Analysing diversification with diversitree

Rich FitzJohn, with GeoSSE by Emma Goldberg

25 March 2012, version 0.9-2

Contents

1 Introduction

The diversitree package includes a number of comparative phylogenetic methods, mostly focusing on analysing diversification and character evolution. These methods all share a common set of tools for performing maximum likelihood (ML) parameter estimation, hypothesis testing and model comparison, and Bayesian parameter estimation through Markov chain Monte Carlo (MCMC). Included methods include:

- Diversification
  - Constant rate birth-death models (Nee et al., 1994)
- Character evolution
  - Discrete trait evolution (Pagel, 1994)
  - Univariate Brownian motion and Ornstein-Uhlenbeck models of continuous trait evolution
- Joint character evolution and diversification:
  - BiSSE: Binary trait speciation and extinction (Maddison et al., 2007), and extensions for incompletely resolved phylogenies (FitzJohn et al., 2009).
  - MuSSE: Multiple State Speciation and Extinction.
  - QuaSSE: Quantitative State Speciation and Extinction (FitzJohn, 2010).
- Joint character evolution and diversification, with speciation-triggered changes in character states:
  - GeoSSE: Geographic State Speciation and Extinction (Goldberg et al., 2011).
  - BiSSEness: BiSSE-Node Enhanced State Shift (Magnuson-Ford and Otto, 2012).
  - ClaSSE: Cladogenetic State Speciation and Extinction (Goldberg and Igić, submitted).

In addition, variants of these models are available:

- "Time dependent": different time epochs have different parameters (implemented for BiSSE, MuSSE), and where rates are arbitrary functions of time (implemented for birth death, BiSSE, and MuSSE).
- MEDUSA-style partitioned analyses, where different regions of the tree have different parameters (implemented for birth-death, BiSSE, MuSSE, QuaSSE, and GeoSSE).
- Marginal ancestral state reconstruction for discrete characters ("Pagel94"), BiSSE, and MuSSE.
- Stochastic character mapping for discrete traits (Bollback, 2006)

In the future, new methods will include

- Reflected Brownian motion for bounded traits
- Stochastic character mapping for discrete traits that affect speciation or extinction rates

For all methods, inference can be carried out under maximum likelihood, or in Bayesian analyses via MCMC (Markov Chain Monte Carlo). Phylogenies can also be simulated under most of the models.

This tutorial is designed to give an overview of the features in diversitree. It does not aim to be a compete reference to the package, or claim to always follow best practice. The manual follows the structure above. Many of the examples are just taken from the online documentation; further examples can be found there. Most are fairly contrived – if you have examples you would rather see here, I would welcome data sets.

1.1 Regenerating this file

This file is written in "Sweave", which allows mixing R code and LaTeX markup. If you want to regenerate the file, or run the empirical examples, you will need some data files, available at http://www.zoology.ubc.ca/prog/diversitree/files/, which must be present in the directory `data`. Some of the code chunks take a very long time to run (total processing time is currently about 40 hours on a 2.8 GHz MacPro), and use the `cacheSweave` package to speed subsequent runs up.

All code requires that the diversitree package be loaded.

```
library(diversitree)
```

2 Constant-rate birth-death models

The ape package already has some support for constant-rate birth-death models, but diversitree duplicates this for completeness. The major differences are (1) the function is not constrained to positive diversification rates (μ can exceed λ), (2) support for both random taxon sampling and unresolved terminal clades (but see ape's `bd.ext`), and (3) run both MCMC and MLE fits to birth death trees. The constant rate birth death model is a special case of the other diversification models implemented in the package, and is the simplest model in diversitree.

Start with a simulated phylogeny, with speciation rate $\lambda = 0.1$ and extinction rate $\mu = 0.03$:

```
set.seed(2)
phy <- tree.bd(c(.1, .03), max.taxa=100)
```

(plotted in figure 1). The first step in any analysis in diversitree is to construct a likelihood function. For constant rate birth death models, this is done with `make.bd`:

```
lik <- make.bd(phy)
```

To see the argument names of the likelihood function, use the `argnames` function

```
argnames(lik)
[1] "lambda" "mu"
```

This shows that `lik` takes a vector of two parameters (λ, μ). It will return the log likelihood of the parameters, following the calculations in (Nee et al., 1994).

```
lik(c(.1, .03)) # -7.74086
[1] -7.740859
```

Most likelihood functions accept additional arguments; these are documented on their online help pages. The only additional argument accepted for birth-death model is to disable conditioning on survival (by default, the likelihood is conditional on two lineages surviving to the present, following (Nee et al., 1994).

```
lik(c(.1, .03), condition.surv=FALSE) # -10.74823
[1] -10.74823
```

```
plot(phy, no.margin=TRUE, show.tip.label=FALSE)
```

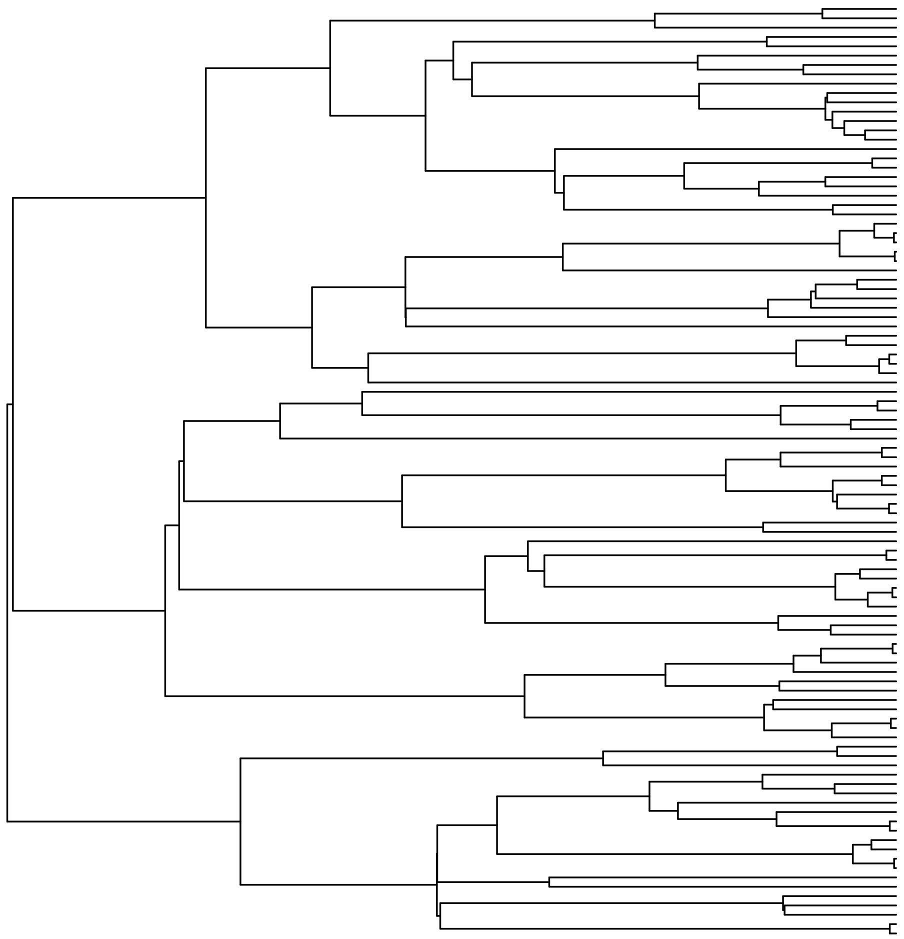

Figure 1: A birth-death tree, with speciation rate $\lambda = 0.1$ and extinction rate $\mu = 0.03$, simulated until it has 100 species (the actual simulation runs until the speciation event that would create the 101st species, but does not add this species).

To do a ML model fit, pass the likelihood function and a starting point guess to the `find.mle` function:

```
fit <- find.mle(lik, c(.1, .03), method="subplex")
```

The final argument here selects the method "`subplex`" for the ML search; other methods are available. (The default for birth-death models is "`nlm`", which may produce warnings about failure to converge for the example here.)

To extract the coefficients from the fitted object, use the `coef` function:

```
coef(fit)
```

```
    lambda         mu
0.09781857 0.05671318
```

and to extract the log-likelihood value at the ML point, use the `logLik` function

```
logLik(fit)
```

```
'log Lik.' -4.148589 (df=2)
```

which extracts the coefficients with some additional information, or extract the `lnLik` element from the list directly:

```
fit$lnLik
```

```
[1] -4.148589
```

Does this model fit much better than a model without extinction (a Yule, or pure birth, model)? You can constrain parameters of likelihood functions using the `constrain` function. To specify that the extinction rate, μ should be zero:

```
lik.yule <- constrain(lik, mu ~ 0)
```

Argument names here must match those given by `argnames`. Run the ML search the same way as above, specifying a single starting parameter (I've used the speciation rate from the full model here):

```
fit.yule <- find.mle(lik.yule, coef(fit)[1], method="subplex")
```

To perform a likelihood ratio test, use the `anova` function[1] The model with the nonzero extinction estimate is preferred, with $\chi^2_1 = 5.7$

```
anova(fit, yule=fit.yule)
```

```
     Df   lnLik    AIC  ChiSq Pr(>|Chi|)
full  2 -4.1486 12.297
yule  1 -7.0118 16.024 5.7264    0.01671
```

[1]This is an unfortunate convention in R: many packages use this function as a general model comparison function, and I've taken their lead here – in a future version, I may add a `lrt` function, which should be clearer. No analysis of variance is performed.

Alternatively, we can use Markov chain Monte Carlo (MCMC) to perform a Bayesian analysis. Here, I will use a uniform prior on the interval $[0, \infty)$ for both parameters by not specifying any prior. The `w` parameter is the tuning parameter. Here, it affects how many function evaluations will be needed per sample, but will not generally affect the rate of mixing (see the online help for `mcmc` for more information).

```
set.seed(1)
samples <- mcmc(lik, fit$par, nsteps=10000, w=.1, print.every=0)
```

The posterior distribution of these parameters, and the code to generate it, is in figure 2.

Analyses can also use trees where only a fraction of species are present in the phylogeny. To demonstrate this, let's drop 25 of the 100 species from the original tree at random:

```
set.seed(1)
phy.sub <- drop.tip(phy, sample(100, 25))
```

When constructing the likelihood function, pass an argument `sampling.f` in, with a value on $(0, 1]$ representing the fraction of species that are descended from the root node that are included in the phylogeny (here, 75/100). Then, run a ML analysis with `find.mle` as before:

```
lik.sub <- make.bd(phy.sub, sampling.f=75/100)
fit.sub <- find.mle(lik.sub, c(.1, .03), method="subplex")
coef(fit.sub)
```

```
    lambda         mu
0.08604488 0.04232557
```

With fewer included species, test to see whether the full model is still preferred over the Yule model:

```
lik.sub.yule <- constrain(lik.sub, mu ~ 0)
fit.sub.yule <- find.mle(lik.sub.yule, coef(fit.sub)[1],
                         method="subplex")
anova(fit.sub, yule=fit.sub.yule)
```

```
     Df   lnLik    AIC  ChiSq Pr(>|Chi|)
full  2 -39.182 82.364
yule  1 -40.501 83.002 2.6388     0.1043
```

In this small tree, the support for the model with extinction is only $\chi^2_1 = 2.6$ (compared with $\chi^2_1 = 5.7$ when we had all species), which is no longer significant at the 5% level.

```
samples$r <- samples$lambda - samples$mu
col <- c("#eaab00", "#004165", "#618e02")
profiles.plot(samples[c("lambda", "mu", "r")], col.line=col, las=1,
              alpha=.75, legend.pos="topright")
abline(v=c(.1, .03, .07), col=col, lty=2)
```

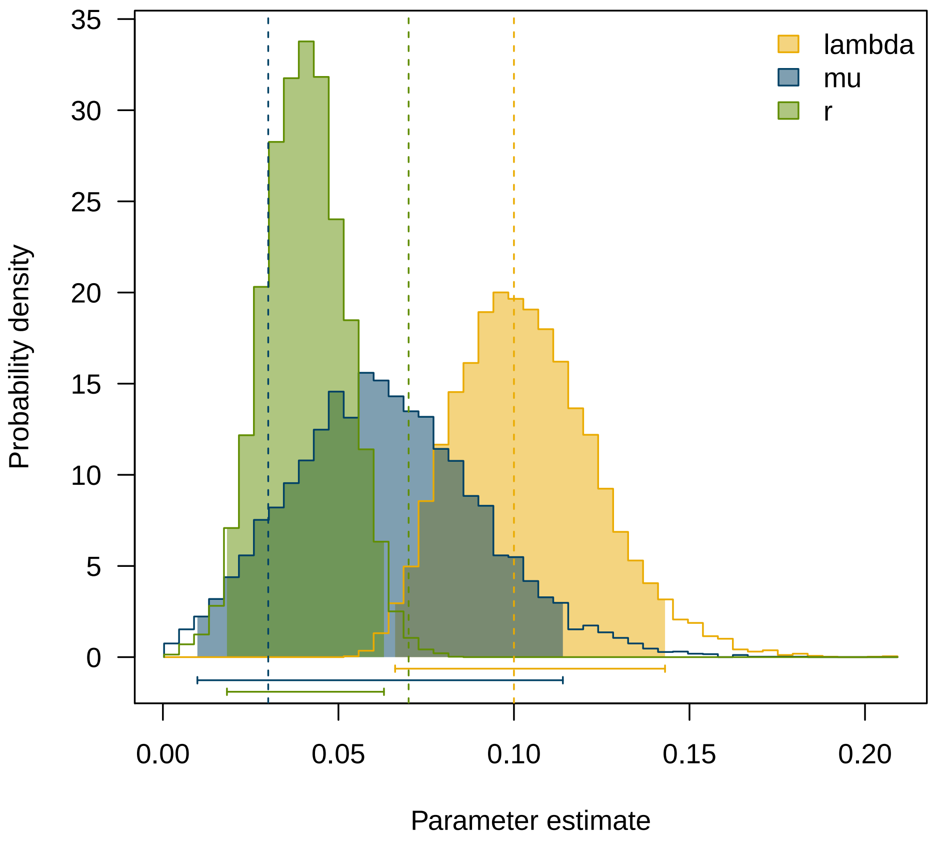

Figure 2: Posterior probability distributions for the parameters of the constant rate birth death model. True values are indicated by the solid vertical lines. The bars at the bottom of the distributions and the shaded areas correspond to the 95% credibility intervals. These include the two parameter λ and μ, though the true diversification rate r lies above the 95% credibility interval for that parameter.

3 Markov models of discrete character evolution

As with the birth death models above, diversitree supports both simulating discrete characters and estimating rates of character evolution. The implemented methods are a little idiosyncratic, as they are primarily here for completeness (representing special cases of the joint diversification-character evolution models), but are also useful in their own right.

First, we will simulate a binary trait on a birth death tree. The tree above with 100 tips is a bit unwieldy to visualise, so we'll make a smaller 50 tip tree:

```
set.seed(2)
phy <- tree.bd(c(.1, .03), max.taxa=50)
```

Then, on this tree, simulate a binary character where the rate of transition from state 0 to state 1 is 0.1, and the reverse rate is 0.2. We'll start the tree in state 0 (specified by `x0`). The argument `model="mk2"` specifies that we will use a Mk2 model:

```
set.seed(1)
states <- sim.character(phy, c(.1, .2), x0=0, model="mk2")
```

The simulation remembers the history at nodes too, which is displayed in figure 3 (eventually full history will be recorded).

Next, build a likelihood function with the `make.mk2` function, and run a ML analysis with `find.mle`, using an initial parameter guess of (0.1, 0.1):

```
lik.mk2 <- make.mk2(phy, states)
fit.mk2 <- find.mle(lik.mk2, c(.1, .1), method="subplex")
coef(fit.mk2)
```

```
       q01        q10
0.08399429 0.23020744
```

In the fit, the q_{10} parameter is higher than q_{01}, the difference being a little larger than in the true model.

See if this difference is statistically justified by running a model where the two q values are constrained to be equal:

```
lik.mk1 <- constrain(lik.mk2, q10 ~ q01)
fit.mk1 <- find.mle(lik.mk1, .1, method="subplex")
anova(fit.mk2, mk1=fit.mk1)
```

```
     Df   lnLik    AIC  ChiSq Pr(>|Chi|)
full  2 -26.522 57.044
mk1   1 -31.095 64.191 9.1466   0.002492
```

This is significant ($\chi^2_1 = 9.1$) so we can conclude that asymmetric model fits better.

3.1 Drawing samples with MCMC

It is possible to run an MCMC analysis. However, care should be taken to choose priors carefully, as while $q_{10}/(q_{01}+q_{10})$ is usually well characterised by the data, the overall rate $(q_{01}+q_{10})$ is poorly defined. For small trees like this, essentially infinite values of character evolution are consistent with the data, with the tip states just drawn from the stationary distribution of the process.

```
plot(phy, show.tip.label=FALSE, no.margin=TRUE)
col <- c("#004165", "#eaab00")
tiplabels(col=col[states+1], pch=19, adj=1)
nodelabels(col=col[attr(states, "node.state")+1], pch=19)
```

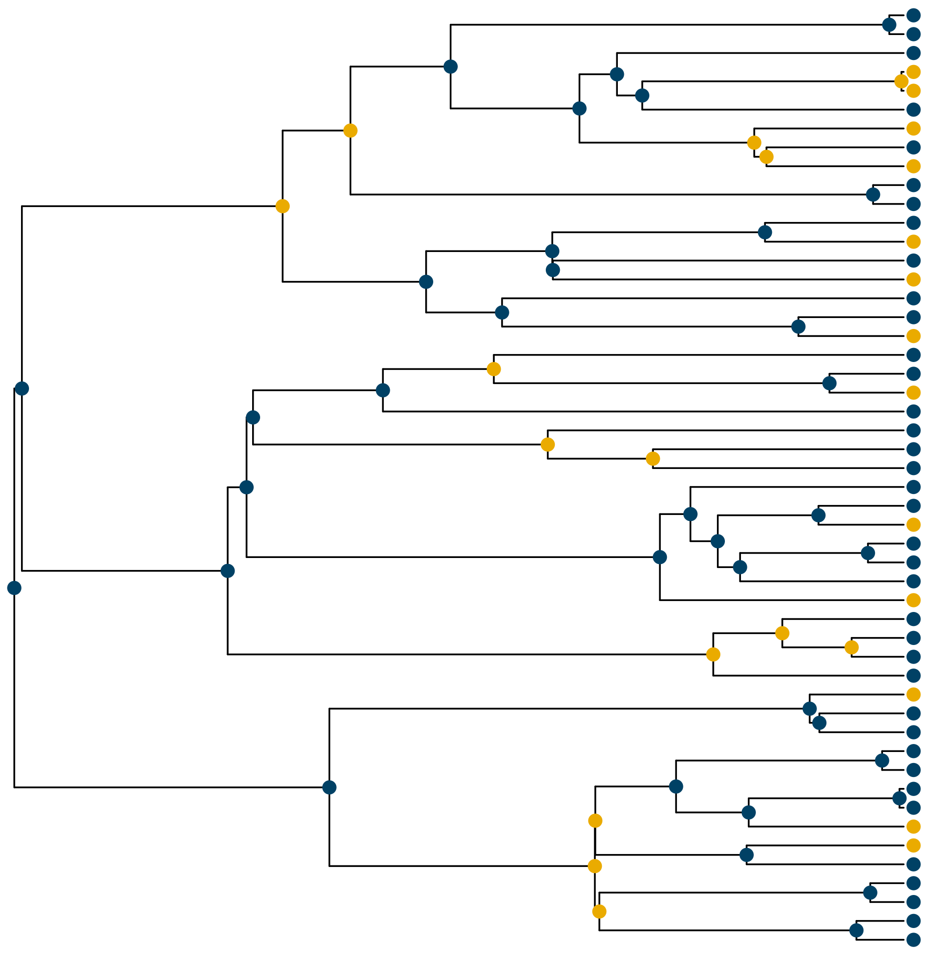

Figure 3: Character history for simulated trait and tree. Blue is state `0`, yellow is state 1.

There are two supplied prior functions, but any function that takes a vector of parameters and returns the log prior probability may be used. First, consider an exponential prior with rate 10, which gives a mean of 1/10, and assume the same prior distribution for both parameters.

```
prior.exp <- make.prior.exponential(10)
```

To run the MCMC, we need to specify a starting point (again, I have used (.1, .1), but the ML point might be preferable). I have discarded the first 500 samples (10%), which is probably overkill for this model, as the autocorrelation between samples is extremely small.

```
set.seed(1)
samples <- mcmc(lik.mk2, c(.1, .1), nsteps=5000, prior=prior.exp,
                w=.1, print.every=0)
samples <- subset(samples, i > 500)
```

The marginal distributions of these parameters are shown in figure 4, and overlap substantially. However, that is because the distributions are correlated (increasing q_{01} fits best if q_{10} is also increased). The q_{01} parameter is estimated to be greater than the q_{10} parameter only a small fraction of the time:

```
mean(samples$q01 > samples$q10)
```

```
[1] 0.002222222
```

```
col <- c("#004165", "#eaab00")
profiles.plot(samples[c("q01", "q10")], col.line=col, las=1,
              legend.pos="topright")
abline(v=c(.1, .2), col=col)
```

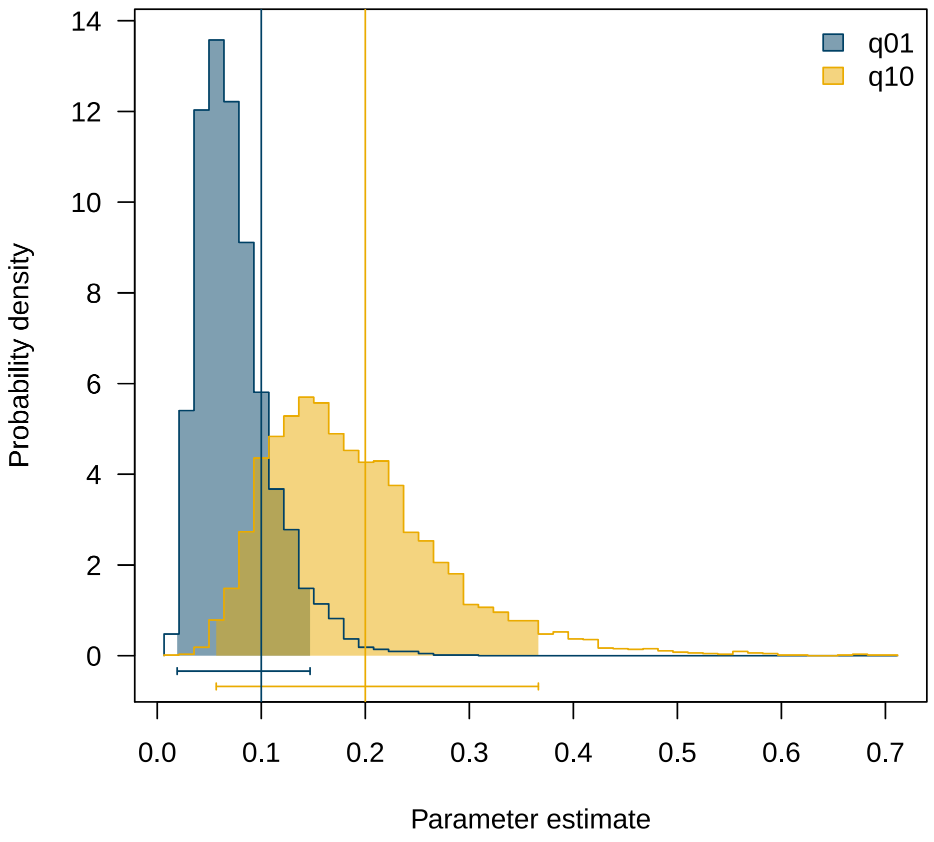

Figure 4: Posterior probability distributions for the parameters of the Mk2 model. True values are indicated by the solid vertical lines.

4 Binary traits and diversification: BiSSE

The BiSSE (Binary State Speciation and Extinction) model combines the features of the constant-rates birth-death model with the two-state Markov model. Again, start with a simulated tree. The parameters here are in the order λ_0, λ_1, μ_0, μ_1, q_{01}, q_{10}, so the parameters below correspond to an asymmetry in the speciation rate where state 1 speciates at twice the rate as state 0. All other parameters are equal between states.

```
pars <- c(0.1, 0.2, 0.03, 0.03, 0.01, 0.01)
set.seed(4)
phy <- tree.bisse(pars, max.t=30, x0=0)
states <- phy$tip.state
head(states)
```

```
 sp4  sp8  sp9 sp14 sp15 sp16
   0    1    1    1    1    0
```

This gives a 52 species tree, shown with its true history in figure 5. The character states are now stored in the `states` vector. This vector is named, so that each element can be easily associated with a tip in the tree.

The `make.bisse` takes as its first two arguments a tree and set of character states (these are the only mandatory arguments):

```
lik <- make.bisse(phy, states)
lik(pars) # -159.71
```

```
[1] -159.71
```

To perform an ML search, we need a starting point. The `starting.point.bisse` function produces a basic heuristic guess of a sensible starting point, based on the character-independent birth-death fit. There are no guarantees that this is at all close to the ML point, or that the ML point can be reached from this point while climbing uphill only (which most optimisers assume).

```
p <- starting.point.bisse(phy)
p
```

```
   lambda0    lambda1        mu0        mu1        q01        q10
0.17173579 0.17173579 0.04158360 0.04158360 0.02603044 0.02603044
```

Start an ML search from this point (this may take some time)

```
fit <- find.mle(lik, p)
```

As above, the `fit.mle` object has an element `lnLik` with the log-likelihood value

```
fit$lnLik
```

```
[1] -158.6875
```

and coefficients may be extracted with `coef` (rounded for clarity):

```
round(coef(fit), 3)
```

```
par(mar=rep(0, 4))
col <- c("#004165", "#eaab00")
plot(history.from.sim.discrete(phy, 0:1), phy, col=col)
```

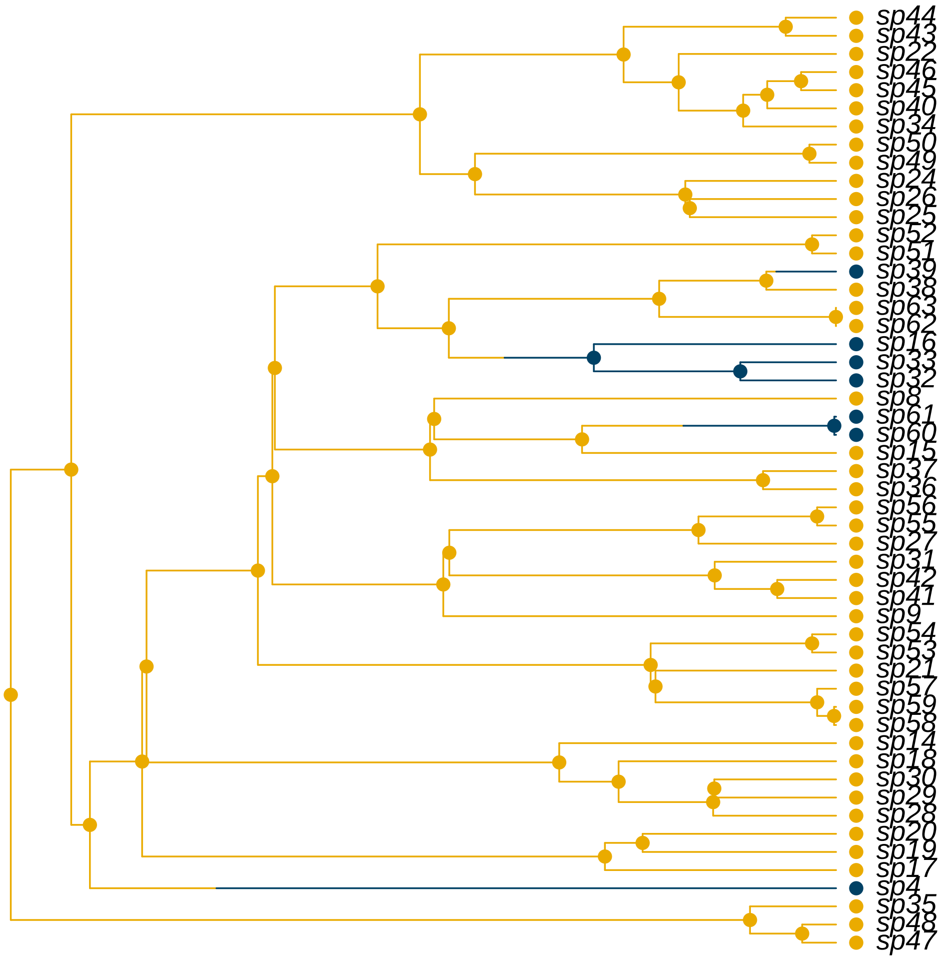

Figure 5: A BiSSE tree, with parameters $\lambda_0 = 0.1$, $\lambda_1 = 0.2$, $\mu_0 = \mu_1 = 0.03$, and $q_{01} = q_{10} = 0.01$. Blue is state 0, yellow is state 1.

```
lambda0 lambda1     mu0     mu1     q01     q10
  0.135   0.178   0.123   0.036   0.000   0.019
```

Let's test the hypothesis that the speciation rates are different for the different states. We can use `constrain` to enforce equal speciation rates to be equal across character states:

```
lik.l <- constrain(lik, lambda1 ~ lambda0)
```

and then start the ML search again:

```
fit.l <- find.mle(lik.l, p[argnames(lik.l)])
fit.l$lnLik # -158.74
```

(the statement "`p[argnames(lik.l)]`" drops the λ_1 element from the starting parameter vector). This fit has quite different parameters to the full model (compare μ_0)

```
round(rbind(full=coef(fit), equal.l=coef(fit.l, TRUE)), 3)
```

```
        lambda0 lambda1   mu0   mu1 q01   q10
full      0.135   0.178 0.123 0.036   0 0.019
equal.l   0.173   0.173 0.172 0.027   0 0.022
```

(the `TRUE` argument forces `coef` to return values for constrained parameters). However, the difference in fits is not statistically supported, with $\chi_1^2 = 0.1$

```
anova(fit, equal.l=fit.l)
```

```
        Df   lnLik    AIC     ChiSq Pr(>|Chi|)
full     6 -158.69 329.37
equal.l  5 -158.74 327.47 0.096466     0.7561
```

4.1 Analysis with MCMC

Because we are fitting six parameters to a tree with only 52 species, priors will be needed so that the posterior distribution is proper. I will use an exponential prior with rate $1/(2r)$, where r is the character independent diversification rate:

```
prior <- make.prior.exponential(1 / (2 * (p[1] - p[3])))
```

The MCMC sampler in diversitree uses slice sampling (Neal, 2003) for parameter updates. The "step size" (argument `w`) does not need to be carefully tuned as it does not affect the rate of mixing – just the number of function evaluations per update. Ideally it will be on the same order as the width of the "high probability region". An easy way of setting this is to run a short chain (say, 100 steps) and use the range of observed samples as a measure of this.

```
set.seed(1)
tmp <- mcmc(lik, fit$par, nsteps=100, prior=prior,
            lower=0, w=rep(1, 6), print.every=0)
w <- diff(sapply(tmp[2:7], range))
w
```

Run the chain for 10,000 steps (this will take a while)

```
samples <- mcmc(lik, fit$par, nsteps=10000, w=w, lower=0, prior=prior,
                print.every=0)
```

The marginal distributions for the two speciation rates are shown in figure 6, which shows the 95% credibility intervals for λ_0 completely overlapping those for λ_1.

4.2 Incomplete taxonomic sampling

Not all phylogenies are complete, but the basic BiSSE calculations assume that they are. If given tree contains a random sample of all extant species, the calculations can be corrected. To demonstrate this, we will generate a larger tree (150 taxa), and drop 50 taxa from it. Here, I am using the same parameters as earlier.

```
pars <- c(0.1, 0.2, 0.03, 0.03, 0.01, 0.01)
set.seed(4)
phy <- tree.bisse(pars, max.taxa=150, x0=0)
states <- phy$tip.state
phy.s <- drop.tip(phy, setdiff(seq_len(150), sample(150, 50)))
states.s <- states[phy.s$tip.label]
```

Calculate what the sampling fraction is for this tree. You can either assume that the sampling fraction is independent of the character state:

```
sampling.f <- 50 / 150
```

or you can assume that it varies with character state

```
sampling.f <- as.numeric(table(states.s) / table(states))
sampling.f
```

```
[1] 0.3846154 0.3284672
```

Pass this in to `make.bisse` and construct a new likelihood function that accounts for the sampling:

```
lik.s <- make.bisse(phy.s, phy.s$tip.state, sampling.f=sampling.f)
```

This can then be optimised, as before:

```
p <- starting.point.bisse(phy)
fit.s <- find.mle(lik.s, p)
fit.s[1:2]
```

4.3 Terminally unresolved trees

Another way that phylogenies might be incompletely resolved is that higher level relationships may be known (say, genera), but little or nothing is known about species relationships within these groups. This results in trees where some "taxa" represent a number of species – "terminal clades". There are a couple of different ways that unresolved clade information may be specified. To demonstrate this, I will use an

```
col <- c("#004165", "#eaab00")
profiles.plot(samples[c("lambda0", "lambda1")], col.line=col, las=1,
              xlab="Speciation rate", legend="topright")
abline(v=c(.1, .2), col=col)
```

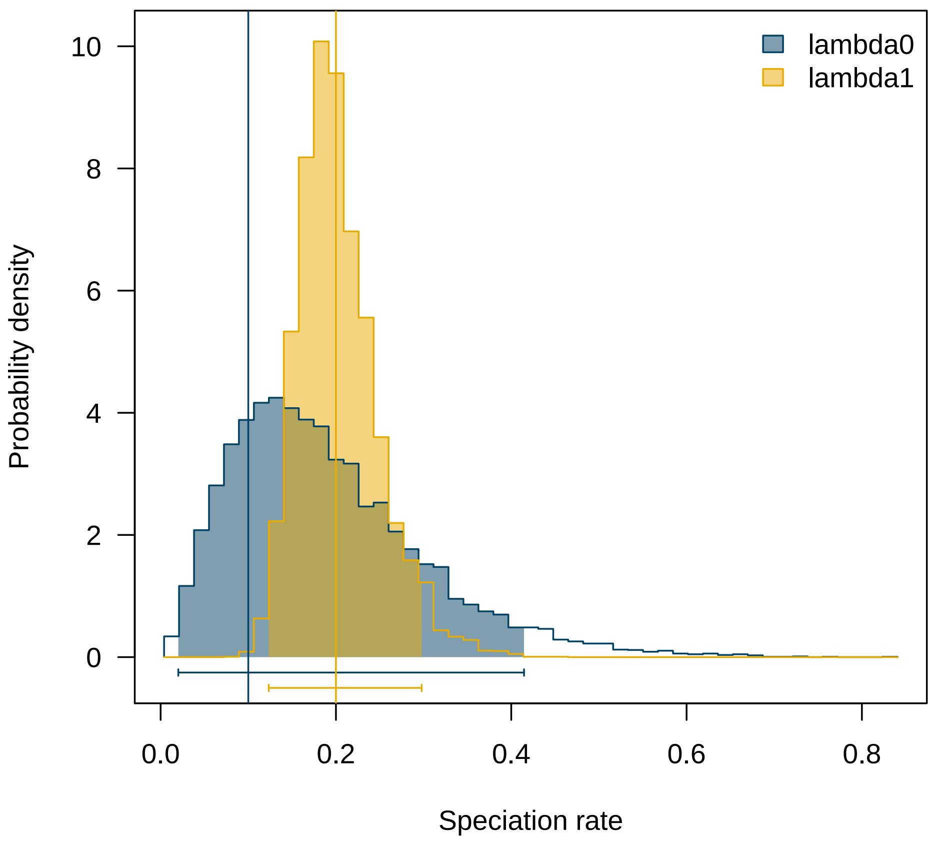

Figure 6: Posterior probability distributions for λ_0 and λ_1 for a BiSSE model. True values are indicated by the solid vertical lines. The bars at the bottom of the distributions and the shaded areas correspond to the 95% credibility intervals.

example of sexual dimorphism in shorebirds; this is the same example as in (FitzJohn et al., 2009). The phylogeny is a supertree constructed by Thomas et al. (2004), and the data on sexual size dimorphism are derived from Lislevand et al. (2007). The required files can be downloaded from `http://www.zoology.ubc.ca/prog/diversitree/files/` Read in the phylogenetic tree

```
tree <- read.nexus("data/Thomas-tree.nex")
```

The tree contains many polytomies; the original tree, and a simplified tree with polytomies converted into clades are shown in figure 7.

The character states are stored as the size of the difference of mass between sexes, divided by the mean across sexes (see FitzJohn et al., 2009).

```
d <- read.csv("data/Lislevand-states.csv", as.is=TRUE)
states <- d$dimorph
names(states) <- d$species
states <- states[tree$tip.label]
names(states) <- tree$tip.label
head(states)
```

```
Catoptrophorus_semipalmatus         Calidris_ferruginea
                -0.09888579                 -0.10222805
           Calidris_canutus            Calidris_maritima
                -0.16058394                 -0.15126050
         Calidris_acuminata               Calidris_mauri
                 0.10164425                 -0.14457831
```

These will need converting to a binary character for use, for example – to convert this into a binary character where an absolute relative difference of 10% would be considered "dimorphic":

```
head((abs(states) > .1) + 0)
```

```
Catoptrophorus_semipalmatus         Calidris_ferruginea
                          0                           1
           Calidris_canutus            Calidris_maritima
                          1                           1
         Calidris_acuminata               Calidris_mauri
                          1                           1
```

The simplest way of working with this tree is to use the `clades.from.polytomies` function. This collapses all daughters of any polytomy into a clade. (Be careful - if you have a polytomy at the base of your tree, the entire tree will collapse into a single clade!) This can be visualised with `plot` functions as normal – see `?plot.clade.tree` for more information. This tree can then be passed into `make.bisse`, along with a plain vector of state names. The catch is that every taxon still needs state information, not just those at the tips. Our state vector `states` includes states for all 350 species, so we are OK to use this.

```
states.15 <- (abs(states) > 0.15) + 0
tree.clade <- clades.from.polytomies(tree)
lik <- make.bisse(tree.clade, states.15)
```

```
par(mar=rep(.5, 4), mfrow=c(1, 2))
plot(tree, show.tip.label=FALSE)
tree.clade <- clades.from.polytomies(tree)
plt <- plot(tree.clade, show.tip.label=FALSE, clade.fill="gray")
```

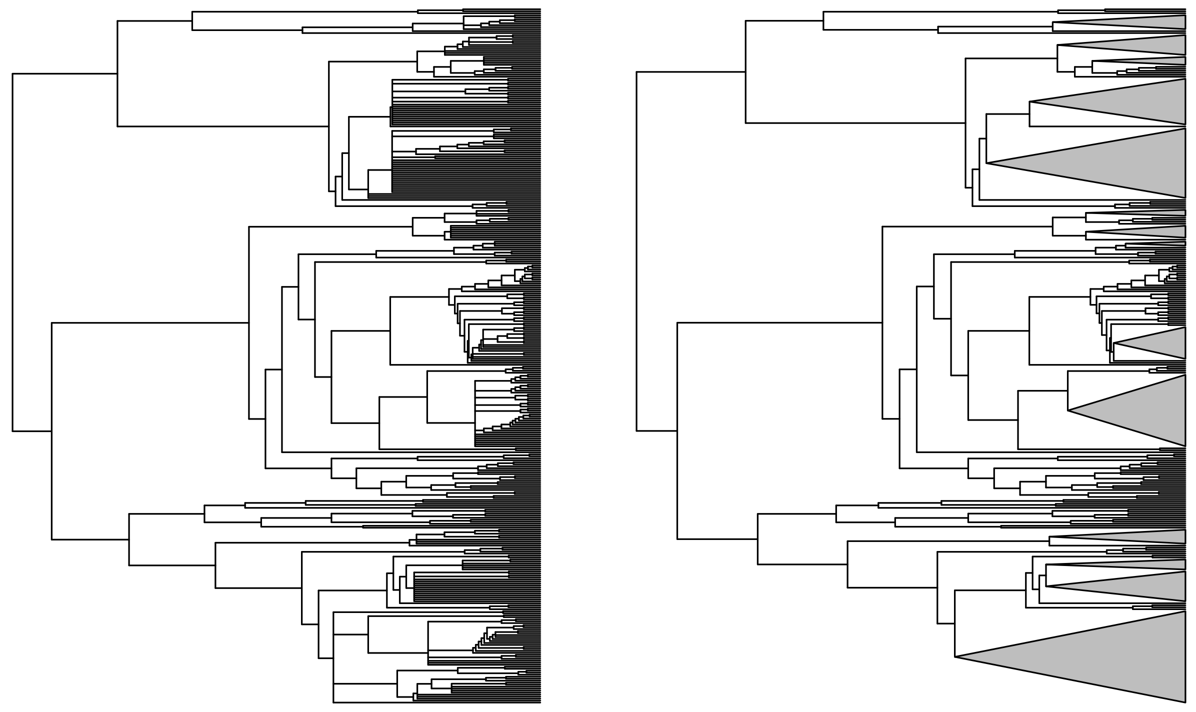
Figure 7: The shorebird supertree (Thomas et al., 2004). On the left, the original tree with all polytomies and 350 species. On the right, the polytomies have been collapsed, to leave 135 tips, with tips representing from 1 to 47 species.

A sensible starting point can still be computed with `starting.point.bisse`; this takes into account the clade information automatically.

```
p <- starting.point.bisse(tree.clade)
```

You can then perform a ML search or MCMC analysis as usual. However, unresolved clades slow down the analysis considerably, and this will take several minutes.

```
fit.full <- find.mle(lik, p)
```

In the full model, speciation rates for dimorphic species (state 1) are greater than those for monomorphic species, but character transition rates away from dimorphism (q_{10}) are greater than the reverse transition (q_{10}):

```
round(coef(fit.full), 3)
```

```
lambda0 lambda1     mu0     mu1     q01     q10
  0.097   0.090   0.000   0.062   0.015   0.017
```

To test whether these differences are significant, we can use a likelihood ratio test. First, construct the reduced models, constraining $\lambda_1 \sim \lambda_0$ or $q_{10} \sim q_{01}$:

```
lik.l <- constrain(lik, lambda1 ~ lambda0)
lik.q <- constrain(lik, q10 ~ q01)
```

Then, rerun the

```
fit.l <- find.mle(lik.l, p[argnames(lik.l)])
fit.q <- find.mle(lik.q, p[argnames(lik.q)])
```

These constrained models are significantly worse fits than the full model ($p \approx 0.04$ for both).

```
anova(fit.full, equal.l=fit.l, equal.q=fit.q)
```

```
        Df   lnLik    AIC    ChiSq Pr(>|Chi|)
full     6 -635.78 1283.5
equal.l  5 -635.80 1281.6 0.040928     0.8397
equal.q  5 -635.79 1281.6 0.024380     0.8759
```

The analysis can also be run with MCMC. Here, I am using an exponential prior with rate $1/(2r)$, as earlier.

```
prior <- make.prior.exponential(1 / (2*p[1]))
```

The MCMC itself takes a very long time to run (approximately 4–5 s/sample)

```
set.seed(1)
samples <- mcmc(lik, coef(fit.full), 10000, w=.3, prior=prior,
                print.every=0)
```

Setting up the likelihood function above assumed that the tree being used contained polytomies, and this is the source of the unresolved clades. However, it is probably more common to have an "exemplar" tree, where the unresolved species were never included in the first place. The tree `Thomas-tree-exemplar.nex` (which was derived from the tree above) does not contain any reference the species that are contained within the unresolved clades.

```
tree.ex <- read.nexus("data/Thomas-tree-exemplar.nex")
states.ex <- states.15[tree.ex$tip.label]
names(states.ex) <- tree.ex$tip.label
```

We need to define a `data.frame` with information about the unresolved clades. The file `Thomas-unresolved.csv` contains the information in the correct format:

```
unresolved <- read.csv("data/Thomas-unresolved.csv", as.is=TRUE)
head(unresolved)
```

```
        tip.label Nc n0 n1
1 Catoptrophorus 47 34 10
2      Gallinago 16 10  1
3       Scolopax  6  1  1
4       Numenius  8  4  4
5         Sterna 37 20  1
6          Larus 17  5  9
```

All the columns here are required:

- `tip.label`: the tip label within the tree
- `Nc`: the total number of species that the tip represents
- `n0`: the number of species known to be in state `0`
- `n1`: the number of species known to be in state 1

(additional columns are fine and will be silently ignored). Note that `Nc` can be greater than `n0 + n1`: this allows for species with unknown state.

This `unresolved` object is passed into the `make.bisse` function:

```
lik.ex <- make.bisse(tree.ex, states.ex, unresolved=unresolved)
```

This likelihood function should be identical to the one created above.

```
lik.ex(coef(fit.full))
```

```
[1] -635.7756
```

```
lik(coef(fit.full))
```

```
[1] -635.7756
```

5 Multiple state characters and diversification: MuSSE

MuSSE (Multiple State Speciation and Extinction) generalises the BiSSE model to allow characters with more than two states. Following Pagel (1994), this can also allow for multiple characters, each of which might be binary, by recoding the states.

To illustrate, we'll start with a simple simulated example with a three-level state. The tree is simulated where character evolution is only possible among neighbouring states (i.e., $1 \to 3$ and $3 \to 1$

transitions are disallowed). All other transitions are equal, and both speciation and extinction rates increase as the character number increases. For a three state case, the parameter vector is in the order $(\lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3, q_{12}, q_{13}, q_{21}, q_{23}, q_{31}, q_{32})$. This order can be seen here (sorry – clunky at the moment)

```
diversitree:::default.argnames.musse(3)
```

```
 [1] "lambda1" "lambda2" "lambda3" "mu1"     "mu2"     "mu3"     "q12"
 [8] "q13"     "q21"     "q23"     "q31"     "q32"
```

(the order of the q parameters is row-wise through the transition rate matrix, skipping diagonal elements).

Simulate a 30 species tree, with the tree starting in state 1.

```
pars <- c(.1,  .15,  .2,  # lambda 1, 2, 3
          .03, .045, .06, # mu 1, 2, 3
          .05, 0,         # q12, q13
          .05, .05,       # q21, q23
          0,   .05)       # q31, q32
set.seed(2)
phy <- tree.musse(pars, 30, x0=1)
```

The tree and its real character history are shown in figure

The states are numbered 1, 2, 3, rather than 0, 1 in BiSSE.

```
states <- phy$tip.state
table(states)
```

```
states
 1  2  3
13 13  4
```

Making a likelihood function is basically identical to BiSSE. The third argument needs to be the number of states. In a future version this will probably be `max(states)`, but there are some pitfalls about this that I am still worried about.

```
lik <- make.musse(phy, states, 3)
```

The argument names here are in the same order as for the simulation. Just adding one more state (compared with BiSSE) has moved us up to 10 parameters.

```
argnames(lik)
```

```
 [1] "lambda1" "lambda2" "lambda3" "mu1"     "mu2"     "mu3"     "q12"
 [8] "q13"     "q21"     "q23"     "q31"     "q32"
```

Rather than start with the full model, and constrain things, here I will start with a very simple model and expand. This model has all λ_i, μ_i, and q_i the same (except for q_{13} and q_{31}, which are still zero).

```
lik.base <- constrain(lik, lambda2 ~ lambda1, lambda3 ~ lambda1,
                      mu2 ~ mu1, mu3 ~ mu1,
                      q13 ~ 0, q21 ~ q12, q23 ~ q12, q31 ~ 0, q32 ~ q12)
argnames(lik.base)
```

```
col <- c("#eaab00", "#004165", "#618e02")
h <- history.from.sim.discrete(phy, 1:3)
plot(h, phy, cex=1, col=col, no.margin=TRUE, font=1)
```

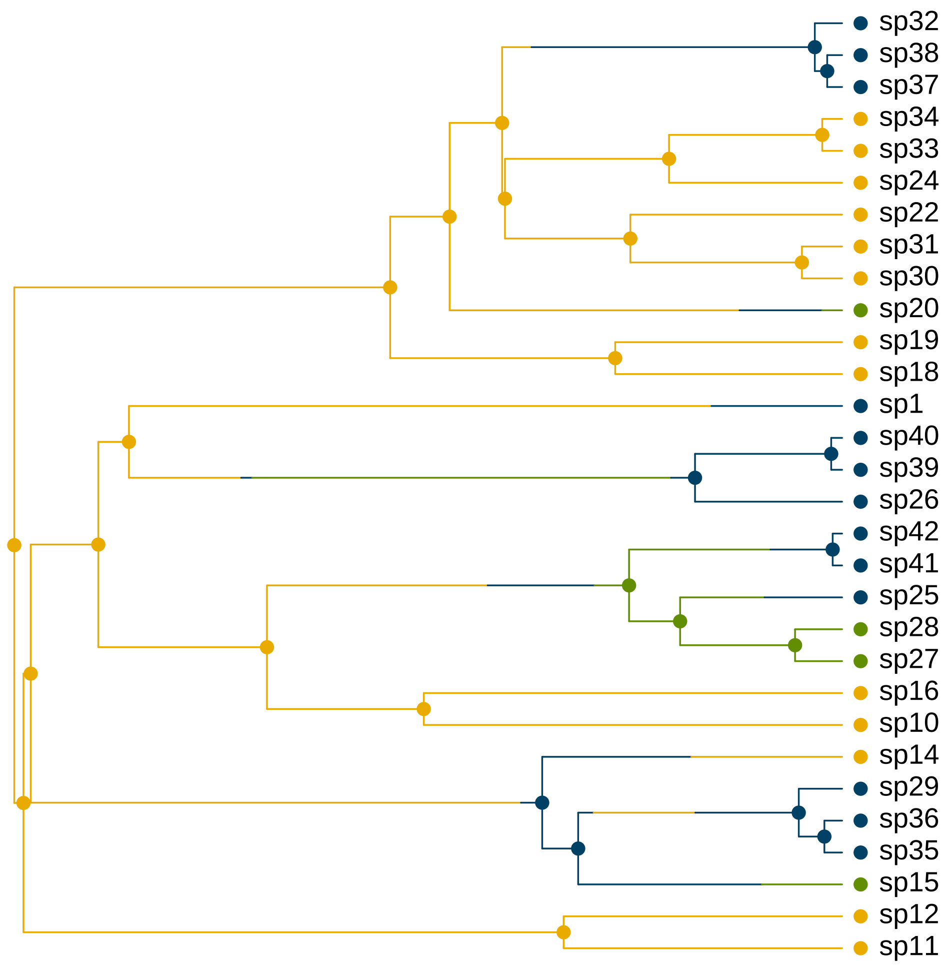

Figure 8: Simulated MuSSE tree. Yellow is state 1, blue is state 2, and green is state 3

```
[1] "lambda1" "mu1"     "q12"
```

Find the ML point for this model

```
p <- starting.point.musse(phy, 3)
fit.base <- find.mle(lik.base, p[argnames(lik.base)])
```

Now, allow the speciation rates to vary

```
lik.lambda <- constrain(lik, mu2 ~ mu1, mu3 ~ mu1,
                        q13 ~ 0, q21 ~ q12, q23 ~ q12, q31 ~ 0, q32 ~ q12)
fit.lambda <- find.mle(lik.lambda, p[argnames(lik.lambda)])
```

There is very little improvement here (this is a small tree)

```
anova(fit.base, free.lambda=fit.lambda)
```

```
            Df   lnLik    AIC   ChiSq Pr(>|Chi|)
minimal      3 -110.64 227.27
free.lambda  5 -110.21 230.42 0.84603     0.6551
```

6 Quantitative traits and diversification: QuaSSE

The QuaSSE method is by far the slowest method in diversitree. This is because to compute the likelihood, we have to solve a system of partial differential equations, which is substantially more complicated than the system of ordinary differential equations or simple algebraic expressions in the other methods. The basic interface is very similar to the other methods, but there are more options to control the behaviour of the integrator.

We'll start with a simulated tree. The tree simulation differs slightly from the other methods, because there is no longer a canonical argument list (speciation and extinction rates are arbitrary functions of the character state). Here is a set of functions; speciation rate is a sigmoidal function which ranges from 0.1 to 0.2 with an inflection point at $x = 0$, extinction is constant at rate 0.03, and the model of character evolution is Brownian motion with diffusion parameter 0.025.

```
lambda <- function(x) sigmoid.x(x, 0.1, 0.2,  0, 2.5)
mu <- function(x) constant.x(x, 0.03)
char <- make.brownian.with.drift(0, 0.025)
```

Simulate the tree:

```
set.seed(1)
phy <- tree.quasse(c(lambda, mu, char), max.taxa=15, x0=0,
                   single.lineage=FALSE)
```

We need to specify the standard deviation for the states; here I will just assume that all taxa have a state standard deviation of 1/200

```
states <- phy$tip.state
states.sd <- 1/200
```

Then, build the likelihood as usual. The difference compared with other models is that we have to specify the speciation and extinction functions (here, `sigmoid.x` and `constant.x`, respectively). There are a number of other provided functions (see `?constant.x` for a list), but any function that takes `x` as the first argument may be used.

```
lik <- make.quasse(phy, states, states.sd, sigmoid.x, constant.x)
```

This can be used in ML calculations as usual. There is a `starting.point.quasse` function that may be useful in selecting sensible starting points, but some effort is still required to convert this into a full vector as it just returns constant rate speciation, extinction, and diffusion rates.

```
p <- starting.point.quasse(phy, states)
p
```

```
    lambda         mu  diffusion
0.16107838 0.02569057 0.03164062
```

Let's ignore drift: the argument list we need is:

```
lik.nodrift <- constrain(lik, drift ~ 0)
argnames(lik.nodrift)
```

```
[1] "l.y0"      "l.y1"      "l.xmid"    "l.r"       "m.c"       "diffusion"
```

A sensible starting point here might be

```
p.start <- c(p[1], p[1], mean(states), 1, p[2:3])
names(p.start) <- argnames(lik.nodrift)
p.start
```

```
      l.y0       l.y1     l.xmid        l.r        m.c  diffusion
0.16107838 0.16107838 0.57097062 1.00000000 0.02569057 0.03164062
```

Lower bounds:

```
lower <- c(0, 0, min(states), -Inf, 0, 0)
```

Then run `find.mle`, as usual. The `control` argument here just tells the subplex algorithm to use an initial step size of 0.1 (rather than 1), which reduces the number of function evaluations somewhat.

```
fit <- find.mle(lik.nodrift, p.start, control=list(parscale=.1),
                lower=lower, verbose=0)
```

Compare this against the constant rate speciation fit:

```
lik.constant <- constrain(lik.nodrift, l.y1 ~ l.y0, l.xmid ~ 0, l.r ~ 1)
fit.constant <- find.mle(lik.constant,
                         p.start[argnames(lik.constant)],
                         control=list(parscale=.1), lower=0,
                         verbose=0)
```

and compare the models – no significant difference, which is not surprising with only a 15 species tree.

```
anova(fit, constant=fit.constant)
```

```
         Df   lnLik    AIC  ChiSq Pr(>|Chi|)
full      6 -51.734 115.47
constant  3 -55.152 116.30 6.8366    0.07729
```

6.1 Primate analysis

Here, I will recreate the analysis of primate diversification from (FitzJohn, 2010), but fitting only speciation functions to keep things relatively simple.

```
phy <- read.nexus("data/Vos-2006.nex")
d <- read.csv("data/Redding-2010.csv")
mass <- log(d$mass)
names(mass) <- d$tip.label
```

Assume standard deviation of 1/50 for all species – this comes from nowhere in particular, and is probably over-confident in the mass estimates for most species.

```
mass.sd <- 1/50
```

Starting point parameter estimates, as above:

```
p <- starting.point.quasse(phy, mass)
p
```

```
    lambda         mu  diffusion
0.19072726 0.11034810 0.03251953
```

Create a piecewise "linear" function. This is linear in the range [(xr[1]), xr[2]], and flat outside this range; this satisfies the condition that the derivatives of the speciation and extinction function with respect to the character state approach zero at the edges of the modelled parameter space.

```
xr <- range(mass) + c(-1,1) * 20 * p["diffusion"]
linear.x <- make.linear.x(xr[1], xr[2])
```

Because we are going to create a reasonable number of models, here is a function that simplifies this, requiring just speciation and extinction functions:

```
make.primates <- function(lambda, mu)
  make.quasse(phy, mass, mass.sd, lambda, mu)
```

and a function that constrains drift to zero

```
nodrift <- function(f)
  constrain(f, drift ~ 0)
```

Create the likelihood functions where speciation is a constant, linear, sigmoidal, or hump-shaped function of log body mass.

```
f.c <- make.primates(constant.x, constant.x)
f.l <- make.primates(linear.x, constant.x)
f.s <- make.primates(sigmoid.x, constant.x)
f.h <- make.primates(noroptimal.x, constant.x)
```

Start by fitting the constant model first (this will take quite a while; each function evaluation takes about 5 s).

```
control <- list(parscale=.1, reltol=0.001)
mle.c <- find.mle(nodrift(f.c), p, lower=0, control=control,
                  verbose=0)
```

Starting points for the constrained analyses based on this constrained fit.

```
p.c <- mle.c$par
p.l <- c(p.c[1], l.m=0, p.c[2:3])
p.s <- p.h <- c(p.c[1], p.c[1], mean(xr), 1, p.c[2:3])
names(p.s) <- argnames(nodrift(f.s))
names(p.h) <- argnames(nodrift(f.h))
```

```
mle.l <- find.mle(nodrift(f.l), p.l, control=control, verbose=0)
mle.s <- find.mle(nodrift(f.s), p.s, control=control, verbose=0)
mle.h <- find.mle(nodrift(f.h), p.h, control=control, verbose=0)
```

The fits can then be compared. These are all against the constant speciation rate fit (listed as "full" in the table). The support is strongest for the "hump" shaped fit.

```
anova(mle.c, linear=mle.l, sigmoidal=mle.s, hump=mle.h)
```

```
          Df   lnLik    AIC  ChiSq Pr(>|Chi|)
minimal    3 -841.40 1688.8
linear     4 -836.07 1680.1 10.666  0.0010911
sigmoidal  6 -832.60 1677.2 17.598  0.0005323
hump       6 -829.00 1670.0 24.799  1.701e-05
```

Run the fits with the drift parameter added, starting from the constrained model's ML parameters:

```
mle.d.l <- find.mle(f.l, coef(mle.l, TRUE), control=control, verbose=0)
mle.d.s <- find.mle(f.s, coef(mle.s, TRUE), control=control, verbose=0)
mle.d.h <- find.mle(f.h, coef(mle.h, TRUE), control=control, verbose=0)
```

and add these to the comparison:

```
anova(mle.c, linear=mle.l, sigmoidal=mle.s, hump=mle.h,
      drift.linear=mle.d.l, drift.sigmoidal=mle.d.s,
      drift.hump=mle.d.h)
```

```
                Df   lnLik    AIC  ChiSq Pr(>|Chi|)
minimal          3 -841.40 1688.8
linear           4 -836.07 1680.1 10.666  0.0010911
sigmoidal        6 -832.60 1677.2 17.598  0.0005323
hump             6 -829.00 1670.0 24.799  1.701e-05
drift.linear     5 -832.50 1675.0 17.806  0.0001360
drift.sigmoidal  7 -830.27 1674.5 22.267  0.0001774
drift.hump       7 -825.37 1664.7 32.071  1.850e-06
```

There is an improvement in the model fit when drift is added. In all cases, the drift parameter is positive, indicating an increase in mass.

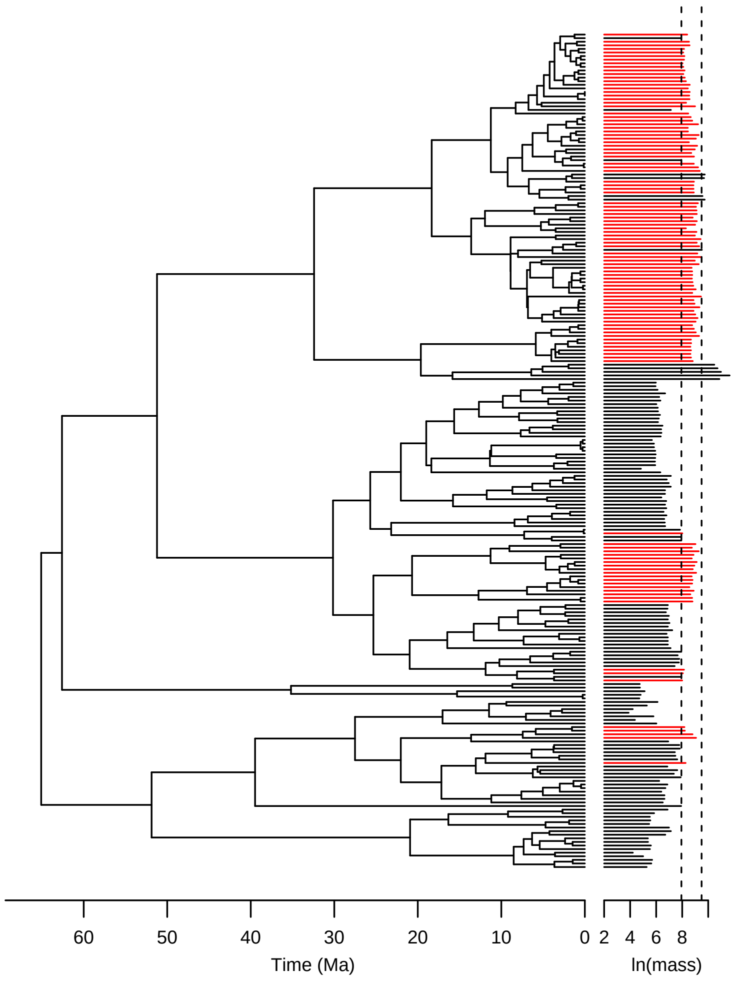

Figure 9: The primate tree from Vos and Mooers (2006). Log body size (in grams) is shown by the horizontal bar for each species (data from Redding et al., 2010). The vertical dashed lines indicate the approximate ranges of body masses in which elevated speciation rates were inferred, and extant species whose mass falls in this range have their mass coloured red.

```
c(linear=coef(mle.d.l)[["drift"]],
  sigmoidal=coef(mle.d.s)[["drift"]],
  hump=coef(mle.d.h)[["drift"]])
```

```
    linear  sigmoidal       hump
0.10458477 0.06759282 0.07929087
```

However, the precise estimate depends strongly on the speciation model assumed.

Next, consider splitting the tree. MEDUSA (Alfaro et al., 2009) identified a shift in diversification rates at the base of the superfamily Cercopithecoidea (old world monkeys). In this tree, this corresponds to node 153. It's easiest to work with node names, so I'm going to add some here

```
phy$node.label <- paste("nd", 1:phy$Nnode, sep="")
```

Then, we can construct "split" QuaSSE objects. For the constant speciation rate case:

```
f.cc <- make.quasse.split(phy, mass, mass.sd, constant.x, constant.x,
                          "nd153", Inf)
argnames(f.cc)
```

```
[1] "l.c.1"       "m.c.1"       "drift.1"     "diffusion.1" "l.c.2"
[6] "m.c.2"       "drift.2"     "diffusion.2"
```

Here, the speciation and extinction functions may either be a single function (as above), in which case all partitions get the same function for speciation and extinction. Alternatively, lists of functions can be added, in which case different partitions may have different functions. The `Inf` indicates that the split should go at the base of the internal edge subtending node `nd153`, which is consistent with MEDUSA. Passing in `0` would put it immediately prior to the node, and any other numeric value would place it at that point in time from the present.

The first set of parameters refers to the "background" group, the second refers refers to the "foreground" clade rooted at `nd153`.

To simplify things, we will constrain drift to be zero and assume that both partitions have the same diffusion coefficient.

```
g.cc <- constrain(f.cc, drift.1 ~ 0, drift.2 ~ 0,
                  diffusion.2 ~ diffusion.1)
argnames(g.cc)
```

```
[1] "l.c.1"       "m.c.1"       "diffusion.1" "l.c.2"       "m.c.2"
```

Generate a starting point from the single partition ML point:

```
p.cc <- c(p.c, p.c[1:2])
names(p.cc) <- argnames(g.cc)
```

At this point, the split function should have basically the same likelihood as the single partition function:

```
mle.c$lnLik - g.cc(p.cc)
```

And run the ML search

```
mle.cc <- find.mle(g.cc, p.cc, control=control, lower=0, verbose=0)
```

Repeat this for linear speciation functions:

```
f.ll <- make.quasse.split(phy, mass, mass.sd, linear.x, constant.x,
                          "nd153", Inf)
g.ll <- constrain(f.ll, drift.1 ~ 0, drift.2 ~ 0,
                  diffusion.2 ~ diffusion.1)
g.lc <- constrain(g.ll, l.m.2 ~ 0)
g.cl <- constrain(g.ll, l.m.1 ~ 0)
```

Generate a starting points: start with the function where both speciation rates are linear functions.

```
p.cc <- coef(mle.cc)
p.ll <- c(p.cc[1], 0, p.cc[2:4], 0, p.cc[5])
names(p.ll) <- argnames(g.ll)
```

Run the ML searches for this model:

```
mle.ll <- find.mle(g.ll, p.ll, control=control, verbose=0)
```

Then generate starting points for models with just one of the sections of the tree having a linear speciation function:

```
p.lc <- c(coef(mle.ll)[1:3], p.ll[c(4, 5, 7)])
p.cl <- c(p.ll[c(1, 3, 4)], coef(mle.ll)[5:7])
```

and run the ML search:

```
mle.lc <- find.mle(g.lc, p.lc, control=control, verbose=0)
mle.cl <- find.mle(g.cl, p.cl, control=control, verbose=0)
```

We can then compare the models again:

```
anova(mle.c, linear=mle.l, sigmoidal=mle.s, hump=mle.h,
      part.constant=mle.cc,
      part.linear.bg=mle.lc,
      part.linear.fg=mle.cl,
      part.linear=mle.ll)
```

```
               Df   lnLik    AIC  ChiSq Pr(>|Chi|)
minimal         3 -841.40 1688.8
linear          4 -836.07 1680.1 10.666  0.0010911
sigmoidal       6 -832.60 1677.2 17.598  0.0005323
hump            6 -829.00 1670.0 24.799  1.701e-05
part.constant   5 -828.59 1667.2 25.622  2.731e-06
part.linear.bg  6 -828.26 1668.5 26.290  8.291e-06
part.linear.fg  6 -826.02 1664.0 30.771  9.498e-07
part.linear     7 -828.32 1670.7 26.160  2.939e-05
```

This supports the model with a linear “foreground” rate of speciation (lowest AIC value). Looking at the coefficients for this model:

```
coef(mle.cl)
```

```
      l.c.1       m.c.1  diffusion.1       l.c.2       l.m.2       m.c.2
0.141392843  0.068877966  0.031964900  1.972507892 -0.193181732  0.005147044
```

The speciation rate in the foreground clade is a negative function of body size (`l.m.2` is negative) – increasing body size decreases the speciation rate.

6.2 Controlling the calculations

There are a number of ways to control the behaviour of the integrator: most of these are a speed/accuracy trade-off. These should be supplied in the list passed in as the argument `control`. The `make.quasse` function attempts to select defaults that result in acceptable performance and accuracy, but this is not always possible.

- `method`: one of `fftC` or `fftR` to switch between C (fast) and R (slow) back-ends for the integration. Both use non-adaptive fft-based convolutions. Eventually, an adaptive methods-of-lines approach will be available.
- `dt.max`: Maximum time step to use for the integration. By default, this will be set to 1/1000 of the tree depth. Smaller values will slow down calculations, but improve accuracy.
- `nx`: The number of bins into which the character space is divided (default=1024). Larger values will be slower and more accurate. For the `fftC` integration method, this should be an integer power of 2 (512, 2048, etc).
- `r`: Scaling factor that multiplies `nx` for a "high resolution" section at the tips of the tree (default=4, giving a high resolution character space divided into 4096 bins). This helps improve accuracy while possibly tight initial probability distributions flatten out as time progresses towards the root. Larger values will be slower and more accurate. For the `fftC` integration method, this should be a power of 2 (2, 4, 8, so that `nx*r` is a power of 2).
- `tc`: where in the tree to switch to the low-resolution integration (zero corresponds to the present, larger numbers moving towards the root). By default, this happens at 10% of the tree depth. Smaller values will be faster, but less accurate.
- `xmid`: Mid point to centre the character space. By default this is at the mid point of the extremes of the character states.
- `tips.combined`: Get a modest speed-up by simultaneously integrating all tips? By default, this is `FALSE`, but speedups of up to 25% are possible with this set to `TRUE`.
- `w`: Number of standard deviations of the normal distribution induced by Brownian motion to use when doing the convolutions (default=5). Probably best to leave this one alone.

7 Geographic distributions and diversification: GeoSSE (by Emma Goldberg)

The GeoSSE model (Geographic State Speciation and Extinction) combines features of the constant-rates birth-death model with a three-state Markov model. It differs from BiSSE in parameterising the model to represent diversification and range shifts among two regions, which includes allowing widely-distributed species whose ranges may change in conjunction with a speciation event. See Goldberg et al. (2011) for a full explanation of the model assumptions.

7.1 Parameters and a tree

Simulating trees under GeoSSE is not yet part of `diversitree`, but it can be done with a separate program, SimTreeSDD. A simulated tree is included and shown in Figure 10.

```
data("geosse")
phy <- geosse.phy
```

```
statecols <- c("AB"="purple", "A"="blue", "B"="red")
plot(phy, tip.color=statecols[phy$tip.state+1], cex=0.5)
```

An extremely crude starting point for parameter estimation can be obtained with:

```
p <- starting.point.geosse(phy)
p
```

```
      sA       sB      sAB       xA       xB       dA       dB
2.018992 2.018992 2.018992 1.009496 1.009496 1.009496 1.009496
```

The parameters are: speciation within region A (`sA`), speciation within region B (`sB`), between-region speciation (`sAB`), extinction from region A (`xA`), extinction from region B (`xB`), dispersal from A to B (range expansion, `dA`), and dispersal from B to A (`dB`).

7.2 Model construction and constraining

Constructing and constraining likelihood functions works as for the other models. Here we will consider the full model, a model without between-region speciation, and a model without regional dependence of speciation or extinction rates.

```
lik1 <- make.geosse(phy, phy$tip.state)
lik2 <- constrain(lik1, sAB ~ 0)
lik3 <- constrain(lik1, sA ~ sB, xA ~ xB)
```

7.3 Maximum likelihood

ML parameter estimation and model comparisons:

```
ml1 <- find.mle(lik1, p)
p <- coef(ml1)
ml2 <- find.mle(lik2, p[argnames(lik2)])
ml3 <- find.mle(lik3, p[argnames(lik3)])
```

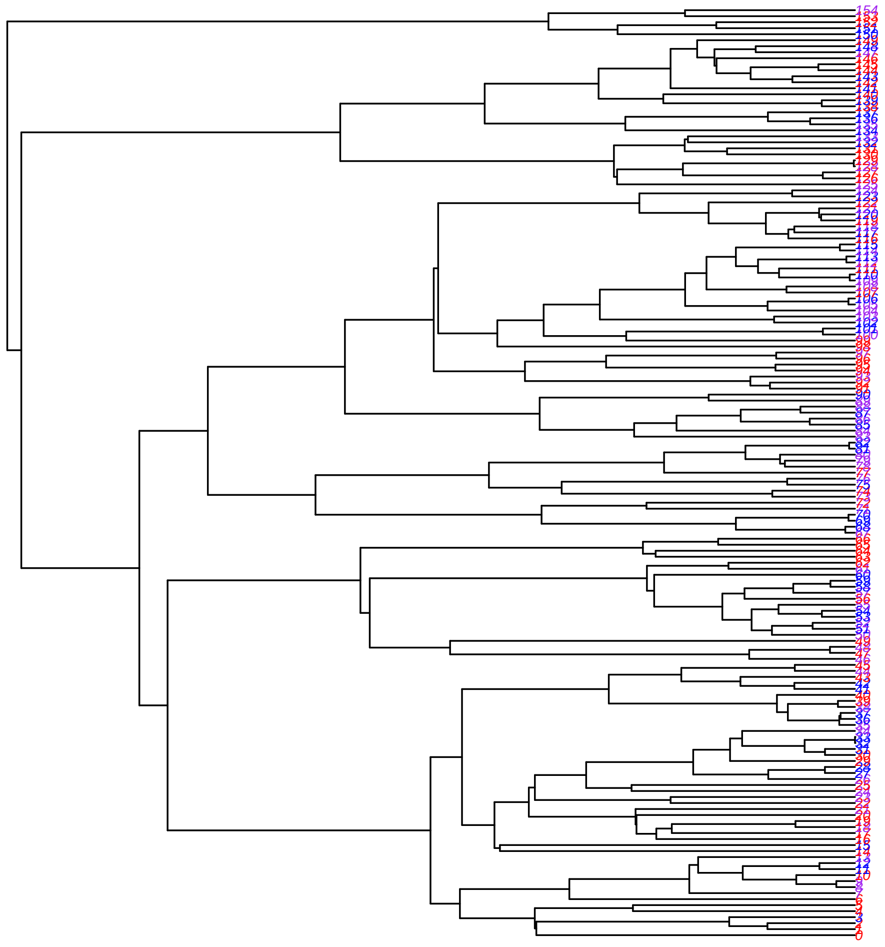

Figure 10: A GeoSSE tree simulated with `params = c(1.5, 0.5, 1.0, 0.7, 0.7, 2.5, 0.5)`. The tip state colors are purple for species present in both regions (AB), blue for species only in region A, and red for species only in region B.

```
round(rbind(full = coef(ml1),
            no.sAB = coef(ml2, TRUE),
            eq.div = coef(ml3, TRUE)), 3)
```

```
          sA    sB   sAB    xA    xB    dA    dB
full   1.448 0.330 0.450 0.713 1.083 3.098 0.000
no.sAB 1.741 0.354 0.000 1.123 1.415 3.372 0.000
eq.div 0.812 0.812 0.556 0.263 0.263 1.566 0.636
```

```
anova(ml1, no.sAB = ml2, eq.div = ml3)
```

```
       Df   lnLik    AIC   ChiSq Pr(>|Chi|)
full    7 -295.06 604.13
no.sAB  6 -295.61 603.23  1.1037   0.293450
eq.div  5 -300.94 611.88 11.7544   0.002803
```

On this tree, we reject the model of equal speciation and extinction in the two regions, concluding that there are regional differences in diversification. Including the between-region mode of speciation does not, however, significantly improve the fit.

7.4 Markov chain Monte Carlo

We will only consider the 6-parameter model here. Use the ML rate estimates as a starting point. Place a broad exponential prior on each parameter.

```
p <- coef(ml2)
prior <- make.prior.exponential(1/2)
```

Use a pilot run to obtain reasonable step sizes:

```
set.seed(1)
tmp <- mcmc(lik2, p, nsteps=100, prior=prior, w=1, print.every=0)
w <- diff(sapply(tmp[2:7], quantile, c(0.025, 0.975)))
```

Now the real analysis, which will take awhile to run:

```
mcmc2 <- mcmc(lik2, p, nsteps=10000, prior=prior, w=w)
```

Marginal posterior distributions are shown in Figure 11. We can also compare rate estimates by looking at posterior probabilities of their differences.

```
mcmc2diff <- with(mcmc2, data.frame(s.diff=sA-sB,
                                    x.diff=xA-xB,
                                    d.diff=dA-dB,
                                    div.A=sA-xA,
                                    div.B=sB-xB))
colMeans(mcmc2diff > 0)
```

```
s.diff x.diff d.diff  div.A  div.B
1.0000 0.7529 0.9564 0.9388 0.2256
```

We correctly and confidently recover regional biases in speciation and dispersal, and positive net diversification in region A. With less confidence, we recover regional bias in extinction and negative net diversification for region B (i.e., extinction exceeds speciation in region B).

```
col1 <- c("red", "orange", "blue", "purple", "black", "gray")
col2 <- col1[c(1,3,5)]
mcmc2diff <- with(mcmc2, data.frame(s.diff=sA-sB, x.diff=xA-xB, d.diff=dA-dB))
par(mfrow=c(2,1), mar=c(3, 4, 0, 1))
profiles.plot(mcmc2[2:7], col.line=col1, xlab="", ylab="")
legend("topright", argnames(lik2), col=col1, lty=1)
profiles.plot(mcmc2diff, col.line=col2, xlab="", ylab="")
legend("topright", colnames(mcmc2diff), col=col2, lty=1)
title(xlab="rate", ylab="posterior probability density", outer=T, line=-1)
```

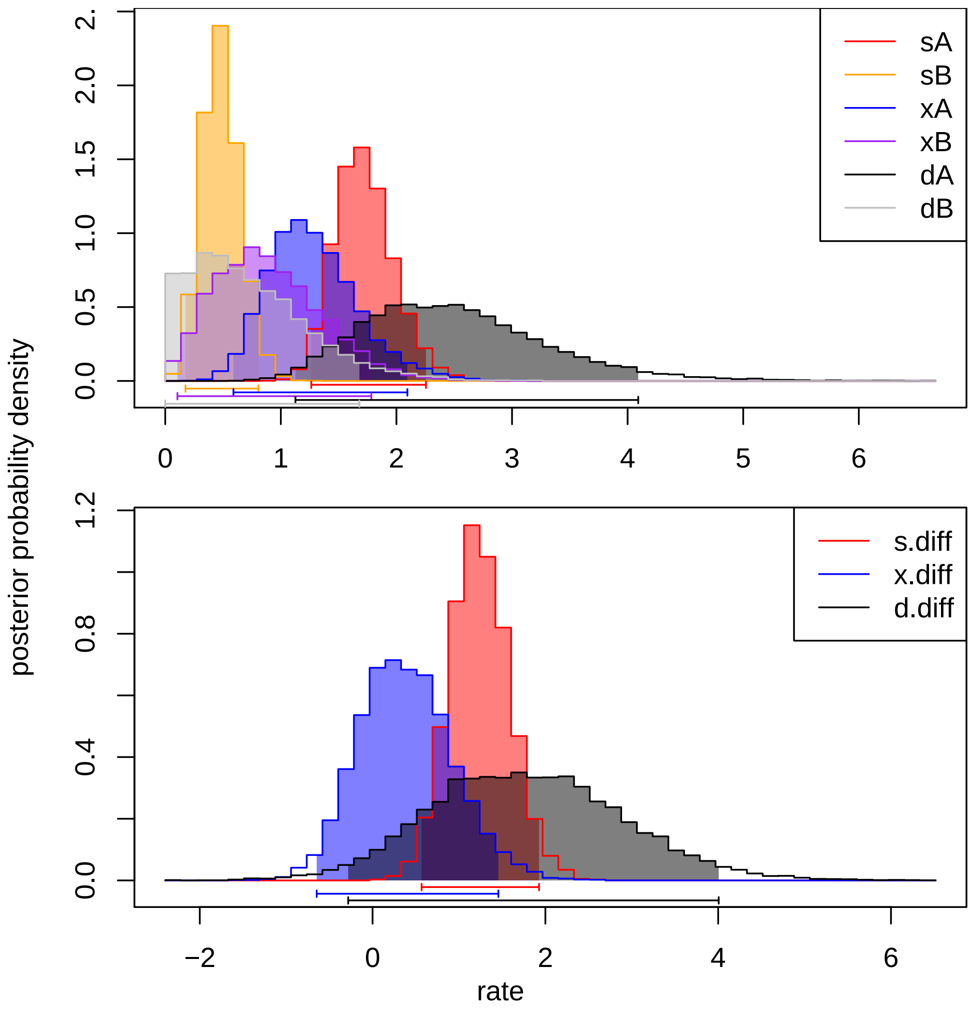

Figure 11: Posterior probability distributions for the six-rate GeoSSE model, for the tree shown in Figure 10. Uncertainty is largest for dispersal and smallest for speciation. Regional differences in speciation and, to some extent, in dispersal are recovered.

7.5 Additional options

GeoSSE likelihood functions can be built with randomly-incomplete sampling (FitzJohn et al., 2009). There is not currently support for unresolved clades.

```
p <- coef(ml1)
lik1(p)
```

```
[1] -295.0626
```

```
lik4 <- make.geosse(phy, phy$tip.state, sampling.f=c(0.9, 0.6, 0.4))
lik4(p)
```

```
[1] -309.3042
```

When using the likelihood function, one can condition on survival of the clade (not done by default):

```
lik4(p, condition.surv=TRUE)
```

```
[1] -309.3042
```

External information about the geographic distribution of the common ancestor of the clade can be enforced by fixing the root state. For example, if you are absolutely positive that the MRCA was found only in region B:

```
lik4(p, root.p=c(0,0,1), root=ROOT.GIVEN)
```

```
[1] -340.7872
```

Use this procedure with caution, and only in the face of truly external data, e.g., fossil or geologic information.

7.6 Multi-clade analysis

Some applications of GeoSSE have combined multiple clades into a single analysis (Anacker et al., 2010; Goldberg et al., 2011). This has the advantage of providing a larger dataset and hence presumably more power, but it is important to keep in mind the assumptions that go into such an analysis. First, it treats all clades as evolving according to the same model, with the same values for the rate parameters. This can be tested by first fitting the clades individually, or it may be inherent to the hypothesis at hand. Second, it assumes that the clades are independent of each other, since their likelihoods are simply multiplied together to form a joint likelihood function. In some cases, this may be a defensible approximation, for example, when the clades being considered are only very distantly related. If you have decided that you can convince yourself and your reviewers that a multi-clade analysis is appropriate, here is one way to do it.

Assemble the data as lists of trees and character state vectors. Each list element is one clade. Here we will use two trees from the chaparral study, one each from the posterior sets for *Ceanothus* and *Arctostaphylos*. The trees `phy.cea` and `phy.arc` and the tip states `chars.cea` and `chars.arc` (A = chaparral, B = forest) were read in with the call to `data("geosse")` above.

```
phy.cea <- read.tree(file="data/cea.tre")
phy.arc <- read.tree(file="data/arc.tre")
temp <- read.csv("data/cea.dat", header=FALSE, as.is=TRUE)
chars.cea <- structure(temp[[2]], names=temp[[1]])
temp <- read.csv("data/arc.dat", header=FALSE, as.is=TRUE)
chars.arc <- structure(temp[[2]], names=temp[[1]])
rm(temp)
```

Create an individual likelihood function for each clade:

```
trees <- list(phy.cea, phy.arc)
states <- list(chars.cea, chars.arc)
sampl <- list(c(0.913, 0.941, 0.875),
              c(0.674, 0.533, 0.750))
liks <- mapply(make.geosse, trees, states, sampl)
```

Now define the joint, multi-clade likelihood function:

```
lnL.multi <- combine(liks)
```

Now `lnL.multi()` can be used like any other likelihood function, for example, it can be constrained:

```
lnL.multi.constrained <- constrain(lnL.multi, sAB ~ 0)
```

evaluated at arbitrary points:

```
p <- c(0.19, 0.08, 0.00, 0.29, 0.48, 1.29, 0.87)
lnL.multi(p)
```

```
[1] -383.4771
```

or used in `find.mle()` or `mcmc()`.

8 Topics not covered

I have not covered several included models here, but information can be found in the online documentation.

- MEDUSA-style "split" models. Following (Alfaro et al., 2009), these models allow different regions of the tree to have different parameters. This is implemented for birth-death models (`make.bd.split`, which is identical to MEDUSA), BiSSE (`?make.bisse.split`), and MuSSE (`?make.musse.split`).
- Time-dependent models. In these, time is divided into "epochs", each of which may have different parameters. This is implemented for BiSSE (`?make.bisse.td`) and MuSSE (`?make.musse.td`). Another model of time-dependence is also possible, where rates are arbitrary functions of time: implemented for plain birth-death models (`?make.bd.t`), BiSSE (`?make.bisse.td`) and MuSSE (`?make.musse.td`).

- Brownian motion. A very simple-minded Brownian motion likelihood calculation is included. This allows estimation of the diffusion parameter of a Brownian motion process for the evolution of a single continuous trait (`?make.bm`).
- Stochastic character mapping. Currently implemented only for Markov models of discrete character evolution (?asr.stoch.mkn).

References

Alfaro M.E., Santini F., Brock C., Alamillo H., Dornburg A., Rabosky D.L., Carnevale G., and Harmon L.J. 2009. Nine exceptional radiations plus high turnover explain species diversity in jawed vertebrates. Proceedings of the National Acadamy of Sciences, USA 106:13410–13414.

Anacker B.L., Whittall J.B., Goldberg E.E., and Harrison S.P. 2010. Origins and consequences of serpentine endemism in the California flora. Evolution 65:365–376.

Bollback J.P. 2006. Simmap: Stochastic character mapping of discrete traits on phylogenies. BMC Bioinformatics 7:88.

FitzJohn R.G. 2010. Quantitative traits and diversification. Systematic Biology 59:619–633.

FitzJohn R.G., Maddison W.P., and Otto S.P. 2009. Estimating trait-dependent speciation and extinction rates from incompletely resolved phylogenies. Systematic Biology 58:595–611.

Goldberg E.E. and Igić B. submitted. Tempo and mode in plant breeding system evolution. Proceedings of the Royal Society of London Series B .

Goldberg E.E., Lancaster L.T., and Ree R.H. 2011. Phylogenetic inference of reciprocal effects between geographic range evolution and diversification. Systematic Biology 60:451–465.

Lislevand T., Figuerola J., and Székely T. 2007. Avian body sizes in relation to fecundity, mating system, display behavior, and resource sharing. Ecology 88:1605.

Maddison W.P., Midford P.E., and Otto S.P. 2007. Estimating a binary character's effect on speciation and extinction. Systematic Biology 56:701–710.

Magnuson-Ford K. and Otto S.P. 2012. Linking the investigations of character evolution and species diversification. The American Naturalist .

Neal R.M. 2003. Slice sampling. Annals of Statistics 31:705–767.

Nee S., May R.M., and Harvey P.H. 1994. The reconstructed evolutionary process. Philosophical Transactions of the Royal Society B: Biological Sciences 344:305–311.

Pagel M. 1994. Detecting correlated evolution on phylogenies: a general method for the comparative analysis of discrete characters. Proceedings of the Royal Society of London Series B 255:37–45.

Redding D.W., DeWolff C., and Mooers A.Ø. 2010. Evolutionary distinctiveness, threat status and ecological oddity in primates. Conservation Biology 24:1052–1058.

SimTreeSDD. 2010. Simulating phylogenetic trees under state-dependent diversification.

Thomas G.H., Wills M.A., and Székely T. 2004. A supertree approach to shorebird phylogeny. BMC Evolutionary Biology 4:28.

Vos R.A. and Mooers A.Ø. 2006. A new dated supertree of the Primates. chapter 5. *In* Inferring large phylogenies: the big tree problem. (R.A. Vos) PhD. thesis, Simon Fraser University.