Running Coalescent Analyses With coalescentMCMC

Emmanuel Paradis

October 6, 2012

Coalescent analyses have emerged in the recent years as a powerful approach to investigate the demography of populations using genetic data. The coalescent is a random process describing the coalescent times of a genealogy with respect to population size and mutation rate. In the majority of cases, the genealogy of individuals within a population is unknown. So a coalescent analysis typically consider integrating over the "likely" genealogies to make inference on the dynamics of the population. This uses computer-intensive methods such as Monte Carlo simulations of Markov chains. Besides, if priors are defined on the distributions of the parameters, Bayesian inference can be done. Several methods have been proposed for such integrations, although currently there is no consensus on which method is the best or which ones are the most appropriate in some circumstances [1].

coalescentMCMC aims to provide a general framework to run coalescent analyses. In its current (and early) version, the package provides only a simple MCMC algorithm based on Hastings's ratio.

coalescentMCMC has three main groups of functions that have different roles:

- the function `coalescentMCMC` itself which runs the chain;
- some functions doing operations on tree which are called by the previous one to move from one tree to another;
- some functions to infer demography from genealogies under various coalescent models which are typically used to analyse the output of a chain run.

The motivating idea behind coalescentMCMC is that the user can have full control over the analysis. The options of the main function are:

```
coalescentMCMC(x, ntrees = 10000, burnin = ntrees,
               ini.tree = NULL, quiet = FALSE)
```

where `ntrees` are the number of (accepted) trees to output, `burnin` is the number of trees discarded before output starts, and `ini.tree` is the initial tree (if not provided, a UPGMA tree from a JC69-based distance matrix is used). The proposed trees are not output, though this may be easily modified. The code of the function is relatively simple:

```
> library(coalescentMCMC)
> body(coalescentMCMC)
```

```
{
    if (is.null(tree0)) {
        d <- dist.dna(x, "JC69")
        X <- phangorn::phyDat(x)
        tree0 <- as.phylo(hclust(d, "average"))
    }
    n <- Ntip(tree0)
    nodeMax <- 2 * n - 1
    TREES <- vector("list", ntrees)
    LL <- numeric(ntrees)
    lnL0 <- phangorn::pml(tree0, X)$logLik
    i <- j <- 0L
    if (!quiet) {
        cat("Running the Markov chain:\n")
        cat("Nb of proposed trees    Nb of accepted trees\n")
    }
    while (i <= ntrees) {
        if (!quiet)
            cat("\r     ", j, "                   ", i, "            ")
        j <- j + 1L
        tr.b <- NeighborhoodRearrangement(tree0, n, nodeMax)
        lnL.b <- phangorn::pml(tr.b, X)$logLik
        ACCEPT <- if (is.na(lnL.b))
            FALSE
        else {
            if (lnL.b >= lnL0)
                TRUE
            else rbinom(1, 1, exp(lnL.b - lnL0))
        }
        if (ACCEPT) {
            lnL0 <- lnL.b
            tree0 <- tr.b
            if (j > burnin) {
                i <- i + 1L
                LL[i] <- lnL0
                TREES[[i]] <- tree0
            }
        }
    }
    class(TREES) <- "multiPhylo"
    if (!quiet)
        cat("\nDone.\n")
    list(trees = TREES, logLik = LL)
}
```

The accepted trees are stored in the list `TREES`; it appears that all proposed trees `tr.b` can also be stored and output with minor modifications of the code. The coding of the Hastings's ratio is clear (`ACCEPT`) and this part can also be tailored at will.

The above implementation uses only neighborhood rearrangement as proposed in [2] calling the function `NeighborhoodRearrangement` at each cycle of the chain. This can modified by using other functions described in `?treeOperators`.

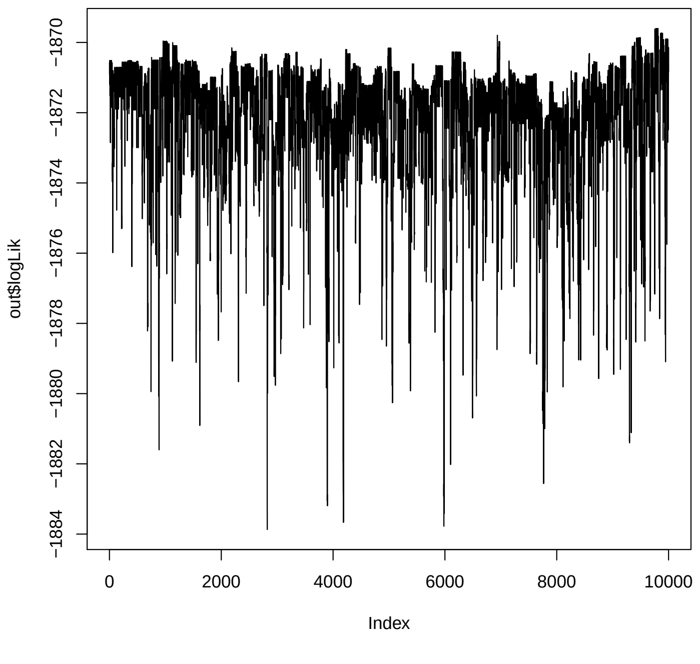
Let us now consider a very simple analysis with the woodmouse data available in `ape`. After loading the data, we run an MCMC with the default settings, thus outputing 10,000 trees after a burn-in period of 10,000 steps:

```
> data(woodmouse)
> out <- coalescentMCMC(woodmouse)

Running the Markov chain:
Nb of proposed trees    Nb of accepted trees
      67207                        10000
Done.
```

A total of 67,207 trees have been proposed, so among them 57,207 have not been accepted (including the 10,000 that were initially accepted but were discarded because of the burn-in period). We can examine how the log-likelihood of the accepted trees evolved along the chain:

```
> plot(out$logLik, type = "l")
```



The log-likelihood was relatively stable between −1872 and −1871. For the sake of simplicity, we consider only the last 1000 trees from the chain for further analysis. Extracting them is easy with standard R operators:

```
> s <- 9001:1e4
> o <- list(trees = out$trees[s], logLik = out$logLik[s])
```

The list `o` is now a subset of the full output `out`. On each of those extracted trees, we wish to estimate Θ. This can be done with the function `theta.tree` in pegas. The help page of this function tells us that it returns a list with named elements:

```
> library(pegas)
> theta.tree(o$trees[[1]], .01)
```

```
$theta
[1] 0.02127424

$se
[1] 0.00568578

$logLik
[1] 83.29005
```

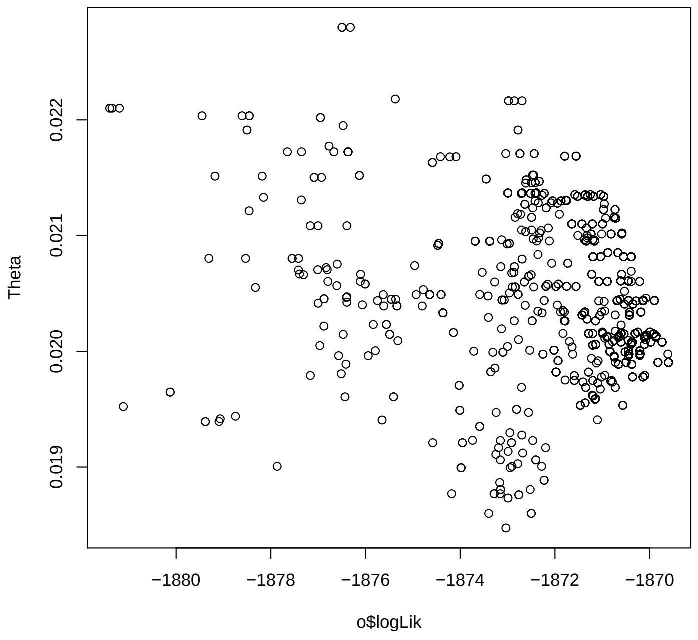
So we build a function "on-the-fly" to return only the value we want. Again, we use a standard R function, here `sapply`, to perform the operation in simple way:

```
> Theta <- sapply(o$trees, function(x) theta.tree(x, .01)$theta)
> head(Theta)
```

```
[1] 0.02127424 0.02102289 0.02103378 0.02103378 0.02134836 0.02129919
```

We can plot the estimated values against the log-likelihood of each tree:

```
> plot(o$logLik, Theta)
```



Finally, we do a global estimation of Θ with the mean weighted by the tree log-likelihoods:

```
> weighted.mean(Theta, o$logLik)
```

```
[1] 0.02046285
```

The above analysis is rather simplistic: a full coalescent analysis would be much more complicated. However, it aims to show here the potential of using R for such analyses.

References

[1] J.~Felsenstein. Trees of genes in populations. In O.~Gascuel and M.~Steel, editors, *Reconstructing Evolution: New Mathematical and Computational Advances*, pages 3–29. Oxford University Press, Oxford, 2007.

[2] M.~K. Kuhner, J.~Yamato, and J.~Felsenstein. Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. *Genetics*, 140:1421–1430, 1995.