Phylogenetics

The Newick utilities: high-throughput phylogenetic tree processing in the Unix shell

Thomas Junier\textsuperscript{1,2,*} and Evgeny M. Zdobnov\textsuperscript{1,2,3}

\textsuperscript{1}Department of Genetic Medicine and Development, University of Geneva Medical School, \textsuperscript{2}Swiss Institute of Bioinformatics, 1 rue Michel-Servet, 1211 Geneva, Switzerland and \textsuperscript{3}Imperial College London, South Kensington Campus, SW7 2AZ, London, UK

Associate Editor: Alex Bateman

\textbf{ABSTRACT}

Summary: We present a suite of Unix shell programs for processing any number of phylogenetic trees of any size. They perform frequently-used tree operations without requiring user interaction. They also allow tree drawing as scalable vector graphics (SVG), suitable for high-quality presentations and further editing, and as ASCII graphics for command-line inspection. As an example we include an implementation of bootscanning, a procedure for finding recombination breakpoints in viral genomes.

Availability: C source code, Python bindings and executables for various platforms are available from http://cegg.unige.ch/newick-utils. The distribution includes a manual and example data. The package is distributed under the BSD License.

Contact: thomas.junier@unige.ch

Received on March 3, 2010; revised on April 27, 2010; accepted on April 29, 2010

\section{INTRODUCTION}

Phylogenetic trees are a fundamental component of evolutionary biology, and methods for computing them are an active area of research. Once computed, a tree may be further processed in various ways (Table 1). Small datasets consisting of a few trees of moderate size can be processed with interactive GUI programs. As datasets grow, however, interactivity becomes a burden and a source of errors, and it becomes impractical to process large datasets of hundreds of trees and/or very large trees without automation.

Automation is facilitated if the programs that constitute an analysis pipeline is to make them capable of reading from standard input and writing to standard output—such programs are called \textit{filters}.

Although there are many automatable programs for computing trees [e.g. PhyML (Guindon and Gascuel, 2003), PHYLIP (Felsenstein, 1989)], programs for processing trees [e.g. TreeView (Page, 2002), iTOL (Letunic and Bork, 2007)] are typically interactive. Here, we present the Newick utilities, a set of automatable filters that implement the most frequent tree-processing operations.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Program & Function \\
\hline
nw_clade & Extracts clades (subtrees), specified by labels \\
nw_distance & Extracts branch lengths in various ways (from root, from parent, as matrix, etc.) \\
nw_display & Draws trees as ASCII or SVG (suitable for further editing for presentations or publications), several options \\
nw_match & Reports matches of a tree in a larger tree \\
nw_order & Orders tree nodes, without altering topology \\
nw_rename & Changes node labels \\
nw_reroot & Reroots trees on an outgroup, specified by labels \\
nw_trim & Trims a tree at a specified depth \\
nw_topology & Retains topological information \\
\hline
\end{tabular}
\caption{Selected Newick utilities programs and their functions}
\end{table}

2 RESULTS

The Newick utilities have the following features:

$\bullet$ no user interaction is required;

$\bullet$ input is read from a file or from standard input; output is written to standard output;

$\bullet$ all options are passed on the command line (no control files);

$\bullet$ options are self-documenting (option -h).

2.1 Example: Bootscanning

Bootscanning (Salminen, 1995) locates recombination breakpoints by identifying (locally) closest relatives of a reference sequence. An example implementation is as follows:

(1) produce a multiple alignment of all sequences, including the reference;

(2) divide the alignment into equidistant windows of constant size (e.g. 300 bp every 50 bp);

(3) compute a maximum-likelihood tree for each window;

(4) root the trees on the appropriate outgroup (not the reference);
A recombination breakpoint is predicted near position 450, as the nearest neighbor changes abruptly.

We wish to thank the members of the E.Z. group for feedback and beta testing.

The Newick utilities add tree-processing capabilities to a shell user's toolkit. Since they have no hard-coded limits, they can handle large amounts of data; since they are non-interactive, they are easy to automate into pipelines, and since they are filters, they can easily automate into pipelines, and since they are filters, they can easily and use more resources than their C equivalents. The difference is particularly apparent for large trees (Fig. 2).

3 DISCUSSION

The Newick utilities add tree-processing capabilities to a shell user’s toolkit. Since they have no hard-coded limits, they can handle large amounts of data; since they are non-interactive, they are easy to automate into pipelines, and since they are filters, they can easily work with other shell tools.

Tree processing may also be programmed using a specialized package [e.g., BioPerl (Stajich et al., 2002), APE (Paradis et al., 2004) or ETE (Huerta-Cepas et al., 2010)], but this implies knowledge of the package, and such programs tend to be slower and use more resources than their C equivalents. The difference is particularly apparent for large trees (Fig. 2).

3.1 Python bindings

To combine the advantages of a high-level, object-oriented language for the application logic with a C library for fast data manipulation, one can use the Newick utilities through Python’s ctypes module. This allows one to code a rerooting program in 25 lines of Python while retaining good performance (Fig. 2). A detailed example is included in the documentation.

Some users will feel more at ease working in the shell or with shell scripts, using existing bioinformatics tools; others will prefer to code their own tools in a scripting language. The Newick utilities are designed to meet the requirements of both.

ACKNOWLEDGEMENTS

We wish to thank the members of the E.Z. group for feedback and beta testing.

REFERENCES


