Introduction to phylogenies in R

Simon Joly

BIO 6008 - Fall 2015

Contents

Introduction

This document represents an introduction to using phylogenies in R. It asssumes that you already know the basics of R. If you don't, you should start by looking at a R tutorial on the web.

The idea with this tutorial is to guide you through different functions in order to give you an idea of what can be done with phylogenies in R. You will of course learn much more tricks in the following classes.

R ressources

R Studio

For this class, you need to install R on your computer. You will also need to install R studio, which is a very practical way to use R. But more importantly, it allows to use KnitR very easily, which you will need for your assignments.

Packages

There are lots of packages for phylogenetic analyses in R. I won't enumerate them all here, but you can have a good idea of the options available by looking at the phylogenetic R vignette maintainned by Brian O'Meara. It is mostly oriented towards phylogenetic comparative methods, but it is a good start.

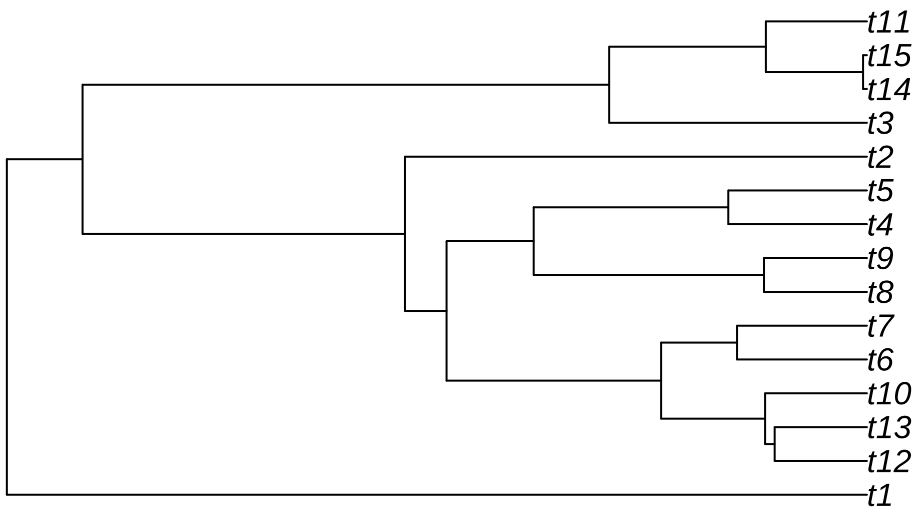
The most basic package for using trees in R is *ape*, which allows you to read and plot trees.

Importing and plotting trees

Simulate a tree

Throughout these exercises, we will often use simulated trees, which are very useful for pedagogical purposes. Trees can be simulated using several functions, but here is an example to simulate one tree with 15 species.

```
require(phytools)
tree <- pbtree(n=15,nsim=1)
plot(tree)
```



You save the tree in nexus format to a file. But before you do so, it is a good idea to set the working directory to the same folder where your script is saved. You can do that in RStudio in the menu Session>Set Working Directory>To Source File Location.

```
require(ape)
write.nexus(tree, file="My_first_tree.tre")
```

Simulating characters

Characters can also be easily simulated in R. For instance, you could simulate a character using a Brownian Motion (BM) model using the following code.

```
trait1 <- fastBM(tree, sig2=0.01, nsim=1, internal=FALSE)
# To get trait values for tree tips:
trait1
```

```
##           t1          t12          t13          t10           t6
##  0.131512395  0.153825786  0.235751638  0.272334931  0.028465290
##           t7           t8           t9           t4           t5
## -0.074704313  0.159132019  0.005590557  0.339806388  0.281136359
##           t2           t3          t14          t15          t11
##  0.080255045 -0.423295826 -0.593806290 -0.579557673 -0.423688043
```

Now, let's save this trait to a file to pretend it is our original data.

```
write.table(matrix(trait1,ncol=1,dimnames=list(names(trait1),"trait1")), file="mytrait.csv", sep=";")
```

Now that we have simulated a tree and a character, let's erase what we have done so far from the R environment and pretend these are our data for the next sections.

```
rm(list=ls())
```

Import data into R

Here is how you should import your data into R.

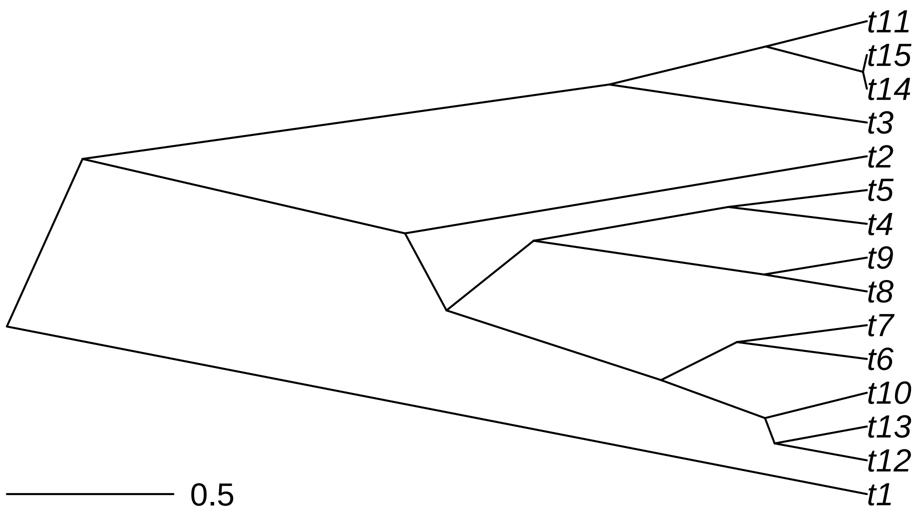
```
tree <- read.nexus(file="My_first_tree.tre")
trait1 <- read.csv2(file="mytrait.csv",dec=".")
```

The tree format in ape contains several information and it is useful to know how to access them. For instance, the tip labels can be accessed using `tree$tip.label` and the branch lengths using `tree$edge.length`. Will will see more options in other exercises, but if you want more detailed information on how the objects "phylo" are organized, you can have a look the help file `?read.tree` or at this document prepared by Emmanuel Paradis, the author of `ape`.

Plot trees

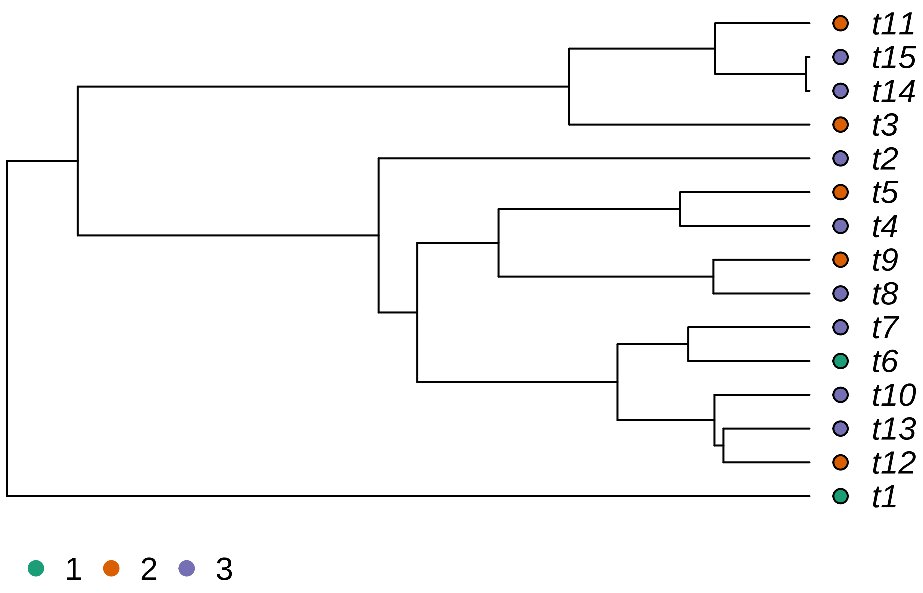
Plotting trees is one of the very interesting aspects of using R. Options are numerous and possibilities large. The most common function is `plot.phylo` from the ape package that has a lot of different options. I strongly suggest that you take a close look at the different options of the function `?plot.phylo`. Here is a basic example.

```
plot(tree, type="c")
add.scale.bar()
```



But R is also interesting to plot characters alongside trees. If you have a categorical character, you could use it to color the tips of the phylogeny.

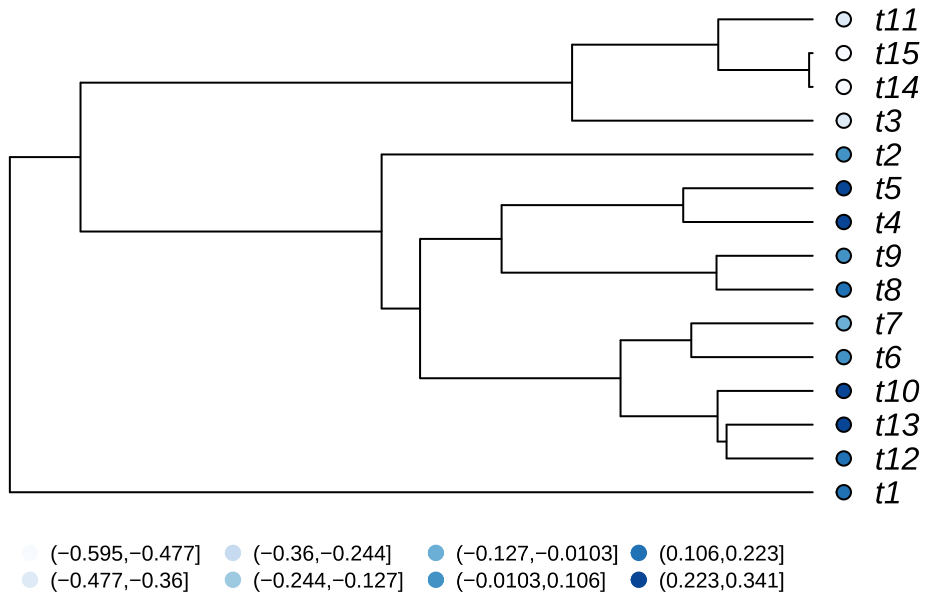
```
# Generate a random categorical character
trait2 <- as.factor(sample(c(1,2,3),size=length(tree$tip.label),replace=TRUE))
# Create color palette
library(RColorBrewer)
ColorPalette1 <- brewer.pal(n = length(levels(trait2)), name = "Dark2")
plot(tree, type="p", use.edge.length = TRUE, label.offset=0.2,cex=1)
tiplabels(pch=21,bg=ColorPalette1[trait2],col="black",cex=1,adj=0.6)
op<-par(xpd=TRUE)
legend(0,0,legend=levels(trait2),col=ColorPalette1,
       pch=20,bty="n",cex=1,pt.cex=1.5,ncol=length(levels(trait2)))
```



```
par(op) #reset graphical parameters to defaults
```

A similar result could be obtained with a continuous variable. Here, we will use the Brownian Motion model, which we will study in a further class, to simulate the continuous character.

```
# Breakdown continuous trait in categories
trait1.cat <- cut(trait1[,1],breaks=8,labels=FALSE)
# Create color palette
ColorPalette2 <- brewer.pal(n = 8, name = "Blues")
# Plot the tree
plot(tree, type="p", use.edge.length = TRUE, label.offset=0.2,cex=1)
tiplabels(pch=21,bg=ColorPalette2[trait1.cat],col="black",cex=1,adj=0.6)
op<-par(xpd=TRUE)
legend(0,0,legend=levels(cut(trait1[,1],breaks=8)),
       col=ColorPalette2,pch=20,bty="n",cex=0.7,pt.cex=1.5,ncol=4)
```

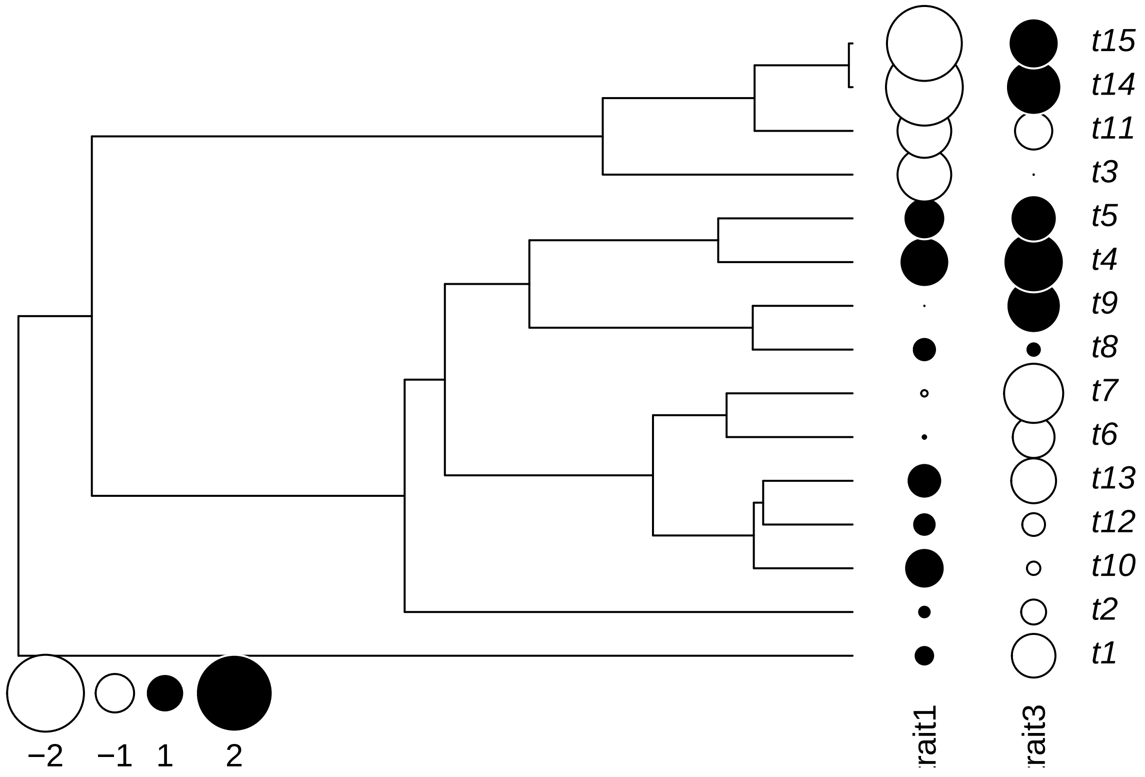


```
par(op) #reset graphical parameters to defaults
```

As expected from a character simlated with Brownian motion, you can see that closely related species tend to have more similar character values.

Another option to represent a continuous parameter is to use the function `table.phylo4d` from the `adephylo` package to represent the trait where its values are represented by sizes of different sizes and colors. It is also possible to plot multiple characters at the same time.

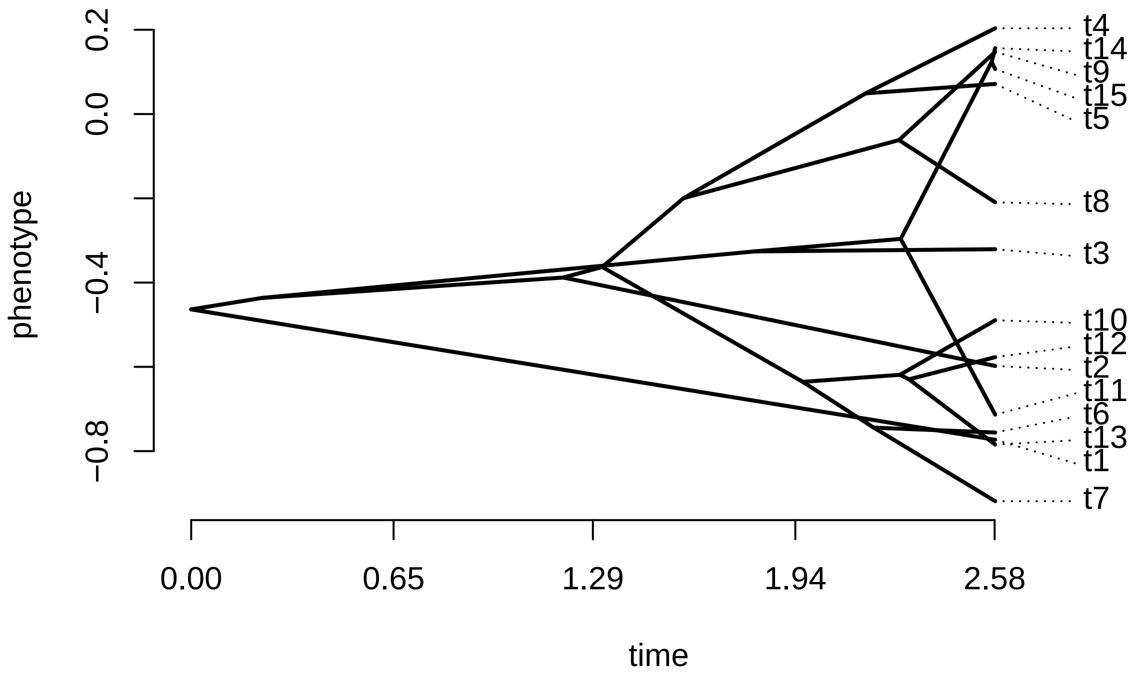
```
library(phylobase)
library(adephylo)
trait3 <- fastBM(tree, sig2=0.1, nsim=1, internal=FALSE) #simulate a faster evolving trait
trait.table <- data.frame(trait1=trait1[,1], trait3)
obj <- phylo4d(tree, trait.table) # build a phylo4d object
op <- par(mar=c(1,1,1,1))
table.phylo4d(obj,cex.label=1,cex.symbol=1,ratio.tree=0.8,grid=FALSE,box=FALSE)
```



```r
par(op) #reset graphical parameters to defaults
```

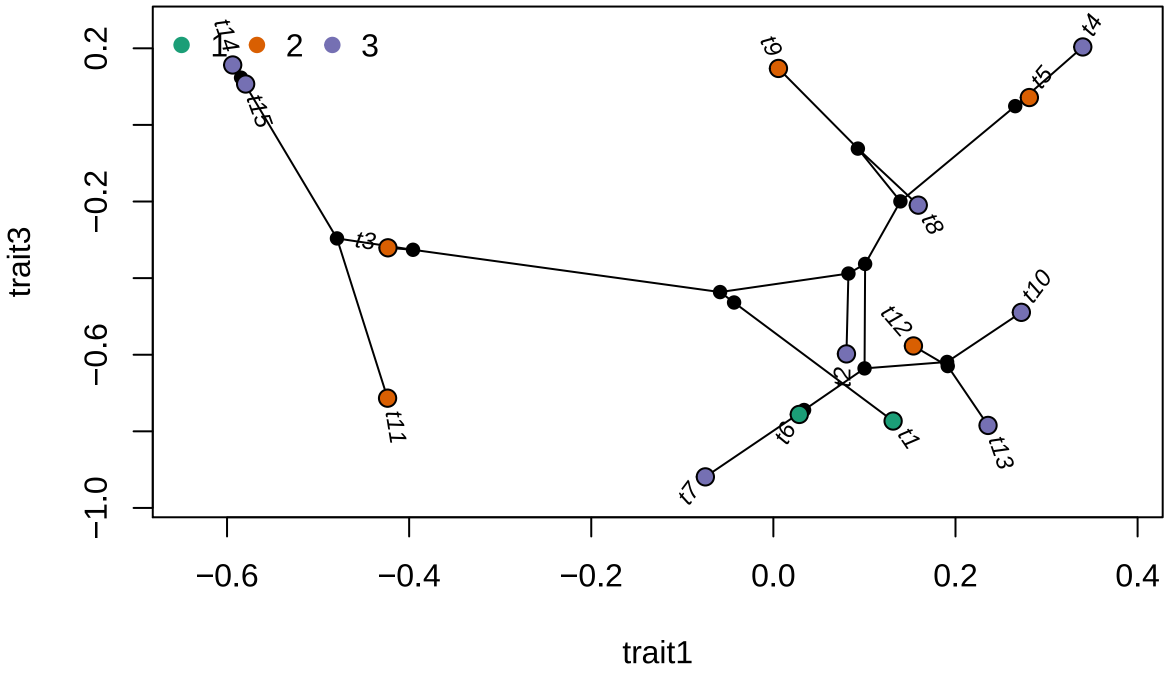
You can also represent a traitgram:

```r
require(phytools)
phenogram(tree,trait3,spread.labels=TRUE)
```



Finally, it is also possible to represent a tree on a 2-dimensional plot, coloring points with the categorical variable.

```
phylomorphospace(tree,trait.table)
points(trait.table,pch=21,bg=ColorPalette1[trait2],col="black",cex=1.2,adj=1)
legend("topleft",legend=levels(trait2),
       col=ColorPalette1,pch=20,bty="n",cex=1,pt.cex=1.5,ncol=length(levels(trait2)))
```



Handling multiple trees

In several cases, it is important to know how to handle multiple trees in R. These are normally stored in a `multiPhylo` object. Let's see an example.

```
trees <- pbtree(n=15,nsim=10)
trees
```

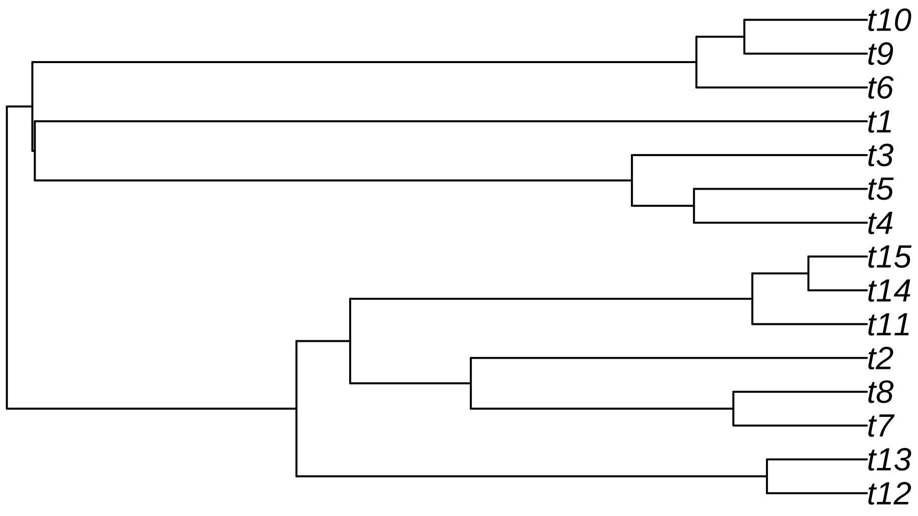
```
## 10 phylogenetic trees
```

You can see that the object is not the same as a phylo object. For instance, if you use the code `plot(trees)`, you will be prompted to hit enter to pass from one tree to the other. To access to individual trees, you need to use the following technique.

```
trees[[1]]
```

```
##
## Phylogenetic tree with 15 tips and 14 internal nodes.
##
## Tip labels:
##  t12, t13, t7, t8, t2, t11, ...
##
## Rooted; includes branch lengths.
```

```
plot(trees[[1]])
```

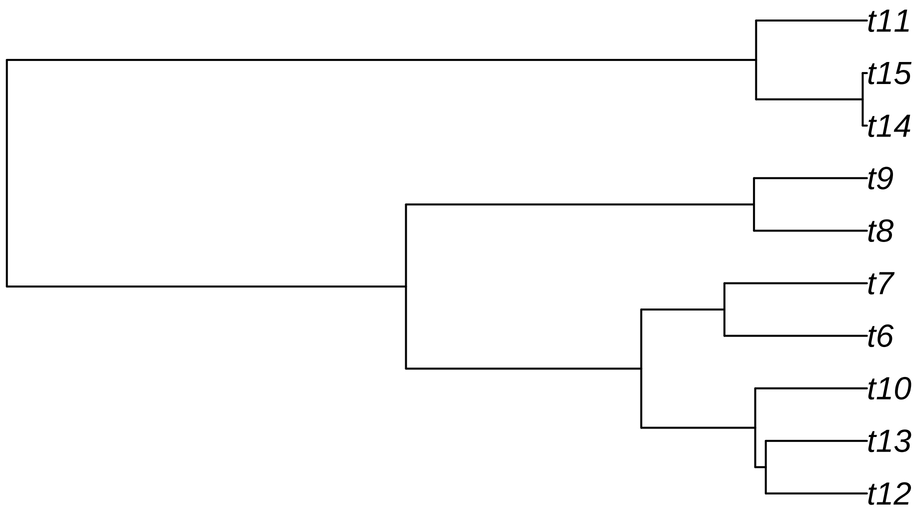


Manipulating trees

There are several manipulations that can be made to trees. Here are a few examples.

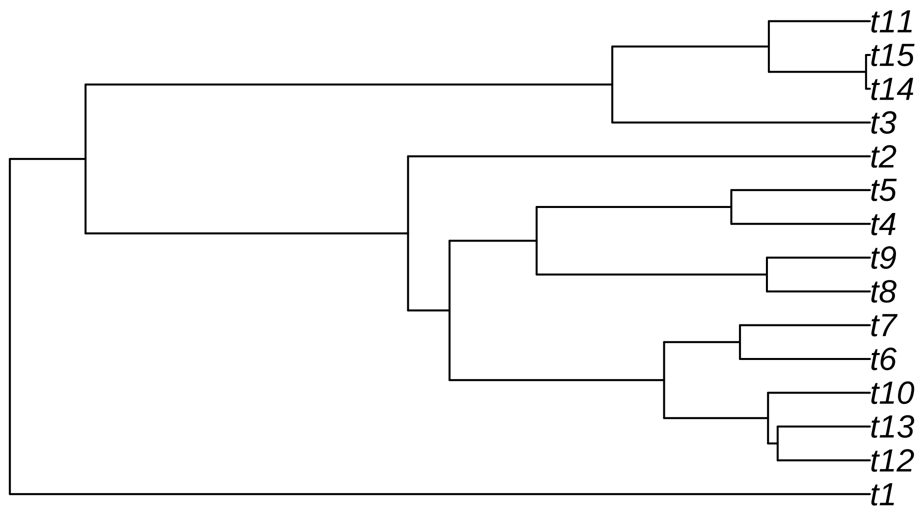
Drop tips

```
plot(drop.tip(tree,c("t1","t2","t3","t4","t5")))
```



Reroot trees

```
plot(root(tree,"t1"))
```



Get cophenetic distances

```
cophenetic.phylo(tree)
```

```
##           t1       t12       t13       t10        t6        t7        t8
## t1  0.000000 5.1637261 5.1637261 5.1637261 5.1637261 5.1637261 5.1637261
## t12 5.163726 0.0000000 0.5537062 0.6115882 1.2357552 1.2357552 2.5239338
## t13 5.163726 0.5537062 0.0000000 0.6115882 1.2357552 1.2357552 2.5239338
## t10 5.163726 0.6115882 0.6115882 0.0000000 1.2357552 1.2357552 2.5239338
## t6  5.163726 1.2357552 1.2357552 1.2357552 0.0000000 0.7801388 2.5239338
## t7  5.163726 1.2357552 1.2357552 1.2357552 0.7801388 0.0000000 2.5239338
## t8  5.163726 2.5239338 2.5239338 2.5239338 2.5239338 2.5239338 0.0000000
## t9  5.163726 2.5239338 2.5239338 2.5239338 2.5239338 2.5239338 0.6182759
## t4  5.163726 2.5239338 2.5239338 2.5239338 2.5239338 2.5239338 2.0010497
## t5  5.163726 2.5239338 2.5239338 2.5239338 2.5239338 2.5239338 2.0010497
## t2  5.163726 2.7730345 2.7730345 2.7730345 2.7730345 2.7730345 2.7730345
## t3  5.163726 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808
## t14 5.163726 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808
## t15 5.163726 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808
## t11 5.163726 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808 4.7093808
##            t9        t4        t5       t2       t3        t14        t15
## t1  5.1637261 5.1637261 5.1637261 5.163726 5.163726 5.16372606 5.16372606
## t12 2.5239338 2.5239338 2.5239338 2.773034 4.709381 4.70938076 4.70938076
## t13 2.5239338 2.5239338 2.5239338 2.773034 4.709381 4.70938076 4.70938076
## t10 2.5239338 2.5239338 2.5239338 2.773034 4.709381 4.70938076 4.70938076
## t6  2.5239338 2.5239338 2.5239338 2.773034 4.709381 4.70938076 4.70938076
## t7  2.5239338 2.5239338 2.5239338 2.773034 4.709381 4.70938076 4.70938076
## t8  0.6182759 2.0010497 2.0010497 2.773034 4.709381 4.70938076 4.70938076
## t9  0.0000000 2.0010497 2.0010497 2.773034 4.709381 4.70938076 4.70938076
## t4  2.0010497 0.0000000 0.8320314 2.773034 4.709381 4.70938076 4.70938076
## t5  2.0010497 0.8320314 0.0000000 2.773034 4.709381 4.70938076 4.70938076
## t2  2.7730345 2.7730345 2.7730345 0.000000 4.709381 4.70938076 4.70938076
## t3  4.7093808 4.7093808 4.7093808 4.709381 0.000000 1.54676106 1.54676106
## t14 4.7093808 4.7093808 4.7093808 4.709381 1.546761 0.00000000 0.02312826
## t15 4.7093808 4.7093808 4.7093808 4.709381 1.546761 0.02312826 0.00000000
## t11 4.7093808 4.7093808 4.7093808 4.709381 1.546761 0.60678260 0.60678260
##           t11
## t1  5.1637261
## t12 4.7093808
## t13 4.7093808
## t10 4.7093808
## t6  4.7093808
## t7  4.7093808
## t8  4.7093808
## t9  4.7093808
## t4  4.7093808
## t5  4.7093808
## t2  4.7093808
## t3  1.5467611
## t14 0.6067826
## t15 0.6067826
## t11 0.0000000
```

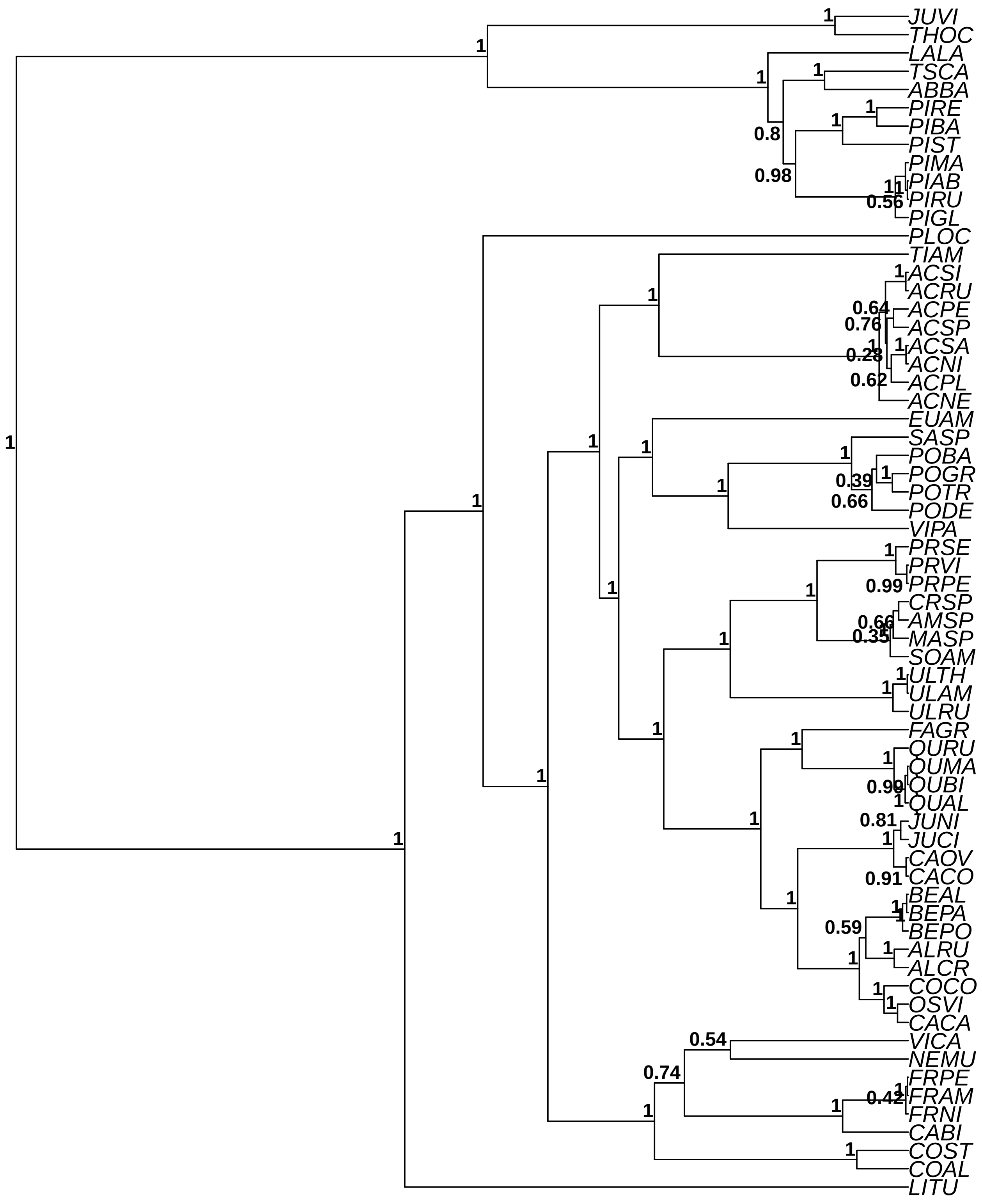
Ploting Posterior Probability values

The package `phyloch` has very useful functions to read trees from BEAST or MrBayes. Unfortunately, it has to be install from the binaries available from Christoph Heibl's webpage. Once ou have downloaded the package, you can install it using RStudio from the menu 'Tools>Install Packages' and select the package that you have just downloaded. You also need to install the package `XML`, which you can do using the command `install.packages("XML")`.

Read Bayesian trees

The functions `read.beast` and `read.MRBayes` parse chronograms in NEXUS format as produced by TreeAnnotator or output by MrBayes, respectively. Here is an example with a tree obtained from TreeAnnotator.

```
## Warning: package 'XML' was built under R version 3.1.3
```



```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [12]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
## [23] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
## [34]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [45] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [56]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
```

> If you want to drop tips from trees in phyloch packages, use the function `drop.tip2` from the `phyloch` package, and not the `drop.tip` from the `ape` package.

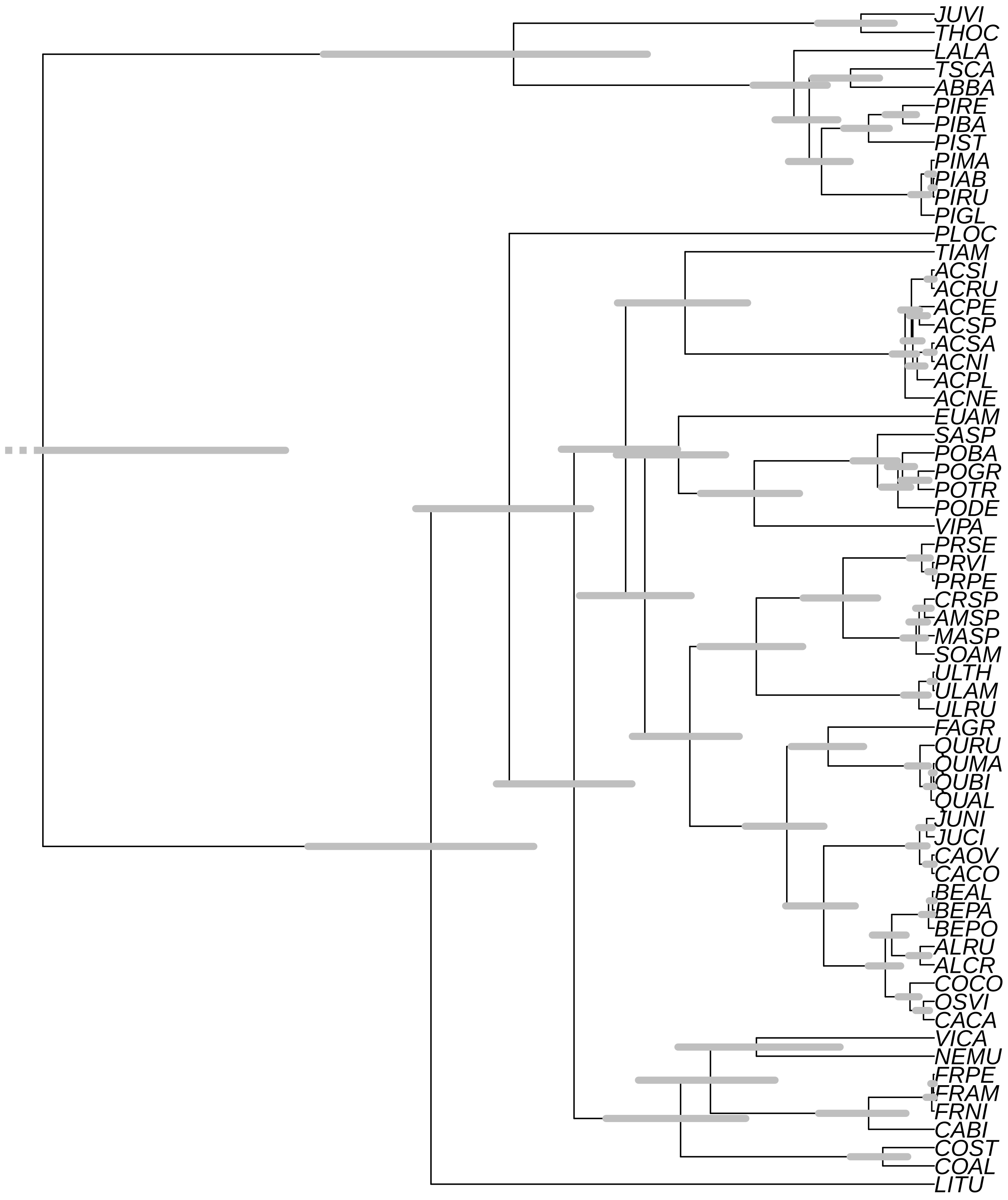
The function `read.beast` give a lot of annotations to the tree, which you can see using the command `str(tree)`. For instance, you can access the posterior probability values using `beasttree$posterior`.

Plot node bars

The `phyloch` package can also plot bars representing the highest posterior density (HPD) intervals of node ages.

```r
plot(beasttree)
HPDbars(beasttree, label = "height_95%_HPD")
```

```
## Warning in HPDbars(beasttree, label = "height_95%_HPD"): HPD bar(s) for nodes 66 exceed(s) plot regi
##   Try setting "x.lim" to c(-0.05212, 0.21959)
```



Info on Rmarkdown: http://rmarkdown.rstudio.com.