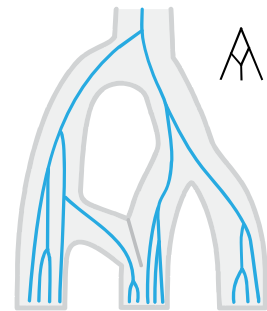


BP&P

VERSION 4.5 (June 2022)

© Copyright 2002-2022, Ziheng Yang, Thomas Flouris



The software package is provided “as is” without warranty of any kind. In no event shall the author or his employer be held responsible for any damage resulting from the use of this software, including but not limited to the frustration that you may experience in using the package. The program is distributed under the GNU GPL v3.

Table of Contents

BP&P VERSION 4.5 (June 2022)	1
Table of Contents	1
0. Introduction	2
1. Changes since version 4.0	3
2. Compiling and running BPP	3
2.1 Compiling the program	3
2.2 Trial run	4
3. File formats	4
3.1 Sequence data file and individual map (Imap) file	4
3.2 Control file	5
3.2.1 The control variables	5
3.2.2 Analyses A00, A01, A10, and A11	13
3.2.3 The MSci model and the extended Newick format	15
4. Screen and file outputs	17
4.1 A00 screen output	17
4.2 Adjusting step lengths for MCMC moves (finetune)	18
4.3 A01: species tree estimation	19
4.4 A10: species delimitation using rjMCMC	20
4.5 A11: joint species delimitation and species tree estimation	23
5. BPP4-specific features: threading, checkpointing and constraints	24
6. References	27
Appendix A. Definitions of the migration rate in different programs	29
Appendix B. Marginal likelihood calculation using BFdriver 4.0	30
Appendix C. The simulation option	32
C1 Control-file options for simulating under the MSC model	32
C2 Simulating with MSC-M (or IM) model	36
Appendix D. The msci-create facility for generating the Newick notation of MSci models	37

0. Introduction

BPP is a Bayesian Markov chain Monte Carlo (MCMC) program for analyzing DNA sequence alignments from multiple loci and multiple closely-related species under the multispecies coalescent (MSC) model (Rannala and Yang, 2003; Yang, 2002). See Xu and Yang (2016) for a review of the MSC model. The program can be used to conduct four different analyses, specified using two variables in the control file:

- A00 (**speciesdelimitation** = 0, **speciestree** = 0): estimation of the parameters of species divergence times and population sizes under the MSC and MSci models when the species tree model is given (Rannala and Yang, 2003; Flouri et al., 2020a);
- A01 (**speciesdelimitation** = 0, **speciestree** = 1): inference of the species tree when the assignments are given by the user (Rannala and Yang, 2017);
- A10 (**speciesdelimitation** = 1, **speciestree** = 0): species delimitation using a user-specified guide tree (Yang and Rannala, 2010; Rannala and Yang, 2013);
- A11 (**speciesdelimitation** = 1, **speciestree** = 1): joint species delimitation and species tree inference of unguided species delimitation (Yang and Rannala, 2014).

The basic parameters in the MSC model include the species divergence times (τ s) and population size parameters $\theta = 4N\mu$, where N is the effective population size and μ is the mutation rate per site per generation so that θ is the average proportion of different sites between two sequences sampled at random from the population. Both τ s and θ s are measured by the expected number of mutations per site. For a species tree with s species, there are $(s - 1)$ divergence times (τ s) and at most $(2s - 1)$ population size parameters (θ s). Analysis A00 is to estimate those parameters when the species delimitation and species tree is fixed. Analyses A01, A10, and A11 compare different MSC models.

The multispecies-coalescent-with-introgression (MSci) model is introduced in version 4.1, which has hybrid nodes in the tree and at each hybrid node there is a new parameter, the introgression probability (ϕ). See Flouri *et al.* (2020a). For reviews of the MSC model, see Yang (2014: Chapter 9), Xu and Yang (2016), Rannala *et al.* (2020), and Jiao *et al.* (2021). Please also refer to the BPP tutorial, which illustrates the four analyses (Yang, 2015; Flouri et al., 2020b).

The MSC model implemented in the program makes two basic assumptions about the data: there is no recombination between sites of the same locus, and free recombination between loci. Some analyses also assume there is gene flow between species (either continuous migration or episodic introgression), although the MSci model allows episodic introgression (Flouri et al., 2020a). The default model assumes the molecular clock or rate constancy among lineages, and relaxed-clock models are implemented through the variable clock. The ideal loci are thus loosely linked short genomic segments: each segment is short (within 500 or 1000bp) so that recombination within the segment is rare while different segments should be far apart so that recombination between them is so common that the segments have more or less independent histories. It is assumed that the genealogical trees are not affected by natural selection. Protein-coding gene sequences appear to be useable in a BPP analysis since most proteins are performing similar functions in closely related species and the main effect of purifying selection on nonsynonymous mutations is a reduction of the neutral mutation (Shi and Yang, 2018; Thawornwattana et al., 2018). It is a good idea to separate the noncoding and coding regions of the genome into two datasets. The program currently uses the JC69 mutation model (Jukes and Cantor, 1969) to correct for multiple hits, and mutation rate is assumed to be constant over time. The use of the JC model and the clock assumption mean that the program should be limited to closely related species with sequence divergence not much higher than 10%. Work is under way to extend those models.

Note that a major factor that the MSC model accommodates is the genealogical heterogeneity across genome: that is, different regions of the autosomal genome have different gene tree topologies and branch lengths (coalescent times). While the mutational process may also vary along the genome, this heterogeneity is believed to be much less important. Thus multiple genes from the mitochondrial genome should be treated as one ‘locus’ in the MSC-based analysis. Typically the mitochondrial genome has different mutation rate from the nuclear genome, and also a different effective population size from the autosomes. While the model of locus-rate variation (option variable `locusrate`) and the heredity scalar (option variable `heredity`) are designed to deal with this, it may be prudent to analyze the loci from the nuclear genome separately from the single mitochondrial locus. Also the model assumes that the sequences are random samples from the different species. If some sequences from the same species are identical, they should all be used, and it is incorrect to use the haplotypes only, which will lead to biased parameter estimates. Similarly it is incorrect to filter loci based on bootstrap support values and use only loci with high phylogenetic information signal.

How to cite the program. You can cite the BPP tutorial and the original papers that described the methods you used (see above). Describe the priors you used since they are necessary for reproducibility. If you conduct a joint analysis of species delimitation and species tree inference, your method description may look like the following (replace the numbers in green with those you used):

“Joint Bayesian species delimitation and species tree estimation was conducted using the program BPP (Yang, 2015; Flouri et al., 2018). The method uses the multispecies coalescent model to compare different models of species delimitation (Yang and Rannala, 2010; Rannala and Yang, 2013) and species phylogeny (Yang and Rannala, 2014; Rannala and Yang, 2017) in a Bayesian framework, accounting for incomplete lineage sorting due to ancestral polymorphism and gene tree-species tree discordance. The population size parameters (θ s) are assigned the inverse-gamma prior $IG(3, 0.002)$, with mean $0.002/(3 - 1) = 0.001$. The divergence time at the root of the species tree (τ) is assigned the inverse-gamma prior $IG(3, 0.004)$, with mean 0.002 , while the other divergence time parameters are specified by the uniform Dirichlet distribution (Yang and Rannala, 2010: equation 2). Each analysis is run at least twice to confirm consistency between runs.”

1. Changes since version 4.0

- i. Multiple threads
- ii. Introgression model (MSci)
- iii. Local-clock and locus-rates models (`clock` and `locusrate`)
- iv. Topological constraints (`constraint` and `outgroup`)

2. Compiling and running BPP

2.1 Compiling the program

This manual applies to BPP version 4.1. The program is written in C and can be compiled for LINUX, MACOSX, and WINDOWS. It is available at <https://github.com/bpp/bpp>.

BPP is a command-line program, so the best way of running it is from the command line, rather than double-clicking on the file name from your file explorer. If you have not used the command line before, please work through one of the following short tutorials first: <http://abacus.gene.uc1.ac.uk/software/CommandLine.Windows.pdf>

<http://abacus.gene.ucl.ac.uk/software/CommandLine.MACosx.pdf>

WINDOWS and MAC OSX executables will be provided at the download site. For LINUX or MAC OSX, you may need to compile the programs to generate the executable file `bpp`. This needs to be done only once. For example, the following commands use the `gcc` compiler to compile the program and move the generated executable file (`bpp`) into the `bin/` folder.

```
BPP4.0
-----
cd bpp
md bin
cd src
make
mv bpp ../bin
```

The same source code can be compiled into a simulation program (`MCcoal`). See the section *The simulation program (MCcoal)* later in this document for details.

2.2 Trial run

Run the program from a command box (rather than double-clicking the executable) so that you will see the error messages. In the `bpp/` folder, run the program by typing the following command:

On WINDOWS	On LINUX/UNIX/MAC OSX
<code>bin\bpp --cfile bpp.5s.ctl</code>	<code>bin/bpp --cfile bpp.5s.ctl</code>

3. File formats

3.1 Sequence data file and individual map (Imap) file

The sequence file. The sequence alignments are in the phylip/paml format, with one alignment following the other, all in one file. The sequence name should be separated from the sequence by a line break or by at least two spaces. Have a look at the sequence files `test5s.txt` and `ChenLi2001.txt`. An alignment is called a locus. Every locus must have at least 2 sequences, different loci can have different numbers of sequences, and some species may be missing at some loci.

By default (`cleandata = 0`), alignment gaps and ambiguity nucleotides are used in the likelihood calculation, with gaps treated as question marks (see Yang, 2006, pp. 107-108). If `cleandata = 1`, all columns with gaps or ambiguity characters are removed before analysis.

The Imap file. Each sequence name has a tag after `^`. For example, the sequence name `GI01234567^Specimen1` has the individual ID `Specimen1`. An Imap file then maps the individuals (specimens) to the species/populations. For example, the following Imap file maps `Specimen1` to species `A`.

<code>Specimen1</code>	<code>A</code>
<code>Specimen2</code>	<code>A</code>
<code>Specimen3</code>	<code>A</code>
<code>Specimen4</code>	<code>B</code>
<code>Specimen5</code>	<code>C</code>
<code>Specimen6</code>	<code>C</code>

The MSC model implemented in BPP uses only the population ID (such as `A`, `B`, ... in the example), and does not use the individual ID. Also when BPP reads the sequence names, it use the individual ID to retrieve the species ID for each sequence but then ignores the sequence name. The motivation for the use of the Imap file and this two-layer design is that

one may analyze the same sequence data with different population/species assignments, which can be achieved by editing the small Imap file without editing the much bigger sequence data file.

3.2 Control file

3.2.1 The control variables

The default control file name is `bpp.ct1`. Lines beginning with an asterisk are comments. Most often the order of the lines is unimportant, but there are exceptions (as explained below). We use `ChenLi2001.bpp.ct1` in the `examples/` folder to explain the variables in the control file. This is for analysis A00: parameter estimation under the MSC using multiple-loci multiple-species data on a fixed species tree (Rannala and Yang, 2003; Burgess and Yang, 2008). We will then comment on the other analyses (A01, A10, and A11). Please also read the section on the example datasets later in this document.

Below is a copy of the control file `ChenLi2001.bpp.ct1`. Note that if there are s species in the species tree, the model will involve the following parameters: $(s - 1)$ species divergence times (τ s) and $(s - 1)$ ancestral θ s. If a species has at least two sequences at any loci, a θ for that species will be used as well. If there is one species, the model will involve one parameter only, θ for that species. The parameters are ordered as follows: θ s for the extant species, θ s for the ancestral species ($s - 1$ of them), and the divergence times τ s for the ancestral nodes ($s - 1$ of them).

```
seed = -1

seqfile = ChenLi2001.txt
Imapfile = ChenLi2001.Imap.txt
outfile = out.txt
mcmcfile = mcmc.txt

speciesdelimitation = 0      * fixed species delimitation
*speciesdelimitation = 1 0 2 * rjMCMC speciesdelimitation algorithm0(e)
*speciesdelimitation = 1 1 2 1 * rjMCMC speciesdelimitation algorithm1(a m)
speciestree = 0             * species tree fixed
*speciestree = 1           * NNI over species/guide trees

*speciesmodelprior = 1 * 0: uniform LH; 1:uniform rooted trees; 2: uniformSLH; 3: uniformSRooted

species&tree = 4 H C G O
               1 1 1 1
               (((H, C), G), O);
phase = 1 1 1 1 * 0: do not phase, 1: phase diploid unphased sequences

usedata = 1 * 0: no data (prior); 1:seq like
nloci = 53 * number of datasets in seqfile
* model = jc * default
* model = gtr * default
* alphaprior = 1 1 4 * gamma prior for alpha parameter for site rates and ncatG

cleandata = 0 * remove sites with ambiguity data (1:yes, 0:no)?

* thetaprior = 3 0.002 e # inverse gamma(a, b) for theta
thetaprior = 3 0.002 # inverse gamma(a, b) for theta
tauprior = 3 0.03 # inverse gamma(a, b) for root tau & Dirichlet(a) for other tau's
phiprior = 1 1 # beta(a, b) for phi in the MScI model

*locusrate = 0 # (0: No rate variation among loci. This is default)
*locusrate = 1 10.0 10.0 5.0 iid # (1: estimate locus rates mui)
*locusrate = 1 0 0 5.0 iid # (1: estimate locus rates mui)
*locusrate = 2 LocusRateFileName # (2: locus rates from file)
*clock = 1 # (1: strict clock, default)
*clock = 2 10.0 100.0 5.0 iid G # (2: independent-rates)
*clock = 2 a_vbar b_vbar a_vi <prior> <distribution> # (2: independent-rates)
*clock = 3 10.0 100.0 5.0 iid G # (3: correlated-rates)
*clock = 3 a_vbar b_vbar a_vi <prior> <distribution> # (3: correlated-rates)
* heredity = 0 # (0: No variation, 1: estimate, 2: from file) & a_gamma b_gamma
* heredity = 1 4 4 # (0: No variation, 1: estimate, 2: from file) & a_gamma b_gamma
* heredity = 2 heredity.txt # (0: No variation, 1: estimate, 2: from file) & a_gamma b_gamma

* sequenceerror = # model of sequencing errors has changed, to be described later
```

```

finetune = 1: .01 .02 .03 .04 .05 .01 .01 # auto (0 or 1): MCMC step lengths
print = 1 0 0 0 0 * print MCMC samples, locusrates(mu_i,nu_i), heredityscalars, genetrees, locusrateparameters
burnin = 16000
sampfreq = 2
nsample = 200000

```

seed is the random number seed. If you use a positive integer, the program will produce identical results in different runs. This is useful for debugging. If you use -1 , the program will use the wall clock to generate a seed, and different runs will produce different results. It is recommended that you run the same analysis at least twice using different seeds to confirm that the results are stable across runs.

seqfile is the name of the sequence alignment file, while **Imapfile** is the individual map file. The Imapfile is not needed if the data contain only one species. These two files are input.

```

speciesdelimitation = 0      * fixed species delimitation
speciestree = 0             * no change to species tree or guide trees

```

speciesdelimitation = 0 and **speciestree = 0** specify analysis A00: estimation of parameters under the MSC model. For analyses A01, A10, and A11, those two variables may be specified the value 1. See the next subsection below. If those two variables are missing or their lines are commented out, the default value of 0 is assumed. The variable **speciesmodelprior** is used in analyses A01, A10, and A11, and has no effect for analysis A00.

```

species&tree = 4  H  C  G  O
                  1  1  1  1
                (((H, C), G), O);

```

This block is for the MSC model. The specifications for the MSci model with introgression is different. See the section on MSci model.

This block specifies 4 species in the data, which are H (human), C (chimp), G (gorilla), and O (orang). The ‘maximum number of sequences’ at any locus is 1 for H, 1 for C, 1 for G, and 1 for O. The program interprets these numbers as follows. First, if the number of sequences for a species is 2 or greater, θ for that species will be a parameter estimated by the program. Second, the sum of the numbers of sequences over all species specifies the maximum number of sequences at a locus. It will be fine if you specify 2 sequences for a species when there are 10 sequences for that species at some loci in the data file.

BPP always uses θ and τ (age) for every interior (ancestral) node on the species tree, and uses θ for each extant species if and only if that species has more than 2 or more sequences at some loci. In the example here, the parameters are the three ancestral θ s and three node ages (τ s). If you specify 2 or more sequences for a species but there are at most 0 or 1 sequence per locus for that species in the sequence data file, θ for that species will not be identifiable or estimable. In that case, the posterior for the parameter will be the prior. Nevertheless, other results (such as the posterior distribution for other parameters or posterior probabilities for species trees and species delimitations) will still be correct. The same applies to other more complex cases of missing data and parameter unidentifiability. As an example, suppose the species tree and the numbers of sequences for two kinds of loci are as follows:

```

((A,B), (C,D))
locus configuration 1: 1 1 0 0
locus configuration 2: 2 0 0 1

```


Then θ_A , θ_{AB} , θ_{ABCD} , τ_{AB} and τ_{ABCD} are identifiable while θ_B , θ_C , θ_D , θ_{CD} and τ_{CD} are not. For example, it is impossible to estimate θ_D , θ_{CD} and τ_{CD} if no data are available from C.

As another example, suppose the species tree is

((A,B), (C,D));

locus configuration 1: 1 1 0 0

locus configuration 2: 2 0 0 10

Then θ_A , θ_D , θ_{AB} , θ_{CD} , θ_{ABCD} , τ_{AB} , τ_{CD} and τ_{ABCD} are all identifiable while θ_B and θ_C are not.

We are essentially fitting a demographic model of population size change along the path D-CD-ABCD. The information in the data for θ_{CD} and τ_{CD} may be weak so those parameters may be estimated poorly.

The species tree is given in the parenthesis (Newark) format, ending with a semicolon (;).

The tree is fixed if `speciesdelimitation = 0` and is used as the guide tree in the rjMCMC run for species delimitation if `speciesdelimitation = 1` (see below).

`phase = 1 1 1 1` * 0: do not phase, 1: phase diploid unphased sequences

The `phase` variable is specified using a 0-1 flag for each species/population, with 0 (the default) indicating that sequences from that species are fully phased haplotype sequences, while 1 means unphased diploid sequences. If this line is missing, all sequences are assumed to be fully phased (that is, flag 0 for all species). The program does not allow some sequences from a species to be phased sequences while other sequences from the same species to be unphased.

If the flag is 1, each sequence from that species is treated as unphased diploid sequence, with heterozygote sites represented using ambiguity characters YRMKSW. N-? are allowed and are treated as ?, but other ambiguities (HBVD) are not allowed. Each sequence is then expanded/resolved into two sequences, and the program averages over all the possible phase resolutions according to the approach of Gronau *et al.* (2011). For example a sequence with two heterozygote sites R and Y are resolved into two possible phase resolutions: (i) A...T and G...C, and (ii) A...C and G...T. Note that this option forces a sequence with no heterozygote sites to be phased/resolved into two identical sequences. Intuitively the MSC model averages over different phase resolutions of the heterozygote sites in the likelihood calculation. The option may increase the memory usage and CPU time considerably if there are many sequences with many heterozygote sites at one locus.

The simulation program MCcoal has been modified to simulate diploid unphased data as well. This allows one to see the difference between the analysis of the full data and the unphased data.

If the phase flag for a species is 0, all sequences from that species are treated as resolved haplotype sequences, and ambiguities are interpreted in the usual way (for example, Y means one nucleotide that is either T or C).

`usedata = 0` is for running the MCMC without sequence data to generate the prior, while `usedata = 1` is for generating the posterior.

`nloci` specifies the number of loci (alignments). If you have 200 loci in the data file and choose `nloci = 2`, BPP will read the first two loci only.

`cleandata = 1` causes the program to remove all columns in the alignment which have gaps or ambiguity characters. `cleandata = 0` means that those will be used in the likelihood calculation.

`thetaprior = 3 0.002` specifies the inverse-gamma prior $IG(\alpha, \beta)$ for the θ parameters, with the mean to be $\beta/(\alpha - 1)$. In the example, the mean is $0.002/(3 - 1) = 0.001$ (one difference per kb).

Note that all θ parameters in the MSC model (for both modern species and extinct ancestral species) are assigned the inverse-gamma prior with the same parameters. The inverse-gamma is a conjugate prior for θ (Hey and Nielsen, 2007), which means that both the prior and the posterior of θ will be inverse-gamma. Use of the conjugate priors allows the θ parameters to be integrated out analytically, and thus the dimension of the parameter space is reduced. This typically leads to improved mixing of the MCMC. However, with this option, the posterior of the θ parameters will not be produced. To estimate the θ parameters, add the letter e (or E) on the line, as follows

`thetaprior = 3 0.002 e`

Whether θ s are integrated out or estimated, other results (such as the posterior of the τ parameters or the posterior probability of species trees or species-delimitation models) should be identical. A useful strategy may be to run the A01, A10, and A11 analyses without estimating the θ parameters, and after the MAP model (the best species tree or the best species delimitation model) is identified, run the A00 analysis with the model fixed to estimate all parameters.

Some notes about the inverse-gamma distribution. Since BPP3.4, both the θ and τ parameters are assigned the inverse gamma priors rather than the gamma priors in version 3.3 or earlier. One difference is that the gamma is light-tailed while the inverse-gamma is heavy-tailed, so that the inverse-gamma may be less influential than the gamma if your prior mean is much too small. The inverse-gamma distribution $IG(\alpha, \beta)$ has mean $m = \beta/(\alpha - 1)$ if $\alpha > 1$ and variance $s^2 = \beta^2/[(\alpha - 1)^2(\alpha - 2)]$ if $\alpha > 2$, while the coefficient of variation is $s/m = \sqrt{1/(\alpha - 2)}$. If little information is available about the parameters, you can use $\alpha = 3$ for a diffuse prior and then adjust the β so that the mean looks reasonable. Both parameters θ s and τ s in the MSC model are measured by genetic distance, the expected number of mutations/substitutions per site. For example, for the human species, $\theta_H \approx 0.0006$, which means that two random sequences from the human population are different at $\sim 0.06\%$ of sites, less than 1 difference per kb. A sensible diffuse prior is then “`thetaprior = 3 0.002`”, with mean $0.002/(3 - 1) = 0.001$.

`tauprior = 3 0.03` specifies the inverse-gamma prior $IG(\alpha, \beta)$ for τ_0 , the divergence time parameter for the root in the species tree. Other divergence times are generated from the uniform Dirichlet distribution (Yang and Rannala, 2010: equation 2). In the example, the mean is $0.03/(3 - 1) = 0.015$ (which means 1.5% of sequence divergence between the root of the species tree and the present time). If the mutation rate is 10^{-9} mutations/site/year, this distance will translate to a human-orangutan divergence time of 15MY.

`phiprior = 1 1` specifies the beta prior $\text{beta}(a, b)$ with $a = 1$ and $b = 1$ for ϕ , the introgression probability under the MSci model.


```

locusrate = 0                # (0: No rate variation among loci. This is default)
locusrate = 1 10 10 5.0 iid  # (1: estimate locus rates mui)
locusrate = 1 0 0 5.0 iid    # (1: estimate locus rates mui)
locusrate = 2 LocusRateFileName # (2: locus rates from file)

```

`locusrate = 0` (default) means that all loci have the same mutation rate.

`locusrate = 2 LocusRateFileName` specifies the fixed-rates model of locus-rate variation Burgess & Yang (2008). This is the strategy used by Yang (2002), with the relative rates estimated by the distance to an outgroup species. The relative locus rates are listed in the file: there should be as many numbers in the file, separately by spaces or line returns, as the number of loci (`nloci`). The program re-scales those rates so that the average across all loci is 1 and then use those relative rates as fixed constants.

```

locusrate = 1 a_mubar b_mubar a_mui <prior>
locusrate = 1 10 10 5 iid      # (1: estimate locus rates mui)
locusrate = 1 0 0 5 iid       # (1: estimate locus rates mui, mubar=1 fixed)

```

The `locusrate` variable takes 5 arguments, the last one (`prior`) being optional. Parameters `a_mubar` and `b_mubar` are the shape and rate parameters of the gamma distribution for the mean rate across loci (`mubar` or $\bar{\mu}$), while `a_mui` and `prior` are used to specify the locus rates (μ_i) given the mean rate ($\bar{\mu}$). The option `prior` can take two values `dir` (for Gamma-Dirichlet rates for loci), and `iid` (for conditional i.i.d. or hierarchical prior).

Specifically the mean rate across loci ($\bar{\mu}$) is assigned a gamma prior:

$mubar \sim \text{Gamma}(a_mubar, b_mubar)$ with mean a_mubar/b_mubar .

When there are no fossil calibrations in the species tree, the rates should all be relative. In this case we suggest fixing `a_mubar = 0` and `b_mubar = 0` (with 0 meaning ∞), and the program will fix the mean rate across loci at $\bar{\mu} = 1$. Otherwise one can use equal and large values such as `a_mubar = 100` and `b_mubar = 100`, so that $\bar{\mu}$ is nearly fixed at 1.

Given the mean rate $\bar{\mu}$, two priors are available to specify the locus rates μ_i , with $i = 1, 2, \dots, L$, where L is the number of loci (`nloci`). If `prior = dir` (for Gamma-Dirichlet distribution of locus rates), the total rate $L\bar{\mu}$, given the mean rate ($\bar{\mu}$), is partitioned into locus rates (μ_i), using the concentration parameter `a_mui` (α). The model and notation follow Burgess and Yang (2008: equation 4) and dos Reis et al. (2014: equations 3-5).

If `prior = iid` (for conditional-i.i.d. or hierarchical prior of locus rates), the locus rates (μ_i) are i.i.d. given the mean rate ($\bar{\mu}$):

$\mu_i \sim \text{Gamma}(a_mui, a_mui/mubar)$

This model is described in Zhu et al. (2015: equation 8) and implemented in MCMCTREE.

In both the Gamma-Dirichlet and the conditional i.i.d. models, parameter α (`a_mui`) is inversely related to the extent of rate variation among loci, with a large α meaning similar rates among loci. If all loci are noncoding, the rates are probably similar, so $\alpha = 10$ or 20 may be reasonable, while for coding loci or exons, $\alpha = 2$ or 1 may be appropriate. The α parameter may affect the estimates of the population size parameter (θ) for the root node on the species tree (Burgess and Yang, 2008). The L locus rates (μ_i) are parameters in the model. If `a_mubar > 0`, the mean rate ($\bar{\mu}$) is a parameter as well.

Note about the locus-rate prior in IMA. The model of variable rates among loci implemented here has some differences from a similar model implemented in the IMA program (Hey and Nielsen, 2004). The biggest difference appears to be the parametrization. BPP defines mutation rate on a per-nucleotide basis, so the prior specifies that the expectation of the mutation rate per site is constant among loci. IMA defines mutation rate on a per-locus basis, so its prior specifies that the expectation of the mutation rate per locus is the same

among loci. If locus one has 100 sites and locus two has 1000 sites, then IM assumes that the per-site rate for locus one is 10 times for locus two, while BPP assumes the same per-site rate. Also IMA constrains the geometric mean of rates across loci to be one, while BPP constrains their arithmetic mean to be one.

```
clock = 1          # (1: strict clock, default)
clock = 2 10.0 100.0 5.0 iid G          # (2: independent-rates)
clock = 2 a_vbar b_vbar a_vi <prior> <distribution> # (2: independent-rates)
clock = 3 10.0 100.0 5.0 iid G          # (3: correlated-rates)
clock = 3 a_vbar b_vbar a_vi <prior> <distribution> # (3: correlated-rates)
```

clock = 1 (strict clock) means that all branches on the gene tree have the same mutation rate and the molecular clock holds. This is the default.

```
clock = 2 10.0 100.0 5.0 iid G          # (2: independent-rates)
clock = 2 a_vbar b_vbar a_vi <prior> <distribution> # (2: independent-rates)
```

This specifies the local-clock models, in which the mutation rate drifts over branches, independently among loci. The **clock** variable takes 6 arguments. If both the prior option for the variable **locusrate** and the prior option for the variable **clock** are set, they should take the same value. The option **distribution** can be G (for gamma) and LN (for lognormal); this is the distributional form for branch rates (r_{ij}) given the locus rate (μ_i). Parameters **a_vbar** and **b_vbar** are the parameters of the gamma distribution for the average of variances across loci (\bar{v} or \bar{v}), while **a_vi** and **prior** are used to specify the variance (v_i) for locus i given the average (\bar{v}). The option **prior** can take two values: **dir** (for Gamma-Dirichlet for loci), and **iid** (for conditional i.i.d. or hierarchical prior). The specification of the distribution of v_i given the mean \bar{v} follows the same procedure as the specification of the distribution of μ_i given the mean rate $\bar{\mu}$; see notes above about the **locusrate** variable.

Thus the specification (**clock = 2 10.0 100.0 5.0 iid G**) means the following. First \bar{v} is assigned a gamma distribution $G(10.0, 100.0)$, with mean 0.1. Given \bar{v} , the conditional i.i.d. prior means that $v_i | \bar{v} \sim \text{Gamma}(\alpha, \alpha/\bar{v})$ with shape parameter $\alpha = 5.0$ (**a_vi**) and mean \bar{v} , for $i = 1, 2, \dots, L$. Note that α (**a_vi**) here is inversely related to the variance in v_i : use small values of α (2, 1, or 0.5) if you believe that v_i varies among loci (meaning that the clock nearly holds at some loci but is seriously violated at others).

As another example, the specification (**clock = 2 10.0 100.0 5.0 dir LN**) means that \bar{v} is assigned a gamma distribution $G(10.0, 100.0)$, and then the sum $L\bar{v}$ is partitioned into v_i (for $i = 1, 2, \dots, L$) according to the Dirichlet distribution (**prior = dir**) with concentration parameter $\alpha = 5.0$ (**a_vi**). Again large α means the same extent of clock violation at different loci.

Given the locus rate μ_i (specified using the **locusrate** variable) and the variance v_i (specified using the **clock** variable) for locus i , the different lineages may have different rates at the locus, and those rates are independent among loci. If **distribution = G** (for gamma), the rate for species-tree branch j at locus i has the following gamma distribution

$$r_{ij} | \mu_i, v_i \sim \text{Gamma}(\mu_i^2 / v_i, \mu_i / v_i),$$

This has mean μ_i and variance v_i .

Alternatively if **distribution = LN** (for log-normal), the rate for species-tree branch j at locus i has the following the log-normal distribution

$$r_{ij} | \mu_i, v_i \sim \text{LN}(\mu_i, v_i),$$

where μ_i is the mean of the LN distribution and v_i is the variance parameter of the lognormal.

Note that the rate-drift model specifies rates for branches on the species tree (rather than on the gene tree) for each locus, and gene-tree branches residing in the same population or

species have the same rate. For example, if all sequences at a locus are from the same species and all coalescent events occur in that species (before reaching an ancestral species), all branches on the gene tree will have the same rate even if the relaxed-clock model allows rates among species. In contrast, if a gene-tree branch passes several species or populations, the different segments of the branch will have different rates. The branch length on the gene tree is calculated by summing up the lengths of those segments (with the length of each segment being the product of the rate and the time duration for the segment).

```
clock = 3 10.0 100.0 5.0 iid LN      # (3: correlated-rates)
clock = 3 10.0 100.0 5.0 iid G       # (3: correlated-rates)
clock = 3 a_vbar b_vbar a_vi <prior> <distribution> # (3: correlated-rates)
```

The option `clock = 3` is similar to clock 2 but specifies the autocorrelated-rates model. `clock = 3` with the log-normal distribution (**LN**) specifies the geometric Brownian motion model of Rannala and Yang (2007). This assigns a rate to each species-tree branch, that is, to the mid-point of the branch. Given the rate at the species-tree root (μ_i at locus i), the rates for the two branches around the root are specified. Then given the rate for each ancestral branch, the rates for its two daughter branches are specified, by integrating over the rate at the internal node that is ancestral to the daughter branches. See figure 1 and equations 3-8 in Rannala and Yang (2007). The rates for all species-tree branches are thus assigned through a pre-order tree traversal, starting from the root moving to the tips, until all branches are visited.

If `clock = 3` is specified with the gamma distribution (**G**), the model works as follows, using a similar pre-order tree traversal. First the two branches at the species-tree root have the gamma distribution with mean μ_i (`mu_i`) and variance ν_i (`nu_i`). Then given the rate for each ancestral branch, the rates for its two daughter branches are specified as independent gamma variables with the mean to be the rate of the parental branch and with the variance to be ν_i .

Clock 3 is implemented for the MSC model only and is unavailable under the MSci model.

Some advice on the choice of `a_vbar` and `b_vbar`. A larger variance ν_i means more serious violation of the molecular clock at locus i . Note that ν_i will be similar to $\bar{\nu}$, especially if α (`a_vi`) is large, and $\bar{\nu}$ has prior mean `a_vbar/b_vbar`. If (`a_vbar` = 10 `b_vbar` = 100), the prior mean will be 0.1. I think for the log-normal model $\nu = 0.5$ means serious violation of the clock while $\nu < 0.1$ means slight violation.

```
heredity = 0      # (0: No variation)
heredity = 1 4 4  # (1: estimate, & a_gamma b_gamma)
heredity = 2 heredity.txt # (2: from file)
```

`heredity = 0` is the default and means that θ is the same for all loci. `heredity = 1` or `2` specifies two models that allow θ to vary among loci, which may be useful for combined analysis of data from autosomal, mitochondrial, X and Y loci. With such mixed data, the effective population sizes are different among loci, so that a heredity multiplier (inheritance scalars, Hey and Nielsen, 2004) should be applied. Other factors such as natural selection may also cause θ to deviate from the neutral expectation. BPP implements two options for this. The first option (`heredity = 1`) is to estimate the multipliers from the data, using a gamma prior with parameters α and β specified by the user. In the example above, a gamma prior $G(4, 4)$, with mean $4/4 = 1$, is specified for the multiplier for each locus. The MCMC should then generate a posterior for the multiplier for each locus. The second option (`heredity = 2`) is for the user to specify the multipliers in a file, and the multipliers will then

be used as fixed constants in the MCMC run. The file simply contains as many numerical values as the number of loci, separated by spaces or line breaks.

Genome	Heredity scalar
Nuclear autosome	1
X chromosome	0.75
Y chromosome	0.25
Mitochondrial	0.25

Note. The effect of the locus-specific mutation rates and the locus-specific heredity multipliers are different. A locus rate is used to multiply all θ s and τ s for the locus, while a heredity multiplier is used to multiply all θ parameters for the locus but not the τ s. Nevertheless, those parameters are quite likely to be strongly correlated, especially when the species tree is small.

Ziheng note on 3 March 2011.

The model of sequence errors is rewritten, so that the old option is unavailable. Use
`sequenceerror = 0` `# (0: default: No error)`

```
finetune = 0: .01 .02 .03 .04 .05 .01 .01 # auto (0 or 1): MCMC step lengths
finetune = 1: .01 .02 .03 .04 .05 .01 .01 # auto (0 or 1): MCMC step lengths
```

This is about the step lengths used in the proposals in the MCMC algorithm. The first value, before the colon, is a switch, with 0 meaning no automatic adjustments by the program and 1 meaning automatic adjustments by the program. Following the colon are the step lengths for the proposals used in the program. If you choose to let the program adjust the step lengths, `burnin` has to be >200 , and then the step lengths specified here will be the initial step lengths, and the program will try to adjust them using the information collected during the burnin step. This option appears to work fine. Some notes about manually adjusting those finetune step lengths are provided below in section 3.2.

```
print = 1 0 0 0 0 * print MCMC samples, locusrates(mu_i,nu_i), heredityscalars, genetrees, locusrateparameters
burnin = 16000
sampfreq = 2
nsample = 200000
```

`print` has several flags (0 for off and 1 for on) that control the printouts during the MCMC. The first flag concerns the MCMC samples to be written into the main sample file (`mcmc.txt`). The second flag is for locus-specific rate parameters (rate μ_i , variance parameter ν_i , and species-tree branch rates for locus i). The third flag is for locus-specific heredity scalars if those are estimated from the data. The fourth flag is for printing locus-specific gene trees. The fifth flag is for printing locus-specific substitution-rate parameters (qmat for the Q matrix, freqs for base frequencies, and alpha for gamma rates for sites).

MCMC samples are taken after the burnin, and in this example, are taken every 2 iterations, with a total of 200,000 samples taken. The total number of MCMC iterations is `burnin + output × nsample`. The resulting file can be large. For analysis A00 (`speciesdelimitation = 0`, `speciestree = 0`), this file is readable from R or Andrew Rambaut's TRACER. For other analyses (A01, A10, and A11), the sample file is not readable by R or TRACER.

`print = -1` means that BPP will bypass the MCMC. Instead it will read the MCMC sample and summarize the results. Thus with `print = 1`, the `mcmc.txt` file will be output, but with `print =`

-1, it will be the input. Make sure that useful files are not overwritten. The `print = -1` option is useful for combining the MCMC samples from multiple runs to produce the posterior summary. Suppose you run the same analysis 3 times in different folders. After a week you are tired so you can kill the jobs. You can then merge the `mcmc.txt` files from the three runs into one file. Note that if the last line of the first file is incomplete, you should delete the incomplete line, and if the first line of the second file is a header line, you should delete that header line. You then run the program to summarize the posterior using the combined sample. Do not combine the MCMC samples from different analyses (with different data files or priors).

3.2.2 Analyses A00, A01, A10, and A11

Here we describe the specifics of the four different analyses.

A00 (`speciesdelimitation = 0`, `speciestree = 0`), for estimation of the parameters of species divergence times and population sizes (τ s and θ s) under the MSC model when the species phylogeny is given (Rannala and Yang, 2003), has been explained in detail above using the control file `ChenLi2001.bpp.ct1` as an example.

If there is only one species, the MSC model will become the single-population coalescent (Kingman, 1982). Take a look at `examples/you2001.bpp.ct1`, which is for analyzing a sample of 61 human sequences from Yu et al. (2001) to estimate the single parameter $\theta = 4N\mu$. There is no need for the `Imap` file, or the need to tag the sequence names in the sequence file (`you2001.txt`): the sequence names are read and then ignored. Multiple loci may be included in the sequence file. There is no need for a species tree, so the block for specifying species names and species tree looks like this:

```
species&tree = 1 H
               100 * max number of sequences
```

In the single-species analysis, the printout on the monitor includes the posterior mean of θ , and posterior means of μ_{MRCA} for the loci, calculated up to that point in the MCMC run. If you have many loci, only the first few μ_{MRCA} are printed on the monitor.

The `mcmc` sample file lists $1 + g + 1$ columns for g loci: θ , μ_{MRCA} for the g loci, and the log likelihood. The μ_{MRCA} are not parameters, but are sometimes of interest as well.

Common features of analyses A01, A10, and A11. Note that A00 is a within-model inference: there is one well-specified model (the MSC model on a fixed species tree) and the parameters in the model are all well defined. The objective of the analysis is to estimate those parameters. In contrast, A01, A10, and A11 are all trans-model inferences. They move between different models, and the main objective is to calculate the posterior probabilities for those models. Each of those models is an instance of the MSC model, but the species delimitation (the number and nature of the species) and/or the species phylogeny may differ between models. In analyses A01, A10, and A11, the inverse-gamma prior specified using `thetaprior` applies to all θ parameters in all models: in other words, there may be thousands of θ parameters across the models, and each one is assigned the same IG prior. Similarly each of those MSC models (if they specify two or more delimited species) may have a parameter τ_0 for the divergence time of the root, and all those τ_0 parameters are assigned the same inverse-gamma prior. Another difference is that the sample file `mcmc.txt` is readable in R or by TRACER for analysis A00, but not for A01, A10, and A11.

A01 means species tree estimation when the assignments and delimitation are fixed.

```
speciesdelimitation = 0 *
speciestree = 1 * NNI/SPR over species trees
speciesmodelprior = 1 * 0: uniform LH; 1:uniform rooted trees; 2: uniformSLH; 3: uniformSRooted
```

This invokes the NNI or SPR algorithm to change the species tree topology, while species delimitation is fixed (so that the number of species and the assignment of individuals to species are fixed).

A10, for species delimitation using a user-specified guide tree (Yang and Rannala, 2010; Rannala and Yang, 2013), is specified using

```
speciesdelimitation = 1 0 2 * speciesdelimitation algorithm0 and finetune(e)
speciesdelimitation = 1 1 2 1 * speciesdelimitation algorithm1 finetune (a m)
speciesmodelprior = 1 * 0: uniform LH; 1:uniform rooted trees
```

The first line specifies rjMCMC algorithm 0, with $\varepsilon = 2$ in equations 3 and 4 of Yang & Rannala (2010). Reasonable values for ε are 1, 2, 5, etc.

The second line specifies rjMCMC algorithm 1, with $\alpha = 2$ and $m = 1$ in equations 6 and 7 of Yang & Rannala (2010). Reasonable values are $\alpha = 1, 1.5, 2$, etc. and $m = 0.5, 1, 2$, etc.

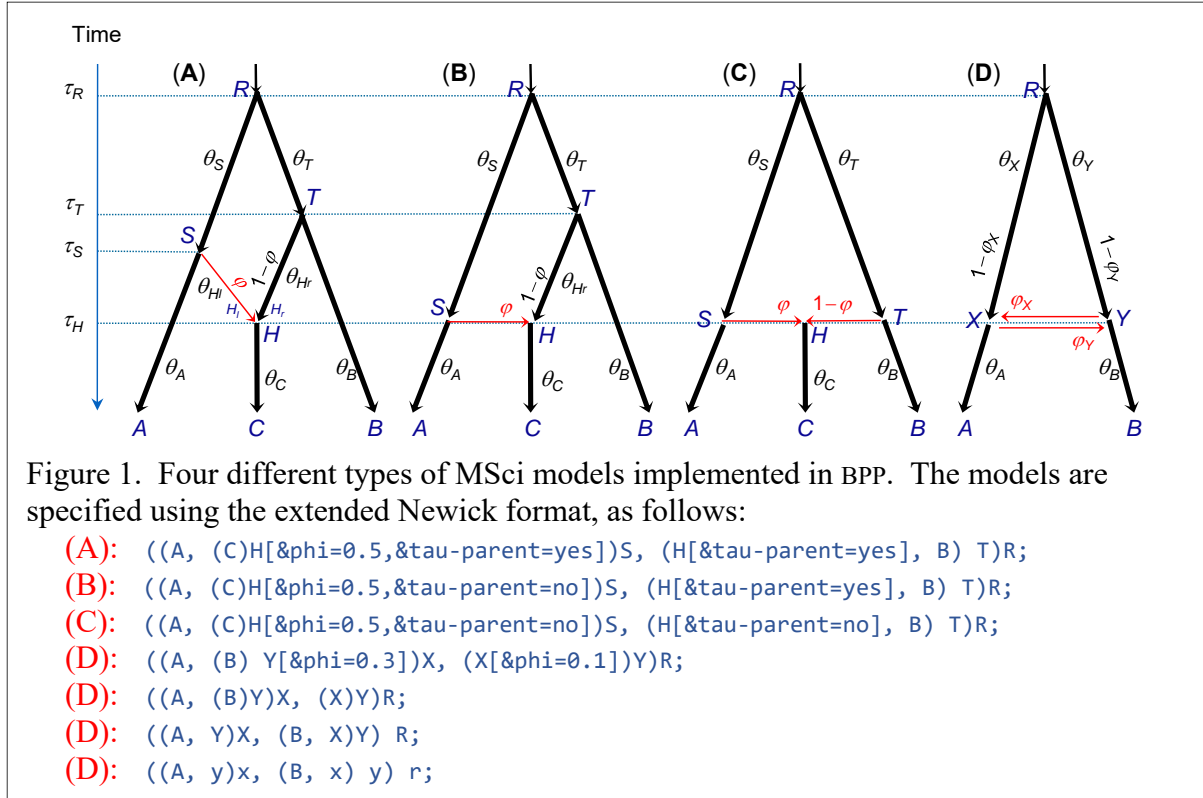
The two algorithms in theory should produce identical results. The variable `speciesmodelprior` specifies Priors 0 and 1. Prior 0 means equal probabilities for labeled histories (which are rooted trees with internal nodes ordered by their age). This is the prior used by Yang & Rannala (2010: equation 2). Prior 1 means equal probabilities for rooted trees. This is now the default. The prior with the user specified probabilities for nodes described by Rannala and Yang (2013) is deleted in the current version. You will have to use version 2.2 for that.

A11, for joint species delimitation and species tree inference or for unguided species delimitation (Yang and Rannala, 2014), is specified as follows.

```
speciesdelimitation = 1 0 2 * rjMCMC speciesdelimitation algorithm0(e)
*speciesdelimitation = 1 1 2 1 * rjMCMC speciesdelimitation algorithm1(a m)
speciestree = 1 * NNI over species trees
speciesmodelprior = 1 * 0: uniform LH; 1:uniform rooted trees; 2: uniformSLH; 3: uniformSRooted
```

In this case, BPP will use the rjMCMC algorithm (either algorithm 0 or algorithm 1 of Yang and Rannala, 2010) to change the species delimitation model and the NNI/SPR move to change the species tree topology.

For A11, `speciesmodelprior` can take the four values 0, 1, 2, 3, which mean Priors 0, 1, 2, 3, respectively. As mentioned above, Prior 1 (which is the default) assigns equal probabilities to the rooted species trees, while Prior 0 means equal probabilities for the labeled histories (rooted trees with the internal nodes ordered by age). Priors 2 and 3 assign equal probabilities for the numbers of species ($1/s$ each for 1, 2, ..., s species given s populations) and then divided up the probability for any specific number of species among the compatible models (of species delimitation and species phylogeny) either uniformly [Prior 3] or in proportion to the labeled histories [Prior 2]. Priors 2 and 3 are mentioned by Yang and Rannala (2014) and implemented by Yang (2015). Prior 3 may be suitable when there are many populations.



3.2.3 The MSci model and the extended Newick format

Since version 4.1, BPP implements the MSci model allowing episodic introgression between species or populations (Flouri et al., 2020a). Currently only the A00 analysis is available. In other words, the user has to specify the number of introgression events, their directions, and the populations involved. The program will then estimate the parameters in the MSci model using MCMC. We hope to implement MCMC moves that change the MSci model later.

Four types of MSci models are implemented, illustrated in figure 1. The MSci model involves three types of parameters: the species divergence or hybridization times (τ s), the population size parameters (θ s), and the introgression probabilities (ϕ s).

The MSci model is specified in the species&tree block using a notation that involves some modifications to the extended Newick notation of Cardona et al. (2008). Here is an example

```
species&tree = 3  A  B  C
                  4  4  4
                ((A,(C)H[tau-parent=no])S,(H[tau-parent=yes],B)T)R;
```

The rules are as follows. ‘(A, B)S’ specifies two branches from S to A and from S to B, while ‘(A)H’ specifies one branch from H to A. Every branch in the tree model is represented once. Each tip species occurs once. Internal nodes for speciation nodes may and may not be labeled, but hybridization nodes must be labeled. The extended Newick notation may not be unique.

Each hybrid node has two parental species. Models A-C are distinguished using a meta-data variable `tau-parent`. The value `yes` means that the parental population exists and has a separate age parameter τ , with an associated θ parameter, while the value `no` means that no such parameter exists. For example, in model (A), the parameters include τ_S , τ_T , θ_{Hl} (for H-left), and θ_{Hr} (for H-right); in model (B), τ_S and θ_{Hl} are not parameters in the model while τ_T and θ_{Hr} are; and finally in model (C), none of those four parameters exists in the model. Model (D) specifies a bidirectional introgression event between species A and B.

At the beginning of the run, BPP prints out a block of output on the screen defining the

internal structure of the species tree with introgressions. You should read this block to identify the parental branch to which the introgression probability (ϕ) is assigned.

Species tree contains hybridization/introgression events.

```
Hybridization events: 1
Bidirectional introgressions: 0
Label      Node  Child1  Child2  Parent
A           0    N/A    N/A     4
B           1    N/A    N/A     6
C           2    N/A    N/A     5
R           3     4     6     N/A
S           4     0     5     3
H           5     2    N/A     4   [tau = 0, phi = 0.189589, prop_tau = 1, has_phi = 1]
phi_H : S -> H
T           6     7     1     3
H           7    N/A    N/A     6   Mirrored hybridization node [Hybrid = H (5)]   [tau = 1,
phi = 0.810411, prop_tau = 0, has_phi = 0]
```

For example the above block of output is for model B (figure 1B in Flouri et al. (2020a). The hybridization node H has two parents, S and T. The output specifies that ϕ_H is assigned onto the horizontal SH branch, while the vertical branch TH has $1 - \phi_H$. The beta prior specified in the control file applies to ϕ_H , not $1 - \phi_H$.

Since version 4.4.0, BPP by default assigns the introgression probability to the horizontal introgression branch in models B and D. In the case of model D, this differs from figure 1D of Flouri et al. (2020a), where ϕ was assigned on the vertical branch. In model A, both parental branches are vertical while in model C both are horizontal. In those two models, BPP assigns the introgression parameter ϕ to one of the two parental branches depending on the Newick notation. You should read the screen output (like the block above) to know the definition of ϕ in the model. For models A, B, and C, you can use the keyword `defphi` to overwrite the program default and assign ϕ to a particular parental branch. Thus if we use the following tree/newick notation

```
((A,(C)H[tau-parent=no])S,(H[tau-parent=yes, defphi],B)T)R;
```

The parameter will be assigned to the vertical branch TH, as can be seen from the following screen output.

```
Label      Node  Child1  Child2  Parent
A           0    N/A    N/A     4
B           1    N/A    N/A     6
C           2    N/A    N/A     5
R           3     4     6     N/A
S           4     0     5     3
H           5     2    N/A     4   [tau = 0, phi = 0.255822, prop_tau = 1, has_phi = 0]
T           6     7     1     3
H           7    N/A    N/A     6   Mirrored hybridization node [Hybrid = H (5)]   [tau = 1,
phi = 0.744178, prop_tau = 0, has_phi = 1]   phi_H : T -> H
```

We suggest that you use `defphi` for models A and C only, and use the default for models B and D so that the introgression probability ϕ is assigned to the horizontal introgression branch.

There is in BPP an `msci-create` facility for generating the extended Newick notation for the MSci model. We prepare a text file (called `msci.txt`, say) in which we specify a binary species tree, with internal nodes labeled, and then list the introgression branches to add onto the binary tree to construct the MSci model. We then run the program as follows and copy the output Newick notation into the control file for inference. You may want to edit the resulting Newick tree to add `defphi` if necessary.

```
bpp --msci-create msci.txt
```

Please see Appendix C for more details.

4. Screen and file outputs

4.1 A00 screen output

First we consider the simple analysis under the MSC model with the species tree fixed (A00: `speciesdelimitation` = 0, `speciestree` = 0). We use this case to explain the acceptance proportions of MCMC moves. The screen outputs for analyses A01, A10, and A11 will have differences, which will be described later.

Make the window wider, with 100 or 120 columns, say, before you run the program. (On Windows, you right-click the window title bar and choose Properties – Layout and change Window Size Width.) Pay attention to screen outputs, especially at the start of the run, to make sure that the control file and sequence data file are read correctly by the program, and that the acceptance proportions are reasonable. Use Ctrl-C to terminate the run, if needed.

Here is an outline of the steps taken by the program. The sample output is from analyzing the example dataset `ChenLi2001.txt`, on a fixed species tree. The differences for the species delimitation analysis are discussed later. The program first prints out the species tree, as well as a population-population table, which describes the descendant-ancestor relationship between populations and which you may ignore. It then defines the θ and τ parameters involved in the model.

The program then reads and processes the sequence data file.

It then generates the initial values for parameters θ s and τ s by using the inverse-gamma priors and the initial gene trees and coalescent times by sampling from the prior. The program prints out the initial θ s and τ s, as well as the initial log MSC gene-tree density `lnpG0` (Rannala and Yang, 2003) and the log sequence likelihood `lnL0` (Felsenstein, 1981), and starts the MCMC.

```
MCMC settings: 2000 burnin, sampling every 2, 20000 samples
Approximating posterior, using sequence data
(Settings: cleandata=0 print=1 saveconP=1 moveinnode=1)

Starting MCMC...

Initial parameters, np = 10.
Genetrees generated from the MSC density.
  0.00093 0.00101 0.00108 0.00100 0.00106 0.00107 0.00098 0.01461 0.00922 0.00805
lnpG0 = 2458.5040 lnL0 = -43947.4120
-4% 0.14 0.00 0.00 0.04 0.45 0.0009 0.0010 0.0011 0.0147 0.0071 0.0058 2330.02 -43766.9113
(nsteps = 5)
Current Pjump: 0.14267 0.00245 0.00057 0.04467 0.46200
Current finetune: 0.01000 0.02000 0.03000 0.04000 0.05000
New finetune: 0.00447 0.00015 0.00030 0.00552 0.08706

-2% 0.15 0.84 0.42 0.02 0.20 0.0007 0.0008 0.0006 0.0141 0.0066 0.0054 2305.87 -43727.7353
(nsteps = 5)
Current Pjump: 0.15443 0.84480 0.42057 0.01533 0.19600
Current finetune: 0.00447 0.00015 0.00030 0.00552 0.08706
New finetune: 0.00217 0.00119 0.00046 0.00026 0.05433

-1% 0.27 0.39 0.30 0.53 0.44 0.0007 0.0008 0.0005 0.0143 0.0063 0.0049 2364.60 -43723.3313
(nsteps = 5)
Current Pjump: 0.25658 0.37186 0.28314 0.50667 0.42800
Current finetune: 0.00217 0.00119 0.00046 0.00026 0.05433
New finetune: 0.00182 0.00155 0.00043 0.00052 0.08488

0% 0.27 0.28 0.27 0.21 0.15 0.0006 0.0008 0.0006 0.0148 0.0062 0.0052 2374.94 -43727.0222 0:05
(nsteps = 5)
Current Pjump: 0.26527 0.28023 0.27257 0.21200 0.14800
Current finetune: 0.00182 0.00155 0.00043 0.00052 0.08488
New finetune: 0.00158 0.00143 0.00038 0.00035 0.03944

5% 0.29 0.30 0.30 0.35 0.51 0.0007 0.0008 0.0006 0.0151 0.0061 0.0051 2379.62 -43728.0652 0:12
10% 0.32 0.32 0.33 0.33 0.52 0.0007 0.0008 0.0006 0.0145 0.0061 0.0052 2399.95 -43723.9478 0:17
15% 0.31 0.31 0.32 0.34 0.52 0.0007 0.0008 0.0006 0.0146 0.0061 0.0052 2274.55 -43725.6273 0:22
20% 0.31 0.31 0.32 0.34 0.53 0.0007 0.0008 0.0005 0.0147 0.0061 0.0052 2409.03 -43725.7590 0:28
```

Then on the same line, it prints out a percentage progress indicator (with negative values for burnin), followed by the acceptance proportions for the MCMC moves (highlighted in red in the sample output), and by the posterior means of the parameters (in this example there are three θ parameters and three τ parameters). The last two numbers before the time used is the log MSC density (for the current gene trees) and log sequence likelihood (average from the past samples). In this example, the program uses four rounds of automatic step-size adjustments, so that the acceptance proportions are close to 0.3 or lie in the interval (0.15, 0.7). See below.

The program will then read and process the file `mcmc.txt` to calculate the mean, min, max, median, and percentiles, and histogram information. This may take quite some time if many samples (say, 1M) are collected in the file. The program also generates a file `FigTree.tre`, which can be read by Andraw Rambaut's `figtree` program to make a publication-quality tree, showing the HPD intervals for divergence times (τ s) as node bars. See the notes at the end of the file.

The `mcmc.txt` sample file generated in analysis A00 can be read in R or TRACER.

4.2 Adjusting step lengths for MCMC moves (finetune)

You can use the automatic adjustment of the `finetune` variable (MCMC step lengths), which appears to be reliable, but make sure that the acceptance proportions are neither too small nor too large. Below are notes for manual adjustments of the step lengths. Often I use automatic adjustments to generate good step lengths and then copy them into the control file, so that the next time the automatic procedure starts with already good step lengths.

Below are some notes about adjusting the step lengths manually (`finetune = 0`).

First note the following line in the control file `ChenLi2001.bpp.ct1`. Here `finetune = 0` means that MCMC step lengths will not be adjusted automatically, and they are used as given.

```
finetune = 0: .01 .02 .03 .04 .05 .01 .01 # auto (0 or 1): MCMC step lengths
```

There are seven `finetune` steplengths here. They are in a fixed order and always read by the program even if the concerned proposal is not used. The first five of them are ε_1 , ε_2 , ε_3 , ε_4 , ε_5 , described in Rannala & Yang (2003). These are the step lengths used in the MCMC proposals that (1) change internal node ages in the gene tree, (2) prune and re-graft nodes in the gene tree, (3) update θ s, (4) update τ s using the rubber-band algorithm, and (5) implements the mixing step. The 6th and 7th are for the proposals that change the locus rates or heredity multipliers and that change the sequencing errors, respectively. If the model assumes the same rate for all loci and does not use heredity multipliers, the 6th proposal step is not used. If the model assumes no sequencing errors, the 7th step is not used. The acceptance proportions for the first five proposals are always printed out on the screen, but those for the 6th and 7th are printed out only if the concerned proposal is used in the model. In the example above, only the first five proposals are used in the algorithm and we will change the step lengths so that the acceptance proportions become close to 30%. If the acceptance proportion is too small (say, <0.10), decrease the corresponding `finetune` parameter. If the acceptance proportion is too large (say, >0.80), increase the `finetune` parameter.

Run the program for a small number of iterations and look at the screen output for the acceptance proportions.

```
InpG0 = 2440.0952 lnL0 = -43946.6769
0% 0.13 0.00 0.00 0.04 0.43 0.0009 0.0009 0.0008 0.0150 0.0067 0.0057 2399.70 -43752.8348 0:05
5% 0.14 0.00 0.00 0.04 0.44 0.0009 0.0009 0.0008 0.0148 0.0064 0.0057 2340.71 -43738.0172 0:09
10% 0.14 0.00 0.00 0.04 0.44 0.0009 0.0009 0.0008 0.0149 0.0064 0.0056 2383.91 -43739.3830 0:14
```

Here the second acceptance proportion, at 0.00, is much too small, which means that the

corresponding step-length (0.02 in the control file) is much too large. Terminate the run (Ctrl-C) and decrease the value in the control file. Then run the program again (use the up ↑ and down ↓ arrow keys to retrieve past commands). Repeat this process a few times until every acceptance proportion is neither too small nor too large. In this example, changing 0.02 to 0.002 brings the acceptance proportion to ~24%, which is slightly too small but already good enough.

Those MCMC proposals are used in all four analyses (A00, A01, A10, A11), so that the description here applies to all of them. Note that the finetune parameters affect the efficiency of the MCMC or how fast one can obtain reliable results. In theory they do not change the results if all runs using different finetune parameters are long enough to generate reliable results.

4.3 A01: species tree estimation

Suppose we use the control file `bpp.4s.ctl` to run analysis A01: species tree estimation with species delimitation and assignment fixed (`speciesdelimitation = 0`, `speciestree = 1`).

The screen output will look like this:

```
5% 0.28 0.28 0.28 0.35 0.29 0.0000 0.0033 0.0138 2403.43 -43721.0651 0:13
10% 0.27 0.27 0.27 0.36 0.30 0.0000 0.0028 0.0142 2304.08 -43723.0804 0:21
15% 0.26 0.27 0.26 0.36 0.30 0.0000 0.0025 0.0143 2308.11 -43723.8564 0:27
^^ Pjump for MCMC moves ^^ PSPR theta tau lnpG E(lnL)
```

Here the five Pjump values are the acceptance proportions for the five conventional MCMC moves, as discussed above. PSPR is the acceptance proportion for the SPR/Nodeslider moves that change the species phylogeny (Rannala and Yang, 2017). The next two numbers are posterior means of θ for the root population (-1 is printed if θ s are integrated out) and τ_0 for the root age. The last two numbers are the log MSC gene-tree density (Rannala and Yang, 2003) and the average log sequence likelihood (Felsenstein, 1981).

The MCMC sample of species trees is collected in the file `mcmc.txt`. Below are two lines from that file. The numbers after : are the branch lengths (τ s), while those after # are θ s. I hope to change the output format to be more consistent with the Newkark format.

```
((H #0.000708: 0.004728, C #0.000740: 0.004728) #0.002814: 0.001190, G #0.000574: 0.005918)
#0.004132: 0.007626, O #0.002036: 0.013544) #0.003835;
((H #0.000708: 0.004728, C #0.000740: 0.004728) #0.002814: 0.001190, G #0.000574: 0.005918)
#0.003740: 0.007351, O #0.002786: 0.013268) #0.003506;
```

The BPP summary of the sample will look like the following.

```
Read tree sample, count trees & splits
tree 20000 (((H, C), G), O);
20001 trees read, 1 distinct trees.

Species in order:
1. H
2. C
3. G
4. O

(A) Best trees in the sample (1 distinct trees in all)
20001 1.00000 1.00000 (O, (G, (H, C)));

(B) Best splits in the sample of trees (2 splits in all)
20001 1.00000 1100
20001 1.00000 1110
```

```
(C) Majority-rule consensus tree
(0, (G, (H, C) #1.000000) #1.000000);

(D) Best tree (or trees from the mastertree file) with support values
(0, (G, (H, C) #1.000000) #1.000000); [P = 1.00000]
```

Section (A) lists the species trees in decreasing order of posterior probabilities. From this you can easily identify the 95% or 99% credibility set of species trees. Section (B) lists the splits (or bipartitions) and their posterior probabilities. The splits here take into account the location of the root, and may be different from the splits for unrooted trees. Section (C) prints the majority-rule consensus tree, with posterior probabilities for nodes.

4.4 A10: species delimitation using rjMCMC

We apply rjMCMC algorithm to the `frogs` data (A10: `speciesdelimitation = 1`, `speciestree = 0`).

```
cd frogs\r1
..\..\bin\bpp ..\A10.bpp.ct1
```

The screen output will look like the following. The species delimitation models that can be generated from the fixed guide tree are listed, together with their prior probabilities calculated by BPP. (As a check, if you use `usedata = 0`, the MCMC should be sampling from this prior distribution.) The species delimitation model is represented using four 0-1 flags for the four interior nodes 6, 7, 8, 9 in the guide tree, with 0 for ‘collapsed’ and 1 for ‘resolved’. Note that the tips in the guide tree are numbered 1, 2, ..., s for s potential species, while the interior (ancestral) nodes are numbered $s + 1$, $s + 2$, ..., $2s - 1$, with $s + 1$ to be the root of the guide tree. The numbering is through a tree-traversal algorithm, fixed by the program. This same order is used to specify the divergence time parameters (τ s), so you can work out the order by looking at the list of nodes in the screen output (look at the “population by population table”, “# species divergence times in the order:”, etc.).

```
Number of species-delimitation models = 5
  delimitation model  1: 000 prior 0.20000
  delimitation model  2: 100 prior 0.20000
  delimitation model  3: 101 prior 0.20000
  delimitation model  4: 110 prior 0.20000
  delimitation model  5: 111 prior 0.20000

[Note: Ancestral nodes in order:  5 KCLH  6 KC  7 LH]

MCMC settings: 8000 burnin, sampling every 2, 100000 samples
Approximating posterior, using sequence data
(Settings: cleandata=1 print=1 saveconP=1 moveinnode=1)

Starting rjMCMC...
PrSplit = 0.500000
rj algorithm 1: new theta from G(a=2.00, m=1.00)

Starting species-delimitation model: 111

root dist = 0.00095

Initial parameters, np = 3.
Genetrees generated from the MSC density.
  0.00108 0.00074 0.00075
lnpG0 = 752.4533 lnL0 = -3402.2670
-3% 0.70 0.13 0.00 0.20 0.36  3 0.0198 111 P[5]=0.8750 -1.0000 0.0007 787.23 -3127.1399
(nsteps = 5)
Current Pjump:      0.69618 0.13189 0.00000 0.19758 0.35650
Current finetune:   5.00000 0.00100 0.00100 0.00100 0.30000
New finetune:      18.97702 0.00041 0.00001 0.00063 0.36913

-2% 0.70 0.30 0.00 0.19 0.27  3 0.0209 111 P[5]=0.8575 -1.0000 0.0007 775.05 -3125.7744
(nsteps = 5)
Current Pjump:      0.69565 0.29861 0.00000 0.18742 0.27050
Current finetune:   18.97702 0.00041 0.00001 0.00063 0.36913
New finetune:      71.87933 0.00041 0.00000 0.00037 0.32779
```



```

-1% 0.70 0.31 0.00 0.20 0.33  3 0.0126 111 P[5]=0.8990 -1.0000 0.0007 732.14 -3126.0675
(nsteps = 5)
Current Pjump:      0.69528 0.30537 0.00000 0.20442 0.33250
Current finetune:  71.87933 0.00041 0.00000 0.00037 0.32779
New finetune:      99.00000 0.00042 0.00000 0.00024 0.37030

 0% 0.69 0.30 0.00 0.23 0.26  3 0.0165 111 P[5]=0.9215 -1.0000 0.0007 723.36 -3125.7568 0:11
(nsteps = 5)
Current Pjump:      0.69354 0.29812 0.00000 0.23367 0.26400
Current finetune:  99.00000 0.00042 0.00000 0.00024 0.37030
New finetune:      99.00000 0.00042 0.00000 0.00018 0.31993

 5% 0.70 0.30 0.00 0.29 0.34  2 0.0260 110 P[5]=0.8731 -1.0000 0.0007 773.34 -3125.8363 0:25
10% 0.70 0.30 0.00 0.30 0.34  3 0.0234 111 P[5]=0.8778 -1.0000 0.0007 745.39 -3125.8263 0:39
15% 0.70 0.30 0.00 0.30 0.34  2 0.0225 110 P[5]=0.8879 -1.0000 0.0007 762.56 -3125.9276 0:53

^^ Pjump for MCMC moves ^^      Prj      P[model 5]      theta      tau      lnPG      E(lnL)

```

The starting species delimitation model is generated by choosing at random one of the models defined by the guide tree or starting tree.

After the chain has started, the five ratios after the % sign are the acceptance proportions for the conventional MCMC moves discussed above.

Next there are three numbers related to the rjMCMC move, highlighted in red above. The rest of the line shows the posterior mean for θ for the root and posterior mean for τ_0 (which exist in all species-tree or species delimitation models), the current log MSC density and average log sequence likelihood

The three numbers related to the rjMCMC move, “2 0.0225 110” in the example, mean that the current model is 110 (using the flags of fig. 1 in Yang and Rannala, 2010), and the rjMCMC move has the acceptance proportion 0.0225. [What does the number 2 mean?] In general the larger this proportion, the more efficient the rjMCMC algorithm is. However there is no optimal acceptance proportion for the rjMCMC move, and a value close to 0 may not necessarily mean a problem. If one model has posterior probability close to 1, the acceptance proportion should be near 0 as well. Thus both poor mixing of the rjMCMC algorithm and extreme posterior model probabilities can cause the acceptance proportion for the rjMCMC to be close to 0. It has been noted that if the rjMCMC algorithm is suffering from poor mixing, different starting species trees often lead to different results.

Next the posterior probability for the best species-tree model (the most frequently visited tree model up to now) is printed. In the example, “P[5]=0.8879” means tree model 5 (111) is the most favoured model, with the posterior at 0.8879.

After the MCMC is finished, the program will summarize the sample. The output looks like the following. The seven delimitation models are listed again, together with their posterior and prior probabilities. The “Guide tree with posterior probability for presence of nodes” can be copied into TreeView.

Summarizing the species-delimitation sample in file mcmc.txt

Number of species-delimitation models = 5

	model	prior	posterior
1	000	0.20000	0.00000
2	100	0.20000	0.00000
3	101	0.20000	0.00038
4	110	0.20000	0.10019
5	111	0.20000	0.89943

[Note: Ancestral nodes in order: 5 KCLH 6 KC 7 LH]

Guide tree with posterior probability for presence of nodes
((K, C) #0.999620, (L, H) #0.899810) #1.000000;

The MCMC sample file mcmc.txt. As the number of parameters changes when the

rjMCMC moves between models, the mcmc sample file may not be very useful, so you can ignore it. Right now the header line is generated using the starting species tree and should be ignored. After the header line, each line of output has the following numbers, separated by TABS: iteration number, the number of parameters, the tree, the sampled parameter values, and lnL.

Gen	np	tree	tau_5KCLH	tau_6KC	tau_7LH	lnL	
2	3	111	0.00079675	0.0007372		0.00065989	-3122.248
4	3	111	0.00074995	0.00067776		0.00059457	-3122.427
...							
786	3	111	0.00037556	0.00010047		2.5634e-05	-3129.184
788	3	111	0.00065693	7.9848e-05		3.6489e-05	-3116.988
790	2	110	0.00044831	5.7933e-05		-3125.574	
792	2	110	0.00060611	0.00029978		-3122.610	
794	2	110	0.00060611	0.00048884		-3122.912	
796	2	110	0.00041675	0.00027494		-3130.812	
798	2	110	0.00034214	0.0002618		-3142.615	
800	3	111	0.00044038	0.00033698		9.8583e-05	-3129.386
802	3	111	0.00060202	0.00047807		7.7594e-05	-3121.725

If you know the unix command grep, you can retrieve the lines for the same tree model to summarize the posterior for parameters in that model.

```
grep "3111" mcmc.txt > result.Tree111.txt
```

In theory this should give you the same posterior as if you run analysis A00 with the species tree fixed at tree 111. In practice I think it is simpler that you edit the Imap file and the control file to run analysis A00.

Notes about running the rjMCMC algorithms.

- The rjMCMC algorithms for species delimitation allow the chain to move from one model to another but can have mixing problems. Make sure you get very similar results from multiple runs using Algorithm0 and Algorithm1 and you get the same results whatever the starting species tree is.
The starting tree is chosen by the program at random and will vary among runs. Make sure that some of your runs are started with the one-species model, some from the fully resolved tree, and some from other trees in between. If you get consistent results among runs with different starting trees and using the two algorithms, you are unlikely to have a convergence or mixing problem.
If you have a computer with multicore, you can run those different combinations or replicates in different folders at the same time.
- In medium-sized or large datasets with multiple loci, we have come across cases where the chain is stuck at the one-species model, or have difficulty moving into the one-species model. The problem is less common when there are only 1 or 2 loci. If you have a similar problem, you may start the analysis with 1 locus, then 2 loci, etc. to observe how the results change (you can do this by changing `nloci` in the control file as there is no need to change the sequence file).
- The program used the burnin to adjust the step lengths for the proposals in the MCMC algorithm. If the rjMCMC stays in species model 0, which does not have any τ parameter, no information is collected during the burnin about the proposals to change τ and the automatically adjusted step length for τ can be very poor.
- You probably need to evaluate the impact of the priors on θ s and τ . See the note about the gamma distribution later in this document.

4.5 A11: joint species delimitation and species tree estimation

Again, we use the frogs example (A11.bpp.ctl) to illustrate joint species delimitation and species tree estimation (A11: speciesdelimitation = 1, speciestree = 1).

The screen output looks like this:

```
Initial parameters, np = 3.
Genetrees generated from the MSC density.

lnpG0 = 711.7015 lnL0 = -3350.8540
-3% 0.70 0.13 0.00 0.22 0.35 4 3 0.0039 0.1455 P(4)=0.9965 -1.0000 0.0007 775.17 -3126.4790
-2% 0.70 0.31 0.00 0.21 0.28 4 3 0.0100 0.1260 P(4)=0.9925 -1.0000 0.0008 745.60 -3125.8524
-1% 0.70 0.30 0.00 0.22 0.35 4 3 0.0039 0.1000 P(4)=0.9980 -1.0000 0.0007 750.41 -3126.2339
0% 0.70 0.30 0.00 0.23 0.25 4 3 0.0077 0.0920 P(4)=0.9845 -1.0000 0.0009 775.04 -3126.2217 0:13
5% 0.70 0.30 0.00 0.24 0.34 4 3 0.0231 0.0967 P(4)=0.9368 -1.0000 0.0007 750.67 -3125.9098 0:30
10% 0.70 0.30 0.00 0.24 0.34 4 3 0.0189 0.0954 P(4)=0.9474 -1.0000 0.0007 732.19 -3125.9363 0:46
^^ Pjump for MCMC moves ^^ S p Prj PSPR P(S=4) theta tau0 lnpG E(lnL)
```

Here S is the number of species in the current model, and p is the number of parameters in the current model. Prj is acceptance proportion for the rj MCMC moves, while $PNNI$ is acceptance proportion for the NNI or SPR moves. $P(4)=0.9474$ means that based on the states sampled, the number of species 4 has the highest posterior probability (compared with 1 species, 2 species, etc.) and the probability is 0.9474.

The next two numbers are posterior means of θ for the root population (-1 is printed if θ s are integrated out) and τ_0 for the root age. These are followed by the log MSC gene-tree density and the average log likelihood.

The MCMC sample file mcmc.txt has lines like the following.

```
((K: 0.000359, C: 0.000359): 0.000271, (H: 0.000034, L: 0.000034): 0.000596); 4
((K: 0.000368, C: 0.000368): 0.000500, (H: 0.000000, L: 0.000000): 0.000868); 3
((K: 0.000378, C: 0.000378): 0.000505, (H: 0.000000, L: 0.000000): 0.000883); 3
```

The numbers after : are branch lengths (τ s). If θ s are estimated, they will be shown after the symbol #. Zero-length branches represent collapsed nodes, meaning that the descendent populations at that node belong to the same species. In the first sample above, there are four distinct species: A, B, C, and D, while in the second, there are three: A, BD, and C.

The summary of the sample by BPP has four sections, as follows.

```
Summarizing the species-tree sample in file mcmc.txt
read tree 100000 ((K, C), (H, L));
(A) List of best models (count postP #species SpeciesTree)
16131 0.16131 0.16131 4 (K C L H) ((L, H), (K, C)); 0011 1100
12736 0.12736 0.28867 4 (K C L H) (K, (C, (L, H))); 0011 0111
9552 0.09552 0.38419 4 (K C L H) (H, (L, (K, C))); 1100 1110
8009 0.08009 0.46428 4 (K C L H) (L, (H, (K, C))); 1100 1101
7901 0.07901 0.54329 4 (K C L H) (C, (K, (L, H))); 0011 1011
6638 0.06638 0.60967 4 (K C L H) (K, (L, (C, H))); 0101 0111
...
(B) 5 species delimitations & their posterior probabilities
95530 0.95530 4 (K C L H)
3638 0.03638 3 (K C LH)
533 0.00533 3 (KH C L)
291 0.00291 3 (K CH L)
8 0.00008 3 (KC L H)
(C) 8 delimited species & their posterior probabilities
99701 0.99701 C
99459 0.99459 K
96362 0.96362 L
95538 0.95538 H
3638 0.03638 LH
533 0.00533 KH
```

291	0.00291	CH
8	0.00008	KC

(D) Posterior probability for # of species

P[1] =	0.0000	prior[1] =	0.23810
P[2] =	0.0000	prior[2] =	0.23810
P[3] =	0.0447	prior[3] =	0.28571
P[4] =	0.9553	prior[4] =	0.23810

Section (A) lists the best models in the decreasing order of the posterior probabilities. Here a model is a full MSC model that species both the species delimitation and species phylogeny. From this section, one can easily construct the 95% or 99% credibility sets of models. This section is further summarized to produce sections B, C, and D. Section (B) gives the posterior probabilities for the top few delimitations. Section (C) lists the delimited species and their posterior probabilities, and section (D) lists the posterior probability for the number of species, together with the prior probabilities calculated by BPP.

5. BPP4-specific features: threading, checkpointing and constraints

Threads. BPP4 can use Single-Instruction-Multiple-Data (SIMD) instruction sets (also called vector instructions) available on modern processors. It automatically detects and uses the best instruction set available so you don’t have to do anything about this. However, you can manually specify and force an instruction set in the control file using the *arch* tag. For instance, *arch=SSE* forces BPP to use the SSE instruction set even if AVX (which has twice the register size of SSE and thus in theory yields twice the speedup) is present on the system. To completely disable vector instruction, one can use the *arch=CPU* option. The available values for the *arch* tag are ‘CPU’, ‘SSE’, ‘AVX’ and ‘AVX2’.

Threads. BPP4 can use multiple cores/threads. The control variable *threads* has the following syntax.

Threads = 8 19 1

This means BPP will use 8 threads, starting from core/thread 19, with increment 1; in other words hardware threads 19-36 will be used. The software threads are pinned onto the hardware threads and they not allowed to migrate. We found that this helps with performance.

Modern clusters based on microprocessors use a common shared-memory configuration called NUMA (for non-uniform memory access). The cluster typically consists of two or four microprocessors interconnected on a local bus to a shared memory on a single motherboard.

You can use commands like *lscpu* to see the machine architecture. For example, on our server *lscpu* gives the following output:

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
Byte Order:	Little Endian
CPU(s):	144
On-line CPU(s) list:	0-143
Thread(s) per core:	2
Core(s) per socket:	18
Socket(s):	4
NUMA node(s):	4
...	

NUMA node0 CPU(s):	0-17,72-89
NUMA node1 CPU(s):	18-35,90-107
NUMA node2 CPU(s):	36-53,108-125
NUMA node3 CPU(s):	54-71,126-143

This means that hardware threads 1-18 are on CPU0, while threads 73-90 are the hyperthreads on CPU0. The specification `Threads = 8 19 1` thus uses 8 threads on CPU1. Our recommendation is that you do some tests yourself using threads = 2 or 4 or 8, and use options that work well for your servers. Adjust the step lengths first. Then use `burnin=0` and a small `nsample` so that the running time is about 1-2 minutes. Change threads and record running time.

Make sure all threads for the same BPP job are on the same CPU. Use `lscpu` to see the machine configuration and `htop` to examine the load on the hardware threads.

Checkpoint. This option instructs BPP to create a checkpoint file (save the progress) after a specified number of MCMC iterations. The MCMC loop in BPP consists of $N = \text{burnin} + \text{sampfreq} * \text{nsample}$ iterations, numbered as 1,2,...,N. There are two formats, where X,Y are whole numbers:

```
checkpoint = X
checkpoint = X Y
```

In the first format, a single checkpoint is created after X MCMC iterations (including burnin).

In the second format, a checkpoint is created after X steps, and then additional checkpoints are created every Y steps. For example, if X=10000 and Y=10000, the first checkpoint is created at MCMC iteration 10000, and further checkpoints at iterations 20000, 30000, etc.

The checkpoint files are named `OUTFILE.Z.chk`, where `OUTFILE` is the label specified for the `outfile` option, and Z is the number of the checkpoint file, starting from 1 and incrementing with every new checkpoint file.

To resume from a checkpoint file, use the `resume` switch, i.e.
`bpp --resume checkpoint-file.chk`

Topological constraints. Topological constraints on species trees can be specified for analysis A01 (with `speciesdelimitation = 0` and `speciestree=1`) and analysis A11 (with `speciesdelimitation = 1` and `speciestree=1`). Note that species trees are always rooted in BPP, whether you use the clock or relaxed clock. The control variable `constraintfile` in the control file has the format

```
constraintfile = constraints.txt
```

Inside the constraint file, three keywords are allowed: `define`, `constraint`, and `outgroup`. The `define` keyword is used to assign a name or alias to a clade, `constraint` defines a clade or subtree, and `outgroup` means that species not on the list (the ingroup species) form a clade.

```
define g1 as (G,H);
constraint = (D,E,F,g1); # identical to constraint (D,E,(G,H));
outgroup = A,B,C;
constraint = (A,B);
```

Both the equal sign (=) and spaces are accepted. In the example above, there are eight species in the dataset: A, B, C, D, E, F, G, and H. Line 1 defines an alias `g1` for clade (G,H), which may be used in subsequent `define` or `constraint` statements. Line 3 specifies A B C as the outgroups; this is interpreted to mean that species not on the list, (that is, D E F G H) form a clade. Since this information is already in line 2, line 3 has no effect in this case.

You can also use the keyword `NOT` (case-insensitive) for `define` to specify the complement of the species on the list. Suppose in the data we have 10 species of ratites (flightless birds), and two outgroup species, chicken and ostrich. You can specify the constraints as follows:

```
define ratites as NOT (chicken, ostrich);
constraint = (chicken, (ostrich, ratites));
```

Note that BPP interprets `outgroup` to mean that all species not on the list form a clade. In theory the `outgroup` keyword is unnecessary because you can always achieve the same thing by using `define` (especially with the `NOT` option) and `constraint`. However `outgroup` may be convenient to use in some situations so we have kept it. If there are four species (A, B, C, D),
`outgroup = B,C,D;`
 achieves nothing since all the 15 rooted species trees are allowed, whereas
`outgroup = C,D;`
 means (A, B) form a clade so that 3 rooted species trees are allowed.

Finally if multiple compatible constraints are specified, BPP merges them into one so that

```
constraint (E,F,G,H);
constraint (F,G,H);
constraint (G,H);
is equivalent to
constraint (E,(F,(G,H)));
```

If any two constraints are in conflict, BPP will abort with an error message. For example the following will cause an error.

```
constraint = (E,F,G,H);
constraint = (G,H);
constraint = (F,G);
```


6. References

- Burgess, R., and Z. Yang. 2008. Estimation of hominoid ancestral population sizes under Bayesian coalescent models incorporating mutation rate variation and sequencing errors. *Mol. Biol. Evol.* 25:1979-1994.
- Cardona, G., F. Rossello, and G. Valiente. 2008. Extended Newick: it is time for a standard representation of phylogenetic networks. *BMC Bioinformatics* 9:532.
- dos Reis, M., T. Zhu, and Z. Yang. 2014. The impact of the rate prior on Bayesian estimation of divergence times with multiple loci. *Syst. Biol.* 63:555-565.
- Felsenstein, J. 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* 17:368-376.
- Flouri, T., X. Jiao, B. Rannala, and Z. Yang. 2018. Species tree inference with BPP using genomic sequences and the multispecies coalescent. *Mol. Biol. Evol.* 35:2585-2593.
- Flouri, T., X. Jiao, B. Rannala, and Z. Yang. 2020a. A Bayesian implementation of the multispecies coalescent model with introgression for phylogenomic analysis. *Mol. Biol. Evol.* 37:1211-1223.
- Flouri, T., B. Rannala, and Z. Yang. 2020b. A tutorial on the use of BPP for species tree estimation and species delimitation. Pages 5.6.1-16 in *Phylogenetics in the Genomic Era* (C. Scornavacca, F. Delsuc, and N. Galtier, eds.). No Commercial Publisher.
- Gronau, I., M. J. Hubisz, B. Gulko, C. G. Danko, and A. Siepel. 2011. Bayesian inference of ancient human demography from individual genome sequences. *Nature Genet.* 43:1031-1034.
- Hey, J., and R. Nielsen. 2004. Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of *Drosophila pseudoobscura* and *D. persimilis*. *Genetics* 167:747-760.
- Hey, J., and R. Nielsen. 2007. Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. *Proc Natl Acad Sci U S A* 104:2785-2790.
- Jiao, X., T. Flouri, and Z. Yang. 2021. Multispecies coalescent and its applications to infer species phylogenies and cross-species gene flow. *Nat. Sci. Rev.* 8:nwab127 (DOI: 10.1093/nsr/nwab127).
- Jukes, T. H., and C. R. Cantor. 1969. Evolution of protein molecules. Pages 21-123 in *Mammalian Protein Metabolism* (H. N. Munro, ed.) Academic Press, New York.
- Kingman, J. F. C. 1982. The coalescent. *Stochastic Process Appl.* 13:235-248.
- Rannala, B., S. V. Edwards, A. D. Leaché, and Z. Yang. 2020. The multispecies coalescent model and species tree inference. Pages 3.3:1-20 in *Phylogenetics in the Genomic Era* (C. Scornavacca, F. Delsuc, and N. Galtier, eds.). No Commercial Publisher.
- Rannala, B., and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics* 164:1645-1656.
- Rannala, B., and Z. Yang. 2007. Inferring speciation times under an episodic molecular clock. *Syst. Biol.* 56:453-466.
- Rannala, B., and Z. Yang. 2013. Improved reversible jump algorithms for Bayesian species delimitation. *Genetics* 194:245-253.
- Rannala, B., and Z. Yang. 2017. Efficient Bayesian species tree inference under the multispecies coalescent. *Syst. Biol.* 66:823-842.
- Shi, C. M., and Z. Yang. 2018. Coalescent-based analyses of genomic sequence data provide a robust resolution of phylogenetic relationships among major groups of gibbons. *Mol. Biol. Evol.* 35:159-179.
- Thawornwattana, Y., D. A. Dalquen, and Z. Yang. 2018. Coalescent analysis of phylogenomic data confidently resolves the species relationships in the *Anopheles gambiae* species complex. *Mol. Biol. Evol.* 35:2512-2527.
- Xu, B., and Z. Yang. 2016. Challenges in species tree estimation under the multispecies coalescent model. *Genetics* 204:1353-1368.
- Yang, Z. 1993. Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Mol. Biol. Evol.* 10:1396-1401.
- Yang, Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J. Mol. Evol.* 39:306-314.
- Yang, Z. 2002. Likelihood and Bayes estimation of ancestral population sizes in Hominoids using data from multiple loci. *Genetics* 162:1811-1823.
- Yang, Z. 2006. *Computational Molecular Evolution*. Oxford University Press, Oxford, UK.

- Yang, Z. 2014. *Molecular Evolution: A Statistical Approach*. Oxford University Press, Oxford, England.
- Yang, Z. 2015. The BPP program for species tree estimation and species delimitation. *Curr. Zool.* 61:854-865.
- Yang, Z., and B. Rannala. 2010. Bayesian species delimitation using multilocus sequence data. *Proc. Natl. Acad. Sci. U.S.A.* 107:9264-9269.
- Yang, Z., and B. Rannala. 2014. Unguided species delimitation using DNA sequence data from multiple loci. *Mol. Biol. Evol.* 31:3125-3135.
- Yu, N., Z. Zhao, Y. X. Fu, N. Sambuughin, M. Ramsay, T. Jenkins, E. Leskinen, L. Patthy, L. B. Jorde, T. Kuromori, et al. 2001. Global patterns of human DNA sequence variation in a 10-kb region on chromosome 1. *Mol. Biol. Evol.* 18:214-222.
- Zhu, T., M. dos Reis, and Z. Yang. 2015. Characterization of the uncertainty of divergence time estimation under relaxed molecular clock models using multiple loci. *Syst. Biol.* 64:267-280.

Appendix A. Definitions of the migration rate in different programs

Definitions of migration rate in the literature are confusing. The following table establishes the correspondence of parameter definitions among a few programs, with DNA sequences in mind. Both IMA and Genetree use the *backward* migration rate, where migration is *backward* because we take the coalescent worldview of running time backwards so that an $i \rightarrow j$ backward migration actually means a migration from populations j to i in the real world. Migrate and 3s use the forward or natural migration rates.

Biological parameters	Symbols used in programs			
	3s & bpp	Migrate 3.2 (Peter Beerli, Dec 2010)	IMa2 (Jody Hey, Dec 2009)	Genetree 9.0 (Bob Griffiths)
The proportion of individuals in population i that are immigrants from population j	m_{ji}	$m_{j \rightarrow i}$ or m_{ji}	$M_{i \rightarrow j}$ (In IMA2, migration is backward.)	
The expected number of immigrant individuals in population i (from population j) per generation.	$M_{ji} = N_i m_{ji}$	$M_{ji} \Theta / 4$ (Θ in Migrate is θ in 3s, both based on the per-site mutation rate)	$\theta_i m_{i \rightarrow j} / 4$	-m Allows the specification of a backward migration rate matrix. If there are s subpopulations this matrix has the dimensions s by s .
Parameters used in the program, expressed using parameters in 3s		Parameter M_{ji} in Migrate is $4M_{ji}/\theta_i$ in 3s.	Parameter $m_{i \rightarrow j}$ in IMA2 is $4M_{ji}/\theta_i$ in 3s.	

Note.— We assume a diploid autosomal locus with $\theta = 4N\mu$, where N is the (effective) population size and μ is the mutation rate per site per generation, so that θ is the expected proportion of differences between two DNA sequences sampled at random from the population.

The term “backward migration” has another interpretation, that is, immigration as opposed to emigration; in this case time still runs forward. In many population genetics models, the proportion of individuals that emigrate out of the current population (that is, the forward migration rate) is typically hard to deal with, as emigrants may die trying to cross the ocean or they may move to unsampled locations, etc. It is often easier to deal with the proportion of individuals in the current population that are immigrants, or the backward migration rate. We do not use this interpretation and will refer to the proportion of individuals in the current population that are immigrants simply as the migration rate m , and reserve the term “backward migration” for the coalescent-worldview interpretation in which time runs backwards.

Appendix B. Marginal likelihood calculation using BFdriver 4.0

Ziheng Yang, 1 December 2016

The C program BFdriver generates control files and job subscription scripts for running MCMC (using BPP or MCMCtree) to calculate the marginal likelihood (or the Bayes factor), as described in Rannala and Yang (2016, Syst Biol).

The program takes a control file you provide (such as bpp.ctl) and generates $K = 16$ control files with different beta values, which are used to run bpp to sample from the different power posterior distributions. The program also generates job submission scripts and submit the jobs using qsub. All generated control files and output files are in the same current directory. The frogs dataset in the bpp release (Yang 2015 Curr Zool) is used as the example.

You need the following: a linux system with SUN grid engine managing job submission (including commands such as qsub, qstat, qdel, etc.), and a C compiler. If you don't have this job submission system, you can use BFdriver to generate the control files and run the MCMC jobs from the command line.

(a) Compiling and running BFdriver

```
cc -o BFdriver -O3 BFdriver.c tools.c -lm
BFdriver <controlfilename> <npoints> <scriptname.sh>
BFdriver A00.ctl 16 tmp.sh
```

You may need to edit the following two lines inside BFdriver.c, and if you do, remember to compile the program after editing. Here bpp is assumed to be on your search path. You can use a full path for the executable program, such as /home/gooduser/bin/bpp3.3. Also the second line is for submitting the jobs using qsub. Here the limits are set to 4G of RAM and 360 hours of running time. Check those values if necessary (and recompile).

```
fprintf(fcommand, "      echo \"bpp %s.b$I.ctl > log.b$I.txt\" > %s\n", ctrlf, scriptf);
fprintf(fcommand, "      qsub -S /bin/bash -l h_vmem=4G -l tmem=4G -l h_rt=360:0:0 -cwd %s\n", scriptf);
```

(b) Running the program

Create a folder inside frogs/, say bf1:

```
mkdir frogs/bf1
cd frogs/bf1
```

Prepare a control file (A00.ctl, say) for the A00 analysis in the folder. Check that it works. This should have a fixed species tree, which is ((K, C), (L, H)). Species delimitation and species tree estimation should be turned off. The control file specifies the priors for theta, tau, and also specifies burnin, nsample, sampfreq etc. Run bpp to confirm that the control file works. Then run BFdriver as follows:

```
BFdriver A00.ctl 16 tree1.sh
```

Here A00.ctl is the control file we have prepared. $K = 16$ is the number of points in the Gauss-Legendre quadrature algorithm for numerical integration. You can use 8 for testing, and 16 or 32 for real calculation. tree1.sh is the temporary script file for job submission using qsub. The BFdriver command does a few things. First it reads the control file specified (A00.ctl) and creates 16 control files with names like A00.b01.ctl, ..., A00.b16.ctl. Each of those control files has the same content as A00.ctl except that one extra line is

inserted at the beginning, like the following

```
BayesFactorBeta = 0.122298 * w=0.124629.ct1
```

This specifies the beta value when the control file is used to run bpp.

Second BFdriver creates a file named betaweights.txt, which lists the beta values and Gauss-Legendre weights. I have copied those values into an excel file in frogs/BFdriver.frogs.xls.

Third BFdriver creates a file named commands, which has the bash shell scripts for submitting the 16 jobs using qsub. You use the following to submit the jobs.

```
source commands
```

You can look at the content of tree1.sh to see the script for the last job:

```
more tree1.sh
```

which should have the content like the following:

```
bpp --cfile bpp.b16.ct1 > log.b16.txt
```

You can use qstat to check the status of the 16 jobs you have submitted. When the jobs are running, they generate output files in the current folder, such as mcmc.b01.txt, out.b01.txt, and log.b01.txt (which logs the screen output). After all jobs are finished, you can use grep to extract the line with "BFbeta" from screen log (this command is at the bottom of the file commands).

```
grep BFbeta log.b*.txt
```

Then copy the ElnfX values into the excel file, and estimate the logarithm of the marginal likelihood by summing (weights * ElnfX / 2) over the 16 points.

This gives the log marginal likelihood to be [-3185.93](#).

(c) Exercise & results

Duplicate the calculation for the alternative species tree (tree2) in the folder /bf2. Change the species tree topology to (((L, H), C), K), and use tree2.sh as the temporary job script file. Everything else should be the same as described above.

My runs gave the log marginal likelihood for tree 2 to be [-3186.14](#). The ratio of the posterior probabilities for the two trees is then estimated to be $P_1/P_2 = \exp\{-3185.93 - (-3186.14)\} = \exp\{0.21\} = 1.24$. With BPP4.0 and the inverse gamma priors on θ s and τ s, the MCMC runs (A01) gave the posterior probabilities for trees 1 and 2 as [0.167](#) and [0.137](#), with the ratio [1.22](#). The MCMC runs and the marginal likelihood calculations seem to agree with each other. (Note that with BPP3 and the gamma priors on θ s and τ s, the posteriors are 0.16 and 0.13 with the ratio 1.2 (Yang 2015 fig. 4).

(d) Common errors and problems

Check the bpp control file by running bpp at the command line before submitting the jobs. Make sure that the bpp program is on your path or use a full path. You may have to edit the source file BFdriver.c and recompile.

Appendix C. The simulation option

The simulation option of BPP 4 is the MCCOAL program in version 3.4 and earlier. This can be used to simulate gene trees and sequence alignments at multiple loci under the MSC (Rannala and Yang, 2003) and MSci (Flouri et al., 2020a) models. While the inference program `bpp` implements the JC model only, the simulation program allows GTR+G (and its special cases) and allows among-loci heterogeneity in the process of sequence evolution. The different loci can have different exchangeability parameters (a, b, c, d, e, f) and base compositions in the GTR model, different overall evolutionary rates, different extent of among-site rate variation (reflected in the alpha parameter for the gamma model), and different among-branch rate drifts (violation of the clock). See Shi and Yang (2018) for examples of such simulations. The simulation option also includes a continuous migration model, with a user-specified migration rate matrix between species/populations.

Look at the README.txt file for compiling the program. To run the program, type one of the following. The default control file name is `MCcoal.ct1`. Always look at the screen output to confirm that the program reads the control file correctly.

```
bpp --simulate MCcoal.ct1
bpp --simulate MCcoalMigration.ct1
```

C1 Control-file options for simulating under the MSC model

Here is a sample control file.

```
seed = 12345
seqfile = MySeq.txt 0 * comment out this line if you don't want seqs
treefile = MyTree.tre * comment out this line if you don't want trees
Imapfile = MyImap.txt
* concatfile = concatenatedfile.txt * concatenated alignment
modelparafile = modelparas.txt * comment out this line if you don't want seqs

species&tree = 4 A B C D
                3 2 1 1
((A #0.01, B #0.01) :0.01 #.01, (C, D) :0.011 #0.01) : 0.012 #0.01;
phase = 1 1 1 0 * 0: do not phase (fully phased seqs), 1: diploid unphased seqs

loci&length = 100 1000 * number of loci & number of sites at each locus
* locusrate = 0 (default: same rate for all loci)
* locusrate = mu_bar a_mui prior
* clock = 1 (default: strict clock)
* clock = 2 v_bar a_vi prior dist (independent-rates)
* clock = 3 v_bar a_vi prior dist (correlated-rates)

model = 7 * model: 0:JC69, 7:REV (GTR)
Qrates = 0 10 5 5 5 5 10 * 1: fixed; 0: dirichlet, for TC TA TG CA CG AG
basefreqs = 0 10 10 10 10 10 * 0: random, Dirichlet(aT,aC,aA,aG), for base frequencies
alpha_siterate = 0 100 20 5 * G(a, b) for alpha for sites & K for discrete gamma
```

seed. Use `-1` to simulate different datasets every time the program is run. If you use a positive integer the data will stay exactly the same, which is a bad idea.

seqfile, treefile, Imapfile, concatfile, modelparafile. These are names of files to be generated. If you want to simulate gene trees and not sequence files, you can comment out the line for the `seqfile`. `concatfile` will have the sequence alignment concatenated across loci. The sequence files can become very large. `modelparafile` records the parameter values for the GTR model, as well as the locus rate and the rates for nodes on the species tree. There is one line of output per locus.

By default `MCcoal` prints out the sequence alignment at each locus, but if you use file

format 1 (seqfile = MySeq.txt 1), the program will print out site pattern counts instead. The file may then be smaller. This format is readable by bpp and 3s, but not by other programs.

phase. This variable is used in the same way as in the inference (data analysis) mode. Each species is assigned a switch/flag, with 0 meaning fully phased (haplotype) sequences and 1 for unphased diploid sequences. With the specifications in the example (sampling configuration 3 2 1 1 and phase = 1 1 1 0), the program will simulate 6, 4, 2, 1 sequences for A, B, C, D, respectively, and then combine every two sequences from A, B, C into one single diploid sequence with heterozygote sites represented using ambiguity characters (YRMKSW). If you use the same random number seed, the diploid data generated using the sampling configuration (3 2 1 1) with phase = 1 1 1 0 should match the fully resolved sequences generated using the sampling configuration (6 4 2 1) with phase = 0 0 0 0. The program also prints out the fully resolved sequences in a file with the suffix _full (or MySeq_full.txt in the example).

species&tree. This block specifies the number and names of species, the species tree, and the parameters under the multispecies coalescent model, including the species divergence times (τ) and population size parameters ($\theta = 4N\mu$). For the example above, 100 loci, each of 1000 sites, will be simulated, on the species tree ((A, B), (C, D)), with 3, 2, 1, 1 sequences for A, B, C, D, so that there are 7 sequences at each locus. The divergence time parameters (τ s) are after ‘.’ in the tree, while the population size parameters (θ s) are after ‘#’. Thus we have 0.01 for all θ s, and $\tau_{ABCD} = 0.012$, $\tau_{AB} = 0.01$, and $\tau_{CD} = 0.011$. We need θ_A and θ_B because 2 or more sequences are sampled from species A and B. Parameters θ_C and θ_D are unnecessary in this case as only one sequence is sampled from C and D: if you specify them, they will be ignored by the program. Note that both θ and τ are measured by the expected number of mutations per site, so that $\theta = 0.01$ means that two sequences sampled from the population are 1% different. For reference, the estimate for the human species is $\theta \approx 0.0006$.

model. This can take two possible values: 0 for JC69 and 7 for GTR. The next few lines are relevant for the GTR model.

Qrates specifies the exchangeability parameters in the GTR model (a, b, c, d, e, f for TC, TA, TG, CA, CG, and AG, in Yang 1994 JME 39:105-111). There are two options, either to have those rates fixed for all loci or to have them sampled at random. The first has the following format (note that the first 0-1 number is a switch):

```
Qrates = 1 2 1 1 1 1 2 * 1: fixed; for TC TA TG CA CG AG
```

The input is one integer value (0 or 1) which acts as a switch followed by six real numbers. The line above fixes the rates at $a = 2$, $b = c = d = e = 1$ and $f = 2$, so that the transition rate is twice as high as the transversion rate, and the model corresponds to K80 or HKY. Note that only the relative rates matter, because the rate matrix (Q) is scaled so that the average rate (over the base frequencies) is 1 and branch lengths are measured in the expected number of mutations/substitutions per site. Nevertheless the program expects six rate parameters in the input here.

```
Qrates = 0 10 5 5 5 5 10 * 0: dirichlet, for TC TA TG CA CG AG
```

The second option, above, is to sample $a b c d e$ from the Dirichlet distribution for every locus. The above specifies Dir(10, 5, 5, 5, 5, 10), with parameters $\alpha_a = 10$, $\alpha_b = \alpha_c = \alpha_d = \alpha_e = 10$, and $\alpha_f = 10$. The rates generated from the Dirichlet sum to 1, and they are rescaled. Note that larger values for those parameters mean less variance, while the mean for a , say, is given by α_a/α with $\alpha = \alpha_a + \alpha_b + \alpha_c + \alpha_d + \alpha_e + \alpha_f$. The specification here gives on average a transition/transversion rate ratio of 2 (if the base frequencies are fixed at $\frac{1}{4}$ each), but it varies among loci.

basefreqs. The base frequency parameters ($\pi_T, \pi_C, \pi_A, \pi_G$) can also be fixed or sampled from the Dirichlet distribution, like the Qrates. The two options are as follows.

```
basefreqs = 1 0.15 0.35 0.15 0.35 * 1: fixed; Base frequencies are for TCAG.
basefreqs = 0 10 10 10 10 * 0: random, Dirichlet(aT, aC, aA, aG), for base frequencies
```

alpha_siterate. The gamma shape parameter (α) for rate variation among sites of the same locus can be fixed or sampled from a gamma distribution. The possible options are as follows. When alpha is fixed, the same value is used for all loci. Otherwise different α s are used for different loci. Note that given alpha, the rates for sites at the same locus have a $G(\alpha, \alpha)$ distribution with mean 1 (Yang, 1993, 1994). The fourth option below will allow the program to sample α from $G(100, 20)$, with mean 5, for each locus and then use $G(\alpha, \alpha)$ to describe the among-site rate variation for the locus.

```
alpha_siterate = 1 0      * 1: alpha fixed at 0(inf), one rate for all sites at the locus
alpha_siterate = 1 5.6 0  * 1: alpha fixed at 5.6, K = 0(inf) for continuous gamma
alpha_siterate = 1 5.6 5  * 1: alpha fixed at 5.6, K = 5 categories for discrete gamma
alpha_siterate = 0 100 20 5 * 0: alpha sampled from G(100, 20), with mean 5, K = 5 for discrete gamma
```

locusrate. The variables `locusrate` and `clock` are used to specify relaxed clock models, with rate variation among branches and among loci. These are the same models as implemented in the inference program. Please see notes in section 3.2.1. for details.

`locusrate` specifies variable rates for different loci. The default is 0, meaning that all loci have the same rate. To specify variable rates for loci, use the following format.

```
locusrate = 0 (default)
locusrate = mu_bar a_mui prior
locusrate = mu_bar a_mui
locusrate = 1.0 5.0 dir
```

Here `mu_bar` ($\bar{\mu}$) is the mean rate across loci.

If `prior = dir` (gamma-Dirichlet), the total rate $L\bar{\mu}$ is partitioned to generate the locus-rates (μ_i) according to the Dirichlet distribution with concentration parameter α (`a_mui`). Large α (10 or 100, say) means that the rates are similar among loci, while small values (1 or 0.5, say) mean that rates are highly variable among loci. If `prior = iid` (conditional i.i.d.), the locus rates (μ_i) are i.i.d. given the mean rate ($\bar{\mu}$):

$$\mu_i \sim \text{Gamma}(a_{\text{mui}}, a_{\text{mui}}/\bar{\mu})$$

clock. This is used to specify strict-clock or relaxed-clock models. The default is `clock = 1` for strict clock, while `clock = 2` is the independents-rates model, and `clock = 3` is the correlated-rates model (which is implemented for the MSC model with no introgression and is unavailable for the MSci model).

```
clock = 1 (default: strict clock)
clock = 2 v_bar a_vi prior dist * independent-rates model
clock = 2 0.1 10 dir G * gamma-dirichlet for loci & gamma for branches
clock = 2 0.1 10 iid LN * iid for loci & log-normal for branches
clock = 3 v_bar a_vi prior dist * correlated-rates model
clock = 3 0.1 10 iid G * iid for loci & gamma for branches
clock = 3 0.1 10 iid LN * iid for loci & log-normal for branches
```

The specification (`clock = 2 0.1 10 dir G`) means the following. First the average variance parameter is $\bar{v} = 0.1$. Then the total $L\bar{v}$ is partitioned into v_i (for $i = 1, 2, \dots, L$) according to the Dirichlet distribution with concentration parameter $\alpha = 10$ (`a_vi`).

In contrast, the specification (`clock = 2 0.1 10 iid LN`) means that the variance parameter for locus i is generated as $v_i | \bar{v} \sim \text{Gamma}(\alpha, \alpha/\bar{v})$ with shape parameter $\alpha = 10$ (`a_vi`) and mean $\bar{v} = 0.1$.

In both prior models (`prior=dir` and `iid`), α (`a_vi`) is inversely related to the variance in v_i among loci: use small values of α (2, 1, or 0.5) if you believe that the clock nearly holds at some loci but is seriously violated at others, and large values (such as 10 or 100) if the clock

is violated in the same extent at different loci.

clock = 2. Given the locus rate μ_i (specified using the **locusrate** variable) and the variance parameter ν_i (specified using the **clock** variable) for locus i , the rates for species-tree branches are specified as follows. If **dist = G** (for gamma), the rate for branch j at locus i has the following gamma distribution with mean μ_i and variance ν_i :

$$r_{ij} \mid \mu_i, \nu_i \sim \text{Gamma}(\mu_i^2 / \nu_i, \mu_i / \nu_i).$$

If **dist = LN** (for log-normal), the rate for branch j at locus i has the following the log-normal distribution

$$r_{ij} \mid \mu_i, \nu_i \sim \text{LN}(\mu_i, \nu_i).$$

clock = 3. The correlated-rates model is specified similarly, and for the MSC model only. Given the locus rate μ_i (specified using the **locusrate** variable) and the variance parameter ν_i (specified using the **clock** variable) for locus i , the rates for species-tree branches are specified recursively, starting from the root towards the tips.

If **dist = G** (for gamma), the rate for branch j at locus i has the following gamma distribution with shape parameter $\alpha = \mu_i^2 / \nu_i$ and with the mean to be the rate for the parental branch of j or $r_{i,p(j)}$:

$$r_{ij} \mid \mu_i, \nu_i, r_{i,p(j)} \sim \text{Gamma}\left(\frac{\mu_i^2}{\nu_i}, \frac{\mu_i^2}{\nu_i r_{i,p(j)}}\right).$$

If **dist = LN** (for log-normal), the rate for branch j at locus i has the log-normal distribution with the mean to be the parental rate and variance parameter ν_j :

$$r_{ij} \mid \nu_i, r_{i,p(j)} \sim \text{LN}(r_{i,p(j)}, \nu_i).$$

After the rates for branches (populations) on the species tree are generated for the locus, they are used to calculate the gene-tree branch lengths. Gene-tree branches residing in the same species/populations have the same rate (the rate of that species at the locus), while those residing in different species may have different rates. A branch on a gene tree may have segments residing in different species/populations, and the length of the branch (measured by the expected number of mutations/substitutions per site) is calculated by adding up those segments.

C2 Simulating with MSC-M (or IM) model

```
seed = -1

treefile = mytree.tre
Imapfile = myimap.txt
seqfile = mydata.txt
*concatfile = concat.txt
modelparafile = modelparas.txt

#phase = 0 0 0
species&tree = 3  A  B  C
                  4  4  4
                  ((A #0.015, B #0.025) S #0.015 :0.01, C #0.025) :0.02 #0.025;

loci&length = 2000 500

model = 0      * JC

migration = 8
  A B 0.12
  B A 0.21
  B C 0.23
  C B 0.32
  C A 0.31
  A C 0.13
  S C 0.1
  C S 0.2
```

The above control file (MCcoal.im-3s-saturated.ct1) is used for simulating gene trees and multilocus sequence alignments under the MSC-migration model (also known as the migration model). The internal node in the three species tree ((A,B),C) is labeled S, and this is used below to specify migrations in the model. In this case, the model is saturated with 8 migration events, 6 involving contemporary species (A, B, C) and 2 between S (the ancestor of A and B) and C.

The migration rate from populations i to j is defined as $M_{ij} = N_j m_{ij}$, where m_{ij} is the migration rate per generation in population j from population i , or the proportion of individuals in population j that are immigrants from population i . In the example above, $M_{C \rightarrow S} = 0.2$, which means that on average 0.2 individuals are immigrants from population C to population S. Time runs forward in our definition of parameters here.

This control file format is used in version 4.?.? and later of BPP, and is different from the old format, in which the labels and order for populations are fixed by the program.

Appendix D. The msci-create facility for generating the Newick notation of MSci models

Summary. From BPP release 4.4.0, we have changed the definition of introgression probability (φ) so that it is assigned to the horizontal (introgression) branch. This note illustrates the msci-create facility in BPP, which generates the extended Newick notation for the MSci model from a data file which contains a binary species tree with introgression events specified using source and target branches on the tree.

The `--msci-create` option was implemented since BPP 4.2.1 (<https://github.com/bpp/bpp/issues/114>). The user prepares a data file (named `msci.txt` in this note), which includes a binary species tree in the Newick format and specifies the introgression events by identifying the source and target branches involved in the introgression. Run BPP using the following command. This generates the Newick notation for the MSci model, which can be copied into the BPP control file.

```
bpp --msci-create msci.txt
```

The data file `msci.txt` uses four commands: `tree`, `define`, `hybridization`, and `bidirection`.

- `tree` defines the binary species tree, in Newick format.
- `define` defines an ancestral species or internal node label, as the most recent common ancestor of the tip species. Alternatively you can label the internal nodes on the Newick tree.
- `hybridization` defines a hybridization/introgression event by specifying the source and target branches (which represent the source and target populations involved).
- `bidirection` defines a bidirectional introgression (BDI) event.

One may think of the binary species tree as describing the history of species divergences and add introgression events onto it as new (horizontal) branches. The introgression probability is assigned to the newly created introgression branch.

Figure 1 illustrates the specifications of models A, B, C, and D of Flouri et al. (2020). In models A, B1, and C, the introgression event is from the source branch RA to the target branch TC, with a new introgression branch SH created. The introgression probability ($\varphi = 0.1$) is assigned to the newly added introgression branch (SH) while $1 - \varphi$ is assigned to the other parent branch (TH). Models A, B1, and C are distinguished by using the keyword `tau` (which means the same as `tau-parent` in the Newick notation of the MSci model).

Model A assumes that the two parental species S and T of the hybridization node H have distinct ages from H, with $\tau_S > \tau_H$ and $\tau_T > \tau_H$. Thus we have “`hybridization R A, T C as S H tau=yes, yes phi=0.10`”: the first `yes` means that parental node S on the source branch RA has a distinct tau from node H and that the second `yes` means that parental node T on the target branch TC has a distinct tau from node H. Note that there are often multiple ways of specifying the same MSci model and here you can specify model A by starting with the binary tree `((A, C), B)` and adding branch TH as a introgression event. Model A might appear to be nonsensical biologically since only contemporary species can exchange migrants. Nevertheless, model A may be used to represent introgressions from a ghost species not sampled in the sequence dataset. Suppose at time τ_S a speciation event generated two species SA and SH, and at τ_H , species SH contributed migrants into species THC, but species SH has since become extinct or is otherwise not sampled in the data. This scenario matches model A.

Model B1 assumes $\tau_S = \tau_H$ and $\tau_T > \tau_H$ so there is no new tau for parent S of the hybridization node H, and there is a new tau for parent T. Thus we have “`tau=no, yes`”. Model B2 works similarly. In model C, none of parents S and T has a distinct age from H, so we have “`tau=no, no`”. One interpretation is that species C is a hybrid species.

In model D for the bidirectional introgression (BDI) model, there is no distinction of source and target branches. We specify two phi values, assigned to the introgression (horizontal) branches: $\varphi_X = 0.1$ for node X (or into node X) and $\varphi_Y = 0.2$ for node Y (fig. 1).

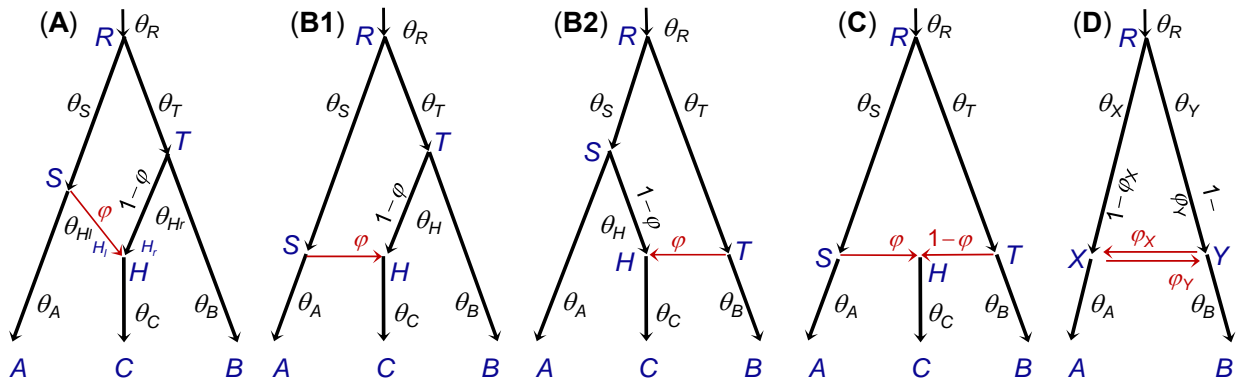


Figure 1. Models A, B, C, and D from Flouri et al. (2020). We have changed the definition of introgression probability ϕ in models B2 and D. In Flouri et al. (2020) and in BPP 4.3.8 or earlier, ϕ is assigned to the vertical branch. From BPP 4.4.0, it is assigned to the horizontal (introgression) branch.

Example msci.txt files for generating models A, B, C, and D as well as the generated Newick notation are as follows. See text for more explanations.

#model A, version 1

```
tree (A,(B,C));
define T as B,C
define R as A,B
hybridization R A, T C as S H tau=yes, yes phi=0.10
```

#Model A, version 2

```
tree (A,(B,C)T)R;
hybridization R A, T C as S H tau=yes, yes phi=0.10
```

#The generated Newick notation for model A is
 ((H[phi=0.1,tau-parent=yes],A)S, (B,(C)H[phi=0.9,tau-parent=yes])T)R;

#Model B1

```
tree (A,(B,C)T)R;
hybridization R A, T C as S H tau=no, yes phi=0.10
```

#The generated Newick notation for model B1 is
 ((H[phi=0.1,tau-parent=no],A)S, (B,(C)H[phi=0.9,tau-parent=yes])T)R;

#Model B2

```
tree ((A,C)S,B)R;
hybridization R B, S C as T H tau=no, yes phi=0.10
```

#The generated Newick notation for model B2 is
 ((A,(C)H[phi=0.9,tau-parent=yes])S, (H[phi=0.1,tau-parent=no],B)T)R;

#Model C

```
tree (A,(B,C)T)R;
hybridization R A, T C as S H tau=no, no phi=0.40
```

#The generated the Newick notation for model C is
 ((H[phi=0.4,tau-parent=no],A)S, (B,(C)H[phi=0.6,tau-parent=no])T)R;

#model D

```
tree (A,B)R;
bidirection A R, B R as X Y phi=0.1,0.2
```

#This generates the Newick notation for model D as

In the next example, we have two BDI events between species A and B. Note that the first bidirection command creates new nodes X and Y, which are used in the next bidirection command to specify the BDI event at nodes Z and W. Introgression probabilities are specified as $\varphi_X = 0.1$, $\varphi_Y = 0.2$, $\varphi_Z = 0.15$, and $\varphi_W = 0.25$, as defined in figure 2.

```
tree (A,B)R;
bidirection A R, B R as X Y phi=0.10,0.20
bidirection X R, Y R as Z W phi=0.15,0.25
```

```
#This generates the Newick notation
(((A,Y[&phi=0.2])X, W[&phi=0.25])Z, ((B,X[&phi=0.1])Y,
Z[&phi=0.15])W)R;
```

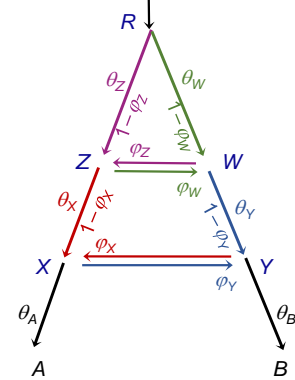


Figure 2. Two BDI events between A and B.

The following example is from the github/wiki site. There is a hybridization event involving the same source and target branches (VA): species V split into two species at time τ_P , which merged into one species at time τ_Q .

```
tree (((A,B)V,C)X,(D,E)W)R;
hybridization R W,X C as Y H tau=yes, yes phi=0.4
bidirection X H, Y H as S T phi=0.2,0.3
hybridization V A as P Q phi=0.333
```

```
# The hybridization event from edge R-W to edge X-C,
creates nodes Y and H. The introgression probability
phi_H = 0.4 is assigned to the introgression edge Y-H.
The BDI event between H-X and H-Y creates new nodes S
and T, with phi_S = 0.2 into node S and phi_T = 0.3 into node T.
Finally the hybridization event at node Q merges the two
branches A V into a circle. The commands create the
following MSci model:
```

```
(((((A)P[&phi=0.667,tau-parent=yes],P[&phi=0.333,tau-
parent=yes])Q,B)V, ((C)H[&phi=0.6, tau-
parent=yes],T[&phi=0.3])S)X, ((H[&phi=0.4,tau-
parent=yes],S[&phi=0.2])T, (D,E)W)Y)R;
```

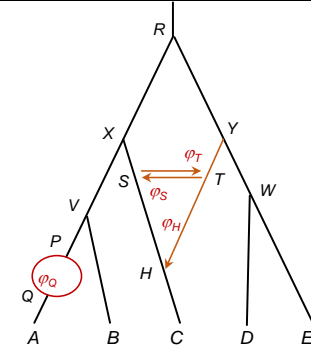


Figure 3. A big MSci model.

Further notes about the data file msci.txt

- The species tree has to be binary and should not include redundant pairs of parentheses. For example $((A, B))$ is not correct while (A, B) is.
- Internal node labels can be defined either in the binary species tree as node labels or by using the define command. The hybridization and bidirection commands may create new nodes, which can be used later to specify other introgression events. As usual, node labels as well as species names are case-sensitive.
- A branch is specified using the nodes at its two ends. Both mother-daughter and daughter-mother specifications are accepted; for example, branch R A can be specified as branch A R.
- The symbol '#' indicates comments: the text after '#' on the same line is ignored.