BIOINFORMATICS APPLICATIONS NOTE

Vol. 24 no. 3 2008, pages 450–451
doi:10.1093/bioinformatics/btm602

Databases and ontologies

Automated programming for bioinformatics algorithm deployment

Gil Alterovitz[1,2,3,*], Adnaan Jiwaji[2] and Marco F. Ramoni[1,3]

[1]Children's Hospital Informatics Program at the Division of Health Sciences and Technology, Harvard University and Massachusetts Institute of Technology, [2]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology and [3]Harvard Partners Center for Genetics and Genomics, Harvard Medical School, Boston, USA.

Received on July 28, 2007; revised and accepted on December 1, 2007

Advance Access publication January 3, 2008

Associate Editor: Chris Stoeckert

ABSTRACT

Summary: Many bioinformatics solutions suffer from the lack of usable interface/platform from which results can be analyzed and visualized. Overcoming this hurdle would allow for more widespread dissemination of bioinformatics algorithms within the biological and medical communities. The algorithms should be accessible without extensive technical support or programming knowledge. Here, we propose a dynamic wizard platform that provides users with a Graphical User Interface (GUI) for most Java bioinformatics library toolkits. The application interface is generated in real-time based on the original source code. This platform lets developers focus on designing algorithms and biologists/physicians on testing hypotheses and analyzing results.

Availability: The open source code can be downloaded from: http://bcl.med.harvard.edu/proteomics/proj/APBA/

Contact: gil@mit.edu

1 INTRODUCTION

Previous work has sought to reduce the process of bioinformatics development through algorithm and data structure template libraries (Pitt *et al*., 2001; Stajich and Lapp, 2006). On SourceForge, about 1075 applications have been categorized as 'Bioinformatics' and Bioinformatics.org has close to 275 bioinformatics projects. There are numerous other bioinformatics libraries on such open source websites that are not being used because they are not accessible to many biologists. Out of the 1075 applications on SourceForge, only ~30 have a Graphical User Interface (GUI) associated with the application. This means that about 97% of the bioinformatics algorithms might not be utilized to their full potential due to a lack of a graphical interface platform.

Using bioinformatics toolkits generally requires a series of similar sequential steps including: loading data files, choosing the analytical method, choosing analytical method options, running the analysis, displaying results (with graphs and with statistics) and saving the results. This pattern can be exploited to form a wizard application platform that can guide a user through these steps regardless of the details of the particular toolkit.

Developers of bioinformatics algorithms spend little time on developing interfaces for algorithms since GUI's, in and of themselves, are not strongly rewarded by the methods-based journal peer-review process. However, if the interface could automatically parse their code and make an appropriate GUI, then developers are much more likely to adopt such a platform.

2 APPROACH AND APPLICATIONS

2.1 System architecture

The Automated Programming for Bioinformatics Algorithm (APBA) deployment platform uses the Model-View-Controller design concept to reduce dependencies on the source code. The main entry point into the system is the parser that takes in the source code and generates an internal representation of the system. The wizard maker uses this internal representation to form the appropriate components in the GUI. Changes in the parser representation are reflected in the GUI through an observer. Events in the GUI need to access either the internal representation or the original source code itself. Access to the source code is done via Adapters, (Fig. 1) that use the internal representation to access the methods in the original source code.

Figure 1 shows, how the backend source code is decoupled from the internal system surrounded in the dotted box.

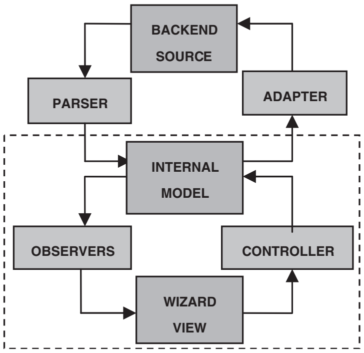
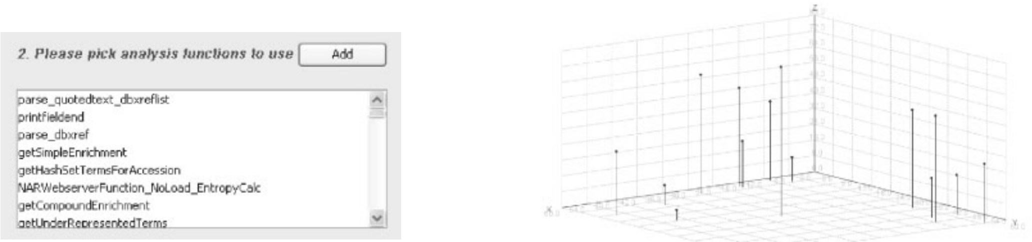


Fig. 1. The object model of the dynamic wizard platform.

*To whom correspondence should be addressed.



2. Please pick analysis functions to use

parse_quotedtext_dbxreflist
printfieldend
parse_dbxref
getSimpleEnrichment
getHashSetTermsForAccession
NARWebserverFunction_NoLoad_EntropyCalc
getCompoundEnrichment
getUnderRepresentedTerms

2.1: The user can choose what functions to run (see above). The user can set variables for analysis and storage in the workspace (not shown).

2.2: An example of a 3D plot generated from the program.

Fig. 2. Screenshots of the wizard.

2.2 Dynamically generating the wizard

The APBA platform allows a graphical interface framework to be used with any Java bioinformatics toolbox. If the source code is not tagged, then all the functions are shown in one screen in a hierarchical tree view. However, functions can be separated into more than one screen by their purpose. This is done by tagging the source code with tags in the JavaDoc comment section of each class or method using an Xdoclet type tag. Some examples of tags that are used include:

(i) Tagging the class with the main() method as @MainClass. Optionally, tags such as @Picture and @Introduction can also be placed this class. The values of these tags are used to get the picture and introduction message for the welcome screen as shown in the results section of the article.

(ii) Tagging functions that are used for loading data, analyzing data and saving data with the @Load, @Analysis and @Save tags, respectively in the comment area of the method. These can also be used to guide the user through a sequence of functions that help answer a specific biological question.

Qdox (http://qdox.codehaus.org) is used at runtime to parse the code for tags and form the internal representation. This is used internally to form the GUI, register GUI event and determine appropriate screens in the wizard to expose the method for the user. The source code is accessed through the Adapter which is made using the Java reflection library.

2.3 Functionality and features

The APBA platform provides basic functionalities commonly required for bioinformatics toolkit-based analyses (Fig. 2). The first basic feature is the ability to load input data files from directories or online. To perform any analysis, the user has to load these files to build the background information. If desired, the user can initially select from several biologically motivated questions in order to constrain the function space. The next step is to perform the desired calculations. The analysis screen shows the available functions based on class structure. The user can then set variables for the parameters of the analysis functions and run the functions using a drop down menu to assign parameter values for each run.

All the variables that are created by the user and the results of analysis functions are stored in a tree structure in the workspace. The variables in the workspace can be manipulated by drag-and-drop. Once the analysis functions are run, the user can visualize the results. An interface allows various types of 2D or 3D graphs with the option of choosing which variables are plotted in each axis. The user can also save variables and plots. Variables from the workspace can also be saved in Excel readable form or in data files for future use in the wizard platform.

The actions that the users take during a session are saved in Java code form. This enables users to see the underlying toolkit program structure, allows them to easily return to where they had stopped in their previous session, and lets them edit and the code that calls the toolkits libraries.

3 DISCUSSION AND CONCLUSION

The APBA platform presented here was tested on a number of bioinformatics toolkits. Two of these are included online (see availability) with and without the platform for comparison (see availability): Open Biomedical Ontology-Based Exploration and Search (OBOES)(Alterovitz *et al*., 2007) and ReadSeq (http://sourceforge.net/projects/readseq/). ReadSeq is a program for conversion of biosequence data. ReadSeq was run without tagging the source code to validate that the platform can be used without any changes to the source code. This showed that program is generic because it worked for a variety of toolkit types: OBOES is an ontology analysis-based bioinformatics application and ReadSeq is sequence analysis-type application.

Programming and debugging for experimental bioinformatics applications can be especially tiring because the people doing the work are not primarily programmers; they are primarily biological researchers. This platform, not only increases efficiency, but also reduces programming errors because the underlying platform is already pre-written and extensively tested for accuracy. Built in open source java, the platform can also be integrated with commonly used packages such as Weka and R. Biology students have also found the platform useful for getting quick results while learning about the underlying programming.

Conflict of Interest: none declared.

REFERENCES

Alterovitz,G. *et al*. (2007) An information theoretic framework for ontology-based bioinformatics. In *Proceedings of the Information Theory Applications Workshop*, San Diego, CA.

Pitt,W.R. *et al*. (2001) The bioinformatics template library–generic components for biocomputing. *Bioinformatics*, **17**, 729–737.

Stajich,J. and Lapp,H. (2006) Open source tool and toolkits for bioinformatics: significance and where are we? *Brief. Bioinform*., **7**, 287–296.