bio-go: A flexible tools Bioinformatics library

Ryan A. Hagenson[1]

1 Omaha's Henry Doorly Zoo and Aquarium

DOI: 10.21105/joss.01741

Software

- Review
- Repository
- Archive

Submitted: 12 August 2019
Published: 17 September 2019

Summary

`bio-go` starts from the known intersection of Bioinformatics data types as abstract data types (interfaces), providing functions abstracted over these abstract data types, as well as concrete data types implementing these interfaces to bootstrap development. Semantics of abstract data types are validated via extensive property testing. Flexibility is achieved through a separation of "why-how-what" at the import level such that *why* one is using the library does not force one into *how* it is done or *what* the exact details of the implementation become.

Bioinformatics libraries that informed the development of this one are: BioPython (Cock et al., 2009), Rust-Bio (Köster, 2015), and Bioconductor (Gentleman et al., 2004). Property testing is done via https://github.com/leanovate/gopter/.

Statement of Need

Bioinformatics projects often require building custom, small tools for the purpose of complementing larger tools' inflexibility. Developers are forced to balance size with flexibility – the larger the program and the more people using it, the less flexible its developers can be to changing it for the better. `bio-go` is intended to be used in producing flexible tools that can change according to the specific needs of ongoing Bioinformatics projects.

`bio-go` was designed to be approachable by the end-user programmers who make up a large portion of practicing Bioinformaticians – those with immediate research problems to solve, but without the time or resources to build and thoroughly test a multitude of solutions to the same intermediate problems. The programmer should explore solutions laterally across the import tree (e.g., `bio/sequence/immutable` to `bio/sequence/mutable`) until the approach is tuned to the specific needs of the current research problem.

Acknowledgements

I acknowledge the valuable input on Bioinformatics data types and their semantics afforded to me by Sean C. West and Ryan P. Ehrlich, as well as the friendly ears of Cynthia L. Frasier and Timothy M. Sefczek who have helped me determine how to develop with biologists as end users.

References

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. doi:10.1093/bioinformatics/btp163

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., et al. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*.

Köster, J. (2015). Rust-Bio: a fast and safe bioinformatics library. *Bioinformatics*, *32*(3), 444–446. doi:10.1093/bioinformatics/btv573