

Plans for the Phylogenetic Comparative Methods Benchmark Database (PhyCoMB)

August 30, 2017

CONTENTS

1	Introduction	2
2	Proposal Content	2
2.1	Community resource for benchmark tests of model performance	2
2.2	References	4
2.3	Later thoughts	4
3	Users	4
3.1	Viewer	4
3.2	Contributor	4
3.3	Administrator	5
4	Workflows	5
4.1	Viewer workflows	5
4.1.1	Explore performance of various methods for a particular task	5
4.1.2	Explore performance of a particular method for various tasks	5
4.2	Contributor workflows	5
4.2.1	Contribute a new method	5
4.2.2	Contribute a new element	5
4.2.3	Contribute a new performance result	6
4.3	Administrator workflows	6
4.3.1	Alter the groupings of elements	6
4.3.2	Approve an addition from a Contributor	6
4.3.3	Deprecate a method or element	6
4.3.4	Manage users	6
5	Views [TODO]	6
6	Database Tables [TODO]	6
7	Downloads	6
7.1	Obtaining testing files	6
7.2	Obtaining methods scripts	7
7.3	Obtaining performance reports	7

1 INTRODUCTION

This document is intended to be a high-level description of the functionality and structure of what PhyCoMB will eventually look like. Ultra-briefly: The goal is to allow users to compare the performance of different phylogenetic methods. There will be a web interface that allows for browsing those performance results, and for contributing new results. Behind that will be a database of the testing datasets, methods, and results.

2 PROPOSAL CONTENT

Here is what was originally proposed to NSF/BSF. It is still pretty much what I have in mind, but the subsequent sections provide much more detail.

PhyCoMB will challenge new methods in a standardized manner, revealing their strengths and weaknesses early in their lifecycle. It will focus on testing if discrete traits affect rates of speciation and extinction, but the work will also support questions of trait evolution and lineage diversification separately.

2.1 Community resource for benchmark tests of model performance

A paper introducing a new phylogenetic comparative method typically includes essential simulations that examine its power and bias and reveal the parameter space in which it will be most beneficial. These simulations typically follow the assumptions of the underlying model. Poor behavior, however, may arise when those assumptions are not met. Because all empirical datasets have been shaped by processes outside the assumptions of any model, it seems self-evident that methods should routinely be tested in many situations beyond their specific focus. Why is this not standard practice for comparative methods developers? It is time-consuming or impossible for a single developer to craft diverse testing datasets that encompass the biological phenomena likely to ‘break’ his/her new method. Furthermore, there is not a culture—during development, peer review, and empirical application—of valuing and hence requiring robustness testing in this field. Thus, phylogenetic comparative methods papers typically discuss possible artifacts but fall short of providing concrete guidance about whether a method can be reliably applied to data at hand.

We aim to break these barriers to the routine, rigorous testing of new phylogenetic comparative methods by making robustness testing more straightforward. We will develop a suite of tests that can easily be deployed to assess the performance of a new method and compare its behavior against other methods designed for the same questions. The product will be called **PhyCoMB: Phylogenetic Comparative Methods Benchmarking** (Fig. 1). Benchmark tests are standard practice in many fields, from computer hardware to bioinformatics. For example, BALiBASE (the Benchmark Alignment dataBASE; Thompson et al. 1999, 2005) has been widely adopted for assessing the performance of multiple sequence alignment algorithms. Further examples of standardized benchmark tests are Assemblathon and BUSCO for genome assembly (Bradnam et al. 2013; Simão et al. 2015) and T. Warnow’s resources for phylogeny estimation (<http://www.cs.utexas.edu/~phylo/datasets>). PhyCoMB’s structure will mirror other benchmark suites:

- *Tasks* are specific questions within the domain of phylogenetic comparative methods. They include tests of whether evolving traits affect rates of speciation and extinction (our focal task here), clade-specific diversification rate shifts, irreversible evolution, and discrete trait correlations.

- *Methods* are procedures designed to accomplish a task. They consist of a model or other technique (e.g., BiSSE, sister clades), plus a statistical inference framework (e.g., AIC, model averaging, sign test).

- *Elements* are collections of trees and optionally traits, all with the same properties. They may arise from empirical or simulated data. A method is applied to each tree/trait item within an element.

- *Reference sets* are chosen to present particular challenges to a method, within the context of a task. Each is a group of elements. For example, the ‘power’ reference set includes elements with small and large trees. The ‘pseudoreplication’ reference set contains trees in which traits change rarely and are accidentally associated with diversification shifts. The ‘diversification heterogeneity’ reference set contains trees with complex shifts in speciation, on which neutral traits are evolved.

- *Curated benchmarks* are the heart of PhyCoMB. The underlying database will eventually contain dozens of reference sets with many thousands of elements, but we will carefully select a subset to be the benchmark test for a task. For example, the benchmark for trait-dependent diversification might consist of 16 elements forming a progression of increasingly challenging power tests, 24 elements with different forms of pseudoreplication, and 36 elements with different forms of diversification heterogeneity and neutral trait evolution. Manual curation will maximize the diversity

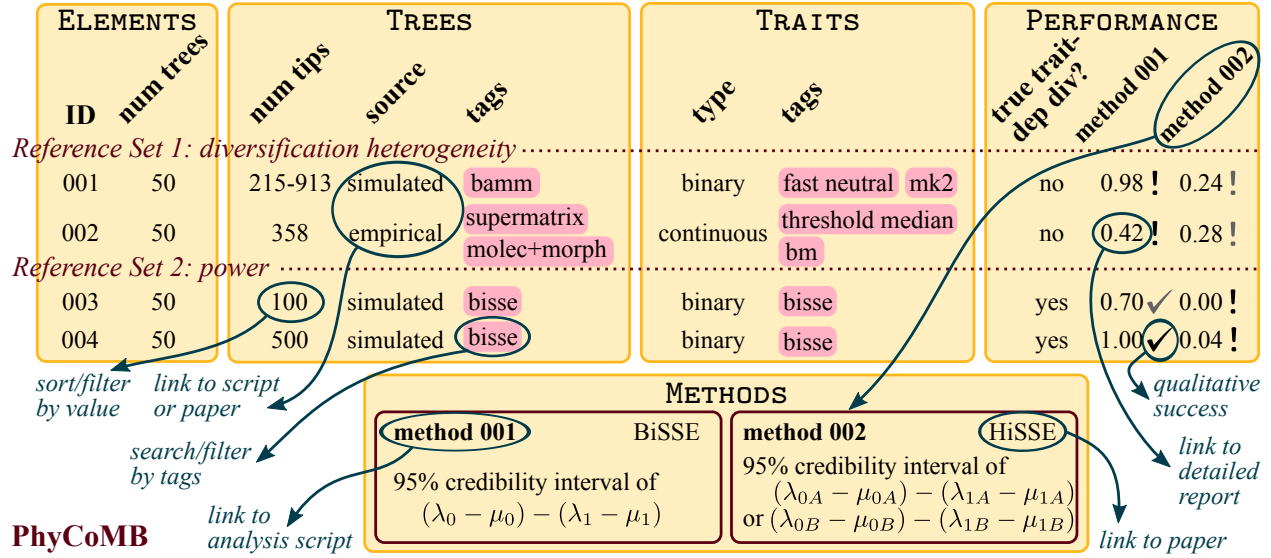


Figure 1: Vision for PhyCoMB. The Phylogenetic Comparative Methods Benchmark database will provide test datasets and reports of method performance. Example annotations are shown for four elements (comprising trees and traits), and interactive behavior of the web interface is noted. Compared are two methods, which test for trait-dependent diversification by assessing the difference in estimated net diversification rates. Performance is reported as the proportion of trees on which significant trait-dependent diversification is inferred. Reference set 1 reveals the BiSSE-based method is prone to incorrectly associate two kinds of neutral traits with diversification rate, while the HiSSE-based method is less so. Reference set 2 reveals that HiSSE has much lower power than BiSSE in this test. (Note that HiSSE is much more effective under model averaging; Beaulieu & O’Meara 2016.)

of challenges faced by a method while keeping the amount of testing manageable. Clearly-defined benchmarks will allow the performance of different methods to be easily compared by those developing new methods and those applying them to empirical systems.

- *Annotations* will make the results obtained from PhyCoMB easier to explore and interpret. Each element will be labelled with, e.g., the number of tips and sampling completeness, the type of diversification process for simulated trees, construction methods for empirical trees, and the model of evolution for simulated traits. Contributors will note the origin of the element’s data (literature citation or generating script).

- *Reports* will allow developers to see how a new method performs relative to others, and they will allow users to understand the strengths and weaknesses of a variety of methods for a given task.

For an empiricist, PhyCoMB will provide not only structured information about methods, but also the means for directed testing. This includes scripts for creating test data that reflect properties of real data, and for running methods on them. It can thus enhance the impact of empirical work by aiding demonstrations of the robustness of a method for the data and question at hand. Such improved connections between methods development and empirical use will bring greater stability and transparency to the field (Cooper et al. 2016).

The first life-stage of PhyCoMB will be initiated by this proposal. We will generate a collection of elements and benchmarks (soliciting ideas during the workshop; see Broader Impacts), and we will build a web-based interface for developers and end users. This first stage will demonstrate substantial benefits to the phylogenetics community: straightforward comparisons of old and new approaches, reproducibility of results, early warnings of methodological weaknesses, and highlights of when methods are particularly powerful and robust. The second stage that we envision for PhyCoMB is gradual community adoption beyond the timeframe of the proposed work. PhyCoMB will evolve through contributions of new elements, as future work uncovers complex scenarios that challenge the assumptions of our methods.

2.2 References

- Beaulieu, J.M., B.C. O'Meara, 2016. Detecting hidden diversification shifts in models of trait-dependent speciation and extinction. *Systematic Biology* 65:583–601.
- Bradnam, K.R., J.N. Fass, A. Alexandrov, P. Baranay, M. Bechner, I. Birol, S. Boisvert, J.A. Chapman, G. Chapuis, R. Chikhi, et al., 2013. Assemblathon 2: Evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience* 2:1.
- Cooper, N., G.H. Thomas, R.G. FitzJohn, 2016. Shedding light on the dark side of phylogenetic comparative methods. *Methods in Ecology and Evolution* 7:693–699.
- Simão, F.A., R.M. Waterhouse, P. Ioannidis, E.V. Kriventseva, E.M. Zdobnov, 2015. BUSCO: Assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* 31:3210–3212.
- Thompson, J.D., P. Koehl, R. Ripp, O. Poch, 2005. BALiBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics* 61:127–136.
- Thompson, J.D., F. Plewniak, O. Poch, 1999. BALiBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics* 15:87–88.

2.3 Later thoughts

This subsection wasn't part of the proposal. But it explains some ways in which the current plan differs from what was proposed.

Task needs to be defined a bit more carefully. For example, even within the realm of state-dependent diversification, one approach/question is hypothesis testing and another approach/question is parameter estimation. Results for those questions would be reported differently. And even a given method on a given element could perform better on one question than the other. Below, *Task* is assumed to include this refinement. But we could instead have a separate *Question* component that intersects with *Task* to specify the problem.

If we are going to include methods that co-infer the phylogeny along with answering the comparative methods question, we'd need to allow for input data to be a sequence alignment rather than a tree. That is a potentially promising direction for methods development, but it's beyond the immediate scope of PhyCoMB.

3 USERS

Different kinds of users will interact with PhyCoMB in different ways. They will have different goals and permissions. One person could be a different type of user at different times, e.g., I could do some work as an Administrator, then add data as a Contributor, then look through results as a Viewer.

3.1 Viewer

Can browse PhyCoMB contents through the web interface. Cannot make changes that affect anyone else. No login required.

3.2 Contributor

All the functionality of a Viewer. Additionally, Contributors can see and use the parts of the web interface for uploading *Elements*, *Methods*, and *Performance*. Login required.

Should the newly uploaded material be immediately integrated into the database and visible to all users, or should it not be published until an Administrator approves the submission?

Contributors can't delete information, but perhaps they could flag *Elements*, etc. that are no longer useful, so an Administrator could delete them.

3.3 Administrator

Can approve the uploads and suggested deletions made by Contributors. Can also make changes to the structure (e.g., which *Elements* are in a *BenchmarkSet*). Perhaps the interface will be different than what a Viewer or Contributor sees. Login required.

4 WORKFLOWS

Here are some examples of how different kinds of users might commonly interact with PhyCoMB.

4.1 Viewer workflows

4.1.1 Explore performance of various methods for a particular task

The user's goal is to learn about how various methods perform for questions they're interested in. This means seeing which *Methods* are available for a given *Task*, and their *Performance*, summarized in a *Report*. This is the most important user interface component to design well.

1. Select the desired *Task*. See ?? for what that might look like.
2. Select the desired *Methods*. See ?? for what that might look like.
3. Select the desired *Elements*. See ?? for what that might look like.
4. Browse the *Report*. See ?? for what that might look like and the types of interactivity needed.
5. Download the *Report*. See [Sect. 7.3](#).

4.1.2 Explore performance of a particular method for various tasks

The user's goal is to learn about how a method they're interested in performs for various questions. This means seeing which *Tasks* are addressed by a given *Method*, and its *Performance*, summarized in a *Report*.

1. Select the desired *Method*.
2. Select the desired *Task*, or default to all relevant *Tasks*.
3. Select the desired *Elements*. See ?? for what that might look like.
4. Browse the *Report*. See ?? for what that might look like and the types of interactivity needed.
5. Download the *Report*. See [Sect. 7.3](#).

4.2 Contributor workflows

4.2.1 Contribute a new method

The user's goal is to add a *Method* to the ones available, thinking that it will be useful for other people.

1. Browse existing *Methods* to determine if it's already available.
2. Provide a script that completely runs the *Method* when provided with an *Element*, returning the value for *Performance* of a *Task*.
3. Fill in a form with basic information. This info helps to categorize the *Method* in the database, and it will be passed along to future users who see the *Method*.
 - Specify what *Task* it is for. Maybe also what *ReferenceSet* it is most interesting for, e.g., if method was designed to fix a particular weakness.
 - Explain why the new *Method* is worthwhile—what was the purpose in creating it?
 - Provide info that becomes values in columns: what category of model/technique, what kind of statistical inference, etc. (see ?? for the required fields).
4. Upload *Performance* results for at least the *BenchmarkSet* of *Elements*.
5. See a *Report* to confirm that things look correct.
6. Request that an Administrator accept the new *Method*?

4.2.2 Contribute a new element

The user's goal is to add an *Element* to the ones available, thinking that it will be useful for other people.

1. Browse existing *Elements* to determine if it or something similar is already available.

- (a) If the new *Element* is better than an existing one, include a deprecation request for the Administrator?
- 2. Provide data files for the new *Element*.
 - (a) Generating script, if applicable. Link to empirical data source. etc. Needs to be completely documented and reproducible.
 - (b) The actual tree and trait files. Or if using *Tree* and/or *Trait* from another *Element*, link to that.
- 3. Fill in a form with basic info: This info helps to categorize the *Element* in the database, and it will be passed along to future users who see the *Element*.
 - Specify what *Task* it is for, and what *ReferenceSet* it will belong to.
 - Explain why the new *Element* is worthwhile—what was the purpose in creating it?
 - Provide info that becomes values in columns: simulated/empirical, sim regime, etc. (see ?? for the required fields).
 - Can some info be obtained automatically from the uploaded files? e.g., tree size
- 4. Upload *Performance* results for all(?) *Methods* that apply to that *ReferenceSet*.
- 5. See a *Report* to confirm that things look correct.
- 6. Request that an Administrator accept the new *Element*.

4.2.3 *Contribute a new performance result*

This could be part of the workflow in contributing a new *Method* or *Element*. But it could also be done separately. In that case, find combinations of *Method* + *Element* that haven't yet be run, and fill in the values.

4.3 Administrator workflows

4.3.1 *Alter the groupings of elements*

Alter the *Elements* in a *ReferenceSet*. Alter the *Elements* and/or *ReferenceSets* in a *BenchmarkSet*.

4.3.2 *Approve an addition from a Contributor*

Check the new *Method* or *Element* and publish it (make it available to all users) if it looks good. Maybe a general form for other requests, e.g., to deprecate an *Element*?

4.3.3 *Deprecate a method or element*

Don't entirely delete from the database, but hide from most operations?

4.3.4 *Manage users*

Field requests from users who wish to become Contributors? Or just let that be automatic, and then revoke Contributor status if someone causes trouble?

5 VIEWS [TODO]

6 DATABASE TABLES [TODO]

7 DOWNLOADS

After users view and filter the results, they may want to download information for use offline. Each download should be a single zipped file, which contains only plain text files with a logical directory/folder structure.

7.1 Obtaining testing files

When an *Element* is downloaded, the user should receive:

- Information about it, written in an auto-generated text file.
- The script that generated it, if any.
- The tree files and (if relevant) trait files. Could have an option to omit these if the script is sufficient.

These files should all have sensible names.

When multiple *Elements* are downloaded together, each should be in a separate directory. If they have tree or trait files in common, could have an option to use symlinks instead of duplicating the content. Also, a spreadsheet (CSV file, one row per *Element*) should be included so it's easy to see which *Elements* have which attributes (columns in *Tree*, membership in *ReferenceSet*, etc.).

If an entire *ReferenceSet* is requested for download, each *Element* within it should be in a separate directory.

If an entire *BenchmarkSet* is requested for download, each *ReferenceSet* within it should be in a separate directory.

7.2 Obtaining methods scripts

When a *Method* is downloaded, the user should receive:

- Information about it, written in an auto-generated text file.
- The script to run it.

When multiple *Methods* are downloaded together, each should be in a separate directory.

7.3 Obtaining performance reports

The user should be able to download a CSV file that looks basically like the results table *Report* (?? or ??). Either the full report could be requested, or only to include those rows (*Elements*) and columns that are visible after interacting with the report view.

Are additional columns needed, e.g., directory names of *Elements* and *Methods* if they are downloaded?