TreeRot.v3

Department of Biology, Boston University, Boston, MA 02215

Please send questions and suggestions to: msoren@bu.edu

Suggested citation:

Sorenson, M.D. & E.A. Franzosa. 2007. TreeRot, version 3. Boston University, Boston, MA.

Description:

TreeRot aids in the determination of decay or Bremer support indices (Bremer 1988) by generating a command file for PAUP or PAUP* (Swofford 1993, 1999). The command file includes **1)** a constraint statement for each node in a given shortest or strict consensus tree and **2)** commands to search for trees inconsistent with each of these constraint statements in turn. Compared to the shortest unconstrained tree, the number of additional steps required in the shortest tree(s) that is inconsistent with a given node is the decay index or Bremer support index for that node. For nodes with decay indices of more than 2 or 3, the constraint statement approach is much more effective than simply finding all trees 1, 2, 3, etc. steps longer than the shortest tree and then examining their strict consensus for which nodes are lost. Note also that a similar approach can be taken in analyses using other optimality criteria - see Lee & Hugall (2003) for an example involving maximum likelihood.

*****New Features, version 3*****

TreeRot.v3 has been completely recoded in Perl and should now run on any operating system!

TreeRot.v3 provides additional information about the results of partitioned analyses in which multiple equally parsimonious trees are found in a given constrained search. For each data partition, TreeRot now provides both the average difference in tree length between constrained and unconstrained searches and the minimum and maximum differences in tree length (for single partition analyses, all these values are the same). As shown by Lambkin et al. (2002), equally parsimonious trees found in a constrained search may require different numbers of extra steps for a given data partition; in some cases, one tree may require extra steps (suggesting support for that node in the given data partition), whereas another tree may require fewer steps (suggesting conflict with that node in the given data partition). Averaging across trees sweeps this potentially interesting conflicting signal under the rug. See Lambkin et al. (2002) for more on this issue.

TreeRot.v3 now summarizes your results in a NEXUS format tree file, in which the decay index for each node (and partition) is embedded in the tree file. This file can be viewed in FigTree (http://tree.bio.ed.ac.uk/software/) which will show the decay indices on each node. Values for different data partitions are shown on separate trees (one for each partition).

****New Features, version 2****

TreeRot.v2 includes an option to generate the commands needed to calculate "partitioned" Bremer support (Baker & DeSalle 1997, Baker et al. 1998). Partitioned Bremer support provides a measure of how different partitions of the data contribute to the decay index for each node in the context of the combined data analysis. The partial Bremer index for each data partition is determined by

subtracting the number of steps for that partition in the most parsimonious tree(s) from the number of steps for that partition in the shortest tree(s) without the node in question. Partial decay indices may be either positive or negative for an individual data partition, but the sum of the partial decay indices for a given node equals the overall decay index if the partitions specified are mutually exclusive but together comprise all of the characters in the original analysis.

TreeRot.v2 will also parse the PAUP log file and calculate decay indices or partial decay indices after all of the searches in the command file have been completed. Although this makes it very easy to obtain decay indices for your tree, I recommend a careful examination of the log file to evaluate whether the random addition replicates for each constrained search have consistently found trees of the same length. If not, additional replicates are likely to yield shorter trees, resulting in a smaller decay index.

Instructions:

1) In PAUP, find the shortest tree or trees for a given analysis. Save either the shortest tree or the strict consensus tree to a NEXUS tree file (e.g., duck.tree). This tree file must be saved in the same folder/directory as TreeRot. If more than one tree is saved TreeRot will use only the first tree. Note that by definition, nodes not in the strict consensus tree have a decay index of 0.

2) Double-click the TreeRot.sh icon (or if that doesn't work, open a command line window - the Terminal program in OSX - and type perl TreeRot.pl). Enter 1 to generate a PAUP command file. Follow the prompts to enter a) the name of the tree file (e.g., duck.tree), b) a name for the command file that TreeRot will generate (e.g., duck.constraints), and c) a name for the PAUP log file (e.g., duck.results). The latter is specified within the command file and will not be created until the command file is executed in PAUP.

3) If you want to generate partitioned Bremer support indices, answer yes (y) at the prompt and then follow additional prompts to enter the number of partitions and their names. Names of the partitions must correspond exactly to names of character sets in your PAUP data file. These should be mutually exclusive character sets that together comprise all the characters used in the original analysis.

4) Return to PAUP and execute the command file generated by TreeRot (e.g., duck.constraints). Results will be shown on the screen and also written to disk in a log file, the name of which was specified in TreeRot (e.g., duck.results). The relevant node will be written to the screen and to the log file after each search in the form of: Taxon1,Taxon2,>-----<Taxon3,Taxon4,Taxon5,Taxon6, etc. Subtract the number of steps in the shortest, unconstrained tree(s) from the number of steps in the shortest tree(s) found in each of the constrained searches to get your decay/support indices, or...

5) Launch TreeRot again and enter 2 to parse the PAUP log file. Follow the prompts to enter the name of the log file (e.g., duck.results) and the name of a file for the parsed results (e.g., duck.summary). Open this file in PAUP or a word processor. The length of the unconstrained analysis is given first followed by a list of taxa in each group and the length of the shortest trees without that group. The decay index for that node is shown to the right on the same line. For partitioned analyses, tree lengths and decay indices are broken down by data partition and the average, minimum, and maximum length difference for each partition over the shortest trees that were found is given (note that partitioned decay indices may be negative and/or fractional). TreeRot.v3 also writes to the same file in NEXUS format a tree that you can view in FigTree (http://tree.bio.ed.ac.uk/software/) to see decay indices on each node - this is substantially easier than going through the text output. It is, however, advisable to also examine the PAUP log file to confirm that the same island(s) of trees were consistently found across replicate runs - if not, more thorough searching of tree space may be necessary.

Getting TreeRot to run on your system:

The source code for the updated program is included in TreeRot.pl. The syntax is based on Perl 5.8. To run the program in script mode simply navigate to its location at the command line (using Terminal under OS X or the Command Prompt, cmd.exe, under Windows) and type "perl TreeRot.pl" (no quotes) and then press enter. Since OS X is based on Unix it comes with a version of Perl already installed. Windows does not include Perl by default, but a free version can be downloaded and installed from ActiveState (the distribution is called ActivePerl).

The file TreeRot.exe is an executable version of TreeRot compiled for the Windows platform. The executable contains all the TreeRot source code in addition to the necessary Perl components. If you choose to use the executable, you DO NOT need to install Perl separately on Windows. The executable can be run by double-clicking its icon (which will open and prepare a command prompt for you). TreeRot.exe was compiled using the free PAR (v0.973) and PAR::Packer (v0.975) Perl modules under ActiveState Perl 5.8.8 build 820.

The file TreeRot.sh is a compiled version of TreeRot in the form of a Mac OS X shell script. Double clicking the program icon in Mac OS X will launch a terminal window and run the program (just like the windows executable). The shell script is smaller and runs faster than the windows version because it can take advantage of the built-in Perl functionality of OS X. The attached version of TreeRot.sh was compiled on an intel-based mac running OS 10.4. There may be issues getting this version of the shell script to run on other versions of OS X or on power pc-based macs. To compile a version of TreeRot for your platform, download TreeRot.pl, navigate to its location in a Terminal window, and then type "perlcc -B -o TreeRot.sh TreeRot.pl" (no quotes) and then press enter. This will call OS X's built-in Perl compiler to make an executable version of TreeRot called TreeRot.sh. The *.sh file extension lets OS X know that this file wants to open and run in a terminal window.

Additional Notes:

1) Before executing the command file, make sure that all settings in PAUP are identical to those used in finding the original shortest unconstrained tree: e.g. inclusion/exclusion of taxa and characters, character weights, and step matrices. (Decay indices can be calculated for analyses using character weights and step matrices, although the values obtained will reflect these weighting schemes and should be interpreted appropriately.) The first search command in the command file generated by TreeRot is without constraints and should result in a tree(s) with the same number of steps as that found in the original search. If not, then some setting in PAUP has been changed or the shortest tree(s) is not being found.

2) By default, TreeRot specifies 20 replicate heuristic searches with random addition of taxa for each constraint statement. Note that for decay indices to be accurate, the shortest tree inconsistent with the constraint statement for each node must be found. Because failure to find the shortest tree(s) in each search results in an overestimation of decay indices, searching strategy is just as important as in the original unconstrained search. Depending on your dataset, you may want to edit the search statements in the command file generated by TreeRot. Additional replicates, exhaustive searches, branch and bound searches, and other search options can be implemented in these statements (see the PAUP manual). For example, you may want to limit the number of trees saved in each random addition replicate and increase the number of replicates to better explore tree space when working with large numbers of taxa. Be aware that additional options previously specified in PAUP dialog boxes or commands may be in effect at the time the command file is executed.

3) If you start additional searches on the same dataset after determining decay indices, be aware that the last constraint specified in the command file will still be in effect after execution of the

command file is completed.

4) With large datasets, you can execute the command file and leave it to run for hours (or days or weeks): a search will be conducted for each node. PAUP will sometimes stop, however, for a prompt (such as when MAXTREES has been reached). Before executing the command file, you may want to set MAXTREES to automatically increase or alternatively to "leave unchanged and don't prompt" (the effectiveness of the search may be reduced in the latter).

5) If searching for the shortest trees inconsistent with certain nodes is problematic because of multiple islands and/or large numbers of trees, you can input all the constraint statements by executing the command file and then stopping the first search. All of the constraint statements are then available in the dialog boxes for heuristic, branch and bound, or exhaustive searches, etc., allowing additional searching for shortest trees inconsistent with any particular node. Examine the command file to figure out which taxa belong to each node.

6) TreeRot works with NEXUS tree files generated by PAUP, either with or without a taxon translation table. TreeRot works with trees exported either as rooted or unrooted but with rooted trees, the last constraint statement may be a duplicate of the penultimate. TreeRot.v3 should no longer have limitations on number of taxa and length of taxon names, but let me know if you run into trouble.

7) No warranty is expressed or implied regarding the accuracy of your results or the compatibility of TreeRot with any other programs. Please let me know if you discover any bugs (msoren@bu.edu).

References:

Lambkin, C. L., M. S. Y. Lee, S. L. Winterton, and D. K. Yeates. 2002. Partitioned Bremer support and multiple trees. Cladistics-the International Journal of the Willi Hennig Society 18:436-444.

Lee, M. S. Y., and A. F. Hugall. 2003. Partitioned likelihood support and the evaluation of data set conflict. Systematic Biology 52:15-22.

Baker, R.H., and R. DeSalle. 1997. Multiple sources of character information and the phylogeny of Hawaiian Drosophilids. Systematic Biology 46:654-673.

Baker, R. H., X. B. Yu, and R. DeSalle. 1998. Assessing the relative contribution of molecular and morphological characters in simultaneous analysis trees. Molecular Phylogenetics and Evolution 9:427-436.

Bremer, K. 1988. The limits of amino acid sequence data in angiosperm phylogenetic reconstruction. Evolution 42:795-803.

Swofford, D.L. 1993. Phylogenetic analysis using parsimony (PAUP), version 3.1. Illinois Natural History Survey, Champaign, IL.

Swofford, D. L. 1999. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods), version 4.0. Sinauer Associates, Sunderland, Massachusetts.