Lab 07: Intro to phylogenetics in R

Carrie Tribble

2/28/2018

Integrative Biology 200

Principles of Phylogenetics

University of California, Berkeley

In this lab we will introduce some of the basic packages and functions for running comparative analyses in R. This is an R Markdown document. You can run the code by copying from the PDF into your R Console or you can open the .Rmd file and run the lines of code in each 'code chunk'. If you open the .Rmd file, you can make edits and save your very own PDF with your own additions and answers to the question. 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. If you choose to modify the markdown file, please remove the `include=FALSE` line from the r setup code chunk. This way I'll see the output of your code in your final knitted document. Also change the author name to your name.

Before beginning this lab, please set your working directory to an appropriate folder on your computer, for example:

```
setwd("~/Desktop/lab07")
```

Part 1: Basic Tree Functions and Data Structure

There are many packages in R that are commonly used for phylogenetic comparative methods. To avoid having to individually install each package, this will install all relevant packages at once. This will take a bit, but you'll be ready to run all the phylogenetic analyses you could ever want! Be sure to uncomment these lines of code when you want to run them.

```
install.packages("ctv")

library(ctv)

install.views("Phylogenetics")
```

First, we will go over some basic tree functions in R. We will need to load some of the packages you just installed. You can specify a tree (with branch lengths) using the newick parenthetical format:

```
library(ape)

tree <- read.tree(text = "(((A:0.2,B:0.1):0.3,(C:0.5,D:0.1):0.2):0.1,E:0.5);")

plot(tree, edge.width=2)
```

Because we specified branch lengths in the tree file, this is a phylogram rather than a cladogram. We can extract those branch lengths using the following function:

```
tree$edge.length
```

We can also examine the structure of the data object. When working in R, the str command can be helpful, especially when you are dealing with new data formats such as phylogenies.

```
str(tree)
```

Let's look at how nodes and tips are labeled in this data structure. This can be helpful if you are trying to identify a particular tip or node of your tree to remove or to label.

```
tree$edge
```

The edge matrix contains the beginning and ending node number for all the nodes and tips in the tree. By convention, the tips of the tree are numbered 1 through n for n tips; and the nodes are numbered n + 1 through n + m for m nodes. m = n - 1 for a fully bifurcating tree. This is just to keep track of which nodes are internal and which are leaves.

This will make more sense if we plot the labels on the tree itself.

```
plot(tree, edge.width = 2, label.offset = 0.1, type = "cladogram")

nodelabels()

tiplabels()
```

Next, we we will load some sample data. Load the following packages, and the sample dataset on Teleost fishes:

```
library(adephylo)

library(phylobase)

data(mjrochet)
```

We have now downloaded a file containing a tree and associated trait data. Take a look at the structure of the mjrochet object. Then save the tree as a separate object. Then you can plot the tree to see what we've downloaded.

```
str(mjrochet)

teleost_tree <- read.tree(text=mjrochet$tre)

plot(teleost_tree)
```

Examine a list of the tips and choose one to reroot the tree with. Feel free to choose a different taxa to reroot the tree with.

```
teleost_tree$tip.label

teleost_reroot <- root(teleost_tree, "Lutjanus_purpureus")
```

Now, let's plot both the original and rerooted trees to compare

```
plot(teleost_tree, main = "Original")

plot(teleost_reroot, main = "Rerooted")
```

Let's plot some character data on the tree. We can link the trait data with the tree by making a Phylo4D object. This is a fancy way of saying that we are creating a new data structure in R where the trait data is

associated with the tips of the tree object. We will get the character matrix from the same mjrochet object where we got the tree. Finally, we can plot all these characters next to the tips of the tree.

```
teleost_4d <- phylo4d(x=teleost_tree, tip.data=mjrochet$tab)

table.phylo4d(teleost_4d, cex.lab=.5,show.node=FALSE)
```

This tutorial by Liam J. Revell has some great pointers for plotting characters on trees (for discrete AND continuous traits): http://www.phytools.org/Cordoba2017/ex/15/Plotting-methods.html. We'll see some other methods later today and in next week's lab.

Part 2: Maximum Likelihood Ancestral State Reconstruction of Discrete Characters

In R we have used a number of functions. Almost every command you have run is a function that takes certain arguments as the inputs and produces some output, such as a value, a datatable, or a plot. We can also define our own functions. Here's an example of a silly function. Taking the mean of a vector with some element NA will produce NA as the answer unless you specify to remove NA values when calculating the mean. To get around that, let's create our own function that removes the NA elements automatically.

```
mean_na <- function(vec) {

  m <- mean(vec, na.rm = T)

  return(m)

}
```

You can test out our new function.

```
test <- c(1,3,4,NA, 8, 1)

mean(test)

mean_na(test)
```

Exercise A

Page 2 of lecture 17 describes a continuous-time Markov model for a binary character that has two rates of change: alpha is the rate of transitioning from state 0 to 1, and beta is the rate of transitioning from state 1 to 0. See: http://ib.berkeley.edu/courses/ib200/lect/lect17.pdf.

Write two functions to calculate the probability of each possible transition over a branch of length t. Each function should take as input alpha, beta, and t. As a reminder, alpha and beta are the instantaneous rates of change - your function should calculate the probabilites. Insert your code below.

```
#insert your code here
```

Great! Now, let's use those functions to estimate the probabilities of states on a simple tree.

```r
t = read.tree(text="((A:0.39,B:0.39):0.93,C:1.32);")

plot(t, show.tip.label=FALSE, main = "Test Tree")

nodelabels()

tiplabels()

edgelabels(t$edge.length)
```

Exercise B

Assume alpha = 2 and beta = 3. Now estimate the probabilities that the tree has the following states:

Node 1 = 0
Node 2 = 0
Node 3 = 1
Node 4 = 1
Node 5 = 0

Insert your code below.

```r
#insert your code here
```

It would be a pain if we had to do this manually every time! Thankfully, there are packages with functions that can do these calculations for us. We'll use an example from Liam J. Revell's Phytools package. Phytools is one of the most commonly used R packages for phylogenetic comparative methods, and the Phytools blog can be incredibly helpful: http://blog.phytools.org/

We will get some sample data from the Phytools package.

```r
library(phytools)

data(anoletree)

x <- getStates(anoletree, "tips")

tree <- anoletree
```

Plot the data on the tree!

```r
plotTree(tree, type = "fan", fsize = 0.9, ftype = "i")

cols <- setNames(palette()[1:length(unique(x))], sort(unique(x)))

tiplabels(pie = to.matrix(x, sort(unique(x))), piecol = cols, cex = 0.2)

add.simmap.legend(colors = cols, prompt = FALSE,
                  x = 0.9 * par()$usr[1], y = -max(nodeHeights(tree)))
```

Here, we will fit a single-rate continuous time Markov model to the data and estimate the of the ancestral nodes of our tree. In the above exercise, we asked you to calculate the probabilities using a two-rate model.

Since there are 6 character states, the instantaneous rate matrix looks like this:

```
-   a   a   a   a   a
a   -   a   a   a   a
a   a   -   a   a   a
a   a   a   -   a   a
a   a   a   a   -   a
a   a   a   a   a   -
```

where the diagonals are -5a

Fit the model using the following code. "ER" means equal rates! Then calculate the log likelihood of the model.

```
fitSR <- rerootingMethod(tree, x, model = "ER")

fitSR$loglik
```

We can plot the results of our ancestral state reconstruction, now showing the ancestral states at the nodes.

```
plotTree(tree, type = "fan", fsize = 0.9, ftype = "i")

nodelabels(node = as.numeric(rownames(fitSR$marginal.anc)),
           pie = fitSR$marginal.anc, piecol = cols, cex = 0.5)

tiplabels(pie = to.matrix(x, sort(unique(x))), piecol = cols, cex = 0.2)

add.simmap.legend(colors = cols, prompt = FALSE,
                  x = 0.9 * par()$usr[1], y = -max(nodeHeights(tree)))
```

Now let's fit a symmetrical-rates model, that allows different (but symmetrical) rates for each character state transition. For a symmetrical-rate 6 state model, the transition rate matrix looks like:

```
-   a   b   c   d   e
a   -   f   g   h   i
b   f   -   j   k   l
c   g   j   -   m   n
d   h   k   m   -   o
e   i   l   n   o   -
```

where the diagonal elements are defined as -1 times the sum of the other row elements, for example, row 1's diagonal element is -(a + b + c + d + e).

We fit the symmetrical rates model using the same code as above, but specifying "SYM" for symmetrical instead of "ER" for equal rates. Also calculate the log likelihood of this model.

```
fitSYM <- rerootingMethod(tree, x, model = "SYM")

fitSYM$loglik
```

Exercise C

Plot the tree again, but show the ancestral states inferred using the symmetrical-rates model. Send me screen shots of both reconstructed ancestral states. Even though the single-rate model is simply a special case of the symmetrical-rates model, are the ancestral state reconstructions the same?

Which model fits the data better? Calculate the likelihood ratio test: D = 2 * (loglike of alternative model - loglike of null model). There are 15 parameters in the alternative (SYM) model and 1 parameter in the SR model, so we have 15 - 1 = 14 degrees of freedom. Look up D in a chi-squared distribution table and report the p-value. Is the SYM model supported over the SR model?

```
#insert your code here
```

Part 3: Correlated Evolution of Discrete Traits

Pagel (1994) described an elegant model to test for correlated evolution of discrete traits. Here, we can model two binary traits (with states 0 and 1) as one combined multistate trait (with states 00, 01, 10, 11), illustrated in the table below.

	Binary Traits		Multistate
	Trait1	Trait2	Combined
OTU1	0	1	01
OTU2	1	1	11
OTU3	1	0	10
OTU4	0	0	00
OTU5	1	1	11

We can then perform an ancestral state reconstruction and estimate the rates of transition between all four 'states'. Figure 1 illustrates the basic set-up of the model. As you can see, there are separate parameters for transitioning between all 4 states. Let's think about a biological example. Suppose we are interested in testing for a correlation between two binary traits: presence/ absence of tubers and presence/ absence of rhizomes. Are plants that have tubers more likely to also have rhizomes? We can rephrase this question as, does the rate of evolving rhizomes depend on if the plant already has tubers? If the traits are correlated, we expect that the rate of evolving tubers (trait 1) depends on the presence of rhizomes (trait 2).

We can compare the statistical fit of the fully correlated model (Fig. 1) to the statistical fit of an alternative model. In the alternative model, we force some rates to be equal such that the rate of transitioning between states of one character is the same, regardless of the state of the other character. In our biological example above, we constrain the rates of evolving tubers with and without rhizomes to be the same.

Let's simulate some fake data and give it a try.

First, let's simulate a tree with 200 tips.

```
tree <- pbtree(n = 200, scale = 1)

plotTree(tree, type = "fan")
```

Now, let's simulate two traits evolving independently. We do this by building a rate matrix and simulating the evolution of a binary trait, twice. Keep in mind that even though we use the same rate matrix, the traits are still evolving independently.

```
Q <- matrix(c(-1,1,1,-1),2,2)

rownames(Q) <- colnames(Q) <- letters[1:2]

Q

binary1 <- sim.history(tree,Q)
```

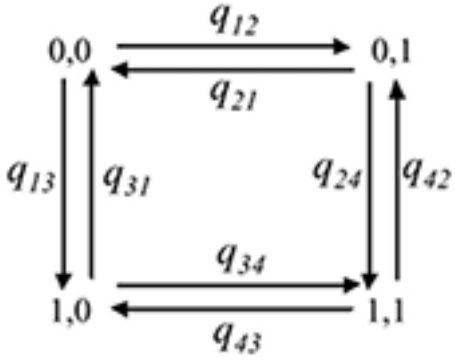

Figure 1: Image from Pagel and Meade (2006). The arrows indicate transition rates between states.

```
binary2 <- sim.history(tree,Q)
```

Let's plot the results of our simulation to see if they look correlated or not. Remember, we simulated the data separately, so they really shouldn't be correlated!

```
par(mfrow=c(1,2))

plotSimmap(binary1, setNames(c("blue","red"),
                             letters[1:2]), ftype="off", lwd=1)

plotSimmap(binary2, setNames(c("blue","red"),
                             letters[1:2]), ftype="off", lwd=1, direction="leftwards")
```

And now, let's fit Pagel's test for correlated evolution to the data:

```
x <- binary1$states

y <- binary2$states

fit <- fitPagel(tree, x, y)

fit
```

Exercise D

Take a close look at this output. Does your test indicate that traits are correlated or not? Which model was favored, and was the difference statistically significant?

Let's try this again, but this time we will simulate data to be correlated. Don't worry too much about understanding the code we use to simulate the data. Just note that we are simulating the two traits together and then later separately them, thus insuring that they are correlated!

```
Q <- matrix(c(0,0.5,0.5,0,
              2,0,0,2,
```

```
              2,0,0,2,
              0,0.5,0.5,0),
            4,4,byrow=TRUE)

rownames(Q) <- colnames(Q) <- c("aa","ab","ba","bb")

diag(Q) <- -rowSums(Q)

tt <- sim.history(tree, t(Q))

tt1<-mergeMappedStates(tt,c("aa","ab"),"a")

tt1<-mergeMappedStates(tt1,c("ba","bb"),"b")

tt2<-mergeMappedStates(tt,c("aa","ba"),"a")

tt2<-mergeMappedStates(tt2,c("ab","bb"),"b")
```

Plot the data to see if they look correlated:

```
par(mfrow=c(1,2))

plotSimmap(tt1, setNames(c("blue","red"),
                         letters[1:2]), ftype="off", lwd=1)

plotSimmap(tt2,setNames(c("blue","red"),
                         letters[1:2]), ftype="off", lwd=1, direction="leftwards")
```

Exercise E

Run Pagel's test on the data we just simulated. This time, does the test indicated that the data are correlated? Which model is supported, and what is the p-value? Show your code for running the test below.

HINT: You will need to extract the states from tt1 and tt2 using `getStates(tt1,"tips")` for example.

```
#insert your code here
```

Some content in this lab is drawn from the IB200 2016 lab by Will Freyman and the Phytools blog