Running GeoHiSSE

Daniel Caetano and Jeremy M. Beaulieu

Getting started

This tutorial gives some basic information on how to set up and execute parameter estimates for a series of models using GeoHiSSE (Caetano et al., 2018). The GeoHiSSE models can be used to infer ancestral ranges, rates of dispersion and extirpation as well as testing hypothesis about range-dependent diversification processes.

The main difference between GeoSSE and GeoHiSSE is that here we implement models that allow for diversification rate variation both within and between geographical areas. Such models are more adequate to empirical data then homogeneous diversification rates implied by GeoSSE (as well as BiSSE). The GeoHiSSE models belong to the same category of Hidden-Markov models as HiSSE. Thus, the concepts will be familiar to you if you have some experience with HiSSE (and vice-versa).

Best place to install the package with the new functions provided here is from our github repository using the package `devtools`:

```
library( devtools )
## At the moment you need to point to the development branch on github.
install_github(repo = "thej022214/hisse", ref = "master")
```

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 range-independent process

Here we will simulate a phylogenetic tree using neutral geographical ranges. We will incorporate three different rates of diversification. Thus, the correct process here is: "rates of diversification vary independently of the geographic ranges".

We use a simulation here just because it is an easy way to produce data and because we know the underlying diversification process. Otherwise, if you have an empiricial dataset, all the steps we show here apply. Just make sure to substitute the phylogeny and data with your datatset.

```
## Generate a list with the parameters of the model:
pars <- SimulateGeoHiSSE(hidden.areas = 1, return.GeoHiSSE_pars = TRUE)
pars
```

```
## $model.pars
##     A B
## s01 0 0
## s0  0 0
```

```
## s1  0 0
## x0  0 0
## x1  0 0
## d0  0 0
## d1  0 0
##
## $q.01
##     01A 01B
## 01A  NA   0
## 01B   0  NA
##
## $q.0
##    0A 0B
## 0A NA  0
## 0B  0 NA
##
## $q.1
##    1A 1B
## 1A NA  0
## 1B  0 NA
##
## attr(,"class")
## [1] "list"          "GeoHiSSE_pars"
```

The object `pars` is a list with all the parameter values for this model in the correct order and format, but all values are 0. Thus, we need to populate these parameters with numbers in order to perform the simulation.

```
pars$model.pars[,1] <- c(0.1, 0.1, 0.1, 0.03, 0.03, 0.05, 0.05)
pars$model.pars[,2] <- c(0.2, 0.2, 0.2, 0.03, 0.03, 0.05, 0.05)
pars$q.01[1,2] <- pars$q.01[2,1] <- 0.05
pars$q.0[1,2] <- pars$q.0[2,1] <- 0.05
pars$q.1[1,2] <- pars$q.1[2,1] <- 0.05
pars
```

```
## $model.pars
##        A    B
## s01 0.10 0.20
## s0  0.10 0.20
## s1  0.10 0.20
## x0  0.03 0.03
## x1  0.03 0.03
## d0  0.05 0.05
## d1  0.05 0.05
##
## $q.01
##      01A  01B
## 01A   NA 0.05
## 01B 0.05   NA
##
## $q.0
##      0A   0B
## 0A   NA 0.05
## 0B 0.05   NA
##
## $q.1
```

```
##      1A   1B
## 1A   NA 0.05
## 1B 0.05   NA
##
## attr(,"class")
## [1] "list"          "GeoHiSSE_pars"
```

Now we can use the parameters with the same function we applied before `SimulateGeoHiSSE` to generate both the data and the phylogeny.

Here we will set the seed for the simulation, so the outcome of the simulation is always the same. Note that you can change the seed or skip this lines to generate a different, random, dataset.

```
set.seed(1234)
sim.geohisse <- SimulateGeoHiSSE(pars=pars, hidden.areas = 1, x0 = "0A", max.taxa = 200)
```

```
## [1] "Simulating the phylogeny..."
## [1] "Simulation finished!"
```

Setting up the models

In this tutotial we will fit a total of four models. Two models with a range-indendent diversification process and two other models in which the range have an effect on the diversification rate of the lineages (each with either one or two rate classes).

Note that the function to estimate the parameters of the model is commented out below. Just uncomment and run to perform the estimate of the models. Here we will load results from a previous estimate.

Models 1 and 2 below do not include hidden classes. Note that in this case the model will have 3 speciation parametes and 2 extirpation parameters.

```
## Model 1 - Dispersal parameters vary only, no range-dependent diversification.
speciation <- c(1,1,1)
extirpation <- c(1,1)
trans.rate <- TransMatMakerGeoHiSSE(hidden.areas=0)
mod1 <- GeoHiSSE(phy = sim.geohisse$phy, data = sim.geohisse$data, f=c(1,1,1),
                 speciation=speciation, extirpation=extirpation,
                 hidden.areas=FALSE, trans.rate=trans.rate)

## Model 2. Canonical GeoSSE model, range effect on diversification
speciation <- c(1,2,3)
extirpation <- c(1,2)
trans.rate <- TransMatMakerGeoHiSSE(hidden.areas=0)
mod2 <- GeoHiSSE(sim.geohisse$phy, data = sim.geohisse$data, f=c(1,1,1),
                 speciation=speciation, extirpation=extirpation,
                 hidden.areas=FALSE, trans.rate=trans.rate)
```

Models 3 and 4 below have 2 hidden rates each. In this case the models will have twice the number of parameters: 6 speciation parameters and 4 extirpation parameters.

```
## Model 3. GeoHiSSE model with 1 hidden area, no range-dependent diversification.
## Note below how parameters vary among hidden classes but are the same within each
##      hidden class.
speciation <- c(1,1,1,2,2,2)
extirpation <- c(1,1,2,2)
trans.rate <- TransMatMakerGeoHiSSE(hidden.areas=1)
mod3 <- GeoHiSSE(sim.geohisse$phy, data = sim.geohisse$data, f=c(1,1,1),
```

```
                  speciation=speciation, extirpation=extirpation,
                  hidden.areas=TRUE, trans.rate=trans.rate)

## Model 4. GeoHiSSE model with 1 hidden area, no range-dependent diversification.
speciation <- c(1,2,3,4,5,6)
extirpation <- c(1,2,3,4)
trans.rate <- TransMatMakerGeoHiSSE(hidden.areas=1)
mod4 <- GeoHiSSE(sim.geohisse$phy, data = sim.geohisse$data, f=c(1,1,1),
                  speciation=speciation, extirpation=extirpation,
                  hidden.areas=TRUE, trans.rate=trans.rate)
```

Load the fit of the models:

```
load( "geohisse_vignette.Rsave" )
```

Now that we have the fit for the 4 models we can check their parameter estimates.

First model assumes a homogeneous diversification rate across the tree independent of the ranges.

```
mod1
```

```
##
## Fit
##             -lnL              AIC             AICc           n.taxa
##         -803.026         1614.052         1614.257          200.000
## n.hidden.classes
##            1.000
##
## Model parameters:
##
##        s0A        s1A       s01A        x0A        x1A    d0A_01A
## 0.10495443 0.10495443 0.10495443 0.04700056 0.04700056 0.04523465
##    d1A_01A
## 0.06328808
```

Second model assumes a range-dependent diversification process without hidden states. This means that diversification shifts occurr across the branches of the tree and are correlated to the ranges.

```
mod2
```

```
##
## Fit
##             -lnL              AIC             AICc           n.taxa
##        -802.1938        1618.3876        1618.9709         200.0000
## n.hidden.classes
##           1.0000
##
## Model parameters:
##
##        s0A        s1A       s01A        x0A        x1A    d0A_01A
## 0.11383796 0.10000999 0.08864935 0.04545565 0.06020702 0.05333405
##    d1A_01A
## 0.05346983
```

Third model incoporates shifts in diversification across the tree but these are independent of the ranges. Note that the parameter estimes vary between hidden classes but are held the same among different ranges within each hidden class. This is an example of our more complex null model in GeoHiSSE. This model can be extended to fit up to 5 hidden classes.

```
mod3
```

```
##
## Fit
##             -lnL              AIC             AICc           n.taxa
##        -810.6062        1639.2124        1640.1597         200.0000
## n.hidden.classes
##           2.0000
##
## Model parameters:
##
##          s0A          s1A          s01A          x0A          x1A
## 4.996388e+00 4.996388e+00 4.996388e+00 7.411601e+02 7.411601e+02
##      d0A_01A      d1A_01A       d0A_0B       d1A_1B     d01A_01B
## 7.983518e-03 9.980557e+01 2.339666e-05 2.339666e-05 2.339666e-05
##          s0B          s1B          s01B          x0B          x1B
## 1.742381e-01 1.742381e-01 1.742381e-01 1.670225e-01 1.670225e-01
##      d0B_01B      d1B_01B       d0B_0A       d1B_1A     d01B_01A
## 6.429553e-02 9.702651e-02 2.339666e-05 2.339666e-05 2.339666e-05
```

Finally, the third model describes a range-dependent diversification process while also accounting for multiple rate classes. This is the most complex model in this set.

```
mod4
```

```
##
## Fit
##             -lnL              AIC             AICc           n.taxa
##        -798.7843        1627.5686        1630.1773         200.0000
## n.hidden.classes
##           2.0000
##
## Model parameters:
##
##          s0A          s1A          s01A          x0A          x1A
## 1.371612e-01 7.618960e-02 1.668092e-01 7.176987e-02 7.841790e-02
##      d0A_01A      d1A_01A       d0A_0B       d1A_1B     d01A_01B
## 6.001005e-02 7.188738e-02 7.811913e-03 7.811913e-03 7.811913e-03
##          s0B          s1B          s01B          x0B          x1B
## 6.434724e-02 1.261423e-01 2.066618e-09 7.637460e-03 4.241552e-02
##      d0B_01B      d1B_01B       d0B_0A       d1B_1A     d01B_01A
## 4.497654e-02 3.689476e-02 7.811913e-03 7.811913e-03 7.811913e-03
```

Computing Akaike Weights.

Akaike weights are important to evaluate the relative importance of each of the models to explain the variation observed in the data. This quantity takes into account pennalties associated to the number of free parametes.

Models with higher weight show better fit to the data and, as a result, have more weight when performing model averaging (see below).

To compute model weight we can use one of the functions of the package. This will work with both HiSSE and GeoHiSSE objects.

```
GetModelWeight(model1 = mod1, model2 = mod2, model3 = mod3, model4 = mod4)
```

```
##       model1       model2       model3       model4
## 8.963818e-01 1.025742e-01 3.083187e-06 1.040898e-03
```

```
## As the number of models in the set grows, naming each model in the set can become hard.
## So one can use a list (created by some automated code) as an imput also:
list.geohisse <- list(model1 = mod1, model2 = mod2, model3 = mod3, model4 = mod4)
GetModelWeight(list.geohisse)
```

```
##       model1       model2       model3       model4
## 8.963818e-01 1.025742e-01 3.083187e-06 1.040898e-03
```

Model averaging and plotting.

Now we can model average the results. Note that this step will reflect the Akaike model weights that we computed above.

For this we need first to perform a marginal reconstruction for each of the models in the set. This will reconstruct the hidden states at the nodes of the phylogeny. Then we can use this information to compute the model average for the rates.

These can take a while to run. We will load the results of previous analyses. Uncomment the code below to perform the reconstructions.

```
recon.mod1 <- MarginReconGeoSSE(phy = mod1$phy, data = mod1$data, f = mod1$f,
                                pars = mod1$solution, hidden.areas = mod1$hidden.areas,
                                root.type = mod1$root.type, root.p = mod1$root.p,
                                aic = mod1$AIC, n.cores = 4)
recon.mod2 <- MarginReconGeoSSE(phy = mod2$phy, data = mod2$data, f = mod2$f,
                                pars = mod2$solution, hidden.areas = mod2$hidden.areas,
                                root.type = mod2$root.type, root.p = mod2$root.p,
                                aic = mod2$AIC, n.cores = 4)
recon.mod3 <- MarginReconGeoSSE(phy = mod3$phy, data = mod3$data, f = mod3$f,
                                pars = mod3$solution, hidden.areas = mod3$hidden.areas,
                                root.type = mod3$root.type, root.p = mod3$root.p,
                                aic = mod3$AIC, n.cores = 4)
recon.mod4 <- MarginReconGeoSSE(phy = mod4$phy, data = mod4$data, f = mod4$f,
                                pars = mod4$solution, hidden.areas = mod4$hidden.areas,
                                root.type = mod4$root.type, root.p = mod4$root.p,
                                aic = mod4$AIC, n.cores = 4)
```

```
## Load previous results:
load( "geohisse_recons_vignette.Rsave" )
```

The results are phylogenetic trees with information on the nodes.

```
recon.mod1
```

```
##
## Phylogenetic tree with 200 tips and 199 internal nodes.
##
## Tip labels:
##  sp14, sp16, sp21, sp23, sp26, sp30, ...
## Node labels:
##  1, 3, 1, 3, 3, 1, ...
##
## Rooted; includes branch lengths.
```

```
recon.mod2
```

```
##
## Phylogenetic tree with 200 tips and 199 internal nodes.
##
## Tip labels:
##  sp14, sp16, sp21, sp23, sp26, sp30, ...
## Node labels:
##  1, 3, 1, 1, 3, 1, ...
##
## Rooted; includes branch lengths.
```

```
recon.mod3
```

```
##
## Phylogenetic tree with 200 tips and 199 internal nodes.
##
## Tip labels:
##  sp14, sp16, sp21, sp23, sp26, sp30, ...
## Node labels:
##  2, 6, 6, 4, 6, 4, ...
##
## Rooted; includes branch lengths.
```

```
recon.mod4
```

```
##
## Phylogenetic tree with 200 tips and 199 internal nodes.
##
## Tip labels:
##  sp14, sp16, sp21, sp23, sp26, sp30, ...
## Node labels:
##  1, 3, 1, 1, 3, 1, ...
##
## Rooted; includes branch lengths.
```

Now that we have the AIC associated with each model and their reconstruction across the nodes of the tree we can compute the model average:

```
recon.models <- list(recon.mod1, recon.mod2, recon.mod3, recon.mod4)
model.ave.rates <- GetModelAveRates(x = recon.models, type = "tips")
```

```
## Warning in CheckReconBounds(x = list(rates.tips.turnover,
## rates.tips.net.div, : Models in position 3 have parameters outside the
## bounds defined by 'bound.matrix' argument. These will NOT be included in
## the reconstruction.
```

The error message appeared here because the function uses the argument `bound.par.matrix` in order to exclude models with parameter estimates outside the pre-defined bounds. You can use these bounds to make sure that models with poor MLE estimates will not influence the model averages.

The result of the reconstrution is a matrix with the parameter estimates for each of the tips species averaged over all models. Note that for the GeoSSE model there is no "extinction" parameter associated with widespread (01) lineages. Also not that one can change the type of model averaging (between tips, nodes, and both) when callin the `GetModelAveRates` function.

```
head( model.ave.rates )
```

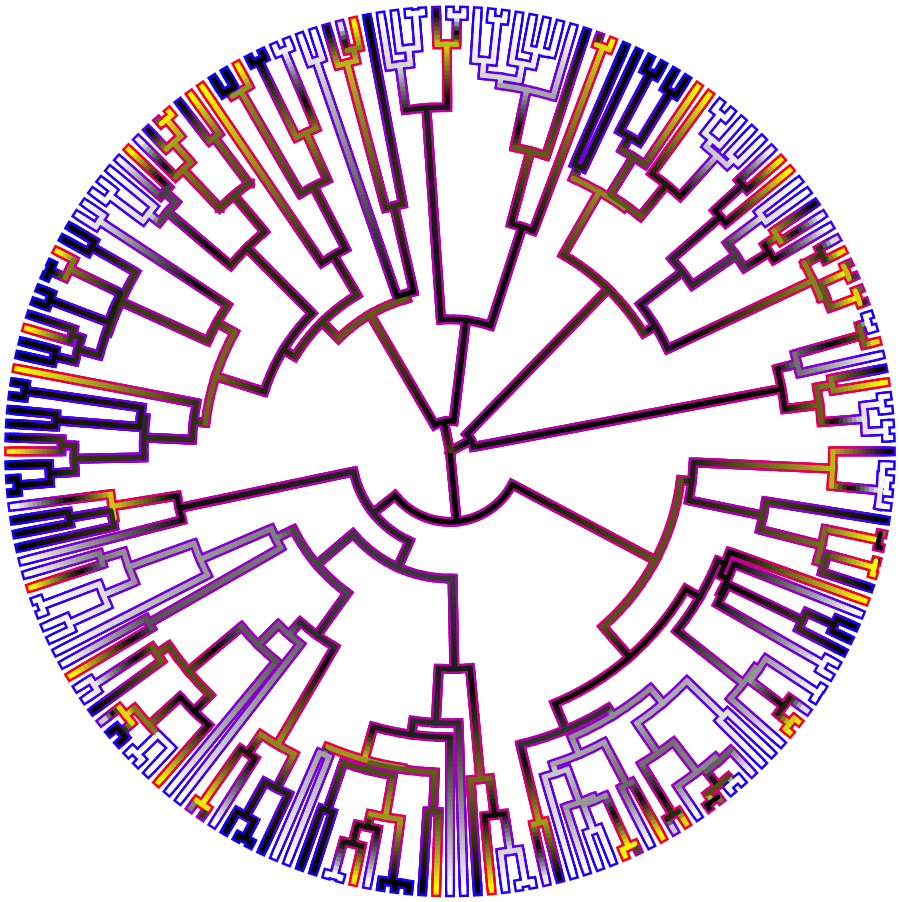
```
##   taxon state.0 state.1 state.01  turnover   net.div speciation
```

```
## 1  sp14       0       0        1 0.3133588 0.3133588  0.1031989
## 2  sp16       1       0        0 0.1527295 0.1058221  0.1058792
## 3  sp21       1       0        0 0.1527351 0.1058224  0.1058822
## 4  sp23       1       0        0 0.1527147 0.1058211  0.1058713
## 5  sp26       1       0        0 0.1527019 0.1058203  0.1058645
## 6  sp30       0       0        1 0.3137521 0.3137521  0.1033111
##   extinct.frac extinction
## 1    0.0000000 0.00000000
## 2    0.4428098 0.04685031
## 3    0.4428264 0.04685293
## 4    0.4427659 0.04684335
## 5    0.4427281 0.04683735
## 6    0.0000000 0.00000000
```

Finally, we can plot the use the resulting data matrix to make a plot of the results.

```
plot.geohisse.states(x = recon.models, rate.param = "net.div", type = "fan",
                     show.tip.label = FALSE, legend = FALSE)
```

```
## [1] "Using default colors: white (state 1), black (state 2), and yellow (state 0)."
```



```
## $rate.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 200 tips and 199 internal nodes.
##
## (2) A mapped continuous trait on the range (0.104231, 0.318219).
```

```
##
##
## $state.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 200 tips and 199 internal nodes.
##
## (2) A mapped continuous trait on the range (0, 2.002).
```

A brief note about the new fGeoHiSSE functions

As of version 1.8.7, we now provide a new set of functions that execute more complex and potentially faster version of the GeoHiSSE model described by Caetano et al. (2018). One of the main differences here is that the model allows up to 10 hidden categories, and implements a more efficient means of carrying out branch calculations. Specifically, we break up the tree into sets of branches whose branch calculations are independent of one another. We then carry out all descendent branch calculations simultaneously, combine the probabilities based on their shared ancestry, then repeat for the next set of descendent branches. In testing, we've found that as the number of taxa increases, the calculation becomes much more efficient. For instance, with 100,000 tips, a single tree traversal with the canonical GeoSSE model in the original code took 10 minutes, whereas in `fGeoHiSSE` the same traversal took about 30 seconds. In future versions, we will allow for multicore processing of these calculations to further improve speed. Also, note this function will eventually completely replace the version of `GeoHiSSE` that is currently available.

There are a couple major differences with this version of `GeoHiSSE` that user should understand. First, while this version allows for cladogenetic events to be turned off (i.e., `assume.cladogenetic=FALSE`), it does not revert to a three-state MuSSE model as it does in `GeoHiSSE`. Instead, no lineage speciation and extinction are allowed in the widespread state, only transitions out of it. If a three-state MuSSE model is needed, we direct users to read the vignette on how to generate a three-state model in MuHiSSE. Second, `fGeoHiSSE` requires the use of `TransMatMakerfGeoSSE()` for generating the transition rates.

Finally, the other main difference is that, like `hisse`, we employ a modified optimization procedure. In other words, rather than optimizing birth and death separately, `fGeoHisse` optimizes orthogonal transformations of these variables: we let τ define net turnover, and we let ϵ define the extinction fraction. However, these transformations are slightly more complicated due to the dynamics associated with the widespread taxa. For a geographic-based model, we define turnover as,

$$\tau_{00i} = s_{00i} + x_{00i}$$
$$\tau_{11i} = s_{11i} + s_{11i}$$
$$\tau_{01i} = s_{00i} + s_{11i} + s_{01i}$$

We define extinction fraction as

$$\epsilon_{00i} = x_{00i}/s_{00i}$$
$$\epsilon_{11i} = x_{11i}/s_{11i}$$

and because there is no lineage extinction for widespread ranges, $\epsilon_{01i} = 0$.

It is straightforward to convert back to original speciation and extinction, s and x, respectively:

$$s_{00i} = \tau_{00i}/(1 + \epsilon_{00i})$$
$$s_{11i} = \tau_{11i}/(1 + \epsilon_{11i})$$

$$s_{01i} = \tau_{01i} - s_{00i} - s_{11i}$$

$$x_{00i} = (\tau_{00i} * \epsilon_{00i})/(1 + \epsilon_{00i})$$
$$x_{11i} = (\tau_{11i} * \epsilon_{11i})/(1 + \epsilon_{11i})$$

Also, note that the output from `fGeoHiSSE` can be used and processed using available functions. For example, the output can automatically be used to obtain model averages (i.e., `GetModelAveRates()`), generate estimates of the uncertainty in the parameter estimates (i.e., `SupportRegionGeoSSE()`), calculate the marginal probabilities for states at nodes (i.e., `MarginReconfGeoSSE()`), and plotting the rate variation on the tree (i.e., `plot.geohisse.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).

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