Bioinformatics Advance Access published May 13, 2010

The Newick Utilities: High-throughput Phylogenetic tree Processing in the UNIX Shell

Thomas Junier [1,2*] and Evgeny M. Zdobnov [1,2,3]

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

Associate Editor: Dr. Alex Bateman

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`

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 (see table 1). Small data sets consisting of a few trees of moderate size can be processed with interactive GUI programs. As data sets 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 can easily communicate data with each other. One way of doing this in the UNIX shell environment is to make them capable of reading from standard input and writing to standard output – such programs are called *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.

*to whom correspondence should be addressed

Program	Function
`nw_clade`	extracts clades (sub-trees), 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`	re-roots trees on an outgroup, specified by labels
`nw_trim`	trims a tree at a specified depth
`nw_topology`	retains topological information

Table 1. Selected Newick Utilities programs and their functions

RESULTS

The Newick Utilities have the following features:

- no user interaction is required
- input is read from a file or from standard input; output is written to standard output
- all options are passed on the command line (no control files)
- the input format is Newick (Archie *et al.*, 1986)
- the output is in plain text (Newick, ASCII graphics, or SVG)
- there are no limits to the number or size of the input trees.
- each program performs one function, with some variants
- the programs are self-documenting (option `-h`)

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)

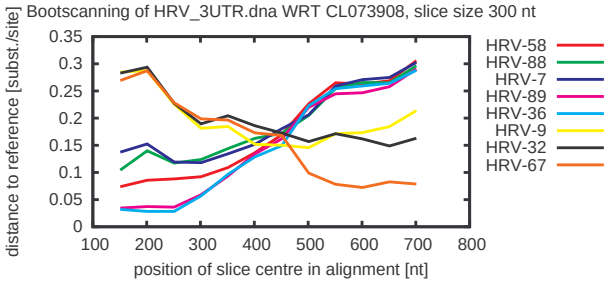

Fig. 1. Bootscanning using PhyML, EMBOSS, Muscle , Newick Utilities, GNUPlot and standard UNIX shell programs. The species with the lowest distance is the reference's nearest neighbor (by distance along tree branches). A recombination breakpoint is predicted near position 450, as the nearest neighbor changes abruptly.

3. compute a maximum-likelihood tree for each window
4. root the trees on the appropriate outgroup (not the reference)
5. from each tree, extract the distance (along the tree) from the reference to each of the other sequences
6. plot the result (Fig. 1)

The distribution includes a Bash script, `bootscan.sh`, that performs the procedure with Muscle (Edgar, 2004)(step 1), EMBOSS (Rice *et al.*, 2000)(step 2), PhyML (step 3), GNUPlot (step 6), and Newick Utilities for steps 4 and 5. This method was used to detect breakpoints in human enterovirus (Tapparel *et al.*, 2007).

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).

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.

FUNDING

This work was supported by INFECTIGEN and the Swiss National Science Foundation grant 3100A0-112588 to E.Z

ACKNOWLEDGEMENTS

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

REFERENCES

Archie, J., Day, W., Felsenstein, J., Maddison, W., Meacham, C., Rohlf, F., and Swofford, D. (1986). `http://evolution.genetics.washington.edu/phylip/newicktree.html`.

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Felsenstein, J. (1989). PHYLIP - Phylogeny Inference Package (version 3.2). *Cladistics*, **5**, 164–166.

Guindon, S. and Gascuel, O. (2003). A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, **52**, 696–704.

Huerta-Cepas, J., Dopazo, J., and Gabaldon, T. (2010). ETE: a python Environment for Tree Exploration. *BMC Bioinformatics*, **11**, 24.

Letunic, I. and Bork, P. (2007). Interactive Tree Of Life (iTOL): an online tool for phylogenetic tree display and annotation. *Bioinformatics*, **23**, 127–128.

Page, R. (2002). Visualizing phylogenetic trees using TreeView. *Curr Protoc Bioinformatics*, **Chapter 6**, Unit 6.2.

Paradis, E., Claude, J., and Strimmer, K. (2004). APE: Analyses of Phylogenetics and Evolution in R language. *Bioinformatics*, **20**, 289–290.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rice, P., Longden, I., and Bleasby, A. (2000). EMBOSS: the European Molecular Biology Open Software Suite. *Trends Genet.*, **16**, 276–277.

Salminen, M. (1995). Identification of breakpoints in intergenotypic recombinants of HIV type I by bootscanning. *AIDS Research and Human Retroviruses*, **11**, 1423–1425.

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G., Korf, I., Lapp, H., Lehvslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D., and Birney, E. (2002). The Bioperl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618.

Tapparel, C., Junier, T., Gerlach, D., Cordey, S., Van Belle, S., Perrin, L., Zdobnov, E., and Kaiser, L. (2007). New complete genome sequences of human rhinoviruses shed light on their phylogeny and genomic features. *BMC Genomics*, **8**, 224.

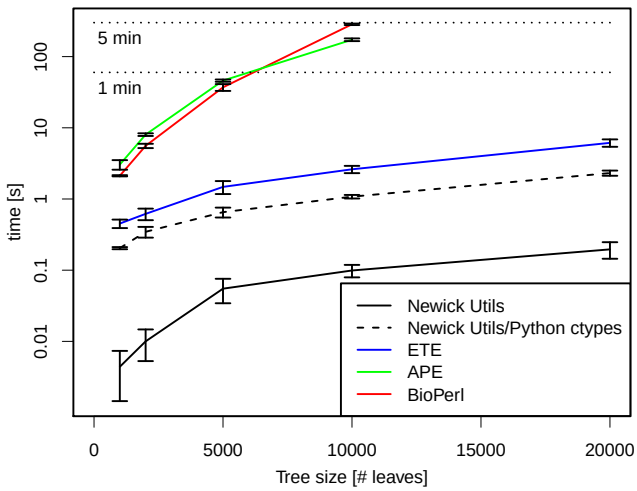

Fig. 2. Average run times (10 samples) of rerooting tasks on various tree sizes in different implementations. The task involved reading, rerooting, and printing out the tree as Newick. Runs of the BioPerl and APE implementation on the 20,000-leaf tree did not complete. Error bars show 1 standard deviation. Computer: 3-GHz 64-bit Intel Core 2 Duo, 1 Gb RAM, Linux 2.6. Made with R (R Development Core Team, 2008).