An rbrothers tutorial

Jan Irvahn and Vladimir N. Minin
Department of Statistics, University of Washington Seattle, WA, 98195, USA

This is a tutorial demonstrating the usage of the java program, DualBrothers [2], in R. To get started, install the rbrothers R package from R-forge.

```
> install.packages("rbrothers", repos="http://R-Forge.R-project.org")
```

The rJava package and a couple of other packages will need to be installed.

Start R and load the rbrothers library.

```
> library(rbrothers)
```

We walk through the rbrothers workflow using four strains of HIV-1 isolate KAL153 from Kaliningrad, Russia [1]. Copy the KAL153.phy file from the rbrothers package to your current working directory.

```
> my.align = read.dna(file = system.file("extdata/KAL153/KAL153.phy",
     package = "rbrothers"))
> write.dna(my.align, "KAL153.phy")
```

Now you can run DualBrothers with a single command. The 'seed' argument is any integer used to seed the package's pseudo-random number generator. The 'alignment' argument is the name of the input file (less the '.phy' extension). The 'format' argument tells rbrothers the format of the input file, either 'interleaved' or 'fasta'. The 'par_lambda' argument sets the prior mean number of substitution process change-points. The 'top_lambda' argument sets the prior mean number of topology breakpoints. You can find more information about the details of the DualBrothers program at `http://faculty.biomath.ucla.edu/msuchard/htdocs/DualBrothers/`.

```
> db <- dualbrothers(seed = 123, alignment = "KAL153",
     format = "interleaved", par_lambda = 5, top_lambda = 0.693)
```

When your alignment has more than six tips rbrothers constructs a set of candidate trees. You need to provide a value for the 'window.size' argument in this situation as mentioned in the help file: `help(dualbrothers)`. If the set of candidate trees is too small you have the option to bootstrap additional candidate trees via the 'boot' argument.

If you already have the output files of a DualBrothers run you can read the information in directly.

```
> db <- readdb("KAL153")
```

It is a good idea to check the convergence of the MCMC chain; if the chain has not converged it should be rerun for a longer amount of time. The 'length' argument of the dualbrothers command allows the user to specify how long the MCMC chain should run if stationarity is not initially achieved. A trace plot of the log likelihoods of the MCMC chain and an autocorrelation plot can be created with a single command.

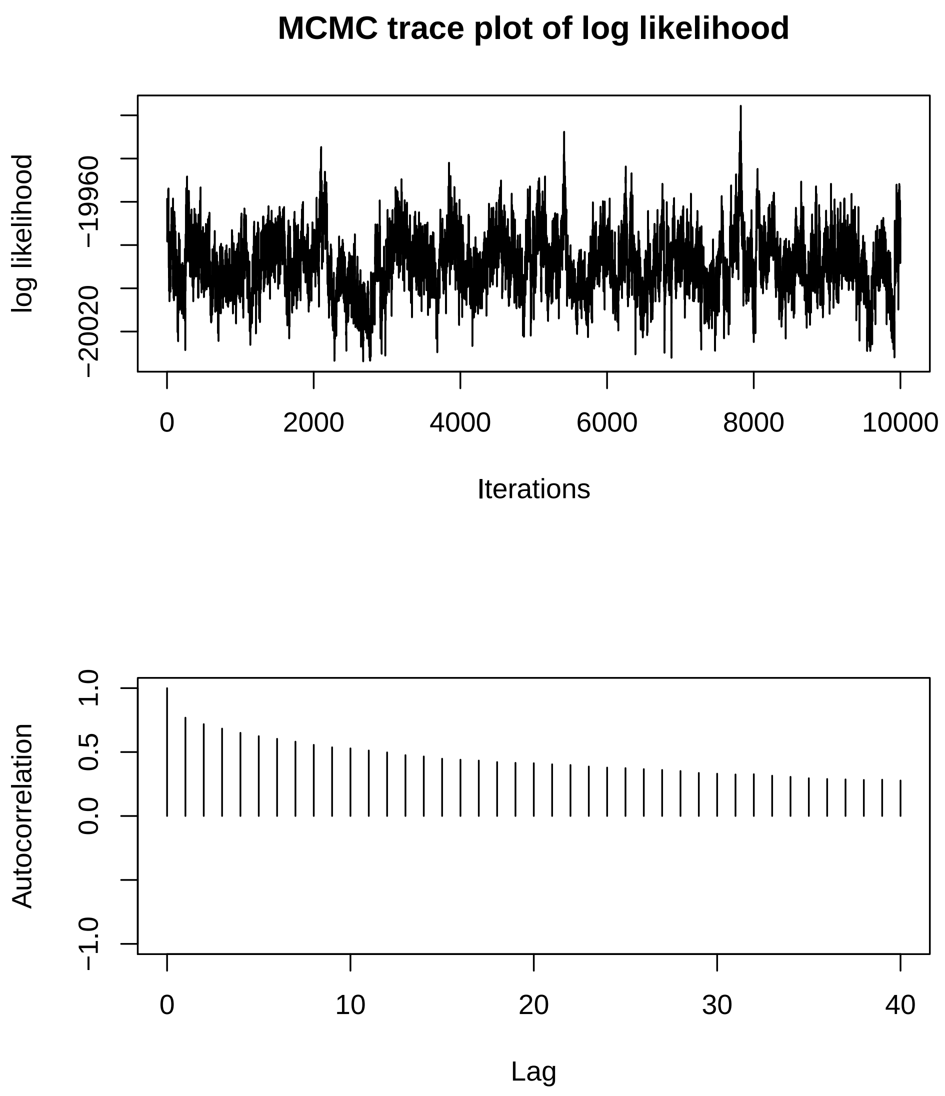


Figure 1: The top plot shows the trace plot of the log likelihood of the MCMC chain. The bottom plot shows the autocorrelation of the log likelihood of the MCMC chain.

```
> plotmcmc(db)
```

The trace plot indicates that the MCMC chain reached stationarity; the autocorrelation plot shows significant autocorrelation when subsampling the chain every 200 iterations. When increasing the length of a chain we suggest increasing the subsampling parameter, ‘subsample’, so as to avoid saving large amounts of output. The ‘burnin’ parameter controls the number of iterations discarded at the beginning of the MCMC run.

Summary plots are easily created in rbrothers. The plot command produces two graphs on top of each other. The bottom plot is the posterior probability of a breakpoint at each nucleotide position. The top plot is the posterior probability of each tree at each nucleotide position. The trees are color coded; the user can provide a list of their preferred colors via

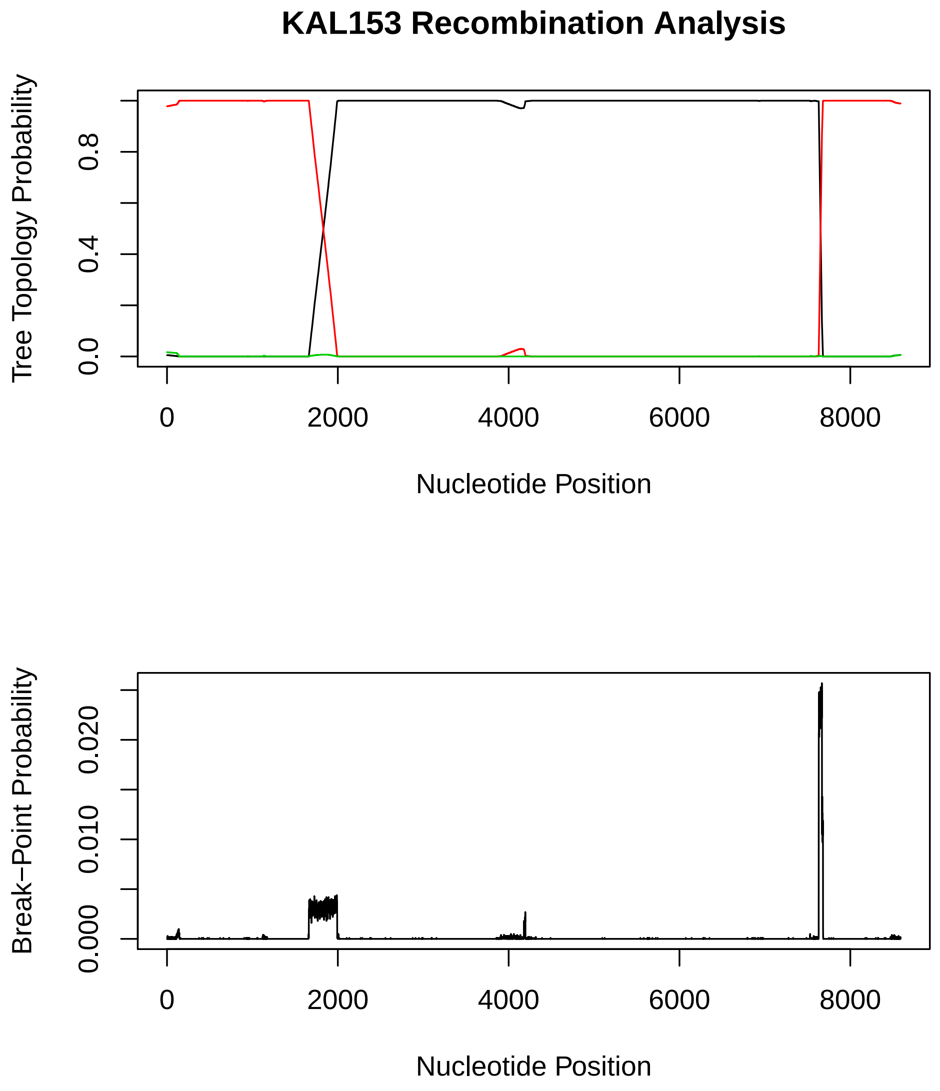


Figure 2: The top plot shows site specific posterior probabilities for the most probable phylogenetic tree topologies. The bottom plot shows the site specific posterior probability of a breakpoint.

the ‘color’ argument. The help file for plot.db explains the arguments of the plot command: `help(plot.db)`.

```
> plot(db)
```

A separate command, plottree.db, can be used to see the most probable trees. This function uses ape’s plot.phylo command so the type argument (controlling the style to display the phylogenetic tree) can be set to “phylogram”, “cladogram”, “fan”, “unrooted”, or “radial”.

```
> plottree.db(db, type = "phylogram")
```

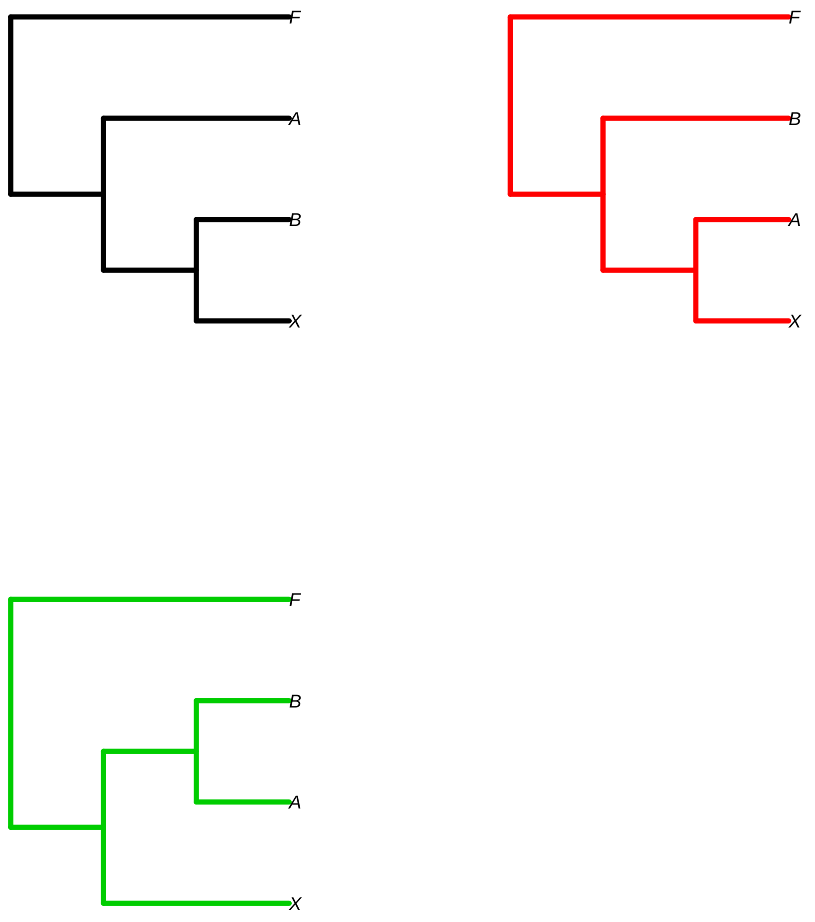


Figure 3: These are the most probable phylogenetic tree topologies.

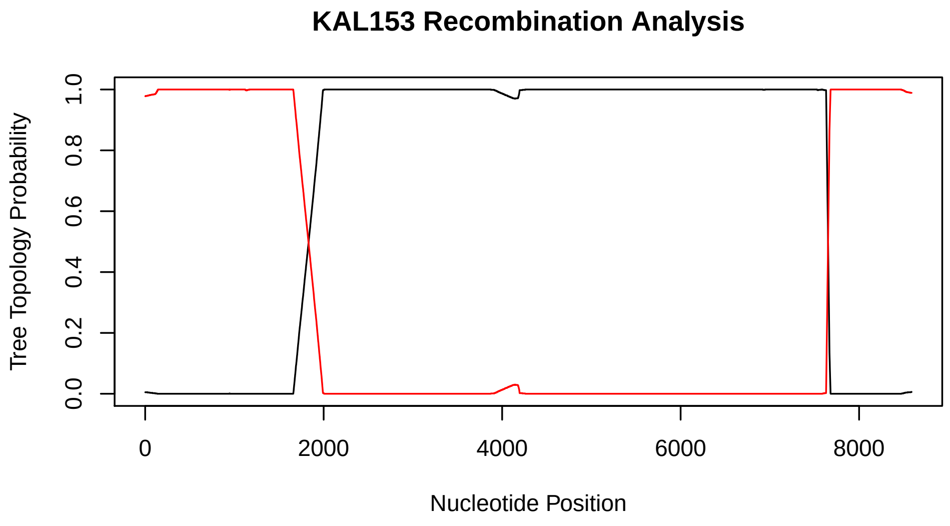
The threshold argument allows you to only plot trees that have a posterior probability greater than the threshold provided at some point along the alignment. The 'seetrees' argument allows the user to replace the posterior probability of a breakpoint plot with the trees associated with the posterior tree probability plot.

```
> plot(db, seetrees = TRUE, threshold = 0.5, type = "cladogram")
```

The average number of breakpoints, the posterior probability of at least one breakpoint and DualBrothers parameters associated with the run can be easily accessed.

```
> summary(db)
```

```
DualBrothers output for KAL153
4 sequences of length 8588
```



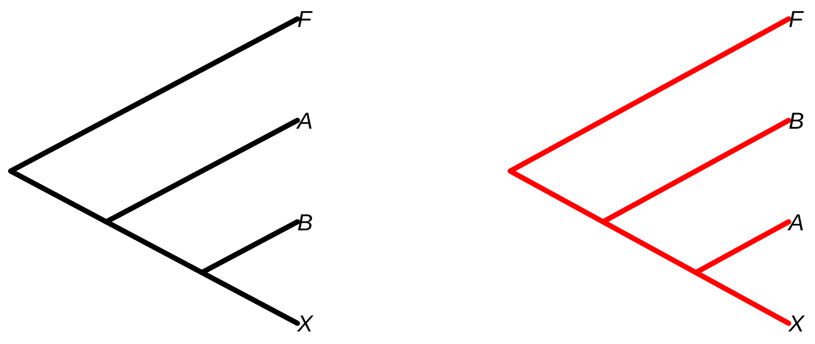


Figure 4: The top plot shows site specific posterior probabilities for the two phylogenetic trees whose posterior probability exceeded the specified threshold, 0.5. The bottom plot shows the two trees.

```
MCMC settings:
length of the MCMC chain: 2100000
burn-in length: 100000
subsample frequency: 200

Prior parameters:
prior mean number of substitution process change points: 5
prior mean number of topology change points: 0.693
```

```
average number of breakpoints in the posterior: 2.1
The posterior probability of at least one breakpoint was 1
(> 0.999  required for a Bayes factor > 1000).
3 trees considered
```

You can calculate a 95% Bayesian credible interval for a breakpoint if you can provide a nucleotide position interval that contains exactly one breakpoint, for example, (1000,3000).

```
> breakpointCI(db, 1000, 3000)

The 95% credible interval for a single break point between nucleotide
number 1000 and nucleotide number 3000 is:
 2.5% 97.5%
 1666  1985
```

References

[1] Liitsola,K. *et al.* (1998) HIV-1 genetic subtype A/B recombinant strain causing an explosive epidemic in injecting drug users in Kaliningrad. *AIDS,* **12**, 1907-1919.

[2] Minin,V. *et al.* (2005) Dual multiple change-point model leads to more accurate recombination detection. *Bioinformatics*, **21**, 3034-3042.