Lab 4: Modeling Discrete Trait Evolution

Comparative Biology and Macroevolution

5/3/2019

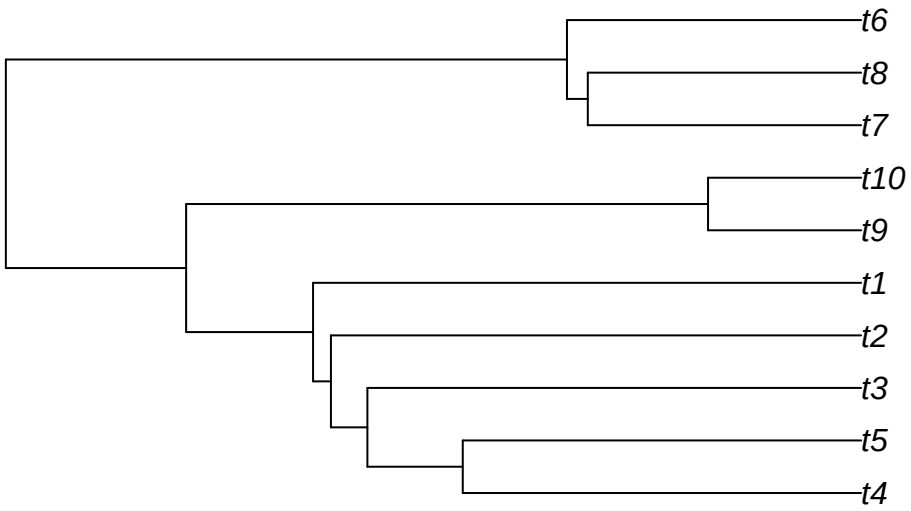
Classwork

Load libraries

```
library(ape)
library(geiger)
library(phytools)
```

Read in tree and data

```
simpleTree <- read.tree(file = "simpleTree.tre")
simpleTree <- ladderize(simpleTree)
plot(simpleTree)
```




```
simpleData <- read.csv(file = "simpleData.csv", stringsAsFactors = F)
simpleData
```

```
##    Taxon        State
## 1     t1 Non-Venomous
## 2     t2     Venomous
## 3     t3     Venomous
## 4     t4     Venomous
## 5     t5     Venomous
## 6     t6 Non-Venomous
## 7     t7     Venomous
## 8     t8     Venomous
## 9     t9 Non-Venomous
## 10   t10 Non-Venomous
```

```
# cbind binds two vectors as columns
cbind(simpleData$Taxon, simpleTree$tip.label) # check trait and tip order
```

```
##       [,1]  [,2]
##  [1,] "t1"  "t6"
##  [2,] "t2"  "t7"
##  [3,] "t3"  "t8"
##  [4,] "t4"  "t9"
##  [5,] "t5"  "t10"
##  [6,] "t6"  "t1"
##  [7,] "t7"  "t2"
##  [8,] "t8"  "t4"
##  [9,] "t9"  "t5"
## [10,] "t10" "t3"
```

Question: What is wrong with the order of the data?

```
rownames(simpleData) <- simpleData$Taxon # set rownames of traits
simpleData <- simpleData[match(simpleTree$tip.label, rownames(simpleData)),] # match traits and tips
cbind(simpleData$Taxon, simpleTree$tip.label) # verify trait and tips are same order
```

```
##       [,1]  [,2]
##  [1,] "t6"  "t6"
##  [2,] "t7"  "t7"
##  [3,] "t8"  "t8"
##  [4,] "t9"  "t9"
##  [5,] "t10" "t10"
##  [6,] "t1"  "t1"
##  [7,] "t2"  "t2"
##  [8,] "t4"  "t4"
##  [9,] "t5"  "t5"
## [10,] "t3"  "t3"
```

```
State <- simpleData$State # create vector of trait states
names(State) <- simpleData$Taxon # set names for trait states
State <- as.factor(State) # convert discrete trait into factor
State
```

```
##           t6           t7           t8           t9          t10
## Non-Venomous     Venomous     Venomous Non-Venomous Non-Venomous
##           t1           t2           t4           t5           t3
## Non-Venomous     Venomous     Venomous     Venomous     Venomous
## Levels: Non-Venomous Venomous
```

We can see the strings are now factors, with 2 Levels: Non-Venomous and Venomous

Fit models using `fitDiscrete()`. The `fitDiscrete` function uses a maximum likelihood frameowrk to extimate parameters and the likelihood for univariate datasets.

```
fitER <- fitDiscrete(phy = simpleTree, dat = State, model = "ER") # equal transition rates
fitSYM <- fitDiscrete(phy = simpleTree, dat = State, model = "SYM") # symmetric transition rates
fitARD <- fitDiscrete(phy = simpleTree, dat = State, model = "ARD") # all rates different
```

Take a look inside the returned list

```
fitER
```

```
## GEIGER-fitted comparative model of discrete data
##  fitted Q matrix:
##                   Non-Venomous   Venomous
##     Non-Venomous   -0.8190595  0.8190595
##     Venomous        0.8190595 -0.8190595
```

```
##
##  model summary:
##  log-likelihood = -6.600026
##  AIC = 15.200052
##  AICc = 15.700052
##  free parameters = 1
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

How many parameters are in the equal-rates (ER) model?

Select model using AIC

```
Venom <- c(fitER$opt$aic, fitSYM$opt$aic, fitARD$opt$aic)
names(Venom) <- c("Equal Rates", "Symmetric Rates", "All Rates Different")
Venom
```

```
##         Equal Rates     Symmetric Rates All Rates Different
##            15.20005            15.20005            16.72950
```

Question: Why are the AIC Scores identical for Equal Rates and Symmetric Rates?

```
fitER
```

```
## GEIGER-fitted comparative model of discrete data
##  fitted Q matrix:
##                   Non-Venomous   Venomous
##     Non-Venomous   -0.8190595  0.8190595
##     Venomous        0.8190595 -0.8190595
##
##  model summary:
##  log-likelihood = -6.600026
##  AIC = 15.200052
##  AICc = 15.700052
##  free parameters = 1
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

```
fitSYM
```

```
## GEIGER-fitted comparative model of discrete data
##  fitted Q matrix:
##                  Non-Venomous   Venomous
##     Non-Venomous   -0.8190595  0.8190595
##     Venomous        0.8190595 -0.8190595
##
##  model summary:
##  log-likelihood = -6.600026
##  AIC = 15.200052
##  AICc = 15.700052
##  free parameters = 1
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

Marginal ancestral statereconstruction

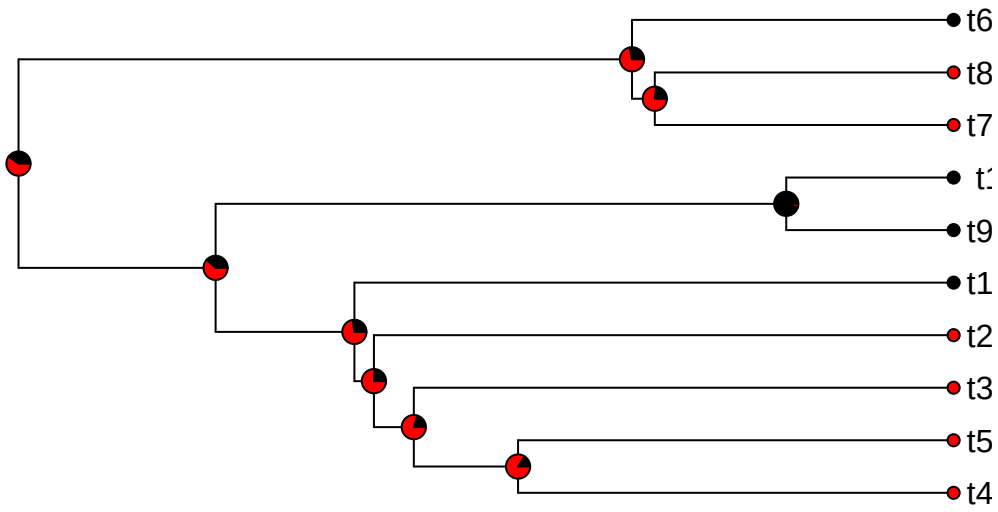
Marginal ancestral state estimation for each internal node of the tree using maximum likelihood.

```
marginal_ER_fit <-  rerootingMethod(tree = simpleTree, x = State, model = "ER")
marginal_ER_fit
```

```
## Ancestral character estimates using re-rooting method
## of Yang et al. (1995):
##    Non-Venomous Venomous
## 11     0.401438 0.598562
## 12     0.280924 0.719076
## 13     0.232534 0.767466
## 14     0.383877 0.616123
## 15     0.949767 0.050233
## 16     0.272260 0.727740
## 17     0.239883 0.760117
## 18     0.203119 0.796881
## 19     0.158418 0.841582
##
## Estimated transition matrix,
## Q =
##              Non-Venomous  Venomous
## Non-Venomous    -1.066515  1.066515
## Venomous         1.066515 -1.066515
##
## **Note that if Q is not symmetric the marginal
## reconstructions may be invalid.
##
## Log-likelihood = -6.660988
```

Plot the estimated marginal ancestral states on the tree

```
plot(simpleTree, show.tip.label = F)
tiplabels(simpleTree$tip.label, adj = -0.5, frame = "none")
nodelabels(node = as.numeric(rownames(marginal_ER_fit$marginal.anc)), pie = marginal_ER_fit$marginal.an
    piecol = c("black", "red"), cex = 0.6)
tiplabels(pie = to.matrix(State, sort(unique(State))), piecol = c("black", "red"),
    cex = 0.3)
```



Stochastic Character Mapping

An alternative method is to use an MCMC approach to sample character histories from their posterior probability distribution. This is called stocastic character mapping (Huelsenbeck et al. 2003). Instead of getting a probability distribution for the characters at nodes, we get a sample of histories for our discrete character's evolution on the tree.

Use AIC-selected model for stochastic character mapping. Simulate and plot 100 character histories

```
mtrees <- make.simmap(tree = simpleTree, x = State, model = "ER", nsim = 100)
```

```
## make.simmap is sampling character histories conditioned on the transition matrix
##
## Q =
##               Non-Venomous  Venomous
## Non-Venomous    -1.066515  1.066515
## Venomous         1.066515 -1.066515
## (estimated using likelihood);
## and (mean) root node prior probabilities
## pi =
## Non-Venomous     Venomous
##          0.5          0.5

## Done.
```

Plot all 100 histories

Get colors for the states

```
cols <- setNames(object = palette()[1:length(unique(State))], nm = sort(unique(State)))
cols
```

```
## Non-Venomous     Venomous
##      "black"        "red"
```

```
cols <- setNames(object = c("black", "red"), nm = unique(State)) # This does the same thing
cols
```

```
## Non-Venomous     Venomous
##      "black"        "red"
```

```
par(mfrow = c(10, 10)) # set plot window to 10 rows by 10 columns
null <- sapply(X = mtrees, FUN = plotSimmap, colors = cols, lwd = 1, ftype = "off") # plot
```

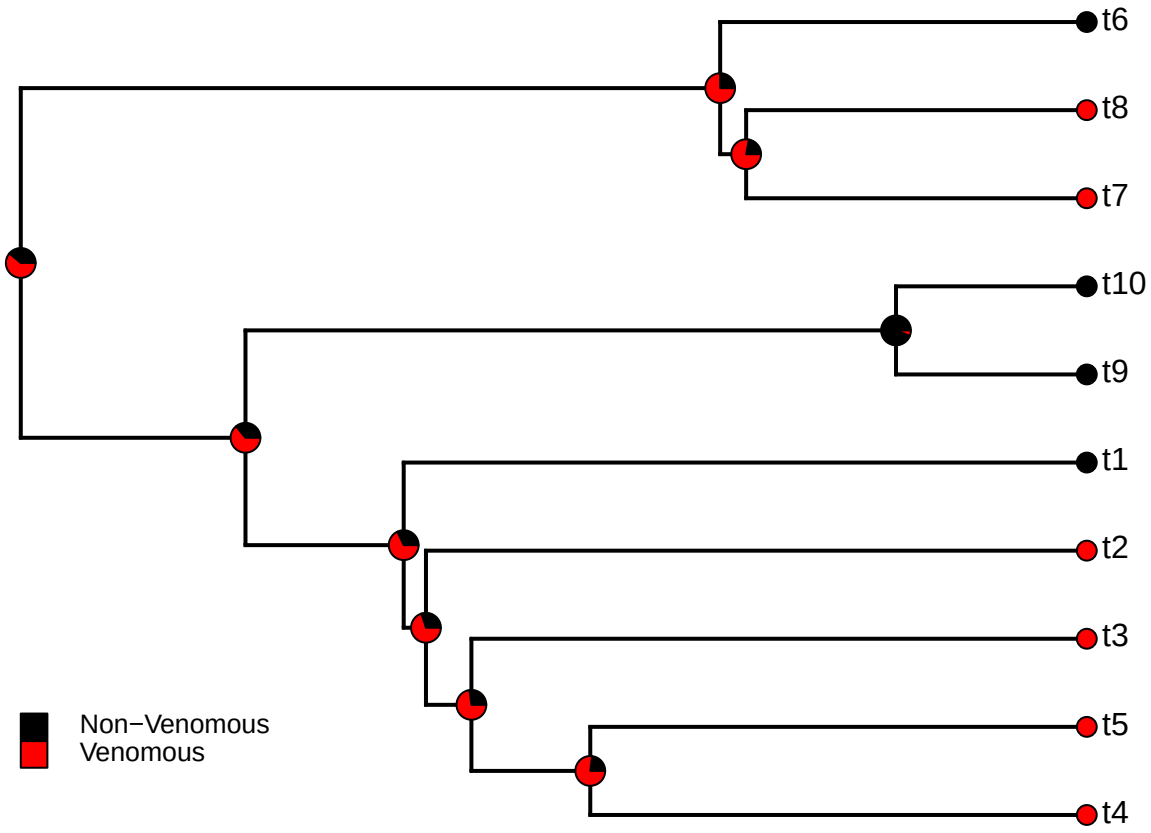
With an aggregate of stocahstic character mpas, we can estimate the number of changes of each type, the proportion of time spent in each state, and the posterior probabilities that each internal node is in each state.

```
pd <- describe.simmap(tree = mtrees, plot = FALSE)
pd # This has important information about the ancestral states
```

```
## 100 trees with a mapped discrete character with states:
##  Non-Venomous, Venomous
##
## trees have 7.71 changes between states on average
##
## changes are of the following types:
##      Non-Venomous,Venomous Venomous,Non-Venomous
## x->y                  3.69                  4.02
##
## mean total time spent in each state is:
##      Non-Venomous Venomous    total
## raw      2.524000 4.027131 6.551131
## prop     0.385277 0.614723 1.000000
```

Plot posterior probability of states on nodes with legend

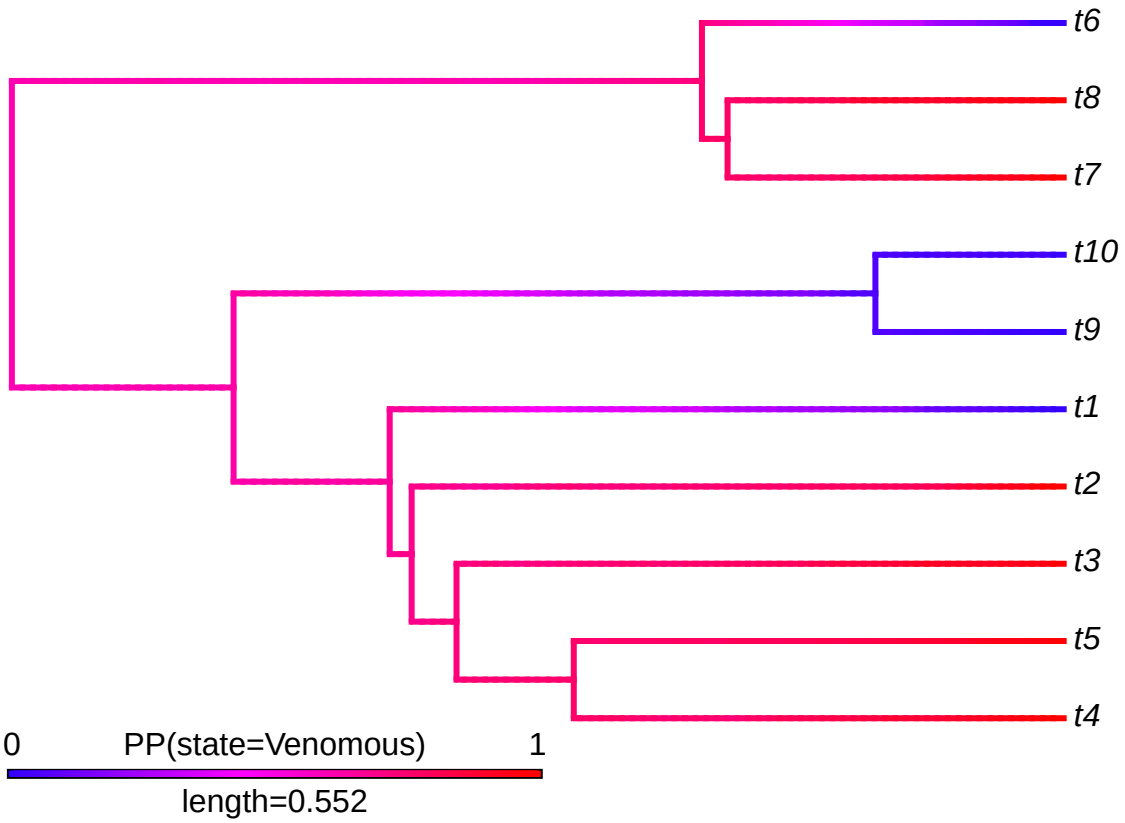
```
par(mfrow = c(1,1)) # reset graphing parameters
plot(pd)
add.simmap.legend(colors = cols, prompt = F, x = 0, y = 2, fsize = 0.8)
```



Plot posterior probability of states on branches

```
densityMap(mtrees)
```

```
## sorry - this might take a while; please be patient
```



Examine how quickly the color changes along the branches. You can think of this as the amount of uncertainty around when a transition occured. When is there more or less uncertainty about the timing of transitions?

Homework

Did grunts originate on reefs?

1. Reconstruct ancestral habitat states of grunts using stochastic character mapping. Standard lab report format applies.
2. Use `grunt.tre` and `grunts.csv`
3. Fit ER and ARD models (why are you not fitting SYM to reef/non-reef habitat states?). Select best model using AIC.
4. Use the AIC-selected model for stochastic mapping (minimum 100 simulated character histories). Plot posterior probability of states on nodes with legend. Plot posterior probability of states on branches using `densityMap()`
5. Provide a biological interpretation of your selected model. Describe the evolutionary history of habitat association in grunts.