Section 2: Checking implementation

The easiest way to check that the likelihoods calculated by `MiSSE()`, `hisse()`, and `MuHiSSE()` are correct is to take the likelihood calculated by `MiSSE()` and add this to the likelihood of a simple Markov transition model. As described in Caetano et al. (2018), an SSE model jointly maximizes the probability of the observed states at the tips *and* the observed tree, given the model. Thus, if the character is completely disassociated from the rate differences in the tree (with what we call a character independent model) then this test should work.

First step is load the following packages:

```
suppressPackageStartupMessages(library(hisse))
suppressPackageStartupMessages(library(corHMM))
suppressPackageStartupMessages(library(diversitree))
```

We will calculate the likelihood of a simple BiSSE model, where the diversification rates are the same for states 0 and 1, on a simulated dataset that contains fossils:

```
pars <- c(0.1, 0.2, 0.03, 0.03, 0.01, 0.01)
set.seed(4)
phy <- NULL
while( is.null( phy ) ){
  phy <- tree.bisse(pars, max.t=30, x0=0, include.extinct=TRUE)
}
k.samples <- data.frame(taxon1="sp12", taxon2="sp12", timefrompresent=3.164384,
        state=1, stringsAsFactors=FALSE)
hidden.states=FALSE
phy.k <- hisse:::AddKNodes(phy, k.samples)
fix.type <- hisse:::GetKSampleMRCA(phy.k, k.samples)
nb.tip <- Ntip(phy.k)
nb.node <- phy.k$Nnode
gen <- hisse:::FindGenerations(phy.k)

data <- data.frame(taxon=names(phy$tip.state), phy$tip.state,
       stringsAsFactors=FALSE)
data <- hisse:::AddKData(data, k.samples)
data.new <- data.frame(data[,2], data[,2], row.names=data[,1])
data.new <- data.new[phy.k$tip.label,]

dat.tab <- hisse:::OrganizeDataHiSSE(data.new, phy=phy.k, f=c(1,1),
        hidden.states=FALSE)
edge_details <- hisse:::GetEdgeDetails(phy=phy.k,
        intervening.intervals=strat.cache$intervening.intervals)
fossil.taxa <- edge_details$tipward_node[which(edge_details$type ==
        "extinct_tip")]
pars.bisse <- c(0.1+0.03, 0.1+0.03, 0.03/0.1, 0.03/0.1, 0.01, 0.01)

model.vec <- numeric(48)
model.vec[1:6] = pars.bisse
phy$node.label = NULL
```

```
cache <- hisse:::ParametersToPassfHiSSE(model.vec, hidden.states=hidden.states,
        nb.tip=Ntip(phy.k), nb.node=Nnode(phy.k), bad.likelihood=-300, f=c(1,1),
        ode.eps=0)
cache$psi <- 0.01
hisse.full <- hisse:::DownPassHiSSE(dat.tab, gen, cache, root.type="madfitz",
        condition.on.survival=TRUE, root.p=NULL, node=fix.type$node,
        state=fix.type$state, fossil.taxa=fossil.taxa, fix.type=fix.type$type)
```

Next we will calculate the likelihood of *just* the tree using `MiSSE()`:

```
dat.tab <- hisse:::OrganizeDataMiSSE(phy=phy.k, f=1, hidden.states=1)
model.vec <- c(0.1+0.03, 0.03/0.1, rep(0,51))
cache = hisse:::ParametersToPassMiSSE(model.vec=model.vec, hidden.states=1,
        fixed.eps=NULL, nb.tip=nb.tip, nb.node=nb.node, bad.likelihood=
        exp(-500), ode.eps=0)#
cache$psi <- 0.01
gen <- hisse:::FindGenerations(phy.k)
MiSSE.logL <- hisse:::DownPassMisse(dat.tab=dat.tab, cache=cache, gen=gen,
        condition.on.survival=TRUE, root.type="madfitz", root.p=NULL,
        fossil.taxa=fossil.taxa, node=fix.type$node, fix.type=fix.type$type)
```

Finally, I will use `corHMM` to calculate the likelihood of *just* character data:

```
library(corHMM)
char.logL <- corHMM(phy.k, data, rate.cat=1, model = "ER", node.states = "none",
        fixed.nodes=FALSE, p=0.01, root.p="maddfitz")
```

```
## You specified 'fixed.nodes=FALSE' but included a phy object with node labels. These node labels have
## State distribution in data:
## States:  1   2
## Counts:  2   17
## Calculating likelihood from a set of fixed parameters
```

We can compare the likelihoods obtained from `hisse()` against the sum of the tree *and* the character:

```
tot.logL <- char.logL$loglik + MiSSE.logL
comparison <- identical(round(hisse.full,3), round(tot.logL,3))
print(comparison)
```

```
## [1] TRUE
```

This confirms that the calculations are correct. I will show the same using `MuHiSSE()`:

```
library(diversitree)
pars <- c(.1,  .15,  .2, .1,
          .03, .045, .06, 0.03,
          .05, .05, .00,
          .05, .00, .05,
          .05, .00, .05,
          .00, .05, .05)
set.seed(2)
phy <- NULL
while( is.null( phy ) ){
    phy <- tree.musse(pars, 30, x0=1, include.extinct=TRUE)
}
k.samples <- data.frame(taxon1="sp20", taxon2="sp37", timefrompresent=8.54554,
                        state1=0, state2=1, stringsAsFactors=FALSE)
```

```
phy.k <- hisse:::AddKNodes(phy, k.samples)
fix.type <- hisse:::GetKSampleMRCA(phy.k, k.samples)
nb.tip <- Ntip(phy.k)
nb.node <- phy.k$Nnode
gen <- hisse:::FindGenerations(phy.k)

states <- phy$tip.state
states <- data.frame(phy$tip.state, phy$tip.state,
row.names=names(phy$tip.state))
states <- states[phy$tip.label,]
states.trans <- states
for(i in 1:Ntip(phy)){
    if(states[i,1] == 1){
        states.trans[i,1] = 0
        states.trans[i,2] = 0
    }
    if(states[i,1] == 2){
        states.trans[i,1] = 0
        states.trans[i,2] = 1
    }
    if(states[i,1] == 3){
        states.trans[i,1] = 1
        states.trans[i,2] = 0
    }
    if(states[i,1] == 4){
        states.trans[i,1] = 1
        states.trans[i,2] = 1
    }
}

data <- data.frame(taxon=names(phy$tip.state),
        states.trans[,1], states.trans[,2], stringsAsFactors=FALSE)
data <- hisse:::AddKData(data, k.samples, muhisse=TRUE)
data.new <- data.frame(data[,2], data[,3], row.names=data[,1])
data.new <- data.new[phy.k$tip.label,]

pars.muhisse <- c(rep(0.1+0.03,4), rep(0.03/.1, 4), 0.05,0.05,0, 0.05,0,0.05,
                  0.05,0,.05, 0,0.05,.05)
model.vec = rep(0,384)
model.vec[1:20] = pars.muhisse
cache <- hisse:::ParametersToPassMuHiSSE(model.vec=model.vec, hidden.states=FALSE,
        nb.tip=Ntip(phy.k), nb.node=Nnode(phy.k), bad.likelihood=exp(-500),
        f=c(1,1,1,1), ode.eps=0)
cache$psi <- 0.01
gen <- hisse:::FindGenerations(phy.k)
dat.tab <- hisse:::OrganizeData(data.new, phy.k, f=c(1,1,1,1),
        hidden.states=FALSE, includes.fossils=TRUE)
fossil.taxa <- which(dat.tab$branch.type == 1)

muhisse.full <- hisse:::DownPassMuHisse(dat.tab, gen=gen, cache=cache,
        root.type="madfitz", condition.on.survival=TRUE, root.p=NULL,
        node=fix.type$node, state=fix.type$state,
        fossil.taxa=fossil.taxa, fix.type=fix.type$type)
```

```
## Trait independent model should be loglik_tree + loglik_character ##
#Part 1: MiSSE loglik:
dat.tab <- hisse:::OrganizeDataMiSSE(phy=phy.k, f=1, hidden.states=1)
model.vec <- c(0.1+0.03, 0.03/0.1, rep(0,51))
cache = hisse:::ParametersToPassMiSSE(model.vec=model.vec, hidden.states=1,
        fixed.eps=NULL, nb.tip=nb.tip, nb.node=nb.node,
        bad.likelihood=exp(-500), ode.eps=0)#
cache$psi <- 0.01
edge_details <- hisse:::GetEdgeDetails(phy=phy.k,
        intervening.intervals=strat.cache$intervening.intervals)
fossil.taxa <- edge_details$tipward_node[which(edge_details$type == "extinct_tip")]
gen <- hisse:::FindGenerations(phy.k)
MiSSE.logL <- hisse:::DownPassMisse(dat.tab=dat.tab, cache=cache, gen=gen,
        condition.on.survival=TRUE,
        root.type="madfitz", root.p=NULL, fossil.taxa=fossil.taxa,
        node=fix.type$node, fix.type=fix.type$type)

#Part 2: corHMM loglik:
char.logL <- corHMM(phy.k, data, rate.cat=1, model = "ER", node.states = "none",
        fixed.nodes=FALSE, p=0.05, root.p="maddfitz")
```

```
## You specified 'fixed.nodes=FALSE' but included a phy object with node labels. These node labels have
## State distribution in data:
## States:  1   2   3   4
## Counts:  12  21  6   3
## Calculating likelihood from a set of fixed parameters
```

```
tot.logL <- char.logL$loglik + MiSSE.logL

comparison <- identical(round(muhisse.full,3), round(tot.logL,3))
print(comparison)
```

```
## [1] TRUE
```

References

Caetano D.S., O'Meara B.C., Beaulieu J.M. 2018. Hidden state models improve state-dependent diversification approaches, including biogeographic models. Evolution, 72:2308-2324.