PHYBASE: an R package for phylogenetic analysis

(10/25/2009)

Liang Liu，Lili Yu, and Scott Edwards

1. Installation of PHYBASE in R

   Before installing PHYBASE, you have to make sure that R has already been installed on your computer. Otherwise, you need to download R at http://www.r-project.org/ and follow the instruction to install R on your computer.

   There are two ways to install a package in R. If the package is available at the R repository, you can use the command *install.packages*("*packagename*") in R to install the package. Alternatively, you may download the .gz file of the package on your computer and install it using the command *R CMD INSTALL packagename* at the Mac terminal (or linux, or unix). To install an R package on Windows, you need to download the compilers for Windows at http://www.stat.osu.edu/~liuliang/research/phybase.html.

   You are now ready to install PHYBASE. Since PHYBASE has been uploaded to the R repository, you may use the command ***install.packages("phybase")*** in R to install PHYBASE. Alternatively, you may download the zip file phybase_1.1.tar.gz (the latest version of phybase) at http://www.stat.osu.edu/~liuliang/research/phybase.html. Open the terminal window and get to the folder where the file phybase_1.1.tar.gz is located. Type ***R CMD INSTALL phybase_1.1.tar.gz*** to install the package on MacOS, Linux, or Unix. To install the package on Windows, it requires installing the gcc compiler on your computer. More details on installing PHYBASE on Windows are available at the phybase website.

2. Basic knowledge for using R.

   The pound sign # is used for adding comments in R. Thus anything after # will not be executed by R.

   (i) Assignment

```
a <- 3                                  # assign a value to variable "a"
a <- c(1,3,5)                           # assign a vector to "a"
a <- matrix(1,nrow=3,ncol=5)            #assign a matrix to "a"
a <- 1:100                              # assign a sequence to "a"
a <- runif(100)          #generate 100 random numbers from uniform(0,1)
a <- rnorm(100)          # generate 100 random numbers from uniform(0,1)
```

   (ii) Print out variables

```
print(a)                 #print out values of a
```

```
a                                   # or simply type “a” to see the value of a
```

(iii) Arithmetic operation

```
a <- 5
b <- a + 1 + 2 * 3 + 2^2 + exp (10) + log (30) + 45/3
c <- floor(b)                # the largest integer among those that are less than b
a <- 1:100
mean (a)                     #calculate the mean of vector a
var (a)                      #variance
min(a)                       #minimum
b <- a +100                  #print out b to see the result
b <- matrix (a, nrow=20, ncol=5) #now, b is a matrix
mean(b)                      # calculate the mean of all values in matrix b.
```

(iv) Getting help

```
?mean                    #help for using command “mean”
?install.packages
```

3. Using PHYBASE in R

PHYBASE provides functions to read, write, manipulate, simulate, estimate, and summarize phylogenetic trees (gene trees and species trees). The input/output functions can read and write phylogenetic trees in the Nexus and Phylip format. The trees are read in as a string and then transformed to a matrix which describes the relationship of nodes and branch lengths. The nodes matrix provides an easy access for developers to further manipulate the tree, while the tree string provides interface with other phylogenetic R packages such as "ape". The input/output functions can also be used to change the format of tree files between Nexus and Phylip. Basic functions are available in the package for manipulating phylogenetic trees such as deleting and swapping nodes, rooting and unrooting trees, changing the root of the tree. The package also includes functions such as "consense", "coaltime, "popsize", "treedist" for summarizing phylogenetic trees. Please refer to the reference manual for the complete list of commands in phybase. The major features of Phybase include:

- Read and write sequences in Nexus and Phylip format.
- Read and write trees in Nexus and Phylip format.
- Summarize trees.
- Simulate DNA sequences from gene trees
- Simulate gene trees from a species tree under the multispecies coalescent model.
- Single locus and multilocus bootstrapping (Seo 2008).
- Calculate the probability density function of a set of gene trees given a species tree. (Rannala and Yang 2003)
- Manipulate trees.
- Estimate species trees using STAR/STEAC (Liu et al 2009a and 2009b), and MT (maximum tree) (Liu et al 2009c).

To use the functions in PHYBASE, you need to open R and load the library “phybase” by simply typing *library(phybase)* in R. The following codes are used to

demonstrate how to use functions in PHYBASE to read and write sequences and phylogenetic trees, bootstrap multilocus sequences, simulate sequences from a species tree, and use STAR to estimate species trees. We have italicized parameters within each function – the names of these parameters cannot be changed and are specific to the function. Throughout we have used the prefix 'mammal' to designate user-specific file names that can be changed depending on the study. Due to the special format, copy and paste the following codes in R may not work. We have prepared a txt file rcode_manual.txt for you to copy and paste these codes.

PHYLOGENETIC ANALYSIS USING PHYBASE

1. Open phybase and read DNA sequences from a nexus file

```
#upload the phybase library
library(phybase)

#read the sequence file. make sure that data file mammal.nex is under your working directory
mammal.data<-read.dna.seq(file="mammal.nex")

#get the DNA sequences
mammal.sequence<-mammal.data$seq

#get gene partitions
mammal.gene<-mammal.data$gene

#get taxa names (names of OTUs in nexus file, not necessarily species names).
mammal.taxaname<-mammal.data$name

#write sequences to a phylip file (we can use this routine to change the file format).
write.dna(sequence=mammal.sequence, file= "mammal.phy", format= "phylip",
name=mammal.taxaname)
```

2. Read and write phylogenetic trees

```
#read trees from a file
mammal.treefile<-read.tree.string(file= "treefile.t")

#get tree strings. there are 200 trees in file treefile.t
mammal.trees<-mammal.treefile$tree

#get taxa names (names of taxa in tree 1 if the tree file is in the phylip format; names of taxa in the
#translation table if the tree file is in the nexus format)
mammal.names<-mammal.treefile$names
```

```
#trees can be read from the command line, as follows; the numbers before the # sign are branch lengths
#leading to speciation events (not gene divergence events) in units of substitutions per site; the
#numbers after the # sign are theta (θ=4Nμ) in units of substitutions per site; the

#read a species tree from the command line
mammal.sptree<-"(((S1:0.005#0.01 , S2:0.005#0.01 ) : 0.00025 #.01, S3:0.00525#0.01):0.00025 #0.01 ,
S4:0.0055#0.01) #.01;"

#get species names
mammal.spname<-species.name(str=mammal.sptree)

#type mammal.spname to check on the names in this vector. It is always a good practice to check every
#step to see if it is correct. Here the species names are S1, S2, S3, and S4.
mammal.spname

#get the relationship of nodes in the tree
mammal.nodematrix<-read.tree.nodes(str=mammal.sptree, name=mammal.spname)$nodes

#type to check on tree matrix format
mammal.nodematrix
```

Notes: Columns 1-6 of nodematrix are as follows: “parental node”, “offspring1”, “offspring2”, “branch length”, “theta (population size, θ)”, and “relative mutation rate”. The population size parameter is $\theta = 4\mu N$, where N is the effective population size. The branch length T is in mutation units, i.e., $T = \mu t$, where μ is the average mutation rate per site per generation across all populations in the species tree and t is the number of generations. In addition to the overall average mutation rate μ which is shared by all populations, each population has its own relative mutation rate $r_i = \mu_i / \mu$ (tree 1 in figure 1). It follows that the mean of relative mutation rates is always 1, because $\frac{1}{n}\sum_i r_i = \frac{1}{n}\sum_i \mu_i / \mu = 1$. Thus the mutation rate of population i is $\mu_i = r_i \times \mu$ and the branch length measured as the number of mutations per site is $\tau_i = \mu_i \times t_i = T_i \times r_i$. It is easy to see that if all relative mutation rates equal to 1, all populations will have the same mutation rate μ and the species tree is clocklike (or ultrametric). For the case of variable relative mutation rates, the species tree becomes non-ultrametric (tree 2 in figure 1).

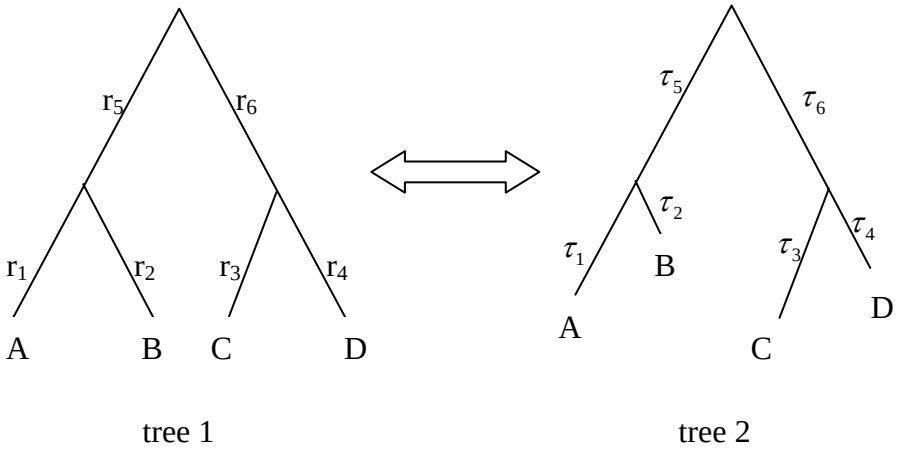
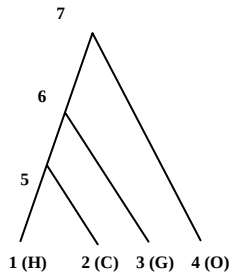


Figure 1: A species tree with variable relative mutation rates. In tree 1, r_i is the relative mutation rate of population *i* (represented by branch *i* in tree 1). Thus the mutation rate of population *i* is $\mu_i = r_i \mu$. The branch length of population *i* is $T_i = \mu_i t_i$, where t_i is the number of generations that population i extended over history. To convert branch lengths in tree 1 to the number of mutations per site (this is one of the most commonly used units for branch lengths in species trees), branch length T_i in tree 1 has to be multiplied by the relative mutation rate r_i, which results in tree 2 with branch length $\tau_i = r_i T_i$. Tree 2 is a non-ultrametric species tree with branch lengths in mutation units (the number of mutations per site).

Column 6 in nodematrix, relative mutation rate, is used when using the non-clocklike species tree model. Rows represent nodes in the tree. The first n (in this example, n=4) rows are used to denote terminal nodes while rows > n are internal nodes in the tree. Since terminal nodes do not have offspring, the values for “offspring1” and “offspring2” of the first n rows are “-9” which is used to denote missing data. The root is row 7. Thus the first column of row 7 is -9 because the root does not have ancestral nodes. The terminal nodes have the same order as the species names as they appear in the vector “name”. For example, the species names in the vector “name” are H, C, G, O. So node 1 (row 1 in nodematrix) is taxon H, and node 2 is C and so on.

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
[1,]	5	-9	-9	0.00500	0.01	-9
[2,]	5	-9	-9	0.00500	0.01	-9
[3,]	6	-9	-9	0.00525	0.01	-9
[4,]	7	-9	-9	0.00550	0.01	-9
[5,]	6	1	2	0.00025	0.01	-9
[6,]	7	5	3	0.00025	0.01	-9
[7,]	-9	6	4	-9.00000	0.01	-9



```
# you may change population sizes
mammal.nodematrix[,5] <- runif(7, 0.0001, 0.01)

# assign a relative mutation rate to each branch
mammal.nodematrix[,6] <- 7*rdirichlet(1, c(5,5,5,5,5,5,5))

#Check how mammal.nodematrix has been updated
mammal.nodematrix

#write mammal.nodematrix to a tree string
mammal.treestr<-write.subtree(inode=7, nodes=mammal.nodematrix, nspecies=4, inodeindex=7)

#write tree strings to a file
write.tree.string(X=mammal.treestr, format= “Nexus”, file= “mammal.tree.nex”,
name=mammal.spname)
```

3. Simulate gene trees from a species tree

A – Clocklike species tree:

We use sim.coaltree.sp to generate a gene tree from a species tree under the coalescent model (Rannala and Yang 2003 formula) assuming a molecular clock. Species S1, S2, S3, and S4 have 4, 3, 1, and 2 alleles respectively (specified by seq=c(4,3,1,2)). Sim.coaltree.sp returns two values: gt (a gene tree), and the height (gene tree height). Use the dollar sign ($) to recover these values.

```
#simulate a gene tree
mammal.simtree <- sim.coaltree.sp(rootnode=rootoftree(mammal.nodematrix),
nodematrix=mammal.nodematrix, nspecies=4, seq=c(4,3,1,2), name=paste("S",1:4,sep=""))

#get the simulated gene tree string
mammal.genetree <- mammal.simtree$gt

#get the height of the simulated gene tree
mammal.treeheight <- mammal.simtree$height

#write the simulated gene tree to a file
write.tree.string(X=mammal.genetree, format= “nexus”, file= “gtree.txt”)
```

To generate multiple gene trees from a species tree using sim.coaltree.sp we can write a loop:

```
#generate 100 gene trees
ngenetrees<-100
```

```
#construct a vector to store the 100 simulated gene trees.
mammal.genetrees<-rep(" ", ngenetrees)

#use a loop to simulate 100 gene trees
for(i in 1:ngenetrees) {
      mammal.genetrees[i]<-sim.coaltree.sp(rootnode=rootoftree(mammal.nodematrix), nodematrix=
                  mammal.nodematrix, nspecies=4, seq=c(4,3,1,2), name=paste("S",1:4,sep=""))$gt
}
```

B – Non-clocklike species tree:

We use sim.coaltree.sp.mu to generate gene trees from a species tree under the non-clocklike species tree model. We need to define an ultrametric species tree with variable relative mutation rates (tree 1 in figure 1) which is actually a non-ultrametric species tree (tree 2) if the branch length is converted to mutation units. "sim.coaltree.sp.mu" can take care of the branch length conversion inside the function. So we do not need to do the conversion part. What we need is just an ultrametric species tree with variable relative mutation rates. We assume that relative mutation rates (tree 1 in figure 1) vary across populations (branches) in the species tree. Relative mutation rates follow a dirichlet distribution with parameter alpha. Small alphas result in mutation rates with large variance, i.e., the molecular clock assumption is seriously violated. Large alphas can produce mutation rates that approximately follow the molecular clock assumption. For example, the mutation rates generated from alpha=200 are (1.039, 1.009, 1.01, 0.907, 1.049, 1.07, 0.916). See Liu et al. 2009 Syst. Biol. for examples of this algorithm.

```
#simulate 100 gene trees from a non-clocklike species tree
mammal.genetrees <- sim.coaltree.sp.mu(sptree=mammal.sptree, spname=mammal.spname,
seq=c(1,1,1,1), numgenetree=100, method = "dirichlet",alpha=5.0)$gt
```

You may use pre-specified relative mutation rates (the mean of relative mutation rates must be 1).

```
#pre-specified relative mutation rates
relativeMu <- c(1.039, 1.009, 1.01, 0.907, 1.049, 1.07, 0.916)

#simulate 100 gene trees from a species tree with pre-specified mutation rates
mammal.genetrees <- sim.coaltree.sp.mu(sptree=mammal.sptree, spname=mammal.spname,
seq=c(1,1,1,1), numgenetree=100, method= "user", alpha= relativeMu)$gt
```

We can generate gene trees with multiple alleles sampled from each species. Because multiple alleles may coalesce in the terminal populations, we have to assign a population size to each terminal population.

```
#generate a gene tree with (2, 3, 4, 5) alleles from each of the 4 species S1, S2, S3, and S4.
mammal.genetree <- sim.coaltree.sp.mu(sptree=mammal.sptree, spname=mammal.spname,
seq=c(2,3,4,5), numgenetree=1, method="dirichlet", alpha=5.0)$gt

#write the gene tree to a file
write.tree.string(mammal.genetree, format="nexus", file="gtree.txt")
```

Then we can generate DNA sequences from a gene tree using Kimura 2 parameter model (1=JC, 2=K2P, 3=HKY, 4=GTR). The sequences are coded as 1, 2, 3, 4 which can be easily converted to A, C, G, and T.

```
#read the node matrix of the gene tree
mammal.genetreenode <- read.tree.nodes(str=mammal.genetree)
mammal.gnodematrix <- mammal.genetreenode$nodes

#get OUT names for the gene tree
mammal.taxaname <- mammal.genetreenode$names

#generate sequences of 500 base pairs
dna <- sim.dna(nodematrix=mammal.gnodematrix, seqlength=500, model=2, kappa=2,
rate=c(1,1,1,1,1,1), frequency=c(1/4,1/4,1/4,1/4))

#the sequences generated from sim.dna are composed of 0, 1, 2, 3 corresponding to A, C, G, T.
dna[dna==1] <- "A"
dna[dna==2] <- "C"
dna[dna==3] <- "G"
dna[dna==4] <- "T"

#write sequences to a file in phylip, mrbayes, or best format.
write.dna(dna, name=mammal.taxaname, file="dna.txt", format="nexus", program="mrbayes")
write.dna(dna, name=mammal.taxaname, file="dna.phy", format="phylip")
write.dna(dna, name=mammal.taxaname, file="dna.best", format="nexus", program="best")
```

simSeqfromSp is a wrapper to simulate DNA sequences from a species tree and write the sequences to a file. For example, we can use this wrapper to generate sequences from 2 gene trees (ngene=2) simulated from a species tree. The sequences are generated under the GTR substitution model (model=4).

```
#use the wrapper simSeqfromSq to simulate sequences from a species tree
simSeqfromSp(sptree=mammal.sptree, spname=mammal.spname, ntaxasp=c(1,1,1,1), ngene=2,
theta=0, noclock=0, simsequence=1, murate="Dirichlet", alpha=5, seqlength=100, model=4, kappa=2,
rate=c(1,1,1,1,1,1), frequency=c(1/4,1/4,1/4,1/4), outfile="seqfromSp.txt", format="phylip")
```

4. Multilocus bootstrap (Seo 2008) for STAR, STEAC, MT and other methods.

In this approach, genes are sampled at random with replacement, and then sites within each sampled gene are sampled at random with replacement (Seo 2008). If you look at the resulting file you can determine which gene was sampled in each replicate by examining the length of the gene in each phylip block). Taxa with missing data for a particular gene are not included when that gene is sampled. A gene is regarded as missing if all sites in that gene are '?' or 'N'.

We use the mammal data set as an example to demonstrate how to generate bootstrap samples. Since there are 20 genes in the mammal data set, if we want to generate 2 bootstrap samples, the number of blocks of sequences in the output phylip file is 20*2 = 40. For each block (or gene), we can estimate a gene tree using phylip or phyml, paup, or any other program. The estimated gene trees are then used to build STAR, STEAC or MT trees.

#read a sequence file
mammal.data<-read.dna.seq(*file*= "mammal.nex")

#get DNA sequences
mammal.sequence<-mammal.data$*seq*

#get gene partitions
mammal.gene<-mammal.data$*gene*

#get taxa names
mammal.taxaname<-mammal.data$*name*

#bootstrap
bootstrap.mulgene(*sequence*=mammal.sequence, *gene*=mammal.gene, *name*=mammal.taxaname, *boot*=2, *outfile*="mammalboot.txt")

One now has a vector of bootstrap replicates across genes and sites. One can now use phyml, paup or any other program to make gene trees of each bootstrap replicate. These gene trees should be collected into a file of newick format trees without any symbols other than taxa, branch lengths and parentheses (for example, no bootstrap proportions or consensus trees should be in the file). Each gene tree should be separated by a semi-colon.

5. Building a MT, STAR, or STEAC species tree

We first build a MT and STAR tree from a gene tree vector which can be either read from the command line as follows, or from a tree file using read.tree.string

```
#read a tree vector from the comman line. The tree vector consists of three gene trees
genetreevector<-c("((((H:0.00302,C:0.00302):0.00304,G:0.00605):0.01029,O:0.01635):0.1,W:0.11635
);","((((H:0.00402,G:0.00402):0.00304,C:0.00705):0.00929,O:0.01635):0.1,W:0.11635);",
"((((H:0.00302,C:0.00302):0.00304,G:0.00605):0.01029,O:0.01635):0.1,W:0.11635);");

#get taxa names
mammal.taxaname<-species.name(genetreevector[1])
```

In this example, the command retrieves the taxa names (OTUs) from the first gene tree of the vector. In some cases some of the gene trees may not have all the taxa, because these taxa may be missing a particular gene (see above). In this cases one wants to sample the taxa names from a gene tree that has all the species in it.

```
#this command assigns to the vector ‘mammal.spname’ the names in the vector ‘mammal.taxaname’.
mammal.spname <- mammal.taxaname
```

We use a matrix to define the relationship between taxa in gene trees and species in the species tree, i.e. which taxa in gene trees belong to which species in the species tree.

```
#construct a matrix to denote the species-taxa relationship
species.structure<-matrix(0,5,5)
diag(species.structure)<-1
species.structure
```

In the matrix species.structure, columns represent taxa and rows represent species. If the value at row i and column j is 1, it means that the j^{th} taxon belongs to the i^{th} species. For example, the matrix “species.structure” indicates that the first taxon belongs to the first species, and so on.

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	0	0	0	0
[2,]	0	1	0	0	0
[3,]	0	0	1	0	0
[4,]	0	0	0	1	0
[5,]	0	0	0	0	1

We may use function “spstructure” to create a species.structure when multiple sequences are sampled from species. For example, we can use spstructure to create a species.structure for the case where the first 2 sequences belong to the first species, next 3 sequences belong to the second species, next 4 sequences belong to the third species, 1 sequence belong to the fourth species, and 1 sequence belong to the last species.

```
seq<-c(2,3,4,1,1)
species.structure1<-spstructure(seq)
species.structure1
```

We calculate the MT, STAR, and STEAC trees.

maxtree(genetreevector, *spname*=mammal.spname, *taxaname*=mammal.taxaname, *species.structure*=species.structure)

star.sptree(genetreevector, *speciesname*=mammal.spname, *taxaname*=mammal.taxaname, *species.structure*=species.structure, *outgroup*=”W”, *method*="nj")

steac.sptree(genetreevector, *speciesname*=mammal.spname, *taxaname*=mammal.taxaname, *species.structure*=species.structure, *outgroup*=”W”, *method*="nj")

6. Building a bootstrapped STAR, STEAC or MT tree from bootstrap replicates.

In this step we read in the entire vector of bootstrapped gene trees and then make this vector into a matrix whose columns are the number of bootstrap replicates and whose rows are the number of genes in the original data set. For example, there were 20 genes in the mammal dataset and I did 1000 bootstrap replicates, my gene tree vector should have 20,000 gene trees in it. I then need to transform this vector into a matrix with 1000 columns (1 for each replicate) and 19 rows, each corresponding to a different gene).

```
#read the original nexus file and assign names to OTUs in gene trees and species tree
mammal.data<-read.dna.seq(file="mammal.nex")
mammal.sequence<-mammal.data$seq

mammal.gene<-mammal.data$gene
mammal.ngene<-dim(mammal.gene)[1]
mammal.spname<-mammal.data$name
mammal.taxaname <- mammal.spname

# read the gene tree vector and assign it to variable ‘mammal.boottrees’. There must be 1000*20 trees
#in the boostrap tree file “mammal.trees.txt” produced from PhyML. Mammal.trees.txt is not available.
#So replace it with your bootstrap #tree file.
mammal.boottrees <- read.tree.string("mammal.trees.txt")$tree

#make a matrix called ‘test.treematrix’ from the gene tree vector
mammal.treematrix<-matrix(mammal.boottrees, ncol=1000, nrow=mammal.ngene, byrow=FALSE)

#make a matrix (as above) assigning each allele in the gene trees to each species in the species tree
```

```
#there are 54 species in the species tree
species.structure<-matrix(0, 54, 54)

#link each allele in the gene trees with each species in the species tree; there is 1:1 mapping in this
#example, since only the diagonal of the matrix has 1s in it.
diag(species.structure)<-1

#create an empty vector that will be filled with STAR trees (or MT or STEAC trees) made from each
#column in the matrix 'mammal.treematrix'
mammal.startrees<-rep(" ",1000)

#calculate star trees
for(i in 1:1000){
    mammal.startrees[i]<-star.sptree(mammal.treematrix[,i],speciesname=mammal.spname,taxaname
    =mammal.taxaname, species.structure=species.structure, outgroup="Opossum", method="nj")
}

#write star trees to a file
write.tree.string(mammal.startrees, file="mammal.startrees.nex")
```

You can now take the nexus file of STAR trees (in this example 1000 trees) and make a consensus tree from them using Paup or Phylip or any other program.

References:

Liu, L., L. Yu, L. Kubatko, D. K. Pearl, and S. V. Edwards. 2009a. Coalescent methods for estimating phylogenetic trees. Molecular Phylogenetics and Evolution 53:320-328.

Liu, L., L. Yu, D. K. Pearl, and S. V. Edwards. 2009b. Estimating species phylogenies using coalescence times among sequences. Systematic Biology Systematic Biology Advance Access published on July 16, 2009:doi:10.1093/sysbio/syp1031.

Liu, L., L. Yu, and D. K. Pearl. 2009c. Maximum tree: a consistent estimator of the species tree. J Math Biol 60:95-106.

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.

Seo, T. K. 2008. Calculating bootstrap probabilities of phylogeny using multilocus sequence data. Mol. Biol. Evol. 25:960-971.