Diversification

michelle

3/23/2020

R Markdown

#This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

#install.packages(c("ape", "TreeSim", "geiger", "diversitree", "devtools"))

```
library(ape)
library(TreeSim)
```

```
## Loading required package: geiger
```

```
## Registered S3 method overwritten by 'geiger':
##   method            from
##   unique.multiPhylo ape
```

```
library(geiger)
library(diversitree)
```

#devtools::install_github("thej022214/hisse")

```
library(hisse)
```

```
## Loading required package: deSolve
```

```
## Loading required package: GenSA
```

```
## Loading required package: subplex
```

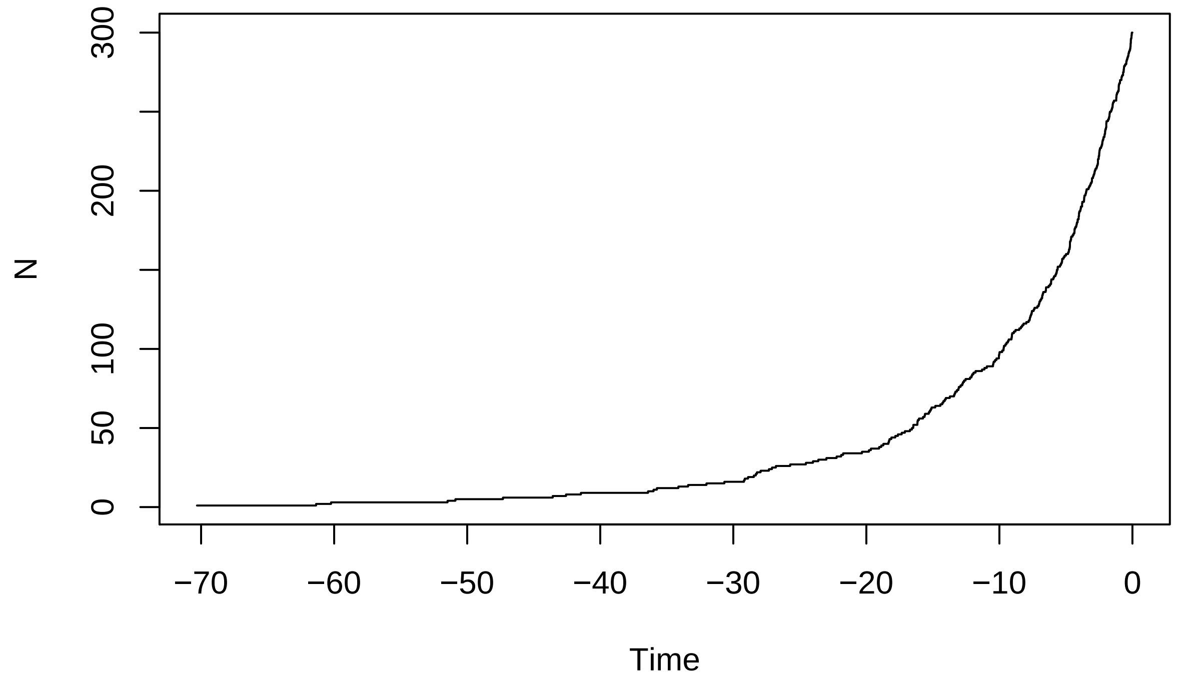
```
## Loading required package: nloptr
```

#Let's initially look just at diversification alone. Simulate a 30 taxon tree with only speciation, no extinction:

```
my.tree <- TreeSim::sim.bd.taxa(n=300, numbsim=1, lambda=0.1, mu=0)[[1]]
```

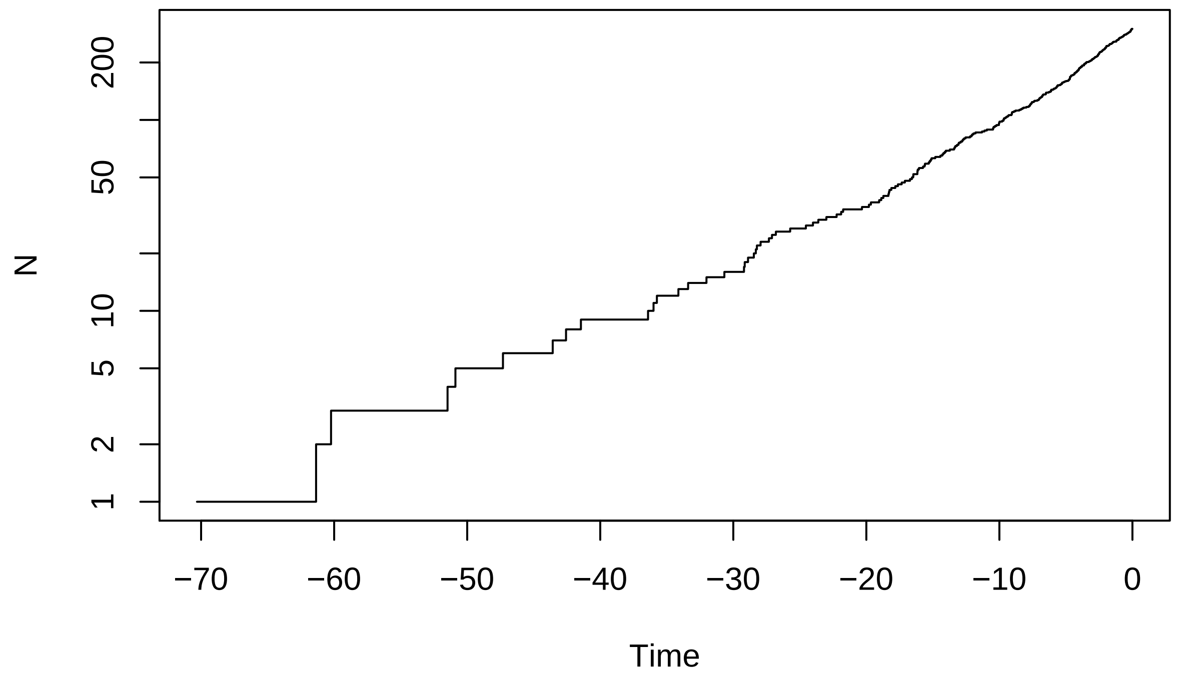
#As always, plot it: #stop("How to plot a tree")

```r
ape::ltt.plot(my.tree)
```



#You should see it increasing exponentially. Let's put it on a log scale:

```r
ape::ltt.plot(my.tree, log="y")
```

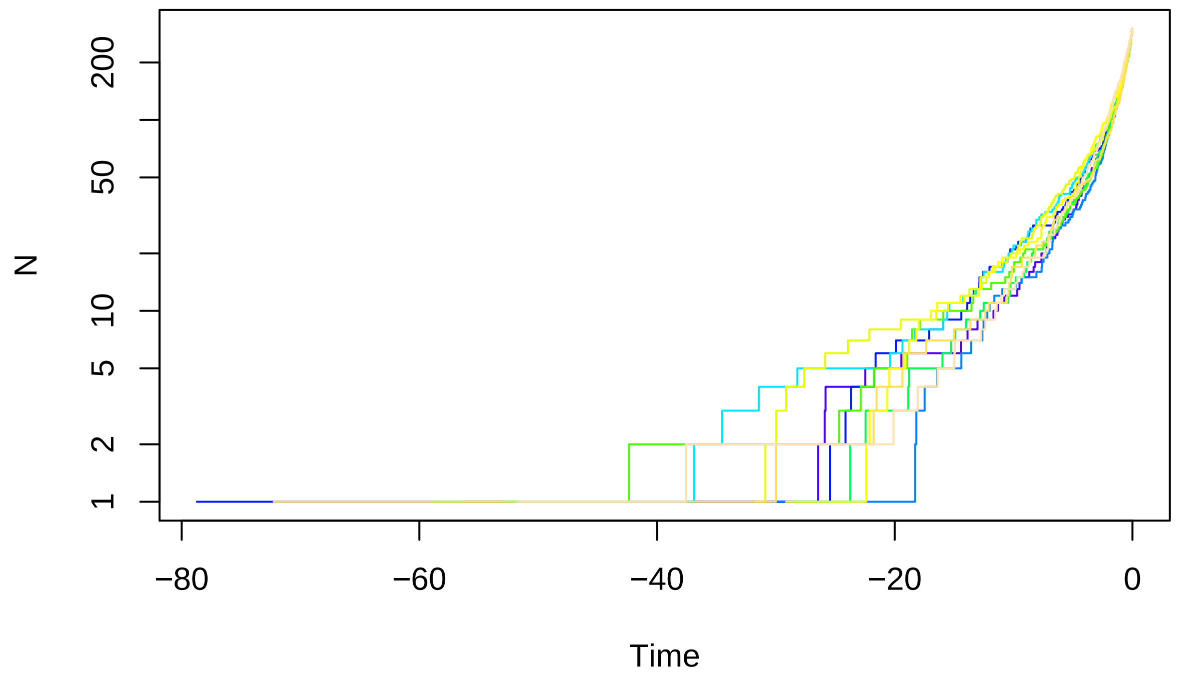


#We can look at multiple trees:

```
yule.trees <- TreeSim::sim.bd.taxa(n=300, numbsim=10, lambda=0.1, mu=0, complete=FALSE)
```

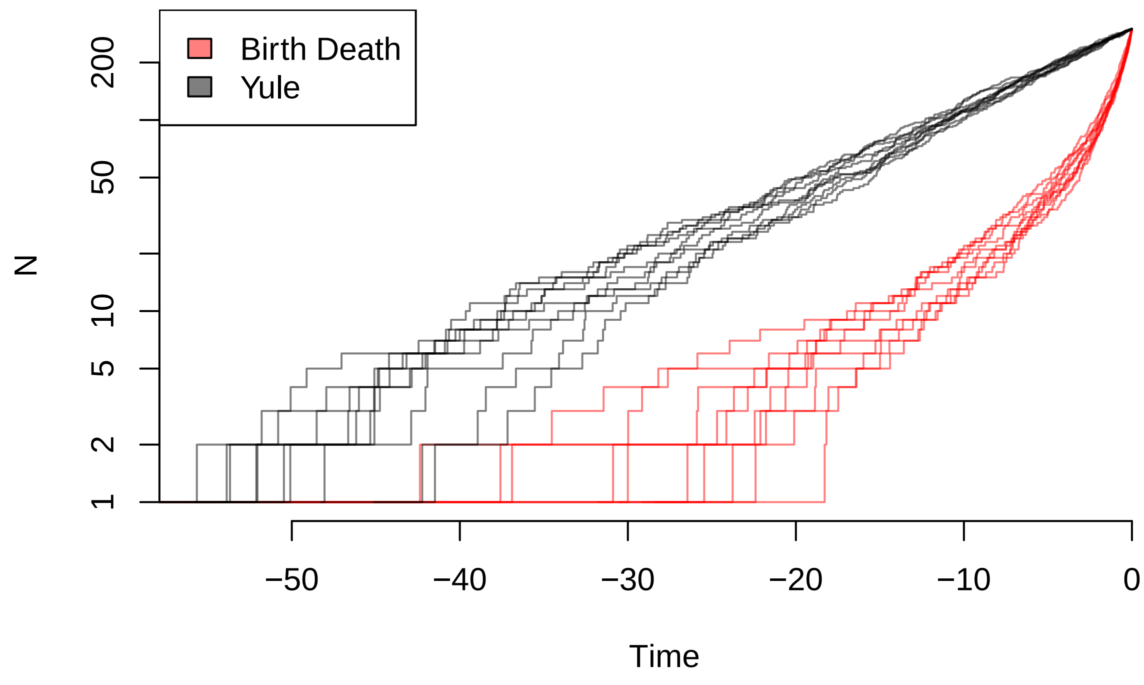
#stop("How to do a multiple ltt plot?") #We can also look at trees with birth and death

```
bd.trees <- TreeSim::sim.bd.taxa(n=300, numbsim=10, lambda=1, mu=.9, complete=FALSE)
ape::mltt.plot(bd.trees, log="y", legend=FALSE)
```



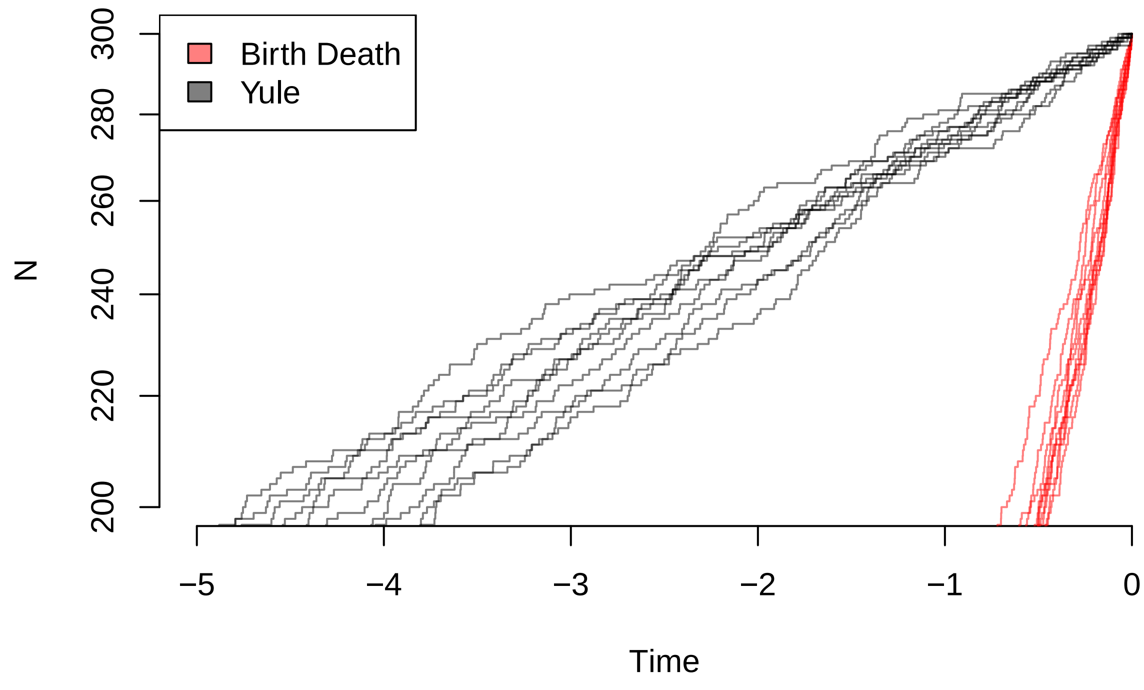
#And compare them:

```
 depth.range <- range(unlist(lapply(yule.trees,ape::branching.times)), unlist(lapply(bd.trees,ape::branc
max.depth <- sum(abs(depth.range)) #ape rescales depths
plot(x=c(0, -1*max.depth), y=c(1, ape::Ntip(yule.trees[[1]])), log="y", type="n", bty="n", xlab="Time",
colors=c(rgb(1,0,0,0.5), rgb(0, 0, 0, 0.5))
list.of.both <- list(bd.trees, yule.trees)
for (i in sequence(2)) {
    tree.list <- list.of.both[[i]]
    for (j in sequence(length(tree.list))) {
        ape::ltt.lines(tree.list[[j]], col=colors[[i]])
    }
}
legend("topleft", legend=c("Birth Death", "Yule"), fill=colors)
```

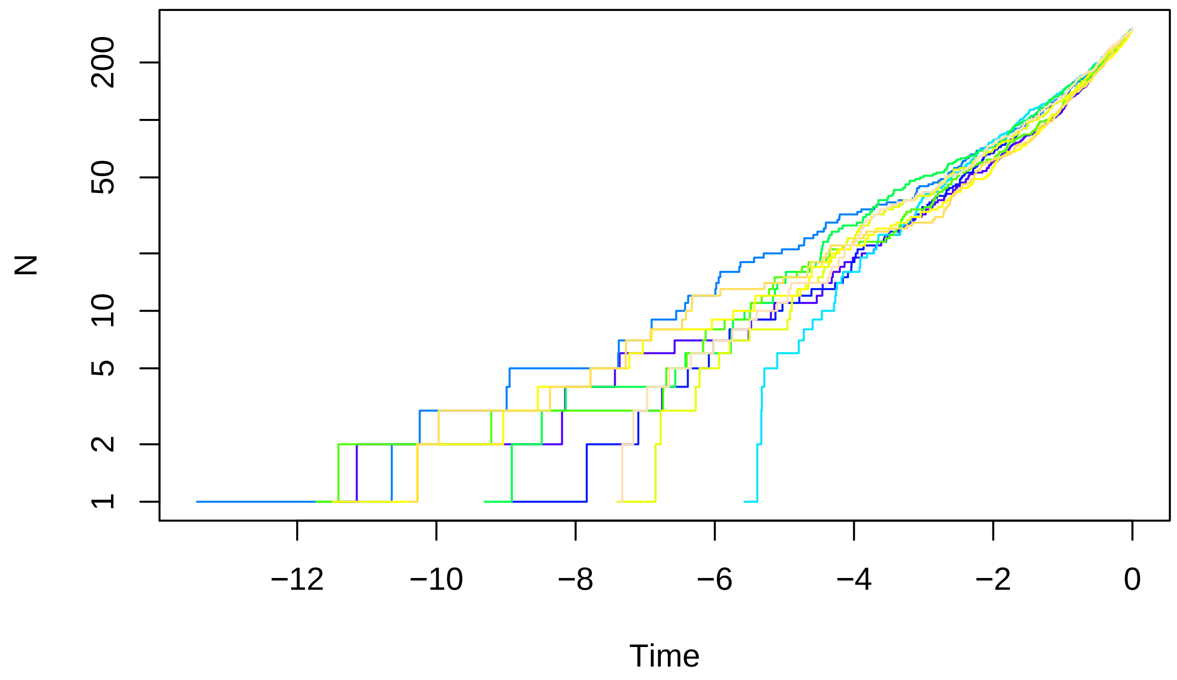


#And zooming in on the final part of the plot

```
depth.range <- range(unlist(lapply(yule.trees,ape::branching.times)), unlist(lapply(bd.trees,ape::branc
max.depth <- sum(abs(depth.range)) #ape rescales depths
plot(x=c(0, -5), y=c(200, ape::Ntip(yule.trees[[1]])), log="y", type="n", bty="n", xlab="Time", ylab="N
colors=c(rgb(1,0,0,0.5), rgb(0, 0, 0, 0.5))
list.of.both <- list(bd.trees, yule.trees)
for (i in sequence(2)) {
    tree.list <- list.of.both[[i]]
    for (j in sequence(length(tree.list))) {
        ape::ltt.lines(tree.list[[j]], col=colors[[i]])
    }
}
legend("topleft", legend=c("Birth Death", "Yule"), fill=colors)
```



```
my.trees <- TreeSim::sim.bd.taxa(n=300, numbsim=10, lambda=1, mu=.5, complete=FALSE)
ape::mltt.plot(my.trees, log="y", legend=FALSE)
```



```
speciation.rates <- c(0.1, 0.1, 0.1, 0.2) #0A, 1A, 0B, 1B
extinction.rates <- rep(0.03, 4)
transition.rates <- c(0.01,0.01,0, 0.01, 0, 0.01, 0.01,0,0.01, 0,0.01,0.01)
pars <- c(speciation.rates, extinction.rates, transition.rates)
phy <- tree.musse(pars, max.taxa=50, x0=1, include.extinct=FALSE)
sim.dat.true <- data.frame(names(phy$tip.state), phy$tip.state)
sim.dat <- sim.dat.true
```

# Now to hide the “hidden” state

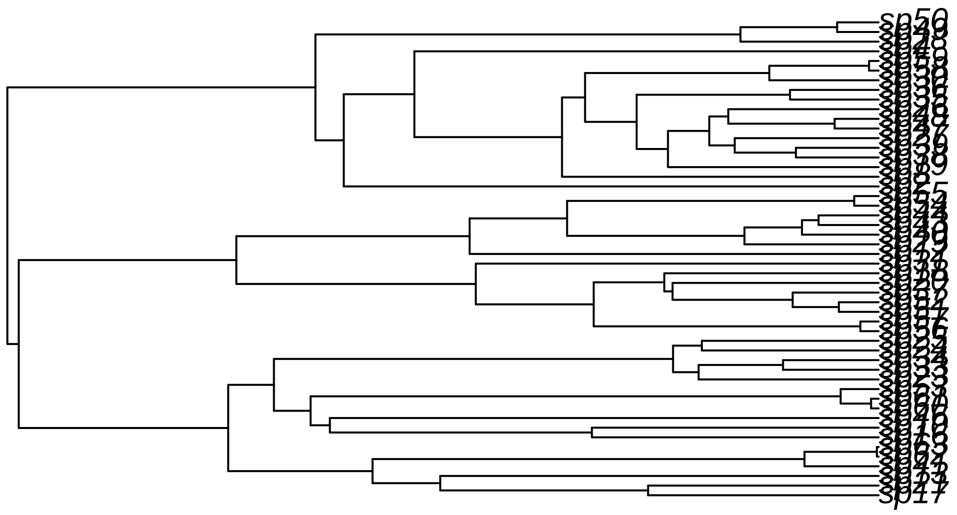
```
sim.dat[sim.dat[,2]==3,2] = 1
sim.dat[sim.dat[,2]==4,2] = 2
```

# and convert states 1,2 to 0,1

```
sim.dat[,2] = sim.dat[,2] - 1
```

#As always, look at what we have wrought:

```
plot(phy)
```



```r
knitr::kable(cbind(sim.dat, true.char=sim.dat.true$phy.tip.state))
```

	names.phy.tip.state.	phy.tip.state	true.char
sp2	sp2	0	1
sp4	sp4	0	1
sp8	sp8	0	1
sp10	sp10	0	1
sp12	sp12	0	1
sp13	sp13	0	1
sp16	sp16	1	2
sp17	sp17	0	1
sp18	sp18	0	3
sp19	sp19	0	1
sp20	sp20	0	3
sp21	sp21	0	3
sp23	sp23	0	1
sp24	sp24	0	1
sp25	sp25	0	1
sp26	sp26	0	1
sp27	sp27	0	1
sp28	sp28	0	1
sp29	sp29	0	1
sp30	sp30	0	1
sp31	sp31	0	1
sp33	sp33	0	1

	names.phy.tip.state.	phy.tip.state	true.char
sp34	sp34	0	1
sp35	sp35	0	1
sp36	sp36	0	1
sp37	sp37	0	3
sp38	sp38	0	1
sp39	sp39	0	1
sp40	sp40	0	1
sp41	sp41	0	1
sp43	sp43	0	1
sp44	sp44	0	1
sp46	sp46	0	1
sp47	sp47	0	1
sp48	sp48	0	1
sp49	sp49	0	1
sp50	sp50	0	1
sp51	sp51	0	3
sp52	sp52	0	3
sp53	sp53	0	1
sp54	sp54	0	3
sp55	sp55	0	3
sp56	sp56	0	3
sp57	sp57	0	3
sp58	sp58	0	3
sp59	sp59	0	3
sp60	sp60	0	1
sp61	sp61	0	1
sp62	sp62	0	1
sp63	sp63	0	1
#Let's	walk through a couple	of examples. Tak	e, for instance, the following index vectors:

```
turnover.anc = c(1,1,0,0)
eps.anc = c(1,1,0,0)
```

```
turnover.anc = c(1,2,0,0)
```

#Thus, a full hisse model would thus be

```
eps.anc = c(0,0,0,0)
```

```
trans.rates = TransMatMaker(hidden.states=TRUE)
trans.rates
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    4    7   10
## (1A)    1   NA    8   11
## (0B)    2    5   NA   12
## (1B)    3    6    9   NA
```

```
trans.rates.nodual = ParDrop(trans.rates, c(3,5,8,10))
trans.rates.nodual
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    3    5    0
## (1A)    1   NA    0    7
## (0B)    2    0   NA    8
## (1B)    0    4    6   NA
```

```
trans.rates.nodual.equal16 = ParEqual(trans.rates.nodual, c(1,6))
trans.rates.nodual.equal16
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    3    5    0
## (1A)    1   NA    0    6
## (0B)    2    0   NA    7
## (1B)    0    4    1   NA
```

```
trans.rates.nodual.allequal = ParEqual(trans.rates.nodual, c(1,2,1,3,1,4,1,5,1,6,1,7,1,8))
trans.rates.nodual.allequal
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    1    1    0
## (1A)    1   NA    0    1
## (0B)    1    0   NA    1
## (1B)    0    1    1   NA
```

```
trans.rates.nodual.allequal = trans.rates.nodual
trans.rates.nodual.allequal[!is.na(trans.rates.nodual.allequal) & !trans.rates.nodual.allequal == 0] =
trans.rates.nodual.allequal
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    1    1    0
## (1A)    1   NA    0    1
## (0B)    1    0   NA    1
## (1B)    0    1    1   NA
```

```
trans.rates.bisse = TransMatMaker(hidden.states=FALSE)
trans.rates.bisse
```

```
##     (0) (1)
## (0)  NA   2
## (1)   1  NA
```

```
pp = hisse(phy, sim.dat, f=c(1,1), hidden.states=TRUE, turnover.anc=turnover.anc,
           eps.anc=eps.anc, trans.rate=trans.rates.nodual.allequal)
```

```
## Initializing...
## Finished. Beginning bounded subplex routine...
## Finished. Summarizing results...
```

```
turnover.anc = c(1,2,0,3)
eps.anc = c(1,2,0,3)
```

```
trans.rates <- TransMatMaker(hidden.states=TRUE)
trans.rates.nodual.no0B <- ParDrop(trans.rates, c(2,3,5,7,8,9,10,12))
trans.rates.nodual.no0B
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    2    0    0
## (1A)    1   NA    0    4
## (0B)    0    0   NA    0
## (1B)    0    3    0   NA
```

```
pp = hisse(phy, sim.dat, f=c(1,1), hidden.states=TRUE, turnover.anc=turnover.anc,
           eps.anc=eps.anc, trans.rate=trans.rates.nodual.allequal, output.type="net.div")
```

```
## Initializing...
## Finished. Beginning bounded subplex routine...
## Finished. Summarizing results...
```

```
turnover.anc = c(1,1,2,2)
eps.anc = c(1,1,2,2)
```

```
trans.rates = TransMatMaker(hidden.states=TRUE)
trans.rates.nodual = ParDrop(trans.rates, c(3,5,8,10))
```

```
trans.rates.nodual.allequal = ParEqual(trans.rates.nodual, c(1,2,1,3,1,4,1,5,1,6,1,7,1,8))
trans.rates.nodual.allequal
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    1    1    0
## (1A)    1   NA    0    1
## (0B)    1    0   NA    1
## (1B)    0    1    1   NA
```

#Now we want three specific rates:

```
trans.rates.nodual.threerates <- trans.rates.nodual
```

# Set all transitions from 0->1 to be governed by a single rate:

```
to.change <- cbind(c(1,3), c(2,4))
trans.rates.nodual.threerates[to.change] = 1
```

# Now set all transitions from 1->0 to be governed by a single rate:

```
to.change <- cbind(c(2,4), c(1,3))
trans.rates.nodual.threerates[to.change] = 2
```

# Finally, set all transitions between the hidden state to be a single rate (essentially giving # you an estimate of the rate by which shifts in diversification occur:

```r
to.change <- cbind(c(1,3,2,4), c(3,1,4,2))
trans.rates.nodual.threerates[to.change] = 3
trans.rates.nodual.threerates
```

```
##      (0A) (1A) (0B) (1B)
## (0A)   NA    1    3    0
## (1A)    2   NA    0    3
## (0B)    3    0   NA    1
## (1B)    0    3    2   NA
```

```r
pp = hisse(phy, sim.dat, f=c(1,1), hidden.states=TRUE, turnover.anc=turnover.anc,
           eps.anc=eps.anc, trans.rate=trans.rates.nodual.allequal)
```

```
## Initializing...
## Finished. Beginning bounded subplex routine...
## Finished. Summarizing results...
```

system(“ls”)

```r
load("/Users/modoi/Downloads/testrecon1.rda")
```

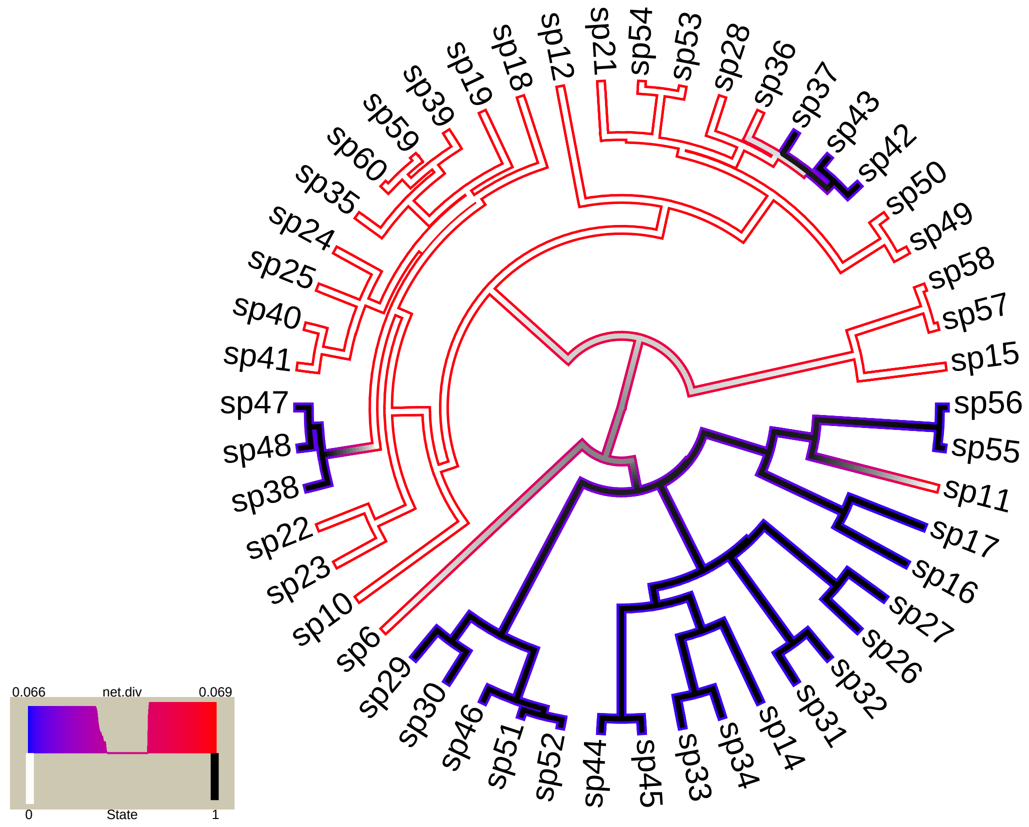
```r
load("testrecon1.rda")
class(pp.recon)
```

```
## [1] "hisse.states"
```

```r
pp.recon
```

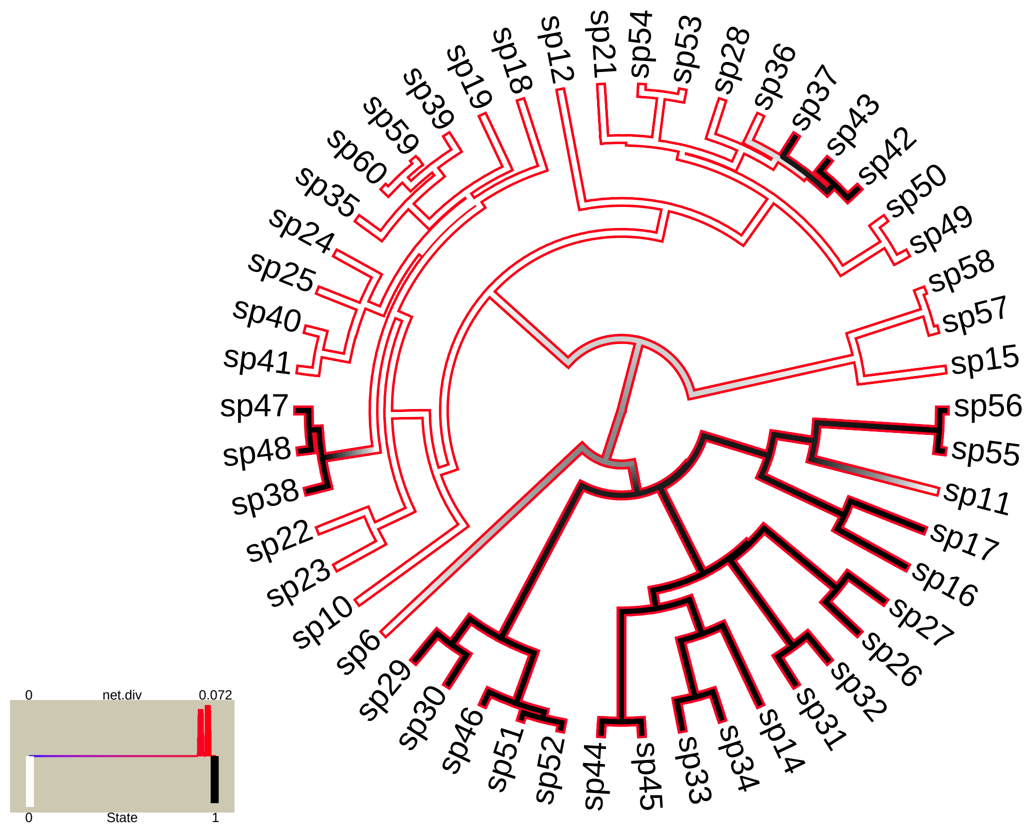
```
##
## Phylogenetic tree with 50 tips and 49 internal nodes.
##
## Tip labels:
##  sp15, sp57, sp58, sp49, sp50, sp42, ...
## Node labels:
##  1, 3, 1, 1, 1, 1, ...
##
## Rooted; includes branch lengths.
```

```r
plot.hisse.states(pp.recon, rate.param="net.div", show.tip.label=TRUE)
```



```
## $rate.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
##
## (2) A mapped continuous trait on the range (0.066161, 0.069176).
##
##
## $state.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
##
## (2) A mapped continuous trait on the range (0, 1.001).
```

```
plot.hisse.states(pp.recon, rate.param="net.div", show.tip.label=TRUE, rate.range=c(0,0.072))
```



```
## $rate.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
##
## (2) A mapped continuous trait on the range (0, 0.072072).
##
##
## $state.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
##
## (2) A mapped continuous trait on the range (0, 1.001).
```

```
pp.recon$aic
```

```
## [1] 364.8615
```

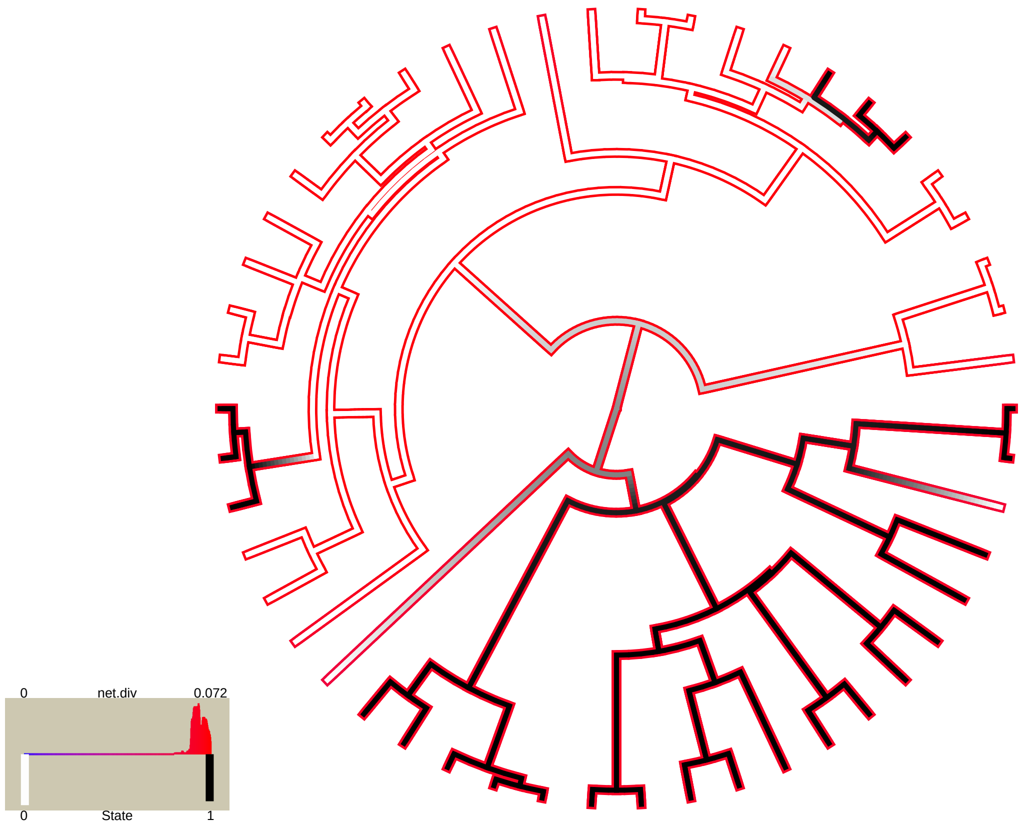
```
pp.recon = MarginRecon(phy, sim.dat, f=c(1,1), hidden.states=TRUE, pars=pp$solution,
```

```
load("/Users/modoi/Downloads/testrecon1.rda")
load("/Users/modoi/Downloads/testrecon2.rda")
load("/Users/modoi/Downloads/testrecon3.rda")
```

```
hisse.results.list = list()
load("testrecon1.rda")
hisse.results.list[[1]] = pp.recon
load("testrecon2.rda")
hisse.results.list[[2]] = pp.recon
load("testrecon3.rda")
hisse.results.list[[3]] = pp.recon
```

# Now supply the list the plotting function

```
plot.hisse.states(hisse.results.list, rate.param="net.div", show.tip.label=FALSE, rate.range=c(0,0.072)
```



```
## $rate.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
##
## (2) A mapped continuous trait on the range (0, 0.072072).
##
##
## $state.tree
## Object of class "contMap" containing:
##
## (1) A phylogenetic tree with 50 tips and 49 internal nodes.
```

```
##
## (2) A mapped continuous trait on the range (0, 1.001).
```

# First, suck in all the files with .Rsave line ending in your working directory:

```r
files = system("ls -1 | grep .rda", intern=TRUE)
```

# Create an empty list object

```r
hisse.results.list = list()
```

# Now loop through all files, adding the embedded pp.recon object in each

```r
for(i in sequence(length(files))){
  load(files[i])
  hisse.results.list[[i]] = pp.recon
  rm(pp.recon)
}
```