**A Brief Intro to Using Phylogenies in R**

Patrick R. Stephens

May 14, 2018

**Introduction**

This document is a short primer on working with phylogenies in the R programming environment. It covers how to import and manipulate phylogenies, how to sync up a tree with a set of species trait data, one widely used method for testing for phylogenetic signal, and some basics of tree plotting. Examples here focus on data extracted from the supertree of all mammals, first published by Bininda-Emmonds et al (2007) and modified by Fritz et al. (2009) and PanTHERIA (Jones et al. 2009).

*Cited:*

Bininda-Emonds, O. R. P., M. Cardillo, K. E. Jones, R. D. E. MacPhee, R. M. D. Beck, R. Grenyer, S. A. Price et al. 2007. The delayed rise of present-day mammals. Nature 446:507-512.

Fritz, S. A., O. R. P. Bininda Emonds, and A. Purvis. 2009. Geographical variation in predictors of mammalian extinction risk: big is bad, but only in the tropics. Ecology letters 12:538-549.

Jones, K. E., J. Bielby, M. Cardillo, S. A. Fritz, J. O'Dell, C. D. L. Orme, K. Safi et al. 2009. PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals. Ecology 90:2648-2648.

Paradis, E. 2006. Analysis of phylogenetics and evolution with R. Springer Science + Business Media: New York, New York.

**Part : 1 Basic Tree/ Trait Operations**

**Loading a tree from a file**

Tree files are generally text files that store trees in newick or nexus format, where the tree is represented as a series of parentheses and commas. For example two trees with their corresponding newick format representations (branch lengths are ignored here):



To do nearly anything with a phylogenetic tree, you first need to load the ape library.

library(ape)

**Exercise 1.** Load the ape library. Use the “Index” to look at a list of all of the commands in the ape library.

Once ape is attached, you can load trees in newick format using the read.tree command:

ftree <- read.tree("sample\_newick\_tree.txt")

Open the file you just loaded with a text editor. Note that it contains nothing but the tree itself. Nexus format is a bit more complicated than newick, and text files using it are read using the read.nexus command:

ftree2 <- read.nexus("sample\_NEXUS\_tree.txt ")

Both of these commands covert a tree from a text file to a R object of type “phylo.” Once a tree is in memory, you can plot it using the plot command:

plot(ftree2)

**Exercise 2.** Load “sample\_newick\_tree.txt” and “sample\_NEXUS\_tree.txt” into memory. Plot them both. How are they different?

**Exercise 3.** Open up the text files for the two trees using a text editor. How are they different?

You might now be wondering “Why on earth would I want to add all that extra stuff for NEXUS format, if plain old newick works fine?” In addition to storing a tree, nexus format allows you to have blocks with additional data. For example, you could have a block of species trait data, a block containing sequence data, and a tree block all in one file. In practice, it’s often easier to organize trait data separately and store trees in newick format. Regardless, many of the phylogenies that you will find online, for example the widely used the mammal supertree, are distributed in nexus format.

Next let’s take a look at the structure of the phylo objects that we created.

> str(ftree2)

List of 4

$ edge : int [1:42, 1:2] 23 24 25 26 27 27 28 28 26 25 ...

$ Nnode : int 21

$ tip.label : chr [1:22] "A\_american" "A\_fowleri" "A\_terrestr" "A\_boreas" ...

$ edge.length: num [1:42] 0.2535 0.175 0.186 0.0516 0.0106 ...

- attr(\*, "class")= chr "phylo"

- attr(\*, "order")= chr "cladewise"

Again, the tree is an object of type “phylo” (usually called for in functions using “phy”). It consists of a list of four items:

$edge: a matrix showing the position of all tips and nodes in relation to each other. Each row of the matrix represents a branch. The tips are the numbers 1-T where T is the number of tip taxa. The interior nodes are the remaining numbers, starting at T+1.

$Nnode: displays the number of interior nodes. Note that when the number of interior nodes is less than [T -1] where T is the number of tip labels, this indicates that the tree has polytomies (unresolved relationships). In this case, both trees are fully resolved

$tip.label: the names of the tips of the tree, generally species names.

$edge.length: a vector containing the lengths of all the branches, in this case in units of percentage sequence divergence.

**Tree Manipulation**

**drop.tip:** the drop.tip command can be used to remove tips from a phylogenetic tree using either the tip labels or the node numbers corresponding to tips. These two sets of commands do the same thing:

> plot(ftree)

> ftree3 <- drop.tip(ftree, 4)

> plot(ftree3)

> plot(ftree)

> ftree3 <- drop.tip(ftree, "A\_boreas")

> plot(ftree3)

Note that an underscore is part of the actual tip label, but does not appear in the plot. You can also pass a list into the drop.tip command and get rid of more than one species at once:

> plot(ftree)

> ftree3 <- drop.tip(ftree, c(21,22))

> plot(ftree3)

**Exercise 4:** Using the drop.tip command, create a tree that has no species of *Rana* in it and plot it. Do this again using the “subtree = T” switch. How is the resulting tree different?

**Exercise 5:** Drop the outgroup taxa from ftree using drop.tip as above. Export the tree to your working directory using the write.tree command. Look at the resulting file using a text editor. Was the tree saved using NEXUS or newick format?

**extract.clade**: the extract.clade command can be used to create a new tree from a subtree of another tree.

> plot(ftree)

> ftree3 <- extract.clade(ftree, 37)

> plot(ftree3)

If you aren’t sure which interior node number you should target to extract the clade that you interested in, you can use the nodelabels command to get a plot with node numbers:

> plot(ftree)

> nodelabels()

**Exercise 6**: use the extract.clade command to extract the “*Pseudacris + Hyla*” clade from ftree. Export this new tree as a NEXUS format text file.

These commands and others can be used to manipulate a tree by hand. However, in general a much more sensible way of pruning your tree is to match it up to a set of data you are interested in analyzing.

**An aside: Common data quality assumptions of phylo methods in R**

There are four common assumptions phylogenetic comparative methods in R. These aren’t generally mathematical assumptions of a statistical method (those vary widely), rather they tend to be assumptions that authors of the packages that implement comparative methods make about how your data will be organized. Many functions check some or all of these assumptions and correct for them during the process of data analysis if needed. However, over the years I have found that a good many of the odd results I get from phylogenetic analyses comes from violating one of these assumptions:

1. The data and tree match perfectly (i.e., all species in data occur in tree and vs versa, and the spelling of species labels in data matches that of the tree tip labels)

2. No missing data (complete case analysis). Missing data in trait data sets are most commonly indicated by either NA or -999.

3. The phylogenetic tree is fully dichotomous (i.e., fully resolved)

4. The order of species in data matches that of the tip labels in the tree.

Below I show one example of dealing with all of these data issues.

**Aligning a tree and a set of species trait data**

The treedata command from the geiger library makes it extremely easy to align a tree and a data vector or a data frame with row names corresponding to the tip labels of a tree.

> library(geiger)

First let’s load some sample data and a tree. The tree is from a published supertree of all mammals (Fritz et al. 2009) pruned down to members of the family Canidae.

> ctree <- read.tree("canid\_tree.txt")

> str(ctree)

List of 4

$ edge : int [1:56, 1:2] 35 36 37 38 39 39 39 38 40 41 ...

$ Nnode : int 23

$ tip.label : chr [1:34] "Canis\_lupus" "Canis\_latrans" "Canis\_simensis" "Canis\_adustus" ...

$ edge.length: num [1:56] 0.7 2.9 1.8 0.2 1.4 1.4 1.4 0.1 0.2 1.3 ...

- attr(\*, "class")= chr "phylo"

- attr(\*, "order")= chr "cladewise"

Note that the tree is not fully, resolved. You can also see this visually if you plot the tree. Some comparative methods require a fully resolved tree. We can fix this either by pruning the tree down until it is fully resolved or by using the multi2di command:

> ctree <- multi2di(ctree)

> ctree <- (ctree)

> str(ctree)

List of 4

$ edge : int [1:66, 1:2] 35 36 37 38 38 39 40 41 42 42 ...

$ Nnode : int 33

$ tip.label : chr [1:34] "Canis\_lupus" "Canis\_latrans" "Canis\_simensis" "Canis\_adustus" ...

$ edge.length: num [1:66] 0.7 2.9 0 3.4 0 0 2.5 0.1 0.8 0 ...

- attr(\*, "class")= chr "phylo"

- attr(\*, "order")= chr "cladewise"

Now we will load some data, and pull data on canid body mass. These data are a small subset of PanTHERIA (Jones et al. 2009):

> cdata <- read.csv("Canid\_traits.csv")

> str(cdata)

'data.frame': 35 obs. of 14 variables:

$ Binomial : Factor w/ 35 levels "Atelocynus\_microtis",..: 2 3 4 5 6 7 1 8 9 10 ...

$ Genus : Factor w/ 13 levels "Atelocynus","Canis",..: 2 2 2 2 2 2 1 3 4 5 ...

$ Species : Factor w/ 35 levels "adustus","alpinus",..: 1 3 17 19 22 29 23 30 6 2 ...

$ Binomial.1 : Factor w/ 35 levels "Atelocynus microtis",..: 2 3 4 5 6 7 1 8 9 10 ...

$ ActivityCycle : int 1 2 2 2 2 3 -999 1 2 2 ...

$ AdultBodyMass\_g : num 10392 9659 11989 31757 8247 ...

$ AdultHeadBodyLen\_mm: num 745 828 872 1055 707 ...

$ AgeatEyeOpening\_d : num -999 7.5 11.9 14 -999 ...

$ AgeatFirstBirth\_d : num -999 -999 365 548 -999 ...

$ BasalMetRate\_mLO2hr: num -999 -999 3699 11254 -999 ...

$ BasalMetRateMass\_g : num -999 -999 10450 33100 -999 ...

$ DietBreadth : int 6 6 1 1 6 1 1 7 6 1 ...

$ HabitatBreadth : int 1 1 1 1 1 1 1 1 1 1 ...

$ HomeRange\_km2 : num 1.01 2.95 18.88 159.86 11.93 ...

Next we will pull the body mass data and assign species names to it:

> cmass <- cdata$AdultBodyMass\_g

> names(cmass) <- cdata$Binomial

In these data, empty cells are symbolized by -999

> hist(cmass)

> range(cmass)

[1] -999.00 31756.51

We want to prune the data down to species for which body mass data are available:

> cmass <- cmass[cmass > 0]

Finally, many methods assume trait data that are roughly normally distributed among species. This is obviously not true of our data, they appear to be lognormal. We will transform the data accordingly:

> hist(cmass)

> hist(log10(cmass))

> cmass <- log10(cmass)

Now we are ready to align the data and the tree. The treedata command assumes that the input is a phylo object and data with names that correspond to the tip labels of the tree. In the case of a vector, the elements of the vector should be named. In the case of a data-frame, the names should be row names. The treedata command will prune species from the phylogeny not found in the data, and eliminate data for species not found in the tree:

> CMout <- treedata(ctree, cmass)

Warning message:

In treedata(ctree, cmass) :

The following tips were not found in 'data' and were dropped from 'phy':

Lycalopex\_fulvipes

Lycalopex\_griseus

In this case, two species found in the tree were not present in the data. These are the two species that we eliminated earlier because no body mass estimates were available for them.

> str(CMout)

List of 2

$ phy :List of 4

..$ edge : int [1:62, 1:2] 33 34 35 36 36 37 38 39 40 40 ...

..$ Nnode : int 31

..$ tip.label : chr [1:32] "Canis\_lupus" "Canis\_latrans" "Canis\_simensis" "Canis\_adustus" ...

..$ edge.length: num [1:62] 0.7 2.9 0 3.4 0 0 2.5 0.1 0.8 0 ...

..- attr(\*, "class")= chr "phylo"

..- attr(\*, "order")= chr "cladewise"

$ data: num [1:32, 1] 4.02 3.98 4.08 4.5 3.92 ...

..- attr(\*, "dimnames")=List of 2

.. ..$ : chr [1:32] "Canis\_adustus" "Canis\_aureus" "Canis\_latrans" "Canis\_lupus" ...

.. ..$ : NULL

> plot(CMout[[1]])

> hist(CMout[[2]])

We can now extract our pruned tree . . .

> CMtree <- CMout[[1]]

. . .and data. Note that by default we get a dataframe with one variable:

> CMass2 <- CMout[[2]]

> head(CMass2)

[,1]

Canis\_adustus 4.016720

Canis\_aureus 3.984919

Canis\_latrans 4.078787

Canis\_lupus 4.501833

Canis\_mesomelas 3.916312

Canis\_simensis 4.157211

This is easily remedied:

> CMass2 <- CMout[[2]][,1]

> head(CMass2)

Canis\_adustus Canis\_aureus Canis\_latrans Canis\_lupus

4.016720 3.984919 4.078787 4.501833

Canis\_mesomelas Canis\_simensis

3.916312 4.157211

Finally, not that the tip labels of the phylogeny are in a different order than the species in the vector or rows in a dataframe:

|  |
| --- |
| > str(CMtree)  List of 4  $ edge : int [1:62, 1:2] 33 34 35 36 36 37 38 39 40 40 ...  $ Nnode : int 31  $ tip.label : chr [1:32] "Canis\_lupus" "Canis\_latrans" "Canis\_simensis" "Canis\_adustus" ...  $ edge.length: num [1:62] 0.7 2.9 0 3.4 0 0 2.5 0.1 0.8 0 ...  - attr(\*, "class")= chr "phylo"  - attr(\*, "order")= chr "cladewise" |

Some methods assume that the order of data in a vector matches the order of the tip labels in a tree that is used with the data. Ancestral character reconstruction in particular, which we will come to shortly, will give you very strange results if you don’t make one final correction:

> CMass2 <- CMass2[CMtree$tip.label]

> head(CMass2)

Canis\_lupus Canis\_latrans Canis\_simensis Canis\_adustus

4.501833 4.078787 4.157211 4.016720

Canis\_aureus Canis\_mesomelas

3.984919 3.916312

**Exercise 7:** Using the data and tree that we gave you, create a vector of species that have data for “age at eye opening,” and create a tree that matches it. Fix all four of the data issues I mentioned above.

**Testing for Phylogenetic Signal Using Blomberg’s K**

Most phylogenetic comparative methods assume that the traits used in your analysis show phylogenetic signal. For continuous traits *phylogenetic signal* can be defined as a pattern where disparity in trait values among species scales with phylogenetic distance, in other words that more closely related species tend to have trait values that differ less than those of more distantly related species. Using a phylogenetically informed statistical method such as independent contrasts with data that lack phylogenetic signal is at best unnecessary, and can in some cases inflate rates of type II error. For this reason, the assumption that either your data or residuals of a model based on your data show statistically significant phylogenetic signal should generally be tested before choosing to use phylogenetic comparative methods (as opposed to more conventional statistics) for hypothesis testing.

The two most widely accepted measures of phylogenetic signal are Blomberg’s K (Blomberg at al. 2002) and Pagel’s lambda (Pagel 1999). Testing for statistically significant phylogenetic signal using Blomberg’s K is quite straightforward using the phylosignal command from the picante library.

> library(picante)

> phylosignal(CMass2, CMtree)

K PIC.variance.obs PIC.variance.rnd.mean PIC.variance.P

1 1.133742 0.02153078 0.06677143 0.001

PIC.variance.Z

1 -3.689463

This method implements a randomization test described in Blomberg and Ives (2003), and reports the value of K. A K value of 1 indicates that a trait shows the amount of phylogenetic signal that would be expected under a Brownian motion model of character evolution, a value of zero indicates a complete lack of phylogenetic signal. Values of K between 0 and 1 indicate lower signal than expected under Brownian motion and are generally interpreted as “partial phylogenetic dependence.” A value of K greater than one indicates stronger phylogenetic signal than expected under Brownian motion. Blomberg’s K has been shown to be a robust measure of phylogenetic “effect size” sensu Nakagawa and Cuthill (2007, discussed in Münkemüller et al. 2012). Blomberg’s K is most often used as a standalone measure of phylogenetic signal and/ or to test for statistically significant signal. Pagel’s lambda can also be used this way, but is more commonly used as a signal parameter in statistical models.

*Cited:*

Blomberg, S. P., and T. Garland, Jr. 2002. Tempo and mode in evolution: phylogenetic inertia, adaptation and comparative methods. Journal of Evolutionary Biology 15:899-910.

Blomberg, S. P., T. Garland, Jr., and A. R. Ives. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. Evolution 57:717-745.

Nakagawa, S., & Cuthill, I. C. (2007). Effect size, confidence interval and statistical significance: a practical guide for biologists. *Biological reviews*, *82*(4), 591-605.

Münkemüller, T., Lavergne, S., Bzeznik, B., Dray, S., Jombart, T., Schiffers, K., & Thuiller, W. (2012). How to measure and test phylogenetic signal. *Methods in Ecology and Evolution*, *3*(4), 743-756.

Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature 401:877–884.

**Exercise 8:** Use phylosignal to determine whether age at eye opening exhibits statistically significant phylogenetic signal. Does it show stronger or weaker signal than body mass? How does the value of K change depending upon whether you use raw trait data or normalized (in this case logged is close enough) data?

**Exercise 9 (Optional):** Use phylosignal to test for phylogenetic signal on random trait values drawn from a normal distribution. What range of K values seems typical when you do this? How often is the signal of these random values “statistically significant?”

**Part 2: Basic Tree Plotting and Ancestral Character Reconstruction**

We have already seen how to create a basic plot of a tree using the plot function in ape. When ape is loaded and used with a phylo object, plot calls the plot.phylo functions. There are many options for tree plotting.

**Exercise 10:** Plot the tree that you created in exercise 7 as a radial tree, a phylogram, and cladogram (use the documentation for plot.phylo to figure out how to do this).

**Adding a scale bar**

For a figure of a phylogram that is not ultrametric, particularly one with branch lengths in units of percentage sequence divergence, a small scale bar is commonly added. The add.scale.bar command adds a scale bar to an existing plot of a phylogeny in R.

> plot(ftree)

> add.scale.bar()

For figures of ultrametric trees, and chronograms in particular, a scale bar along the bottom of the figure that shows the total divergence accumulated at different depths in the phylogeny is common. The axisPhylo command allows this element to easily be added to a tree plot in R.

> plot(ftree)

> axisPhylo()

**Exercise 11:** Add a scale bar to a plot of the tree that you generated in exercise 7. Is a scale running along the entire X-axis of the plot or a scale bar embedded within the plot more informative?

**Visualizing Continuous Character Evolution**

This section describes how to perform an ancestral character reconstruction using maximum likelihood, and one way to plot continuous traits on a tree. In this example, we have a tree with the same species found in a vector of trait data, and where the species in the trait vector occur in the same order as the tip labels of the tree. Maximum likelihood reconstructs continuous trait values, here we will bin traits and their reconstructed values for visualization purposes.

First create a color pallet for the trait bins:

> mypallete <- c("red","orange","yellow","green","blue","purple")

Next, create indices that will allow us to bin the traits an their reconstructed values:

> divider <- (max(CMass2+0.0001)-min(CMass2))/6

> index <- floor((CMass2-min(CMass2))/divider)

Next we will use the ace command from ape to perform an ancestral character reconstruction of body mass:

> CMtree$edge.length <- CMtree$edge.length+0.001

> recon <- ace(CMass2, CMtree, CI = F)

> recon.values <- recon$ace

> recon.index <- floor((recon.values-min(CMass2))/divider)

Note that we added a small amount to all of the branch lengths. This was to eliminate zero length branches, which will cause an error with the ace function. To obtain accurate estimates of ancestral character states, it’s essential to use a tree that is fully resolved and has accurate branch length information, as well as to carefully evaluate the model of character evolution used. However, if we are using character reconstruction merely as a method to visualize phylogenetic patterns of character variation (as in this example) these steps aren’t strictly necessary.

Now we are ready to create a plot. Remember that to use the indices we created to call colors from the pallet you need to add 1 to the bin score when plotting (see below), since there is no color in position 0. First plot the tree, adjusting the label.offset so that there is enough room for the trait data at the tips:

> plot(CMtree, label.offset = 0.1, edge.width = 2)

Next add in the tip values and reconstructed interior node values:

> nodelabels(pch = 21, cex = 2, bg = mypallete[recon.index+1])

> tiplabels(pch = 21, cex = 2, bg = mypallete[index+1])

Finally, add a legend to the figure:

> pts = c(0,1,2,3,4,5)

> legend("topright", pch = 21, pt.cex = 2, c("0.00-0.25","0.25-0.50","0.50-0.75","0.75-1.00","1.00-1.26","1.26-1.51"), title = "Log 10 Mass", pt.bg =

mypallete[pts+1])

You will often need to fiddle with the pt.bg, cex and other settings such as label.offset quite a bit to get a nice looking plot.

**Exercise 12:** Perform an ancestral character reconstruction and plot any of the continuous traits (besides mass) in the data set that we gave you.

**Visualizing Discrete Character Evolution**

Here we will use the ace function to reconstruct a discrete trait, activity cycle, using maximum likelihood, and create a plot summarizing trait variation and the reconstructed probabilities of the character states of interior nodes. First we need to get the trait data and align it with our tree:

#pulling the trait

> AC <- cdata$ActivityCycle

> names(AC) <- cdata$Binomial

> AC <- AC[AC > 0]

#aligning the tree and the data

> out2 <- treedata(ctree, AC)

Warning message:

In treedata(ctree, AC) :

The following tips were not found in 'data' and were dropped from 'phy':

Atelocynus\_microtis

Lycalopex\_fulvipes

Lycalopex\_griseus

Lycalopex\_vetulus

Vulpes\_ferrilata

Vulpes\_macrotis

#cleaning up the tree

> CAtree <- out2[[1]]

> CAtree <- multi2di(CAtree)

> CAtree$edge.length <- CAtree$edge.length+0.001

#cleaning up the trait data

> CAC <- out2[[2]][,1]

> CAC <- CAC[CAtree$tip.label]

Now we can perform the character reconstruction. In this example, we assume equal transition rates among character states. Just as with a continuous character, carefully evaluating the accuracy your tree topology, branch lengths, and model of character evolution is critical if you want to get accurate estimates of ancestral character states. Here we are only using character reconstruction as a way to visually summarize variation among clades.

> anC <- ace(CAC, CAtree, type = "d")

> states <- anC$lik.anc

Finally, we are ready to plot the tree. Note the second line of code where we create a custom color palette named “co.”

> plo t(CAtree, edge.width = 2, label.offset = 1)

> co <- c("white", "gray", "black")

> tiplabels(pch = 22, bg = co[as.numeric(CAC)], cex = 2, adj = 1)

> nodelabels(pie = states, piecol = c("white", "gray", "black"), cex = 0.5)

> axisPhylo()

Finally we will add a legend:

> pts = c(1,2,3)

> legend("topright", pch = 22, pt.cex = 2, c("nocturnal","crepuscular","diurnal"), title = "Activity Cycle", pt.bg = co[pts])

Again, you will often have to spend a bit of time fiddling with the graphics parameters to get a plot like this to be legible.

**Exercise 13:** Perform an ancestral character reconstruction and plot of diet breadth using the data set and tree that we gave you.