HyPhy 2.5 – a customizable platform for comparative sequence analysis using sophisticated evolutionary models.

Sergei L. Kosakovsky Pond, Ryan Velazquez[1], Art F.Y. Poon,[2], Stephen Shank,[1], Steven Weaver,[1], N Lance Hepler, [3,4], Ben Murrell [3,5], Sadie Wisotsky,[1,6], Stephanie J Spielman, [1,7] Simon D.W. Frost, [8], Spencer V Muse [6]

[1]Institute for Genomics and Evolutionary Medicine, Temple University, Philadelphia, PA, USA,

[2]Pathology and Laboratory Medicine - Western University, London, Ontario, Canada,

[3]University of California, San Diego, CA, USA

[4]10x Genomics...

[5] Karolinska Institute, Stockholm, Sweden

[6] North Carolina State University, Raleigh, NC, USA

[7] Rowan University, Glassboro, NJ, USA

[8] Department of Veterinary Medicine, Cambridge University, Cambridge, United Kingdom

***Corresponding author:** E-mail: spond@temple.edu

Associate Editor: TBD

Abstract

`HyPhy` (HYpothesis testing using PHYlogenies) is a scriptable open-source package for fitting a broad range of evolutionary models to multiple sequence alignments and hypothesis testing primarily in the maximum likelihood statistical framework. It has become a popular choice for characterizing various aspects of the evolutionary process: natural selection, evolutionary rates, recombination, and epistasis. The 2.5 release includes a completely re-engineered computational core and analysis library that introduce new classes of evolutionary models and statistical tests, delivers dramatic performance and stability enhancements, improves usability, streamlines end-to-end analysis workflows, and makes it easier to develop custom analyses.

Key words: evolutionary analysis, natural selection, hypothesis testing, statistical inference, software engineering

Letter

Introduction

`HyPhy` is an open source software package for comparative sequence analysis using stochastic evolutionary models. It is written in C++ and implements a domain specific scripting language (HBL, or `HyPhy` Batch Language) similar in syntax and conventions to JavaScript. `HyPhy` follows the common design paradigm (e.g. R(Team *et al.*, 2013), RevBayes(Höhna *et al.*, 2016), BEAST 2.x(Bouckaert *et al.*, 2014)) of coupling a compiled computational engine with compute-intensive core and relevant data types, with a rich and extensible library of scripts or modules which are used to implement models, analyses, and high-level algorithms. `HyPhy` is distributed with

a sizable library of HBL modules and pre-written analyses. Since its initial release in 2005 [@doi:10.1093/bioinformatics/bti079] HyPhy has become an integral tool for the bioinformatics community with over 2,500 peer-reviewed citations and approximately 1,000 jobs processed each week on the companion Datamonkey web application [@doi:10.1093/bioinformatics/bti320; @doi:10.1093/bioinformatics/btq429; @doi:10.1093/molbev/msx335]. Extensions and improvements to `HyPhy` have been ongoing since its inception, with active feedback between users and developers producing new features tailored to the specific needs of the research community. Here we announce the release of the newest version of `HyPhy` (version 2.5) and document how the software has been packaged for easy use in a variety of settings, optimized for larger datasets, redesigned to follow modern best practices, extended to include a broader set of evolutionary models and pre-packaged analyses.

The source code, installation instructions, and documentation for `HyPhy` is available at `github.com/veg/hyphy` and `hyphy.org`, and the accompanying JavaScript result visualization module – at `github.com/veg/hyphy-vision` and `vision.hyphy.org`. In addition, a public instance of the `HyPhy`-powered web-application implementing the most popular analyses is available at `www.datamonkey.org` (Weaver *et al.*, 2018).

New Approaches

Part of the software ecosystem

Software development practices and usage patterns have decisively shifted away from monolithic packages towards modular and reusable components that must be installed, versioned, and maintained in complex software ecosystems (**?**). Recognizing that users of `HyPhy` vary greatly in their technical proficiency, from biologists unfamiliar with the command line to bioinformaticians who want to incorporate `HyPhy` into their own software, there are multiple ways to interact with the package Ryan to sketch out a figure

1. CLI ; keywords and interactive [simple example]
2. Electron app
3. Web applications; Datamonkey, Galaxy, MEGA
4. Library [currently broken]

Optimized for Larger Datasets

`HyPhy` has been optimized extensively tuned to enable analysis of modern large-scale molecular data including methods. Core numerical and optimization routines take advantage of vector processing (SIMD), multiprocessing (OpenMP), and distributed systems (OpenMPI) for fine and coarse-grain parallelization as appropriate to a specific analysis. We found that the majority of popular analyses implemented in `HyPhy` do not significantly benefit from GPU acceleration

(Stamatakis, 2019), hence we currently do not offer this capability. Even when fully accelerated, phylogenetic likelihood calculation for complex models on large datasets remains a bottleneck which cannot be overcome with more code tuning. However, we were able to integrate recent algorithmic advances from the fields of machine learning and natural language processing (Blei *et al.*, 2003) to limit the number of expensive likelihood calculations for the key application of high-throughput screening of alignments consisting of thousands of sequences for evidence of natural selection (Murrell *et al.*, 2013). Finally, we provide high performance computing infrastructure[@doi:10.1093/bioinformatics/bti320; @doi:10.1093/bioinformatics/btq429; @doi:10.1093/molbev/msx335] free of charge to the global community, helping to democratize access to scientific computing resources [@doi:10.1016/j.ijinfomgt.2013.05.010].

Redesigned to Follow Modern Best Practices in Scientific Software

The design of the original `HyPhy` implementation emphasized convenient writing and execution of single HBL scripts. This decision manifested in the easy to use command line prompt which guided users interactively through selecting the desired analysis and choosing the specific analysis parameters. Also, the HBL itself was designed to allow sophisticated models to be specified and fit with concise scripts, facilitated by extensive use of a global namespace. As the common use of `HyPhy` has shifted over the last decade from one-off analyses toward larger studies and use within pipelines and web-applications, `HyPhy` has been redesigned to address the requirements of these changing use cases.

Usage as a typical command line tool (i.e. an executable name followed by key word arguments) has been added alongside the interactive command line prompt. This change, along with the ability to use relative paths to files, has made using HyPhy in pipelines and batch analyses seamless. In addition to being easier to use, the package is also now easier to install. HyPhy is now installable with `bioconda` [@doi:10.1038/s41592-018-0046-7] and other package managers, like `homebrew`, which has become the defacto package manager for scientific software. Users no longer need to concern themselves with dependencies, environments and build processes but can simply type `conda install hyphy`.

The syntax, semantics, and structure of the standard HBL libraries have also been re-designed and improved, providing a more reliable package that is easier to extend and behaves more similarly to familiar language environments like JavaScript. Examples include: controlled vocabulary terms for molecular evolutionary modeling, namespaces for larger library management, basic functional-style programming (e.g. `map, filter` etc), JavaDoc inline documentation, etc ....

We are also standardizing the way `HyPhy` analyses operate. A command-line analysis is expected to collect required inputs and options from the user, either via keyword command line arguments, or prompts, execute the analysis, report progress and key results to the screen (or file) in MarkDown format, which can also be rendered as formatted report documents, and a complete set of results to a JSON file. For the most popular analyses, we have developed a JavaScript interactive visualization library (`vision.hyphy.org`) that reads the JSON files and renders analysis-appropriate views (tables, charts, alignments, trees, etc) of the results. JSON files can be readily parsed and processed in `Python`, `R`, or any number of tools for subsequent analysis.

Software engineering

Software quality control and rigorous testing are expected to be a part of any modern development cycle; a recent study shows this to be an aspirational goal in the field of molecular evolution (**?**). For `HyPhy`2.5 we have implemented extensive automated testing as a part of a continuous integration (CI) pipeline which which both expedites development and helps ensure healthy software is delivered. Our suite includes unit tests for over 90% of HBL functions, method tests on all the core analyses, and likelihood testing [@url:https://gitlab.com/testiphy/testiphy] which compares the likelihood values calculated by `HyPhy` with values calculated by other popular maximum-likelihood software packages and informs the developers if discrepancies are identified. For the core, we use C++ development and testing toolkits to perform additional quality controls such as static code analysis, testing for memory leaks and other issues, and instrumented code profiling.

Analyses and models supported

Although the `HyPhy` package provides for near limitless customization via writing HBL scripts, users can run many common analyses without needing to concern themselves with the HBL at all. The `HyPhy` package comes with pre-written HBL scripts for easily performing some of the most commonly used analyses. These analyses can be executed in the various places HyPhy is available and include:

- aBSREL[@doi:10.1093/molbev/msv022; @doi:10.1093/molbev/msr125] (adaptive Branch-Site Random Effects Likelihood) - Detect positive/diversifying selection at individual branches
- BGM [@doi:10.1093/bioinformatics/btn313] (Bayesian Graphical Models) - Test for co-evolving sites
- BUSTED [@doi:10.1093/molbev/msv035] (Branch-Site Unrestricted Statistical Test for Episodic Diversification) - Test gene-wide selection at pre-defined lineages

- FADE (FUBAR Approach to Directional Evolution) - Evaluate if sites are subject to directional selection (not yet published)
- FEL [@doi:10.1093/molbev/msi105] (Fixed Effects Likelihood) - Inferring non-synoymous and synonymous substitution rates on a per-site basis for smaller datasets using maximum likelihood
- FUBAR [@doi:10.1093/molbev/mst030](Fast, Unconstrained Bayesian AppRoximation) - Infer non-synonymous and synonymous substitution rates on a per-site basis for larger datasets using Bayesian methods
- GARD [@doi:10.1093/molbev/msl051] (Genetic Algorithm for Recombination Detection) - Screen multiple sequence alignment for recombination
- MEME [@doi:10.1371/journal.pgen.1002764] (Mixed Effects Model of Evolution) - Test the hypothesis that individual sites have been subject to episodic positive/diversifying selection
- RELAX [@doi:10.1093/molbev/msu400] - Evaluate whether the strength of natural selection has been relaxed or intensified along a specific set of branches
- SLAC [@doi:10.1093/molbev/msi105] (SingleLikelihood Ancestor Counting) - Infer non-synonymous and synonymous substitution rates on a per-site basis

Add a description of different types of models and inference techniques available. Sergei will write

Discussion

Make `HyPhy` great again... COVFEFE!

Acknowledgments

This work was supported in part by grants R01 AI134384 (NIH/NIAID), R01 GM093939 (NIH/NIGMS) and U01 GM110749 (NIH/NIGMS). JOW was funded by an NIH-NIAID Career Development Award (K01AI110181) and the California HIV/AIDS Research Program (ID15-SD-052).

References

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan): 993–1022.

Bouckaert, R., Heled, J., Kühnert, D., Vaughan, T., Wu, C.-H., Xie, D., Suchard, M. A., Rambaut, A., and Drummond, A. J. 2014. Beast 2: a software platform for bayesian evolutionary analysis. *PLoS computational biology*, 10(4): e1003537.

Höhna, S., Landis, M. J., Heath, T. A., Boussau, B., Lartillot, N., Moore, B. R., Huelsenbeck, J. P., and Ronquist, F. 2016. Revbayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic Biology*, 65(4): 726–736.

Murrell, B., Moola, S., Mabona, A., Weighill, T., Sheward, D., Kosakovsky Pond, S. L., and Scheffler, K. 2013. Fubar: a fast, unconstrained bayesian approximation for inferring selection. *Molecular biology and evolution*, 30(5): 1196–1205.

Stamatakis, A. 2019. A review of approaches for optimizing phylogenetic likelihood calculations. In *Bioinformatics*

and Phylogenetics, pages 1–19. Springer.

Team, R. C. *et al.* 2013. R: A language and environment for statistical computing.

Weaver, S., Shank, S. D., Spielman, S. J., Li, M., Muse, S. V., and Kosakovsky Pond, S. L. 2018. Datamonkey 2.0: a modern web application for characterizing selective and other evolutionary processes. *Molecular biology and evolution*, 35(3): 773–777.