BioStructures.jl: read, write and manipulate macromolecular structures in Julia

Joe G Greener[1], Joel Selvaraj[2], Ben J Ward[3]

[1]Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK
[2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India
[3]The Earlham Institute, Norwich Research Park, Norwich, UK

Abstract

Summary: Robust, flexible and fast software to read, write and manipulate macromolecular structures is a prerequisite for productively doing structural bioinformatics. We present BioStructures.jl, the first dedicated package in the Julia programming language for dealing with macromolecular structures and the Protein Data Bank. BioStructures.jl builds on the lessons learned with similar packages to provide a large feature set, a flexible object representation and high performance.
Availability and implementation: BioStructures.jl is freely available under the MIT license. Source code and documentation are available at https://github.com/BioJulia/BioStructures.jl. BioStructures.jl is compatible with Julia versions 0.6 and later and is system-independent.
Contact: j.greener@ucl.ac.uk

Introduction

Open source software packages to parse files from the Protein Data Bank (PDB) (Berman et al. 2000) and manipulate macromolecular structures exist in many languages (Hamelryck and Manderick 2003; Grant et al. 2006; Stajich et al. 2002; Goto et al. 2010; Loriot, Cazals, and Bernauer 2010; Lafita et al. 2019). Such packages must strike a balance between a powerful and useful representation of molecules, fast performance, easy integration with other tools and tolerance to the ambiguities in PDB data.

Julia is a high-performance, dynamically-typed, open source programming language (Bezanson et al. 2017). Since its first release in 2012 it has grown rapidly in popularity, particularly in the scientific computing community, with version 1.0 being released in 2018. To date it has over 13 million downloads and over 3,000 packages registered for community use. In particular the ability to write performant code in a high-level language means that Julia can solve the "two-language problem" of having to prototype code in one language and then write a performant version in another language. BioStructures.jl is a Julia package to read, write and manipulate macromolecular structures. Whilst other Julia packages have provided functionality related to structural bioinformatics (Zea et al. 2017; Greener, Filippis, and Sternberg 2017), BioStructures.jl is the first dedicated package and contains all the main features that structural bioinformaticians need to be productive in Julia. It is designed to be used for standard structural analysis tasks, interactive data analysis, and to act as a platform on which others can build to create more specific tools. BioStructures.jl is part of BioJulia, an organisation that provides bioinformatics infrastructure for the Julia language.

Features

BioStructures.jl has the following features:

- Read in PDB, mmCIF and MMTF (Bradley et al. 2017) files into a hierarchical representation of structure. The parsers have been tested on the whole PDB, only throwing errors on a small number of known ambiguous cases.
- Write out PDB, mmCIF and MMTF files. The ability to read and write freely between these file formats is not available in many similar packages.
- Read mmCIF and MMTF files into a dictionary, e.g. allowing access to header information. MMTF files are decoded with the related package MMTF.jl.
- Iterate over structures at various levels, e.g. iterate over atoms in a residue or residues in a chain.
- Select various structural elements using pre-defined or custom selectors, e.g. collect all Cβ atoms (Cɑ in the case of glycine) from standard residues.
- Retrieve amino acid sequences and integrate with the broader BioJulia ecosystem, for example allowing fast sequence alignments.
- Spatial calculations including distances, bond angles, dihedral angles, contact maps and distance maps. Contact and distance maps can be plotted.
- Superimposition of structures and calculation of the RMSD.
- Download files and data from the RCSB PDB including functions to maintain a local copy of the PDB.
- Interoperability with the broader Julia ecosystem, e.g. exporting to a data frame or creating a graph of contacting residues.
- Visualisation of molecular structures in a pop-up window or Jupyter notebook using the related package Bio3DView.jl (https://github.com/jgreener64/Bio3DView.jl), which is a wrapper around 3Dmol.js (Rego and Koes 2015).
- Easy installation with Julia’s package manager.
- Comprehensive test suite, continuous integration build testing and a benchmark suite to test for performance regressions.

- Thorough online documentation and in-code docstrings.

- Fully open source with a permissive MIT license.

- Faster than similar packages at most tasks. Our benchmarks, summarised in Figure 1 and described further at https://github.com/jgreener64/pdb-benchmarks, indicate that the package has competitive or superior performance to 14 other commonly used packages from both interpreted and compiled languages. For example, parsing the small PDB entry 1CRN takes 0.76 ms/1.9 ms/1.1 ms in the PDB/mmCIF/MMTF formats after just-in-time (JIT) compilation on a standard desktop computer. It does this whilst using a hierarchical structure representation, allowing variation between models in a structure, and accounting for alternative locations at the atom and residue levels (see below). These features take time to execute but increase the utility and flexibility of the package.

Design considerations

BioStructures.jl is heavily influenced by the Bio.PDB module of Biopython (Hamelryck and Manderick 2003), the design of which has proved effective. The structure object has a hierarchical type system of the form ProteinStructure - Model - Chain - AbstractResidue - AbstractAtom. Atoms with alternative locations are stored in a DisorderedAtom container and residues with alternative locations (i.e. point mutations with different residue names) are stored in a DisorderedResidue container. Function calls fall back to the default atom or residue, so alternative locations can be ignored if the user is not interested in them, but building alternative locations into the type system allows correct representation of many more aspects of the PDB. Whilst BioStructures.jl retains the flexibility of Bio.PDB, its implementation in Julia allows it to have superior speed.

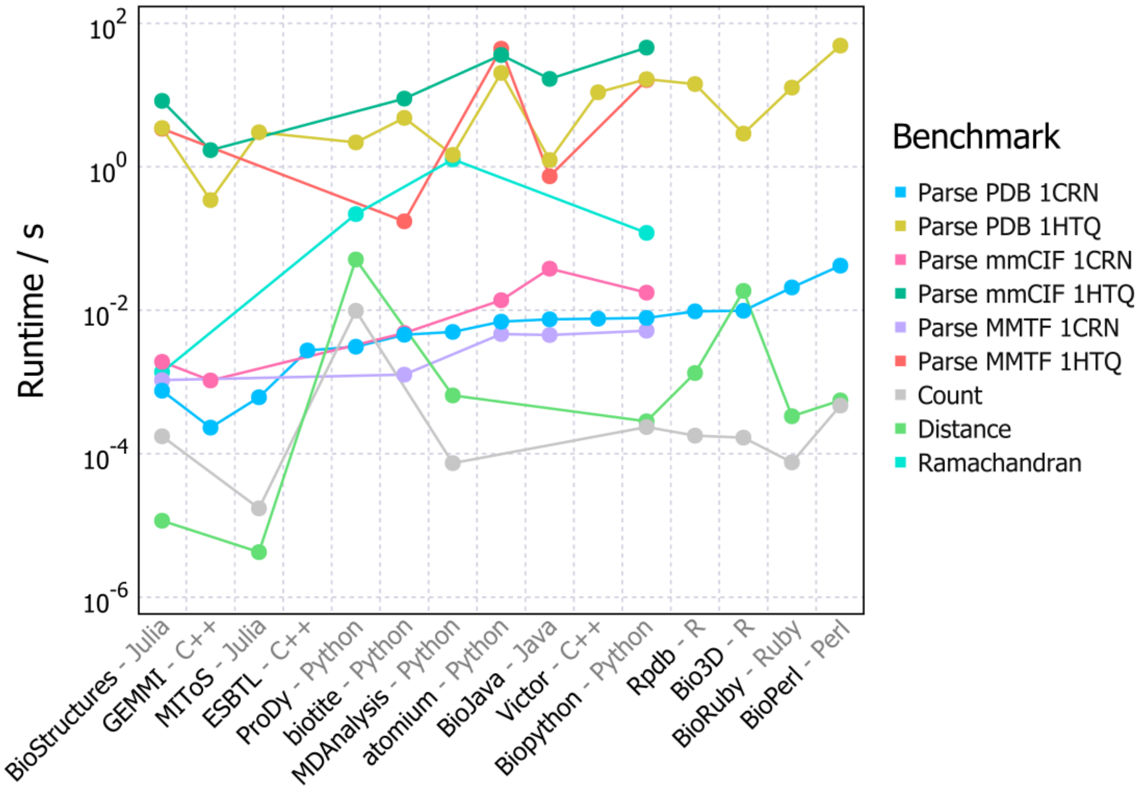

Figure 1 Performance of structural bioinformatics tasks in 15 packages (Zea et al. 2017; Loriot, Cazals, and Bernauer 2010; Bakan, Meireles, and Bahar 2011; Kunzmann and Hamacher 2018; Gowers et al. 2016; Ireland and Martin 2020; Lafita et al. 2019; Hirsh et al. 2015; Hamelryck and Manderick 2003; Grant et al. 2006; Goto et al. 2010; Stajich et al. 2002) covering 7 programming languages. Comparison should be treated with caution since each package does something slightly different and may use a different object representation or do less error checking. MIToS, for example, does not read files into a hierarchical representation of structure. The tasks are reading a small (1CRN) and a large (1HTQ) PDB entry (Gajda 2013) in the PDB, mmCIF and MMTF formats; counting the number of alanine residues in adenylate kinase (1AKE); calculating the distance between residues 50 and 60 of chain A in adenylate kinase; and calculating the Ramachandran ϕ/ψ angles in adenylate kinase. In each case the mean time of the fastest implementation for each software that makes use of the provided API is given. Tasks are not implemented in packages where there is no obvious API for implementation. Times for Julia packages are measured after JIT compilation. Packages are ordered by increasing time to parse PDB 1CRN, with BioStructures first. See https://github.com/jgreener64/pdb-benchmarks for more details. The version of the benchmarks presented here is archived at Zenodo with DOI 10.5281/zenodo.3753016.

Acknowledgements

We would like to thank the BioJulia, Julia and Biopython communities for discussion, assistance and support.

Conflict of Interest: none declared.

References

Bakan, A., L. M. Meireles, and I. Bahar. 2011. “ProDy: Protein Dynamics Inferred from Theory and Experiments.” *Bioinformatics* 27 (11): 1575–77.

Berman, H. M., J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. 2000. “The Protein Data Bank.” *Nucleic Acids Research* 28 (1): 235–42.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah. 2017. “Julia: A Fresh Approach to Numerical Computing.” *SIAM Review* 59 (1): 65–98.

Bradley, A. R., A. S. Rose, A. Pavelka, Y. Valasatava, J. M. Duarte, A. Prlić, and P. W. Rose. 2017. “MMTF - An Efficient File Format for the Transmission, Visualization, and Analysis of Macromolecular Structures.” *PLoS Comput. Biol.* 13 (6): e1005575.

Gajda, M. J. 2013. “hPDB – Haskell Library for Processing Atomic Biomolecular Structures in Protein Data Bank Format.” *BMC Research Notes* 6 (1).

Goto, N., P. Prins, M. Nakao, R. Bonnal, J. Aerts, and T. Katayama. 2010. “BioRuby: Bioinformatics Software for the Ruby Programming Language.” *Bioinformatics* 26 (20): 2617–19.

Gowers, R., M. Linke, J. Barnoud, T. Reddy, M. Melo, S. Seyler, J. Domański, et al. 2016. “MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations.” *Proceedings of the 15th Python in Science Conference*, 98–105.

Grant, B. J., A. P. C. Rodrigues, K. M. ElSawy, J. A. McCammon, and L. S. D. Caves. 2006. “Bio3d: An R Package for the Comparative Analysis of Protein Structures.” *Bioinformatics* 22 (21): 2695–96.

Greener, J. G., I. Filippis, and M. J. E. Sternberg. 2017. “Predicting Protein Dynamics and Allostery Using Multi-Protein Atomic Distance Constraints.” *Structure* 25 (3): 546–58.

Hamelryck, T., and B. Manderick. 2003. “PDB File Parser and Structure Class Implemented in Python.” *Bioinformatics* 19 (17): 2308–10.

Hirsh, L., D. Piovesan, M. Giollo, C. Ferrari, and S. C. E. Tosatto. 2015. “The Victor C Library for Protein Representation and Advanced Manipulation.” *Bioinformatics* 31 (7): 1138–40.

Ireland, S. M., and A. C. R. Martin. 2020. “Atomium - A Python Structure Parser.” *Bioinformatics* .

Kunzmann, P., and K. Hamacher. 2018. “Biotite: A Unifying Open Source Computational Biology Framework in Python.” *BMC Bioinformatics* 19 (1): 346.

Lafita, A., S. Bliven, A. Prlić, D. Guzenko, P. W. Rose, A. Bradley, P. Pavan, et al. 2019. “BioJava 5: A Community Driven Open-Source Bioinformatics Library.” *PLoS Comput. Biol.* 15 (2): e1006791.

Loriot, S., F. Cazals, and J. Bernauer. 2010. “ESBTL: Efficient PDB Parser and Data Structure for the Structural and Geometric Analysis of Biological Macromolecules.” *Bioinformatics* 26 (8): 1127–28.

Rego, N., and D. Koes. 2015. “3Dmol.js: Molecular Visualization with WebGL.” *Bioinformatics* 31 (8): 1322–24.

Stajich, J. E., D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, et al. 2002. “The Bioperl Toolkit: Perl Modules for the Life Sciences.” *Genome Res.* 12 (10): 1611–18.

Zea, D. J., D. Anfossi, M. Nielsen, and C. Marino-Buslje. 2017. “MIToS.jl: Mutual Information Tools for Protein Sequence Analysis in the Julia Language.” *Bioinformatics* 33 (4): 564–65.