Running Coalescent Analyses With coalescentMCMC

Emmanuel Paradis

October 27, 2013

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 = 3000, burnin = 1000, frequency = 1,
               tree0 = NULL, model = NULL, quiet = FALSE)
```

where `ntrees` are the number of trees to output, `burnin` is the number of trees discarded before output (of trees) starts, `frequency` is the sampling frequency along the Markov chain, and `tree0` is the initial tree (if not provided, a UPGMA tree from a JC69-based distance matrix is used). `model` is either `NULL` is which case Θ is assumed to be constant, or `"time"` in which case a model where Θ

follows an exponential growth is used.[1] Finally, `quiet` is a logical parameter controlling whether to print an indication of the progress of the chain.

The code of the function is relatively simple:

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

{
    if (is.null(tree0)) {
        d <- dist.dna(x, "JC69")
        tree0 <- as.phylo(hclust(d, "average"))
    }
    X <- phyDat(x)
    n <- length(tree0$tip.label)
    nodeMax <- 2 * n - 1
    nOut <- ntrees
    nOut2 <- ntrees * frequency + burnin
    getlogLik <- function(phy, X) pml(phy, X)$logLik
    TREES <- vector("list", nOut)
    LL <- numeric(nOut2)
    TREES[[1L]] <- tree0
    lnL0 <- getlogLik(tree0, X)
    LL[1L] <- lnL0
    if (is.null(model)) {
        np <- 1L
        para.nms <- "theta"
        two2n <- 2:n
        K4theta <- length(two2n)
        tmp <- choose(two2n, 2)
        getparams <- function(phy, bt) {
            x4theta <- rev(diff(c(0, sort(bt))))
            sum(x4theta * tmp)/K4theta
        }
        f.theta <- function(t, p) p
    }
    else {
        np <- 2L
        para.nms <- c("theta0", "rho")
        getparams <- function(phy, bt) {
            out <- nlminb(c(0.02, 0), function(p) -dcoal.time(phy,
                p[1], p[2], log = TRUE))
            out$par
        }
        f.theta <- function(t, p) p[1] * exp(p[2] * t)
    }
    params <- matrix(0, nOut2, np)
    i <- 2L
    j <- 0L
    k <- 0L
```

[1] Other models will be implemented later. See the vignette "CoalescentModels".

```
if (!quiet) {
    cat("Running the Markov chain:\n")
    cat("  Number of trees to output:", ntrees, "\n")
    cat("  Burn-in period:", burnin, "\n")
    cat("  Sampling frequency:", frequency, "\n")
    cat("  Number of generations to run:", ntrees * frequency +
        burnin, "\n")
    cat("Generation    Nb of accepted trees\n")
}
bt0 <- branching.times(tree0)
params[1L, ] <- para0 <- getparams(tree0, bt0)
nodesToSample <- (n + 2):nodeMax
while (k < nOut) {
    if (!quiet)
        cat("\r  ", i, "                ", j, "           ")
    target <- sample(nodesToSample, 1L)
    THETA <- f.theta(bt0[target - n], para0)
    tr.b <- NeighborhoodRearrangement(tree0, n, nodeMax,
        target, THETA, bt0)
    if (!(i%%frequency) && i > burnin) {
        k <- k + 1L
        TREES[[k]] <- tr.b
    }
    lnL.b <- getlogLik(tr.b, X)
    LL[i] <- lnL.b
    bt <- branching.times(tr.b)
    params[i, ] <- para <- getparams(tr.b, bt)
    i <- i + 1L
    ACCEPT <- if (is.na(lnL.b))
        FALSE
    else {
        if (lnL.b >= lnL0)
            TRUE
        else rbinom(1, 1, exp(lnL.b - lnL0))
    }
    if (ACCEPT) {
        j <- j + 1L
        lnL0 <- lnL.b
        tree0 <- tr.b
        para0 <- para
        bt0 <- bt
    }
}
LL <- cbind(LL, params)
colnames(LL) <- c("logLik", para.nms)
LL <- mcmc(LL, start = 1, end = i - 1)
attr(TREES, "TipLabel") <- TREES[[1L]]$tip.label
for (i in seq_len(nOut)) TREES[[i]]$tip.label <- NULL
class(TREES) <- "multiPhylo"
j <- 1
```

```
    list.trees <- ls(envir = .coalescentMCMCenv)
    if (l <- length(list.trees))
        j <- 1 + as.numeric(sub("TREES_", "", list.trees[l]))
    assign(paste("TREES", j, sep = "_"), TREES, envir = .coalescentMCMCenv)
    if (!quiet)
        cat("\nDone.\n")
    LL
}
```

The trees are stored in the list `TREES_xxx` with xxx being 1, 2, ..., for the successive chains run during a session (see below). The coding of the Hastings's ratio is clear (`ACCEPT`) and this part can also be tailored at will.

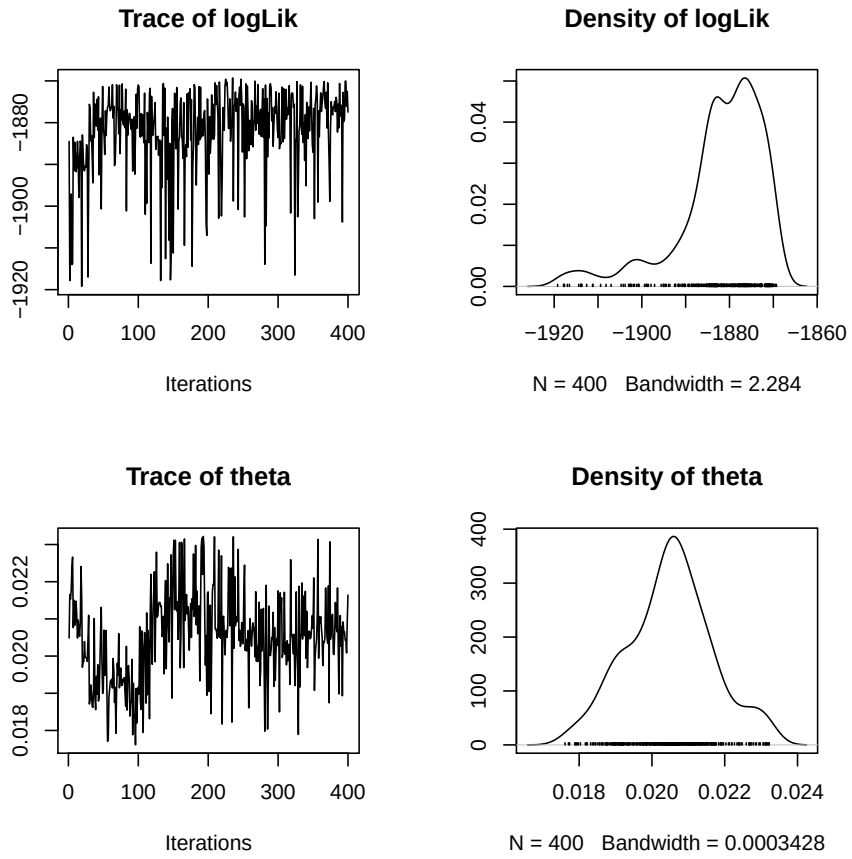
The above implementation uses only neighborhood rearrangement as proposed in [2] calling the function `NeighborhoodRearrangement` at each cycle of the chain. This can modified by using other functions described in `?treeOperators`.

Let us now consider a very simple analysis with the woodmouse data available in `ape`. For the purpose of this vignette, we run a very light analysis in order to produce small outputs in a reasonable time (it seems that in practice the above default values are a minimum).

```
> data(woodmouse)
> out <- coalescentMCMC(woodmouse, ntrees = 300, burnin = 100)

Running the Markov chain:
  Number of trees to output: 300
  Burn-in period: 100
  Sampling frequency: 1
  Number of generations to run: 400
Generation    Nb of accepted trees
   400                  62
Done.
```

The output object is of class `"coda"`, so we can visualise it with the package of the same name (which has already been loaded):

```
> plot(out)
```



The log-likelihood was relatively stable between −1870 and −1880. The trees are stored in a special place of the memory (an environment in R's jargon) from where they can be retrieved with a specific function:

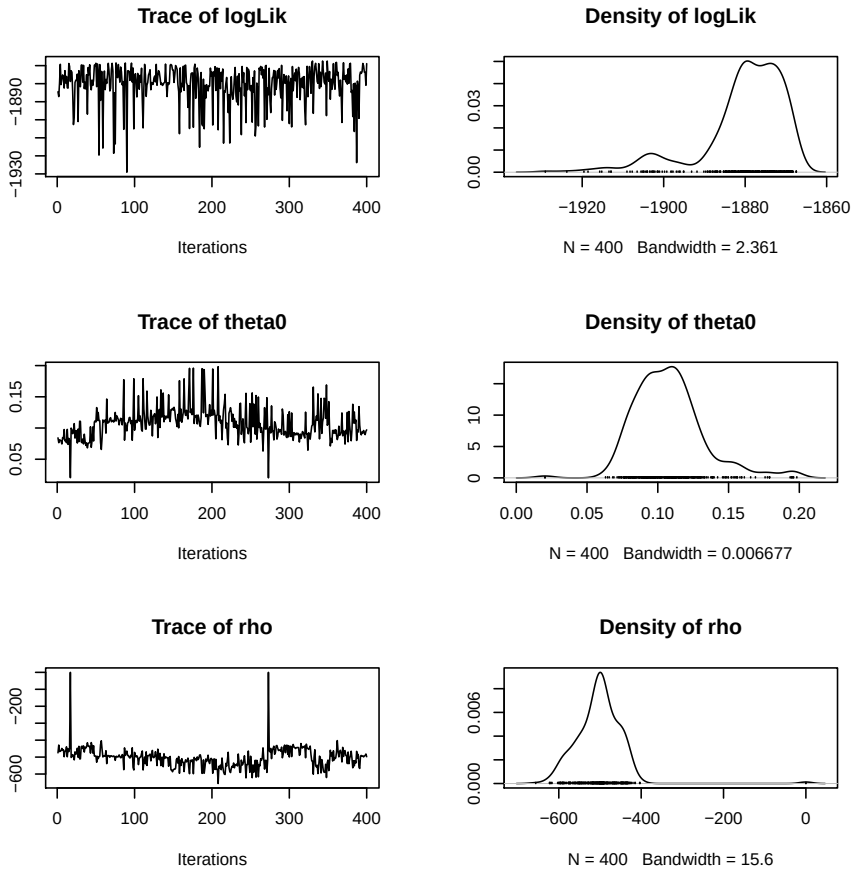
```
> TR <- getMCMCtrees()
> TR

300 phylogenetic trees
```

Note that the trees generated during the burn-in period are not output, but the corresponding values of log-likelihood and Θ are. Hence `out` has 400 rows.

```
> dim(out)

[1] 400   2

> colnames(out)

[1] "logLik" "theta"
```

We now run a model of time-dependent coalescent where Θ follows an exponential change through time:

```
> out2 <- coalescentMCMC(woodmouse, ntrees = 300, burnin = 100, model = "time")

Running the Markov chain:
  Number of trees to output: 300
  Burn-in period: 100
  Sampling frequency: 1
  Number of generations to run: 400
Generation    Nb of accepted trees
   400                  54
Done.

> plot(out2)
```



The change in log-likelihood along the chain is similar to what was observed above. The object `out2` has now three columns:

```
> dim(out2)

[1] 400   3

> colnames(out2)

[1] "logLik" "theta0" "rho"
```

If we try to extract the trees as previously done and R is running in interactive mode, we will be asked which list of trees to extract:

```
> getMCMCtrees()
Several lists of MCMC trees are stored:

1 : TREES_1
2 : TREES_2

Return which number?
```

Because the code of this vignette is not run interactively, we must extract the list explicitly with `get`:

```
> TR2 <- get("TREES_2", envir = .coalescentMCMCenv)
```

We can now compare both coalescent models: the two hypotheses under consideration are:

- H_0: Θ is constant;
- H_1: Θ changes through time following an exponential model.

We need to calculate the likelihood under both hypotheses. This can be done with functions provided in coalescentMCMC (see `?dcoal`). We use the last 100 trees of each chain.[2] Because we are using a list of trees (which is a vector) and also a vector of estimates of Θ, the function to use here is `mapply`. For clarity, we extract the trees and the values of $\hat{\Theta}$ that we need:

```
> tr <- TR[-(1:200)]
> THETA <- out[-(1:300), 2]
> logLik0 <- mapply(dcoal, phy = tr, theta = THETA, log = TRUE)
> summary(logLik0)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  82.11   83.44   83.79   83.76   84.06   85.71
```

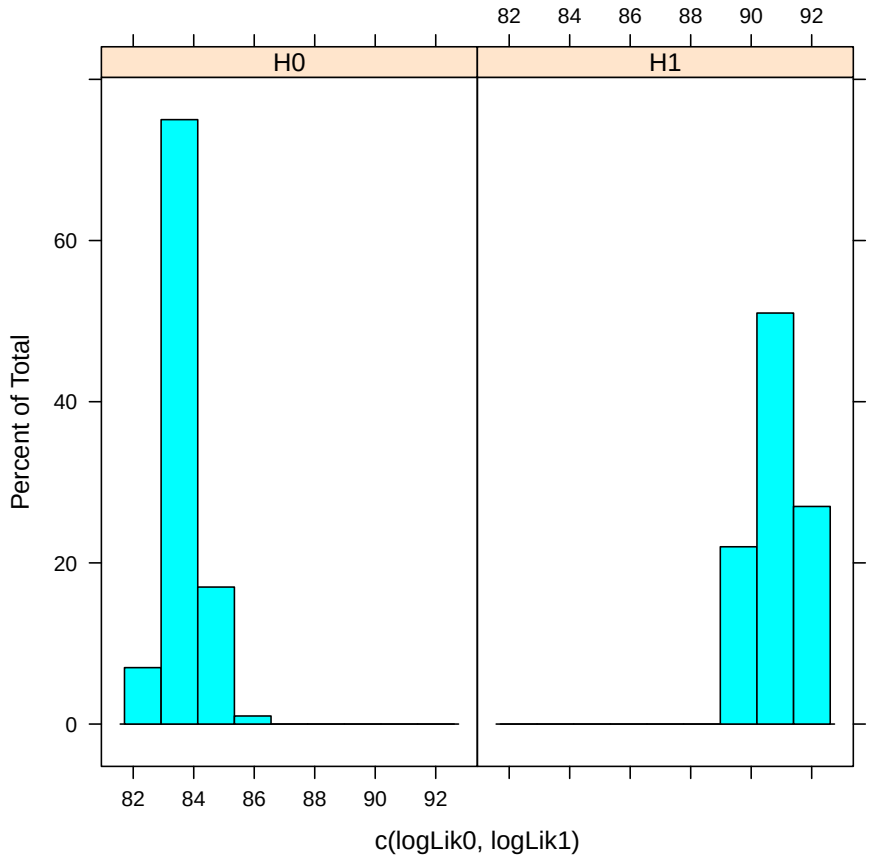
We can now repeat this operation for the second model:

```
> tr2 <- TR2[-(1:200)]
> THETA0 <- out2[-(1:300), 2]
> RHO <- out2[-(1:300), 3]
> logLik1 <- mapply(dcoal.time, phy = tr2, theta = THETA0, rho = RHO, log = TRUE)
> summary(logLik1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  89.29   90.30   91.06   90.92   91.43   92.21
```

A conditional histogram shows the two distributions:

[2] These calculations are quite fast, even with 1000 trees, but we use here a subset of the trees to illustrate how we select some of them which might be useful when running a longer Markov chain.

```
> print(histogram(~c(logLik0, logLik1) | gl(2, 100, labels = c("H0", "H1"))))
```



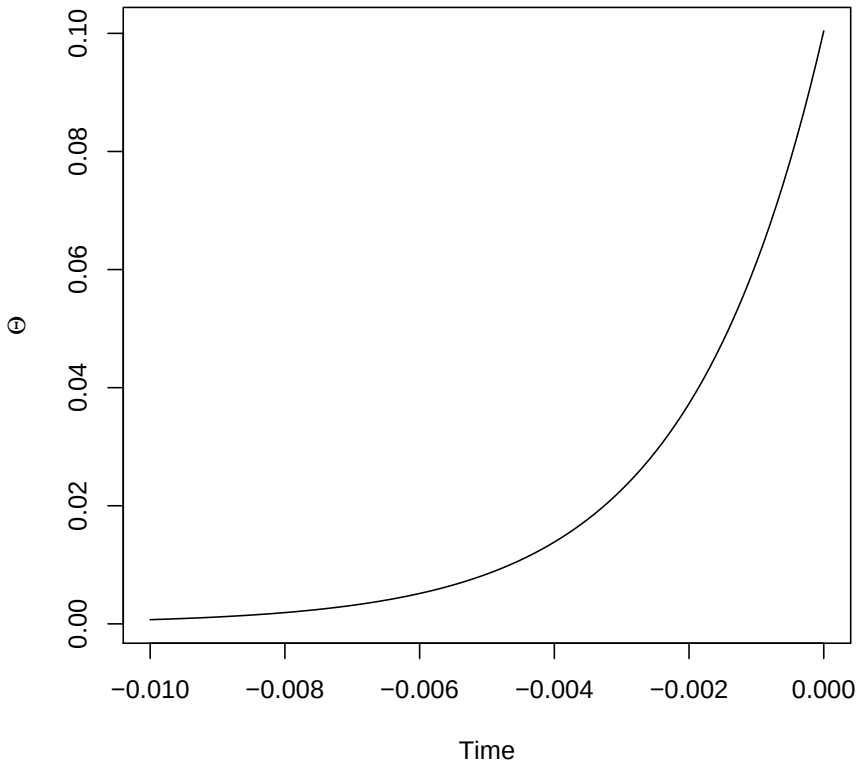
Since the increase in log-likelihood for the second model is about 8, the LRT comparing both models is $\chi^2_1 \approx 16$ which is highly significant ($P \approx 6 \times 10^{-5}$). This suggests that the population of woodmice (*Apodemus sylvaticus*) from where these sequences have been sampled has expanded (reminding that in coalescent models the time scale is reversed so a negative value of ρ means that the population has expanded).

Finally, we produce estimates of the parameters weighted by the likelihoods of the trees:

```
> treeloglik <- out2[-(1:300), 1]
> (theta0ML <- weighted.mean(THETA0, treeloglik))

[1] 0.1004052

> (rhoML <- weighted.mean(RHO, treeloglik))

[1] -495.206
```

We can represent the temporal variation in Θ predicted by this model (remember that the time scale is the one of molecular change):

```
> x <- seq(0, 0.01, 0.0001)
> y <- thetaOML * exp(rhoML * x)
> plot(-x, y, "l", xlab = "Time", ylab = expression(Theta))
```



Other things that could be done with simple R commands include:

- Compute confidence intervals around $\hat{\Theta}_0$ and $\hat{\rho}$ (alternatively, posterior distributions of these parameters if a Bayesian sampling is done);
- Re-run the chain(s) with different initial trees, for instance to run branching chains taking a tree from `TR` or `TR2`.

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.