15-Nov-2016
Manuscript ID: BIOINF-2016-1452
Title: phyx: Phylogenetic tools for Unix

Dear Dr. Smith,

The reviews of your manuscript are now in hand for Bioinformatics and can be found at the foot of this e-mail.

Based on the reports of the referees, the paper has been rejected for publication in its present form. However, the Associate Editor, Janet Kelso, considers that if the paper were substantially rewritten taking into account all the referees' comments it may become acceptable for publication. As you will see, the reviewers were generally positive about the need for software to standardise the processing of phylogenetic datasets. However, in its current form the software is not sufficiently mature to consider for publication: all reviewers encountered significant difficulties with installation and use. Substantially better documentation (including installation instructions, explanations of the implemented function, use cases and test datasets) and proper testing of functions provided in the software are essential for publication. I invite you to address all reviewer comments in a revised manuscript and updated software package.

We ask that major revisions are submitted within one month ideally, but the system will allow a revised paper to be submitted within 90 days of the original decision date.

Please summarise your changes for the editor indicating which changes you have made and which changes you do not wish to make and why. This can be done either in a Response to Reviewers file uploaded alongside your revised manuscript or through the Author Centre where you can enter your responses directly.

Please submit your revised version through the Author Centre by clicking on the purple button 'Click here to Submit a Revision' in the Bioinformatics ScholarOne Manuscripts web site (https://mc.manuscriptcentral.com/bioinformatics).

At this major revision stage we ask that you upload the following revised manuscript files:

EITHER: (i) A .doc or .rtf file of the revised manuscript, with all tables, figures, schemes and equations inserted in the document.

OR: (ii) All necessary LaTeX files that will be required by the typesetter (including bioinfo.cls, bib, .bst and .ps files).

Please can you mark-up the changes made after revision by using the track changes function or highlighting these in red text.

NOTE: Please upload your final version of supplementary materials without any

changes marked. This should be in pdf or Word format, not LaTex.

Please note that if you decide that you would like your figures printed in colour a charge of £350 per colour figure applies. If appropriate, you will be invoiced after your paper has been published in the print journal.

As a reminder, please also note the following excess page charges. You will be notified of any excess page charges when you receive your proofs:

For Original articles - £100/$165 per excess page (over 7 published pages)
For Discovery notes - £100/$165 per excess page (over 4 published pages)
For Application notes - £100/$165 per excess page (over 2 published pages)

On behalf of the Bioinformatics Associate Editor, Janet Kelso, I want to thank you for selecting Bioinformatics to present your work.

Best regards,
Alison Hutchins
Bioinformatics

Here are the comments of the reviewers:

Reviewer: 1

Comments to the Author
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.

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.

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.

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

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.

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

Reviewer: 2

Comments to the Author
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).

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.

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

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).

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?

The current repo stores examples in in `./example_files/` not `./Example/`

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

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`

Missing programs
`pxnni`
`pxtscale`

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

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

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.

pg14
Not sure what `pxmrcacut` does, but I could not view the output Newick string in FigTree or plot it in R using ape.

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

Reviewer: 3

Comments to the Author
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.

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.

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.
