**Phylo-Node: a molecular phylogenetic toolkit using Node.js**

Damien M. O'Halloran\textsuperscript{1,2}*

1. Department of Biological Sciences, The George Washington University, Science and Engineering Hall, Rm 6000, 800 22nd Street N.W., Washington DC 20052, USA.

2. Institute for Neuroscience, The George Washington University, 636 Ross Hall, 2300 I St. N.W. Washington DC 20052, USA.

*Corresponding author address:

Damien O'Halloran, The George Washington University, 636 Ross Hall, 2300 I St.

N.W. Washington DC 20052, USA.

Tel: 202-994-8955

Fax: 202-994-6100

EMAIL: damienoh@gwu.edu
ABSTRACT

Background: Node.js is an open-source and cross-platform environment that provides a JavaScript codebase for back-end server-side applications. JavaScript has been used to develop very fast, and user-friendly front-end tools for bioinformatic and phylogenetic analyses. However, no such toolkits are available using Node.js to conduct comprehensive molecular phylogenetic analysis.

Results: To address this problem, I have developed, Phylo-Node, which was developed using Node.js and provides a stable and scalable toolkit that allows the user to go from sequence retrieval to phylogeny reconstruction. Phylo-Node can execute the analysis and process the resulting outputs from a suite of software options that provides tools for sequence retrieval, alignment, primer design, evolutionary modeling, and phylogeny reconstruction. Furthermore, Phylo-Node provides simple integration and interoperation with other Node modules to develop workflows across multiple components and languages using Node inheritance patterns and a customized piping module to support the production of diverse pipelines.

Conclusions: Phylo-Node is open-source and freely available to all users without sign-up or login requirements. All source code and user guidelines are openly available at the GitHub repository: https://github.com/dohalloran/Phylo-Node

Keywords: Node.js, JavaScript, phylogenetics
BACKGROUND

The cost of whole genome sequencing has plummeted over the last decade and as a consequence, the demand for genome sequencing technology has risen significantly [1]. This demand has meant that producing large complex datasets of DNA and RNA sequence information is common in small research labs, and in terms of human health this boom in sequence information and precipitous drop in sequencing costs has had a direct impact in the area of personalized medicine [2-5]. However, once the sequence information becomes available, perhaps the greater challenge is then processing, analyzing, and interpreting the data. To keep pace with this challenge, the development of new, fast, and scalable software solutions is required to visualize and interpret this information.

JavaScript is a lightweight programming language that uses a web browser as its host environment. JavaScript is cross-platform and supported by all modern browsers. Because JavaScript is client-side, it is very fast, as it doesn’t have to communicate with a server and wait for a response in order to run some code. Web browsers are ubiquitous and require no dependencies to deploy and operate, and so JavaScript represents an obvious solution for visualizing sequence information. Front-end developments using JavaScript have proven to be extremely efficient in providing fast, easy-to-use, and embeddable solutions for data analysis [6-14]. A very active community of developers at BioJS (http://www.biojs.io/) provide diverse components for parsing sequence data types, data visualization, and bioinformatics analysis in JavaScript [6, 7, 15-19].

Node.js provides server-side back-end JavaScript. Node.js is written in C, C++, and JavaScript and uses the Google Chrome V8 engine to offer a very fast cross-platform environment for developing server side Web applications. Node is a
single-threaded environment, which means that only one line of code will be
executed at any given time; however, Node employs non-blocking techniques for I/O
tasks to provide an asynchronous ability, by using callback functions to permit the
parallel running of code. Node holds much potential for the bioinformatic analysis of
molecular data. A community of Node developers provides modules for bioinformatic
sequence workflows (http://www.bionode.io/) which in time will likely parallel the
BioJS community for the number of modules versus components. However, as of
now there are no robust tools for phylogenetic analysis pipelines currently available
using the Node.js codebase. To fill this void I have developed, Phylo-Node, which
provides a Node.js toolkit that goes from sequence retrieval, to primer design, to
alignment, to phylogeny reconstruction, all from a single toolkit. MolPhylo is fast,
easy to use, and offers simple customization and portability options through various
inheritance patterns. The Node package manager, npm (https://www.npmjs.com/),
provides a very easy and efficient way to manage dependencies for any Node
application. Phylo-Node is available at GitHub (https://github.com/dohalloran/Phylo-
Node), npm (https://www.npmjs.com/package/phylo-node), and also BioJS
(http://www.biojs.io/d/phylo-node).

IMPLEMENTATION

Phylo-Node was developed using the Node.js codebase. The Phylo-Node core
contains methods for remote sequence retrieval, and phylogenetic analysis using a
suite of popular software tools. A base wrapper object is used to prepare the
arguments and directory prior to program execution. The base wrapper module is
contained within the ‘Wrapper_core’ directory. An individual software tool can be
easily accessed and executed by importing the module for that tool so as to get
access to the method properties on that object (Figure 1). These method properties are available to the user by using the ‘module.exports’ reference object. Inside a driver script file, the user can import the main module object properties and variables by using the ‘require’ keyword which is used to import a module in Node.js. The ‘require’ keyword is actually a global variable, and a script has access to its context because it is wrapped prior to execution inside the ‘runInThisContext’ function (for more details, refer to the Node.js source code: https://github.com/nodejs). Once imported, the return value is assigned to a variable which is used to access the various method properties on that object. For example: a method property on the ‘phyml’ object is ‘phyml.getphyml()’, which invokes the ‘getphyml’ method on the ‘phyml’ object to download and decompress the PhyML executable. For a complete list of all methods, refer to the ‘README’ file at the GitHub repository (https://github.com/dohalloran/Phylo-Node/blob/master/README.md). In order to correctly wrap and run each executable, new shells must be spawned so as to execute specific command formats for each executable. This was achieved by using ‘child.process.exec’, which will launch an external shell and execute the command inside that shell while buffering any output by the process. Binary files and executables were downloaded and executed in this manner and the appropriate file and syntax selected by determining the user’s operating system. Phylo-Node was validated on Microsoft Windows 7 Enterprise ver.6.1, MacOSX El Capitan ver.10.11.5, and Linux Ubuntu 64-bit ver.14.04 LTS.

RESULTS AND DISCUSSION

Phylo-Node is a toolkit to interface with key applications necessary in building a phylogenetic pipeline (Figure 2). Firstly, Phylo-Node allows the user to remotely
download sequences by building a unique URL and passing this string to the NCBI
e-utilities API (http://www.ncbi.nlm.nih.gov/books/NBK25501/). Any number of genes
can be supplied as command-line arguments to Phylo-Node by accessing the
`fetch_seq.fasta` method on the `fetch_seq` object in order to retrieve sequence
information in FASTA format. The module for remote sequence retrieval is contained
within the ‘Sequence’ directory. Phylo-Node also provides methods on specific
objects to download various executable files using the ‘download’ module. Any
binary can be downloaded using the base module ‘get_executable’ contained within
the ‘Download’ directory, however objects pertaining to specific tools such as PhyML
also contain methods for downloading and unpacking binaries (see README.md file
for details). Phylo-Node then provides modules to execute the following programs
from within the ‘./Tool/Run’ directory: Primer3 [20-22] to facilitate primer design;
Clustal Omega [23], K-align [24], and MUSCLE [25, 26] for multiple sequence
alignments; jModelTest2 [27] and ProtTest3 [28] to determine the best-fit model of
evolution, and PhyML [29, 30] for phylogeny reconstruction. The PhyML executable
is also employed by jModelTest2 and ProtTest3. Primer3 is the most popular
software for primer design, and takes a very lengthy list of input variables to optimize
primer selection. Clustal Omega, K-align, and MUSCLE are very fast and accurate
multiple sequence alignment tools that are commonly used to build robust DNA,
RNA, or protein alignments. PhyML is a popular program for building phylogenies
using maximum likelihood, and ProtTest3 determines the best-fit model of evolution
for protein sequences across 120 different potential models, while jModelTest2
determines best-fit models of nucleotide substitution from DNA sequence
alignments. Together, Phylo-Node provides a novel toolkit that allows the user to go
from raw sequence to phylogeny using Node.
Phylo-Node is highly scalable and customizable, and was inspired by projects such as BioPerl [31] which provides very diverse tools that include Perl modules for many bioinformatic tasks and also parsers and wrappers for diverse sequence formats and applications. BioPerl’s open source structure and architecture allows users to plug new modules into BioPerl pipelines to design new applications. Node.js implements prototypal inheritance as per JavaScript but also provides access to the ‘module.exports’ object which permits easy portability between the Phylo-Node toolkit and any other modules, and also interoperation between different languages by using the ‘child.process.exec’ process. Therefore, Phylo-Node can be integrated with existing Node.js bioinformatics tools [32, 33] or software written in other languages. For example, both jModelTest2 and ProtTest3 require a Java runtime environment (http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html), and by using ‘require’ to import each module, the user can execute the analysis of jModelTest2 and ProtTest3. The ‘prottest’ and ‘jmodeltest2’ modules and driver scripts (index.js) are contained within the respective ‘Prottest3’ and ‘jModelTest2’ directories and sample input is provided in the ‘COX2_PF0016’ sub-directory of the ‘Input_examples’ folder for ProtTest3 and the sample FASTA file aP6.fas (also contained within the ‘Input_examples’ folder) can be used for testing jModelTest2.

To further facilitate the ease of interoperation between various applications and components, the Phylo-Node package also contains a module called ‘phylo-node_pipes’ inside the ‘Pipes’ directory. The ‘phylo-node_pipes’ module allows the user to easily pipe data between different applications by requiring the ‘child_process’ module which provides the ability to spawn child processes. Through ‘phylo-node_pipes’, the user can chain commands together that will be executed in
sequence to build consistent, and extensive pipelines. The ‘Pipes’ directory contains sample driver scripts for using the ‘phylo-node_pipes’ module.

CONCLUSIONS

In conclusion, Phylo-Node is a novel package that leverages the speed of Node.js to provide a robust and efficient toolkit for researchers conducting molecular phylogenetics. Phylo-Node can be easily employed to develop complex but consistent workflows, and integrated with existing bioinformatics tools using the Node.js codebase.
AVAILABILITY AND REQUIREMENTS

- Project name: Phylo-Node
- Project home page: https://github.com/dohalloran/phylo-node
- Operating system(s): Platform independent
- Programming language: Node.js
- Other requirements: none
- License: MIT
- Any restrictions to use by non-academics: no restrictions or login requirements

ACKNOWLEDGMENTS

I thank members of the O’Halloran lab for critical reading of the manuscript, and would like to thank The George Washington University (GWU) Columbian College of Arts and Sciences, GWU Office of the Vice-President for Research, and the GWU Department of Biological Sciences for Funding.

AUTHOR CONTRIBUTIONS

D.O’H. conceived the idea for Phylo-Node, wrote and tested the code, and wrote the manuscript.

COMPETING INTERESTS

The author declares no competing interests.


FIGURE LEGENDS

Figure 1. Workflow for Phylo-Node.

Phylo-Node is organized into a workflow of connected modules and driver scripts. In order to interface with a phylogenetic tool, the base wrapper module is invoked to process command-line requests that are then passed into the software specific module. The input for the specific software can be passed into the base wrapper from: a) sample input directory; b) from a folder specified by the user; or c) by using the sequence retrieval module which is contained within the ‘Sequence’ directory. The ‘Pipes’ folder contains a module for easy piping of data between applications in Phylo-Node. Binaries can be downloaded using the ‘get_executable’ module from within the ‘Download’ folder.

Figure 2. Graphical overview of Phylo-Node applications within the ./Tools/Run directory.

Phylo-Node provides a toolkit for interacting with various applications including: the sequence alignment software Clustal Omega [23], K-align [24], and MUSCLE [25, 26]; the primer design software, Primer3 [20-22]; software for determining the best-fit models of evolution: jModelTest2 [27] and ProtTest3 [28]; and also the phylogeny reconstruction software, PhyML [29, 30]. Phylo-Node also enables the user to retrieve sequences remotely from the NCBI database using Entrez Programming Utilities. A key feature of Phylo-Node is interoperability between languages and other Node modules, which can be easily leveraged to form stable and scalable pipelines. This concept of interoperation and inheritance is highlighted by the brown cog at the bottom of Figure 2 that represents the potential to integrate any other module(s) [require('./module');] with Phylo-Node.
Figure 1

- ./Sequence: download sequences module
- ./Wrapper Core: base wrapper module
- ./Tools/Run: modules for running software tools
- ./Pipes: module for piping data between applications
- ./Input_examples: test data
- ./Download: download binaries module
Figure 2

```
require('./module');
```