# Multilocus species delimitation in BEAST

Graham Jones

2013-10-06

## Contents

**1 Introduction** **1**

**2 Description** **2**
2.1 Terminology . . . 2
2.2 Notation . . . 3
2.3 Species tree prior . . . 3
2.4 Calculations . . . 5
2.4.1 Node height density . . . 5
2.4.2 Origin height density . . . 5
2.5 Implementation in BEAST . . . 6

**3 Usage** **6**
3.1 Running a Dissect analysis . . . 6
3.2 Using SpeciesDelimitationAnalyser . . . 8

**4 Simulations** **8**
4.1 Making simulated sequences . . . 8
4.2 Dissect analysis . . . 10
4.3 Results . . . 10

**5 Obtaining a development version of BEAST** **10**
5.1 Linux instructions . . . 10
5.1.1 Check out the code . . . 10
5.1.2 Build BEAST . . . 11
5.1.3 Copy some 'properties' files from source directory to build directory . . . 11
5.2 Windows instructions . . . 11
5.3 Test BEAST is working . . . 11

**6 Extras** **12**
6.1 Distribution on partitions from prior . . . 12
6.2 R code for density of origin . . . 14
6.3 R code for prior probabilities of number of species . . . 14
6.4 R code for displaying the similarity matrix . . . 16

# 1 Introduction

This note describes a method Dissect for species delimitation using the multi-species coalescent model. It is along the lines of [7] but is not constrained to a guide tree. Suppose the number of individuals is $n$. As usual, time is measured backwards from the present, which is time zero. The basic idea is to regard a species tree with $k$ tips ($1 \leq k \leq n$) as a species tree with $n$ tips in which $n-1-k$ internal node heights have been set to a very small value. Under the multispecies coalescent model, there is no difference between a single population and one that has just speciated at time zero. If some node heights are set exactly to zero, the dimensionality changes, and a reversible-jump MCMC method is needed to sample the species trees. The present algorithm

avoids the need for reversible-jump MCMC, by using a prior on node heights which mixes a density derived from the reconstructed birth-death process [3] with a spike near zero.

The model is implemented in the development version v1.8.0pre of BEAST ([1], [2]), more specifically the multispecies coalescent model *BEAST [4]. The usual birth-death model for the species tree is replaced, and a *BEAST analysis is then run. The 'species' in the analysis contain groups of individuals which are assumed a priori to belong to a single species. These groups may be merged but not split. To explore all possible groupings, the analysis can be run with one individual per species. A tool called SpeciesDelimitationAnalyser for analysing the sampled trees is also available.

Dissect might stand for Division of Individuals into Species using Sequences and Epsilon Collapsed Trees.

# 2 Description

## 2.1 Terminology

From Wikipedia:

> In mathematics, a **partition** of a set X is a division of X as a union of non-overlapping and non-empty subsets, sometimes called "parts" or "blocks" or "cells". More formally, these "cells" are both collectively exhaustive and mutually exclusive with respect to the set being partitioned.

and

> **Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics.

I use the terminology of cluster analysis here. I use **clustering** to mean a partition, that is, a complete decomposition of individuals into one or more clusters. If one knew the true clustering, the clusters would be species. I also use the term **minimal cluster** to mean a set of individuals which are assumed from the outset to belong to the same cluster.

Also from Wikipedia, a definition of the **Rand index** [6]:

> Given a set of $n$ elements $S = \{o_1, \ldots, o_n\}$ and two partitions of $S$ to compare, $X = \{X_1, \ldots, X_r\}$, a partition of $S$ into $r$ subsets, and $Y = \{Y_1, \ldots, Y_s\}$, a partition of $S$ into $s$ subsets, define the following:
>
> - $a$, the number of pairs of elements in $S$ that are in the same set in $X$ and in the same set in $Y$
> - $b$, the number of pairs of elements in $S$ that are in different sets in $X$ and in different sets in $Y$
> - $c$, the number of pairs of elements in $S$ that are in the same set in $X$ and in different sets in $Y$
> - $d$, the number of pairs of elements in $S$ that are in different sets in $X$ and in the same set in $Y$
>
> The Rand index, $R$, is:
>
> $$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

## 2.2 Notation

- $n$ is the number of individuals, and therefore the maximum possible number of species.
- $k$ is the number of species.
- $\lambda$ is the speciation rate.
- $\mu$ is the extinction rate.
- $\alpha = \lambda - \mu$ and $\beta = \mu/\lambda$.
- $w$ is the 'collapsing weight'.
- $\epsilon$ is the 'collapsing height'.
- $\mathbf{1}_A$ is the indicator function for set $A$.

## 2.3 Species tree prior

The reconstructed birth-death process [3] is used as a component of the prior for the species tree. Theorem 2.5 of [3] shows that, conditioned on the origin time $t$, the speciation rate $\lambda$, and the extinction rate $\mu$ the density of the unordered node heights are i.i.d. and are also independent of the number of tips $k$. Let the density of a node height $s$ be $f(s|k,t,\lambda,\mu) = f(s|t,\lambda,\mu)$. In the model, $f(s)$ is replaced with with a mixture of $f(s)$ and another density $m(s)$ for $s$:

$$(1-w)f(s|t,\lambda,\mu) + wm(s) \tag{1}$$

where $w$ is a user-chosen weight in $[0,1]$, and use this density for all the $n-1$ node heights in a tree with $n$ tips. The joint density is then

$$\prod_{i=1}^{n-1} \big((1-w)f(s_i|t,\lambda,\mu) + wm(s_i)\big) \tag{2}$$

where $s_1,\ldots,s_{n-1}$ are unordered node heights. This can be expanded as

$$\sum_{k=1}^{n}(1-w)^{k-1}w^{n-k}\sum_{X\in C(k)}\prod_{i\in X} f(s_i|t,\lambda,\mu)\prod_{i\notin X} m(s_i)$$

where $C(k)$ is the set of subsets of $\{1,\ldots,n-1\}$ of size $k-1$. If $m(s)$ was the Dirac delta function $\delta(s)$ the result would be a distribution in which the trees with $k$ external branches of nonzero length (that is, the trees with $k$ 'real' tips) have total probability mass

$$|C(k)|(1-w)^{k-1}w^{n-k} = \binom{n-1}{k-1}(1-w)^{k-1}w^{n-k}. \tag{3}$$

Note that the product $\prod_{i\in X} f(s_i|t,\lambda,\mu)$ is the density for a reconstructed birth-death process with $k$ tips whose node heights are the $k-1$ nonzero $s_i$. In practice one cannot sample from such a distribution without implementing reversible jump MCMC, but it can be approximated it using

$$m(s) = \epsilon^{-1}\mathbf{1}_{[0,\epsilon]}(s) \tag{4}$$

where $\epsilon$ is small.

Figures 1 and 2 illustrate the densities $f$ and $(1-w)f + wm$ respectively for the case $n = 3$, where there are two internal node heights. One way of sampling trees from the reconstructed birth-death process for $n = 3$, is to pick a point $(x, y)$ from a density such as the one in Figure 1; then choose a random ordering of the tip labels from left to right; then insert $x$ and $y$ between them; and finally join the nodes to form the tree. The same process is shown in Figure 2 for the mixture density $m$. If the point $(x, y)$ is in one of the two 'walls' along the axes, one node will be collapsed. If the point $(x, y)$ is in the 'pillar' near the origin, both nodes will be collapsed.

This is very similar to a model in which a separate reconstructed birth-death process is assumed for each $k$ and a reversible-jump MCMC is used to sample from the trees. Apart from the

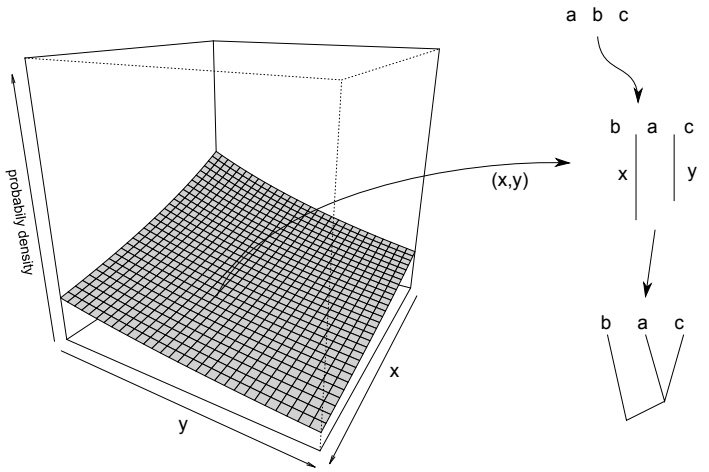


Figure 1: Sampling trees from the usual birth-death density

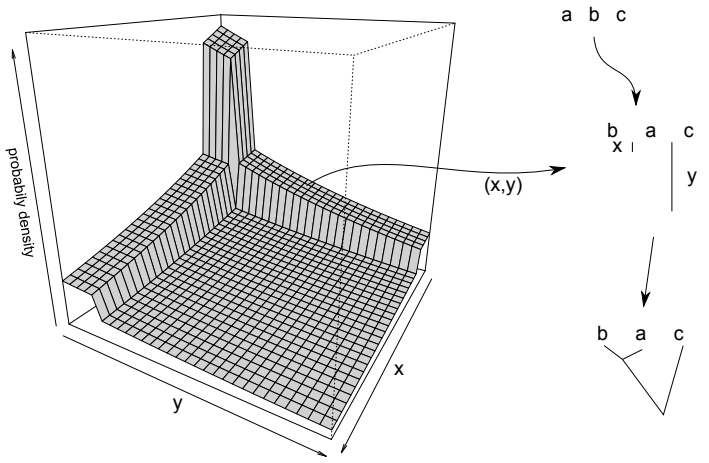


Figure 2: Sampling trees from the mixture density

approximation involving $\epsilon$, the other difference is that the density $q(t|k)$ for $t$ would normally depend on $k$ in the reversible-jump version, whereas in equation (2) there is no such dependence: a single density for $t$ for all $k$ is needed. It seems reasonable to assume a density for $q(t)$ which mixes $q(t|k)$ using the probabilities from expression (3).

The present method is similar in spirit to that of [7]. Their method requires a guide tree which drastically limits the number of clusterings considered.

## 2.4 Calculations

### 2.4.1 Node height density

Conditioned on $k$, $t$, $\lambda$, and $\mu$ the density of an unordered node height $s$ is given by Theorem 2.5 of [3] as

$$f(s|k,t,\lambda,\mu) = \frac{(\lambda-\mu)^2 e^{-(\lambda-\mu)s}}{(\lambda-\mu e^{-(\lambda-\mu)s})^2} \frac{\lambda-\mu e^{-(\lambda-\mu)t}}{1-e^{-(\lambda-\mu)t}} \mathbf{1}_{[0,t]}$$

Setting $\alpha = \lambda - \mu$ and $\beta = \mu/\lambda$ so that $\mu = \beta\lambda$ and $\lambda = \alpha/(1-\beta)$, this can be rewritten as

$$f(s|k,t,\lambda,\mu) = \frac{\alpha(1-\beta)e^{-\alpha s}}{(1-\beta e^{-\alpha s})^2} \frac{1-\beta e^{-\alpha t}}{1-e^{-\alpha t}} \mathbf{1}_{[0,t]} \tag{5}$$

where $s$ is a node height. Note that $k$ does not appear on the right hand side.

### 2.4.2 Origin height density

Usually an improper uniform prior on $[0,\infty)$ is assumed for the origin time $t$ of the tree, which is then conditioned on the number of species $k$. This conditional density for $t$ is shown in Theorem 3.2 of [3] to be

$$\begin{aligned} q(t|k) &= k\lambda^k(\lambda-\mu)^2 \frac{(1-e^{-(\lambda-\mu)t})^{k-1} e^{-(\lambda-\mu)t}}{(\lambda-\mu e^{-(\lambda-\mu)t})^{k+1}} \\ &= k\alpha(1-\beta)e^{-\alpha t} \frac{(1-e^{-\alpha t})^{k-1}}{(1-\beta e^{-\alpha t})^{k+1}} \end{aligned} \tag{6}$$

Using the probabilities from expression (3), the prior density for $t$ is

$$q(t) = \sum_{k=1}^{n} \binom{n-1}{k-1} (1-w)^{k-1} w^{n-k} q(t|k) \tag{7}$$

This can be expressed as

$$\begin{aligned} q(t) &= \sum_{j=0}^{n-1} \binom{n-1}{j} (1-w)^j w^{n-1-j} q(t|j+1) \\ &= \sum_{j=0}^{n-1} \binom{n-1}{j} (1-w)^j w^{n-1-j} (j+1)\alpha(1-\beta)e^{-\alpha t} \frac{(1-e^{-\alpha t})^j}{(1-\beta e^{-\alpha t})^{j+2}} \\ &= \alpha(1-\beta)e^{-\alpha t}(1-\beta e^{-\alpha t})^{-2} \sum_{j=0}^{n-1} (j+1) \binom{n-1}{j} w^{n-1-j} \left( \frac{(1-w)(1-e^{-\alpha t})}{1-\beta e^{-\alpha t}} \right)^j \\ &= a \sum_{j=0}^{n-1} (j+1) \binom{n-1}{j} w^{n-1-j} b^j \end{aligned}$$

where

$$a = \alpha(1-\beta)e^{-\alpha t}(1-\beta e^{-\alpha t})^{-2} \quad \text{and} \quad b = (1-w)(1-e^{-\alpha t})(1-\beta e^{-\alpha t})^{-1}. \tag{8}$$

Then

$$\begin{aligned}
q(t) &= a\left(\sum_{j=0}^{n-1}\binom{n-1}{j}w^{n-1-j}b^j + \sum_{j=0}^{n-1} j\binom{n-1}{j}w^{n-1-j}b^j\right) \\
&= a\left((w+b)^{n-1} + \sum_{i=0}^{n-2}(i+1)\binom{n-1}{i+1}w^{n-2-i}b^{i+1}\right) \\
&= a\left((w+b)^{n-1} + b\sum_{i=0}^{n-2}(n-1)\binom{n-2}{i}w^{n-2-i}b^{i}\right) \\
&= a\left((w+b)^{n-1} + (n-1)b(w+b)^{n-2}\right) \\
&= a(w+b)^{n-2}\left((w+b) + (n-1)b\right) \\
&= a(w+b)^{n-2}(w+nb). \qquad (9)
\end{aligned}$$

### 2.5 Implementation in BEAST

Equations (2) and (9) provide the basis for a new class `BirthDeathCollapseModel` extending `SpeciationModel`. It contains a parameter for the origin height $t$ as well as for $\alpha$ and $\beta$ as in the usual birth-death model.

## 3 Usage

### 3.1 Running a Dissect analysis

BEAUTi can be used to set up most of the analysis. Each species should only be assigned individuals which definitely belong together, since the program will consider merging but never splitting these groups. The word 'species', as it appears in BEAUTi and in the BEAST XML, means a minimal cluster.

Two changes need to be made to the XML. The `birthDeathModel` must be replaced with a `birthDeathCollapseModel`, and an operator must be added for the origin height. The value of $w$ can be set using the element `collapseWeight`. It can either be fixed or estimated. Even when $w$ is fixed, the prior on the number of species $k$ is somewhat diffuse. In this case, the number of trees with $k$ 'real' tips in the prior has the distribution of $1 + X$ where $X$ is a random variable having the binomial distribution with size parameter $n - 1$ and probability parameter $1 - w$. Its mean is thus $1 + (n - 1)(1 - w)$. If the individuals have been assigned in previous work to $k_0$ species, then $w = (n - k_0)/(n - 1)$ seems a reasonable choice. If $w$ is fixed at zero, the value of $\epsilon$ becomes irrelevant, and the model becomes equivalent to the birth-death model as used in *BEAST, except that the origin height is estimated instead of being integrated out analytically.

If you want to estimate $w$, you can add a hyperprior and an operator for it. The model does not permit $k$ to have a more concentrated prior than the binomial distribution decribed above, but you can obtain a more diffuse prior on $k$ with a suitable hyperprior for $w$.

The value of $\epsilon$ is set in the `birthDeathModel` using the attribute `collapseHeight`. It is measured in units of substitutions per site per generation, and should be set to a small value such as 0.0001. Initial testing suggests 0.0001 is small enough for most analyses, in the sense that smaller values will give similar results more slowly. Extremely small values may lead to poor mixing, since it may be difficult for an MCMC move to jump to a value in $[0, \epsilon]$. If $\epsilon$ is too large it will not be possible to distinguish very recent divergences. Consider Figure 2 again. There is a small probability that the point $(x, y)$, even though it has $x < \epsilon$, represents a recent speciation and not a sample from a tree with one collapsed node. When analyzing the results, a larger value than $\epsilon$ can be used as a threshold for assigning individuals to species. The choice of this threshold is inevitably subjective to some extent, and depends on one's definition of 'species'.

Optionally, `tmrcaStatistic`s may be added for groups of interest. A statistic `bdcNClustersStatistic` for the number of clusters can also be added. A trace of the latter can indicate if the MCMC chain is having trouble changing the number of almost collapsed nodes.

### XML for the birth-death-collapse model

```
<birthDeathCollapseModel id="birthDeathCollapse" units="substitutions"
              collapseHeight="0.0001">
  <speciesTree>
    <speciesTree idref="sptree"/>
  </speciesTree>
  <birthMinusDeathRate>
    <parameter id="species.birthDeathCollapse.meanGrowthRate"
                  value="100" lower="0.0" upper="Infinity"/>
  </birthMinusDeathRate>
  <relativeDeathRate>
    <parameter id="species.birthDeathCollapse.relativeDeathRate"
                          value="0.5" lower="0.0" upper="1.0"/>
  </relativeDeathRate>
  <originHeight>
    <parameter id="species.birthDeathCollapse.originHeight"
                value="0.2" lower="0.0" upper="Infinity"/>
  </originHeight>
  <collapseWeight>
    <parameter id="species.birthDeathCollapse.collapseWeight"
                        value="0.5" lower="0.0" upper="1.0"/>
  </collapseWeight>
</birthDeathCollapseModel>
```

### XML for the operator for origin height

```
<scaleOperator scaleFactor="0.75" weight="3">
  <parameter idref="species.birthDeathCollapse.originHeight"/>
</scaleOperator>
```

### XML for a number-of-clusters statistic

```
<bdcNClustersStatistic id="nClusters" name="nClusters">
  <collapseModel>
    <birthDeathCollapseModel idref="birthDeathCollapse"/>
  </collapseModel>
  <speciesTree>
    <speciesTree idref="sptree"/>
  </speciesTree>
</bdcNClustersStatistic>
```

### XML for a tmrca statistic

```
<tmrcaStatistic id="speciesTree.talpoidesHeight" name="speciesTree.talpoidesHeight">
  <speciesTree idref="sptree"/>
  <mrca>
    <taxa>
      <sp idref="Thomomys_talpoides1"/>
      <sp idref="Thomomys_talpoides2"/>
      <sp idref="Thomomys_talpoides3"/>
     </taxa>
  </mrca>
</tmrcaStatistic>
```

### 3.2 Using SpeciesDelimitationAnalyser

You can run this tool using a command line like that for BEAST, with `dr.app.beast.BeastMain` replaced by `dr.app.tools.SpeciesDelimitationAnalyser`.

The last two arguments are always an input file name and an output filename. The input file is the MCMC samples of species trees. The output is a table which lists the clusterings which are found when the nodes with small height have been collapsed. There are three optional arguments, shown in this example:

```
SpeciesDelimitationAnalyser -burnin 10000 -collapseheight .001
  -simcutoff .95 treesamples.txt out.txt
```

- burnin is the number of states to be considered as 'burn-in' which will be ignored.
- collapseheight is the height below which nodes get collapsed. This would normally be equal to or larger then the value of $\epsilon$ used in the BEAST analysis.
- simcutoff is the value above which two clusters are regarded as similar enough to support one another's credibility. The similarity between two clusterings is one minus the Rand index, and is a value between 0 and 1. The idea is that a clustering may not be very common among the samples, but may be similar to a lot of other clusterings, and the value of simcutoff affects which clustering is regarded as the 'best' summary of the results. The summary is somewhat analogous to a maximum clade credibility tree. Smaller values of simcutoff mean taking into account more dissimilar clusterings. This option should be regarded as experimental: set it to 1.0 if you want to ignore it.

Here is an imaginary example of the output to illustrate the format.

| count | fraction | similarity | nclusters | a | b | c | d |
|---|---|---|---|---|---|---|---|
| 6000 | 0.6 | 6000.0 | 1 | 0 | 0 | 0 | 0 |
| 3000 | 0.3 | 3000.0 | 2 | 0 | 1 | 1 | 1 |
| 1000 | 0.1 | 1000.0 | 3 | 2 | 0 | 1 | 2 |

The count column is the number of samples containing the clustering. The fraction is this count divided by the number of samples. The similarity column is a measure of how similar the clustering is to others. It is always at least as big as the count, and when simcutoff is set to 1.0, it is identical. Note that the clusterings are sorted by their similarity.

The remaining columns describe a clustering. The nclusters column is the number of clusters. The other columns are labelled with the names of the 'species' (minimal clusters) used in the BEAST XML for the analysis. The numbers in the columns show how these are grouped; the numbers themseves are arbitrary, but if two columns have the same value, the minimal clusters are grouped together in the clustering. So the table above shows clusterings **abcd**, **a+bcd** and **ad+b+c**.

This can be used to produce into a similarity matrix, which contains the posterior probabilities of each pair of minimal clusters belong to the same cluster. See table 1 for the similarity matrix corresponding to the imaginary example, and Figure 3 for a larger example.

## 4 Simulations

### 4.1 Making simulated sequences

The simulations all use 20 individuals, 4 assigned to each of 5 species **a**, **b**, **c**, **d**, and **e**. The species tree is shown in Figure 4. The effective number of genes in the population was 100,000 at the tips and at the rootward ends of branches, and 200,000 at the tipwards ends of internal branches, varying linearly along the branches. Sequences (genes) with 500 sites were generated for these gene trees using Seq-Gen [5] called with command `seqgen.exe -mHKY -t3.0 -f0.3,0.2,0.2,0.3`. This uses a strict clock and the HKY substitution model, and all genes have the same mutation rate. There is no site rate heterogeneity. The mutation rate $T$ was set to 1e-8 or 4e-8 per site per generation, and the number of genes $G$ was set to 3 or 9, making 4 cases in total. The root of the

| | a | b | c | d |
|---|---|---|---|---|
| a | 1 | .6 | .6 | .7 |
| b | .6 | 1 | .9 | .9 |
| c | .6 | .9 | 1 | .9 |
| d | .7 | .9 | 1 | .9 |

Table 1: Similarity matrix

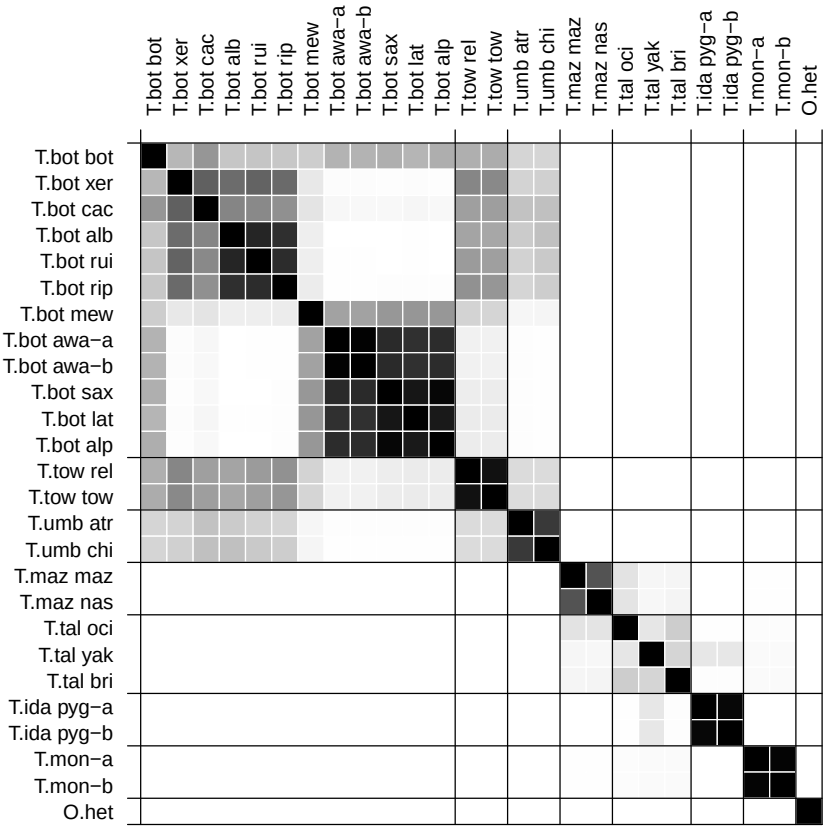


Figure 3: Similarity matrix for *Thomomys* data. w=0.70, $\epsilon$=0.0001.

species tree is at 0.008/T generations, and is therefore 800,000 generations when $T$ = 1e-8 and 200,000 generations when $T$ = 4e-8.

### 4.2 Dissect analysis

From these sequences, BEAST XML files were made and run. In the BEAST analyses, a strict clock and no site rate heterogeneity were assumed. The HKY substitution model was assumed, but with unknown parameters. The relative clock rates of the genes were estimated. The parameter $w$ was estimated; it was given a hyperprior which was a beta distribution with shape parameters 4 and 2. The value of $\epsilon$ was 0.0001. All runs were 15 million generations with the first 5 million discarded as burnin. The threshold in SpeciesDelimitationAnalyser was set to 0.0001, the same as $\epsilon$.

### 4.3 Results

The figures are the end of this paper.

Figures 8, 6, 7, 8 show some examples of gene trees estimated from the simulated sequences using an UPGMA algorithm (upgma from the R package phangorm). These give an idea of the amount of incomplete lineage sorting, and of how often single genes from individuals belonging to different species are identical.

Figures 9, 10, 11, 12 show the resulting similarity matrices.

Figure 13 shows the estimated species tree in the case G=9, T=4e-8. Posterior probabilities are shown on the branches, and smaller values are shown paler.

Figure 14 shows the trace of the number of clusters during the MCMC chain, in the case G=9, T=4e-8. It appears to mix reasonably well (ESS=160 from 10k samples from 10M generations). The posterior mean of $w$ was 7.6.

## 5 Obtaining a development version of BEAST

You will need:

- Java and a Java development kit (JDK) installed.
- Subversion installed. `http://subversion.apache.org/`
- Ant installed. `http://ant.apache.org/`

### 5.1 Linux instructions

#### 5.1.1 Check out the code

Type

```
svn checkout http://beast-mcmc.googlecode.com/svn/trunk/ beast16
```

`beast16` names a directory; you can use another name. You should see a long listing of files ending something like

```
A    beast16/examples/incorrect/testOTFPCLikelihood.xml
 U   beast16
Checked out revision 4366.
```

### 5.1.2 Build BEAST

`cd` to `beast16` and type `ant`. You should see something like this:

```
[gjones@pc158250 beast16]$ ant
Buildfile: build.xml

clean:

init:
     [echo] BEAST: /home/gjones/beast16/build.xml

compile-all:
    [mkdir] Created dir: /home/gjones/beast16/build
    [javac] Compiling 1836 source files to /home/gjones/beast16/build
    [javac] Note: Some input files use or override a deprecated API.
    [javac] Note: Recompile with -Xlint:deprecation for details.
    [javac] Note: Some input files use unchecked or unsafe operations.
    [javac] Note: Recompile with -Xlint:unchecked for details.
     [echo] Successfully complied.

dist-all:
    [mkdir] Created dir: /home/gjones/beast16/build/dist
      [jar] Building jar: /home/gjones/beast16/build/dist/beast.jar
      [jar] Building jar: /home/gjones/beast16/build/dist/beauti.jar

build:

BUILD SUCCESSFUL
Total time: 37 seconds
[gjones@pc158250 beast16]$
```

### 5.1.3 Copy some 'properties' files from source directory to build directory

Type

```
cp beast16/src/dr/app/beast/*.properties  beast16/build/dr/app/beast/
```

## 5.2 Windows instructions

You have to do the same steps: download the code, compile it, and run it via a java command. I use TortoiseSVN which is a GUI for Subversion and Eclipse which is an IDE for Java. This is overkill if you just want to run a development version of BEAST, but I have not tried any other way.

## 5.3 Test BEAST is working

In order to run BEAST you need to execute a java command and supply a 'classpath' which is in this case a list of three places (`.` meaning 'here', `/home/gjones/beast16/build/`, and `/home/gjones/beast16/lib/*`) to tell java where to look for code. Then there is the java class `BeastMain` to run and the XML file `YuleSingleLocus.xml` which is input to BEAST. Under Linux, all as one line:

```
java -classpath '.:/home/gjones/beast16/build/:/home/gjones/beast16/lib/*'
   dr.app.beast.BeastMain beast16/examples/release/starBEAST/YuleSingleLocus.xml
```

Under Windows, the classpath is separated by `;` not `:` and the code is a different place (`bin` not `build`) and file paths use `\` not `/` and the single quotes don't seem to be needed. It might look like this:

```
java -classpath .;C:\workspace\beast16\bin;C:\workspace\beast16\lib/*
   dr.app.beast.BeastMain beast16\examples\release\starBEAST\YuleSingleLocus.xml
```

You should see a normal BEAST run. If you get an error like this

```
Exception in thread "main" java.lang.NoClassDefFoundError:
 jebl/evolution/treemetrics/RootedTreeMetric
....
```

it is probably a problem with the classpath and/or the directory you are in when you issue the command.

# 6 Extras

## 6.1 Distribution on partitions from prior

Probability distributions on the partitions of a finite set are of general interest. Some of these distributions arise as the result of a stochastic process. The *Chinese restuarant process* (see Wikipedia) is one example, and there are also a variety of *preferential attachment processes* (see Wikipedia). One of the latter is known as 'the Yule process' which models the emergence of new genera among species. None of these seems to be the same as the distribution associated with the model used here. The following result gives the probability of a given clustering in the model used here, conditioned on the number of clusters. The point process described in [3] shows why the merging process in the proposition produces the same distribution as sampling a tree using the method shown in Figures 1 and 2, and then slicing through the tree. The result is almost surely already published somewhere, but I can't find it. It may be useful for calculating Bayes factors for different species delimitation hypotheses.

**Proposition.** *Assume a merging process which starts with $n$ objects each assigned to its own cluster. Clusters are then merged by randomly choosing pairs among those available at each step until there are $r$ clusters. Let $b_1, b_2, \ldots, b_r$ be the sizes of the clusters in a particular clustering $B$ produced by this process. Then*

$$\begin{aligned}\Pr(B|r) &= \frac{r!\,(n-r)!\,(r-1)!\,b_1!\;b_2!\ldots b_r!}{(n-1)!\,n!} \\ &= r!\binom{n-1}{r-1}^{-1}\binom{n}{b_1, b_2, \ldots, b_r}^{-1}\end{aligned}$$

**Proof.** First we count the total number $T$ of ordered mergings that make $r$ clusters from $\{1, 2, \ldots, n\}$. Then we count the number $S$ of ordered mergings that make $B$ and obtain $\Pr(B|r) = S/T$. It may help to think of a particular example of $B$ such as

$$B = \{\ \{1, 2, \ldots, b_1\},\ \{b_1{+}1,\ b_1{+}2, \ldots, b_1{+}b_2\}, \ldots, \{n{-}b_r{+}1,\ n{-}b_r{+}2, \ldots, n\}\ \}.$$

We have

$$\begin{aligned}T &= \binom{n}{2}\binom{n-1}{2}\cdots\binom{n-r}{2} \\ &= \frac{n!}{r!}\frac{(n-1)!}{(r-1)!}2^{n-r}\end{aligned}$$

For $S$, we multiply the numbers of ways in which each the clusters of sizes $b_1, b_2, \ldots, b_r$ can be formed by the number of ways that the mergings of different clusters can be interleaved. There are $b-1$ mergings for a cluster of size $b$, so there are

$$\binom{n-r}{b_1-1, b_2-1, \ldots, b_r-1}$$

interleavings, and

$$\begin{aligned} S &= \binom{n-r}{b_1-1, b_2-1, \ldots, b_r-1} \prod_{i=1}^{r} \frac{b_i!(b_i-1)!}{2^{b_i-1}} \\ &= (n-r)!2^{n-r} \prod_{i=1}^{r} b_i! \end{aligned}$$

and the result follows. □

Suppose that there are $s_i$ clusters of size $i$ $(1 \le i \le n)$. Note that $\sum_{i=1}^{n} s_i = r$. Then there are

$$\binom{n}{b_1, b_2, \ldots, b_r} \left( \prod_{i=1}^{n} s_i! \right)^{-1}$$

partitions of $\{1, 2, \ldots, n\}$ which have the same shape (the same partition of the integer n). So the probability that the $r$ clusters have a given shape is

$$r! \binom{n-1}{r-1}^{-1} \left( \prod_{i=1}^{n} s_i! \right)^{-1}$$

Example. Suppose $n = 8, r = 3$. Then

$$\Pr(4+2+2) = 3! \binom{7}{2}^{-1} (0!\,2!\,0!\,1!\,0!\,0!\,0!\,0!)^{-1} = 1/7$$

and there are

$$\binom{8}{4, 2, 2} (0!\,2!\,0!\,1!\,0!\,0!\,0!\,0!)^{-1} = 210$$

partitions of $\{1, 2, \ldots, 8\}$ which have this shape.

The Chinese restuarant process is different from the merging process above. The probabilities for $n = 6$ are shown in table 1. When conditioned on $r = 2$, the Chinese restuarant process gives probabilities in ratio 72:45:20 for 5+1,4+2,3+3, whereas the merging process gives 2:2:1. When conditioned on $r = 3$, the Chinese restuarant process gives 6:8:1 for 4+1+1,3+2+1,2+2+2, whereas the merging process gives 3:6:1.

Table 2: Chinese restuarant process

| Shape | P(partition) | #ptns per shape | P(shape) |
|---|---|---|---|
| 6 | 120/720 | 1 | 120/720 |
| 5+1 | 24/720 | 6 | 144/720 |
| 4+2 | 6/720 | 15 | 90/720 |
| 3+3 | 4/720 | 10 | 40/720 |
| 4+1+1 | 6/720 | 15 | 60/720 |
| 3+2+1 | 2/720 | 60 | 120/720 |
| 2+2+2 | 1/720 | 15 | 15/720 |
| 3+1+1+1 | 2/720 | 20 | 40/720 |
| 2+2+1+1 | 1/720 | 45 | 45/720 |
| 2+1+1+1+1 | 1/720 | 15 | 15/720 |
| 1+1+1+1+1+1 | 1/720 | 1 | 1/720 |

## 6.2 R code for density of origin

```
origindensity <- function(x, n, a, b, w) {
E <- exp(-a*x)
B <- (1 - E) / (1-b*E)
z <- 1
z <- z * a
z <- z * (1-b)
z <- z * E
z <- z / (1-b*E)^2
z <- z * (w + (1-w)*B)^(n-2)
z <- z * (w + n*(1-w)*B)
z
}


log.origindensity <- function(x, n, a, b, w) {
E <- exp(-a*x)
B <- (1 - E) / (1-b*E)
z <- 0
z <- z + log(a)
z <- z + log(1-b)
z <- z - a*x
z <- z - 2 * log(1-b*E)
z <- z + (n-2) * log(w + (1-w)*B)
z <- z + log(w + n*(1-w)*B)
z
}

n <- 50
a <- 1
b <- 0.9
w <- 0.5

cat("mean nof species in prior is ", 1 + (n-1)*(1-w), "\n")
x <- seq(from=0, to=10, length.out=1001)
matplot(x, cbind(log(origindensity(x, n, a, b, w)), log.origindensity(x, n, a, b, w)), type="l")
integrate(origindensity, lower=0, upper=Inf, n=n, a=a, b=b, w=w)
```

## 6.3 R code for prior probabilities of number of species

If $w$ has a beta distribution for its hyperprior, with parameters $a$ and $b$ (`shape` and `shapeB` in the XML) and there are $n$ individuals, then the prior probabilities of number of species is given by

$$\Pr(k = x|n, a, b) = \frac{\Gamma(n)}{\Gamma(x)\Gamma(n+1-x)} \frac{\Gamma(x-1+b)\Gamma(n-x+a)}{\Gamma(n-1+a+b)} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}$$

and the function `px` below calculates this, for $1 \leq x \leq n$.

```
px <- function(x, n, a, b) {
p <- 1
p <- p * gamma(n)
p <- p / gamma(x)
p <- p / gamma(n+1-x)
p <- p * gamma(x-1+b)
p <- p * gamma(n-x+a)
p <- p / gamma(n-1+a+b)
p <- p * gamma(a+b)
p <- p / gamma(a)
p <- p / gamma(b)
p
}

z <- px(1:26, 26, 5,2)
sum(z)
plot(z)
```

## 6.4 R code for displaying the similarity matrix

```
# Read in the table of clusterings
workdir <- "C:/Users/Graham/AAA/Programming/tmp/birthdeathcollapse/w70e0001/"
x <- read.table(paste(workdir, "SDA_OUT.txt", sep=""), header=TRUE)

# Abbreviations for display
renames <- matrix(c(
"Orthogeomys_heterodus1",  "O.het",
"Thomomys_bottae1",  "T.bot awa-a",
"Thomomys_bottae10",  "T.bot mew",
"Thomomys_bottae11",  "T.bot sax",
"Thomomys_bottae12",  "T.bot lat",
"Thomomys_bottae2",  "T.bot awa-b",
"Thomomys_bottae3",  "T.bot xer",
"Thomomys_bottae4",  "T.bot cac",
"Thomomys_bottae5",  "T.bot alb",
"Thomomys_bottae6",  "T.bot rui",
"Thomomys_bottae7",  "T.bot bot",
"Thomomys_bottae8",  "T.bot alp",
"Thomomys_bottae9",  "T.bot rip",
"Thomomys_idahoensis1",  "T.ida pyg-a",
"Thomomys_idahoensis2",  "T.ida pyg-b",
"Thomomys_mazama1",  "T.maz maz",
"Thomomys_mazama2",  "T.maz nas",
"Thomomys_monticola1",  "T.mon-a",
"Thomomys_monticola2",  "T.mon-b",
"Thomomys_talpoides1",  "T.tal oci",
"Thomomys_talpoides2",  "T.tal yak",
"Thomomys_talpoides3",  "T.tal bri",
"Thomomys_townsendii1",  "T.tow tow",
"Thomomys_townsendii2",  "T.tow rel",
"Thomomys_umbrinus1",  "T.umb chi",
"Thomomys_umbrinus2",  "T.umb atr"),
nrow=26, ncol=2, byrow=TRUE)

# Minimal cluster names are column names omitting first 4
mincl.names <- colnames(x)[-(1:4)]
# Check for typos, etc
for (i in 1:length(mincl.names)) {
  stopifnot(mincl.names[i] == renames[i,1])
  }
```

```
# Make the similarity matrix
displaynames <- renames[,2]
nmincls <- length(displaynames)
sim <- matrix(0, ncol=nmincls, nrow=nmincls, dimnames=list(displaynames, displaynames))
for (i in 1:nmincls) {
  for (j in 1:nmincls) {
    coli <- x[,mincl.names[i]]
    colj <- x[,mincl.names[j]]
    w <- coli == colj
    sim[i,j] <- sum(x[w,"fraction"])
  }
}

# change the order of minimal clusters
# This depends on which patterns you want to emphasise
neworder <- c(11,7,8,9,10,13,3,2,6,4,5,12,  24,23,  26,25,  16,17, 20,21,22, 1, 14,15, 18,19)
# Currently recognised groups
dividers <- c(0,12,14,16,18,21,22,24,26)

plot.rectangle <- function(v1,v2,...)
{
polygon(c(v1[1],v2[1],v2[1],v1[1]), c(v1[2],v1[2],v2[2],v2[2]), ...)
}

# Main plotting routine
plot.simmatrix <- function() {
  par(mar= c(0,5,5,0)+.1)
  plot(NULL, xlim=c(0,nmincls), ylim=c(nmincls,0), axes=FALSE, ylab="", xlab="")
  axis(3, at=(1:nmincls)-.5, displaynames[neworder], tick=FALSE, las=2, line=-1)
  axis(2, at=(1:nmincls)-.5, displaynames[neworder], tick=FALSE, las=2, line=-1)
  for (i in 1:nmincls) {
    for (j in 1:nmincls) {
      d <- 1 - sim[neworder[i],neworder[j]]
      plot.rectangle(c(i-1,j-1), c(i,j), col=rgb(d,d,d), border="white")
    }
  }
  for (b in dividers) {
    lines(x=c(-.5,nmincls), y=c(b,b))
    lines(x=c(b,b), y=c(-.5,nmincls))
  }
}

# Display as text, on screen, and to PDF file
print(sim[neworder,neworder], digits=2)
plot.simmatrix()
pdf(file=paste(workdir, "simmatrix.pdf", sep=""))
plot.simmatrix()
dev.off()
}
```

# References


[1] Alexei J Drummond and Andrew Rambaut. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, 7:214, 2007.

[2] Alexei J. Drummond, Marc A. Suchard, Dong Xie, and Andrew Rambaut. Bayesian phylogenetics with BEAUti and the BEAST 1.7. *Molecular Biology and Evolution*, 29:1969–1973, 2012.

[3] T Gernhard. The conditioned reconstructed process. *J. Theo. Biol.*, 253:769–778, 2008.

[4] J Heled and A Drummond. Bayesian inference of species trees from multilocus data. *Mol. Biol. Evol.*, 27:570–580, 2010.

[5] A Rambaut and N C Grassly. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput Appl Biosci*, 13:235–238, 1997.

[6] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association (American Statistical Association)*, 66(336):846–850, 1971.

[7] Z Yang and B Rannala. Bayesian species delimitation using multilocus sequence data. *Proceedings of the National Academy of Sciences of U.S.A*, 107:9264–9269, 2010.

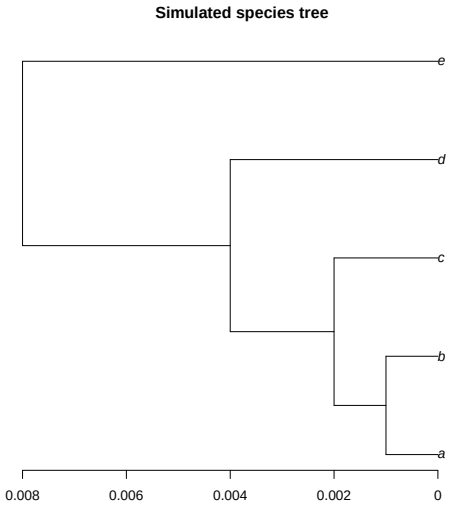


Figure 4: Species tree used in simulations

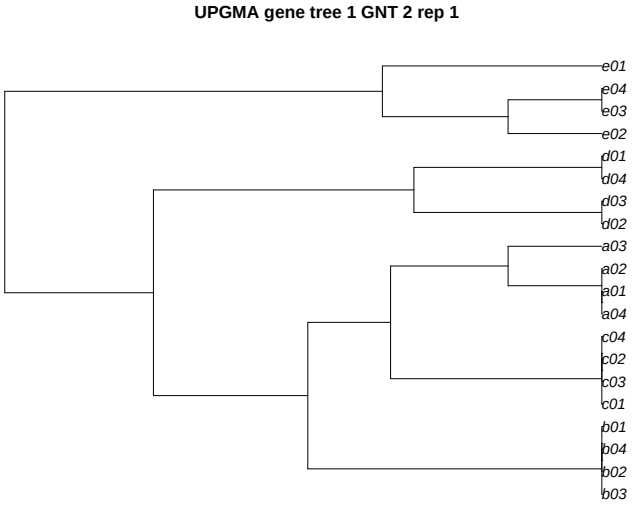


Figure 5: 'Good' example of UPGMA gene tree with T=1e-8

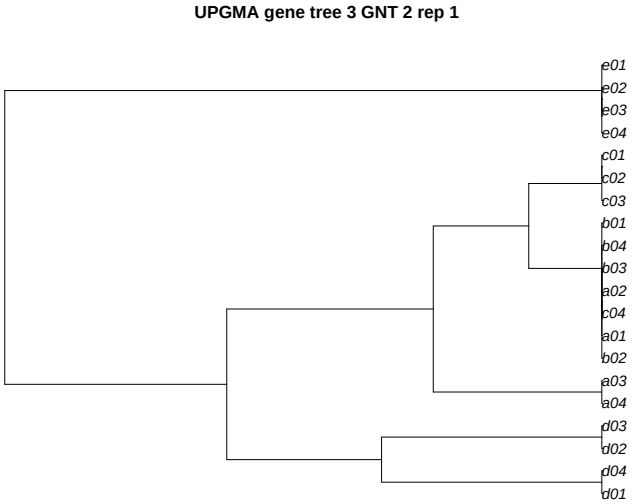


Figure 6: 'Bad' example of UPGMA gene tree with T=1e-8

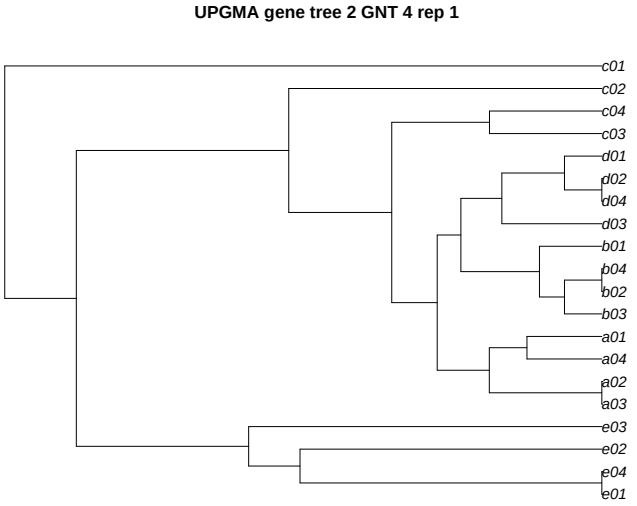


Figure 7: 'Good' example of UPGMA gene tree with T=4e-8

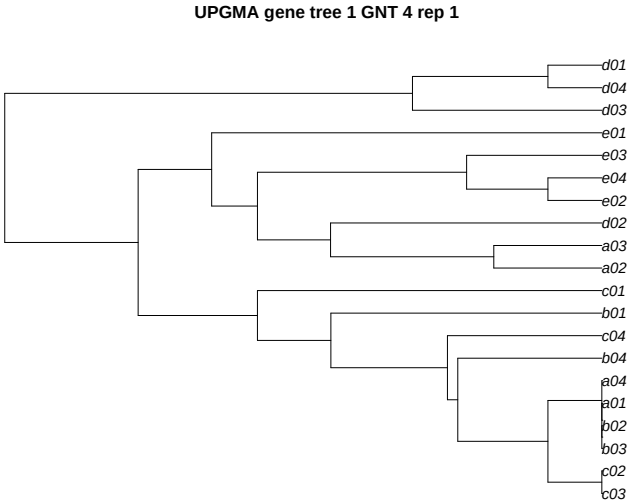


Figure 8: 'Bad' example of UPGMA gene tree with T=4e-8

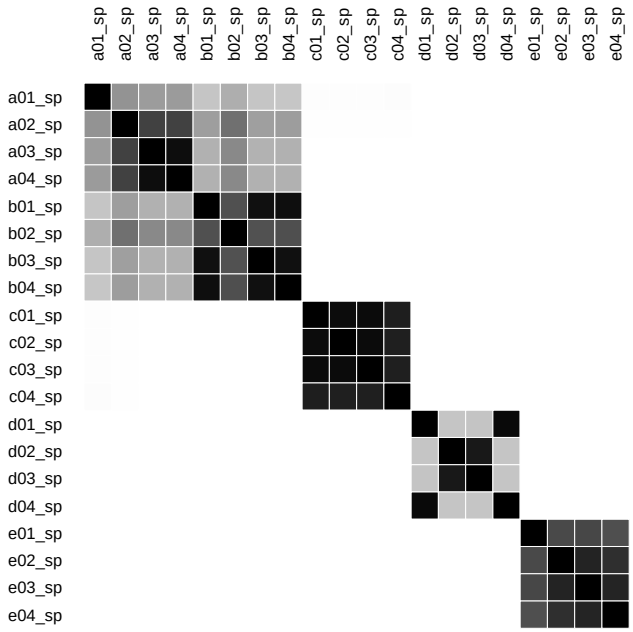


Figure 9: Similarity matrix estimated from simulated data. G=3, T=1e-8, $w \sim$ B(4,2), $\epsilon$=0.0001.

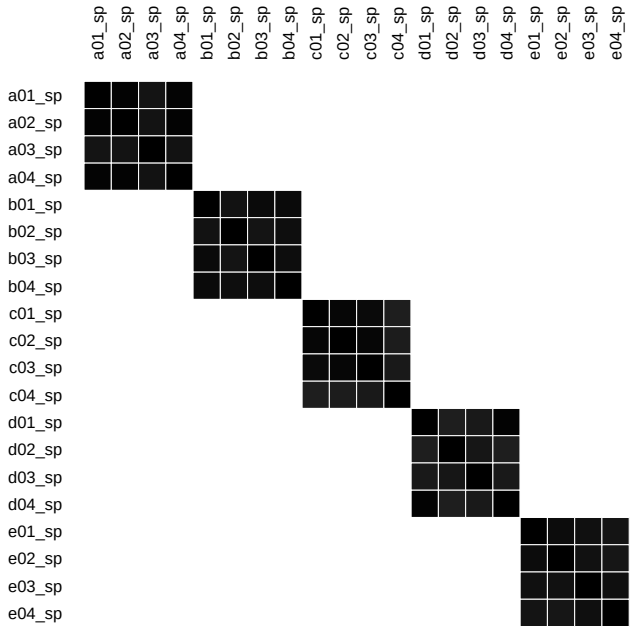


Figure 10: Similarity matrix estimated from simulated data. G=9, T=1e-8, $w \sim$ B(4,2), $\epsilon$=0.0001.

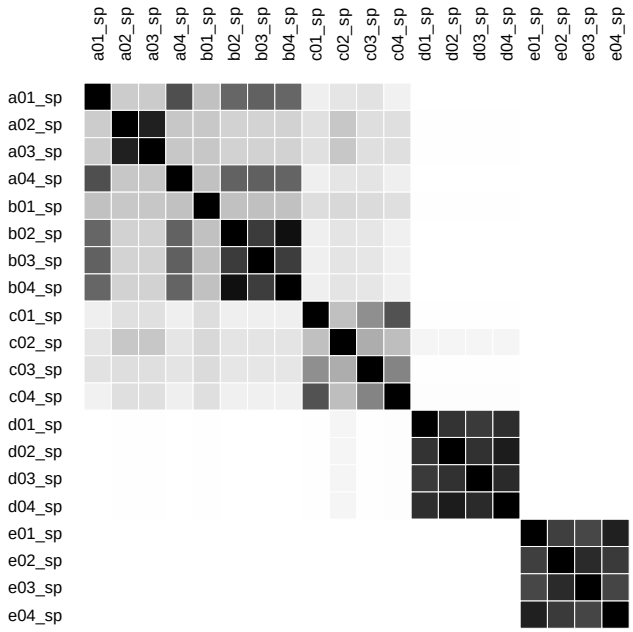


Figure 11: Similarity matrix estimated from simulated data. G=3, T=4e-8, $w \sim$ B(4,2), $\epsilon$=0.0001.

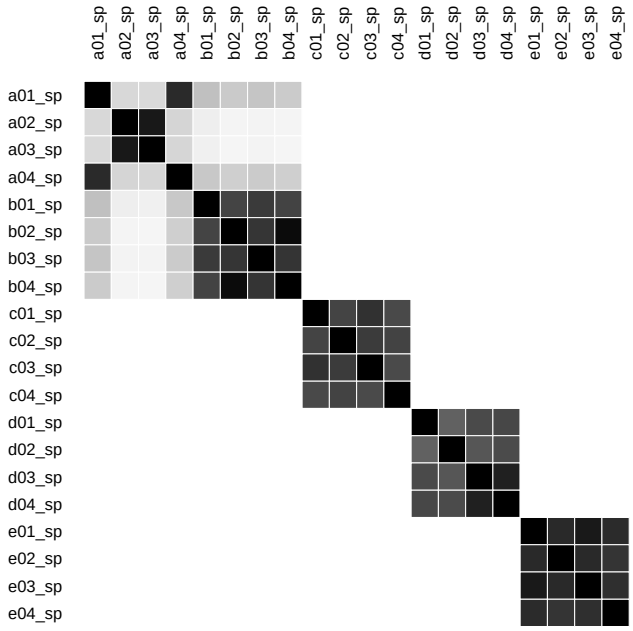


Figure 12: Similarity matrix estimated from simulated data. G=9, T=4e-8, $w \sim$ B(4,2), $\epsilon$=0.0001.

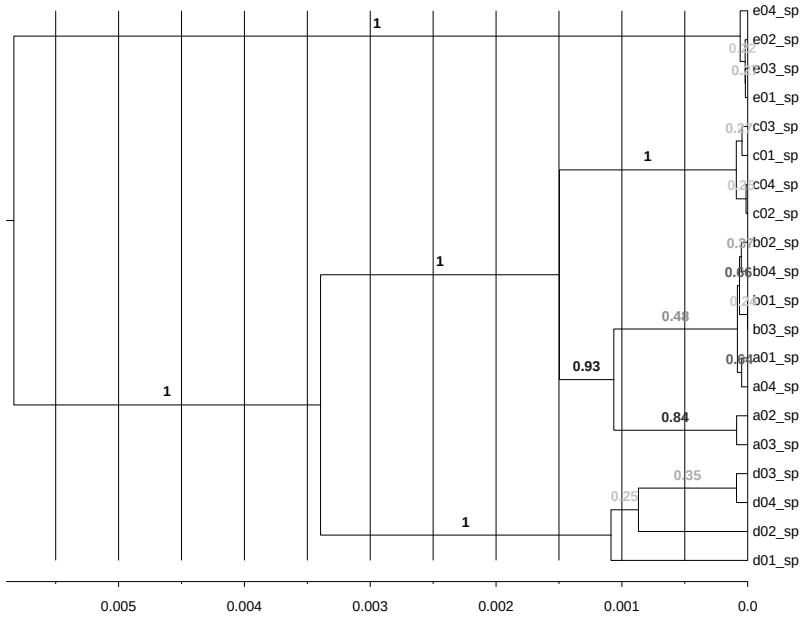


Figure 13: Estimated species tree in case G=9, T=4e-8

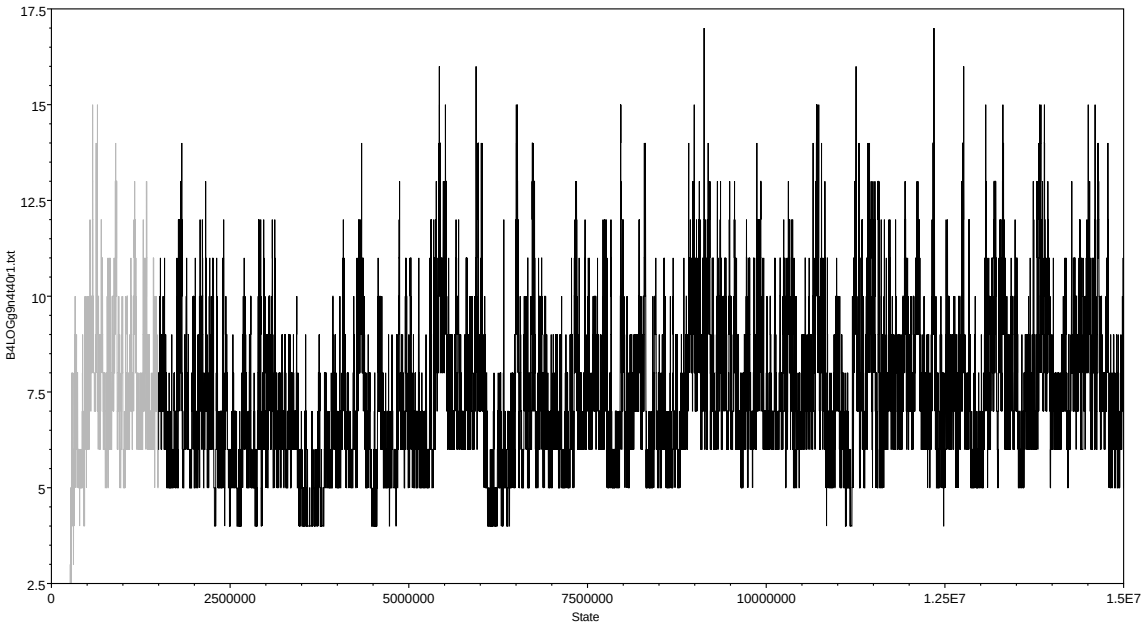


Figure 14: Trace of the number of clusters during the MCMC chain, in case G=9, T=4e-8