Supplementary information for phyx (Phylogenetic tools for Unix)

Joseph W. Brown[†], Joseph F. Walker[†], and Stephen A. Smith

Department of Ecology & Evolutionary Biology
University of Michigan, Ann Arbor, MI 48109, USA
[†]Equal contribution

1 Example pipeline

As described above, phyx uses a stream-centric approach to input and output that allows for programs to be linked together without the use of intermediate files. Here, we illustrate how four phyx programs can be linked via piping and a simple shell loop to perform a full analytical pipeline:

1. Clean alignments individually using a Unix for loop (pxclsq).
2. Concatenate cleaned alignments into a supermatrix (pxcat).
3. Infer a ML tree (RAxML (Stamatakis, 2014)).
4. Re-root the tree on the outgroups (pxrr).
5. Remove the outgroups (pxrmt).

which would take the form:

```
for x in *.fa; do pxclsq -s $x -p 1.0 -o $x.phyx; done && pxcat -s *.phyx -o out.concat
-p models && raxml -T 2 -m GTRCAT -p 12345 -q models -s out.concat -n RAxMLout && pxrr
-g s1,s2 -t RAxML_bestTree.RAxMLout | pxrmt -n s1,s2 -o trimmed.tre
```

2 Performance

Although we see phyx as a convenient complement to existing tools, there are clear instances where a phyx implementation is faster and more efficient in terms of memory use. We briefly describe below the performance of some phyx programs relative to other existing tools for common tasks in phylogenomics workflows.

2.1 Sequence cleaning

Cleaning sequences to ensure a certain level of matrix occupancy has become common place in many phylogenomic pipelines (Dunn *et al.*, 2013; Yang and Smith, 2014). Here we compare two programs Gblocks v0.91b (Castresana, 2000) and phyutility v2.2.6 (Smith and Dunn, 2008) to the sequence cleaning procedure of phyx (pxclsq). The file sizes ranged from 10 sequences (234 kB) to 100,000 sequences (2.3 GB), with each sequence being 23,950 base pairs in length (Figure S1). The implementation for sequence cleaning in pxclsq is highly similar to phyutility, in that it focuses on column occupancy (proportion of missing data) as a means of determining whether a sequence region should be cleaned (that is, removed). However, the two programs differ in one key respect. For amino acids, pxclsq treats the ambiguous character “X” as missing data, whereas phyutility does not. We therefore restricted analyses to DNA to avoid this inconsistency. Additionally, although Gblocks implements a more sophisticated range of methods for cleaning sequences, we restricted (using the -b1 argument) cleaning to focus solely on column occupancy to make timings directly comparable. All programs were instructed to remove sites if they contained greater than 60% missing data. We found that phyx outperformed both Gblocks and phyutility for all dataset sizes. Also, for the largest dataset, phyutility was not able to clean the dataset due to a memory allocation error. The test was conducted on a laptop containing 4 processors and 16 GB of memory, with 14 GB of that memory being allocated for phyutility.

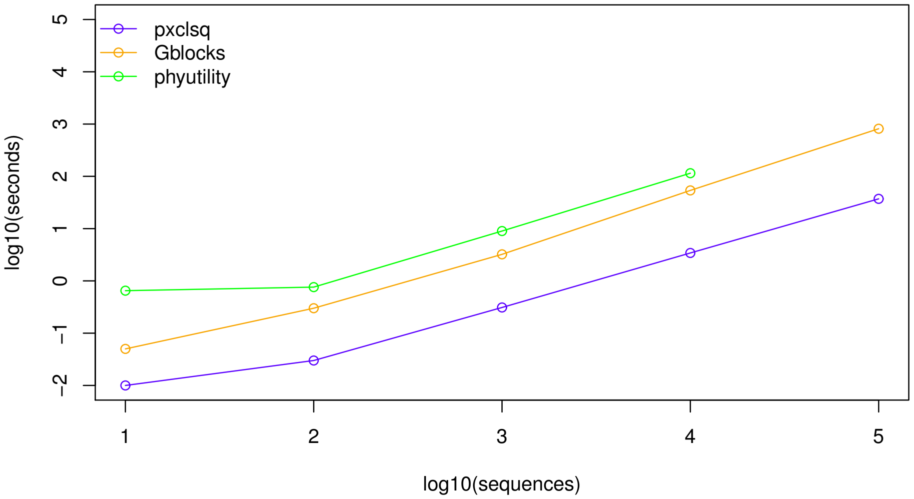

Figure S1: Comparison of alignment cleaning timings.

2.2 Conversion of proteins to codons

Constructing a codon alignment from a protein alignment and its associated nucleotide matrix is necessary for both proper phylogenetic modelling (partitioning) and conducting positive selection tests. Here we tested the processing efficiency of files consisting of 801 amino acid residue sequences for between 10 and 10,000 taxa (file sizes ranging from 8 kB to 77 MB). We found that phyx program aa2cdn was faster than PAL2NAL v14 (Suyama *et al.*, 2006) for all alignment sizes (Figure S2). Importantly, unlike PAL2NAL, aa2cdn does not require that the sequences be in the same order across files, thus avoiding the possiblility of aligning a nucleotide sequence with something other than its corresponding amino acid sequence.

2.3 MCMC log concatenation and resampling

MCMC log files from Bayesian phylogenetic analyses have become common phylogenetic objects. Such analyses are typically replicated (to ensure convergence of the MCMC chains) and run for many millions of generations (to achieve adequate effective sample sizes), resulting in several very large text files, each of which invariably involve a burnin phase (samples that are discarded before sumamrization). Prior to parameter summary, these log files are typically concatenated while removing the burnin phase and potentially resampling (thinning) the individual logs because of memory constraints. The phyx program pxlog carries out these operations on both tree and parameter logs. To assess the performance of pxlog, we compared it to two versions of logcombiner from the BEAST package (Drummond and Rambaut, 2007; Bouckaert *et al.*,

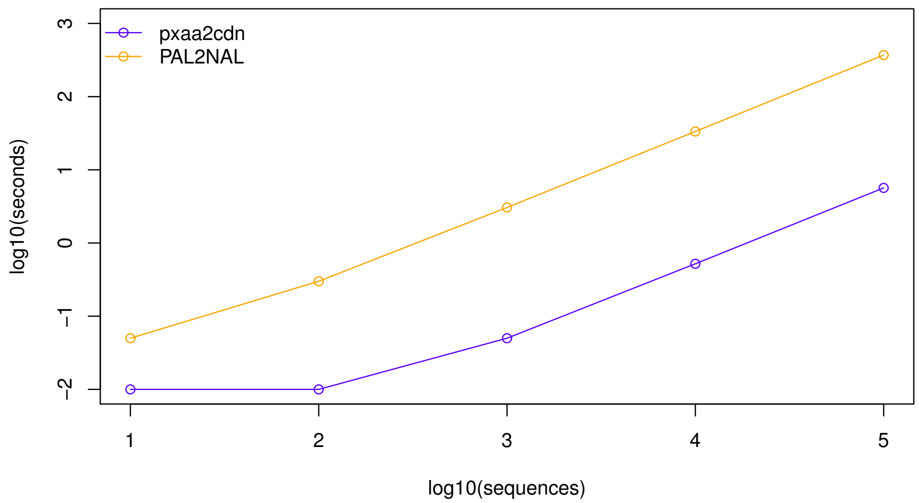

Figure S2: Comparison of timings to convert protein alignments to their corresponding codon alignments.

2014). We ran phylogenetic in analyses in `BEAST` using the data from Magallón *et al.* (2015), a data set which consists of 798 taxa. Five replicate MCMC analyses were performed, each running for 100 million generations and sampling trees every 5000 generations (for a total of 20000 trees sampled in each analysis). In preparation for tree summary, we discarded the first 25% of samples, and further thinned the chains to every 10th sample (for a total of 1500 post-burnin samples per analysis).

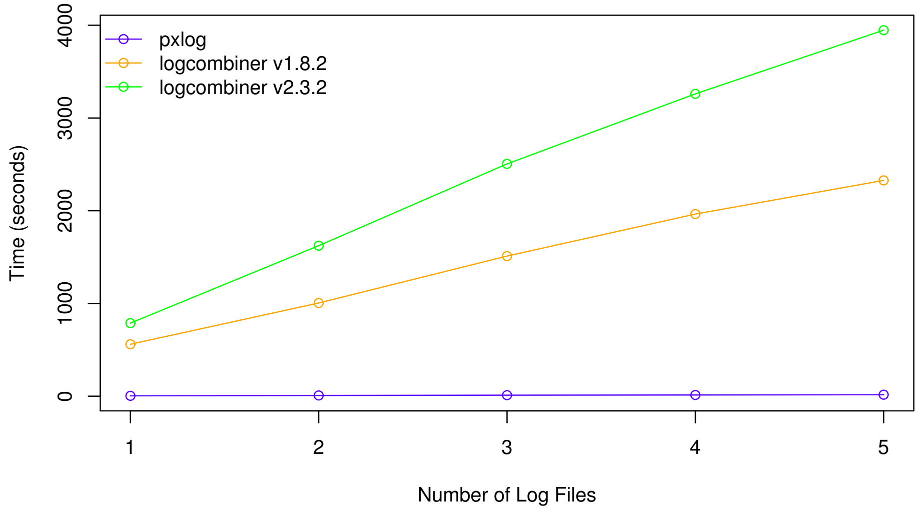

Figure S3: Comparison of MCMC log manipulation timings. Each log file is 2.6 GB and contains 20000 trees.

The timings for the log manipulations by the various programs are displayed in Figure S3. `pxlog` executed much faster than either version of `logcombiner` for any number of input files, taking only a few seconds compared to up to over an hour for the alternative tools. More revealing, however, was the memory usage of the various programs. `pxlog`, being stream-centric (and hence holding only a single tree in memory at any particular instant), consumed only 600 kB of RAM, despite the individual log files totalling 2.6 GB. `logcombiner` is a java-based tool to which we allocated 40 GB of RAM. `logcombiner v1.8.2` was far more memory efficient than the newer version, consuming 2.4 GB of RAM for the full 5 file concatenation. `logcombiner v2.3.2`, on the other hand, consumed 32.6 GB of RAM while executing far more slowly.

References

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), 1–6.

Castresana, J. (2000). Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Molecular Biology and Evolution*, **17**(4), 540–552.

Drummond, A. J. and Rambaut, A. (2007). BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, **7**(1), 1–8.

Dunn, C. W., Howison, M., and Zapata, F. (2013). Agalma: an automated phylogenomics workflow. *BMC Bioinformatics*, **14**, 330.

Magallón, S., Gmez-Acevedo, S., Snchez-Reyes, L. L., and Hernández-Hernández, T. (2015). A metacalibrated time-tree documents the early rise of flowering plant phylogenetic diversity. *New Phytologist*, **207**(2), 437–453. 2014-18158.

Smith, S. A. and Dunn, C. W. (2008). Phyutility: a phyloinformatics tool for trees, alignments and molecular data. *Bioinformatics*, **24**(5), 715–716.

Stamatakis, A. (2014). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, **30**(9), 1312–1313.

Suyama, M., Torrents, D., and Bork, P. (2006). PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments. *Nucleic Acids Research*, **34**(suppl 2), W609–W612.

Yang, Y. and Smith, S. A. (2014). Orthology inference in nonmodel organisms using transcriptomes and low-coverage genomes: Improving accuracy and matrix occupancy for phylogenomics. *Molecular Biology and Evolution*, **31**(11), 3081–3092.