Markov chain Monte Carlo (MCMC) inference of Bayesian posterior probabilities has become one of the primary methods used to infer phylogenies from sequence data. There are a number of reasons this method is appealing, including the ability to use informative priors when appropriate, the relatively intuitive interpretation of posterior probabilities compared to bootstrap and jackknife support values, and the perceived relative speed of MCMC over heuristic methods, among others. However, practical applications of MCMC in this arena suffer to some extent from a problem inherent to the method; it is frequently difficult to determine whether the chain has undergone enough iterations to sufficiently explore the space of possible solutions in order to accurately infer the posterior probability distribution of clades and model parameters.

Heuristic searches of the space of possible topologies and model parameters terminate when they reach a local optimum. While this does not guarantee that a globally optimal solution has been found, it does, at a minimum, guarantee that no better solution will be found using the same heuristic and starting tree. The goal of MCMC methods, in contrast, is integrate over the space of solutions to determine which topological features and parameter values are consistent across the set of reasonably good solutions. Frequently with real data, many of the possible solutions have very low posterior probability. When this is the case it is not necessary to explore the entire space in order to obtain accurate posterior probability estimates; rather an MCMC chain is said to have converged when further exploration of the space of possible solutions does not materially change the inferred posterior probabilities of clades and parameter values. Determining when this has occurred is one of the primary challenges of MCMC in phylogenetics. Failure to appropriately diagnose convergence can lead to inappropriate estimates of topologies and clade posterior probabilities. Indeed, it is possible that the perceived speed advantage of MCMC may in some part be due to the prevalence of inappropriately short chains and a failure to diagnose a lack of convergence.

There are attributes of the phylogeny problem that make achieving and determining convergence (or non-convergence) exceptionally difficult. Phylogenetic problems often involve a large number of interacting continuous model parameters, but typically the posterior probabilities for these parameters are relatively smooth. The space of possible topologies (“tree space”), on the other hand, often involves an astronomical number of discrete (as opposed to continuous) possible solutions, with adjacent solutions potentially differing considerably in their posterior probability. As such, tree space may contain numerous local optima, which can cause the MCMC chain to become stuck and fail to appropriately explore the space of possible solutions. For this reason, MCMC in phylogenetics typically employs a number of additional “heated” chains in order to allow the chain to extricate itself from local optima. These heated chains are effectively exploring a somewhat flattened version of tree space, with the potential for the primary chain to swap to one of those heated chains.

Initially, MCMC convergence in phylogenetics was typically diagnosed using plots of log likelihood as a function of chain length; users would examine the plot and eliminate topologies explored before the point in the chain at which likelihoods level out (the “burn-in”). While a converged chain will typically have a relatively flat distribution of likelihoods, this is a necessary, but not sufficient, condition for diagnosing chain convergence. A chain that is stuck on a single local optimum will also produce a relatively flat likelihood profile, while potentially being far from convergence. Similarly, a chain that is exploring multiple local optima with similar likelihoods may produce a flat likelihood profile without converging in posterior probability estimates.

A somewhat less obvious problem can occur even when there is a single optimum in tree space: for accurate estimates of clade posterior probabilities, it is not always sufficient to have found the optimum solution. If solutions adjacent to the optimal solution are only slightly lower in likelihood, accurate posterior estimates can only be obtained by sufficiently exploring those suboptimal solutions. The number, or proportion, of suboptimal solutions that must be explored to obtain accurate posterior probability estimates is unknown, and necessarily varies from case to case. However, the number of topologies adjacent to the optimum increases rapidly with the number of taxa, with the exact increase depending on the proposal scheme. It is therefore quite likely that the number of trees that must be explored in order to accurately estimate posterior probabilities also increases with the number of taxa even if the proportion of nearby trees that must be explored remains constant. Given the similarity in likelihoods of near-optimal solutions, likelihood profiles are necessarily uninformative about the number of near-optimal trees necessary to achieve convergence in posterior probabilities even in the single-optimum case.

In order to deal with this issue, we previously developed novel convergence diagnostics based on directly examining the posterior probabilities of clades as a function of chain length (e.g., cumulative and sliding window posterior probability plots, Nylander et al. 2007). These diagnostics offer significant performance improvements over likelihood profiles: they help detect when multiple optima are being explored, and can help estimate the number of trees adjacent to a single optimum that are necessary to achieve accurate posterior estimates. While these diagnostics will not necessarily detect chains that are stuck on local optima on their own, other multi-chain diagnostics included in the AWTY software package (“compare” plots, Nylander et al. 2007) potentially can detect these issues when multiple chains have been run. It must be noted that none of these diagnostic plots is sufficient to positively diagnose convergence, but at minimum they represent a much stricter set of necessary conditions than likelihood plots.

**RWTY: R We There Yet?**

RWTY is a new package for the R statistical programming environment that implements many of the features of AWTY (Nylander et al. 2007) as well as some new functionality. RWTY takes output from popular MCMC packages (e.g., MrBayes, BEAST), and produces AWTY-style plots for cumulative posterior probability, sliding window posterior probability, multi-chain comparison, and likelihood profiles. RWTY also allows visualization of MCMC exploration of tree space in a manner similar to TreeSetViz (Amenta and Klingner 2002). Finally, RWTY allows users to determine effective sample size from an MCMC chain for topology for the first time.

At present, RWTY produces five types of plots: cumulative posterior probability, sliding window posterior probability, likelihood, chain comparison, and non-metric multidimensional scaling (NMDS) chain plots. We discuss each of these plot types in turn below. Plots can be created directly using built-in functions, but typical users will interact with RWTY using only two functions: load.trees and analyze.rwty. The analyze.rwty function is intended to automatically determine which plots are possible produce with the provided data, and automatically produce them all.

*Core functionality*

**Load trees (function load.trees):** This function loads trees in to an rwty.trees object. This object contains trees, an associated table of likelihoods (when a .p file is available), and information about the number of generations separating each tree in the chain. The function will attempt to detect the .p file and the number of generations per tree automatically if these arguments are not provided, but this information can also be provided manually. The “type” argument will cause RWTY to load trees using APE’s (CITE) “read.nexus” function if the value is set to “nexus”, but will load trees using the “read.trees” function otherwise.

chain1 <- load.trees(filename, type="nexus", gens.per.tree=1000)

**Main function (function analyze.rwty):** This omnibus function calls all of the analysis and plotting functions of RWTY that are available, given the data provided to it. When provided with a single chain, it produces cumulative and sliding window plots, as well as an NMDS chain plot. Additionally, it searches for a .p file associated with the .t file for the chain, and if one is found it produces a likelihood plot from that. When provided with multiple chains, it produces cumulative, sliding window, and treespace plots for each chain (and a likelihood plot if .p files are available), as well as a chain comparison plot for all pairs of chains.

The “chains” argument should be passed either a single rwty.trees object or a list of rwty.trees objects. The burnin argument tells RWTY how many trees to eliminate from the beginning of each chain prior to analysis, and the window.size argument sets the length (in number of trees) of the sliding window for the sliding window plot. The “step” argument tells RWTY how often to sample the chain for making NMDS treespace plots, as these analyses can be quite time consuming. A step value of ten, for instance, only plots every tenth tree. The labels argument exists so that chains can be named, and these names will be used as labels on plots. Finally, if a filename is provided, all plots will automatically be printed to a pdf by that name in the working directory. In the multi-chain case, one file will be produced for each chain using the arguments passed to “labels” concatenated with the argument passed to “filename” (e.g., “Run1 multitest.pdf” in the example below). Similarly, the “compare chains” plot will be output to “compare multitest.pdf”.

analyze.rwty(chains=list(chain1, chain2), burnin=100, window.size=100, step=10, labels=c("Run1", "Run2"), filename="multitest.pdf")

The return value for a single chain is an object containing a number of data tables that can be used for further analysis and visualization, as well as the plots that RWTY has produced from that data. The return value when multiple chains have been provided is a list of these single-chain objects (one for each chain provided), as well as a separate compare.n object containing data and plots for multi-chain comparisons.

1. Introduction
   1. MCMC
   2. Visualizing convergence problems
      1. LnL plots, why they’re not informative
         1. In space with multiple optima, LnLs may look like convergence when in fact you’re stuck on a local optimum
         2. Even if you have a single optimum, accurate posteriors require getting a decent sampling of the set of all plausible trees. More taxa = more trees adjacent to the optimum, which means more samples needed to get the same proportion of trees adjacent to the optimum. Ergo convergence can take a while even in simple treespaces with a large number of taxa.
      2. Posterior plots
         1. Cumulative
            1. Actually visualize the quantity of interest, i.e., topology.
            2. May take significantly longer to level off, but that is a GOOD thing.
            3. Sample plot of LnL vs. cumulative?
         2. Sliding window
            1. Useful for visualizing whether the chain is mixing well
            2. Cool plot
      3. TreeSetViz style plotting
         1. Collapses treespace to two dimensions, looks at chain behavior over time
         2. Useful for visualizing burnin, exploration of treespace, and differences between chains
         3. Cool plot
      4. Compare N plots
         1. Compare posteriors for clades from two or more runs
         2. Useful for visualizing agreement/disagreement between chains
         3. When applied to multiple chains on the same data, useful for highlighting problems with convergences
         4. When applied to chains from different data, useful for highlighting differences in the support for different clades
         5. Cool plot
2. RWTY
   1. Data types
      1. Trees, .p files detected automatically when available. Basically anything ape can handle.
   2. Interface
      1. Primary user interface is through function analyse.rwty
         1. Single chain with no p file outputs cumulative, sliding window, and TreeSetViz
            1. If a p file is found, LnL plot is returned as well
         2. If multiple chains are sent in a list, analyse.rwty automatically does Compare N analysis and multiple chain TreeSetViz
   3. Return value
      1. RWTY object containing tables, plots, and summary statistics