Lab 08: Phylogenetics in R, Continued

Carrie Tribble

2/27/2018

Integrative Biology 200

Principles of Phylogenetics

University of California, Berkeley

In this lab, we will go over how we analyze continuous trait evolution over a phylogeny. The most commonly implemented models are based on something called Brownian Motion. We will spend some time understanding Brownian Motion and explore how it relates to phylogenetic comparative methods.

Part 1: Simulating Continuous Character Evolution Under Brownian Motion

We often model the evolution of continuous characters using Brownian Motion. Here, we will model the evolution of a continuous character using Brownian Motion. Then, we will show how to use Brownian Motion to study the evolution of continuous characters.

First, we will initialize our simulations by setting the length of time for the simulation (t). This can also be thought of as the number of generations.

```
t <- 0:100
```

Next, let's initialize the instantaneous rate of change. This determines the relative size of jumps that tend to occur. When sig2 (sigma squared) is small, character state changes tend to be smaller. when its big (maximum of 1), individual state changes tend to be larger.

```
sig2 <- 0.01
```

Now, simulate a set of random deviates. In other words, this is the series of character state changes through time (t). Look at the values of x to get an idea of this.

```
x <- rnorm(n = length(t) - 1, sd = sqrt(sig2))

x
```

The character state starts at 0 then changes by the amount (x). Imagine this is the size of some trait, like leaf length. when x is negative, it means it got smaller, and when positive, it got bigger.

Now compute their cumulative sum. We want the cumulative sum because the state a character is in at any time (t) is the sum of all its past transitions. In other words, the character started at "0", got bigger by this much, smaller by that much, bigger again, etc. You add those all up to see the state at the end! If we plot all of the cumulative sums over the values of t, we can track the changes in the trait over time.

We can build up to this by starting to generate trait values over time. Each trait value at time t is calculated by adding a change in trait value given in the X vector to the trait value at t-1.

```
stepwise_values <- numeric()

stepwise_values[1] <- 0
```

```
stepwise_values[2] <- stepwise_values[1] + x[1]

stepwise_values[3] <- stepwise_values[2] + x[2]

stepwise_values[4] <- stepwise_values[3] + x[3]

stepwise_values[5] <- stepwise_values[4] + x[4]
```

Now that we've generated the trait values for the first 5 points in time, we can plot those points to see how the trait values vary as time progresses. Run these lines of code individually rather than as a whole code block.

```
abs_max <- max(abs(stepwise_values))

plot(1, stepwise_values[1],
     xlim = c(0,5), ylim = c(-abs_max,abs_max),
     xlab = "t", ylab = "trait value",
     pch = 20, type = "o")

points(c(2:5), stepwise_values[2:5], pch = 20)

lines(1:5, stepwise_values)
```

We can do this for all of our values of X using the cumsum() function.

```
x <- c(0, cumsum(x))

plot(t, x, type = "l", ylim = c(-3, 3))
```

Repeat these simulations several times by rerunning the previous code. You'll be drawing new random deviates, so the simulations should look slightly different from each other even though we haven't changed the parameters at all.

```
x <- rnorm(n = length(t) - 1, sd = sqrt(sig2))

x <- c(0, cumsum(x))

plot(t, x, type = "l", ylim = c(-2, 2))
```

Now, repeat the simulations but by changing the values of t and sig2, one at a time. Answer the following questions, and include your changed code in your answer.

Notice that this block of code does the same thing we did before but plotting several independent lineages all at once. We generate a matrix of simulations, with each simulation as a row and each unit time (t) as a column.

```
t <- 0:100

sig2 <- 0.01

nsim <- 100

X <- matrix(rnorm(n = nsim * (length(t) - 1), sd = sqrt(sig2)), nsim, length(t) - 1)

sim_matrix <- cbind(rep(0, nsim), t(apply(X, 1, cumsum))) #matrix of simulations
```

Plot the first simulation, then plot all of the simulations, with the first simulation in red.

```r
plot(t, sim_matrix[1, ], xlab = "time", ylab = "phenotype", ylim = c(-3, 3), type = "l")

apply(sim_matrix[2:nsim, ], 1, function(x, t) lines(t, x), t = t)

lines(t, sim_matrix[1, ], xlab = "time", ylab = "phenotype", ylim = c(-3, 3), col = "red")
```

Exercise A

Try this a couple of times, again changing sig2 and t. You might need to change the y axis to fit your character state range! (change the numbers in 'ylim = c(-3, 3)' to whatever you want. Include one example of your changed code in a code block below, along with the answers to the following questions:

1) How does the size of t affect how close the final character state (phenotype) is to the initial character state (which is always 0)?

2) How does the size of sig2 affect how close the final character state (phenotype) is to the initial character state?

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

How does all of this connect to Brownian motion in the context of phylogenetics? We are never given the full distribution of thousands of Brownian motion simulations. Instead, we are given a couple of trait values in the present, and are often interested in reconstruction the ancestral value - the start point of the simulation. Here's a visualization of that.

```r
plot(t, sim_matrix[1, ], xlab = "time", ylab = "phenotype",
     ylim = c(-2, 2), type = "l", col = "grey")

apply(sim_matrix[2:nsim, ], 1, function(x, t) lines(t, x, col = "grey"), t = t)

points(0,0, col = "blue", pch = 20)

text(10,.5, "Ancestral \nValue", col = "blue")

points(x = rep(100, times = 5),
       y = sim_matrix[c(1:5),100],
       col = "red", pch = 20)

text(90, mean(sim_matrix[c(1:5),100]), "Extant\nTrait\nValues", col = "red")
```

So, how does Brownian motion help us understand trait evolution? How can we calculate the ancestral value given the extant trait values? Let's go through the following questions to relate this all back to evolution.

Exercise B

Here's another simulation under Brownian motion. Add 4 additional features to the plot below:

- A horizontal line (in blue) showing the starting trait value. This is also known as the expected value for this Brownian motion simulation.
- 3 verticle lines (in red) illustrating the variance of the distribution at t = 20, t = 60, and t = 100. For example, in the plot below, I've included the variance of the distribution at t = 10. Add on the additional lines to this plot.

```
nsim <- 1000

X <- matrix(rnorm(n = nsim * (length(t) - 1), sd = sqrt(sig2)), nsim, length(t) - 1)

sim_matrix <- cbind(rep(0, nsim), t(apply(X, 1, cumsum)))

plot(t, sim_matrix[1, ], xlab = "time", ylab = "phenotype",
     ylim = c(-5, 5), type = "l", col = "grey")

apply(sim_matrix[2:nsim, ], 1, function(x, t) lines(t, x, col = "grey"), t = t)

segments(x0 = 10, y0 = min(sim_matrix[,10]), y1 = max(sim_matrix[,10]), col = "red" )
```

```
#insert your code here for the completed plot
```

Here's a graphical illustration of the relationship between t and the variance among simulations.

```
v <- apply(sim_matrix, 2, var)

plot(t, v, type = "l", xlab = "time", ylab = "variance among simulations")
```

> **In these simulations, we have discussed the relationships between time, trait values/ phenotype, and variance among simulations. For each of these 3 terms, describe how they would relate to a phylogenetic ancestral state reconstruction of a quantitative trait such as leaf size.**

There's a funtion for simulating Brownian Motion over a phylogeny in Phytools. Given a tree and your Brownian Motion parameters, you can use this function to simulate under the model. Here, we'll simulate a tree, simulate a character evolving over the tree, and plot the tree and associated character values as a 'traitgram' (Ackerly, 2009).

```
tree <- rcoal(n = 30)

x <- fastBM(tree, a=0, sig2=1.0, internal = TRUE)

phenogram(tree, x, spread.labels = TRUE)
```

Let's now simulate a character evolving under the Ornstein-Uhlenbeck process. The OU model is Brownian motion but with two extra parameters: the optimum (theta) and the strength of selection (alpha). We simulate under this model using the same fastBM model as above.

When alpha is close to 0, the OU model collapses down to Brownian motion:

```
x <- fastBM(tree, a=0, sig2=1.0, alpha=0.01, theta=4.0, internal = TRUE)

phenogram(tree, x, spread.labels = TRUE)
```

As alpha increase, we see more of a trend towards the optimum:

```
x <- fastBM(tree, a=0, sig2=1.0, alpha=0.5, theta=4.0, internal = TRUE)
```

```r
phenogram(tree, x, spread.labels = TRUE)
```

Alpha acts as a 'rubber band', pulling the trait to the optimum:

```r
x <- fastBM(tree, a=0, sig2=1.0, alpha=2.0, theta=4.0, internal = TRUE)

phenogram(tree, x, spread.labels = TRUE)
```

The above examples of Brownian motion and Ornstein-Uhlenbeck assume that the same model applies to the entire tree. It is also possible to have time or branch heterogenous models, in which the parameter values or model changes over the tree. If you are interested in these models, check out the R packages OUwie, ouch, and bayou.

Now that we have a good understanding of Brownian Motion, let's apply the model to reconstruction ancestral states for continous traits. We will use sample data on Anoles again.

```r
anole_tree<-read.tree("http://www.phytools.org/eqg2015/data/anole.tre")

svl <- read.csv("http://www.phytools.org/eqg2015/data/svl.csv",
    row.names=1)

svl <- as.matrix(svl)[,1] #convert dataframe to a vector
```

We've read in the tree and character data. Feel free to take a look at those data files to be sure you're comfortable with how they are formatted, etc. Now, we will use the fastAnc funciton in Phytools to reconstruct the ancestral states using maximum likelihood. You can read more about the specifics of the function by looking at the help page. contMap uses the same method as fastAnc but projects those values along the branches of the tree to produce the figure showing changes along branches. Again, read more in the help page for the function if you are interested.

```r
fit <- fastAnc(anole_tree,svl,vars=TRUE,CI=TRUE)

fit_obj <- contMap(anole_tree, svl, plot=FALSE)

plot(fit_obj, type="fan", legend=0.7*max(nodeHeights(anole_tree)),
    fsize=c(0.7,0.9))
```

There are several functions that reconstruct ancestral states in R. Another commonly used function is ace from the ape package. Feel free to explore these additional functions if you like.

Part 2: Correlated Evolution of Continuous Characters

How do we test if two traits are evolving in a correlated manner? Let's simulate a tree and two traits independently evolving on the tree:

```r
tree <- rcoal(n = 100)

x <- fastBM(tree)

y = fastBM(tree)
```

Are the traits correlated if we ignore the phylogeny?

```r
plot(x, y)

abline(lm(y ~ x))
```

```r
cor.test(x, y)
```

I just got p-value = 0.02679, but we know these two traits evolved independently (because we simulated them that way! You can see how easy it is to get a type I error (false positive). As an alternative, let's use phylogenetically independent contrasts (PIC; Felsenstein 1985) to estimate the evolutionary correlation between characters:

```r
x_c <- pic(x, tree)

y_c <- pic(y, tree)

cor.test(x_c, y_c)
```

Now I got a p-value = 0.4457, so we correctly reject the hypothesis that the two characters were correlated.

What happens when we simulate two correlated characters?

```r
x <- fastBM(tree)

y <- 0.4 * x + fastBM(tree, sig2 = 0.2)
```

Exercise C

> **For this most recent simulation, without "correcting" for phylogeny, are the two characters correlated? What is the p-value and correlation coefficient? Using PIC, are x and y correlated? Now what is the p-value and correlation coefficient? Show your work (code) below.**

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

Content in this lab is drawn from the IB200 2016 lab by Will Freyman, the Phytools blog, and an exercise co-written by Jenna Baughman and Carrie Tribble.