DDT: BIOSILICO Vol. 2, No. 4 July 2004

Codifying bioinformatics processes without programming

Thomas C. Hudson, Ann E. Stapleton and Jeffrey L. Brown

Bioinformatics is bimodal, with results gathered either from rigid pipelines, their functions encoded in programs accessible only to specialists, or from ad hoc collections of scripts or web pages, used once, then lost track of or thrown away. We discuss the possibility of developing tools to straddle this divide, in particular, tools using graphical dataflow programming, which have the potential to make bioinformatics flexible, approachable and repeatable.

Thomas C. Hudson
Department of Computer Science
University of North Carolina at Wilmington
601 South College Rd
Wilmington
NC 28403, USA
e-mail: hudsont@uncw.edu.
Ann E. Stapleton
Department of Biological Sciences
University of North Carolina at Wilmington
601 South College Rd
Wilmington
NC 28403, USA
Jeffrey L. Brown
Department of Mathematics and Statistics
University of North Carolina at Wilmington
601 South College Rd
Wilmington
NC 28403, USA

▼ The widespread application of computers to genomics has produced an explosion of information, yet this information is not knowledge. Bioinformatics has had only limited success translating these vast quantities of information into knowledge in order to answer real questions and provide results directly applicable to research questions.

Many modern bioinformatics tools require a biologist to learn a query language to ask a novel question; this can be difficult and, consequently, the bioinformatic tools have been infrequently used in academic publications. Although the AceDB biological database system has been under development since 1989, it was used throughout the *Caenorhabditis elegans* sequencing project and has been made publicly available by the Sanger Institute (http://www.acedb.org). PubMed (http://www.ncbi.nlm.nih.gov/PubMed/) lists remarkably few papers using the AceQuery software, or the various other tools with graphical user interfaces intended to help biologists build tables and queries (e.g. TableMaker) [1], despite their availability on several plant genome database websites. Plant database feedback forums for recent generations of plant databases, such as MaizeDB (now replaced by Maize GDB; http://www.maizegdb.org), contained numerous comments about the difficulty of finding information, even after redesigns intended to improve usability.

One reason biologists find the learning curve for bioinformatics tools so steep is that the query languages and programming languages have a severe impedance mismatch for biologists – these languages deal in concepts that do not match the biologists' model of the domain of inquiry. Biological concepts are at a much higher level than extant programming languages [2].

For example, we want to ask questions like 'Of the genes that are found in only one species, what proportion cannot be assigned to a protein family and how does that proportion compare with the proportion of unassigned genes in the entire dataset?' Another example is 'What is the relative frequency of the various heat-shock proteins (HSPs) in various species?' A rough translation of the first of these questions into a query language begins with:

Select count (*) from sequence, goannotation where sequence.sequenceid = goannotation.sequenceid and goannotation.type = 'functional class' group by sequence.speciesid, goannotation.value.

Neither this structured-query language (SQL) translation nor any other translation we have shown to biologists seems to have anything to do with the original question. Even computer scientists have trouble reducing queries like 'Return accession numbers and definitions of GenBank expressed-sequence tag sequences that are similar (60% identical over 50AA) to calcium channel sequences in SwissProt that have references published since 1995 and mention brain' [3] to code, or relating the code back to the original query. One strategy for answering the query requires access to three websites – PubMed, SwissProt and GenBank – and the BLAST tool.

Observations of biologists using these tools show that they frequently prefer to use

1741-8364/04/$ – see front matter PII: S1741-8364(04)02410-2

reasoning by analogy, for which the extant tools are not well-suited. They find one query that works and attempt to modify it to obtain a different result. This only works for very small, well-controlled changes. For example, a small change in the SQL statement can have large, unanticipated effects that require understanding the theory of relational databases, in addition to the underlying structure of the data; i.e. the behavior expected from the database by a computer scientist is not the same behavior expected by a biologist.

Approaches to sequence analysis

Bioinformatics embrace a broad range of questions and a wide set of approaches to answering questions. Classical high-throughput genome sequencing and analysis work have brought us the pipeline model, where one question is to be answered or one analysis is to be performed for many samples. This creates an emphasis on producing highly efficient software, which in turn emphasizes careful design and analysis of the bioinformatics apparatus. One well-known example of this type is Ensembl (http://www.ensembl.org/) [4], a complicated software system for annotating eukaryotic genomes whose developers described the early versions as inflexible. Although Ensembl can perform many analyses, it requires customized programming to change the questions it asks. Many laboratories have simpler pipelines built from ad hoc computer programs – scripts – that invoke tools and translators in some predefined sequence.

Simultaneously, a browsing or discovery method of bioinformatics has evolved, where an investigator is interested in finding out all the available information about one particular sample. Historically, this has been the province of tools made available on web pages, with users pasting results from one website into a dialog box on the next, or into an Excel spreadsheet for long-term storage. A human solution to the third sample query above would require this approach: the investigator finds references in PubMed, copies the sequence identification numbers from that browser window into another window which is open to SwissProt, then copies the sequences returned by SwissProt into a browser window which is open to a BLAST service on GenBank. Systems such as BioMoby [5] and ISYS [6] have recently attempted to tie together these individual tools and provide workflow support.

The most important question for a scientist beginning a new investigation is determining which approach will work best. Configuring a pipeline requires time from a specialist programmer, slowing down the rate at which the biologist can use the answers to initial questions in order to formulate new questions, whereas the fast turnaround of an exploratory approach typically yields few results without recording methods or scaling to large numbers of samples.

We believe that it is time to recognize a middle way between these two approaches. There are investigations that demand the ability to ask several related questions about large sets of samples, with more flexibility than is afforded by a pipeline, but a higher-level view of the process than those afforded by a discovery tool. We refer to this as 'intermediate-scale' bioinformatics.

Although exploration works well for asking questions about single samples, most exploratory tools do not scale well to questions about larger quantities. We know a biologist might need an intermediate-scale approach when they ask 'How many?' as a starting point, rather than as an end point, of their query. An intermediate-scale tool might also be necessary when dealing with large populations about which the scientist wishes to ask a statistical question.

Because intermediate-scale tools manipulate a representation of the discovery process, they can also be useful for exploratory queries. The benefits might not be obvious when the query is first run, but instead show up some weeks later. Databases are updated with new sequences or new species, and/or algorithms with new parameters or optimizations; scientists need to be able reapply queries they have previously developed. But these changes in data and algorithm usually require changes in the query, encouraging iterative development of a complex representation of the query rather than the templated reuse of some web-accessible tools or the static reuse provided by pipelines. Iteration is particularly crucial in plant genomics, where most important crop species do not and will not have completely sequenced genomes.

The criteria for using an intermediate-scale approach can be summarized, as follows: (i) you are not sure in advance exactly which question to ask, or whether a question is worth asking; (ii) if the question turns out to have been worth asking, then it is worth knowing how you arrived at the answer; and (iii) if the question turns out to have been worth asking, it is likely to be worth asking again later, asking in a slightly different context or asking other similar questions.

Bioinformatics without programming

Bioinformatics is not the first discipline to have faced the problem of useful techniques only being accessible to specialists cross-trained in computer science. Scientific visualization of earth and space sciences data or of engineering simulations reached this point in the late 1980s. Many scientists were capable of generating or accessing the data required, and wished to ask questions that were most easily

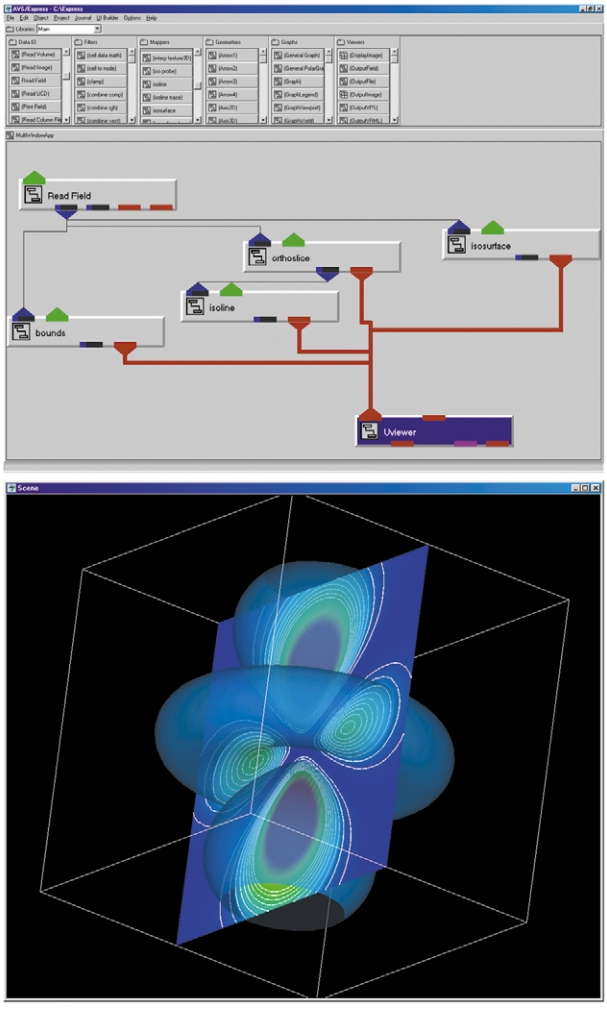

Figure 1. Visualization of electron orbitals and the dataflow graph that creates it. A graphical dataflow program for scientific visualization **(a)** and the image it creates **(b)**. The AVS/Express program (http://www.avs.com) loads a dataset from a disk, then creates a bounding box, an isosurface, an orthoslice (a slice through the volume orthogonal to one of the axes of the dataset), and isolines (contour lines) on the orthoslice, displaying all four elements with a 'Uviewer' rendering module.

answered through visualization. However, they could not just use any visualization techniques best suited to their queries. There were no common frameworks or integration packages for scientific visualization accessible to non-programmers. Therefore, to visualize their data, the domain scientists also needed to master complex, cutting-edge computer programming. Scientific visualization then became a discipline that could only be practiced by a small cadre of highly trained specialists.

This problem has been successfully solved by software packages like AVS/Express (http://www.avs.com) and OpenDX [7]. These programs allow a scientist to use graphical dataflow programming to specify the procedure needed to make a visualization, composing it from simple reusable modules. The scientist needs to understand the data, have a clear goal for the visualization (a question they want to answer) and understand the possible techniques for visualizing the data. They do not need to understand how to program the visualization algorithms (see Figure 1 for an example).

In the computer science community, research into graphical dataflow languages is considered passé because they are not generally applicable. However, for domains with certain characteristics, an appropriate mapping to graphical notations, a limited range of problem scales and sufficiently high-level constructs, visual programming languages are appropriate [8]. Scientific visualization satisfied these criteria and bioinformatics does as well.

What is necessary to build a dataflow system for bioinformatics?

There are three main factors for building a dataflow system: (i) a data model; (ii) a dataflow engine and programming interface; and (iii) a vocabulary of high-level modules or processes to be applied to the data.

The first of these is the most difficult and represents the largest difference between bioinformatics and scientific visualization. We will investigate the way that various dataflow systems have addressed this problem. The second is a relatively simple program to design and implement, and there are open-source examples available today. The third requires programmers to prepare modules in advance. What sets this programming effort apart from scripting and other one-off bioinformatics projects is that the dataflow engine's application programming interface (API) enforces reusability: future queries will be able to reuse all past modules, combining them – without doing any new programming – in new ways to answer new questions.

Systems and infrastructures

Most bioinformaticians are familiar with, and even used to, heavyweight pipelines. For example, ENSEMBL, which attempts to be applicable to any eukaryotic genome [4], and the Mouse Genome Informatics database, which is very specialized for a single species [9].

Pipelines are notorious for their inflexibility. This stems from the fact that they 'represent' the steps of the process as code. Unfortunately, code is not a suitable medium for communication to or manipulation by a non-specialist biologist. Exploratory systems like ISYS and BioMoby are highly flexible, but do not yet include a representation of process. Software for intermediate-scale bioinformatics needs to support moderately large queries and databases

(tens or hundreds of thousands of sequences) and a moderate level of planning. That is, the steps of the experiment to be carried out are first-class values, which are explicitly represented within the computer, allowing plans to be constructed by the user and manipulated by the software [10].

There have been attempts to commercialize tools that do provide a representation of the process of bioinformatics experimentation. Science Factory's Übertool (http://www.science-factory.com/) and INCOGEN's VIBE (http://www.incogen.com/) are two examples of large integrated systems using graphical dataflow programming.

Science Factory ceased operations while this manuscript was in preparation. At the time it closed, Übertool comprised >750 000 lines of code, the result of 24 programmer-years of effort. It implemented >150 analysis modules, including standards such as BLAST, multiple alignment and Hidden Markov models, which were extensible using a custom scripting language. Übertool included support for distributing queries across multiple servers.

VIBE is a client-server program, designed for a multi-tiered environment, ideally with multiple database servers including a DeCypher query accelerator. Like Übertool, it provides implementations of many standard bioinformatics algorithms. A Java API allows users to write their own modules and enables them to be independent of the tool's underlying data schema. Implementers of new analysis techniques need to extend the four Java classes provided by INCOGEN.

There are also open-source frameworks that can easily be adapted to support the building of a new system. Three examples at different levels of complexity are: (i) Ptolemy (http://ptolemy.eecs.berkeley.edu/ptolemyII/), a graphical dataflow engine intended for simulation of electrical engineering phenomena, but equally applicable to bioinformatics; (ii) PISE [11], a toolkit for building web interfaces to traditional UNIX-script-style bioinformatics tools and linking them together; and (iii) JGraph (http://jgraph.sourceforge.net/), a framework for building user interfaces that can handle dataflow graphs.

Scientific visualization proceeds with relatively straightforward data models, although their metadata problem is complex [12]. Bioinformatics has a much more complicated data representation task to overcome. The Gene Ontology consortium is making progress towards a standard controlled vocabulary [13], but a standardized data schema is difficult to achieve. With the current proliferation of project-specific databases, it appears that there is no schema of manageable complexity that will satisfy the needs of every investigation.

Large pipeline projects intended for one specific purpose can carefully design and specify a data model adequate to their purpose. Übertool followed this approach, with a very broad and complete schema. This requires that any organization that wants to use Übertool first translate its data into the tool's schema. VIBE is more focused on the analysis of sequence data, particularly similarity search algorithms.

Ptolemy supplies a dataflow engine that is designed for a discrete data model, where each separate field from a database query flows through a separate path in the dataflow graph. This is a decided weakness of the Ptolemy-based prototype dataflow tool we have built; for the biologists, it makes the graphs more visually complex and requires a deeper understanding of the process, whereas for the computer scientist it interferes with useful optimizations.

Because JGraph is a framework for building only the user interface component of the system, it does not compel use of any particular data model. We have successfully built a JGraph-based dataflow tool with a minimalist integrated data model; query results are manipulated by the system without any expectation of the presence or absence of particular fields.

Reflections on the state of the art

Biologists have embraced visual languages that represent the biological objects and processes they study. A simple example would be the use of taxonomic trees. More recently, BioD notation describes the function of complex biological systems, such as the activation of cyclin DI gene expression [14]. Visual languages have also been used to represent a sequence of procedures in a laboratory (e.g. http://labscape.cs.washington.edu/). However, visual languages that describe the process of study – how analyses bring together multiple data sources and computational techniques to provide new information – have not yet been widely adopted.

Our group at the University of North Carolina at Wilmington believes that there is a real need for a representation of the process by which scientists conduct their inquiries, that visual languages are well-suited to this application, and that there is a great deal of room for innovation in the underlying computer science. In our QueryConnect project (Figure 2, http://www.queryconnect.org/), there are three goals. First, we aim to support the 'middle ground'. A tool should make it easier to assemble a workflow than it is to configure an Ensembl pipeline, but provide more support for this high-level view of a bioinformatics investigation being a planned experiment – a defined sequence of steps – than exploratory tools like BioMoby or ISYS. For the query reuse scenarios, logging and replay is essential. Every analysis that the scientist executes needs to be permanently recorded, and this recording

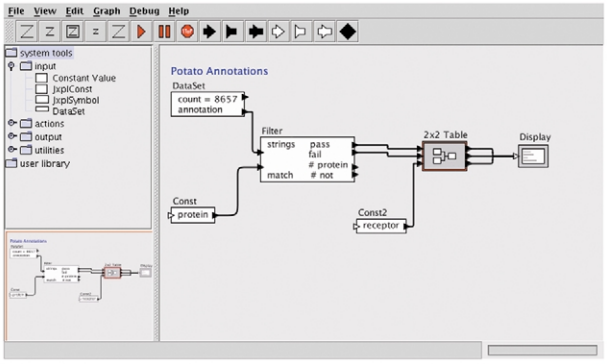

Figure 2. A bioinformatics dataflow prototype built on the Ptolemy II infrastructure. This graph uses a 2×2 table statistical test to determine whether the number of proteins annotated as receptors is significantly different from the total number of known proteins. The 2×2 table is a macro built by a statistician from other dataflow modules and added to the list of possible modules displayed to our biologists.

must include the methods used in the analysis, not just the results. The majority of contemporary bioinformatics proceed without a reliable record – results from one website are copied and pasted into another website, with the final results printed for a laboratory notebook, but no record is kept of the parameters such as BLAST settings or database versions used. This makes many bioinformatics results nonreproducible. Although visual languages are not an absolute prerequisite for logging and replaying analyses, they do a much better job than text in rapidly conveying the relationship between different phases of an analysis when the procedure is branching.

Finally, writing a general-purpose tool for use beyond our own immediate circle of collaborators, we have to face the question of data schemas and ontologies. A well-defined schema increases the power of the tool, but reduces its scope of applicability. Instead, we are working on tools that discover or adapt to database schemas. Scientific visualization has a fairly standard taxonomy of data types which is regular and thus easily handled by a computer program (Box 1). Biological data are much more complex. Standards for some datatypes are beginning to emerge [e.g. Minimum Information About a Microarray Experiment (MIAME); http://www.mged.org/Workgroups/MIAME/miame.html], but there is no overarching framework that allows all these standards to be manipulated together. Some data are text and some are numerical; what is metadata for some analyses is crucial data for others. Because biological sequence data are so much richer than the data used for scientific visualization, we cannot write software to automatically handle any and all biological data independent of its schema to date, and so biological scientists will need to have some understanding of the data they intend to analyze.

The idea of an intermediate-scale tool is certainly an appealing one. It lets the scientists ask more questions without waiting for the programmers, and lets the programmers concentrate on new reusable modules instead of re-implementing slight variations on previous queries. Surprisingly, the toughest part of deploying intermediate-scale tools has been training scientists used to *ad hoc* methods, whose expectations of bioinformatics have often been lowered to questions no more complex than 'find the sequences that satisfy property x'.

Intermediate-scale bioinformatics systems should enable reactive science, where the classical cycle of experiment planning, experimentation, data analysis and planning subsequent experiments becomes so compressed to permit a qualitative change in the way science is done. In atomic force microscopy, where this cycle has historically taken several days, tools like the nanonanipulator have tied all the stages of science together, allowing scientists to understand novel phenomena as they are first observed, and immediately adapt the experiment to isolate and describe the phenomena (http://www.nanomanipulator.com/NanoManipulator.htm) [15]. It is this flexibility to respond rapidly to new discoveries by varying queries and reapplying old analyses in new ways that we hope will accelerate knowledge production in bioinformatics.

Our group targets the software to handle analysis of plant genome sequences. That still leaves us with a great deal of additional work to have a full-featured research tool: (i) to support real-world use and connect to the many different plant sequence databases available, including TIGR gene indices [16], BioExtract, PlantGDB [1] and

Box 1. Data types used in scientific visualization

Data for scientific visualization can often be represented as a space of tensors. For example, a rank-zero tensor in one-dimensional space might be a series of temperature measurements at one point over time. A rank-zero tensor in two-dimensional space might be the set of high temperatures taken at every airport in a country. A rank-one tensor in two-dimensional space might be measurements of currents (direction and speed) across a body of water. A rank-two tensor in three-dimensional space might be the stress at various points on an airframe during landing.

All of these can be represented as a multidimensional array of multidimensional arrays of real numbers. Although different ranks and dimensionalities require different visualization techniques, the software infrastructure necessary to handle them is reusable [12].

Gramene [17]; (ii) to support expert users and expose an explicit interface to SQL; and (iii) to support use by research teams, in addition to providing the support for synchronous and asynchronous collaboration.

We used the JGraph framework to build the user interface for an initial prototype system, which communicated with a server using Enterprise Java Beans in IBM WebSphere application server and DB2 database. As a prototype without any engineering or optimization effort, this code base could perform analyses that manipulated 50 000 sequences in under six minutes. Like VIBE, new analysis modules could be added to the system by extending a Java class. The prototype proved difficult to scale to larger problems apparently as a result of memory management issues in the Java virtual machine used at the server. We are now implementing a distributed, Grid-enabled tool, with a user interface based on Ptolemy II. To avoid having to re-implement all of the standard analyses used by bioinformaticians, we are attempting to provide efficient reuse of pre-existing code (written in any language) that has been exposed as a Web Service (http://www.w3.org/2002/ws/). This tool uses XML to define sets of analyses and is described in Ref. [18].

Despite the amount of work required to build a system of this nature, our preliminary work gives us confidence that this is an approach worth pursuing. There is a fruitful mode of inquiry that lies between the heavyweight pipeline and free exploration, and a payoff from having an interface that breaks free of traditional structures to give biologists another way to model their questions.

References

1 Dong, Q. *et al.* (2004) PlantGDB, plant genome database and analysis tools. *Nucleic Acids Res.* 32, D354–D359

2 Kim, J. (2002) Computers are from Mars, organisms are from Venus. *IEEE Computer* 35, 25–32

3 Eckman, B.A. *et al.* (2001) Extending traditional query-based integration approaches for functional characterization of post-genomic data. *Bioinformatics* 17, 587–601

4 Clamp, M. *et al.* (2003) Ensembl 2002: accommodating comparative genomics. *Nucleic Acids Res.* 31, 38–42

5 Wilkinson, M.D. and Links, M. (2002) BioMOBY: An open source biological web services proposal. *Brief. Bioinform.* 3, 331–341

6 Siepel, A. *et al.* (2001) ISYS: A decentralized, component-based approach to the integration of heterogeneous bioinformatics resources. *Bioinformatics* 17, 83–94

7 Abram, G. and Treinish, L.A. (1998) An extended data-flow architecture for data analysis and visualization. In *Association for Computer Machinery. SIGGRAPH 98, Course Notes No. 35, Interactive Visualization and Web-based Exploration in the Physical and Natural Sciences,* Association for Computer Machinery

8 Citrin, W. (1996) Strategic directions in visual languages research. *Association for Computer Machinery Computing Surveys* 28, 4es, http://doi.acm.org/10.1145/242224.242392

9 Blake, J. *et al.* (2003) MGD: The Mouse Genome Database. *Nucleic Acids Res.* 31, 193–195

10 Watt, D.A. (1990) *Programming language concepts and paradigms,* Prentice Hall

11 Letondal, C. (2001) A Web interface generator for molecular biology programs in Unix. *Bioinformatics* 17, 73–82

12 Treinish, L.A. (1998) Unifying principles of data management for scientific visualization. In *Association for Computer Machinery, SIGGRAPH (Course Notes),* Association for Computer Machinery

13 The Gene Ontology Consortium (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet.* 25, 25–29

14 Cook, D.L. *et al.* (2001) A basis for a visual language for describing, archiving and analyzing functional models of complex biological systems. *Genome Biol.* 2, RESEARCH0012

15 Taylor, I. et al. (1999) Advanced interfaces to scanning probe microscopes. In *Handbook of Nanostructured Materials and Nanotechnology* (Nalwa, H.S., ed.), pp. 271–308, Academic Press

16 Quackenbush, J. *et al.* (2001) The TIGR Gene Indices: analysis of gene transcript sequences in highly sampled eukaryotic species. *Nucleic Acids Res.* 29, 159–164

17 Ware, D. *et al.* (2002) Gramene: a resource for comparative grass genomics. *Nucleic Acids Res.* 30, 103–105

18 Wood, M. *et al.* (2004) Toward a graphical user interface for grid services. In *IEEE Southeastern Conference.* Blacksburg, IEEE.