# Running a Multistate HiSSE model

*Jeremy M. Beaulieu*

## Background

This tutorial gives some basic information on how to set up and execute parameter estimates when using MuHiSSE (Nakov et al., 2018).

In Nakov et al. (2018), we extended the HiSSE and CID model framework to handle a four-state case, representing correlated evolution between two binary characters. Note that a mulitstate version of HiSSE is available in RevBayes (based on its manual and conversations with Sebastian Höhna), in `SecSSE` (Herrera-Alsina et al., 2018), and as part of the model set described in Caetano et al. (2018). Our character state combinations were comprised of marine and planktonic (MP, or state combination 00), marine and benthic (MB, or 01), freshwater and planktonic (FP, or 10), and freshwater and benthic (FB, or 11). As with `hisse`, rather than optimizing $\lambda_i$ and $\mu_i$ separately, `MuHiSSE` optimizes transformations of these variables. Again, we let $\tau_i = \lambda_i + \mu_i$ define "net turnover", and we let $\epsilon_i = \mu_i/\lambda_i$ define the "extinction fraction". This reparameterization alleviates problems associated with over-fitting when $\lambda_i$ and $\mu_i$ are highly correlated, but both matter in explaining the diversity pattern (see discussion of this issue in Beaulieu and O'Meara 2016). The number of free parameters in the model for both net turnover and extinction fraction are specified as index vectors provided to the function call. Each vector contains four entries that correspond to rates associated with the observed states (0 or 1) and the hidden states (A or B). They are always ordered as follows for a given hidden state, $i$: $00_i$, $01_i$, $10_i$, $11_i$.

The set of models included range from (1) the "trivial null" model that assume no variation in diversification, to (2) a multistate speciation and extinction model (MuSSE; FitzJohn 2012) that linked diversification rates to the four character states, but assumed no variation within regimes, to (3) multistate-dependent speciation and extinction models with hidden traits (MuHiSSE) and (4) a set of multistate character-independent models (MuCID) with increasing numbers of hidden traits (MuCID2 through MuCID8). The most complex MuCID8 model has, at it's most complex, eight parameters for turnover and eight parameters for extinction fraction, matching the complexity of the MuHiSSE model, but with diversification parameters unlinked to the observed character states. For any model, we allow the transition rates between alternative state combinations to differ (e.g., $q_{MB \rightarrow MP} \neq q_{MP \rightarrow MB}$), or constrained to be equal (e.g., $q_{MB \rightarrow MP} = q_{MP \rightarrow MB}$). In the analyses presented in Nakov et al. (2018), we disallowed dual transitions (e.g., $q_{MB \rightarrow FP} = 0$), though this need not always be the case.

Below I will demonstrate how to set up various models, and how the likelihoods compare against those from `diversitree` whenever possible. In the end, I will also show how a three-state model can be set up from MuHiSSE and compare the likelihoods against the three-state MuSSE implemented in GeoHiSSE. However, before getting started, be sure to load the `hisse` and `diversitree` packages:

```
suppressWarnings(library(hisse))
```

```
## Loading required package: ape
## Loading required package: deSolve
## Loading required package: GenSA
## Loading required package: subplex
## Loading required package: nloptr
```

```
suppressWarnings(library(diversitree))
```

### Simulating a practice data set

Here I will simulate a simple four-state model using `diversitree`.

```
pars <- c(.1,  .15,  .2, .1, # lambda 1, 2, 3, 4
.03, .045, .06, 0.03, # mu 1, 2, 3, 4
.05, .05, .00,        # q12, q13, q14
.05, .00, .05,     # q21, q23, q24
.05, .00, .05,     # q31, q32, q34
.00, .05, .05)
set.seed(2)
phy <- tree.musse(pars, 30, x0=1)
states <- phy$tip.state
```

Now let's get the likelihood using the same parameters we used to simulate the data:

```
lik <- make.musse(phy, states, 4)
#lik <- make.musse(phy, states, 3)
diversitree.free = lik(pars)
print(diversitree.free)
```

```
## [1] -119.6924
```

Now we can use machinery inside MuHiSSE to demonstrate that we generate the same likelihood. Unfortunately, it takes a little bit of code to get everything into the right format to run properly. This is what is shown below. Note, too, that I am transforming $\lambda_i$ and $\mu_i$ into turnover and extinction fraction:

```
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
  }
}
pars.hisse <- c(pars[1]+pars[5],pars[2]+pars[6],pars[3]+pars[7],pars[4]+pars[8],
                pars[5]/pars[1],pars[6]/pars[2],pars[7]/pars[3],pars[8]/pars[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.hisse
phy$node.label = NULL
cache <- hisse:::ParametersToPassMuHiSSE(model.vec=model.vec, hidden.states=TRUE,
```

```
                                          nb.tip=Ntip(phy), nb.node=Nnode(phy),
                                          bad.likelihood=exp(-500), ode.eps=0)
gen <- hisse:::FindGenerations(phy)
dat.tab <- hisse:::OrganizeData(states.trans, phy, f=c(1,1,1,1), hidden.states=TRUE)
hisse.constrained <- hisse:::DownPassMuHisse(dat.tab, gen=gen, cache=cache,
                                             root.type="madfitz", condition.on.survival=TRUE)
comparison <- identical(round(hisse.constrained,4), round(diversitree.free,4))
print(comparison)
```

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

The implementation is correct. However, in the above case, we did not actually use any hidden states. We can also show that the likelihoods are identical to MuSSE so as long as the dynamics are the same among all included hidden states. Here we will just use hidden states A and B:

```
pars.hisse <- rep(c(pars[1]+pars[5],pars[2]+pars[6],pars[3]+pars[7],pars[4]+pars[8],
                pars[5]/pars[1],pars[6]/pars[2],pars[7]/pars[3],pars[8]/pars[4],
                0.05,0.05,0, 0.05,0,0.05, 0.05,0,.05, 0,0.05,.05, 1,rep(0,6), 1,
                rep(0,6), 1,rep(0,6), 1,rep(0,6)),2)
model.vec = rep(0,384)
model.vec[1:96] = pars.hisse
phy$node.label = NULL
cache <- hisse:::ParametersToPassMuHiSSE(model.vec=model.vec, hidden.states=TRUE,
                                         nb.tip=Ntip(phy), nb.node=Nnode(phy),
                                         bad.likelihood=exp(-500), ode.eps=0)
gen <- hisse:::FindGenerations(phy)
dat.tab <- hisse:::OrganizeData(states.trans, phy, f=c(1,1,1,1), hidden.states=TRUE)
hisse.constrained <- hisse:::DownPassMuHisse(dat.tab, gen=gen, cache=cache,
                                             root.type="madfitz", condition.on.survival=TRUE)
comparison <- identical(round(hisse.constrained,4), round(diversitree.free,4))
print(comparison)
```

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

Again, the likelihoods are identical and the implementation is correct.

## Setting up a MuSSE model using MuHiSSE

As mentioned above, the number of free parameters in the model for both net turnover and extinction fraction are specified as index vectors provided to the function call. Each vector contains four entries that correspond to rates associated with the observed states (0 or 1) and the hidden states (A or B). They are always ordered as follows for a given hidden state, $i$: $00_i$, $01_i$, $10_i$, $11_i$. However, in this case we do not want any hidden states. But first let's set up the "dull null" – i.e., turnover and extinction fraction are the same for all observed state combinations. Note the "f" represents the sampling fraction for each observed state combination, which is a vector ordered in the same manner as for turnover and extinction fraction vectors:

```
turnover <- c(1,1,1,1)
extinction.fraction <- c(1,1,1,1)
f = c(1,1,1,1)
```

Next, we have to set up a transition matrix. There is a function provided to make this easy, and allows users to customize the matrix to fit particular hypotheses. Be sure to look at the options on this function call, for allowing diagonals and for customizing the matrix when you have character independent model.

```
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=0)
print(trans.rate)
```

```
##      (00) (01) (10) (11)
## (00)   NA    3    5    0
## (01)    1   NA    0    7
## (10)    2    0   NA    8
## (11)    0    4    6   NA
```

Now, we can call MuHiSSE and estimate the parameters under this model using the default settings:

```
dull.null <- MuHiSSE(phy=phy, data=states.trans, f=f, turnover=turnover,
                     eps=extinction.fraction, hidden.states=FALSE,
                     trans.rate=trans.rate)
```

If you wanted to set up a true MuSSE model, where the turnover rate parameters are unlinked across the observed state combinations, you would simply do the following:

```
turnover <- c(1,2,3,4)
extinction.fraction <- c(1,1,1,1)
MuSSE <- MuHiSSE(phy=phy, data=states.trans, f=f, turnover=turnover,
                     eps=extinction.fraction, hidden.states=FALSE,
                     trans.rate=trans.rate)
```

## Setting up a character-dependent MuHiSSE model

Setting up a character-dependent MuHiSSE model is relatively straightforward, and relies on all the same tools as above. One important thing to bear in mind, is that again, the states are ordered by state combination within each hidden state. For example, if you want two hidden states, A and B, the order of the parameters in the model is 00A, 01A, 10A, 11A, 00B, 01B, 10B, and 11B. So, in this case, we just need to specify the free parameters we want for diversification. Here we are just going assume turnover varies across the different states:

```
turnover <- c(1,2,3,4,5,6,7,8)
extinction.fraction <- rep(1, 8)
f = c(1,1,1,1)
```

We also have to extend the transition rate matrix. This is done by specifying the number of hidden states:

```
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=1)
print(trans.rate)
```

```
##       (00A) (01A) (10A) (11A) (00B) (01B) (10B) (11B)
## (00A)    NA     3     5     0    17    NA    NA    NA
## (01A)     1    NA     0     7    NA    17    NA    NA
## (10A)     2     0    NA     8    NA    NA    17    NA
## (11A)     0     4     6    NA    NA    NA    NA    17
## (00B)    17    NA    NA    NA    NA    11    13     0
## (01B)    NA    17    NA    NA     9    NA     0    15
## (10B)    NA    NA    17    NA    10     0    NA    16
## (11B)    NA    NA    NA    17     0    12    14    NA
```

Now, we can just plug these options into MuHiSSE:

```
turnover <- c(1,2,3,4)
extinction.fraction <- c(1,1,1,1)
MuHiSSE <- MuHiSSE(phy=phy, data=states.trans, f=f, turnover=turnover,
                     eps=extinction.fraction, hidden.states=TRUE,
                     trans.rate=trans.rate)
```

### Setting up a character-independent MuHiSSE model

Remember, for any character-independent model, the diversification rates *must* be decoupled from the observed states. So, to do this, we simply set the diversification rates to be equal for all states for a given hidden state. Below, I will show how to do this for a character-independent model with two rate shifts in the tree:

```
turnover <- c(1,1,1,1,2,2,2,2)
extinction.fraction <- rep(1, 8)
f = c(1,1,1,1)
```

For the transition rate matrix, I included a setting called `make.null` that simply replicates the transition model across the hidden states. This way the transition rates are not impacted by changes in the diversification rate regime – that is, $q_{MB,A \rightarrow MP,A} = q_{MB,B \rightarrow MP,B}$.

```
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=1, make.null=TRUE)
print(trans.rate)
```

```
##       (00A) (01A) (10A) (11A) (00B) (01B) (10B) (11B)
## (00A)    NA     3     5     0     9    NA    NA    NA
## (01A)     1    NA     0     7    NA     9    NA    NA
## (10A)     2     0    NA     8    NA    NA     9    NA
## (11A)     0     4     6    NA    NA    NA    NA     9
## (00B)     9    NA    NA    NA    NA     3     5     0
## (01B)    NA     9    NA    NA     1    NA     0     7
## (10B)    NA    NA     9    NA     2     0    NA     8
## (11B)    NA    NA    NA     9     0     4     6    NA
```

Here is a how you would set up the diversification rates for models with three hidden state all the way to eight hidden states:

```
## Three hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4))
extinction.fraction <- rep(1, 12)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=2, make.null=TRUE)

## Four hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4), rep(4,4))
extinction.fraction <- rep(1, 16)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=3, make.null=TRUE)

## Five hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4), rep(4,4), rep(5,4))
extinction.fraction <- rep(1, 20)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=4, make.null=TRUE)

## Six hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4), rep(4,4), rep(5,4), rep(6,4))
extinction.fraction <- rep(1, 24)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=5, make.null=TRUE)

## Seven hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4), rep(4,4), rep(5,4), rep(6,4),
              rep(7, 4))
extinction.fraction <- rep(1, 28)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=6, make.null=TRUE)
```

```
## Eight hidden states
turnover <- c(rep(1,4), rep(2,4), rep(3,4), rep(4,4), rep(5,4), rep(6,4),
              rep(7,4), rep(8,4))
extinction.fraction <- rep(1, 32)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=7, make.null=TRUE)
```

## Relationship of MuHiSSE to a three-state GeoSSE without cladogenetic changes

As of June 2017, I have implemented a three-state MuSSE model as part of the model set for GeoHiSSE (see Caetano et al., 2018). If cladogenetic changes are disallowed, then the geographic model essentially reverts back to three-state MuSSE model. Below I will show how the likelihoods are equivalent between `diversitree`, `GeoHiSSE`, and `MuHiSSE` in the three-state case.

First, let's simulate a three-state model:

```
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)
states <- phy$tip.state
lik <- make.musse(phy, states, 3)
lik.base <- constrain(lik, lambda2 ~ lambda1, lambda3 ~ lambda1,
mu2 ~ mu1, mu3 ~ mu1,
q13 ~ 0, q21 ~ q12, q23 ~ q12, q31 ~ 0.03, q32 ~ q12)
```

Here is the likelihood for a maximally constrained model using `diversitree`:

```
diversitree.constrained = lik.base(c(.1, .03, .05))
print(diversitree.constrained)
```

```
## [1] -111.4925
```

Now, let's compare the likelihood using a three-state model in `GeoHiSSE`:

```
states <- data.frame(phy$tip.state, phy$tip.state, row.names=names(phy$tip.state))
states <- states[phy$tip.label,]
states[states[,1]==3,] = 4
pars.hisse <- c(0.1, 0.1, 0.1, 0.03, 0.03, 0.03, 0.05, 0, 0.05, 0.05, 0, 0.05)
model.vec = rep(0,120)
model.vec[1:12] = pars.hisse
phy$node.label = NULL
cache <- hisse:::ParametersToPassMuSSE(phy, states[,1], model.vec, f=c(1,1,1), hidden.states="TEST1")
geosse.constrained <- hisse:::DownPassMusse(phy, cache, hidden.states=FALSE,
                                            root.type="madfitz", condition.on.survival=TRUE)
comparison <- identical(round(geosse.constrained,4), round(diversitree.constrained,4))
print(comparison)
```

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

We can show that you can obtain the same likelihood using `MuHiSSE`. Warning: as before it's a bit convoluted to format the data properly, but it can be done:

```
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] == 4){
    states.trans[i,1] = 1
    states.trans[i,2] = 0
  }
}
pars.hisse <- c(0.1+0.03,0.1+0.03,0.1+0.03,0,
                0.03/0.1,0.03/0.1,0.03/0.1,0,
                0.05,0,0, 0.05,0.05,0, 0.03,0.05,0, 0,0,0)
model.vec = rep(0,384)
model.vec[1:20] = pars.hisse
phy$node.label = NULL
cache <- hisse:::ParametersToPassMuHiSSE(model.vec=model.vec, hidden.states=TRUE,
                                         nb.tip=Ntip(phy), nb.node=Nnode(phy),
                                         bad.likelihood=exp(-500), ode.eps=0)
gen <- hisse:::FindGenerations(phy)
dat.tab <- hisse:::OrganizeData(states.trans, phy, f=c(1,1,1,0), hidden.states=TRUE)
muhisse.constrained <- hisse:::DownPassMuHisse(dat.tab, gen=gen, cache=cache,
                                               root.type="madfitz", condition.on.survival=TRUE)
comparison <- identical(round(muhisse.constrained,4), round(diversitree.constrained,4))
print(comparison)
```

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

As you can see, in the absence of hidden states, all three implementations provide identical likelihoods.

It is also easy to show how you would set up a three-state model using MuHiSSE. This may be beneficial with rather large trees. With `MuHiSSE`, as well as `fGeoHiSSE`, I've implemented a much more efficient and rather fast tree traversal algorithm. Eventually, both functions will allow for multicore processing during the tree traversal stage.

For this example, note that we are going to assume we do not have a fourth state (i.e., 11).

```
turnover <- c(1,2,3,0)
extinction.fraction <- c(1,1,1,0)
f = c(1,1,1,0)
```

Next, we have to generate, and then modify the transition rate matrix:

```
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=0)
print(trans.rate)
```

```
##      (00) (01) (10) (11)
## (00)   NA    3    5    0
## (01)    1   NA    0    7
## (10)    2    0   NA    8
## (11)    0    4    6   NA
```

We can use the `ParDrop()` function to remove transitions to and from thr fourth state in the model:

```
trans.rate.mod <- ParDrop(trans.rate, c(4,6,7,8))
print(trans.rate.mod)
```

```
##      (00) (01) (10) (11)
## (00)   NA    3    4    0
## (01)    1   NA    0    0
## (10)    2    0   NA    0
## (11)    0    0    0   NA
```

From there the function call is all the same:

```
MuHiSSE <- MuHiSSE(phy=phy, data=states.trans, f=f, turnover=turnover,
                   eps=extinction.fraction, hidden.states=FALSE,
                   trans.rate=trans.rate.mod)
```

If you wanted to set up a three-state model, but include hidden states, you would do the following:

```
turnover <- c(1,2,3,0, 4,5,6,0)
extinction.fraction <- c(1,1,1,0, 1,1,1,0)
f = c(1,1,1,0)
trans.rate <- TransMatMakerMuHiSSE(hidden.traits=1)
trans.rate.mod <- ParDrop(trans.rate, c(4,6,7,8,12,14,15,16))
```

## Other considerations

Like with `hisse` and `GeoHiSSE`, there are functions available to obtain model averages (i.e., `GetModelAveRates()`), generate estimates of the uncertainty in the parameter estimates (i.e., `SupportRegionMuHiSSE()`), calculate the marginal probabilities for states at nodes (i.e., `MarginReconMuHiSSE()`), and plotting the rate variation on the tree (i.e., `plot.muhisse.states()`). Users are encouraged to read other vignettes and help pages provided for more information. For more conceptual discussions of these functions and ideas, readers are also encouraged to read Caetano et al. (2018).

One additional item that is worth mentioning. I would recommend users try multiple random starting points when optimizing any given model with `MuHiSSE`. In Nakov et al. (2018), we found that the default starting values often did not return the highest log likelihood. To alleviate this issue, we performed $\geq 50$ maximum likelihood optimizations for each model, each initiated from a distinct starting point. All functions with `hisse` are provided with `starting.vals` option for these purposes. In the case of Nakov et al. (2018), we wrote our own custom function, which I have provided below using a MuCID3 model. Note that the following function was designed for multicore use:

```
StartingValueTry <- function(phy, data, f, trans.rate){

    freqs <- table(apply(data[,2:3], 1,
                         function(x) switch(paste0(x, collapse=""),
                                            "00" = 1, "01" = 2, "10" = 3, "11" = 4)))
    samp.freq.tree <- Ntip(phy) / sum(freqs / f)
    init.pars <- hisse:::starting.point.generator(phy, 4, samp.freq.tree, yule=FALSE)

    turnover <- c(rep(1,4), rep(2,4), rep(3,4))
    eps <- rep(1,length(turnover))
    NewStarting <- function(iteration){
        turn.start <- exp(rnorm(4, log(init.pars[1]+init.pars[5]), 1))
        eps.start <- runif(1, 0, 1)
        trans.start <- exp(rnorm(12, log(init.pars[9])))
```

```
        starting.vals <- c(turn.start, rep(eps.start,4), trans.start)
        print(starting.vals)
        tmp <- MuHiSSE(phy, data, f=f, turnover=turnover, eps=eps, trans.rate=trans.rate,
                       hidden.states=TRUE, starting.vals=starting.vals)
        save(tmp, file=paste("startingValsTest", iteration, ".Rsave", sep=""))
    }
    mclapply(1:50, NewStarting, mc.cores=50)
}
```

## References


Caetano, D.S., B.C. O'Meara, and J.M. Beaulieu. 2018. Hidden state models improve state-dependent diversification approaches, including biogeographic models. Evolution, https://doi.org/10.1111/evo.13602

FitzJohn R.G. 2012. Diversitree: comparative phylogenetic analyses of diversification in R. Methods in Ecology and Evolution 3:1084-1092.

Herrera-Alsina, L., P. van Els, and R.S. Etienne. 2018. Detecting the dependence of diversification on multiples traits from phylogenetic trees and trait data. Systematic Biology, In press.

Nakov, T., Beaulieu, J.M., and Alverson, A.J. 2018. Freshwater diatoms diversify faster than marine in both planktonic and benthic habitats. bioRxiv, doi: https://doi.org/10.1101/406165.