BIOINFORMATICS APPLICATIONS NOTE Vol. 28 no. 22 2012, pages 2996–2997
doi:10.1093/bioinformatics/bts538

Structural bioinformatics Advance Access publication August 31, 2012

CSB: a Python framework for structural bioinformatics

Ivan Kalev[1,*], Martin Mechelke[1], Klaus O. Kopec[1], Thomas Holder[1], Simeon Carstens[1] and Michael Habeck[1,2,*]

[1]Department of Protein Evolution, Max Planck Institute for Developmental Biology, Spemannstrasse 35 and [2]Department of Empirical Inference, Max Planck Institute for Intelligent Systems, Spemannstrasse 38, 72076 Tübingen, Germany

Associate Editor: Anna Tramontano

ABSTRACT

Summary: Computational Structural Biology Toolbox (CSB) is a cross-platform Python class library for reading, storing and analyzing biomolecular structures with rich support for statistical analyses. CSB is designed for reusability and extensibility and comes with a clean, well-documented API following good object-oriented engineering practice.

Availability: Stable release packages are available for download from the Python Package Index (PyPI) as well as from the project's website http://csb.codeplex.com.

Contacts: ivan.kalev@gmail.com or michael.habeck@tuebingen.mpg.de

Received on July 10, 2012; revised on August 6, 2012; accepted on August 24, 2012

1 INTRODUCTION

The Python programming language is becoming an increasingly popular choice in research. With its comprehensive numerical libraries and dynamic type system, Python facilitates rapid application development. But although rapid prototyping is very practical for experimenting with new techniques or features, systematic use of *ad hoc* scripting often turns into a burden preventing efficient code reuse. This problem is solved with the use of continuously developed, well-abstracted and tested software libraries. Productivity in building solid, reliable and extensible bioinformatics applications could therefore significantly benefit from the practice of using library code.

Here, we introduce the public release of CSB, a Python library designed for solving problems in the field of computational structural biology. CSB improves over existing libraries such as Biopython (Cock *et al*., 2009) with its granular, consistent and extensible object model and also provides new features like a comprehensive statistical API and support for new abstractions and file formats. This project is a quickly growing class library for structural bioinformatics, providing clean object-oriented APIs for working with biological macromolecular structures, sequences, sequence profiles and fragment libraries, and also a significant amount of statistical modules, including many probability distributions and samplers. We put a strong emphasis on quality and reliability achieved through continuous attention to good software design and best practices in test engineering.

*To whom correspondence should be addressed.

2 CORE LIBRARY

CSB consists of several highly branched, hierarchical Python packages. The core library is roughly divided into bioinformatics (*csb.bio*.*) and statistics (*csb.statistics*.*).

The *csb.bio* namespace hosts a number of packages that define all fundamental biological abstractions of the library. For example, *csb.bio.sequence* defines the *AbstractSequence* and *AbstractAlignment* objects and also provides some standard implementations of these abstractions, such as *Sequence*, *SequenceAlignment* and *StructureAlignment*. As suggested by its name, *csb.bio.hmm* deals with HHpred and its profile HMMs (Söding, 2005), while *csb.bio.fragments* contains objects describing protein structure fragment libraries. The package *csb.bio.structure* implements essential CSB objects like *Structure*, *Chain*, *Residue* and *Atom*.

The *Structure* class illustrates examples of CSB's design philosophy. *Structure* instances are hierarchical objects, implementing the composite pattern. Each level in this hierarchy is represented by a class, derived from the base *AbstractEntity*. Every entity thus exposes a standard set of operations such as *AbstractEntity.transform()*, which automatically propagate down the tree when invoked at arbitrary level. Users are free to define their own, pluggable *AbstractEntity* implementations.

Another aspect of the core library is our I/O API for a broad variety of biological file formats (*csb.bio.io*). For example, *csb.bio.io.hhpred* is the first publicly available Python module to date for working with HHpred's HMM and result files (Söding, 2005). Another module, *csb.bio.io.mrc*, implements objects for processing cryo-electron microscopy maps, while *csb.bio.io.clans* provides I/O for CLANS (Frickey and Lupas, 2004). Reading and writing PDB files is done using our PDB API, which is part of *csb.bio.io.wwpdb*. It is worth mentioning that the default PDB parser in CSB differs significantly from existing solutions such as Biopython. *StructureParser* reads and initializes all residues from SEQRES (if available), rather than the ATOM fields in the file. ATOM records are subsequently mapped to the residue objects using a simple and fast alignment algorithm. Therefore, *Chain* objects in CSB always contain the complete primary structure of the PDB chain, as defined by the SEQRES fields. This feature eliminates the need to relate the PDB atoms back to the real sequence of the protein in question—a process which is often difficult and error-prone. When benchmarked over the complete PDB database, our SEQRES mapping algorithm fails for about 250 structures. This is frequently an indication of a PDB format issue. In this case, the parser would raise a characteristic exception upon which the client can switch to an ATOM-based parsing mode.

We compared the performance of *RegularStructureParser* with PDB I/O modules from alternative libraries: Biopython, PyCogent (Cielik *et al.*, 2011) and the C++ based Open Structure (Biasini *et al.*, 2010). As expected, OpenStructure was the fastest and parsed 4000 PDB entries with 0.09 s per structure. CSB is positioned between Biopython (0.19 s) and PyCogent (0.43 s) with 0.32 s per structure, which suggests that the SEQRES mapping feature comes with an acceptable performance overhead.

Our library also hosts a collection of statistical models in the *csb.statistics* namespace. Among these models are standard uni- and multivariate probability distributions such as the Normal and the Gamma distribution and also more exotic distributions such as the multivariate normal inverse Gaussian distribution used to model multivariate heavy-tailed data. Several estimators based on maximum likelihood and Gibbs sampling are implemented. Moreover, we provide a general framework for Markov chain Monte Carlo simulation and implementation of standard schemes such as random walk Metropolis Hastings, Hamiltonian Monte Carlo (Duane *et al.*, 1987) and replica-exchange Monte Carlo (Swendsen and Wang, 1986). Methods to analyze Monte Carlo output are also provided such as, for example, a non-parametric histogram reweighting scheme for the estimation of free energy differences (Habeck, 2012).

3 CSB APPLICATIONS

CSB comes with a simple framework for writing console applications (*csb.apps*). These applications could be seen as short protocols built on top of the core library and consuming its APIs. Each release is bundled with a number of pre-installed, open-source applications. For example, *csb.apps.hhfrag* provides HHfrag, a CSB application for building dynamic fragment libraries (Kalev and Habeck, 2011). BFit is another app, which performs robust superposition of protein structures (Mechelke and Habeck, 2010). Every release package also contains EMBD, an application for sharpening of cryo-electron microscopy maps (Hirsch *et al.*, 2011) using non-negative deconvolution and Promix, an application implementing Gaussian mixture models for identifying rigid domains in structure ensembles (Hirsch and Habeck, 2008).

4 DEVELOPMENT

One of the key design goals of CSB is providing clean, extensible, object-oriented APIs with accompanying API documentation. This project puts a strong emphasis on quality, achieved through systematic use of abstraction, strong encapsulation, separation of responsibilities and refactoring with classic design patterns.

Our development team has adopted a continuous integration model. The reliability of the production code is controlled by CSB's built-in high-coverage unit test framework. Stable builds will be gradually released to the public domain, and nightly builds can be obtained upon request. Portability is also a design goal, so CSB works without modification on every major platform (Windows, Linux and Mac) and any modern Python interpreter (version 2.6 or higher, including Python 3).

Funding: Contract research 'Methoden in den Lebenswissenschaften' of the Baden-Württemberg Stiftung, by Deutsche Forschungsgemeinschaft (DFG) grant HA 5918/1-1 and by the Max Planck Society.

Conflict of Interest: none declared.

REFERENCES

Biasini,M. *et al.* (2010) OpenStructure: a flexible software framework for computational structural biology. *Bioinformatics*, **26**, 2626–2628.
Cielik,M. *et al.* (2011) Abstractions, algorithms and data structures for structural bioinformatics in PyCogent. *J. Appl. Crystallogr.*, **44** (Pt 2), 424–428.
Cock,P.J. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.
Duane,S. *et al.* (1987) Hybrid Monte Carlo. *Phys. Lett. B*, **195**, 216–222.
Frickey,T. and Lupas,A. (2004) CLANS: a Java application for visualizing protein families based on pairwise similarity. *Bioinformatics*, **20**, 3702–3704.
Habeck,M. (2012) Evaluation of marginal likelihoods using the density of states. In Lawrence,N. and Girolami,M. (eds.) *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS) 2012*. Vol. 22, La Palma, Canary Islands. JMLR:W&CP 22.
Hirsch,M. and Habeck,M. (2008) Mixture models for protein structure ensembles. *Bioinformatics*, **24**, 2184–2192.
Hirsch,M. *et al.* (2011) A blind deconvolution approach for improving the resolution of cryo-EM density maps. *J. Comput. Biol.*, **18**, 335–346.
Kalev,I. and Habeck,M. (2011) HHfrag: HMM-based fragment detection using HHpred. *Bioinformatics*, **27**, 3110–3116.
Mechelke,M. and Habeck,M. (2010) Robust probabilistic superposition and comparison of protein structures. *BMC Bioinformatics*, **11**, 363.
Söding,J. (2005) Protein homology detection by HMM–HMM comparison. *Bioinformatics*, **21**, 951–960.
Swendsen,R.H. and Wang,J.-S. (1986) Replica Monte Carlo simulation of spin glasses. *Phys. Rev. Lett.*, **57**, 2607–2609.