BIOL 416 R Tutorial 2 - Phylogenies in R

Mairin Deith

Table of Contents

[Preface 1](#_Toc533002981)

[Tutorial 2 1](#_Toc533002982)

[Review of last week’s tutorial (introduction to R) 1](#_Toc533002983)

[Introduction to phylogenies 2](#_Toc533002984)

[Introduction to packages 3](#_Toc533002985)

[The picante library 3](#_Toc533002986)

[Data types in picante: the phylo class 5](#_Toc533002987)

[Data types in picante: community objects 12](#_Toc533002988)

[Data types in picante: trait objects (consider removing this section) 13](#_Toc533002989)

[Putting it all together - calculating phylogenetic diversity, PD 14](#_Toc533002990)

[Summary 17](#_Toc533002991)

# Preface

These tutorials were developed to teach high-level undergraduate students the fundamentals of working with phylogenetic, community, and trait data in R and R Studio using the picante package.

These instructions were modified from Dr. Steven Kembel’s [“An introduction to the picante package” walkthrough](http://picante.r-forge.r-project.org/picante-intro.pdf) to reflect the learning objectives of BIOL 416 at the University of British Columbia.

# Tutorial 2

January 21 & 23, 2019

## Review of last week’s tutorial (introduction to R)

1. Objects - 0, 1, and 2-D objects
2. Indexing objects - using [] and name-based $ notation
3. Plotting
4. Linear regression and models

**Do this with a line-by-line, something like what’s written below; ask-the-class what different lines of code will do**.

is.human <- TRUE  
  
is.human == TRUE  
is.human == 10  
  
read.csv("  
 ")  
  
data(iris)  
s.l <- iris$Sepal.Length  
s.l.deviation <- iris$Sepal.Length/mean(iris$Sepal.Length)  
  
plot(iris$Sepal.Length ~ iris$Sepal.Width)  
  
lm(iris$Sepal.Length ~ iris$Sepal.Width)

## Introduction to phylogenies

In this course, we’ll be working with phylogenetic trees and evolutionary relationships between species. Evolution and the relatedness between species is increasingly used in conservation biology to identify the evolutionary consequences of extinction and identify whether closely related species share traits that make them more or less susceptible to extinction.

**Phylogenetic tree** = a representation of species’ inter-relatedness; which species share ancestors and when their lineages diverged

*Draw a phylogeny on the board*.

Ultimately, we are interested in a few key pieces of information from the phylogeny (*circle these on the board when describing*):

* *Node* : a point where a lineage splits into multiple descendant lineages
* *Edge/branch* : lines to connect nodes in the tree
* *Tip* : terminal edges/branches that represent currently living lineages
* *Root* : nodes that are the root of all edges/branches

Within the tree, we can represent the branching of lineages as a function of time or mutations.

*Note to self: the common names of the species are:*

* *Strix aluco* - tawny owl
* *Asio otus* - long-eared owl
* *Athene nocturna* - little owl
* *Tyto alba* - barn owl

## Introduction to packages

We are now going to dive in to using R to work with phylogenies.

Packages are pre-arranged sets of data, functions, and other R programming components. Often, when we want to do something in R like perform a statistical test or make a figure, someone has done this before. If they have published their code online in a package, you can borrow the functions they have written in order to do your own work.

*Demonstrate/show off the* [*CRAN task view website*](https://cran.r-project.org/web/views/)*, with specific focus* [*on the Phylogenetics task view*](https://cran.r-project.org/web/views/Phylogenetics.html)*.*

Any time you need to use a specialized set of functions, you can probably find a relevant library on CRAN or through another software source like Git Hub. We are going to dive into a specific library that you’ll use throughout the rest of this course called picante.

## The picante library

picante provides functions for phylogenetic and community analysis. You’ll be using phylogenetic analysis later in the semester for your upcoming assignments.

The first step to using a package is installation. We’ll install the picante package from an online source - for R, most of the time you will be installing packages from CRAN, the Comprehensive R Archive Network. *Again open up the CRAN task view to show the available packages*.

We can install packages in R using the install.packages() command. After the package has been installed, we load it into our R session using the library() command.

# Dependencies are other packages that can be borrowed from  
### install.packages("picante", dependencies = TRUE)  
library(picante)

## Loading required package: ape

## Warning: package 'ape' was built under R version 3.2.5

## Loading required package: vegan

## Warning: package 'vegan' was built under R version 3.2.5

## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.4-0

## Loading required package: nlme

*This might take a few minutes, check around the room to make sure everyone has installed and loaded the library properly.*

Just like with functions, we can use the help() command to find out more information about our newly loaded library. Let’s take a closer look at picante and the functions it adds to our library:

help(package=picante)

This shows us individual help pages for each function that picante provides - you can see there are a lot of available functions that come with picante.

Inside of the help panel, we can also look at something called a vignette - a vignette is a friendly introductory tutorial that is designed to give you an easy entry point to using the package. We’ll be working together through some of the materials presented in picante’s vignette, in fact.

Finally, some packages come with data sets just like the iris data set we looked at last week. picante comes with a data set called phylocom - it includes phylogenetic, community, and trait data.

You can load this with

data("phylocom")

and inspect the contents with

help(phylocom)  
# or  
summary(phylocom)

## Length Class Mode   
## phylo 5 phylo list   
## sample 150 -none- numeric  
## traits 4 data.frame list

Notice that phylocom has three named components that we can access one-at-a-time using the $ notation: phylo, sample, and traits. These are the phylogenetic tree, a data frame of which species are found where, and the traits that those species have.

For example:

# The phylogenetic tree:  
phylocom$phylo  
  
# The community data:  
phylocom$sample  
  
# And finally, trait data:  
phylocom$traits

Let’s talk about what these data types are, because you’ll be using them in your assignments throughout the rest of this course.

### Data types in picante: the phylo class

In picante and other phylogenetic libraries in R, there is a special class of data called the phylo class. This data type represents phylogenetic trees in a way that the computer and R can understand and manipulate.

Ignoring the phylocom data set that is pre-built into picante, we’re going to start with a simple example phylogeny that has four species of owl. Let’s create a phylo class object using the read.tree() command that was added to our library when we installed picante.

We’re not going to build the tree from scratch, rather we’re going to read in a pre-saved phylogeny from the owls.tre Newick file, indicated by the .tre file type.

# Provide this .tre file to the class  
### s <- "owls(((Strix\_aluco:4.2,Asio\_otus:4.2):3.1,Athene\_noctua:7.3):6.3,Tyto\_alba:13.5);"  
### cat(s, file = "owls.tre", sep = "\n")  
owl.tree <- read.tree("owls.tre")  
  
# To make things easier, you can use this command too:  
### owl.tree <- read.tree(file.choose())

Take a look at the tree:

owl.tree

##   
## Phylogenetic tree with 4 tips and 3 internal nodes.  
##   
## Tip labels:  
## [1] "Strix\_aluco" "Asio\_otus" "Athene\_noctua" "Tyto\_alba"   
##   
## Rooted; includes branch lengths.

R tells us there are four tips and three internal nodes (*highlight these on the phylogenetic tree on the board*).

We can take a closer look using the str[ucture] command like this:

str(owl.tree)

## List of 4  
## $ edge : int [1:6, 1:2] 5 6 7 7 6 5 6 7 1 2 ...  
## $ Nnode : int 3  
## $ tip.label : chr [1:4] "Strix\_aluco" "Asio\_otus" "Athene\_noctua" "Tyto\_alba"  
## $ edge.length: num [1:6] 6.3 3.1 4.2 4.2 7.3 13.5  
## - attr(\*, "class")= chr "phylo"  
## - attr(\*, "order")= chr "cladewise"

#### Components of a phylo object

There are four named components in our phylogenetic tree:

1. edge: a 2-dimensional matrix where each row is an edge of the tree

owl.tree$edge

## [,1] [,2]  
## [1,] 5 6  
## [2,] 6 7  
## [3,] 7 1  
## [4,] 7 2  
## [5,] 6 3  
## [6,] 5 4

This tells us that there are six edges in total in our phylogenetic tree. The first edge, [ 5 6 ] is the branch that runs from our root, node #5, to node #6, the splitting point.

*Draw this on the phylogenetic tree on the board edge-by-edge.*

1. Nnode: 3, the number of internal nodes (does not include the tips)
2. tip.label: a vector of names that are used on the tips of the tree

owl.tree$tip.label

## [1] "Strix\_aluco" "Asio\_otus" "Athene\_noctua" "Tyto\_alba"

1. edge.length: a vector of numbers that correspond to the lengths of the edges in $edge

owl.tree$edge.length

## [1] 6.3 3.1 4.2 4.2 7.3 13.5

These are the components that go into a phylo object.

#### What the phylo object looks like underneath

Let’s take a peek at the original file we read into R as a phylo object. We can do this by opening the text file directly in R.

owls(((Strix\_aluco:4.2,Asio\_otus:4.2):3.1,Athene\_noctua:7.3):6.3,Tyto\_alba:13.5);

owls = the name of the tree

The colon : separates the tip labels from the edge lengths, parentheses show us which species should be connected by that branch length.

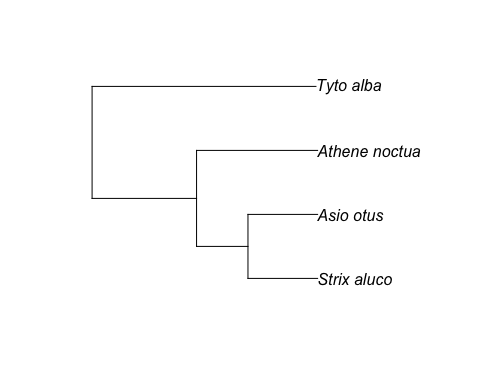
Notice that the tips are shown from bottom to top - we can compare all of this to the results we saw when we called $edge and $edge.lengths in the phylo object.

Everything we see in the phylogenetic tree comes from a relatively simple file.

#### Plotting a phylogenetic tree

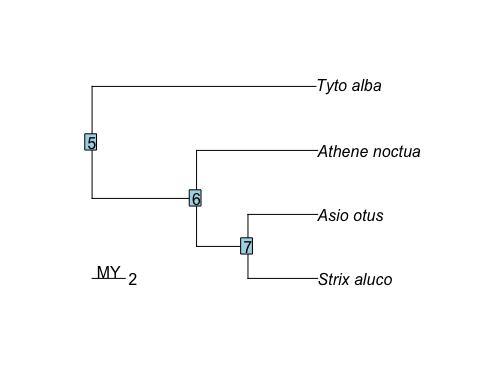
Plotting a phylo object is straightforward:

plot(owl.tree)



We can add modifications like adding tip labels

plot(owl.tree)  
# Adding a horizontal scale bar:  
add.scale.bar()  
# Adding a text label to show the units for the scale bar  
text(1,1.1,"MY")  
# Adding labels to the intermediate nodes  
nodelabels()



#### Modifying phylo objects

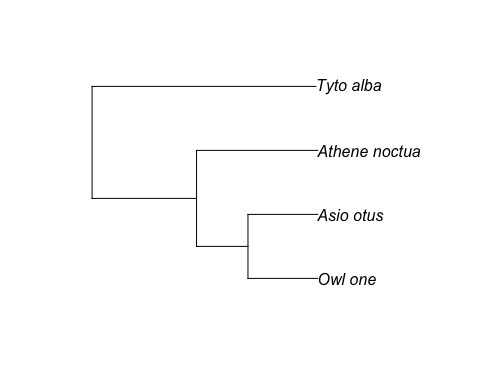
##### Changing tip names

There are two ways to modify the tip names in the phylogenetic tree: you can directly edit the Newick text document or you can modify the names in R itself.

Be careful about how you write these names - R, like many programming languages, cannot handle spaces in names and will have issues if you provide spaces.

When changing the Newick file directly, let’s change the name of *Strix aluco* to Owl One.

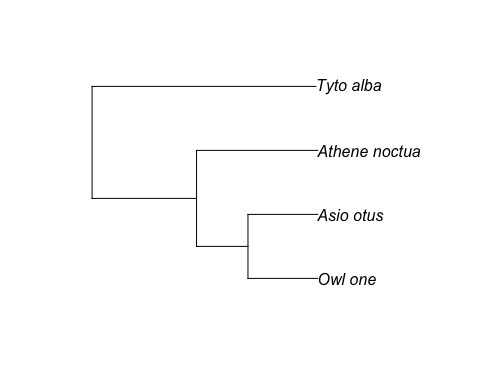
s <- "owls(((Owl\_one:4.2,Asio\_otus:4.2):3.1,Athene\_noctua:7.3):6.3,Tyto\_alba:13.5);"  
cat(s, file = "owls2.tre", sep = "\n")  
owl.tree.2 <- read.tree("owls2.tre")  
  
plot(owl.tree.2)

 We can also change the name using the $tip.label component of the phylogenetic object.

owl.tree.3 <- owl.tree  
owl.tree.3$tip.label

## [1] "Strix\_aluco" "Asio\_otus" "Athene\_noctua" "Tyto\_alba"

owl.tree.3$tip.label[1] <- "Owl\_one"  
  
plot(owl.tree.3)

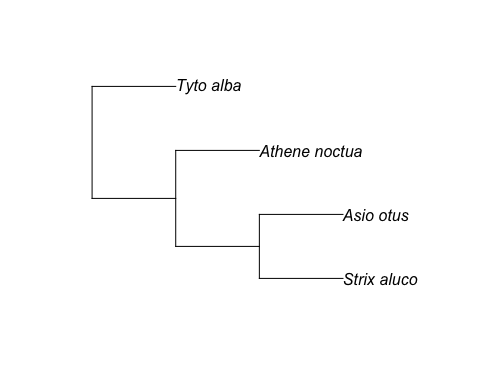


##### Change edge lengths

Similar to how we changed the tip labels for the phylogenetic tree, we can also change the edge lengths of the branches in the tree. Let’s try this by changing the edge lengths equal to 1 million years.

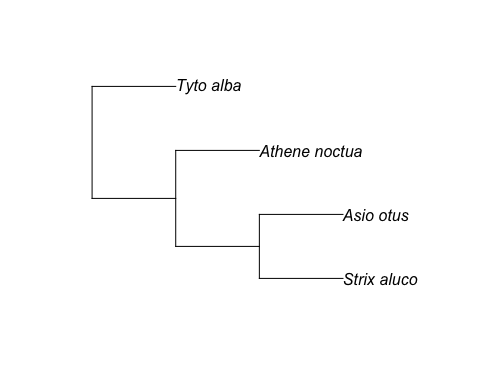
To do this in the Newick file:

s <- "owls(((Strix\_aluco:1,Asio\_otus:1):1,Athene\_noctua:1):1,Tyto\_alba:1);"  
cat(s, file = "owls\_branch1.tre", sep = "\n")  
owl.tree.4 <- read.tree("owls\_branch1.tre")  
  
plot(owl.tree.4)

 Notice that the edge length leading to *Tyto alba* is now much shorter than it used to be - this is because the phylogenetic tree’s branches are all the same size. So if we know that Owl one and *Asio otus* are 1MY apart, and work our way back to the root node, the phylogenetic tree believes that *Tyto alba*’s lineage disappeared only one million years after the root node.

Of course, we can also make this change in R using $edge.length

owl.tree.4 <- owl.tree  
# How many edges are there?  
n.edges <- nrow(owl.tree.4$edge)  
owl.tree.4$edge.length <- rep(1, n.edges)  
  
plot(owl.tree.4)



### Data types in picante: community objects

Often when we work with phylogenetic data, we’re interested in comparing the phylogenetic history of the species observed in a species community. The picante package also uses community data, records of observations of a species at different sites.

Community data is represented as a 2D object (data-frame/matrix) where sites/samples are placed in the rows and taxa in the columns.

So let’s make a new data-frame for our owls at three imaginary sites: SiteA, B, and C:

*On the board, write the following P/A matrix:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Strix\_aluco | Asio\_otus | Athene\_noctua | Tyto\_alba |
| SiteA | 0 | 1 | 1 | 0 |
| SiteB | 0 | 0 | 1 | 1 |
| SiteC | 1 | 0 | 1 | 1 |

To make this data frame in R, we can make a new data frame from scratch and tell R what to put in each column:

owl.comm <- data.frame(Strix\_aluco = c(0,0,1),  
 Asio\_otus = c(1,0,0),  
 Athene\_noctua = c(1,1,1),  
 Tyto\_alba = c(0,1,1)  
)  
  
rownames(owl.comm) <- c("SiteA","SiteB","SiteC")  
owl.comm

## Strix\_aluco Asio\_otus Athene\_noctua Tyto\_alba  
## SiteA 0 1 1 0  
## SiteB 0 0 1 1  
## SiteC 1 0 1 1

Now we have a community data set, showing which species were present at each site.

What picante can do now is match the column names to the tip labels in our phylogenetic tree. **Note that the column names MUST MATCH the tip labels in the phylo object!**

While column names are used to match the species in each community to that species’ position in the phylogeny, row names can be used to give informative site names.

### Data types in picante: trait objects (consider removing this section)

The last type of data that picante can use is trait data. Each species has some quantifiable traits that differ them from other species. For example, our owls might have different body sizes, prey preferences, or timing of behaviours.

We can represent these traits in a data frame like the community data frame - unlike the community data-set however, we have **species in rows and traits as columns**.

Let’s say we’re only interested in body mass and wingspan, and build a trait data-frame from this:

# Data from Wikipedia averages  
owl.traits <- data.frame(  
 row.names = owl.tree$tip.label,  
 body.mass.g = c(625,300,180,450),  
 wing.span.cm = c(92,100,56,87)  
)  
owl.traits

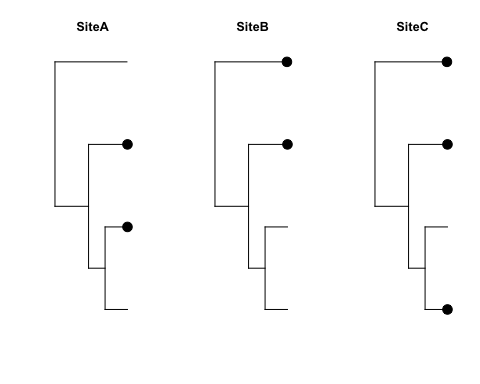
## body.mass.g wing.span.cm  
## Strix\_aluco 625 92  
## Asio\_otus 300 100  
## Athene\_noctua 180 56  
## Tyto\_alba 450 87

### Putting it all together - calculating phylogenetic diversity, PD

Now we have three data objects - a phylogenetic tree for four owl species, a community data-frame showing if they were in sites A, B, and C, and a trait data-set of body mass and wing span.

*Note: show this plot but do not make them graph it out - it includes complex indexing with which and the %in% function, which is a little advanced for these folks.*

pruned.owl <- prune.sample(owl.comm, owl.tree)  
  
par(mfrow = c(1, 3))  
for (i in row.names(owl.comm)) {  
 plot(pruned.owl, show.tip.label = FALSE, main = i)  
 tiplabels(tip = which(pruned.owl$tip.label %in% colnames(owl.comm)[which(owl.comm[i, ] > 0)]),  
 pch = 19, cex = 2)  
}



The next thing we are going to do is calculated phylogenetic diversity, also called PD. Phylogenetic diversity is a measure of how much phylogenetic time is captured by an observed species assemblage. It captures the total **edge length** of a phylogenetic tree that is represented by a community.

*On the phylogenetic tree on the board, show how PD is calculated by adding together the edge lengths between species that are present, assuming that each edge length is only 1MY*.

Of the three sites shown above, which do you think will have the greatest phylogenetic diversity? **Site C**.

Basically, when we calculate phylogenetic diversity, we are calculating how much evolutionary time is represented in that community, or the phylogenetic distinctiveness of its members. If all species are very closely related to one another, PD is low; conversely, if there is a lot of evolutionary differentiation between species present in a community, PD is high.

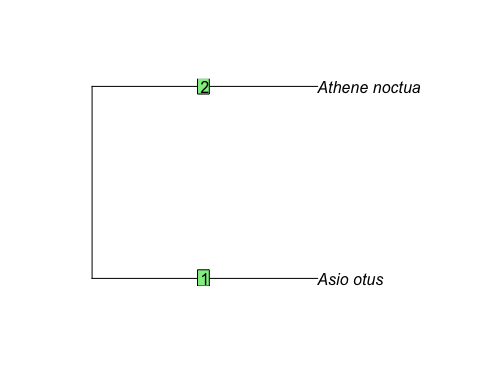
We’re now going to calculate the phylogenetic diversity of our owl community in two ways: first by hand, then using a pre-built function from picante.

Steps:

1. Prune the phylogenetic tree for each community so that it only spans the observed species
2. Decide whether to include the root node or create a new root node for the community
3. Sum together the branch lengths to calculate PD

#### Calculating PD by hand

siteA.absent <- colnames(owl.comm)[  
 which(owl.comm["SiteA",]==0) # Take the time to explain this step!!!!  
 ]  
owl.tree.siteA <- drop.tip(owl.tree, siteA.absent)  
siteA.pd <- sum(owl.tree.siteA$edge.length)  
  
plot(owl.tree.siteA)  
edgelabels()



### Show the above on the screen, then have students write it out on their own for sites B and C  
siteB.absent <- colnames(owl.comm)[  
 which(owl.comm["SiteB",]==0) # Take the time to explain this step!!!!  
 ]  
owl.tree.siteB <- drop.tip(owl.tree, siteB.absent)  
owl.tree.siteB

##   
## Phylogenetic tree with 2 tips and 1 internal nodes.  
##   
## Tip labels:  
## [1] "Athene\_noctua" "Tyto\_alba"   
##   
## Rooted; includes branch lengths.

siteB.pd <- sum(owl.tree.siteB$edge.length)  
  
siteC.absent <- colnames(owl.comm)[  
 which(owl.comm["SiteC",]==0) # Take the time to explain this step!!!!  
 ]  
owl.tree.siteC <- drop.tip(owl.tree, siteC.absent)  
siteC.pd <- sum(owl.tree.siteC$edge.length)  
  
paste0("A:", siteA.pd, " B:", siteB.pd, " C:", siteC.pd)

## [1] "A:14.6 B:27.1 C:34.4"

Site A, where the only species present are “clumped” phylogenetically, has the lowest phylogenetic diversity while Site C has the highest. #### Calculating PD with the built-in picante function

Even though calculating PD by hand is relatively easy, it can be tedious going site-by-site. Luckily, we can use picante’s built-in pd() function to calculate PD for each site automatically.

all.sites.pd <- pd(samp = owl.comm, tree = owl.tree, include.root = FALSE)  
all.sites.pd

## PD SR  
## SiteA 14.6 2  
## SiteB 27.1 2  
## SiteC 34.4 3

Using the built-in function, we get the same answers as we did through hand calculations.

Notice that the function gives us a data frame as output - we can now save this data.frame or use the values within for more calculations.

## Summary

*Give the class another example phylogenetic tree and corresponding .csv file that they can use to fulfil the following instructions, or get them to repeat the analysis with the phylocom dataset in picante*:

1. Plot a complete phylogenetic tree
2. Calculate the total phylogenetic diversity in the entire phylogeny
3. Change the tip labels from scientific names to something else…common names?
4. Change the edge length from MY to thousands of years (\*1000)
5. Calculate PD according to a community .csv file
   1. First by hand
   2. Second using the pd() function