stensjö,K. and Wright,P.C. (2009) Quantitative overview of N2 Fixation in Nostoc punctiforme ATCC 29133 through cellular enrichments and iTRAQ shotgun proteomics. *J. Proteome Res.*, **8**, 187–198.
11. Elhai,J., Kato,M., Cousins,S., Lindblad,P. and Costa,J.L. (2008) Very small mobile repeated elements in cyanobacterial genomes. *Genome Res.*, **18**, 1484–1499.
12. Maglott,D., Ostell,J., Pruitt,K.D. and Tatsuova,T. (2007) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res.*, **35**, D26–D31.
13. Markowitz,V.M., Szeto,E., Palaniappan,K., Grechkin,Y., Chu,K., Chen,I-M.A., Dubchak,I., Anderson,I., Lykidis,A., Mavromatis,K., Ivanova,N.N. and Kyrpides,N.C. (2008) The integrated microbial genomes (IMG) system in 2007: data content and analysis tool extensions. *Nucleic Acids Res.*, **36**, D528–D533.
14. Johnson,M., Zaratskaya,I., Raytselis,Y., Merezhuk,Y., McGinnis,S. and Madden,T.L. (2008) NCBI Blast: a better web interface. *Nucleic Acids Res.*, **36**, W5–W9.
15. Hull,D., Wolstencroft,K., Stevens,R., Goble,C., Pocock,M.R., Li,P. and Oinn,T. (2006) Taverna: a tool for building and running workflows of services. *Nucleic Acids Res.*, **34**, W729–W732.
16. Stajich,J.E., Block,D., Boulez,K., Brenner,S.E., Chervitz,S.A., Dagdigian,C., Fuellen,G., Gilbert,J.G., Korf,I., Lapp,H. *et al.* (2002) The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618.
17. Shrager,J., Waldinger,R., Stickel,M. and Massar,J.P. (2007) Deductive Biocomputing. *PLoS ONE*, **2**, e339.
18. Travers,M., Massar,J.P. and Shrager,J. (2005) The (re)birth of the knowledge operating system. *Proceedings of the International Lisp Conference*. Assoc. of Lisp Users (ALU), Stanford, CA, pp. 357–365.
19. Clamp,M., Cuff,J., Searle,S.M. and Barton,G.J. (2004) The Jalview Java alignment editor. *Bioinformatics*, **20**, 426–427.
20. Alhapel,A., Darley,D.J., Wagener,N., Eckel,E., Elsner,N. and Pierik,A.J. (2006) Molecular and functional analysis of nicotinate catabolism in *Eubacterium barkeri*. *Proc. Acad. Sci. USA*, **103**, 12341–12346.