

*Application Note***BioJava: an Open-Source Framework for Bioinformatics**Holland, R.C.G.<sup>1</sup>, Down, T.<sup>2</sup>, Pocock, M.<sup>3</sup>, Prlić, A.<sup>4\*</sup>, Huen D.<sup>5</sup>, James K.<sup>4</sup>, Foisy, S.<sup>6</sup>, Dräger, A.<sup>7</sup>, Yates, A.<sup>1</sup>, Heuer M.<sup>8</sup>, and Schreiber, M.J.<sup>9</sup>

<sup>1</sup>European Bioinformatics Institute (EMBL-EBI), Genome Campus, Hinxton, Cambridgeshire CB10 1SD, UK <sup>2</sup>Gurdon Institute and Department of Genetics, Cambridge CB2 1QN, UK <sup>3</sup>University Newcastle Upon Tyne, Newcastle Upon Tyne, NE1 7RU, UK <sup>4</sup>Wellcome Trust Sanger Institute, Genome Campus, Hinxton, Cambridgeshire CB10 1SA, UK <sup>5</sup>Department of Genetics, University of Cambridge, Cambridge CB2 3EH, UK <sup>6</sup>Laboratory in Genetics and Genomic Medicine of Inflammation, Montreal Heart Institute, Montreal, Canada, H1T 1C8 <sup>7</sup>Eberhard Karls University Tübingen, Center for Bioinformatics (ZBIT), Germany <sup>8</sup>Harbinger Partners, Inc. St. Paul, Minnesota, USA <sup>9</sup>Novartis Institute for Tropical Diseases, 10 Biopolis Road, Chromos #05-01, Singapore 138670

Associate Editor: Prof. Anna Tramontano

**ABSTRACT**

**Summary:** BioJava is a mature open-source project that provides a framework for processing of biological data. BioJava contains powerful analysis and statistical routines, tools for parsing common file formats, and packages for manipulating sequences and 3D structures. It enables rapid bioinformatics application development in the Java programming language.

**Availability:** BioJava is an open-source project distributed under the Lesser GPL (LGPL). BioJava can be downloaded from the BioJava website [<http://www.biojava.org>]. BioJava requires Java 1.5 or higher.

**Contact:** All queries should be directed to the BioJava mailing lists. Details are available at [<http://biojava.org/wiki/BioJava:MailingLists>].

**Supplementary Information:** Full documentation can be found at the BioJava website.

**1 INTRODUCTION**

BioJava was conceived in 1999 by Thomas Down and Matthew Pocock as an API to simplify bioinformatics software development using Java (Pocock et al., 2000, Pocock 2003). It has since then evolved to become a fully-featured framework with modules for performing many common bioinformatics tasks. The goal of BioJava is to facilitate code re-use and to provide standard implementations that are easy to link to external scripts and applications.

BioJava is an open-source project that is developed by volunteers and coordinated by the Open Bioinformatics Foundation (OBF). It is one of several Bio\* toolkits (Mangalam, 2002). All code is distributed under the LGPL license and can be freely used and reused in any form.

BioJava is a mature project and has been employed in a number of real-world applications and over 50 published studies. A list of these can be found on the BioJava website. According to the project tracking web site Ohloh [<http://www.ohloh.net/projects/biojava>], the BioJava code-base represents an estimated 47 person years worth of effort.

**2 FEATURES**

BioJava contains a number of mature APIs. The ten most frequently used are: 1) Nucleotide and amino acid Alphabets, 2) BLAST parser, 3) Sequence I/O, 4) Dynamic Programming, 5) Structure I/O and manipulation, 6) Sequence manipulation, 7) Genetic Algorithms, 8) Statistical Distributions, 9) Graphical User Interfaces, 10) Serialization to databases. Below follows a short discussion of some of these modules.

At the core of BioJava is a symbolic **alphabet** API which represents sequences as a list of references to singleton symbol objects that are derived from an alphabet. Lists of symbols are stored whenever possible in a compressed form of up to 4 symbols per byte of memory.

In addition to the fundamental symbols of a given alphabet (A, C, G, and T in the case of DNA), all BioJava alphabets implicitly contain extra Symbol objects representing all possible combinations of the fundamental symbols.

The symbol approach allows the construction of higher order alphabets and symbols that represent the multiplication of one or more alphabets. An example is the codon 'alphabet' which is the cubed product of the DNA alphabet, each codon 'symbol' comprising three DNA symbols. Such an alphabet allows construction of views over sequences without modifying the underlying sequence which is useful for tasks such as translation. Other complex alphabets which can be described include conditional alphabets for the construction of conditional probability distributions, and heterogeneous alphabets such as the combination of the codon and protein alphabets for use with a DNA-protein aligning Hidden Markov model (HMM). Other interesting applications of the Alphabet API include chromosomes for genetic algorithms using, but not limited to, integer or binary symbol lists, and the representation of Phred quality scores (Ewing et al., 1998) as a multiplication of the DNA and integer alphabets.

The typical user would most likely start out by using the **sequence input/output** API and the **sequence/feature object model**. These allow sequences to be loaded from a number of common file formats such as FASTA, GenBank and EMBL, optionally manipulated in memory, then saved again or converted into a different format. The simplicity of this process is demonstrated in Figure 1.

Another useful API is the **feature/annotation object model** which associates sequences with located features and unlocated

\* To whom correspondence should be addressed.

annotations. Features can be found either by keyword or by defining a location query from which all overlapping or contained features are returned, while annotations can be retrieved by keyword. The location model handles circular and stranded locations, split locations, and multi-sequence locations allowing features to span complex sets of coordinates.

The **protein structure** API contains tools for parsing and manipulating PDB files (Berman, 2000). It contains utility methods to perform linear algebra calculations on atomic coordinates and can calculate 3D structure alignments. A simple interface to the 3D visualization library Jmol [http://www.jmol.org] is contained as well. An add-on allows the serialization of the content of a PDB file to a database using Hibernate [http://www.hibernate.org].

Other APIs include those for working with chromatograms, sequence alignments, proteomics, and ontologies. Parsers are provided for reading, amongst others, Blast reports (Altschul, 1997), ABI chromatograms, NCBI taxonomy definitions. Recently the BioJavaX module was added which provides more detailed parsing of the common file formats and improved storing of sequence data into BioSQL databases [http://www.biosql.org]. This allows to incorporate BioJava into existing data processing pipelines which use alternative OBF toolkits such as BioPerl (Stajich, 2002).

The BioJava web site provides detailed manuals on how to use the different components. In particular the “CookBook” section provides a quick introduction into solving many problems by demonstrating solutions with documented source code. There is also a section to demonstrate the performance of a few selected tasks via Java WebStart examples. To mention just one: The FASTA-formatted release 4 Drosophila genome sequence can be parsed in <20 seconds on a 1.80GHz Core Duo processor.

### 3 FUTURE DEVELOPMENT

BioJava aims to provide an API that is of use to anyone using Java to develop bioinformatics software, regardless of which specialisation they may work in. Genomic features currently must be manipulated with reference to the underlying genomic sequence, which can make working with post-genomic datasets, such as microarray results, overly complex. Phylogenetics tools are already in development which will allow users to work with NEXUS tree files (Maddison, 1997).

Although the Blast parsing API is widely used, it does not support all of the existing blast-family output formats. We will continue the ongoing effort to add parsers for PSI-Blast and other currently unsupported formats.

Users are welcome to identify further areas of need and their suggestions will be incorporated into future developments.

BioJava is written entirely in the Java programming language, and will run on any platform for which a Java 1.5 run-time environment is available. Java 5 and 6 provide advanced language features, and we shall be taking advantage of these in the next major release, both to aid in maintenance of the library and to make it even easier for novice Java developers to make use of the BioJava APIs.

### 4 CONCLUSIONS

BioJava is one of the largest open-source APIs for bioinformatics software development. It is a mature project with a large user and support community. It offers a wide range of tools for common bioinformatics tasks. The BioJava homepage provides access to the source code and detailed documentation.

### ACKNOWLEDGEMENTS

We want to thank everybody who made code or documentation contribution during the project's life. Each of these contribution is appreciated, though the total list of contributors is too long to be reproduced here. BioJava is not formally funded by any grants. Through the OBF we have received sponsorship from Sun Microsystems, Apple Computers and NESCent. The initial development of the phylogenetics module was undertaken as a Google Summer of Code 2007 project in collaboration with NESCent. Open access publishing costs have been payed for by the Wellcome Trust.

### REFERENCES

- Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997), Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389-3402.
- Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Sindyalov, I.N., Bourne, P.E. (2000) The Protein Data Bank, *Nucleic Acids Res.*, **28**, 235-242.
- Pocock, M., Down, T., Hubbard, T (2000) BioJava: Open Source Components for Bioinformatics. *ACM SIGBIO Newsletter* **20(2)**, 10-12
- Pocock, M., Computational Analysis of Genomes. PhD thesis. University of Cambridge (2003)
- Ewing, B., Hillier, L., Wendl, M.C., Green, P. (1998) Base-calling of automated sequencer traces using phred. *Genome Research*, **8**, 175-185.
- Maddison, D.R., Swofford, D.L., Maddison, W.P. (1997) NEXUS: an extensible file format for systematic information. *Syst Biol.*, **46(4)**, 590-621.
- Mangalam, H. (2002) The Bio\* toolkits – a brief overview. *Brief Bioinform.*, **3**, 396-302.
- Saitou, N., Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Evol. Biol.*, **4(4)**, 406-425.
- Stajich, J.E., Block, D., Boulez, K. et al. 2002. The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611-8.

```
BufferedReader input =
    new BufferedReader(
        new FileReader("mygenbank.file"));

RichSequenceIterator seqsIn =
    RichSequence.IOTools.readGenbankDNA(
        input,
        RichObjectFactory.getDefaultNamespace());

RichSequence.IOTools.writeFasta(
    System.out, seqsIn,
    RichObjectFactory.getDefaultNamespace());
```

**Fig. 1.** Loading a GenBank file with BioJava and writing it out as FASTA. The example demonstrates the use of several convenience methods that hide the bulk of the implementation. If the developer desires a more flexible parser it is possible to make use of the interfaces hidden behind the convenience methods to expose a fully customizable, multicomponent, event based parsing model.