Department of Ecology & Evolutionary Biology

University of Michigan, Ann Arbor

830 North University Avenue

Ann Arbor, MI 48109-1048

2 December, 2016

Dear Bioinformatics editor,

We are submitting a revision of the manuscript entitled *phyx: Phylogenetic tools for Unix* for consideration in Bioinformatics. We have addressed all of the comments and updated the software, manual, and manuscript accordingly. The specific responses to the reviewer queries are presented below.

Thank you for considering our manuscript for Bioinformatics.

Sincerely,

Stephen Smith, Joseph Brown, and Joseph Walker

**Reviewer: 1**

*a). major*

*1. Supplemental part 2.1 - The authors state that phyx outperforms Gblocks and Phyutility in cleaning alignments. It isn’t clear from the text, only by looking at the figure, that this comparison is based only on speed, and not quality. There are a lot of configuration options that can be applied to Gblocks in particular that improve performance and final quality. Considering this is one of the authors’ chosen example cases, and it’s in the supplementary information, it would be nice to see the authors state specifically what settings they used, and how those compare to the settings in phyx. More specifically, “what” is phyx doing when it cleans to sequences.*

We agree that the text was vague, and have therefore updated the supplemental information to make this comparison more explicit.

*2. In general, I think the support and case examples for some of the programs need to be more thoroughly documented. I recognize that is almost impossible to do in a short application note, but I think clear examples on the Github Wiki would suffice. Some of these individual applications have complex and sophisticated algorithms implemented and no documentation that I can find as to what they are actually doing. I believe it is important to show some examples that validate the results being achieved. Using the sequence cleaning example above, showing an output alignment, that shows a comparable final product, between the utilities described, not just a performance increase would help a lot. This is something I believe could be done over time, but I think should at least be addressed someplace in the application note or supplementary information.*

We have completely rewritten the manual (available from the GitHub repo) to include details on individual programs. In addition, we have added tests to the make workflow which ensures that the produced results are correct.

b) minor.

*1. I’m placing this under minor, as the authors clearly state that Linux is the primary target platform, which is logical given the make style of these tools and the inherent utility of their use in a pipeline framework. However, I think addressing this comment would go a long way in helping broaden the user base for these tools. The OS X version failed to compile under several different version of OS X. This includes Mavericks, Yosemite and El Capitan. If there are specific Sierra requirements, they are not mentioned on the Wiki or the Github repo. With one of the primary goals of this suite of utilities is to provide a framework that can work towards being a standardized part of phylogenetic analyses for the sake of reproducibility, then in order to reach the vast majority of evolutionary biologists, support for OS X is critical. I fully recognize the difficulty in developing for OS X, as there are many things Apple does to make this difficult compared to normal POSIX environments, though I believe it’s something the authors will need to overcome to see these great tools reach the audience they are targeting. One possible solution might be to release precompiled OS X binaries in order to minimize some of these challenges.*

We have simplified the installation on Mac OS X and have tested and installed this on roughly 12 Mac OSX machines including versions from 10.9-10.12. We have also clearly documented to different methods for doing so. Hopefully, this will address these concerns. Furthermore, there should be compilation of most of the phyx programs even if some do not compile. Precompiled binaries are challenging when using additional libraries as Mac does not allow for -static compile flags. Nevertheless, hopefully the simplified installation process improves this greatly. I would encourage the reviewer to post a bug report on github if they run into this problem again.

*2. This line is missing a space after PAL2NAL: We found that Phyx was faster than PAL2NALSuyama et al. (2006) under each condition.*

We have fixed this.

*3. In this line, reduce should be reducing: Also, Phyx does not require that the sequences be in the same order, thus reduce error and specifically avoiding aligning a nucleotide sequence with something other than its corresponding amino acid alignment.*

We have fixed this.

*4. In the figure caption for Figure 2, alignments is mis-spelled. - Comparison of timings to convert protein alignemtns to their corresponding codon alignments.*

We have fixed this.

**Reviewer: 2**

*phyx fills a gap in the software ecosystem that will be useful to a wide range of tasks, from tedious daily data wrangling to designing large scale analysis pipelines. I can see several places it will fit into my ongoing research. The SuppInfo provides compelling examples for writing tight phyx pipelines and demonstrates that the phyx programs are efficient when compared to alternative software. The manuscript should be published essentially as-is, but I recommend the authors take steps towards making the software more user friendly (see below).*

We appreciate the comments from the reviewer. Hopefully we have addressed the concerns below.

*Manuscript:* *pg1, line 42 "individual packages" to "individual package"*

*Installation: I installed phyx on OS X 10.11 w/ brew. I had to manually set HNLOPT, HARM, and HOMP to Y to compile the make targets that used armadillo and nlopt. Not sure what went wrong, but it looks like maybe configure couldn't detect the library installs on my system.*

We have simplified the installation for Mac as discussed above. Hopefully, this will solve this problem.

*Any effort towards registering the package through apt-get/brew will help attract more users, though I appreciate registration may be complicated.*

Thanks for the suggestion! This is definitely on our radar for future releases.

*Software: Many programs display little or nothing when executed with bad input/syntax. Some programs segfault when run without arguments. More informative error messages would help, in addition to suggesting the use of the `--help` flag. `--help` arguments from the repo and the manual don't match perfectly (see below).*

This was a terrific suggestion. We have made it so that without arguments, those programs that wait for input timeout within two seconds so that they don’t hang. This should resolve this problem.

*Manual: The manual is off to a great start, but needs to be brought in sync with the current state of the repo -- e.g. folder names don't match and some programs appear to be missing. Perhaps it's an out-of-date LaTeX pdf?*

We have completely rewritten the manual (available from the GitHub repo) and made it up to date.

*The current repo stores examples in in `./example\_files/` not `./Example/`*

We have fixed this.

*Manual example folders missing from repo:*

*`pxaatocdn\_example` missing*

*`pxconsq\_example` missing*

*`pxs2fa\_example` missing*

*`pxs2nex\_example` missing*

*`pxs2phy\_example` missing*

*`pxvcf2fa\_example` missing*

*`pxt2new\_example` missing*

We have addressed this.

*Manual example names that mismatch repo folder names:*

*`pxnw` to `pxnw\_example`*

*`pxsw` to `psxw\_example`*

*`pxtlate` to `pxtlate\_example`*

*`pxbp` to `pxbp\_example`*

*`pxmrca` to `pxmrca\_example`*

*`pxnni` to `pxnni\_example`*

We have addressed this.

*Missing programs*

*`pxnni`*

*`pxtscale`*

We have addressed this

*pg9 The program options are out of date when compared to `**pxrms –help`.*

We have fixed this.

*pg10 `-r List.txt` to `-f List.txt`*

We have fixed this.

*pg13 I couldn't determine out what the output from `pxmrca` meant (although I did finally figure out who KIM, LEE, and THURSTON were). Maybe add more description to the manual and --help output.*

We have fixed this by updating the help message and manual.

*Pg14 Not sure what `pxmrcacut` does, but I could not view the output Newick string in FigTree or plot it in R using ape.*

We cannot replicate this problem. The help clearly states that the program extract subclades, and the example files (and tests) all produce the desired result.

*pg22 The `pxcontrates` example gave me nan values when using the provided files*

Unfortunately, we cannot reproduce this result. We have updated the compilation software, and so hopefully these updates have improved this situation. If the reviewer continues to have trouble, please place an issue on github and we will work through the specific issue. As we have added tests, a compilation problem should manifest as an error in the test.

**Reviewer: 3**

*#General*

*- I think one of the interesting and innovating aspects of the authors' approach is the modeling after the POSIX-style tools, which makes them eminently suitable for direct use in bioinformatics pipelines in a way that other programs mentioned above do not.*

*- Furthermore, the collection of functionalities/operations is unique and useful, the API/user-interface is well-designed, and most importantly, well-documented.*

We appreciate the kind comments about the software.

# Major Revision Issues

*## Tests*

*- Unfortunately, neither the publication nor the manual make mention of any unitttests or indeed tests of any kind. Furthermore, looking at the repository, there is also no evidence of tests of any kind.*

*- I am afraid that I cannot recommend this software for publication without demonstration and documentation of tests. There is no reason for academic software to be held to lower standards that any other kinds of software. This software will be used by people who do not know about the internals of the code, and the results of THOSE people's work, in turn, will be cited/used by other people who may not know about the programs in th first place. At each stage there is trust in that certain critical internals have been given due diligence and care by the previous stage. This trust is essential to our community. When it comes to software development, the trust is in the form that the programs are doing what the authors claim that the programs are doing. Validating that this is indeed the case (and demonstrating that validation) is the authors' responsibility to the community, especially if they are aiming to gain citable academic publication out of this.*

*- Tests should, at the very least, demonstrate that the programs are doing what they are supposed to do given some minimal canonical input. Testing C++ programs dealing with complex data like this is difficult, but certainly not only possible, but dedicatedly pursued by responsible authors (e.g., NCL, phycas).*

*- I recommend testing input/ouput (e.g., by round-tripping data files and ensuring content remains as expected) explicitly and separately from manipulations and operations.*

*- With the latter, simple examples from the manual/documentation/paper for each one will do to start.*

*- In all cases, if semantic-checking of output is too challenging, a simple pattern matching will do (the latter is fragile, in that small tweaks to the programs' writers in terms of spacing, etc., will result in tests failing, but this can easily if tediously be fixed).*

*- More advanced testing would be nice, e.g., for incorrect input and so on. But I recognize that this would be a lot of work and that authors may want to develop these later.*

This was a terrific suggestion, and an embarrassing oversight on our part. We have added tests for all programs to ensure that the results produced are correct. These will continue to be updated with more tests and eventually advanced testing and unit testing as we continue development.

*## Discussion of Other Software*

*"However, each individual packages is limited by the file formats supported, memory requirements, requiring the loading of separate environments (i.e. R or python), or utilizing a graphical user interface which may not be conducive to high throughput processes."*

*- I am not sure what the authors mean by "limited by memory requirements" and how \*any\* program is not affected by this. It is true, that different languages are inherently more or less efficient in memory requirements, but these are also subject to programming logic (a bad C++ program can end up using more memory than a good R or Python program) and usage (if both a C++ program and a Python program can read a 10 1-million leaf trees, does it matter?).*

*- This statement is also, frankly, incorrect its characterization. DendroPy is (a) explicitly written to support a very broad range of formats (NEXUS, NEWICK, NeXML, FASTA, PHYLIP, etc.), (b) is not limited by memory requirements any more or less than any program that runs on a computer (and also provides stream-centric processing infrastructure to minimize memory usage), (c) and does not require loading of separate environments, and does not utilize a graphical user interface.*

*- As a more general point, I recognize the perceived need to justify the publication of software by emphasizing a new role or need that it fulfils that no other existing software does. However, I think this is a misguided notion: there is no problem with developing and publishing software that replicates the functionality of existing software. The community is all the richer for it. Even if a package provides the exact same functionality as one or more existing applications, it almost always has sufficient differences in other ways to be more beneficial to some users/people than others. In the worst case, it can provide critical redundnancy to check the results of other programs or longevity in case other programs become abandoned by the developers. As such, while opening up a paper with a review of limitations of existing software is traditional, this is not necessary. In this case, I feel that the authors pursued tradition and paid the price in clarity and accuracy, incompletely and incorrectly describing existing software. Thus, I recommend the authors simply provide an overview of existing programs, and unless they want to go into actual detail on the differences, refrain from making generalizations about limitations and simply introduce their program as another hat in the ring so to speak.*

Another great suggestion! We have toned down the text to stress that different tools are preferred for different workflows (that is, tools are complementary rather than competitive), and that phyxis an efficient tool for commandline workflows.