Comparative Methods Workshops

Chris Mitchell

Contents

Chapter 1

Welcome

Welcome to the online support materials for the Comparative Research Group at the University of Liverpool. The CRG is made up of staff and students engaging in comparative research across various areas of evolutionary biology.

1.1 For students

The materials here are intended to support you through your LIFE363 honours project. For this project you will be performing a comparative study (see **chapter 1** for more information) on an area of your choosing. At first, this is a daunting task but developing your own research here is excellent experience and gives you the opportunity to research an area that really interests you.

The vast majority of statistics here are performed in R [R Core Team, 2019]. You were introduced to R in LIFE223 as a powerful and flexible tool for statistical analysis. **Chapter 2** contains a brief refresher on some of the basics of R in case you need it. For more detailed recaps, please revisit your materials from LIFE223 as some of the code you wrote is likely to be useful this year!

Throughout this book you will see examples of R code and output like this.

```
print(answer)
```

```
[1] "Forty-two"
```

The code can be copied and pasted into your own version of R as you see fit. However, I would recommend that for the first time you are using a piece of code, type it out for yourself. This will help you get to grips with what each argument means.

You will also see some interactive R windows where you can enter your code

directly into this book and an online version of R will run it. This should give you an opportunity to learn more complex things and develop your R skills dramatically.

The rest of the book is populated with workshops and materials to help you learn specific comparative statistical methods. Some of these will be extensions of what you already met in LIFE223. **Chapter 6** looks at phylogenetically controlled ANOVA and **chapter 14** is all about phylogenetic regression.

Other methods may be entirely new to you such as ancestral state reconstruction (**chapters 7 - 10**) or path analysis (**chapter 15**). Don't be intimidated by this. All the code you need is gathered here and will remain available as long as you need it.

Chapter 2

Introduction

This chapter contains a very brief overview of the research we do in the **Comparative Research Group**. Taxonomically, the work done by group members is extremely broad. We've had projects on primates, octopuses, domestic mammals, birds and more! Here is a sample of titles from previous students.

- Identification of a cognitive niche in benthic octopods and possible areas for future study on cephalopod intelligence.
- Evolutionary precursors for the domestication of Artiodactyla.
- You are not what you eat: Lack of morphological convergence in beak and body size between the nectarivorous avian families Trochilidae and Meliphagidae.
- Investigating how lifestyle factors affect lifespan in reptiles.
- Ecological processes causing encephalisation in Madagascan lemurs.

2.1 What is the comparative method?

The comparative method is a catch-all term for a suite of approaches that involve using comparisons to answer scientific questions. In evolutionary biology, the comparative method refers to making comparisons between species or populations in order to identify patterns and relationships between traits of interest. Used correctly, this approach can be very powerful and allows us to ask large-scale questions about evolutionary patterns, adaptive processes and coevolutionary relationships.

The most basic kind of comparative study is comparing one species or lineage to another. For example, a recent paper made waves in the paleontology community by demostrating (after years of debate) that *Spinosaurus aegypticus* lived an aquatic lifestyle [Ibrahim et al., 2020]. The analysis centered around some newly recovered tail vertebrae with extremely long (1m!) spines. The tail of

Spinosaurus was compared to other animals including terrestrial theropods like *Allosaurus* and semi-aquatic tetratpods such as the crocodile. This comparison showed that the *Spinosaurus* tail was indeed specialised for powerful propulsion through the water (like a crocodile), seemingly settling the debate over whether any non-avian dinosaurs invaded the water.

Other comparative studies take data gathered from many species and search for patterns within that group. Studies like this rely a great deal on work done by others. For example, Simon Reader and colleagues [2011] carried out an extensive literature search looking for examples of five behavioural traits in many species of primate in over 4000 articles published over 75 years. The resulting database included examples of innovation, social learning, tool use, extractive foraging and tactical deception and was used to demonstrate a correlation between these behaviours and brain size, providing evidence of a general intelligence factor in primates similar to that in humans.

2.2 Tree thinking

Comparative studies can be great but there is a problem. In LIFE223 you learned about statistical assumptions. One of the most common and important assumptions of most statistical tests is that data are independent. To run a good comparative study we need to know that the data points we have are independent of each other. In evolutionary biology, we know that this isn't the case!

All living things exhibit a pattern of relatedness which depends on how much shared evolutionary history they have. For example, chimpanzees and human beings diverged about 6-7 million years ago. This means that they have much more shared evolutionary history than chimpanzees and *Spinosaurus* which are separated by hundreds of millions of years.

The best way of visualising this pattern of relatedness is with a phylogenetic tree.

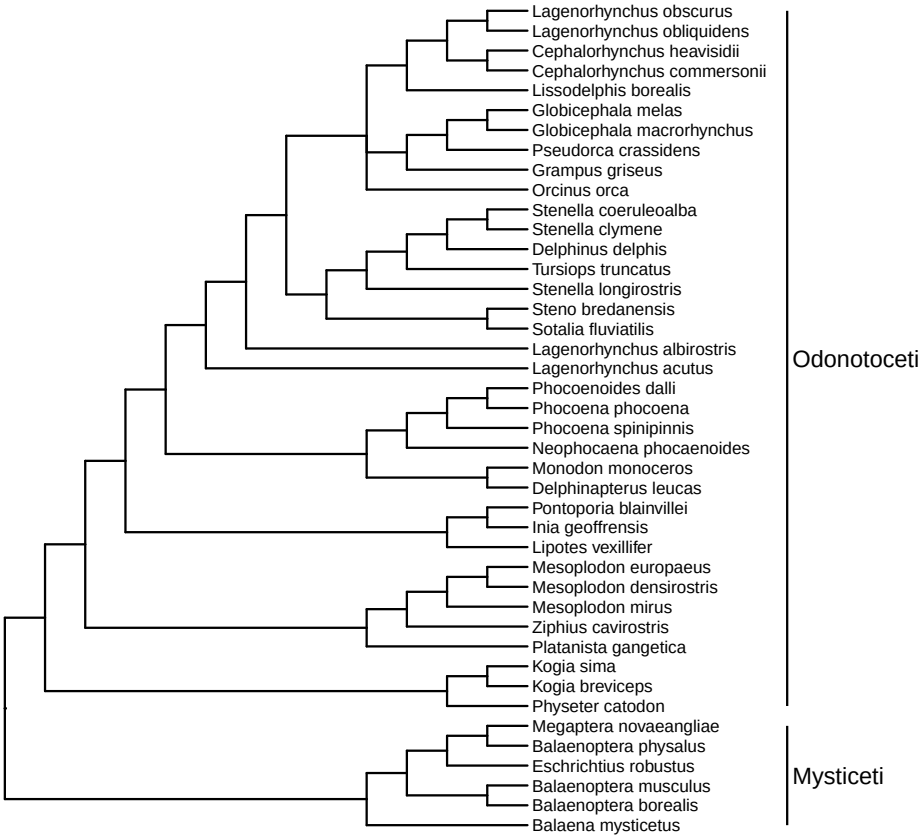


Figure 2.1: A cladogram of 42 cetacean species.

The extant species are displayed on the **tips** of the tree and are connected to each other according to the degree of relatedness by the **branches**. Figure 2.1 shows us the pattern of relatedness of 42 cetacean species. If we wanted to use these species in a comparative study to investigate the evolutionary history of the group, we would not have independent data points. This means that the assumptions of most statistical tests would be violated and we couldn't trust the results!

This is where phylogenetics comes to the rescue. We can use the pattern of relatedness described by the phylogeny to control for the non-indepedence of data points. To show you what I mean, let's consider body size in those 42 species of cetacean. If we were to show the distribution of body size in the group, we would see that the vast majority of the largest sizes are found in the mysticetes whereas the smaller species tend to be odontocetes. If we viewed these data points as all independent we might say that very large bodies have evolved 7 times in the group (once for each mysticete and once for the sperm

whale) whilst small body size has evolved in all the other species (35 times).

In fact, the close relatedness of 6 of the large bodied species suggests that large body size evolved once and not independently for each of these species. Their shared evolutionary history explains why their traits (body size in this case) are so similar. The seventh example of a large body (sperm whales) does not share very much history with the other 6 and this may be of some interest to us. It suggests an independent evolution of large body size and potentially something of interest to us as researchers.

So hopefully you can see how taking phylogeny into account can be illuminating. For a broader (and much more useful) introduction to phylogenetics and its use in evolutionary biology, check out these sources:

- Tree Thinking: An Introduction to Phylogenetic Biology [Baum and Smith, 2012]
- The Comparative Approach in Evolutionary Anthropology and Biology [Nunn, 2011]

Chapter 3

R recap

In LIFE223, we taught you how to use R for statistical analysis and visualising data. This chapter will contain a basic overview of some of the things from 223 that you may find useful as we proceed. You only need to bother with this if you are new to R or have blocked it from your memory since you last used it.

3.1 Basics

R works well as a calculator.

```
6*7
```

```
[1] 42
```

However, R is capable of a great deal more than just simple mathematical operations like multiply and divide. It also has functions that can calculate some common descriptive stats like mean and standard deviation.

```
mean(x)
```

```
[1] 41.96491
```

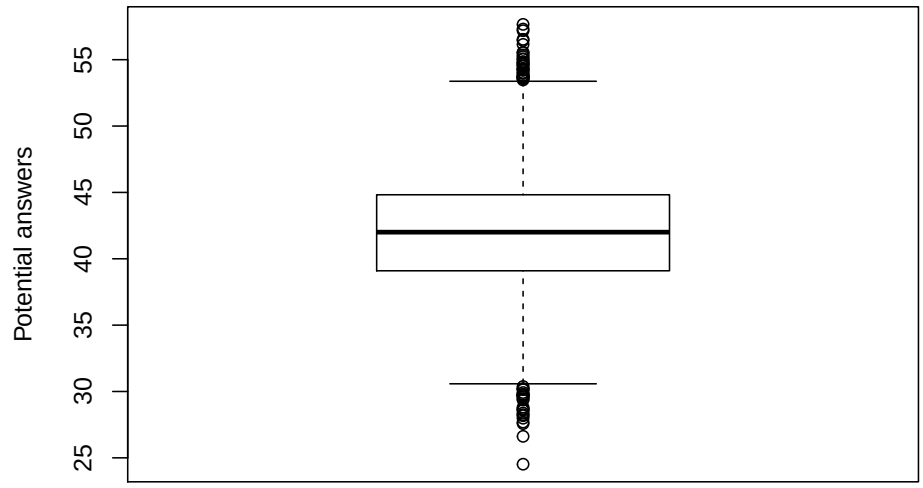
```
sd(x)
```

```
[1] 4.192614
```

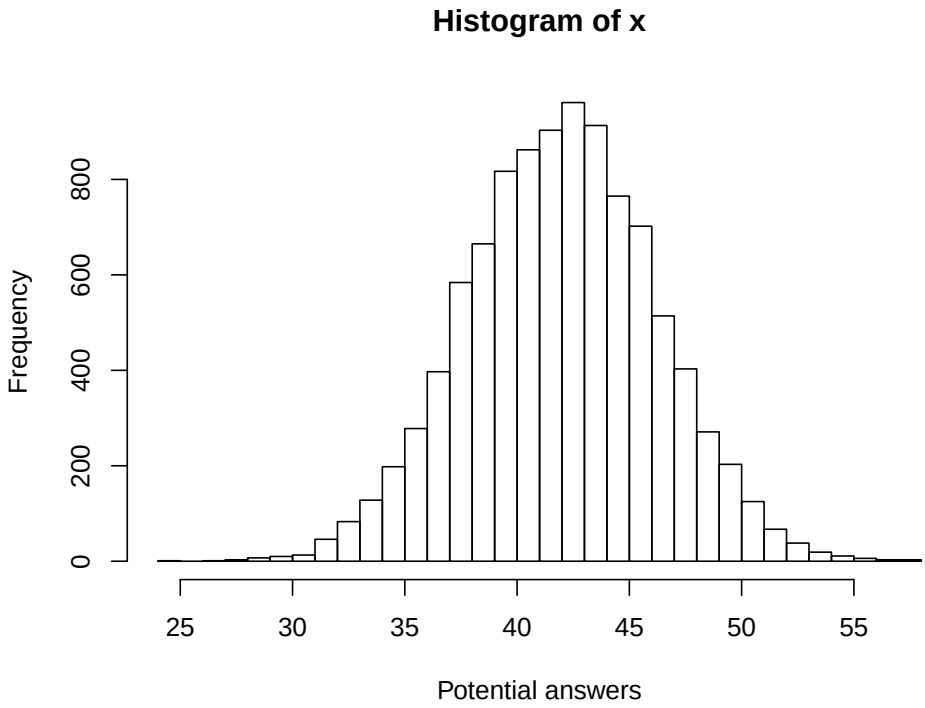
3.1.1 Plotting

R is also a very flexible graphical tool. From LIFE223, you probably remember a few basic plotting functions. Each function in R has arguments that can be added to label axes or change point size as you can see in these plots.

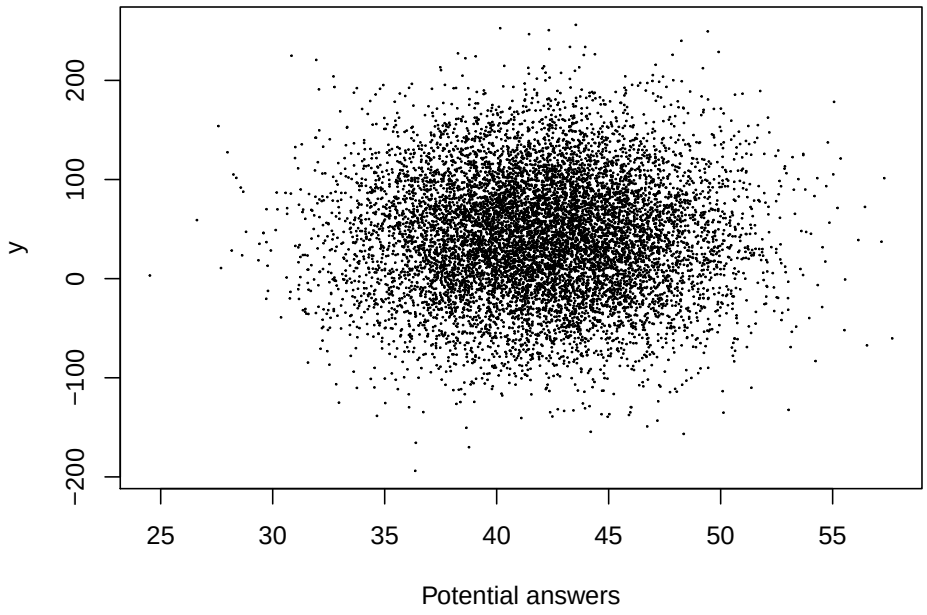
```r
boxplot(x, ylab = "Potential answers")
```



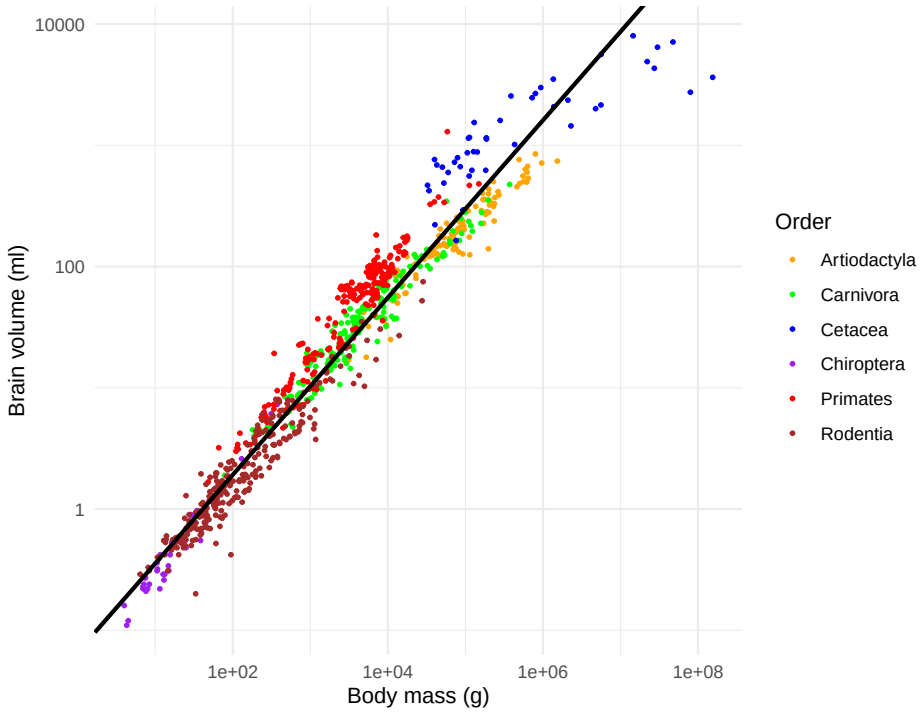

```r
hist(x, xlab = "Potential answers", breaks = 25)
```



```
plot(x, y, xlab = "Potential answers", pch = 19, cex = 0.1)
```



For much of this book, I will actually be doing most plotting in a package called **ggplot2**. This package has a slightly different syntax to get used to but the increased flexibility you have will be a good payoff. Plus the plots look quite nice.



3.1.2 The working directory

The working directory is the folder on your computer where R's attention is focused. This is where you should store any files you need R to open. You can find out the path of the current working directory using the function **getwd()**

```
getwd()
```

```
[1] "/Users/chrismitchell/Google Drive/University of Liverpool/GitHub Stuff/bookdownCR
```

If this isn't the folder we want as our working directory, we can just as easily change it with **setwd()**

```
setwd("~/Desktop/My R Folder")
```

If you are using RStudio there is also a shortcut to do this in the *Files* pane (usually bottom right). Use this pane to navigate to your chosen directory and then use the drop down menu under *More* (look for a blue cog) to set the current folder as your working directory.

If you aren't using RStudio, I'd strongly suggest you start. It's much more user friendly than base R.

3.1.3 Loading data

Data comes in many forms and R is capable of reading most of them if you know the correct functions. One of the most common formats is *comma separated values*. This has the file extension .csv at the end of the filename. If you open a .csv file with MS Excel or Numbers, you will see that it usually looks much like a spreadsheet. To load a .csv data file into R, use the function **read.csv()** as shown here.

```
data <- read.csv(file = "DATAFILE.csv")
```

For other data formats, you may require a different function. For example, data may be provided as a text file (extension .txt). In this case, you need **read.table()**. Note that with this function you need to specify that your data has a header (a top row with names for columns) whereas **read.csv()** assumes this by default.

```
data <- read.table(file = "DATAFILE.txt", header = TRUE)
```

3.1.4 Subsetting

Let's say I want to subset my data based on a certain condition. I can achieve this multiple ways but one of the simplest is the function subset.

```
newdata <- subset(data, species == "Homo sapiens")
```

This function takes a subset of the object *data* and applies the rule that the value of each row in the column *species* be *Homo sapiens*. Thus it extracts the lines of data that are from human beings.

3.2 Errors

Error messages are a part of life with R. You are not expected to be able to interpret every single one immediately and you definitely shouldn't panic or give up when you get one.

Here's a basic error message:

```
data <- read.csv("mydaat.csv")
```

```
Warning in file(file, "rt"): cannot open file 'mydaat.csv': No such file or
directory
```

```
Error in file(file, "rt"): cannot open the connection
```

The message tells me that R "cannot open the connection" and no such file exists. This means that R cannot find the file I was looking for in the current working

directory. It could be because I haven't set the correct working directory or the file is there but in a different format. In this case, the error has appeared because I have spelled the name incorrectly. I have sent R looking for a file called *mydaat.csv* instead of *mydata.csv*. Always remember that R is a useful idiot and will only do exactly what you tell it to do!

3.3 Google

The most important skill you need for using R is the ability to use Google (other search engines are available). It may seem odd but almost any problem you will ever encounter with R can be solved by a quick Google search.

If you come up against a confusing error message, copy and paste the message into Google. You will quickly land on one of the forums where someone else has asked about the same error message. The odds are pretty good you'll discover an explanation for the problem there.

If you don't know how to do something, pop the name of what you want to do into Google and add "in R" at the end and there will almost always be a tutorial on the first page of results with exactly what you need.

Seriously, Google is your strongest ally here. The community of R users has populated the internet with endless advice and guidance for every level from beginner to the most advanced of users. That brings me to my next point…

3.4 Stealing

If imitation is the greatest form of flattery then learning to code in R is just about the most flattering thing you can do. The internet is teeming with examples of R code for all kinds of purposes including in this very book. Take it without thinking twice.

You will have acheived a pretty good level of skill in R when you can take someone else's code and edit it for your own purposes. This is the **core skill** of R and once you can do that, you'll be unstoppable.

Chapter 4

Phylogenetic trees and where to find them

This chapter is a brief overview of some key concepts that may be useful when performing comparative research.

4.1 Phylogeny

Phylogeny is the term used to describe the evolutionary history of a group of species. The most common representation of phylogeny is a phylogenetic tree. There is a lot of terminology around phylogenetic trees. Here we will start with the very basics that will come up a lot in this book.

The **tips** of the tree represent the species/populations/individuals described by the tree. The **branches** of the tree represent the pattern of relationships between species. The **nodes** of a tree represent the most recent common ancestor of the lineages that diverge from that node. A **clade** is a monophyletic grouping of lineages. A grouping is **monophyletic** only if all members of that group descend from a common ancestor to the exclusion of others. for example, humans and apes form a monophyletic grouping but humans, apes and parrots do not.

Here is an example of a phylogenetic tree displaying the relationships of modern dog breeds taken from a nice paper investigating the evolutionary history of the domestic dog [Parker et al., 2017]

The tree shown above is a **cladogram** meaning that the lengths of the branches do not carry any real meaning. This tree is only useful for interpreting the relatedness of the species. It does not give us any information about the amount of evolutionary change or the amount time between nodes.

By contrast, the following tree has **branch lengths**. The tree is based on genetic analysis of 173 species of hymenoptera (bees, ants and wasps) [Peters et al., 2017]. In this tree, the branch lengths represent time (in millions of years) as calculated from analysis of over 3,000 genes and calibrated using fossils.

Branch lengths do not always represent evolutionary distance as time. In some cases, evolutionary distance is represented as the amount of change on each branch. The next tree was built based on a brain development gene (MCPH1) in cetaceans (whales, dolphins and porpoises) [McGowen et al., 2011]. On a tree like this, longer branches indicate more character changes along the branch. In this case the character changes will be changes in genetic sequence but for other trees it may be morphological characters, protein sequences characters or a combination.

4.2 Building trees

Developments in the field of phylogenetics have meant that there are many ways to construct a phylogeny. Many of the modern methods are highly sophistaicated and for now, these are not the subject of this book. However, it may help you to have a brief introduction to the logic behind building a phylogeny.

4.3 Locating trees

4.3.1 File format

Chapter 5

Plotting trees in R

5.1 Phylogenies in R

From LIFE223, you know R as a powerful statistical tool. You will also be aware that it is an incredibly flexible tool for plotting data. In this workshop, we will be working with phylogenies in R and manipulating them to produce informative plots.

5.1.1 Packages used

In this section we'll mostly be using a package called ggtree [Yu et al., 2017, 2018]. To install it, we need another package called **BiocManager** [Morgan, 2019].

```
install.packages("BiocManager")
BiocManager::install("ggtree")
library(ggtree)
```

```
library(ggtree)
```

We will also need to use phylobase [R Hackathon et al., 2019], ggimage [Yu, 2019] and it would help to have the tidyverse packages loaded [Wickham, 2017] since we'll be using the syntax of ggplot2. If you get an error message, make sure the packages are installed first.

```
library(tidyverse)
library(phylobase)
library(ggimage)
```

5.2 Importing your tree

Let's start by importing a tree. Make sure your working directory is set to wherever you have saved the tree_newick file. If you run this line, you should see an object called "tree" appear in your global environment.

```
tree <- read.tree("tree_newick.nwk")
```

If we take a look at the structure of our tree object using the **str** function we can see that the tree is stored as an object of class **phylo**. If you are using a block of trees (more on this subsequent chapters) it will be an object of class **multiphylo**.

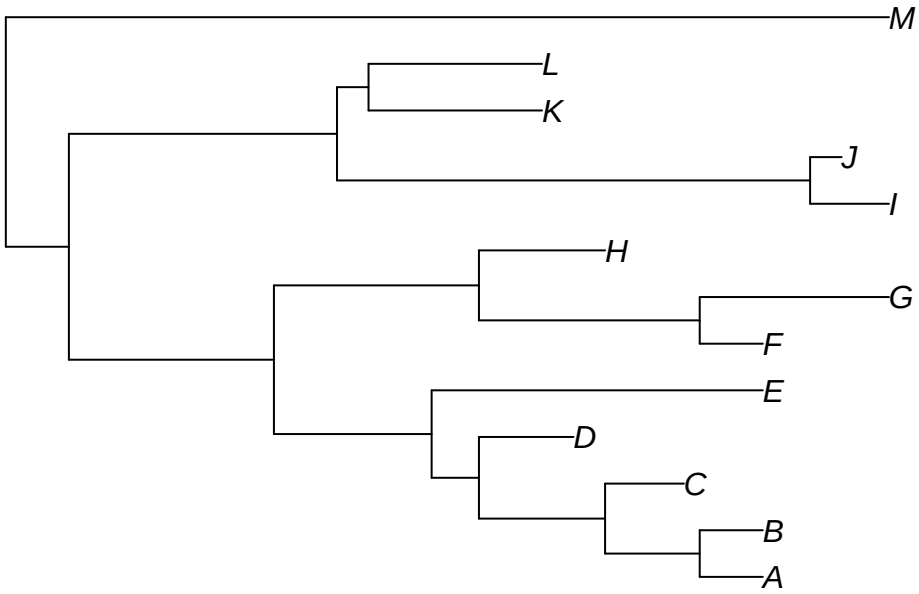
```
str(tree)
```

```
List of 4
 $ edge       : int [1:24, 1:2] 14 15 16 17 18 19 20 20 19 18 ...
 $ edge.length: num [1:24] 4 13 10 3 8 6 4 4 5 6 ...
 $ Nnode      : int 12
 $ tip.label  : chr [1:13] "A" "B" "C" "D" ...
 - attr(*, "class")= chr "phylo"
 - attr(*, "order")= chr "cladewise"
```

We can see a list of 4 elements of the tree object. The first (**edge**) contains the edges (also known as branches) of the phylogeny and their labels. The next is **edge.length** which contains the lengths of the branches if present (see **chapter 3** for more details). **Nnode** specifies the number of nodes and finally **tip.label** contains the labels of the tips. In this case, we just have letters for tip labels.

Things are often clearer when we plot them. We can do this for trees with the **plot** function in base R. This function is incredibly versatile and you should recognise it from LIFE223. Here we are using very different arguments.

```
plot(tree)
```



This plot is fine for a quick check to make sure the tree looks as we expected it to. Let's look at making a more attractive plot with ggtree.

5.3 ggtree

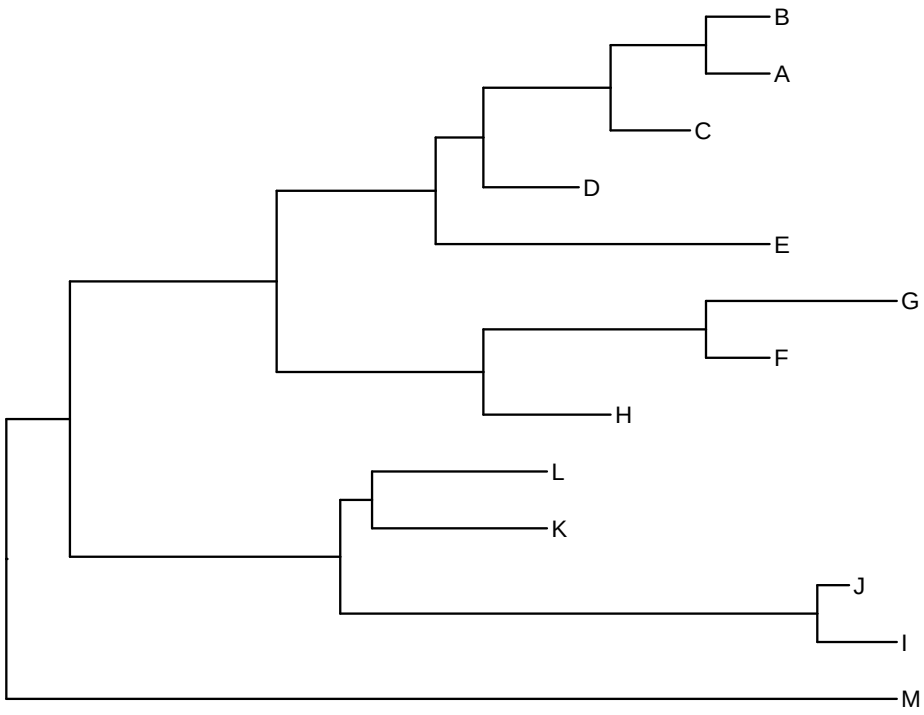
The package ggtree is an extension of **ggplot2**, a popular plotting package from the **tidyverse** family of packages. The syntax we'll be using here is a little different that what you may be used to so don't get intimidated. **ggtree** uses the same syntax as **ggplot2**. This works by creating layers (known as **geoms**) and plotting them over each other to build up the plot.

We'll start by using ggtree to plot our tree. Below is the base layer of the plot. There are many other options we can include to customise our tree. Try some out in this R window to see how they effect your plot.

5.3.1 Geoms

Geoms are new layers to plot on or alongside your tree. Now let's try plotting it whilst adding new layers. These geoms can be combined as you see fit. This gives you a lot of flexibility in how you plot your trees. For example, we can add a geom to include the tip labels for our tree.

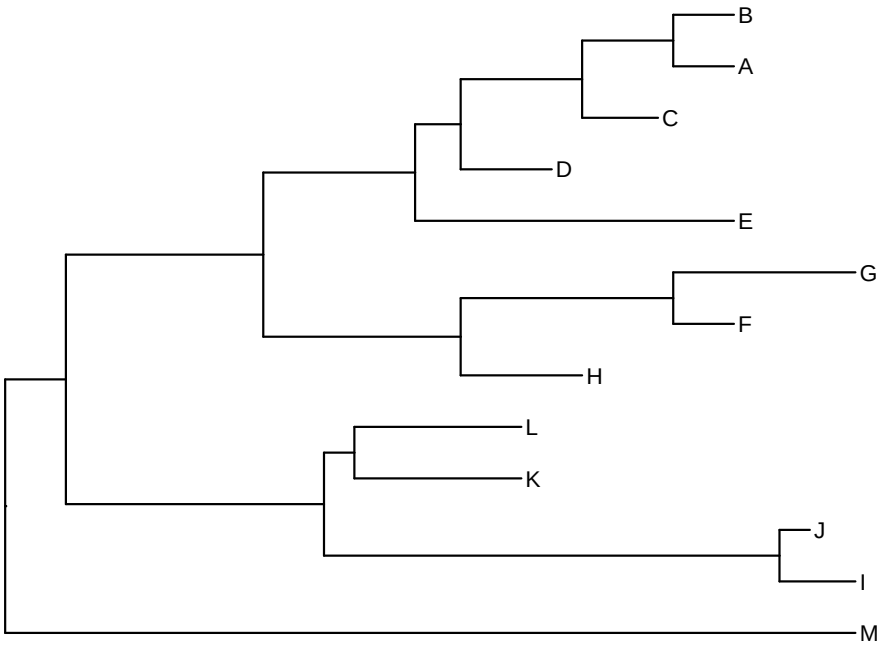
```
ggtree(tree) +
  geom_tiplab()
```



And we can add a title

```
ggtree(tree) +
  geom_tiplab() +
  ggtitle("A phylogeny of letters. For some reason...")
```

A phylogeny of letters. For some reason...



There are many geoms you can use to add more information to your plot. Here are just a few that you may want to investigate.

```
geom_tiplab() #adds tiplables
geom_tippoint() #adds points at the tips
geom_nodepoint() #adds points at the nodes
geom_nodelab() #adds labels for nodes
geom_cladelabel() #adds labels for clades
```

5.3.2 Labelling clades

As an example of what you might like to do with ggtree, let's have a look at adding some labels to identify some clades on our tree. To label clades, we need to be able to identify the node of the most recent common ancestor. The function **MRCA** in the package **phylobase** [R Hackathon et al., 2019] tells us that the common ancestor of C and E is node 17.

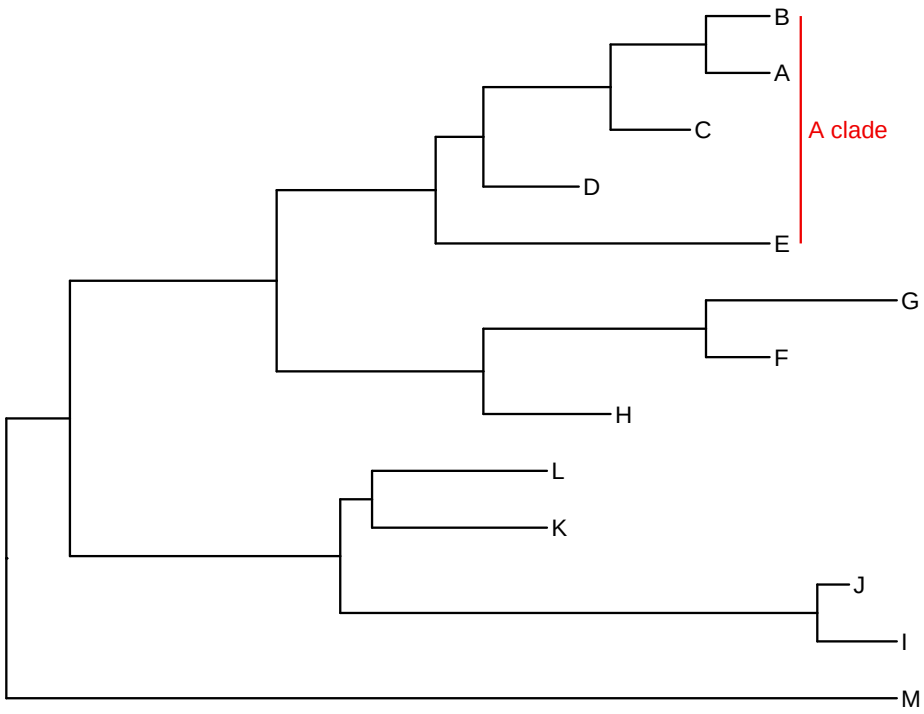
```
phylobase::MRCA(tree, tip = c("C", "E"))
```

```
[1] 17
```

We can now use the **geom_cladelabel** geom to add a simple label for the clade descended from the appropriate node. Take note of the arguments I've added to

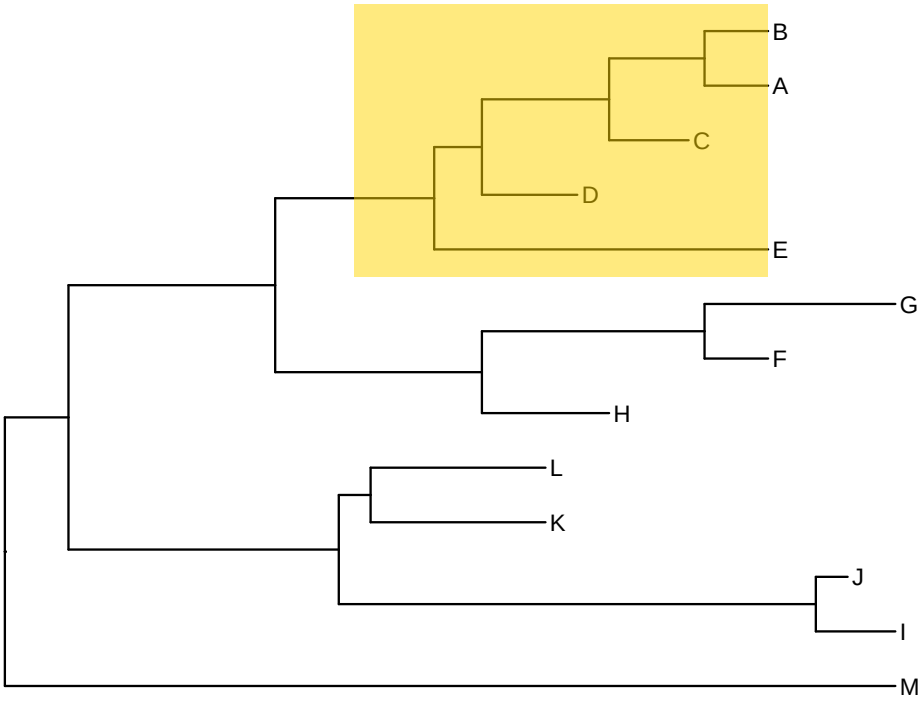
customise the geom. You may want to play around with these options yourself to see how they work.

```
ggtree(tree) +
  geom_tiplab() +
  geom_cladelabel(node=17, label="A clade",
                  color="red2", offset=1)
```



Pretty good but there are other options. This is a matter of personal preference. You may prefer to overlay a translucent rectangle over your clade of interest.

```
ggtree(tree) +
  geom_tiplab() +
  geom_hilight(node=17, fill="gold")
```



Use the R window below to experiment with the available geoms in ggtree. Find a combination that suits you and your tree.

5.4 Adding images to trees

As you probably noted in chapter 3, adding images to a plot is an excellent way to annotate your tree. The ggtree package can do this as you can see here.

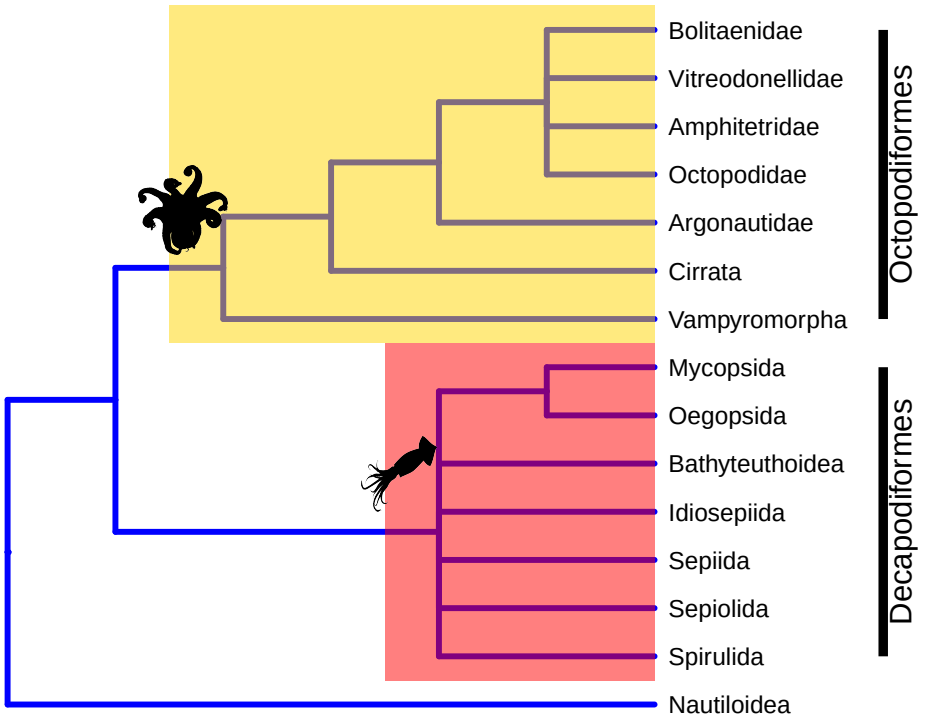


Figure 5.1: Plot of cephalopod families annotated using ggtree and Phylopic.

This phylogeny is annotated in a number of useful ways. The tip labels describe cephalopod families. The superorders (octopodiformes and decapodiformes) are highlighted by gold and red rectangles as well as a bar across the tips. this demonstrates how multiple geoms can combine to make a plot easy to interpret.

The most interesting thing for our purposes are the silhouettes at the root of each superorder. The octopodiformes have an octopus and the decapodiformes have a squid as example taxa from within the superorder.

5.4.1 Phylopic

The silhouettes used for that plot are from a website called Phylopic. Phylopic provides open source biological silhouettes that are free to use. We're now going to look at how to get these silhouettes and use them to annotate our trees.

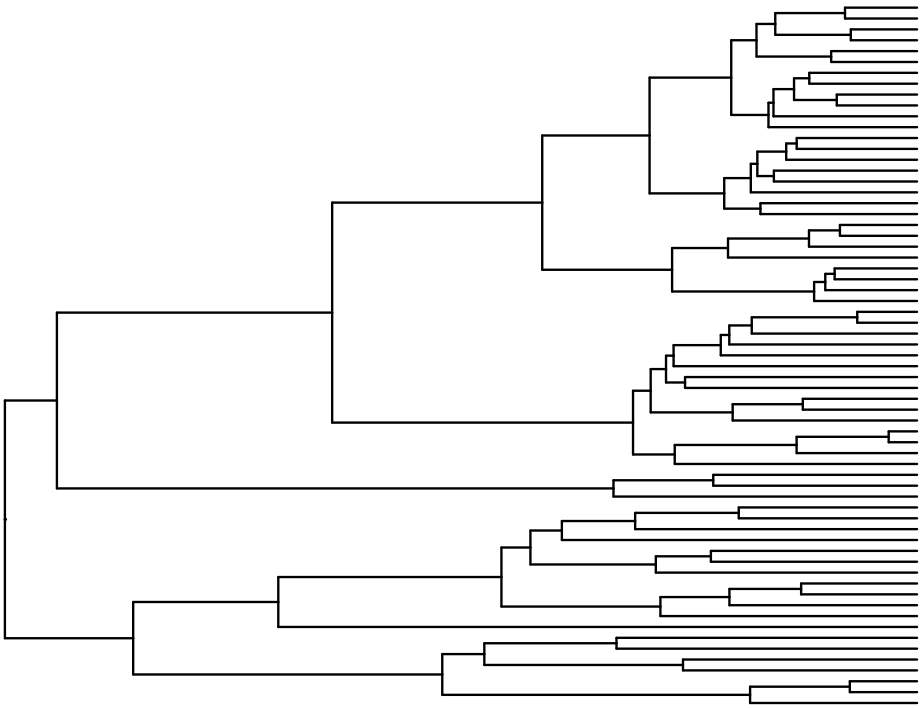
Let's start with loading an example tree. This one is a primate tree courtesy of Randi Griffin. You'll notice that I'm loading this tree using a url. This is because I'm loading a file directly from GitHub, a repository for all sorts of code and the host of this site! Randi (and many other coders) make some of the

things they produce freely available through GitHub. This can be data, files or code.

```
primates <- read.nexus("https://raw.githubusercontent.com/rgriff23/Dissertation/master/Chapter_2/
```

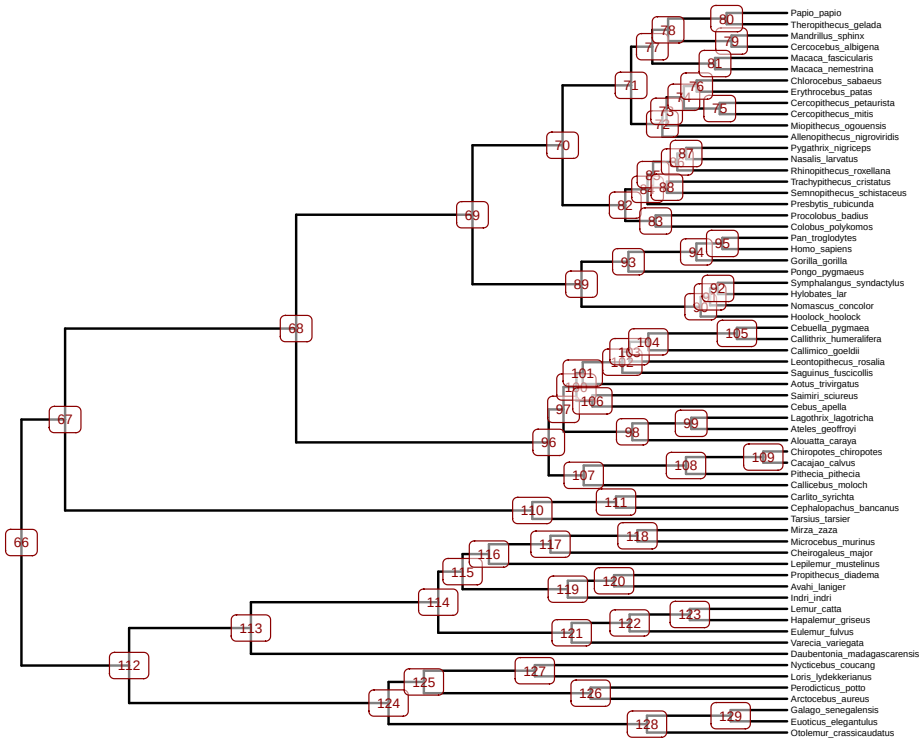
Let's plot the new tree first. Here I'm assigning the plot to a named object (p1) in R. This means that instead of immediately printing out the plot, R stores it in the working directory. The reason for doing this will become clear as we go on. It saves us typing out every line of code each time we want to add a new geom!

```
p1 <- ggtree(primates)
p1
```



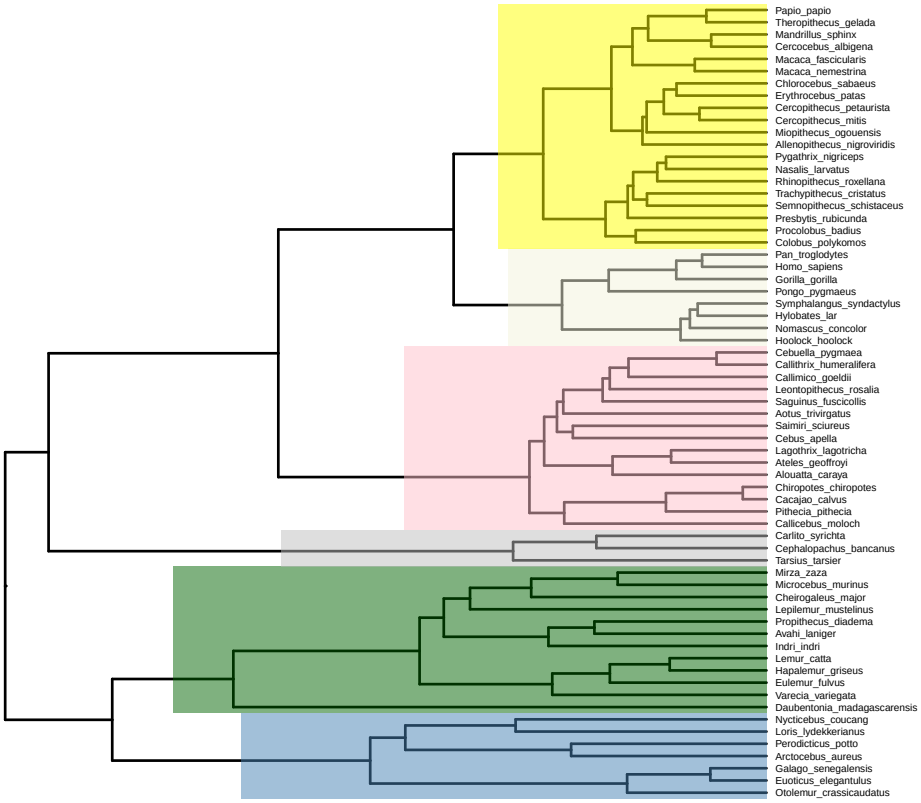
Let's use what we already know about ggtree to customise this plot into something more useful. In particular, this plot is quite useful because it tells us the numbers of each node and we will need that later on.

```
ggtree(primates) +
   xlim(0,90) +
   geom_tiplab(size=1.5) +
   geom_label2(aes(subset=!isTip, label=node), size=2, color="darkred", alpha=0.5)
```



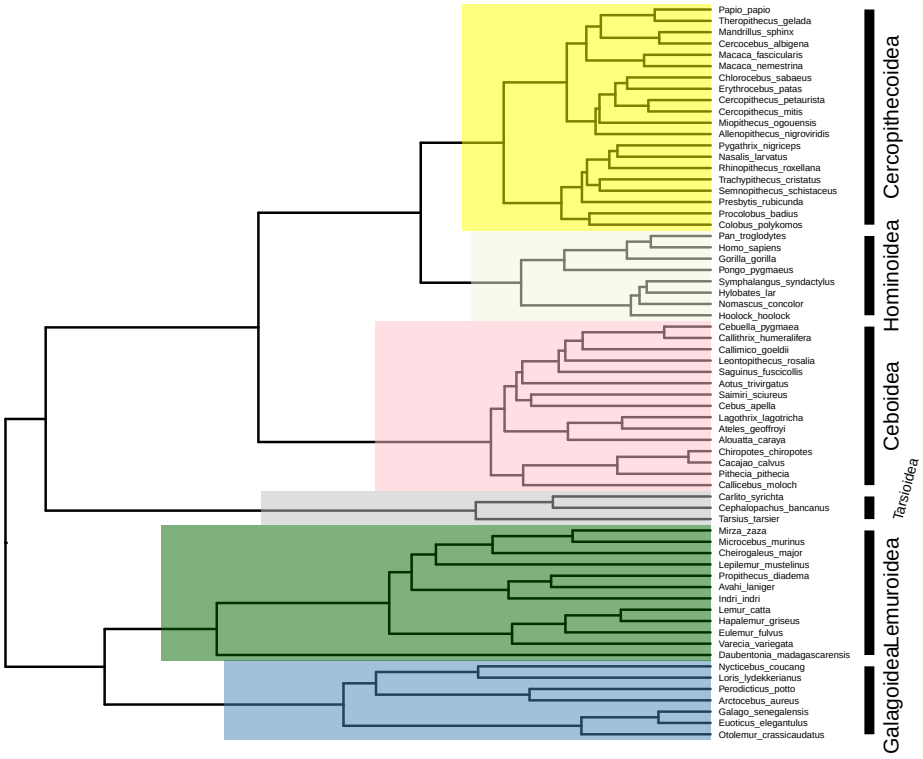
Let's label the 6 primate superfamilies using the node numbers I have extracted from the previous plot. You can choose whatever colours you prefer here. I've also added some useful features to this code. the use of **xlim()** can be very useful when plotting a tree with some extra space for more details. Here I've set the limits of the x dimension (the horizontal) to be between 0 and 100. This gives me space for later annotations.

```
p2 <- ggtree(primates) +
  xlim(0,100) +
  geom_tiplab(size=1.5, offset=0.5) +
  geom_hilight(node=124, fill="steelblue", alpha=0.5) +
  geom_hilight(node=113, fill="darkgreen", alpha=0.5) +
  geom_hilight(node=110, fill="gray", alpha=0.5) +
  geom_hilight(node=96, fill="pink", alpha=0.5) +
  geom_hilight(node=89, fill="beige", alpha=0.5) +
  geom_hilight(node=70, fill="yellow", alpha=0.5)
p2
```



So far so good. Let's add on bars to label the superfamilies like I did for the cephalopod version. This time, I'll add the new details to the object p3 to save retyping. Take note of the arguments in each label. You may need to twist these with some trial-and-error to make sure they suit your plot window.

```
p3 <- p2 +
  geom_cladelabel(124, "Galagoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(113, "Lemuroidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(110, "Tarsioidea", offset=15, barsize=2, angle=75,
                  offset.text=2.5, hjust=0.2, fontsize=2) +
  geom_cladelabel(96, "Ceboidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(89, "Hominoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(70, "Cercopithecoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3)
p3
```



There are some helpful details here, such as the fact that the label for Tarsioidea is off at an angle to avoid overlapping with other labels (*angle = 75*). The extra arguments in these options demonstrate how much control you can exercise over each geom.

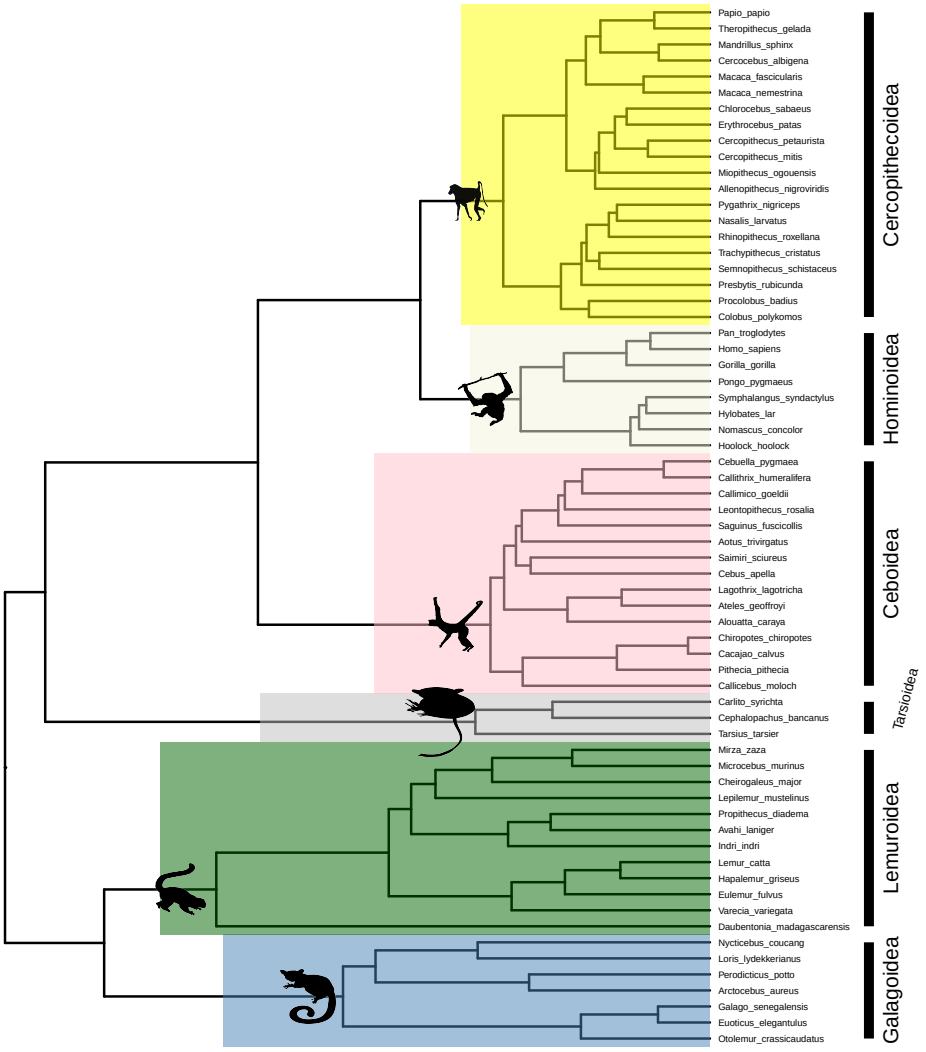
Now let's get to adding images. The way to do this is a little awkward but I think it's worth the hassle. The first thing we have to do is gather the links for each image we want to use. I've chosen to do this by building a small data frame containing the urls to the images on phylopic, the names of the super families I want to label and the nodes I want to plot the images on.

```
images <- data.frame(node = c(124,113,110,96,89,70),
phylopic = c("http://phylopic.org/assets/images/submissions/
7fb9bea8-e758-4986-afb2-95a2c3bf983d.512.png",
"http://phylopic.org/assets/images/submissions/
bac25f49-97a4-4aec-beb6-f542158ebd23.512.png",
"http://phylopic.org/assets/images/submissions/
f598fb39-facf-43ea-a576-1861304b2fe4.512.png",
"http://phylopic.org/assets/images/submissions/
aceb287d-84cf-46f1-868c-4797c4ac54a8.512.png",
"http://phylopic.org/assets/images/submissions/
0174801d-15a6-4668-bfe0-4c421fbe51e8.512.png",
```

```
"http://phylopic.org/assets/images/submissions/
72f2f854-f3cd-4666-887c-35d5c256ab0f.512.png"),
species = c("Galagoidea","Lemuroidea","Tarsioidea",
"Ceboidea","Hominoidea","Cercopithecoidea"))
```

Once we have the urls we need in a nice dataframe, we can pipe them into the **geom_nodelab** geom and the end product should appear.

```
p3 %<+% images +
  geom_nodelab(aes(image = phylopic), geom = "image", size = .04, nudge_x = -4)
```



As you can probably tell, the images don't have to be from Phylopic. You can use any images you have the rights to in exactly the same way!

5.5 Further info

This chapter barely scratches the surface of what ggtree is capable of. For much more detail, have a look at Guangchuang Yu's very own Bookdown covering the topic. You can access the book by clicking here or by running the following code in R once you have ggtree installed.

```
vignette("ggtree", package = "ggtree")
```

Chapter 6

ANOVA

Analysis of variance (ANOVA) is something you should recognise from your quantitative skills course. This chapter will begin with a brief recap before showing you how to perform phylogenetically corrected ANOVA.

6.1 Analysis of variance

Analysis of variance asks if there are differences in the mean values between 3 or more categories. If there are only two categories (Terrestrial/Aquatic for example), then you need a t-test.

In LIFE223 you analysed the results of an experiment in which corncrake hatchlings were raised on four different supplements in addition to their normal diet.

```
Warning: Ignoring unknown parameters: fun.y
```

```
No summary function supplied, defaulting to `mean_se()`
```

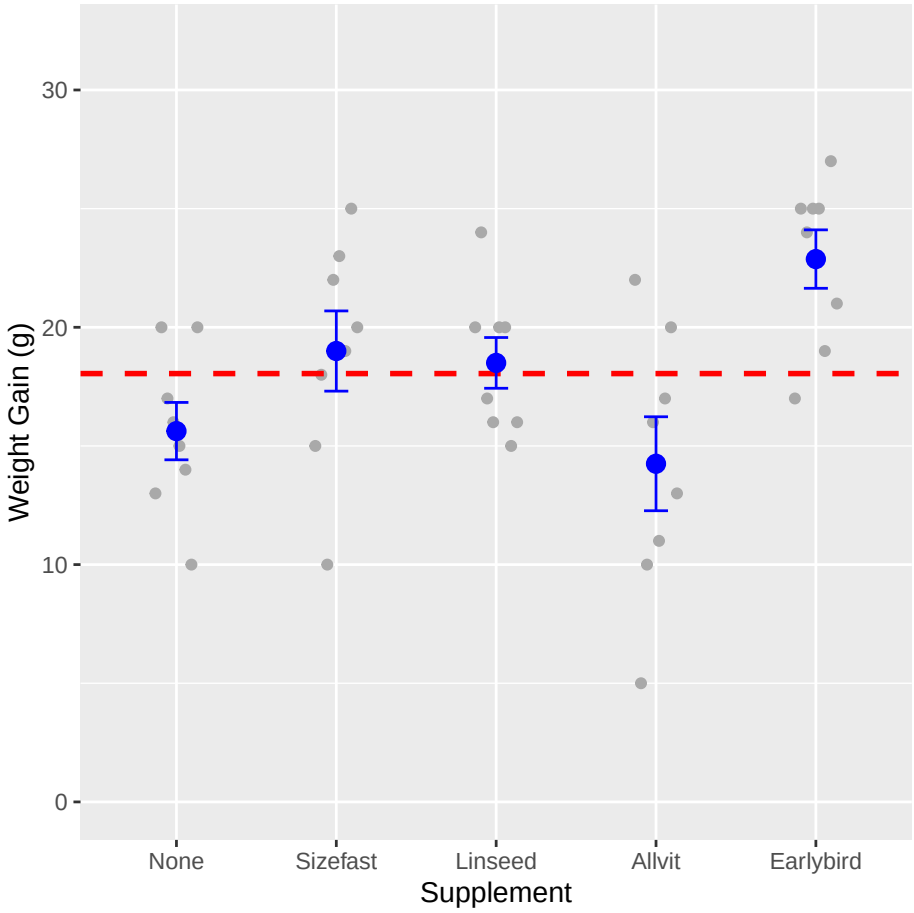


Figure 6.1: Plot of weight gain in corncrake hatchlings reared on four different nutritional supplements and a control group. The mean and standard deviation of each group is plotted in blue. the mean weight gain across the entire sample is plotted in red.

```
corncrake.model <- lm(WeightGain ~ Supplement, data = corncrake)
anova(corncrake.model)
```

```
Analysis of Variance Table

Response: WeightGain
           Df Sum Sq Mean Sq F value   Pr(>F)
Supplement  4 357.65  89.413  5.1281 0.002331 **
Residuals  35 610.25  17.436
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The one-way ANOVA shows that there was a significant effect of supplement on the weight gain of the corncrake hatchlings ($F = 5.1$, $df = 4, 35$, $p < 0.01$). The final step is to perform our multiple comparisons test.

```
corncrake.aov <- aov(corncrake.model)
TukeyHSD(corncrake.aov, ordered = TRUE)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level
    factor levels have been ordered

Fit: aov(formula = corncrake.model)

$Supplement
                     diff       lwr       upr     p adj
None-Allvit         1.375 -4.627565  7.377565 0.9638453
Linseed-Allvit      4.250 -1.752565 10.252565 0.2707790
Sizefast-Allvit     4.750 -1.252565 10.752565 0.1771593
Earlybird-Allvit    8.625  2.622435 14.627565 0.0018764
Linseed-None        2.875 -3.127565  8.877565 0.6459410
Sizefast-None       3.375 -2.627565  9.377565 0.4971994
Earlybird-None      7.250  1.247435 13.252565 0.0113786
Sizefast-Linseed    0.500 -5.502565  6.502565 0.9992352
Earlybird-Linseed   4.375 -1.627565 10.377565 0.2447264
Earlybird-Sizefast 3.875 -2.127565  9.877565 0.3592201
```

The **TukeyHSD** functions shows us the pairwise comparisons between groups. We can see (for example) that *Allvit* was not significantly different from the control (difference = 1.375g, $p = 0.96$) but *Earlybird* was significantly better than the control group (difference = 7.25g, $p = 0.01$).

6.2 Phylogenetic correction

As you know, when trying to run a similar analysis on non-independent data (such as species) we will run into problems. Garland *et al* [1993] developed a simulation based approach to solve this problem. The phylogenetic ANOVA uses computer simulations of traits evolving the phylogenetic tree. The next section contains some example data and a phylogeny to demonstrate the method.

6.3 Example data & analysis

The data we're using is taken from the package geiger [Harmon et al., 2008] so make sure the package is installed and loaded.

```
install.packages("geiger")
library(geiger)
```

Load the data as follows. The tree and data are stored together so we'll need to save the to separate objects called **dat** and **phy**. You probably don't *need* to do this but this is more similar to what you're likely to see when using your own data.

```
data("geospiza")
dat <- geospiza$dat
tree <- geospiza$phy
head(dat)
```

```
               wingL  tarsusL  culmenL    beakD   gonysW
magnirostris 4.404200 3.038950 2.724667 2.823767 2.675983
conirostris  4.349867 2.984200 2.654400 2.513800 2.360167
difficilis   4.224067 2.898917 2.277183 2.011100 1.929983
scandens     4.261222 2.929033 2.621789 2.144700 2.036944
fortis       4.244008 2.894717 2.407025 2.362658 2.221867
fuliginosa   4.132957 2.806514 2.094971 1.941157 1.845379
```

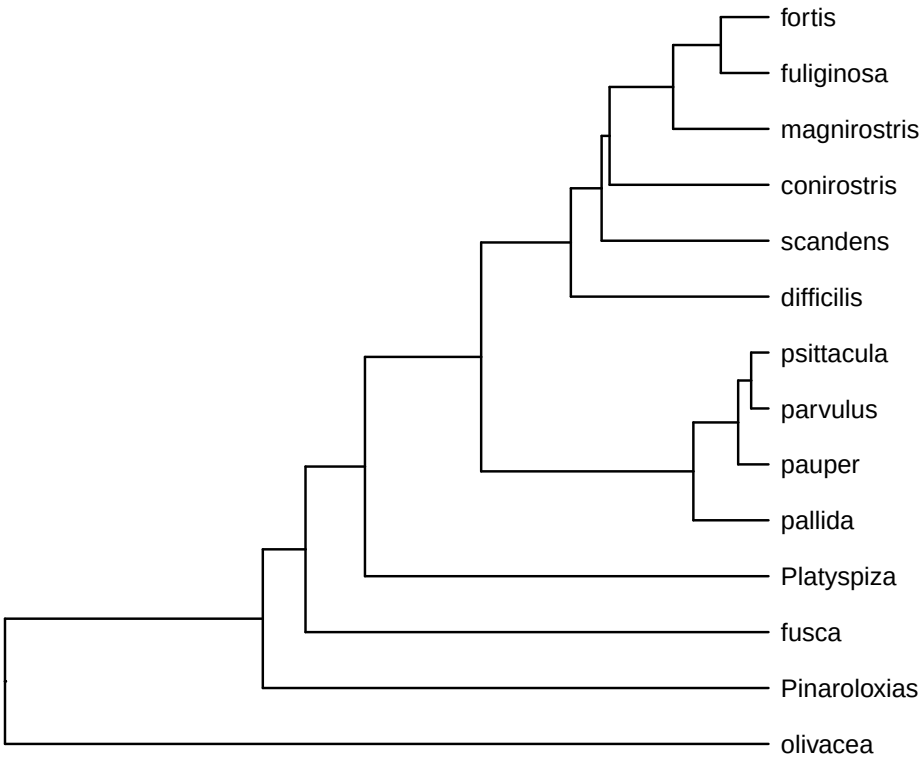


Figure 6.2: Phylogeny of the species contained within the 'geospiza' dataset of the package geiger.

We need to start by defining the categories for the data. It is likely that you will have already done this in your data frame. If so, just make sure the groups are stored as a factor. In this case, we'll just create some random categories to work with for the example.

```
groups <- as.factor(c(rep("A", 4), rep("B", 5), rep("C", 4)))
names(groups) <- rownames(dat)
```

An important step here (and for every phylogenetic analysis) is making sure the tree and data can be compared. To do this, we should make sure that the rownames of the data are species names and not just numbers. In this case they already are but if they aren't for you data, you can use the following code.

```
rownames(data) <- data$SPECIES #the column with species names in the data
```

The geiger package has a very useful function called **name.check** to allow us to check that the rownames of our data match the tip labels of our tree.

```
name.check(tree, dat)
```

```
$tree_not_data
[1] "olivacea"

$data_not_tree
character(0)
```

We can see that *olivacea* is not in our data. For some analyses, mismatches like this are a problem and you will need to drop the tip from the tree. It actually doesn't matter here because the function we will be using can drop it automatically for us. However, let's see how it's done. Note the use of the function **drop.tip** from the package **ape** [Paradis and Schliep, 2018] which is an essential package to have for this kind of work!

```
tree <- ape::drop.tip(tree, tip = "olivacea")
```

Now we have overwritten the old tree with our pruned tree. Let's check the new one matches the data.

```
name.check(tree, dat)
```

```
[1] "OK"
```

All that's left now is to run the analysis. First we extract the column of interest from our data and then simply use the function **aov.phylo**.

```
d1 <- dat[,1]
```

You should notice some similarities and differences from the way you have run ANOVA before. We are still using a formula (the part with $\sim$) but not in a separate **lm** function. We need to specify the tree we want to use (**tree**) and also how many simulations we want to run. There isn't a firm rule about this but general convention is around 1000 when sampling/bootstrapping/simulations are involved.

```
x <- aov.phylo(d1 ~ groups, phy = tree, nsim = 1000)
```

```
Analysis of Variance Table

Response: dat
          Df   Sum-Sq  Mean-Sq F-value  Pr(>F) Pr(>F) given phy
group      2 0.063237 0.031619  3.0067 0.09497           0.1658
Residuals 10 0.105161 0.010516
```

The results table should be very familiar! The only real difference here is that you have been provided with two p-values. The first (*Pr(>F)*) is the p-value without accounting for phylogeny and the second (*Pr(>F) given phy*) is the value when we account for phylogeny. In both cases, there is no significant difference between groups.

As you can see, accounting for phylogeny *usually* raises the p-value (makes it less significant). This shows us that not accounting for phylogeny increases the risk of type I errors (false positives).

6.4 Further info

For further information about the phylogenetic ANOVA, you can read the original paper by Garland *et al* [1993].

Chapter 7

Ancestral State Reconstruction I

This chapter will take you through the code we can use to run ancestral state reconstruction with **categorical** characters. As always, remember to begin by setting your working directory to wherever you have saved the data files.

7.1 Data

The first thing we need to do is load some data. When you're doing this, you need to keep in mind that you should keep your workspace as well organised as possible. In practice, this means giving things good names. "RicksDataV1.1" is not a great name depending on how many datasets you want in there. Neither is "data1" if you plan on having multiple datasets (which we do). So give your data object, and all other objects, simple, useful names. My personal preference is to use the name of the group but whatever works is fine. You need to be able to keep track of everything.

```
macaques <- read.table("macaque_data.txt", header = TRUE)
```

In your environment panel there should be a data frame with 16 observations of 2 variables. This command will show us the top 6 rows of data. It's helpful to have a quick look and see R has loaded what we expected. In this case our data contains 15 species of macaque and one species of baboon alongside data regarding whether they exhibit sexual swellings or not (1/0).

```
head(macaques)
```

```
              species swelling
1    Macaca_arctoides        0
```

```
2   Macaca_assamensis        0
3     Macaca_cyclopis        1
4 Macaca_fascicularis        1
5      Macaca_fuscata        1
6        Macaca_maura        1
```

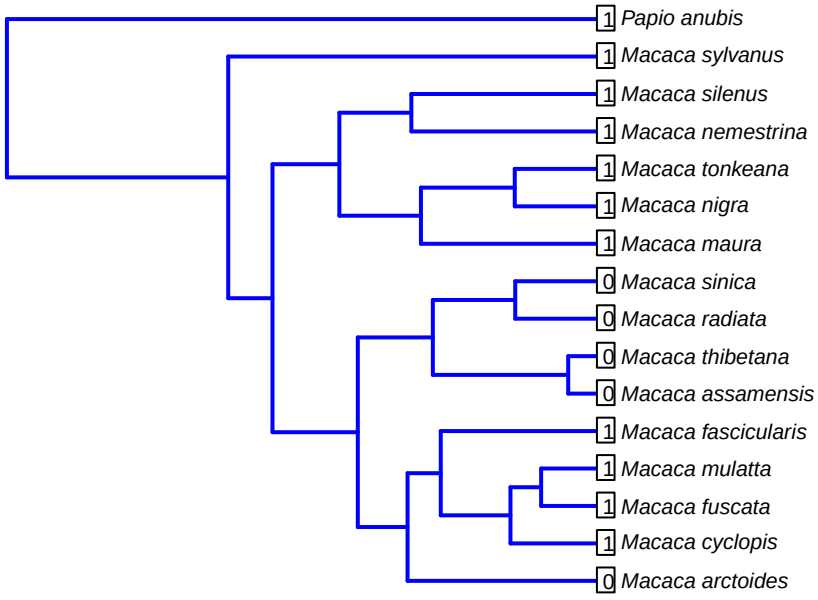
7.2 Trees

Now we need to load the tree using the **read.nexus** function in the package **ape** [Paradis and Schliep, 2018].

```
macaque.tree <- read.nexus("macaque_tree.nex")
```

Let's plot the tree to make sure it loaded correctly. I've used base graphics here rather than ggtree (annotated to let you know what it does). Feel free to have a mess around with these options so you get a feel for what they do. The second function "tiplabels" adds some extra tip labels containing the data from the second column of our macaque data.

```
plot(macaque.tree,          #Tree object
     cex = 0.7,             #Font size for tip labels
     label.offset = 0.3,    #Create a space between tip and label
     edge.color = "blue",   #Paint the branches blue
     edge.width = 2)        #Make the branches thicker
tiplabels(macaques[,2], bg = "white", cex = 0.7)
```



7.3 Parsimony

Let's first generate the most parsimonious reconstruction of the history of this trait. Remember that the most parsimonious history is the one that has the fewest evolutionary transitions. Parsimony is conceptually based upon Occam's razor which states that all else being equal, the simplest explanantion is always the correct one.

The function for this is **MPR**. It takes an unrooted tree and asks you to specify the root. In our case, we'll have to unroot our tree and then re-root it by specifying that *Papio anubis* is our outgroup.

```
mp1 <- MPR(macaques[,2], unroot(macaque.tree), "Papio_anubis")
```

When we investigate mp1, we can see a list of results matched up to numbered nodes on the tree. Some nodes are clearly in state 1 and others in state 0. Interestingly some are indeterminate and could be either 0 or 1 such as nodes 19 and 20.

```
mp1
```

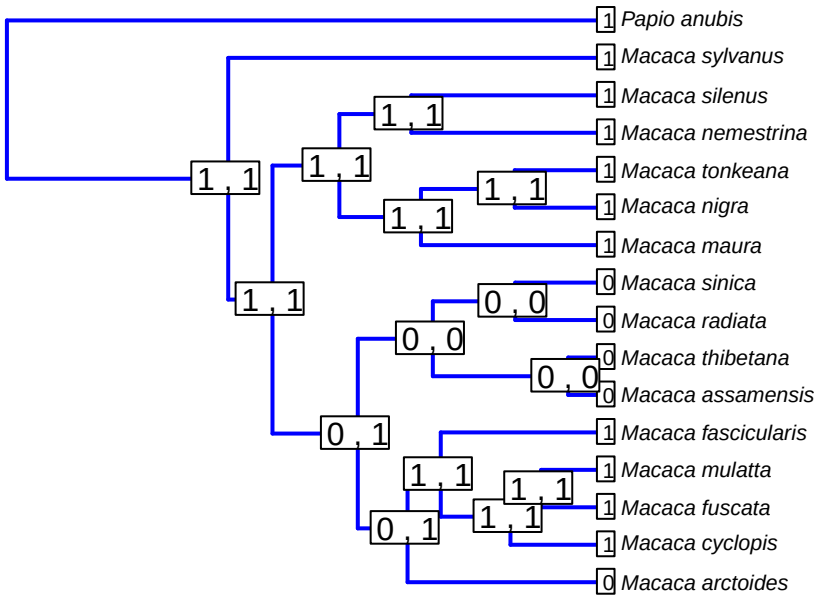
```
   lower upper
17     1     1
18     1     1
19     0     1
20     0     1
21     1     1
22     1     1
23     1     1
24     0     0
25     0     0
26     0     0
27     1     1
28     1     1
29     1     1
30     1     1
```

To get an idea of what this means, we should plot it on the tree. This loop cycles through our results list and combines the lower and upper estimates for each node into a text string that we can then overlay onto that node.

```
mp.nodes <- numeric(0)
for(i in 1:length(mp1[,1])){
  mp.nodes <- append(mp.nodes, paste(mp1[i,1], ",", mp1[i,2]))
}
```

Once we've done that we can plot those expressions onto the tree with the function nodelabels.

```
plot(macaque.tree, cex = 0.7, label.offset = 0.3,
     edge.color = "blue", edge.width = 2)
tiplabels(macaques[,2], bg = "white", cex = 0.7)
nodelabels(mp.nodes, c(18:31), bg = "white")
```



You should note that this isn't a very good plot! There are better ways to represent this information with a little code manipulation. Here's a version using **ggtree** that plots the character states as points on the tips and the reconstructed nodes.

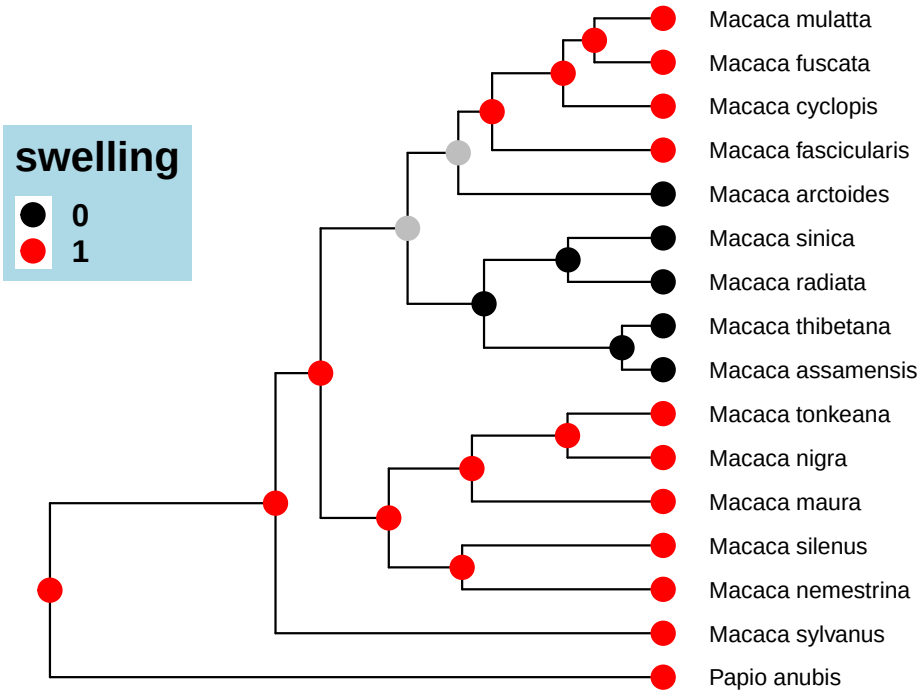


Figure 7.1: Maximum parsimony reconstruction of the evolution of conspicuous sexual swellings in macaques

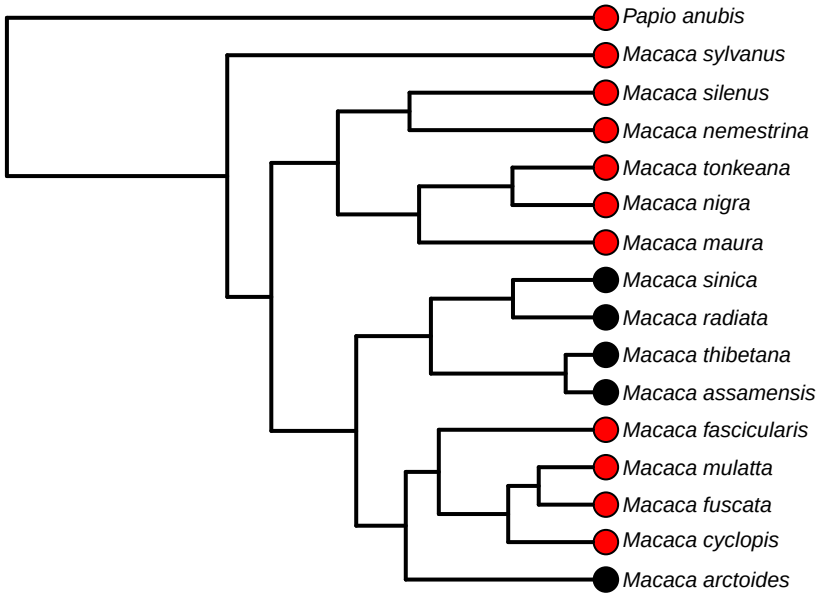
As you can see, the uncertainty in some nodes comes from the fact that there seems to be at least two equally parsimonious histories with gains and losses ocurring in different places. For any serious analysis, this is a highly unsatisfactory outcome!

7.4 Maximum Likelihood

Let's try a different approach. Maximum likelihood is different from parsimony for many reasons but most significantly, it can make use of branch length information. This is very useful in discriminating between possible histories. A longer branch means more evolutionary change (either in time or character change) and so transitions are more likely to occur on longer branches.

Let's replot the tree. Here I've changed the tiplabels function to plot the character states as colours rather than numbers. The **bg** argument is what lets me do this. In this argument I list the states (adding 1 because the first is 0) and then the function passes those states to R to assign colours based on a numbered list of standard colours.

```
plot(macaque.tree, cex = 0.7, label.offset = 0.4, edge.width = 2)
tiplabels(pch = 21, bg = as.numeric(macaques$swelling)+1, cex = 1.7)
```



To run an ancestral state reconstruction using maximum likelihood we can use the function **ace** (ancestral character estimation) in the ape package [Paradis and Schliep, 2018]. In our first reconstruction, we will make the assumption that the rate of evolution of the trait is equal across the tree by setting the model to *ER* (equal rates).

```
m1 <- ace(x = macaques$swelling,  #trait data
          phy = macaque.tree,     #phylogeny
          method = "ML",          #method (Maximum likelihood)
          type = "discrete",      #type of data (continuous or discrete)
          model = "ER")           #Model of evolution
m1
```

```
    Ancestral Character Estimation

Call: ace(x = macaques$swelling, phy = macaque.tree, type = "discrete",
    method = "ML", model = "ER")

    Log-likelihood: -6.906593

Rate index matrix:
  0 1
0 . 1
1 1 .
```

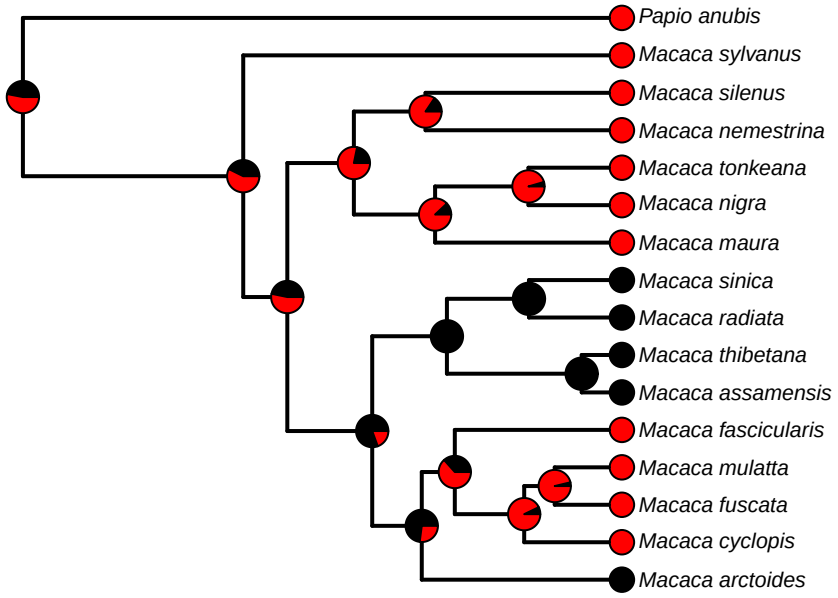
```
Parameter estimates:
 rate index estimate std-err
          1   0.0319  0.0191

Scaled likelihoods at the root (type '...$lik.anc' to get them for all nodes):
         0          1
0.08625654 0.91374346
```

Looking at the results shows us the likelihood at the root (91% in favour of state 1 here). However, it's always best to plot the results. We can represent the likelihoods at each node with a piechart. Generally speaking, piecharts are awful but when used in this way, they can actually add useful information to a plot and that's the most important point about plotting any data. In this plot, the piecharts represent the probability that each node exhibited sexual swelling (red) or concealed estrus (black). We can see that the two uncertain nodes from our parsimony analysis are now more certain. Visual inspection shows that these nodes have a greater tha 75% probability of having exhibited sexual swellings.

```
plot(macaque.tree, cex = 0.7, label.offset = 0.4, edge.width = 2)
tiplabels(pch = 21, bg = as.numeric(macaques$swelling)+1, cex = 1.7)
nodelabels(pie = m1$lik.anc, piecol = c("black", "red"), cex = 0.8)
```

Now we can run a similar analysis but let's assume that rates of evolution can vary by setting model to **ARD** (All Rates Different).

```
m2 <- ace(x = macaques$swelling, phy = macaque.tree,
          method = "ML", type = "discrete", model = "ARD")

plot(macaque.tree, cex = 0.7, label.offset = 0.4, edge.width = 2)
tiplabels(pch = 21, bg = as.numeric(macaques$swelling)+1, cex = 1.7)
nodelabels(pie = m2$lik.anc, piecol = c("black", "red"), cex = 0.8)
```



As you can see, the different model of evolution makes a big difference to the results. Which model you choose to use depends on which assumptions you think are justified. Is it fair to assume that the rate of evolution of conspicuous sexual swelling would be constant across the tree as in the equal rates model?

7.5 Stochastic Character Mapping

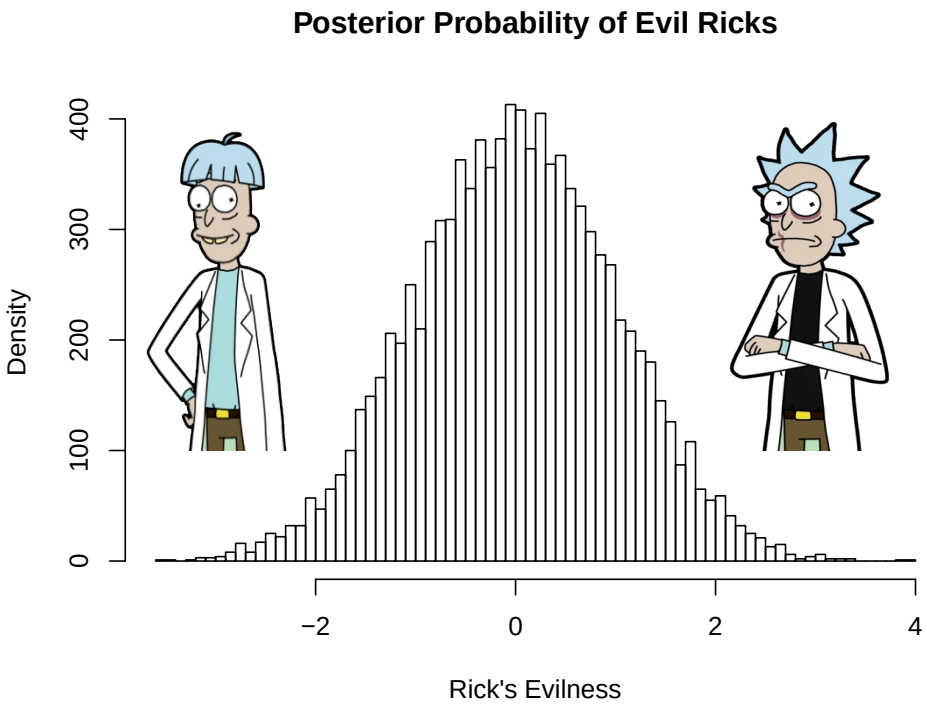
Stochastic character mapping uses an **MCMC** (Markov chain Monte-Carlo) approach to sample possible reconstructions from a posterior probability distribution.

Think of the posterior probability distribution as containing all the possible evolutionary histories of the trait in question. This includes some histories in which everything was in one state right up until a few generations from the present when everything swapped around at the same time to give us the distribution we see today. It also contains a history in which the trait switches between 0 and 1 every other generation essentially at random.

Obviously these kind of histories are biologically absurd but not mathematically impossible. They have low statistical probability. Certain other histories will have a high statistical probability and so there will be many similar histories in the distribution. The distribution can be thought of as a histogram with some parameter that defines each particular history.

7.5.1 An Analogy

Let's say that we were to plot the entire multiverse as such a distribution using the evil tendencies of one particular occupant (Rick Sanchez) of the multiverse as our parameter. All the different Ricks in all the different universes will vary in their evil tendencies. But overall, Rick's character is actually a nihilist meaning his mean evilness is around 0 when taken over the whole multiverse. Given all this, the posterior distribution of evil Ricks in the multiverse might look like this.



MCMC samples this distribution of histories in a chain. If a history has a higher likelihood than the previous sampling, it is accepted. If it is lower then it is rejected from the sample. In this way, MCMC quickly narrows down the possibilities and gives us a sample of quite likely histories.

7.5.2 2-State Characters

Let's see it in action. We'll need the **phytools** package [Revell, 2012] to create our stochastic character map.

```
library(phytools)
```

For this analysis (like other phytools functions) we'll need our data in a named vector rather than a data table. Let's call it swelling. The **names** function attaches the species name to each value in our new vector.

```
swelling <- macaques$swelling
names(swelling) <- macaques$species
swelling
```

```
   Macaca_arctoides   Macaca_assamensis     Macaca_cyclopis Macaca_fascicularis
                  0                   0                   1                   1
     Macaca_fuscata       Macaca_maura      Macaca_mulatta   Macaca_nemestrina
                  1                   1                   1                   1
       Macaca_nigra     Macaca_radiata      Macaca_silenus       Macaca_sinica
                  1                   0                   1                   0
    Macaca_sylvanus    Macaca_thibetana     Macaca_tonkeana        Papio_anubis
                  1                   0                   1                   1
```

Now we can sample character histories assuming an *equal rates* model of evolution using the **make.simmap** function.

```
scm1 <- make.simmap(macaque.tree, x = swelling, model = "ER")
```

```
make.simmap is sampling character histories conditioned on the transition matrix

Q =
            0           1
0 -0.03185011  0.03185011
1  0.03185011 -0.03185011
(estimated using likelihood);
and (mean) root node prior probabilities
pi =
  0   1
0.5 0.5

Done.
```

Q here is the matrix of transition rates which we have constrained to be equal (model = "ER") which explains why the numbers match. As usual with reconstructions, the best thing is to plot them. Here we can use the phytools function **plotSimmap** to plot the special object we've created. It even has a companion function to add a legend. The first line here assigns colours to the traits.

```
cols <- setNames(c("black", "red"), sort(unique(swelling)))
plotSimmap(scm1, cols, pts = F, lwd = 3, fsize = .8)
add.simmap.legend(colors = cols, vertical = F, prompt = F, x = 0, y = 10, fsize = .8)
```

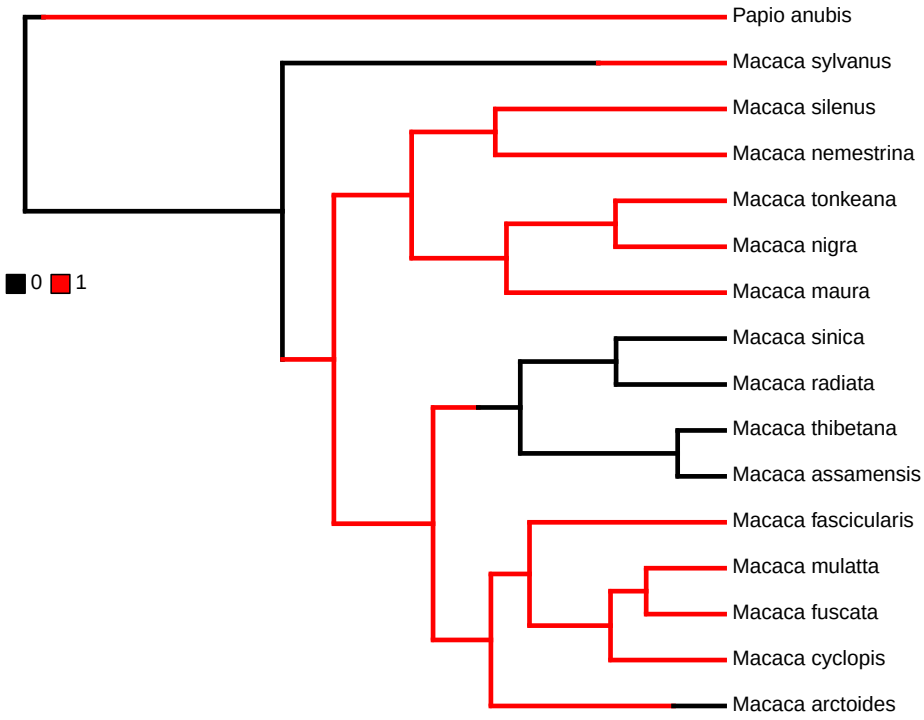


Figure 7.2: Simmap showing a single possible evolutionary history of sexual swelling in macaques.

Here you can see the single history we have sampled (yours will likely differ). The history contains branches painted according to the trait colour we specified and the position of the transitions on the branch mark the exact position the changes are theorised to have taken place. This is an awful lot of certainty for an ancestral state reconstruction! You should note that the one plotted here is very odd. It says that the ancestor of the group had concealed estrus and then this trait was lost 3 times independently, leaving no trace in the extant species. Given the data and tree we provided, it is hard to see how we can have any confidence in this reconstruction. What evidence have we collected that actually supports this?

However, we need to remember that this only one of the many possible histories! Our next step should be to extract a reasonable sample of these histories!

Let's sample 500 and when R has done that, we can use **describe.simmap** to summarize the sample.

```
scm2 <- make.simmap(macaque.tree, swelling, model = "ER", nsim = 500)
```

```
make.simmap is sampling character histories conditioned on the transition matrix
```

```
Q =
            0           1
0 -0.03185011  0.03185011
1  0.03185011 -0.03185011
(estimated using likelihood);
and (mean) root node prior probabilities
pi =
  0   1
0.5 0.5

Done.
```

```
scm2.sum <- describe.simmap(scm2, plot = FALSE)
```

When we call up the summary, we can see some interesting details about our sample. It seems to be saying that transitions from 1 to 0 (a loss of sexual swelling) happen more frequently than gains of sexual swelling.

```
scm2.sum
```

```
500 trees with a mapped discrete character with states:
 0, 1

trees have 2.872 changes between states on average

changes are of the following types:
       0,1   1,0
x->y 0.758 2.114

mean total time spent in each state is:
              0          1   total
raw  17.6835904 71.4988096 89.1824
prop  0.1982857  0.8017143  1.0000
```

As usual, we're going to want a summary plot. The backbone of this plot won't look quite the same as the previous one. You don't want confusing information on your plot so here it would be better to plot a blank backbone (ie a tree with just one colour of branch that doesn't match the colour of the traits) and represent the trait transitions as we did previously with pie charts. In this case the pies represent the proportion of histories in each state (1 or 0) at each node.

```
cols.null <- setNames(c("darkgrey", "darkgrey"), sort(unique(swelling)))
plotSimmap(scm2[[1]], lwd = 3, pts = F, setEnv = T, colors = cols.null, offset = .6)
nodelabels(pie = scm2.sum$ace, piecol = cols, cex = 0.6)
add.simmap.legend(colors = cols, vertical = F, prompt = F, x = 0, y = 10, fsize = .8)
tiplabels(pch = 21, bg = as.numeric(macaques$swelling)+1, cex = 2)
```

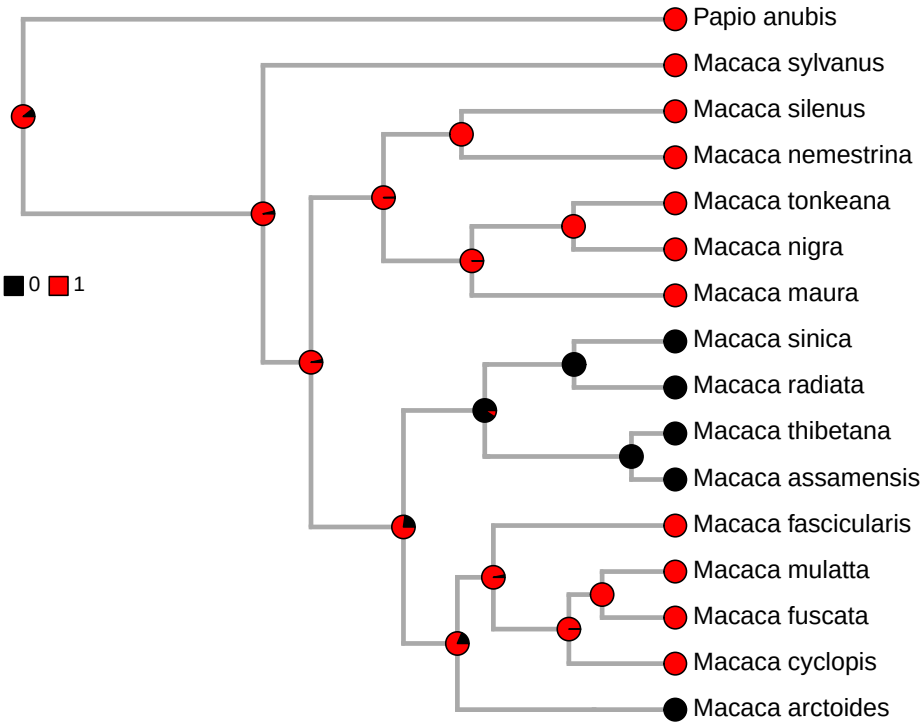


Figure 7.3: Summary of 500 sampled discrete character histories showing the evolution of sexual swellings in macaques.

This analysis gives us a very similar output to the maximum likelihood analysis in the previous section. If you're intrested, give this analysis another try with different models of evolution.

7.5.3 3-State Characters

Stochastic character mapping can also be used for traits with more than one state. For example, burrowing in carnivores can be classified as 0 (no burrowing), 1 (use a burrow dug by another animal) or 2 (dig your own burrow).

7.5.3.1 Data

Let's load some data from a paper which investigated aposematism in terrestrial carnivores [Stankowich et al., 2011]. Don't forget to assign the species names to rownames to keep everything tidy while we manipulate the data. We also have a tree covering all carnivores [Nyakatura and Bininda-Emonds, 2012].

```
carn.tree <- read.nexus("carnivores_tree.nex")
carn.data <- read.table("carnivores_data.txt", header = T)
rownames(carn.data) <- carn.data$Species
```

If you look at the new object **carn.tree** you'll notice it is a multiPhylo object. This means it actually contains a number of trees rather than just one. For more details about this class of object, see chapter 3.

For now, we just want the first one in the list (based on the best estimates used to date the tree). I'll also prune it a bit to get rid of some of the species I'm not interested in for now.

```
carn.tree <- carn.tree[[1]]
carn.tree <- extract.clade(carn.tree, node = "'123'")
```

Unlike the macaque data from earlier, the carnivore data needs a little more tidying. Now that you're more comfortable using R, you should make this standard practice whenever you load data and a tree for an analysis!

We can use the function **name.check** in the package **geiger** to help us out here [Harmon et al., 2008]. This function returns two lists. The first contains all the species that appear in the phylogeny but not in the dataset. The second has the species that occur in the data but not in the tree.

```
geiger::name.check(phy = carn.tree, data = carn.data)
```

```
$tree_not_data
 [1] "Arctocephalus_australis"     "Arctocephalus_forsteri"
 [3] "Arctocephalus_galapagoensis" "Arctocephalus_gazella"
 [5] "Arctocephalus_philippii"     "Arctocephalus_pusillus"
 [7] "Arctocephalus_townsendi"     "Arctocephalus_tropicalis"
 [9] "Bassaricyon_alleni"          "Bassaricyon_beddardi"
[11] "Bassaricyon_lasius"          "Bassaricyon_pauli"
[13] "Callorhinus_ursinus"         "Conepatus_chinga"
[15] "Conepatus_humboldtii"        "Conepatus_semistriatus"
[17] "Cystophora_cristata"         "Dusicyon_australis"
[19] "Erignathus_barbatus"         "Eumetopias_jubatus"
[21] "Halichoerus_grypus"          "Histriophoca_fasciata"
[23] "Hydrurga_leptonyx"           "Leptonychotes_weddellii"
[25] "Lobodon_carcinophaga"        "Lontra_provocax"
[27] "Lutra_nippon"                "Lutra_sumatrana"
[29] "Lycalopex_fulvipes"          "Lycalopex_griseus"
[31] "Lycalopex_gymnocercus"       "Lycalopex_sechurae"
[33] "Lyncodon_patagonicus"        "Martes_gwatkinsii"
[35] "Meles_anakuma"               "Meles_leucurus"
[37] "Melogale_everetti"           "Melogale_orientalis"
[39] "Melogale_personata"          "Mirounga_angustirostris"
[41] "Mirounga_leonina"            "Monachus_monachus"
```

```
[43] "Monachus_schauinslandi"      "Monachus_tropicalis"
[45] "Mustela_felipei"             "Mustela_itatsi"
[47] "Mustela_kathiah"             "Mustela_lutreolina"
[49] "Mustela_nudipes"             "Mustela_strigidorsa"
[51] "Mustela_subpalmata"          "Nasuella_olivacea"
[53] "Neophoca_cinerea"            "Neovison_macrodon"
[55] "Odobenus_rosmarus"           "Ommatophoca_rossii"
[57] "Otaria_flavescens"           "Pagophilus_groenlandicus"
[59] "Phoca_largha"                "Phoca_vitulina"
[61] "Phocarctos_hookeri"          "Procyon_pygmaeus"
[63] "Pusa_caspica"                "Pusa_hispida"
[65] "Pusa_sibirica"               "Spilogale_angustifrons"
[67] "Urocyon_littoralis"          "Vulpes_bengalensis"
[69] "Vulpes_ferrilata"            "Zalophus_californianus"
[71] "Zalophus_japonicus"          "Zalophus_wollebaeki"

$data_not_tree
 [1] "Acinonyx_jubatus"             "Arctictis_binturong"
 [3] "Arctogalidia_trivirgata"      "Atilax_paludinosus"
 [5] "Bdeogale_crassicauda"         "Caracal_caracal"
 [7] "Catopuma_temminckii"          "Chrotogale_owstoni"
 [9] "Civettictis_civetta"          "Crocuta_crocuta"
[11] "Crossarchus_obscurus"         "Cryptoprocta_ferox"
[13] "Cynictis_penicillata"         "Cynogale_bennettii"
[15] "Dologale_dybowskii"           "Eupleres_goudotii"
[17] "Felis_chaus"                  "Felis_manul"
[19] "Felis_margarita"              "Felis_nigripes"
[21] "Felis_silvestris"             "Fossa_fossana"
[23] "Galerella_sanguinea"          "Galidia_elegans"
[25] "Genetta_abyssinica"           "Genetta_angolensis"
[27] "Genetta_genetta"              "Genetta_servalina"
[29] "Genetta_thierryi"             "Helogale_parvula"
[31] "Hemigalus_derbyanus"          "Herpestes_ichneumon"
[33] "Herpestes_javanicus"          "Herpestes_urva"
[35] "Hyaena_brunnea"               "Hyaena_hyaena"
[37] "Ichneumia_albicauda"          "Leopardus_geoffroyi"
[39] "Leopardus_guigna"             "Leopardus_jacobitus"
[41] "Leopardus_pardalis"           "Leopardus_wiedii"
[43] "Leptailurus_serval"           "Liberiictis_kuhni"
[45] "Lynx_canadensis"              "Lynx_lynx"
[47] "Lynx_pardinus"                "Lynx_rufus"
[49] "Macrogalidia_musschenbroekii" "Mungos_gambianus"
[51] "Mungos_mungo"                 "Mungotictis_decemlineata"
[53] "Nandinia_binotata"            "Neofelis_nebulosa"
[55] "Paguma_larvata"               "Panthera_leo"
[57] "Panthera_onca"                "Panthera_pardus"
```

```
[59] "Panthera_tigris"              "Paracynictis_selousi"
[61] "Paradoxurus_hermaphroditus"   "Paradoxurus_zeylonensis"
[63] "Pardofelis_marmorata"         "Poiana_richardsonii"
[65] "Prionailurus_bengalensis"     "Prionailurus_iriomotensis"
[67] "Prionailurus_rubiginosus"     "Prionodon_linsang"
[69] "Prionodon_pardicolor"         "Proteles_cristata"
[71] "Puma_concolor"                "Salanoia_concolor"
[73] "Suricata_suricatta"           "Uncia_uncia"
[75] "Viverra_megaspila"            "Viverra_tangalunga"
[77] "Viverra_zibetha"              "Viverricula_indica"
```

The easiest thing to do first is drop the tips from the tree that we're not interested in. We can pass the whole list to the **drop.tip** function in **ape** for this [Paradis and Schliep, 2018].

```
carn.tree <- drop.tip(carn.tree, geiger::name.check(carn.tree, carn.data)$tree_not_data
geiger::name.check(carn.tree, carn.data)
```

```
$tree_not_data
character(0)

$data_not_tree
 [1] "Acinonyx_jubatus"              "Arctictis_binturong"
 [3] "Arctogalidia_trivirgata"       "Atilax_paludinosus"
 [5] "Bdeogale_crassicauda"          "Caracal_caracal"
 [7] "Catopuma_temminckii"           "Chrotogale_owstoni"
 [9] "Civettictis_civetta"           "Crocuta_crocuta"
[11] "Crossarchus_obscurus"          "Cryptoprocta_ferox"
[13] "Cynictis_penicillata"          "Cynogale_bennettii"
[15] "Dologale_dybowskii"            "Eupleres_goudotii"
[17] "Felis_chaus"                   "Felis_manul"
[19] "Felis_margarita"               "Felis_nigripes"
[21] "Felis_silvestris"              "Fossa_fossana"
[23] "Galerella_sanguinea"           "Galidia_elegans"
[25] "Genetta_abyssinica"            "Genetta_angolensis"
[27] "Genetta_genetta"               "Genetta_servalina"
[29] "Genetta_thierryi"              "Helogale_parvula"
[31] "Hemigalus_derbyanus"           "Herpestes_ichneumon"
[33] "Herpestes_javanicus"           "Herpestes_urva"
[35] "Hyaena_brunnea"                "Hyaena_hyaena"
[37] "Ichneumia_albicauda"           "Leopardus_geoffroyi"
[39] "Leopardus_guigna"              "Leopardus_jacobitus"
[41] "Leopardus_pardalis"            "Leopardus_wiedii"
[43] "Leptailurus_serval"            "Liberiictis_kuhni"
[45] "Lynx_canadensis"               "Lynx_lynx"
[47] "Lynx_pardinus"                 "Lynx_rufus"
[49] "Macrogalidia_musschenbroekii" "Mungos_gambianus"
```

```
[51] "Mungos_mungo"                  "Mungotictis_decemlineata"
[53] "Nandinia_binotata"             "Neofelis_nebulosa"
[55] "Paguma_larvata"                "Panthera_leo"
[57] "Panthera_onca"                 "Panthera_pardus"
[59] "Panthera_tigris"               "Paracynictis_selousi"
[61] "Paradoxurus_hermaphroditus"    "Paradoxurus_zeylonensis"
[63] "Pardofelis_marmorata"          "Poiana_richardsonii"
[65] "Prionailurus_bengalensis"      "Prionailurus_iriomotensis"
[67] "Prionailurus_rubiginosus"      "Prionodon_linsang"
[69] "Prionodon_pardicolor"          "Proteles_cristata"
[71] "Puma_concolor"                 "Salanoia_concolor"
[73] "Suricata_suricatta"            "Uncia_uncia"
[75] "Viverra_megaspila"             "Viverra_tangalunga"
[77] "Viverra_zibetha"               "Viverricula_indica"
```

Dropping species from your dataframe is a little more complex (and in truth not always necessary). One way of doing this is to create a **for loop** that will cycle through the list above and take a subset of the dataframe each time, removing the species in the list as it goes. There are better ways to do this but it might be helpful to become familiar with for loops which are a useful programming tool!

```
pruned.data <- carn.data
for(i in 1:length(geiger::name.check(carn.tree, carn.data)$data_not_tree)){
  pruned.data <- subset(pruned.data, Species!=geiger::name.check(carn.tree, carn.data)$data_not_t
}
geiger::name.check(carn.tree, pruned.data)
```

```
[1] "OK"
```

Once your tree and data are cleaned up we're ready to go!

7.5.3.2 Analysis

As before we need to create a named vector for analysis.

```
burrow<-pruned.data$Burrowing
names(burrow)<-pruned.data$Species
```

Now we can sample a single history and plot it, this time with three colours!

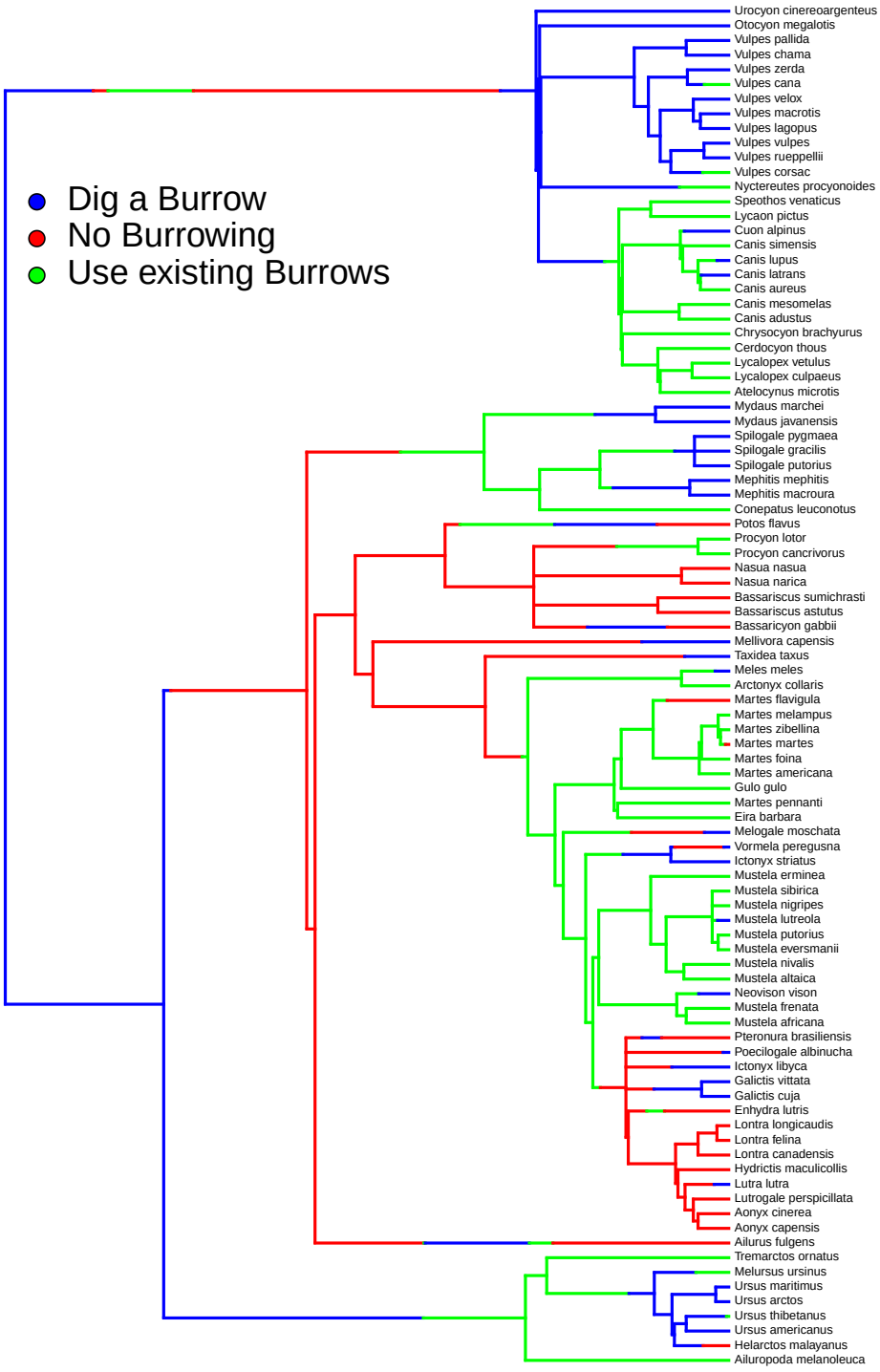
```
scm3<-make.simmap(carn.tree, burrow, model="ER")
```

```
make.simmap is sampling character histories conditioned on the transition matrix

Q =
                     Dig a Burrow No Burrowing Use existing Burrows
Dig a Burrow          -0.05640412   0.02820206           0.02820206
```

```
No Burrowing           0.02820206  -0.05640412           0.02820206
Use existing Burrows   0.02820206   0.02820206          -0.05640412
(estimated using likelihood);
and (mean) root node prior probabilities
pi =
        Dig a Burrow         No Burrowing Use existing Burrows
           0.3333333            0.3333333            0.3333333
```

```
Done.
```

```
cols <- setNames(c("blue", "red", "green"), sort(unique(burrow)))
plotSimmap(scm3, cols, pts = FALSE, lwd = 2, fsize = 0.5)
add.simmap.legend(colors = cols, vertical = TRUE, prompt = FALSE, x = 2, y = 80, fsize
```


Dig a Burrow
No Burrowing
Use existing Burrows
Urocyon cinereoargenteus
Otocyon megalotis
Vulpes pallida
Vulpes chama
Vulpes zerda
Vulpes cana
Vulpes velox
Vulpes macrotis
Vulpes lagopus
Vulpes vulpes
Vulpes rueppellii
Vulpes corsac
Nyctereutes procyonoides
Speothos venaticus
Lycaon pictus
Cuon alpinus
Canis simensis
Canis lupus
Canis latrans
Canis aureus
Canis mesomelas
Canis adustus
Chrysocyon brachyurus
Cerdocyon thous
Lycalopex vetulus
Lycalopex culpaeus
Atelocynus microtis
Mydaus marchei
Mydaus javanensis
Spilogale pygmaea
Spilogale gracilis
Spilogale putorius
Mephitis mephitis
Mephitis macroura
Conepatus leuconotus
Potos flavus
Procyon lotor
Procyon cancrivorus
Nasua nasua
Nasua narica
Bassariscus sumichrasti
Bassariscus astutus
Bassaricyon gabbii
Mellivora capensis
Taxidea taxus
Meles meles
Arctonyx collaris
Martes flavigula
Martes melampus
Martes zibellina
Martes martes
Martes foina
Martes americana
Gulo gulo
Martes pennanti
Eira barbara
Melogale moschata
Vormela peregusna
Ictonyx striatus
Mustela erminea
Mustela sibirica
Mustela nigripes
Mustela lutreola
Mustela putorius
Mustela eversmanii
Mustela nivalis
Mustela altaica
Neovison vison
Mustela frenata
Mustela africana
Pteronura brasiliensis
Poecilogale albinucha
Ictonyx libyca
Galictis vittata
Galictis cuja
Enhydra lutris
Lontra longicaudis
Lontra felina
Lontra canadensis
Hydrictis maculicollis
Lutra lutra
Lutrogale perspicillata
Aonyx cinerea
Aonyx capensis
Ailurus fulgens
Tremarctos ornatus
Melursus ursinus
Ursus maritimus
Ursus arctos
Ursus thibetanus
Ursus americanus
Helarctos malayanus
Ailuropoda melanoleuca

Let's sample 200 possible histories. This may take a few moments. For reports and publications, you should sample more than this. There's no hard rule but 1000 seems to be a good minimum for a proper analysis.

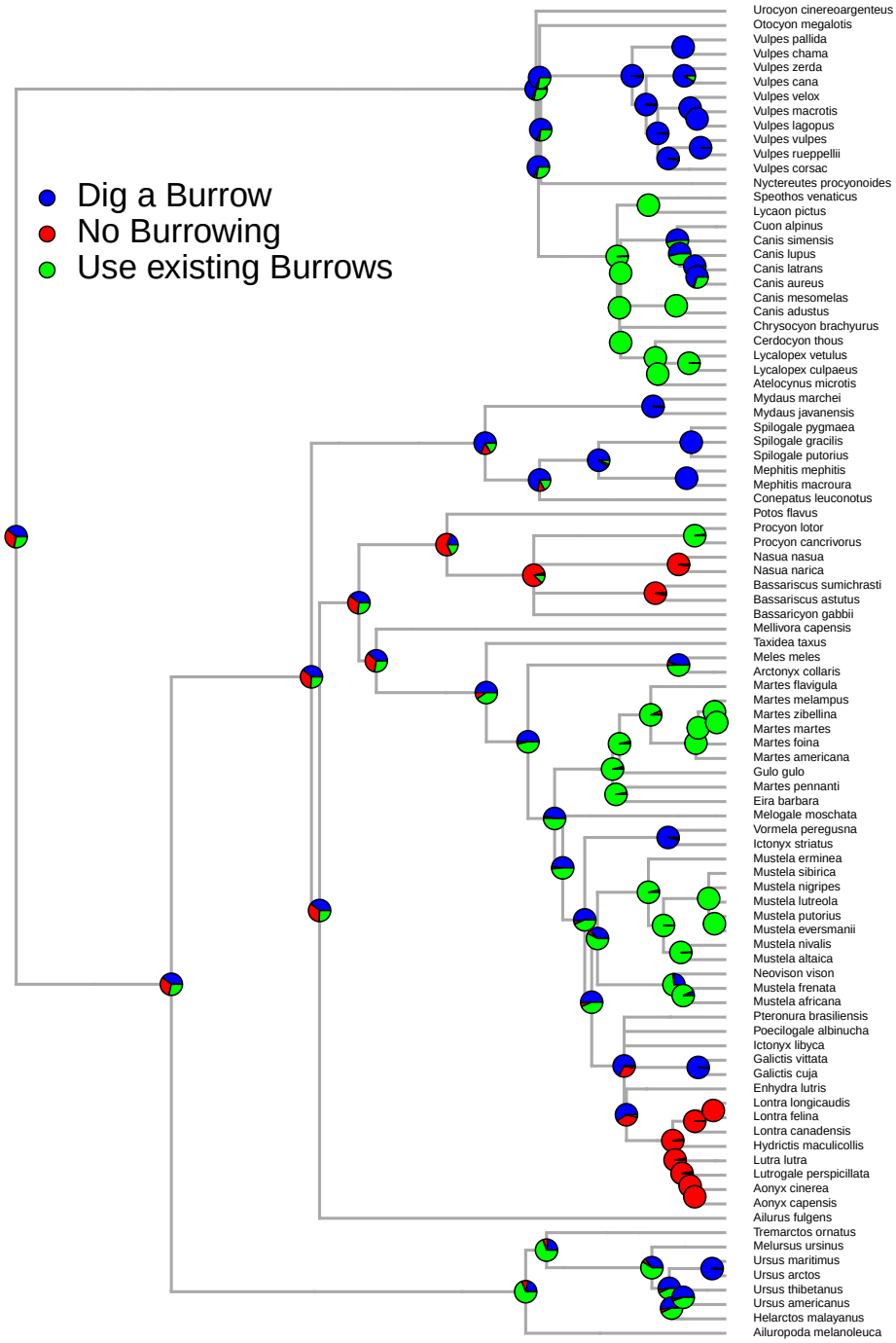
```
scm4 <- make.simmap(carn.tree, burrow, model = "ER", nsim = 200)
```

```
make.simmap is sampling character histories conditioned on the transition matrix

Q =
                     Dig a Burrow No Burrowing Use existing Burrows
Dig a Burrow          -0.05640412   0.02820206           0.02820206
No Burrowing           0.02820206  -0.05640412           0.02820206
Use existing Burrows   0.02820206   0.02820206          -0.05640412
(estimated using likelihood);
and (mean) root node prior probabilities
pi =
        Dig a Burrow         No Burrowing Use existing Burrows
           0.3333333            0.3333333            0.3333333

Done.
```

```
scm4.sum<-describe.simmap(scm4, plot = FALSE)
scm4.sum
```

```
200 trees with a mapped discrete character with states:
 Dig a Burrow, No Burrowing, Use existing Burrows

trees have 52 changes between states on average

changes are of the following types:
     Dig a Burrow,No Burrowing Dig a Burrow,Use existing Burrows
x->y                     7.26                            12.24
     No Burrowing,Dig a Burrow No Burrowing,Use existing Burrows
x->y                     6.925                            5.99
     Use existing Burrows,Dig a Burrow Use existing Burrows,No Burrowing
x->y                            12.265                             7.32

mean total time spent in each state is:
     Dig a Burrow No Burrowing Use existing Burrows total
raw   366.1971509  214.5972284          350.2056207   931
prop    0.3933374    0.2305019            0.3761607     1
```

Finally we can plot the summary of the analysis as before.


Dig a Burrow
No Burrowing
Use existing Burrows
Urocyon cinereoargenteus
Otocyon megalotis
Vulpes pallida
Vulpes chama
Vulpes zerda
Vulpes cana
Vulpes velox
Vulpes macrotis
Vulpes lagopus
Vulpes vulpes
Vulpes rueppellii
Vulpes corsac
Nyctereutes procyonoides
Speothos venaticus
Lycaon pictus
Cuon alpinus
Canis simensis
Canis lupus
Canis latrans
Canis aureus
Canis mesomelas
Canis adustus
Chrysocyon brachyurus
Cerdocyon thous
Lycalopex vetulus
Lycalopex culpaeus
Atelocynus microtis
Mydaus marchei
Mydaus javanensis
Spilogale pygmaea
Spilogale gracilis
Spilogale putorius
Mephitis mephitis
Mephitis macroura
Conepatus leuconotus
Potos flavus
Procyon lotor
Procyon cancrivorus
Nasua nasua
Nasua narica
Bassariscus sumichrasti
Bassariscus astutus
Bassaricyon gabbii
Mellivora capensis
Taxidea taxus
Meles meles
Arctonyx collaris
Martes flavigula
Martes melampus
Martes zibellina
Martes martes
Martes foina
Martes americana
Gulo gulo
Martes pennanti
Eira barbara
Melogale moschata
Vormela peregusna
Ictonyx striatus
Mustela erminea
Mustela sibirica
Mustela nigripes
Mustela lutreola
Mustela putorius
Mustela eversmanii
Mustela nivalis
Mustela altaica
Neovison vison
Mustela frenata
Mustela africana
Pteronura brasiliensis
Poecilogale albinucha
Ictonyx libyca
Galictis vittata
Galictis cuja
Enhydra lutris
Lontra longicaudis
Lontra felina
Lontra canadensis
Hydrictis maculicollis
Lutra lutra
Lutrogale perspicillata
Aonyx cinerea
Aonyx capensis
Ailurus fulgens
Tremarctos ornatus
Melursus ursinus
Ursus maritimus
Ursus arctos
Ursus thibetanus
Ursus americanus
Helarctos malayanus
Ailuropoda melanoleuca

7.6 Further info

For more information about ancestral state reconstruction check out a review of the method by Joy *et al.* [Joy et al., 2016] and chapter 3 of *The comparative approach in evolutionary anthropology and biology* [Nunn, 2011].

For more information about the phytools package [Revell, 2012], the package author Liam Revell maintains an excellent blog here where you'll find lots of useful tips and demonstrations of the package's capabilities as well as some helpful troubleshooting.

Chapter 8

Ancestral State Reconstruction II

Previously, we looked at reconstructing the evolutionary history of binary traits, such as the presence or absence of sexual swellings in macaques, and categorical traits such as the modes of burrowing in carnivores. In this chapter, we'll be applying the same principles to continuous data.

The logic of ancestral state reconstruction applies equally to continuous traits like body size as it does to categorical traits. Here, we'll be looking at the evolutionary history of whales, dolphins and porpoises (Cetacea).

As always, check that you have set your working directory!

8.1 Data

The data we have here is taken from a study of the evolution of cetacean brain and body size [Montgomery et al., 2013]. The reduced version here contains only body mass and the log transformed body mass for 42 species.

```
whale.data <- read.table("whales_data.txt", header = T)
rownames(whale.data) <- whale.data$species
```

8.2 Tree

We also have a tree from the 10ktrees project [Arnold et al., 2010]. For more information about this website, see chapter 3.

```
whale.tree <- read.nexus("whales_tree.nex")
```

We need to check the data and tree match up. Get into this habit! It will save you a lot of time and patience.

```
rownames(whale.data) <- whale.data$species
geiger::name.check(whale.tree, whale.data)
```

```
$tree_not_data
 [1] "Balaenoptera_acutorostrata" "Balaenoptera_bonaerensis"
 [3] "Balaenoptera_edeni"         "Berardius_arnuxii"
 [5] "Berardius_bairdii"          "Caperea_marginata"
 [7] "Cephalorhynchus_eutropia"   "Cephalorhynchus_hectori"
 [9] "Delphinus_capensis"         "Delphinus_tropicalis"
[11] "Eubalaena_australis"        "Eubalaena_glacialis"
[13] "Eubalaena_japonica"         "Feresa_attenuata"
[15] "Hyperoodon_ampullatus"      "Hyperoodon_planifrons"
[17] "Indopacetus_pacificus"      "Lagenodelphis_hosei"
[19] "Lagenorhynchus_australis"   "Lagenorhynchus_cruciger"
[21] "Lissodelphis_peronii"       "Mesoplodon_bidens"
[23] "Mesoplodon_bowdoini"        "Mesoplodon_carlhubbsi"
[25] "Mesoplodon_ginkgodens"      "Mesoplodon_grayi"
[27] "Mesoplodon_hectori"         "Mesoplodon_layardii"
[29] "Mesoplodon_perrini"         "Mesoplodon_peruvianus"
[31] "Mesoplodon_stejnegeri"      "Orcaella_brevirostris"
[33] "Orcaella_heinsohni"         "Peponocephala_electra"
[35] "Phocoena_dioptrica"         "Phocoena_sinus"
[37] "Platanista_minor"           "Sousa_chinensis"
[39] "Stenella_attenuata"         "Stenella_frontalis"
[41] "Tasmacetus_shepherdi"       "Tursiops_aduncus"

$data_not_tree
character(0)
```

Clearly some species need to be dropped from the tree!

```
whale.tree <- drop.tip(whale.tree,
                       geiger::name.check(whale.tree, whale.data)$tree_not_data)
geiger::name.check(whale.tree, whale.data)
```

```
[1] "OK"
```

8.3 Ancestral State Reconstructions

Now we're going to dive in with a reconstruction. We are using **phytools** for this analysis so we should load the package and create a named data vector

[Revell, 2012].

```
require(phytools)
x <- whale.data$log.body.mass
names(x) <- whale.data$species
```

The function we need is called **fastAnc** and it returns the ancestral states in a simple list.

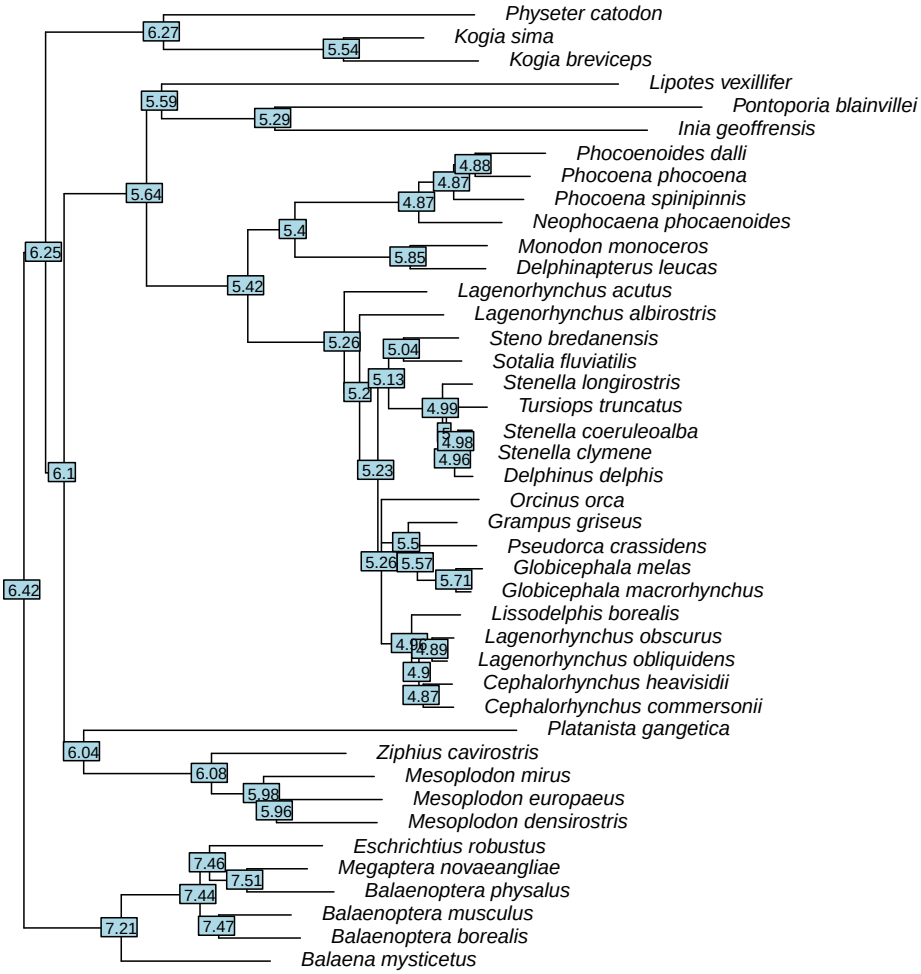
```
ancstates <- fastAnc(tree = whale.tree,    #Our phylogeny
                     x,                    #Our data vector
                     CI = TRUE)            #Estimate 95% confidence intervals
ancstates
```

```
Ancestral character estimates using fastAnc:
      43       44       45       46       47       48       49       50
6.422936 7.205471 7.440591 7.465284 7.456463 7.511707 6.248774 6.097254
      51       52       53       54       55       56       57       58
6.044864 6.078069 5.983733 5.962172 5.641527 5.423179 5.255812 5.204571
      59       60       61       62       63       64       65       66
5.225028 5.260263 4.960018 4.901903 4.871755 4.888973 5.503471 5.567067
      67       68       69       70       71       72       73       74
5.710190 5.130854 4.989830 5.001555 4.960103 4.976035 5.039423 5.403331
      75       76       77       78       79       80       81       82
5.850745 4.870560 4.871380 4.883752 5.590679 5.292187 6.267476 5.540614

Lower & upper 95% CIs:
      lower    upper
43 5.745860 7.100012
44 6.599516 7.811426
45 7.009526 7.871657
46 7.037792 7.892775
47 7.023487 7.889440
48 7.078101 7.945312
49 5.617637 6.879912
50 5.480639 6.713870
51 5.396996 6.692732
52 5.495449 6.660689
53 5.489021 6.478445
54 5.474786 6.449558
55 4.977189 6.305864
56 4.818460 6.027899
57 4.833079 5.678546
58 4.834286 5.574857
59 4.930739 5.519317
60 4.970172 5.550353
61 4.679687 5.240349
```

```
62 4.656292 5.147513
63 4.621817 5.121692
64 4.661301 5.116645
65 5.175817 5.831126
66 5.231290 5.902844
67 5.455762 5.964619
68 4.810518 5.451190
69 4.731256 5.248405
70 4.760819 5.242292
71 4.767162 5.153044
72 4.800446 5.151624
73 4.692067 5.386779
74 4.792073 6.014589
75 5.352644 6.348847
76 4.380205 5.360915
77 4.449659 5.293101
78 4.482168 5.285335
79 4.892863 6.288496
80 4.417990 6.166384
81 5.492347 7.042604
82 4.959783 6.121445
```

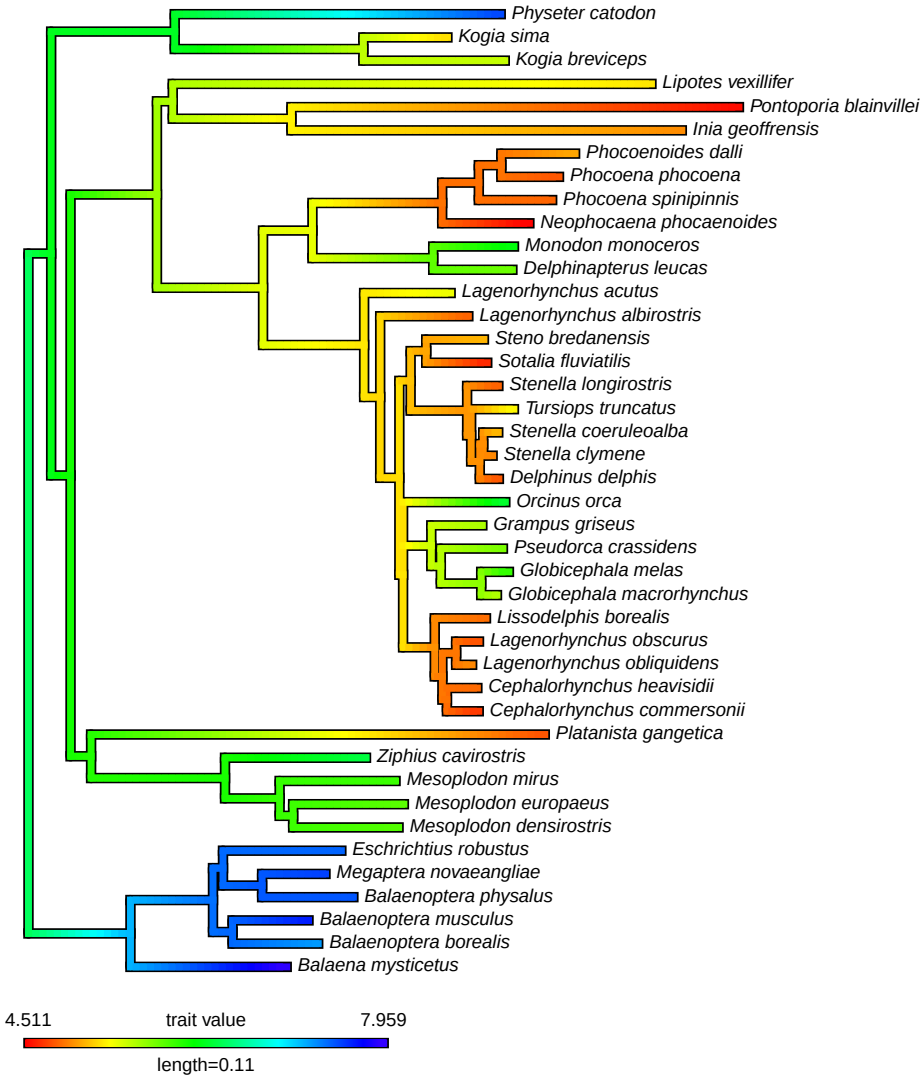
To get an idea of what these results show, we should probably plot it. The **nodelabels** function maps the ancestral states listed in our **ancstates** object onto the nodes of the tree which are listed in the same order.

```
plot(whale.tree, cex = .8, label.offset = .01, no.margin = TRUE)
nodelabels(round(ancstates$ace, digits = 2), cex = .67)
```



As is often the case, there are better ways to plot this information! The function **contMap** calls **fastAnc** and then maps the history of the trait onto the tree as a heatmap. This is a much clearer plot.

```
contMap(whale.tree, x, fsize = .7)
```



8.4 BayesTraits

Simply reconstructing the history of a trait can be very interesting. See some papers by Montgomery *et al.* [2010; 2013] for just a few great examples. However, this methodology is not limited to simply estimating the past.

Most of what we are going to do here could probably be acheived in R either with existing packages or some clever coding. However, the standard package for several analyses has been BayesTraits for some time.

BayesTraits is a command line program, which can make it kind of intimidating. Actually (like R), it's relatively easy to use but can take some getting used to. Fortunately, Randi Griffin has written an excellent R package **btw** that can operate the program from within R.

It's worth noting at this point that **btw** is not written to run BayesTraits for you so that you don't have to understand the program. Randi states very clearly that the package is purely for optimising workflow. In other words, this allows you to have all your data, results and code in one place. You still need to understand how to use the program. Fortunately the manual is very detailed.

First up, download BayesTraits version 3 for your operating system.

IMPORTANT! BayesTraits output files will be written into your working directory. They will overwrite any files with the same name so don't have any files called "data.txt", "tree.nex" or "inputfile.txt" in this directory unless you are ok with losing them.

Next, we need to install **btw**. This isn't a CRAN archived package so we'll be installing directly from Randi Griffin's GitHub. Once installed, we can use BayesTraits from within R!

```
install.packages("devtools")
library(devtools)
install_github("rgriff23/btw")
library(btw)
```

There are some important differences in how R and BayesTraits read data that need to be summarised here.

- The first column of your data must contain species names.
- Species names must match exactly between tree and data (but don't worry about the order).
- No spaces in species names.
- Discrete characters have to be of class character or factor (between 0-9) and NOT integer.
- Ambiguous discrete characters can be represented as 01.
- Missing data must be represented as - rather than NA.

BayesTraits consists of modules (see manual for details) that are numbered and can be called up for different analyses.

If you can't get R and Bayestraits to play nicely together, you may want to consider using Bayestraits directly from the command prompt (Windows) or terminal (Mac). It's fairly straightforward once you've got the hang of it so be patient. Alternatively, all of this can be done with R packages like ape [Paradis and Schliep, 2018], geiger [Harmon et al., 2008] and phytools [Revell, 2012] amongst others.

8.5 Modelling Evolution

If we have some data about traits across a group of animals and an associated tree, we may want to ask about how that trait has evolved over time. For this we can compare the trait to models of evolutionary change.

8.5.1 Brownian Motion

Brownian motion (BM) is the most commonly used model of evolutionary change. In some ways, it can represent a kind of *null model* but do not confuse this! It doesn't mean nothing is changing or that evolution is not taking place.

Brownian motion assumes three things;

- Evolutionary changes in a trait are randomly distributed around a mean of 0.
- Evolutionary changes in a trait are independent of previous changes and changes on other branches.
- Larger changes are more likely to occur on longer branches.

All this means that BM is a *random walk* model in which the trait varies along the branches essentially at random.

We can use BayesTraits (via R) to model the evolution of body size in cetaceans with the assumption of Brownian motion. First we need to isolate our variables into a data table for **btw**. The way to do this is quite simple. We can simply extract the two columns we need (1 and 2) into a new object.

```
BT.data <- whale.data[,c(1,2)]
rownames(BT.data) <- NULL
head(BT.data)
```

```
                species log.body.mass
1       Kogia_breviceps      5.523746
2            Kogia_sima      5.226600
3     Physeter_catodon      7.573065
4  Platanista_gangetica      4.775465
5 Delphinapterus_leucas      5.803457
6     Monodon_monoceros      6.198198
```

This first analysis corresponds to **Continuous: Random Walk Model A ML** in the BayesTraits manual. We can see from the manual that the commands to run this are "4 1 Run". You need to be familiar with BayesTraits to interpret this so the first time you do it, you may want to do it in BayesTraits directly (via the command prompt or terminal). In essence, BayesTraits asks us questions and provides us with options for what we want it to do and **4, 1, Run** are the options to run this analysis.

Given that we know what we want to do ahead of time, we can enter the commands into a command vector in R. To run these commands through BayesTraits, R will write them into a text file so BayesTraits can interpret them when needed. Note that you don't need to enter **Run** into this vector as **btw** will take care of that for us.

```
command_vec1 <- c("4", "1")
```

Note that if you have nodelabels in your tree, there will be an error when running BayesTraits. You can remove nodelabels without effecting the structure of your tree like this.

```
whale.tree$node.label <- NULL
```

I have a path on my desktop just for BayesTraits analyses. Remember that there must be a copy of BayesTraitsV3 stored here. That's all you need as the output will be read back into R by **btw**. You also should remember to change your working directory back if you are finished with BayesTraits. In this chunk, I've saved the existing directory at the start and reset it immediately after the analysis is completed.

```
wd.reset <- getwd()
setwd("~/Desktop/BayesTraits")
m1 <- bayestraits(data = BT.data, tree = whale.tree, commands = command_vec1)
setwd(wd.reset)
```

On we go! The object that should have appeared in your R environment contains all the outputs you need from BayesTraits. Let's have a look at the **results** component of the **Log**.

```
m1$Log$results
```

```
  Tree.No       Lh  Alpha.1 Sigma.2.1
1       1 -31.9823 6.422936  6.315406
```

These results give us the Log likelihood (**Lh**), the reconstructed ancestral node (**Alpha.1**) and the phylogenetically corrected variance of the data (**Sigma.2.1**). The important thing to look at here is the log likelihood. We will use that to compare the BM model to other models.

8.5.2 Directional Evolution

So far we've looked at the random walk model of evolution. In reality, what we are usually interested in is deviations from the random walk model. We can investigate this using similar methods, but with a **directional** model.

An example of a case when we might be interested in a directional model is Cope's rule [Kingsolver and Pfennig, 2004, Hone and Benton, 2005]. Cope's

rule states that over time, lineages tend to have larger body sizes. So basically, on average animals tend to get bigger over evolutionary time.

Let's see if we can detect a trend in cetacean body mass. For this analysis, we need a non-ultrametric tree (a phylogram rather than a chronogram). Luckily that's what we already have. The branch lengths here describe evolutionary distance in terms of genetic change and so shorter branches indicate fewer genetic changes.

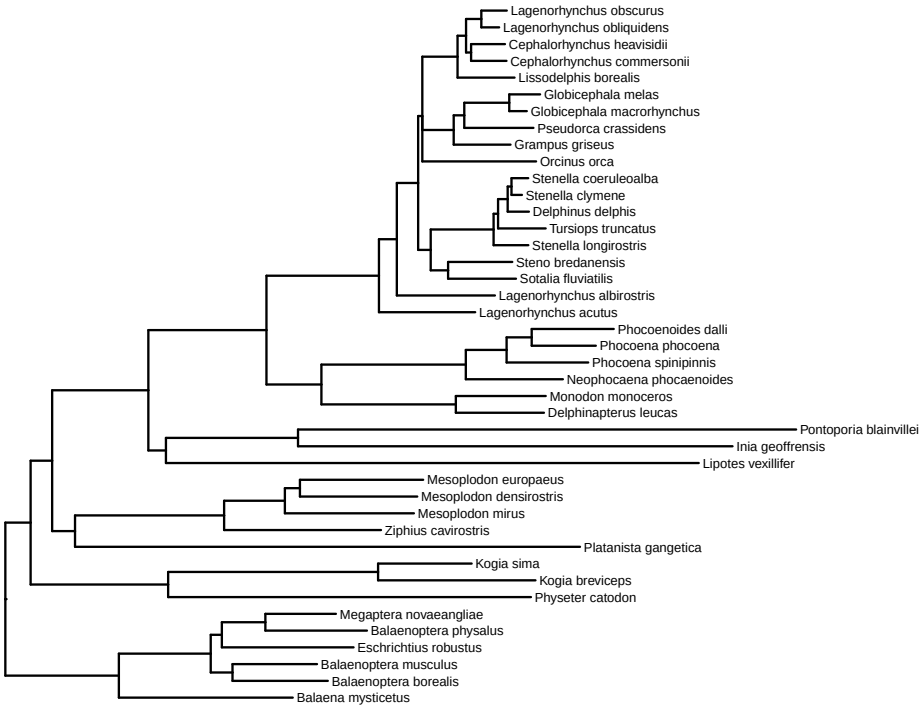


Figure 8.1: Phylogenetic tree of 42 species of cetcaeans with branch lengths proportional to molecular change.

We're using BayesTraits again so the first step is to get our data into the right format.

```
BT.data <- whale.data[,c(1,2)]
rownames(BT.data) <- NULL
```

As before, we run the random walk (BM) model first. Remember we need to set the working directory to a path where BayesTraits is stored.

```
setwd("~/Desktop/BayesTraits")
RW.commands <- c("4", "1")
RWmod <- bayestraits(BT.data, whale.tree, RW.commands)
```

```
RWmod$Log$results
```

```
  Tree.No       Lh  Alpha.1 Sigma.2.1
1       1 -31.9823 6.422936  6.315406
```

```
setwd(wd.reset)
```

The directional model takes a different set of commands.

```
setwd("~/Desktop/BayesTraits")
D.commands <- c("5", "1")
Dmod <- bayestraits(BT.data, whale.tree, D.commands)
Dmod$Log$results
```

```
  Tree.No        Lh  Alpha.1    Beta.1 Sigma.2.1
1       1 -30.22462 7.951217 -12.37922  5.808331
```

```
setwd(wd.reset)
```

Now we need to compare these models! What he have so far is two models and a log likelihood assigned to each. This means we can compare them using a **likelihood ratio test**. The general formula for an LR test is;

$$LR = 2 * (Lh_{ModelB} - Lh_{ModelA})$$

The result is the **likelihood ratio statistic** (LR) which is asymptotically χ^2 distributed with degrees of freedom equal to the difference in the number of parameters between the models. Model A has 1 parameter (the root value) and model B has 2 (the root and the direction of change) so the degrees of freedom are 1.

```
2*(Dmod$Log$results$Lh[1] - RWmod$Log$results$Lh[1])
```

```
[1] 3.515352
```

```
1-pchisq(3.515352, df = 1)
```

```
[1] 0.06080274
```

Note: pchisq gives the proportion of the distribution to the left of the value. To test if the model is better than the null model, we use 1 - pchisq.

The **btw** package has a function that will do all this for us. Be careful with interpretation though. Note that the p-value is different. Take this away from 1 and you have your p-value as above.

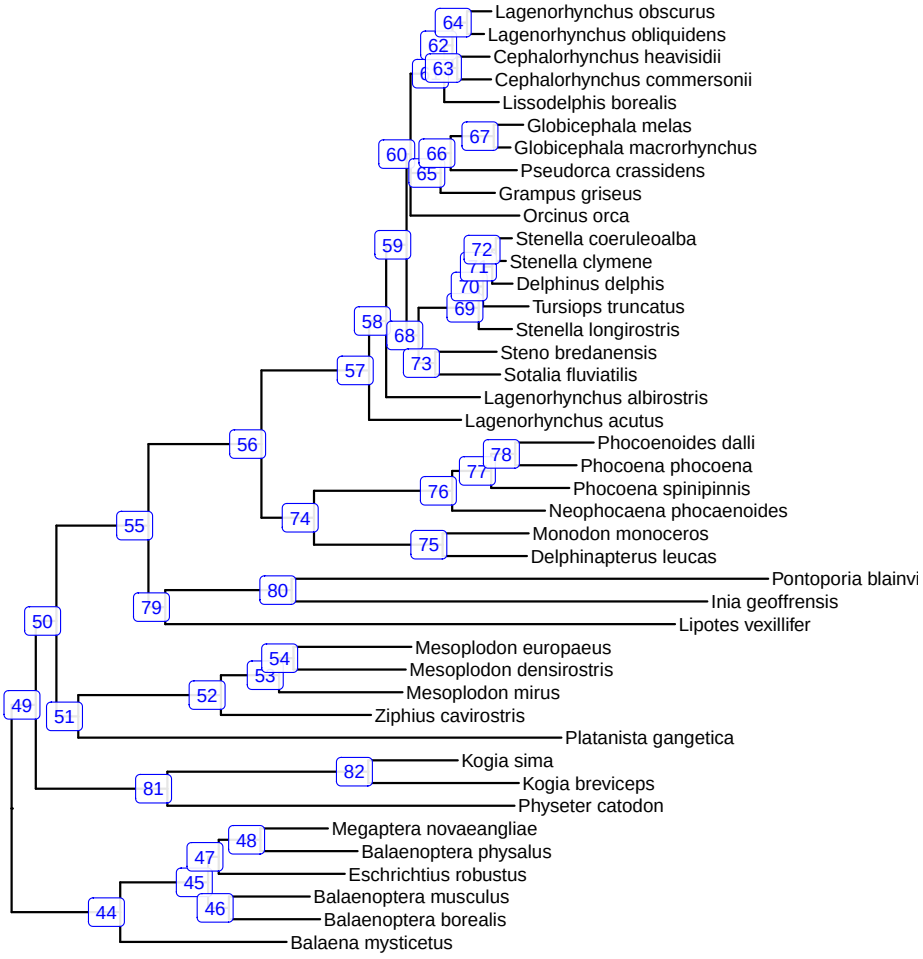
```
lrtest(RWmod, Dmod)
```

```
  model1.Lh model2.Lh   LRstat      pval
1  -31.9823 -30.22462 3.515352 0.9391973
```

So what have we got here? Well we have tested two models of the evolution of body size in cetacea. The first is a random walk (Brownian motion) model of evolution in which we have estimated two parameters. The second is a directional model in which we have estimated 3 parameters. Model comparison showed no significant difference between them (LR = 3.52, p = 0.06) and so we should favour the simpler, 2 parameter model. Thus we have no evidence for a directional trend in cetacean body mass evolution.

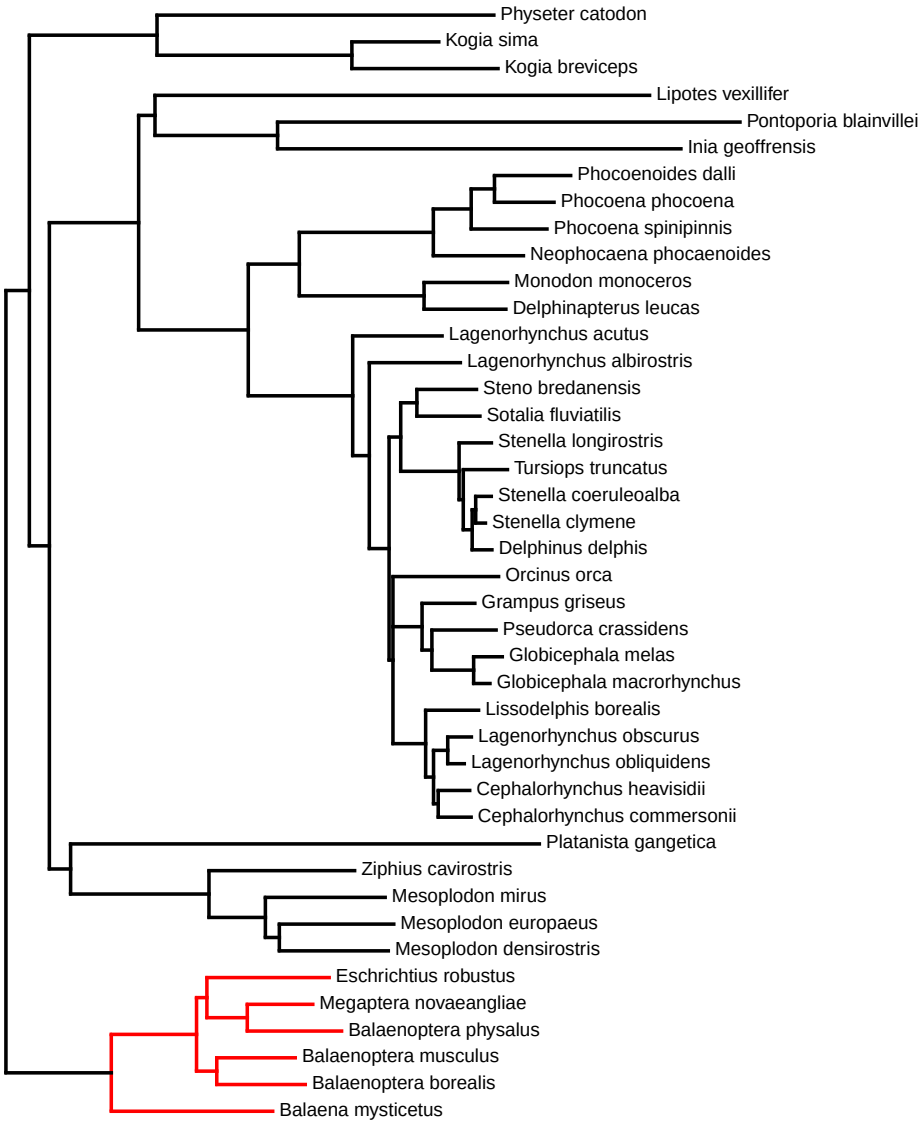
8.6 Changes in the rate of evolution of a trait

Often when investigating the evolution of a continuous trait, we might have reason to suggest that in some lineages, the rate of evolution of that trait changed.



Let's say we have a hypothesis that says the rate of change of body mass changed at the root of mysticetes (node 44). We can paint that onto the tree for demonstration using **paintSubTree** and **plotSimmap** in **phytools** [Revell, 2012].

```
require(phytools)
tree1 <- paintSubTree(whale.tree, 44, "2")
plotSimmap(tree1, lwd = 2, fsize = 0.7)
```



Now we can run the test. Here the function **brownie.lite** in **phytools** compares the single rate model to the multi-rate model we have specified!

```
x <- whale.data$log.body.mass
names(x) <- whale.data$species
fit <- brownie.lite(tree1, x)
fit
```

```
ML single-rate model:
    s^2 se  a   k   logL
value   6.165   1.3453  6.4229  2   -31.9823

ML multi-rate model:
    s^2(1)  se(1)   s^2(2)  se(2)   a   k   logL
value   7.1119  1.6786  1.2663  0.8687  6.6116  3   -30.475

P-value (based on X^2): 0.0825

R thinks it has found the ML solution.
```

Here we've found no evidence of a regime shift in mysticete cetaceans ($p = 0.083$).

8.7 Uncertainty

If you are familiar with cetaceans and their evolutionary history, you might be surprised by our findings so far in this chapter. The prevailing state of knowledge suggests that cetaceans have evolved large body sizes since the transition to the water of an approximately dog-sized ancestor at the root of our tree. Given what we know about the fossil record of cetacea, we would expect to detect an increase in body size over the tree. To solve this puzzle, we need to look at what information we provided our analysis with.

As the old saying goes, *if you put garbage in, you'll get garbage out* and this seems to apply here. For example, let's look closely at our reconstructions. You can see here that both reconstructions have estimated the mass of the ancestor of cetaceans. Remember that these are log transformed data so we have to transform them back if we want to get a straightforward measurement of mass.

```
10^(RWmod$Log$results$Alpha.1)
```

```
[1] 2648111
```

```
10^(Dmod$Log$results$Alpha.1)
```

```
[1] 89375216
```

So depending on our model of evolution the ancestor was either 2,648.1 kg or 89,375.2 kg. A big difference between models so which one we choose really matters.

This is even more of a problem when we look at the fossil record of cetaceans. *Indohyus* (Raoellidae) is thought to be the species that most closely represents the transition to the water by cetacean ancestors [Thewissen et al., 2009] and its mass is estimated at around 10kg. An early species of cetacean called *Pakicetus* was estimated at around 45kg. So we are orders of magnitude away from what the fossil record shows us!

This problem is well understood in phylogenetic comparative methods. In fact, all methods of ancestral state reconstruction perform very poorly when compared to what we know from the fossil record [Webster and Purvis, 2002]. As you might expect, the deeper into your tree you try to estimate an ancestral state, the greater the uncertainty. This is especially clear when you look at estimating the root [Gascuel and Steel, 2014]. The solution is to incorporate fossil data in the analysis [Slater et al., 2012].

8.7.1 Fossils

To demonstrate the importance of fossil data, let's take a closer look at the evolution of body size in cetaceans. With **fastAnc**, we found a mass of around 2,650kg for the root of the cetaceans.

The package **RRphylo** [Castiglione et al., 2018] contains data on fossil and living cetaceans [Serio et al., 2019]. Using these data, we can hopefully perform a more rigorous ancestral state reconstruction [Castiglione et al., 2020]. Note that the values here differ betwwen datasets because the previous dataset used a log10 transformation whereas this one uses a natural log transformation!

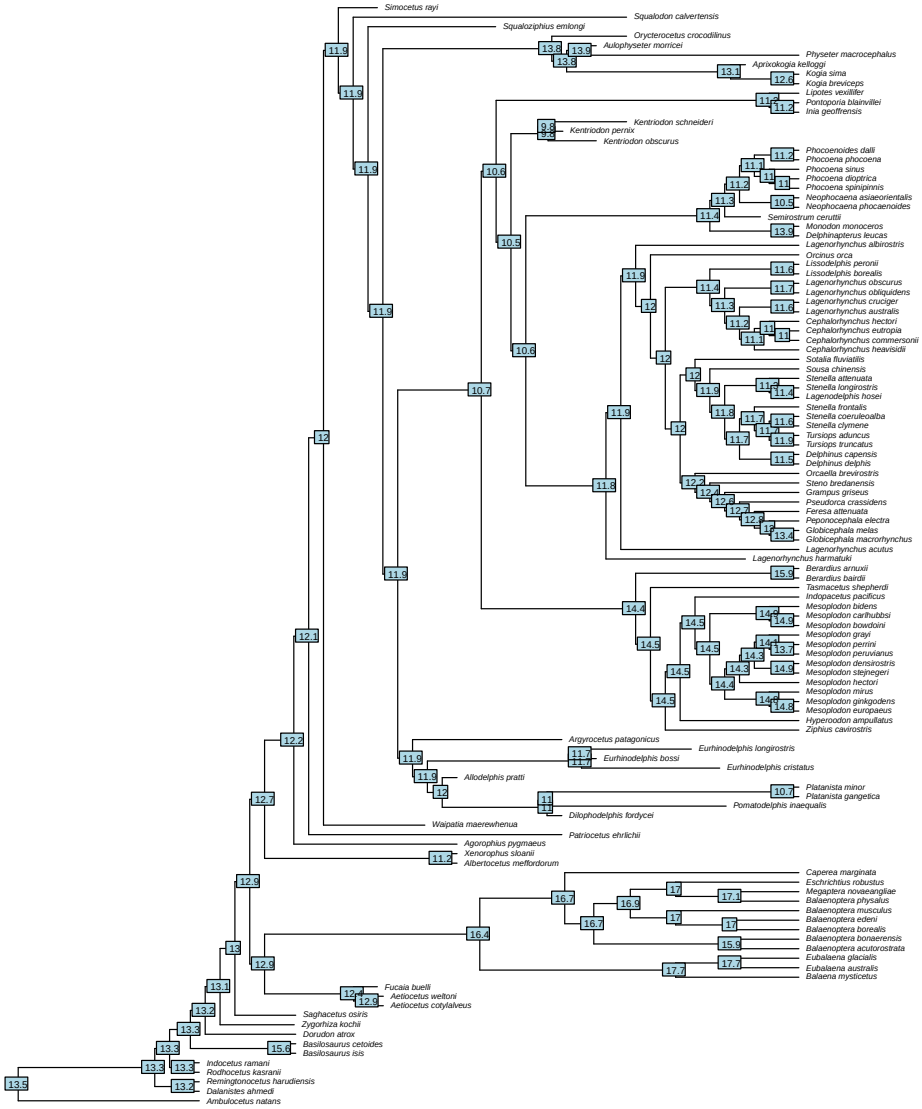
```
library(RRphylo)
data("DataCetaceans")
DataCetaceans$treecet -> treecet
DataCetaceans$masscet -> masscet
```

The **RRphylo** function performs a variant of ancestral state reconstruction called **phylogenetic ridge regression** [Castiglione et al., 2020].

```
RR <- RRphylo(treecet, masscet)
```

RRphylo returns a lot of information as a list. Included in this list is the **tree** used (useful for plotting) and **aces** which contains the estimates for the traits at the nodes.

```
plot(RR$tree, cex = .4, label.offset = .5, no.margin = TRUE)
nodelabels(round(RR$aces, digits = 1), cex = .5)
```



Using this reconstruction, we can extract the mass of the root. Remember that we need **exp** to calculate the untransformed value rather than raising to the power of 10 because of the natural log transformation.

```
exp(RR$aces[[1]])
```

```
[1] 727063.9
```

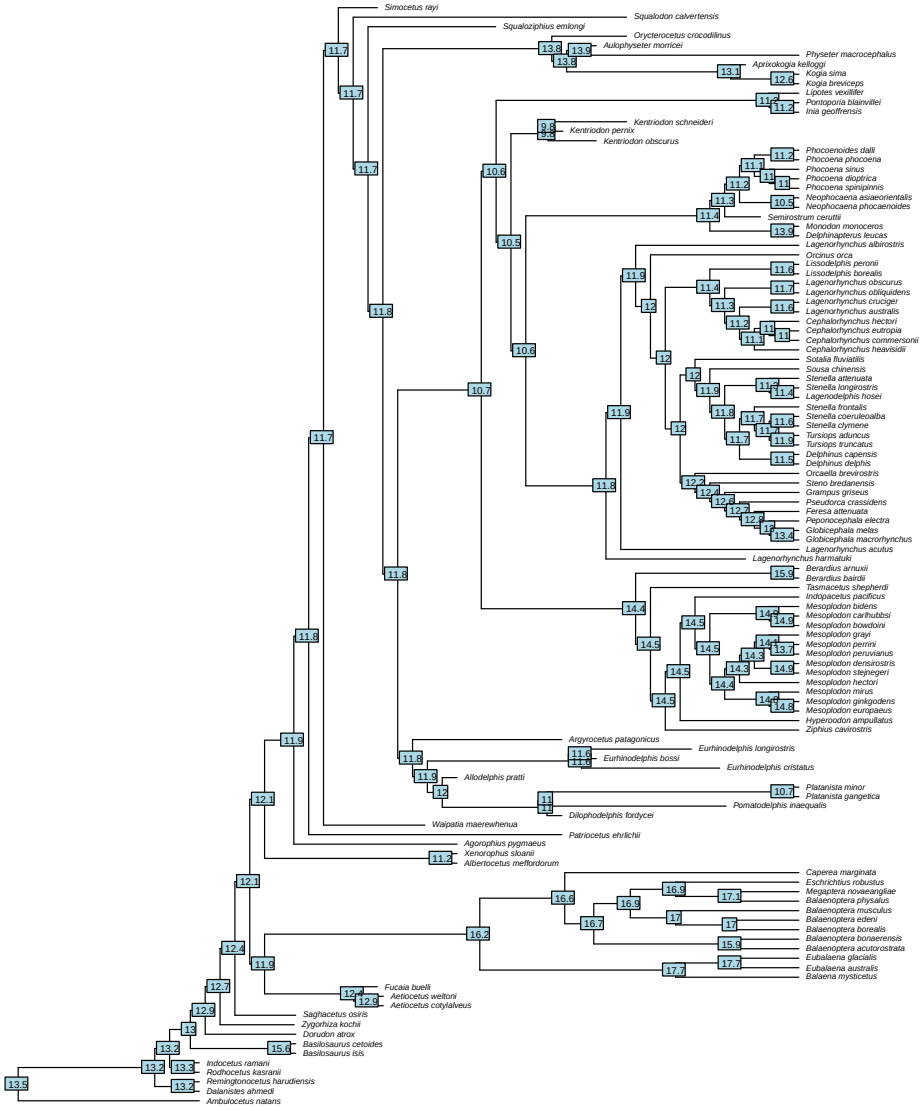
So our new estimate of the mass of the ancestor of cetaceans is 727.1kg. This is much closer to the estimated mass of early archeocete cetaceans like *Ambulocetus natans* at about 430kg and *Indocetus ramani* at around 630kg.

If we still aren't satisfied that we have included the best information we have available, we can actually *fossilise* a node by passing a named list of ancestral states to the **RRphylo** function. Following the example of Castiglione *et al.* [2020], we can set the node of the ancestor of mysticetes to a known mass. Here we are assuming that the most recent common ancestor of all mysticetes can be represented by the species *Mystacodon selenensis* which weighed arond 150kg. Also we need to know that this ancestor is represented by the node labelled 128 in our tree object.

```
x <- log(150000)
names(x) <- "128"
```

Now we can pass this state to the argument **aces** in **RRphylo** and the analysis will hold node 128 at the value we have set. You should be able to see that in the following plot, the ancestor of mysticetes is reconstructed as 11.9 rather than 12.9 in the previous reconstruction.

```
RR2 <- RRphylo(treecet, masscet, aces = x)
plot(RR2$tree, cex = .4, label.offset = .5, no.margin = TRUE)
nodelabels(round(RR2$aces, digits = 1), cex = .5)
```



Hopefully you can see that the more fossil information you include in your reconstructions, the more reliable they are.

8.7.2 Revisiting Mysticete Body Mass

In using the fossil data we have added in here, Castiglione *et al.* [2020] demonstrated that mysticetes actually do conform to Cope's rule because they have an increasing trend in body size over time. This shows just how important adding in fossil data can be if you want the full picture.

This seems to suggest that we should also find a regime shift in mysticetes. Let's have a closer look. We begin again by painting the tree at the specific node leading to mysticetes.

```
require(phytools)
tree2 <- paintSubTree(treecet, 128, "2")
plotSimmap(tree2, lwd = 2, fsize = 0.5)
```

Next we run **brownie.lite** on our expanded dataset.

```
fit <- brownie.lite(tree2, masscet)
fit
```

```
ML single-rate model:
    s^2 se  a   k   logL
value   0.2047  0.0265  13.2057 2   -178.7922

ML multi-rate model:
    s^2(1)  se(1)   s^2(2)  se(2)   a   k   logL
value   0.176   0.0245  0.4012  0.147   13.2056 3   -176.0958

P-value (based on X^2): 0.0202

R thinks it has found the ML solution.
```

There you have it! We can now say that we have evidence in favour of a regime shift in mysticete body size ($p = 0.02$).

8.8 Further info

We've only just scratched the surface of what is possible with ancestral state reconstruction. For some background reading, have a look at chapter 4 of *The comparative approach in evolutionary anthropology and biology* [Nunn, 2011].

Chapter 9

Diversification

9.1 Introduction

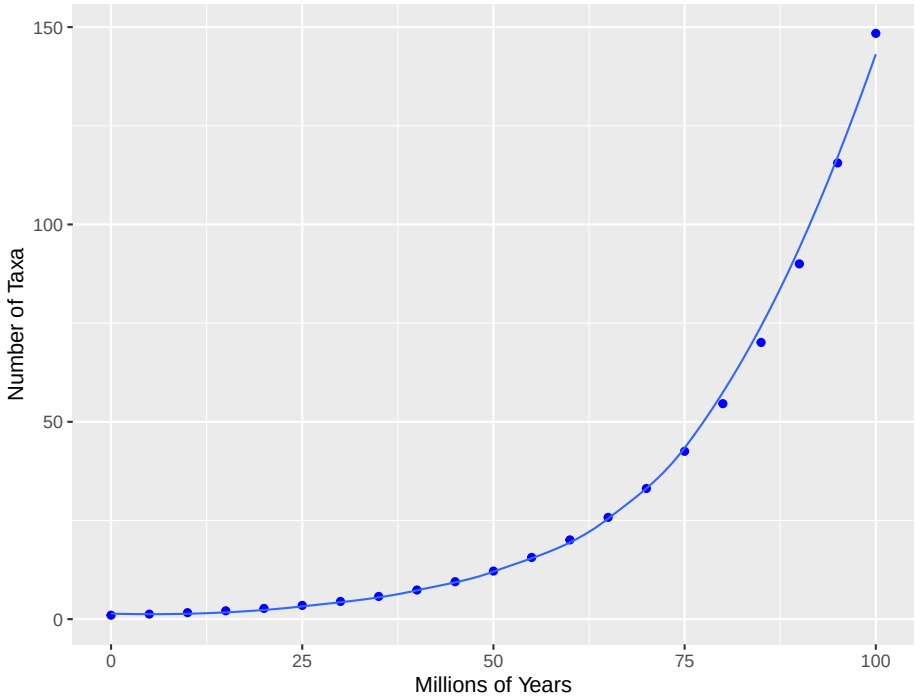
Phylogenetic comparative methods can be used to study the diversification of lineages over time. This can be particularly useful when studying adaptive radiations which are thought to change the rate of speciation. This chapter will begin by studying some diversification models.

9.1.1 Pure Birth Model

The simplest form of diversification model is a **pure birth** model. Under this model, each lineage has a constant probability of speciating. This probability gives us the **birth rate** (b). Over time (t), lineages will speciate and thus the number of lineages (N) will increase exponentially.

$$N_{(t)} = N_{(0)}e^{bt}$$

This plot shows how the number of species would grow over time. Note that in this model, there is no extinction and so the exponential increase is because each speciation event produces two daughter lineages which each can then speciate with probabilty b, creating two more daughter lineages each and so on.



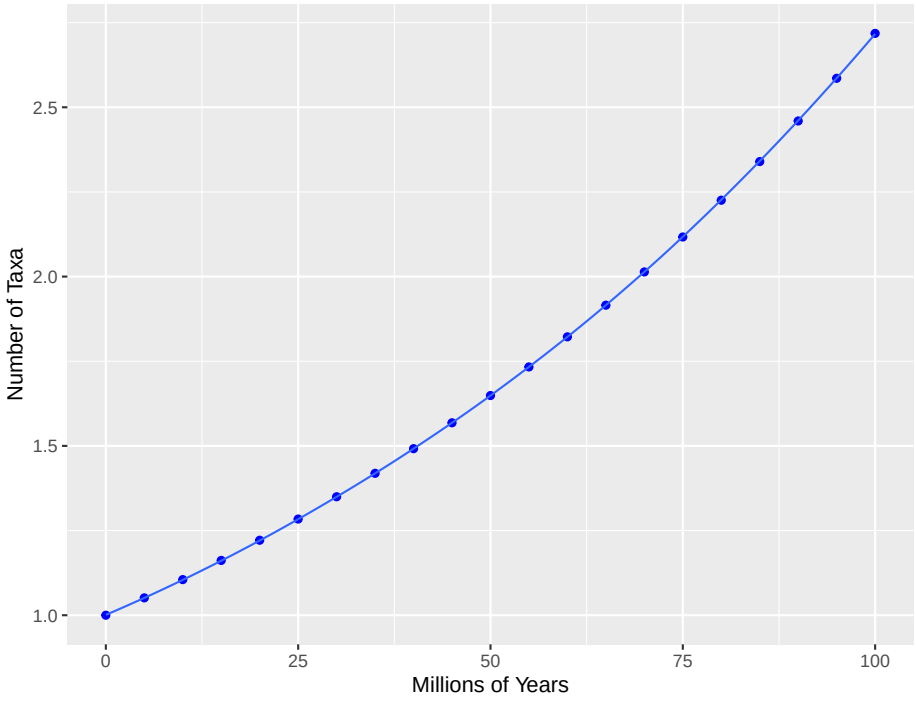
As you can probably tell, the pure birth model is not a very good representation of biological reality. For example, today there are around 90 species of cetacean. A pure birth model in which we have b set at 0.1 (meaning we have 0.1 new species born every million years) would give us over 1 million species of cetacean in 100 million years.

9.1.2 The Birth-Death Model

The **birth-death** model incorporates extinction as well as speciation. Under the simplest form of this model, speciation and extinction occur randomly. Each lineage is assumed to have a constant rate of speciation (b) and extinction (d). Under this model, the number of lineages will increase exponentially with the difference between birth and death rate (b - d).

$$N_{(t)} = N_{(0)}e^{(b-d)t}$$

This graph shows the change in number of taxa over time when the birth rate slightly exceeds death rate.

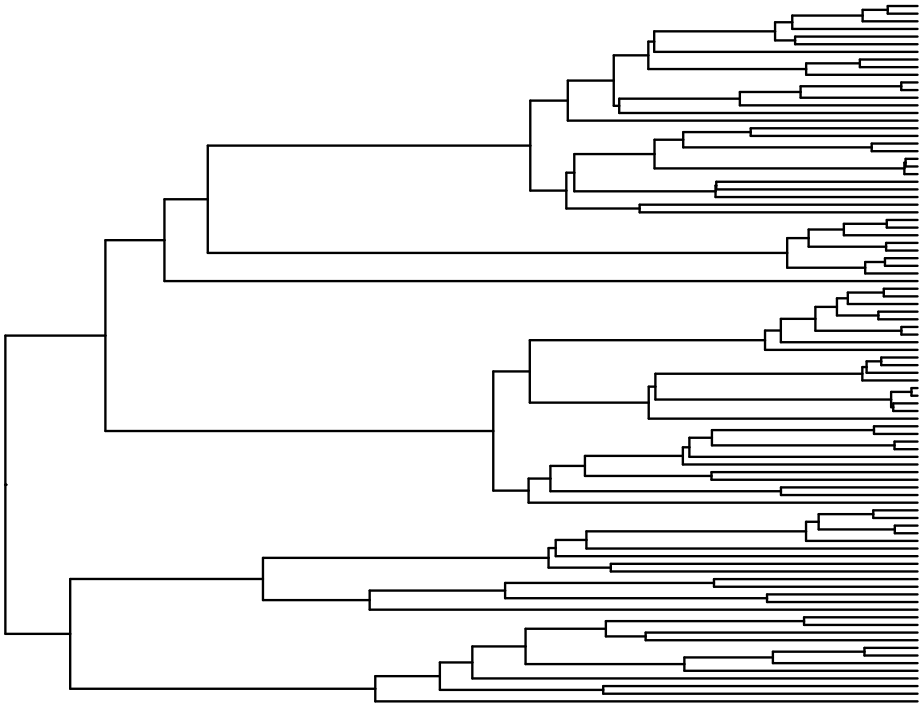


9.1.3 The signature of extinction

Let's investigate the differences between pure birth and birth-death models by simulating some trees.

The function **sim.bd.age** in the package **TreeSim** [Stadler, 2019] allows us to simulate birth-death trees with a fixed age. First we will set the birth rate λ to 0.4 and the death rate μ to 0 as this is a pure-birth tree. Note that these were b and d in the previous sections but λ and μ are widely used.

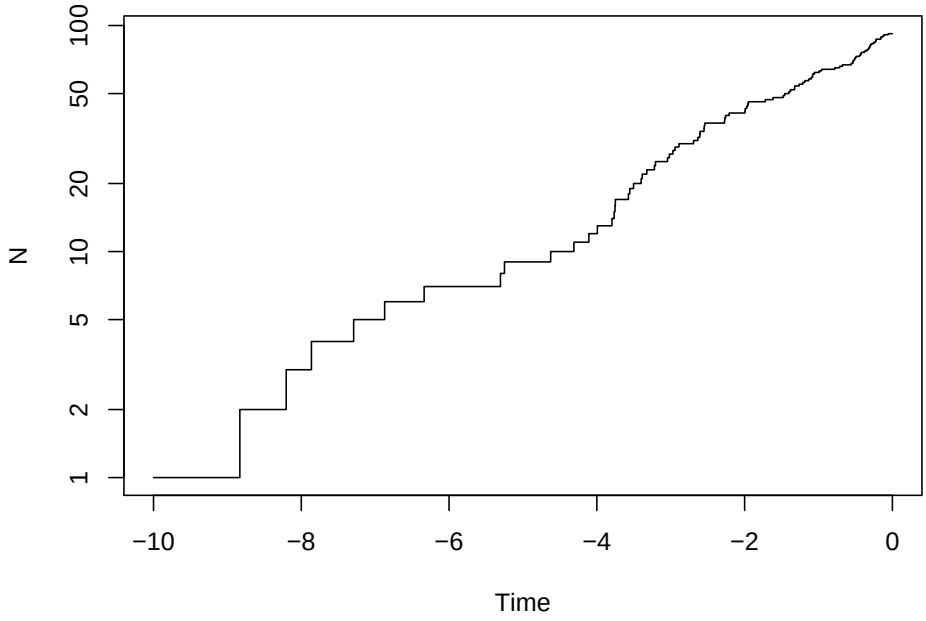
```
library(ape)
library(TreeSim)
tree1 <- sim.bd.age(age = 10, numbsim = 1, lambda = 0.4, mu = 0)[[1]]
ggtree::ggtree(tree1)
```



If you run this command yourself, your tree will look different simply because the simulations will differ each time.

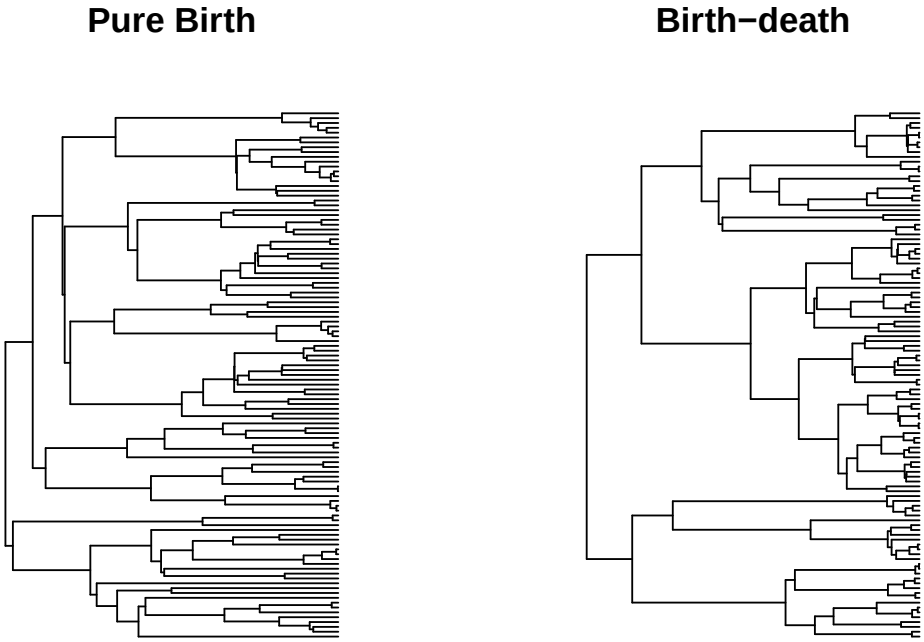
We can plot the pattern of lineage accumulation through time with a lineage-through-time plot. This can be plotted using the function **ltt.plot** in ape [Paradis and Schliep, 2018]. Note that because of the expectation of exponential growth, we should log scale the y-axis

```
ltt.plot(tree1, log = "y")
```



As can be seen from this plot, the increase in lineages over time will be log-linear, just as predicted in previous sections.

What about when we add extinction? We can see here that pure-birth and birth-death trees look quite different. this is because older lineages are more likely to go extinct. This *pull of the present* means that extinction rates appear lower in the present even though there's no reason to assume that they would be any different. We just have a greater proportion of total lineages the closer we are to the present!



9.2 Estimating speciation and extinction rate

Given a phylogeny, we can estimate speciation and extinction rate using the function **birthdeath** in **ape** [Paradis and Schliep, 2018].

```
library(ape)
whales <- read.tree("whaletree.tre")
bd.mod <- birthdeath(whales)
bd.mod
```

```
Estimation of Speciation and Extinction Rates
            with Birth-Death Models

     Phylogenetic tree: whales
        Number of tips: 87
              Deviance: -45.0534
        Log-likelihood: 22.5267
   Parameter estimates:
      d / b = 0   StdErr = 0
      b - d = 0.103623   StdErr = 0.007955189
   (b: speciation rate, d: extinction rate)
   Profile likelihood 95% confidence intervals:
      d / b: [-0.4859418, 0.2661802]
      b - d: [0.08313059, 0.1272357]
```

The output here gives us some derived values such as $b - d$ (net diversification). If you want to cut straight to the speciation and extinction rates, you can use a **phytools** function called **bd** [Revell, 2012].

```
phytools::bd(bd.mod)
```

```
       b        d
0.103623 0.000000
```

This is a nice start but it assumes constant rates across the tree and that doesn't seem like a reasonable assumption to me. We also may have good reason to doubt that the extinction rate is truly 0!

9.3 MEDUSA

The general birth-death models you have seen so far produce a fairly uniform tree of life. In reality, this isn't actually what we observe. Some clades are highly diverse (such as arthropods) and some are not. This implies that the rates of diversification can vary between lineages.

In fact, based on our understanding of evolution, we would expect rates to change all the time! We know that extinction rates fluctuate around a background rate. For example, extinction rates increase dramatically during mass extinctions. Speciation rates increase during adaptive radiations. The next methods to look at are capable of detecting these changes in rate using phylogenies.

MEDUSA stands for Modelling Evolutionary Diversification Using Stepwise AIC [Alfaro et al., 2009]. It fits piecewise models of birth and death to our tree and allows us to ask where rates of birth and death may have changed and why clades might differ in size.

MEDUSA fits models to the tree, splitting the tree and allowing the rate to change at each split. The method proceeds by comparing the old model to the new model using AICc to either accept of reject the shift in rate.

Let's see how it works using the phylogeny of cetaceans.

We can run the analysis using the **medusa** function in geiger [Harmon et al., 2008]. All we have to specify here is the tree *phy = whales* and that we want to fit birth-death models *model = "bd"*.

```
library(geiger)
m1 <- medusa(phy = whales, model = "bd")
```

```
Appropriate  aicc-threshold for a tree of 87 tips is: 4.397737.

Step 1: lnLik=-277.6943; aicc=559.4591; model=bd
```

```
Step 2: lnLik=-266.7389; aicc=543.8371; shift at node 141; model=bd; cut=stem; # shifts

No significant increase in aicc score. Disregarding subsequent piecewise models.

Calculating profile likelihoods on parameter values.

  Model.ID Shift.Node Cut.At Model Ln.Lik.part        r     epsilon     r.low
1        1         88   node    bd   -185.5379 0.076678 1.50783e-08 0.0576867
2        2        141   stem    bd   -81.20097 0.232141 4.73531e-06 0.1616522
     r.high eps.low  eps.high
1 0.0994803       0 0.2537393
2 0.3204785       0 0.2713275
```

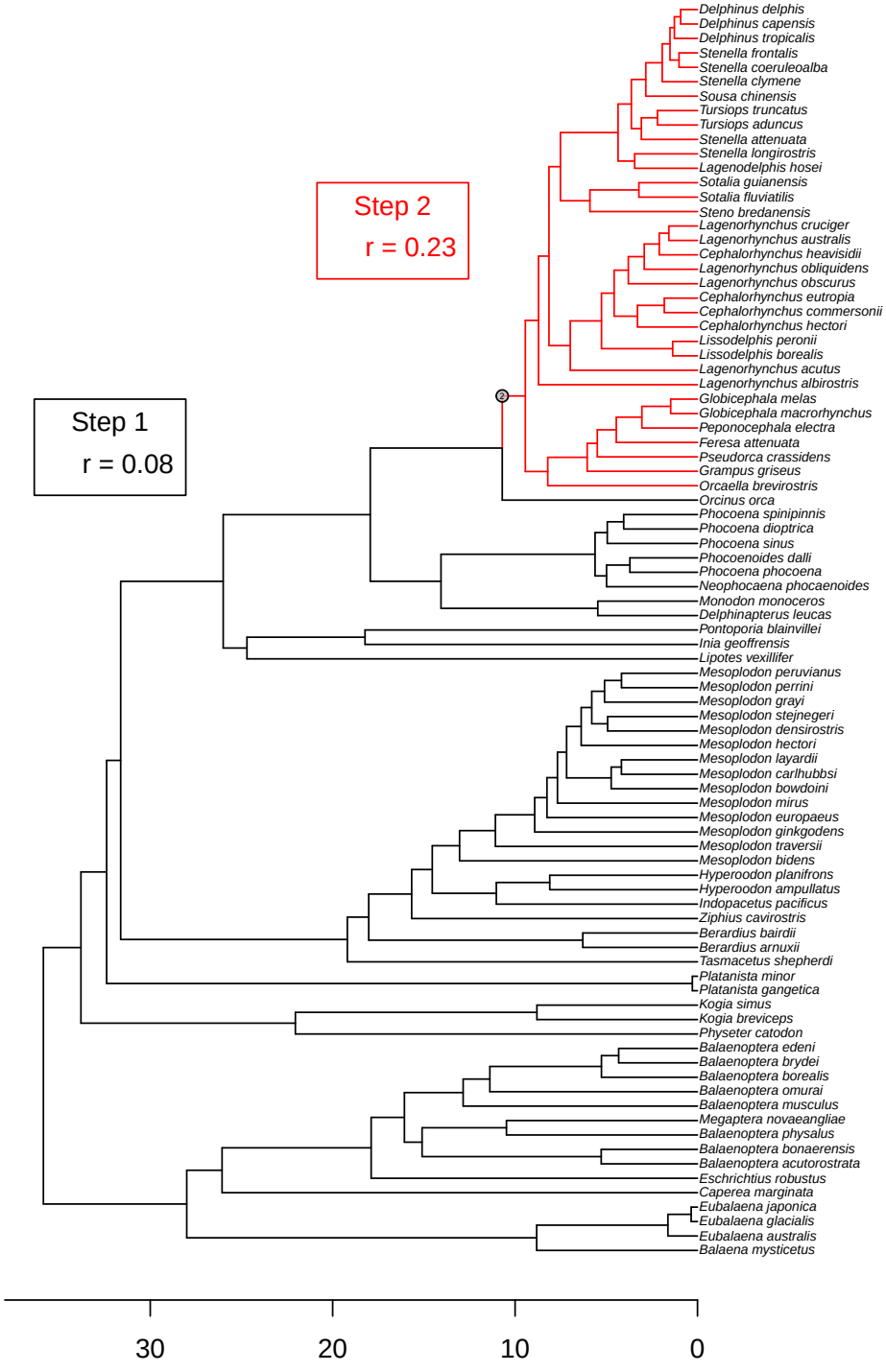
There's a lot of information in this output so let's go through it. Firstly, we are told that the appropriate AICc threshold for a tree of this size is 4.4. You may be used to using a change of 2 in AIC as signifiying a significant difference between models. The appropriate threshold for distinguishing between models using MEDUSA varies depending on the size of the tree. Fortunately, the function simply calculates it and reports it to us.

Next we have the stepwise output. In this case we only have 2 steps. Each step is reported to us and in this case, we can see that step 2 involves cutting the tree at node 141. This means that *medusa* has identfied a change in rate at node 141 and continued with the new model from that node.

Finally, the table gives us parameter values for each step. This is very important in interpreting the output. Note that *epsilon* is the extinction rate, previously known as either d or μ and r is the net diversification rate ($\lambda - \mu$).

The best way to visualise this analysis is by plotting the models on the tree. When we do so we can see that *medusa* has placed the shift near the root of the Delphinidae (excluding interestingly the orca). We can see that the rate of diversification is much greater in the oceanic dolphins.

```
par(mar = c(3.1, 0, 0.1, 0))
plot(m1, label.offset = 0.1)
legend(x = -0.5, y = 60, legend="r = 0.08",
       text.col="Black", title = "Step 1")
legend(x = 15, y = 75, legend="r = 0.23",
       text.col="red", title = "Step 2", box.col = "red")
```


Step 2
r = 0.23
Step 1
r = 0.08
Delphinus delphis
Delphinus capensis
Delphinus tropicalis
Stenella frontalis
Stenella coeruleoalba
Stenella clymene
Sousa chinensis
Tursiops truncatus
Tursiops aduncus
Stenella attenuata
Stenella longirostris
Lagenodelphis hosei
Sotalia guianensis
Sotalia fluviatilis
Steno bredanensis
Lagenorhynchus cruciger
Lagenorhynchus australis
Cephalorhynchus heavisidii
Lagenorhynchus obliquidens
Lagenorhynchus obscurus
Cephalorhynchus eutropia
Cephalorhynchus commersonii
Cephalorhynchus hectori
Lissodelphis peronii
Lissodelphis borealis
Lagenorhynchus acutus
Lagenorhynchus albirostris
Globicephala melas
Globicephala macrorhynchus
Peponocephala electra
Feresa attenuata
Pseudorca crassidens
Grampus griseus
Orcaella brevirostris
Orcinus orca
Phocoena spinipinnis
Phocoena dioptrica
Phocoena sinus
Phocoenoides dalli
Phocoena phocoena
Neophocaena phocaenoides
Monodon monoceros
Delphinapterus leucas
Pontoporia blainvillei
Inia geoffrensis
Lipotes vexillifer
Mesoplodon peruvianus
Mesoplodon perrini
Mesoplodon grayi
Mesoplodon stejnegeri
Mesoplodon densirostris
Mesoplodon hectori
Mesoplodon layardii
Mesoplodon carlhubbsi
Mesoplodon bowdoini
Mesoplodon mirus
Mesoplodon europaeus
Mesoplodon ginkgodens
Mesoplodon traversii
Mesoplodon bidens
Hyperoodon planifrons
Hyperoodon ampullatus
Indopacetus pacificus
Ziphius cavirostris
Berardius bairdii
Berardius arnuxii
Tasmacetus shepherdi
Platanista minor
Platanista gangetica
Kogia simus
Kogia breviceps
Physeter catodon
Balaenoptera edeni
Balaenoptera brydei
Balaenoptera borealis
Balaenoptera omurai
Balaenoptera musculus
Megaptera novaeangliae
Balaenoptera physalus
Balaenoptera bonaerensis
Balaenoptera acutorostrata
Eschrichtius robustus
Caperea marginata
Eubalaena japonica
Eubalaena glacialis
Eubalaena australis
Balaena mysticetus
30
20
10
0

9.4 RPANDA

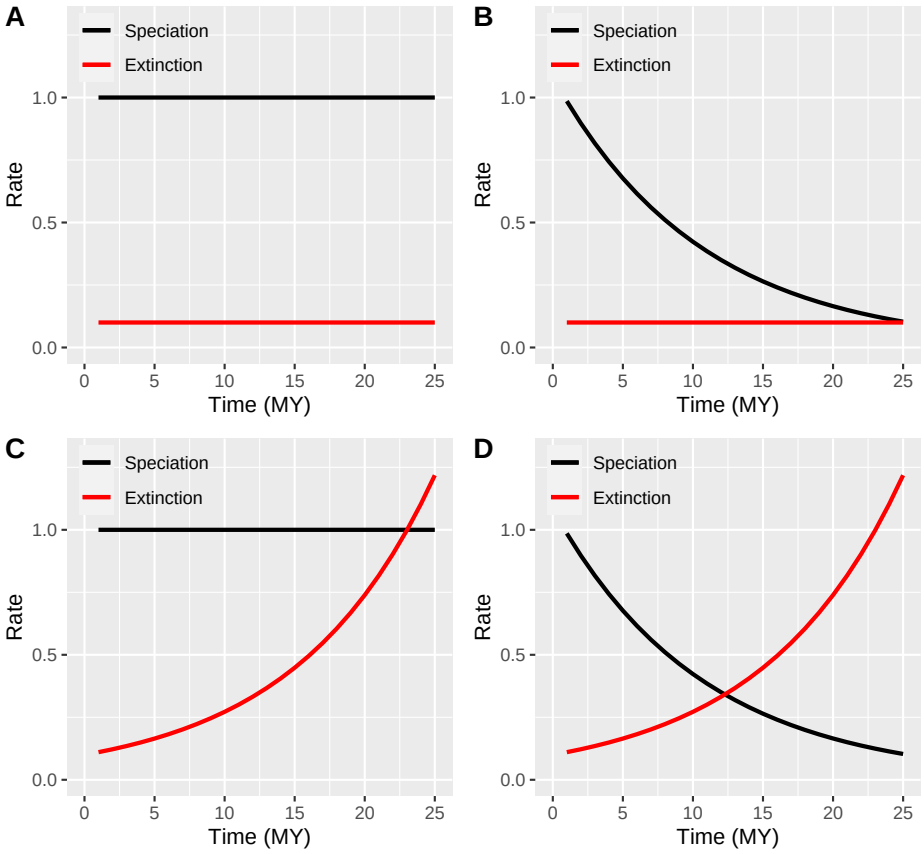
The package **RPANDA** [Morlon et al., 2016] allows us to fit different defined models of variation in the rate of birth and death through time. We can then select the best model by maximum likelihood. This is a good way of testing an *a priori* hypothesis we may have about diversification in a lineage.

The first step is to define a few functions that describe our hypotheses. First we need to define speciation rate (λ) as either constant (**lambda.cst**) or varying exponentially (**lambda.var**). We can do the same for extinction rate (μ).

$$\lambda_{(t)} = \lambda_{(0)} e^{\alpha t} \qquad \mu_{(t)} = \mu_{(0)} e^{\beta t}$$

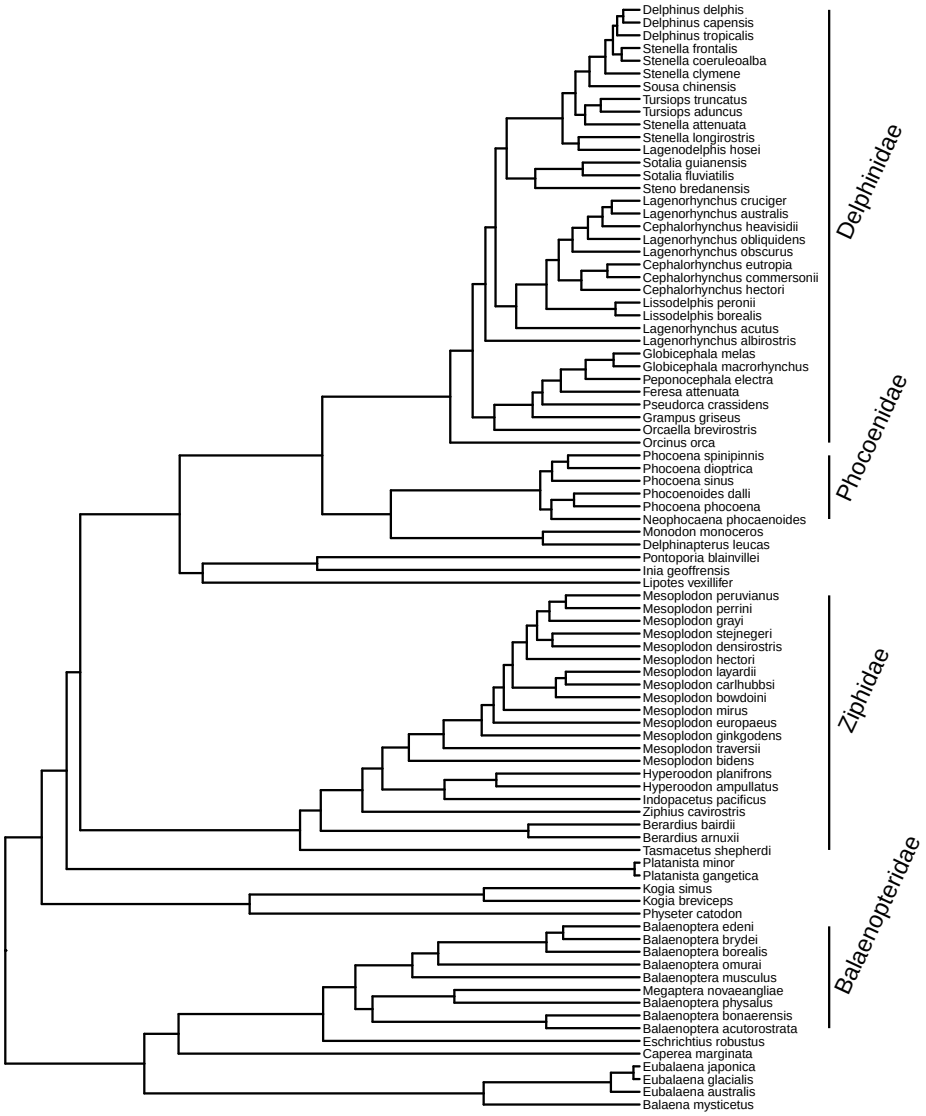
```
lambda.cst <- function(x,y){y}
lambda.var <- function(x,y){y[1]*exp(y[2]*x)}
mu.cst <- function(x,y){y}
mu.var <- function(x,y){y[1]*exp(y[2]*x)}
```

We will test 4 different combinations of these scenarios. I find it easier to visualise models like this graphically so these plots describe the various models.



In RPANDA, we need to specify which clades are most likely to have diversification shifts to test our hypotheses. For this example, we will look at 4 major radiations of cetaceans. These are; *Balaenopteridae, Ziphidae, Phocoenidae* and *Delphinidae*.

```
library(RPANDA)
```



We can extract the relevant sub-trees using **extract.clade** in **ape** [Paradis and Schliep, 2018] in combination with **MRCA** in **phylobase** [R Hackathon et al., 2019]. Don't forget to also make another subtree of all the species not in those groups.

```
delphinidae.tree <- extract.clade(whales, node = phylobase::MRCA(whales, tip = c("Orci
phocoenidae.tree <- extract.clade(whales, node = phylobase::MRCA(whales, tip = c("Phoc
ziphidae.tree <- extract.clade(whales, node = phylobase::MRCA(whales, tip = c("Mesoplo
balaenopteridae.tree <- extract.clade(whales, node = phylobase::MRCA(whales, tip = c("
```

```
othercetaceans.tree <- drop.tip(whales,c(balaenopteridae.tree$tip.label,
                                         delphinidae.tree$tip.label,
                                         phocoenidae.tree$tip.label,
                                         ziphidae.tree$tip.label))
```

Now we can fit some birth-death models! Let's start with both parameters constant (**bcstdcst**). The function we need is **fit_bd**. This function requires a lot of information so let's take it slowly.

The first argument is just our tree (*phylo*). Then we need to include the total length of the tree (*tot_time*). We don't know this offhand but we can calculate it using **max(branching.times(TREE))**. The arguments *f.lamb* and *f.mu* take our previously defined functions for λ (birth rate) and μ (death rate). *lamb_par* and *mu_par* similarly take our initial values for λ and μ which here, will be 0.4 and 0 respectively. **cst.lamb** and **cst.mu** are both logical (they can be true or false) and ask if λ and μ are constant (they are). The argument f is the fraction of extant species included in the whole phylogeny. Here we have 87 out of the 89 cetacean species. Finally, **dt** is set to the recommended value in the vignette for the function. In short, **dt** is the length of the time interval on which functions are assumed to be constant. Smaller values mean a finer-grain analysis and longer computing times. Larger values mean the analysis will be quicker but lower resolution.

```
bcstdcst <- fit_bd(phylo = delphinidae.tree,
                   tot_time = max(branching.times(delphinidae.tree)),
                   f.lamb = lambda.cst, f.mu = mu.cst,
                   lamb_par = 0.4, mu_par = 0,
                   cst.lamb = TRUE, cst.mu = TRUE,
                   f = 87/89, dt = 1e-3)
```

The analysis has returned a number of details. Most importantly for us, the likelihood and AICc will allow for model comparison.

```
bcstdcst
```

```
model :
[1] "birth death"

LH :
[1] -83.51998

aicc :
[1] 171.415

lamb_par :
[1] 0.2189443

mu_par :
```

```
[1] 2.413309e-09
```

Let's now evaluate the other 3 models using the same function. Be mindful of the changes in argument for each model. Especially the **expo.** and **cst.** arguments which differ depending on the model.

```
bvardcst <- fit_bd(phylo = delphinidae.tree,
                   tot_time = max(branching.times(delphinidae.tree)),
                   f.lamb = lambda.var, f.mu = mu.cst,
                   lamb_par = c(0.4,-0.05), mu_par = 0,
                   expo.lamb = TRUE, cst.mu = TRUE,
                   f = 87/89, dt = 1e-3)
bcstdvar <- fit_bd(phylo = delphinidae.tree,
                   tot_time = max(branching.times(delphinidae.tree)),
                   f.lamb = lambda.cst, f.mu = mu.var,
                   lamb_par = 0.4, mu_par = c(0.1,0.05),
                   cst.lamb = TRUE, expo.mu = TRUE,
                   f = 87/89, dt = 1e-3)
bvardvar <- fit_bd(phylo = delphinidae.tree,
                   tot_time = max(branching.times(delphinidae.tree)),
                   f.lamb = lambda.var, f.mu = mu.var,
                   lamb_par = c(0.4, -0.05), mu_par = c(0.1, 0.05),
                   expo.lamb = TRUE, expo.mu = TRUE,
                   f = 87/89, dt = 1e-3)
```

We can put together a nice table to present the results. But in truth, all we really need is the AICc so pay close attention to that.

```
results <- data.frame("bcstdcst" = c(bcstdcst$LH[1], bcstdcst$aicc[1]),
                      "bvardcst" = c(bvardcst$LH[1], bvardcst$aicc[1]),
                      "bcstdvar" = c(bcstdvar$LH[1], bcstdvar$aicc[1]),
                      "bvardvar" = c(bvardvar$LH[1], bvardvar$aicc[1]))
rownames(results) <- c("LH", "AICc")
knitr::kable(results)
```

	bcstdcst	bvardcst	bcstdvar	bvardvar
LH	-83.51998	-81.62979	-81.93556	-80.99625
AICc	171.41497	170.03377	170.64532	171.32584

The table shows us that the best AICc for Delphinidae is the *bvardcst* model, although there's barely anything in it. You may note that by a mathematical quirk (see the help for this function for details) the μ for this model is negative. Clearly this can't be the case! This can happen sometimes with this calaculation but all rates should be interpreted as *absolute* values. In other words, ignore the sign.

```
bvardcst$mu_par
```

```
[1] -1.521115e-07
```

Table 9.1: AICc scores for 4 different birth-death models fitted to five cetacean sub-trees.

	Balaenopteridae	Delphinidae	Phocoenidae	Ziphidae	Other Cetaceans
bcstdcst	56.73336	171.4150	31.79207	132.6440	119.4690
bvardcst	58.44456	170.0338	34.13348	130.1843	117.5562
bcstdvar	58.32515	170.6453	41.79207	127.6094	115.3723
bvardvar	65.67244	171.3258	64.13348	133.2251	118.9789

Table 9.2: Estimates of speciation and extinction rates from birth-death modelling of cetacean evolution

	Balaenopteridae	Delphinidae	Phocoenidae	Ziphidae	Other.Cetaceans
lambda	0.0734308	0.1405102	0.1410234	0.0948515	0.1857764
	NA	0.1257642	NA	NA	NA
mu	0.0000000	-0.0000002	-0.0000001	0.0000001	0.8313274
	NA	NA	NA	NA	-0.1742352

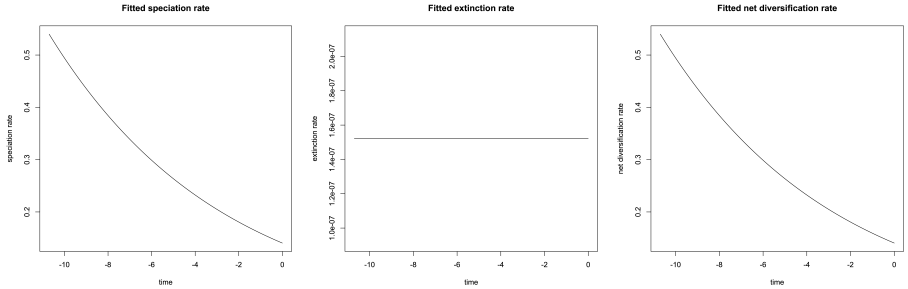
Next up, we need to evaluate the other sub-trees. To do this the way I've done above would take a lot of time and code. So I'm going to use a shortcut and write a function to run it all for me. The full code to do this is taken from this online exercise. The code to extract the results is also available there. It's not shown here to keep things clear.

As before, the first thing to look at is the AICc scores for each model and each tree. We need to select the best model for each sub-tree by choosing the AICc closest to 0.

The final step is to look at the extracted parameters from the favoured models. The top two rows show birth rates and the bottom two show death rates. Where two values are given for a parameter, remember this is because the rate was variable in the model. The first value is the maximum parameter estimate at time $t = 0$ (the present). The second value is the rate of change of the parameter running from the present into the past.

Focusing in on the Delphinidae tree again, we can begin to plot some interesting features. The function **plot_fit_bd** will produce three plots describing the speciation, extinction and diversification rates through time.

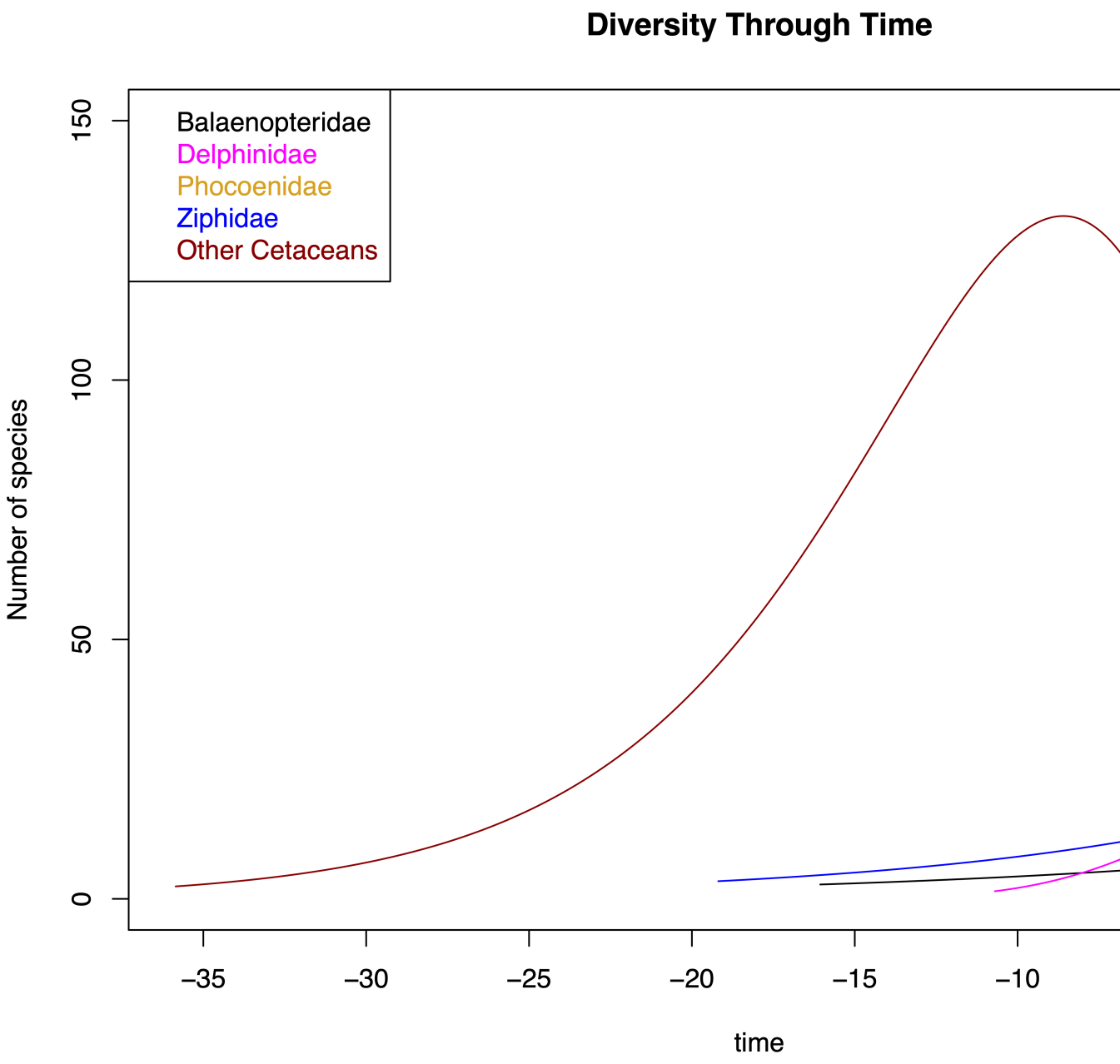
```
plot_fit_bd(results$delphinidae.res$bvardcst, max(branching.times(delphinidae.tree)))
```



The function **plot_dtt** will produce a plot of diversity through time. Here we can see how Delphinidae diversity has increased over the tree.

```
plot_dtt(results$delphinidae.res$bvardcst,
         max(branching.times(delphinidae.tree)),
         N0 = Ntip(delphinidae.tree))
```

Let's compare the diversification in all the lineages we've analysed. To do so easily, we would need to edit the **plot_dtt** function. Like so much else, this has already been done for us and the code can be found here.



We can see the rise of each lineage with the start of each line. We can begin to pick out patterns of interest. For example, the diversity of "other cetaceans"

increases markedly but begins to decline just as Delphinidae begin to diversify. We can also see that the rate of diversification of Delphinidae is greater than other lineages, perhaps suggesting we might like to look at why the oceanic dolphins diversified so quickly!

9.5 BAMM & BAMMtools

BAMM stands for Bayesian Analysis of Macroevolutionary Mixtures and is an open source piece of software. BAMM uses a process called reversible-jump Markov Chain Monte-Carlo (rjMCMC) simulation to estimate diversification rates [Rabosky, 2014]. The software and accompanying documentation can be found here.

To use BAMM, you must first download the program following these instructions. For this demonstration, we will use the same cetacean tree we have been using in this chapter as well as some pre-prepared instructions for BAMM to run. Both are available here.

Run the analysis by storing your tree and the *divcontrol.txt* file in the same directory as BAMM and then run this command from your terminal.

```
./bamm -c divcontrol.txt
```

Once run, you should see a number of files appear in the same directory. This is your output and this is where **BAMMtools** comes in [Rabosky et al., 2014]. BAMMtools is a package designed to interpret the outputs of BAMM analyses. Let's see how it works.

The file to import into R is the *event_data.txt* file stored in the BAMM directory.

```
library(BAMMtools)
events <- read.csv("event_data.txt")
```

For this demonstration, I will load the example data from the package. It ran on the same tree but is from a longer MCMC chain and so is likely to be more informative as well as being a closer representation of what you are likely to want to run in a proper analysis.

```
data(whales, events.whales)
```

We use the function **getEventData** to extract the relevant data from the report. This creates an object of class **bammdata** which is essential for most functions in BAMMtools.

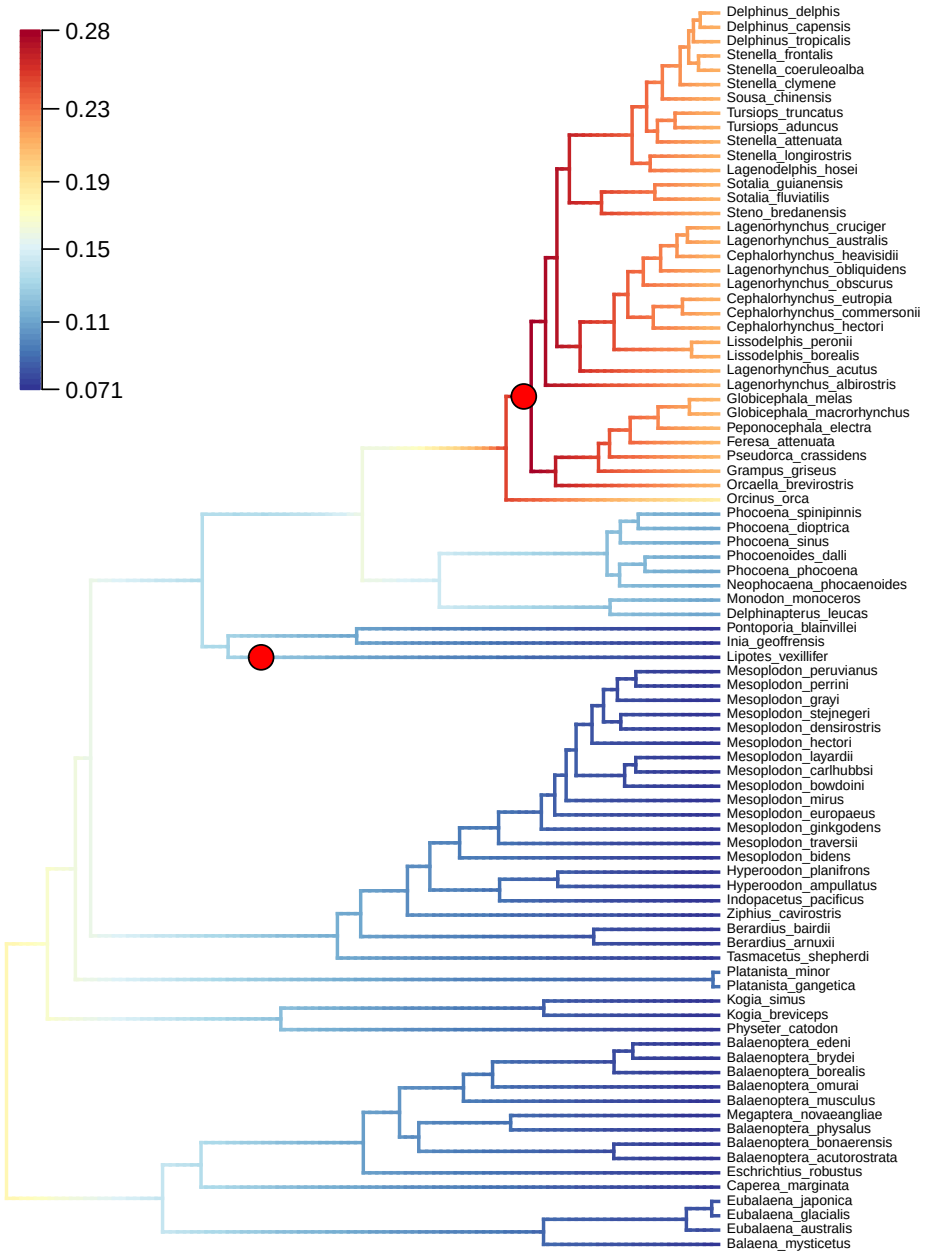
```
ed <- getEventData(whales, events.whales, burnin = 0.25)
```

```
Processing event data from data.frame

Discarded as burnin: GENERATIONS <  2495000
```

```
Analyzing  1501  samples from posterior

Setting recursive sequence on tree...

Done with recursive sequence
```

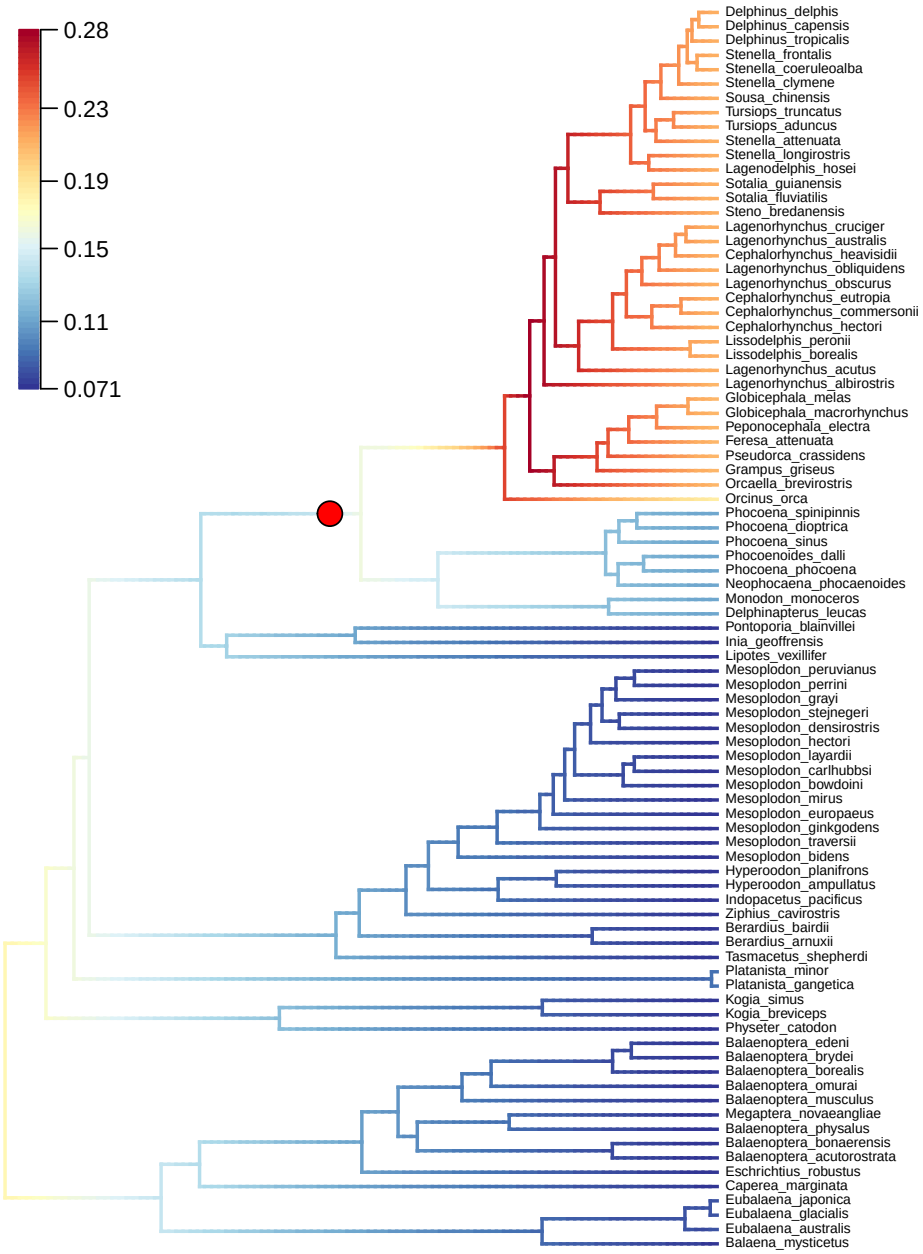
You are certainly going to want to plot your analysis. We can do so by using the function **plot.bammdata**.

```
par(mar = c(0,0,0,0))
bamm.whales <- plot.bammdata(ed, lwd=2, labels = T, cex = 0.5)
addBAMMshifts(ed, cex=2)
addBAMMlegend(bamm.whales, nTicks = 4)
```



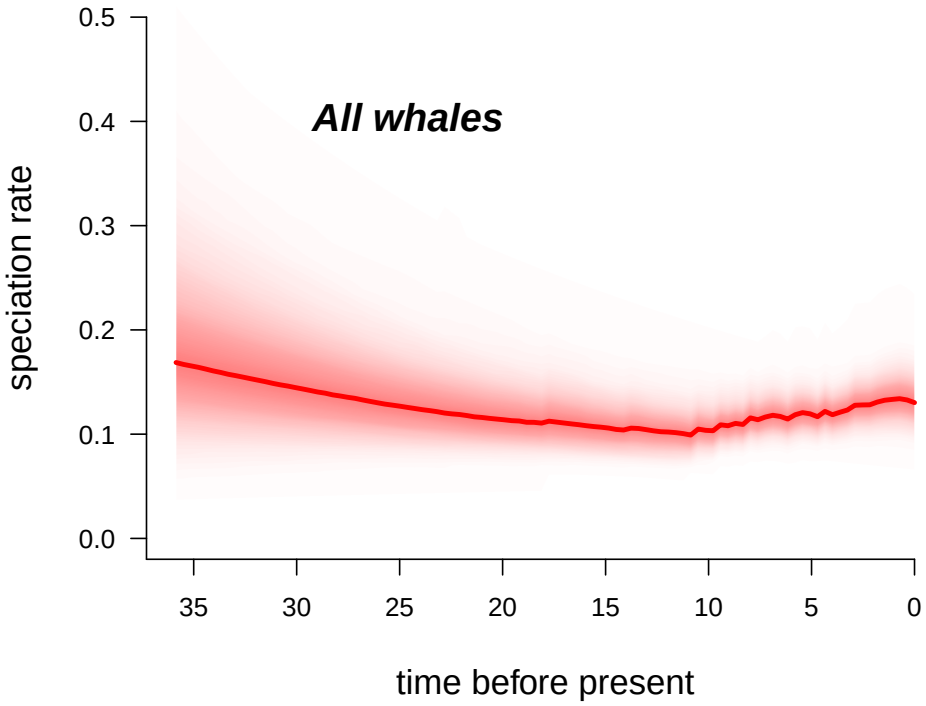
BAMM has clearly identified a shift early in the history of Delphinidae, matching some other methods we've used. Also in this plot, we can see a potentially spurious shift on the branch leading to *Lipotes vexillifer*. Remember that this is just one sample from the posterior distribution and another one would be different. The colour of the branches tells us the story of the whole sample.

```
par(mar = c(0,0,0,0))
bamm.whales <- plot.bammdata(ed, lwd=2, labels = T, cex = 0.5)
addBAMMshifts(ed, cex=2, index = 42)
addBAMMlegend(bamm.whales, nTicks = 4)
```



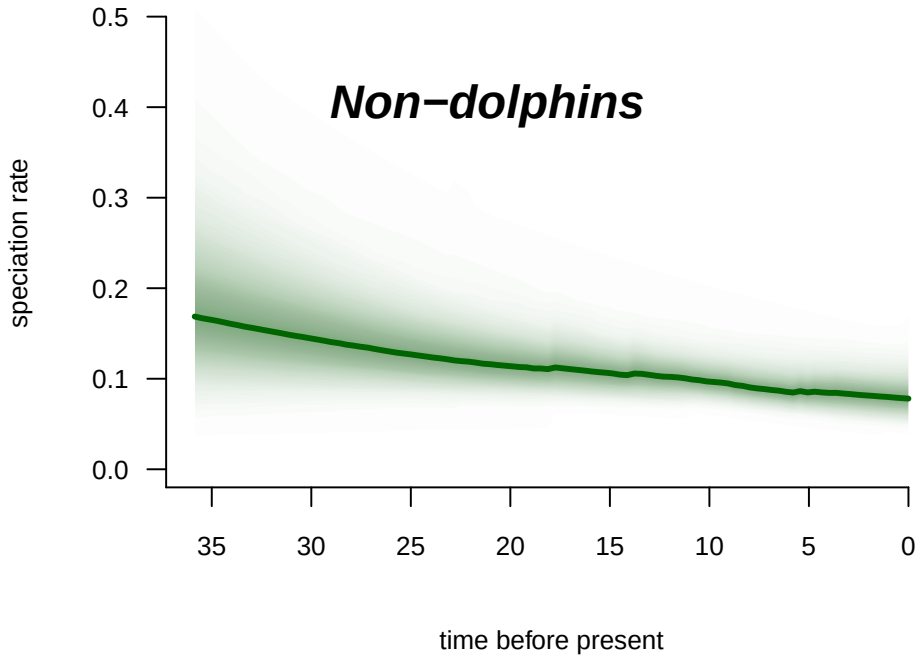
Let's plot the speciation rate through time for some lineages of interest. We can use the function **plotRateThroughTime** for this.

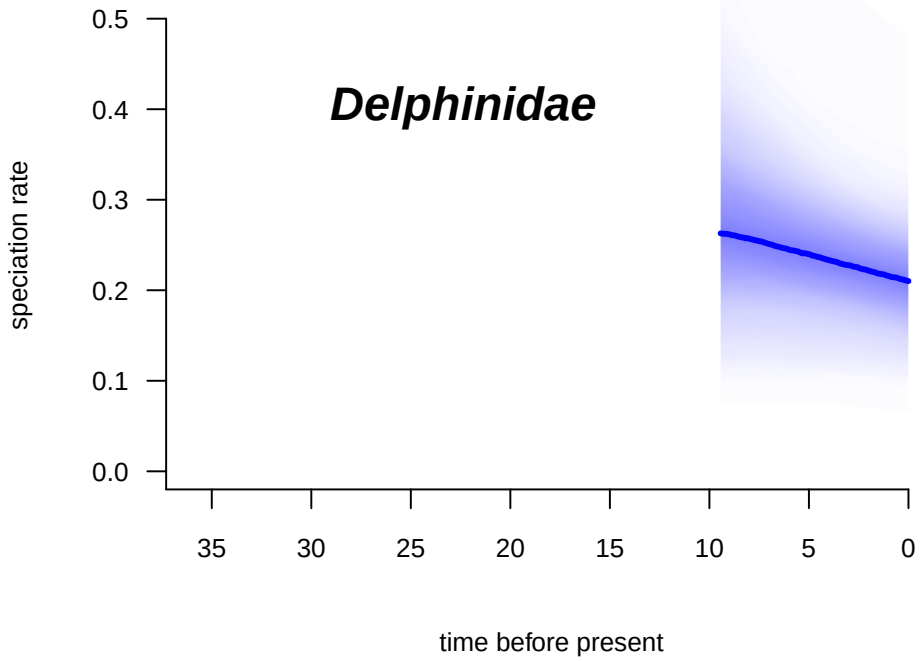
```
plotRateThroughTime(ed, intervalCol="red", avgCol="red", ylim=c(0,.5))
text(x=30, y= 0.4, label="All whales", font=4, cex=1.5, pos=4)
```



We can also compare lineages. Here we are plotting the speciation rate in dolphins against the rest of cetaceans.

```
par(mfrow = c(2,1), mar = c(0,0,0,0))
plotRateThroughTime(ed, intervalCol="darkgreen", avgCol="darkgreen", node=141, nodetype
text(x=30, y= 0.4, label="Non-dolphins", font=4, cex=1.5, pos=4)
plotRateThroughTime(ed, intervalCol="blue", avgCol="blue", node=141, ylim=c(0,.5), cex
text(x=30, y= 0.4, label="Delphinidae", font=4, cex=1.5, pos=4)
```





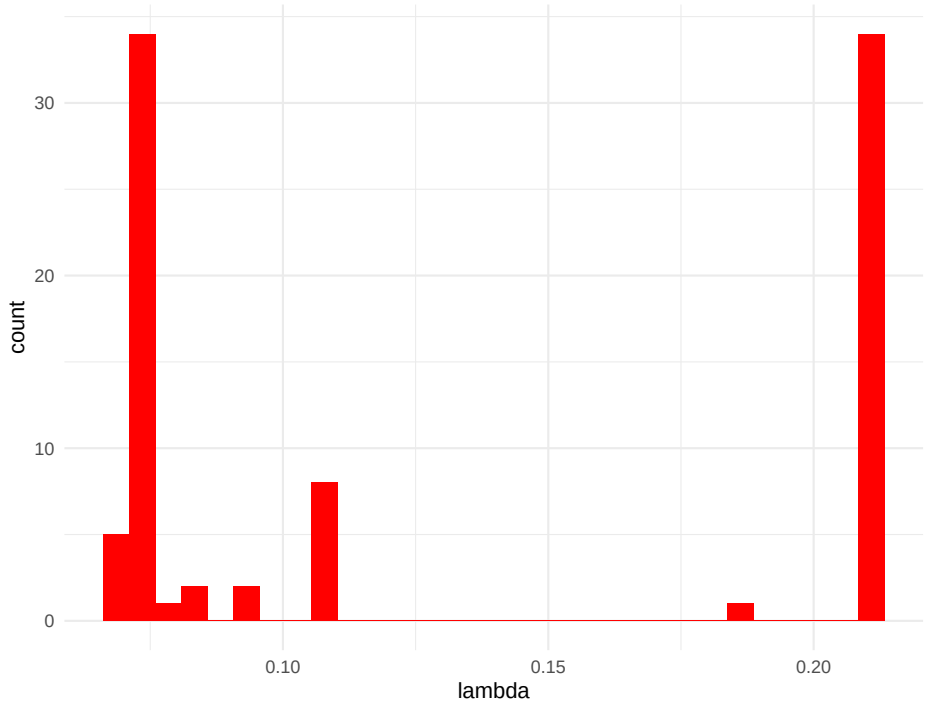
If we desire, we can extract the rates at the tips and analyse them using **get-**

TipRates.

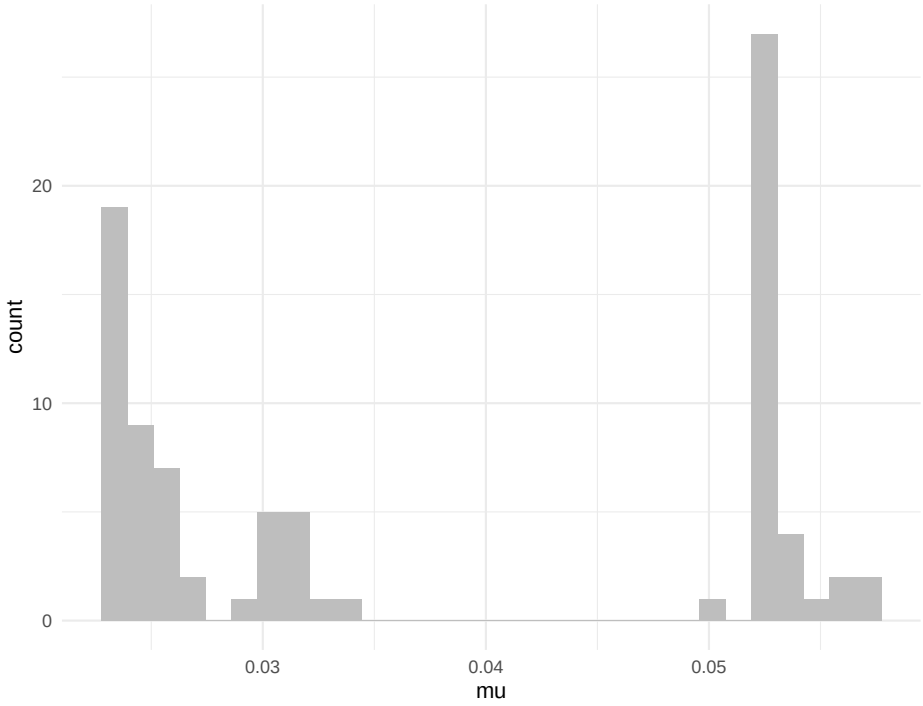
```
tip.rates <- getTipRates(ed)
```

A simple histogram of the speciation rates and extinction rates show strongly bimodal distributions in both cases.

```
require(ggplot2)
d <- data.frame("lambda" = tip.rates$lambda.avg)
ggplot(d, aes(x = lambda)) +
  geom_histogram(fill = "red") +
  theme_minimal()
```

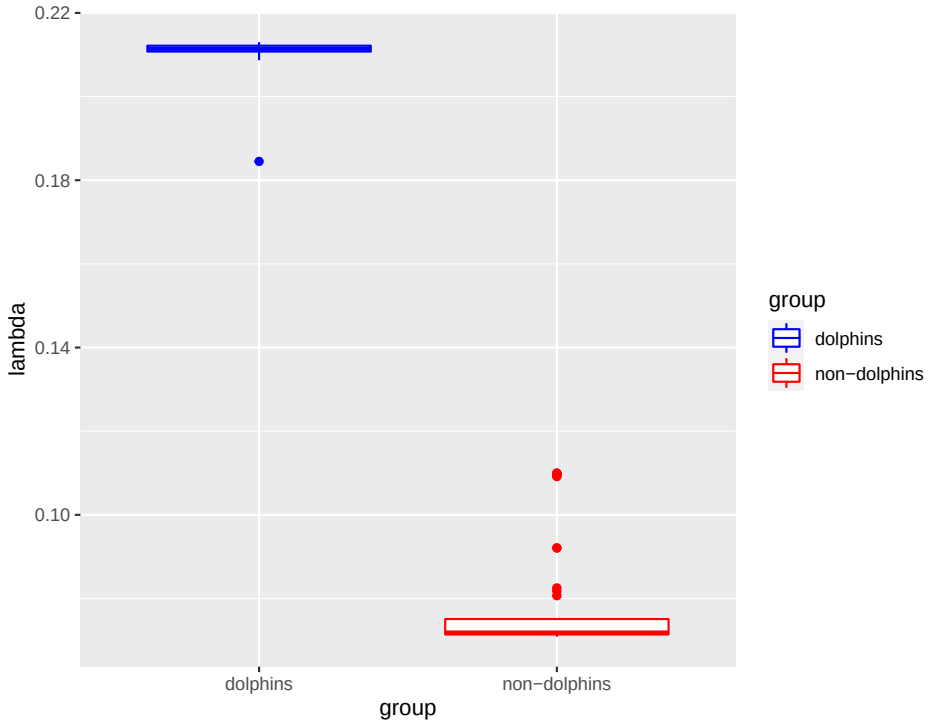



```
require(ggplot2)
d <- data.frame("mu" = tip.rates$mu.avg)
ggplot(d, aes(x = mu)) +
  geom_histogram(fill = "grey") +
  theme_minimal()
```



It appears on closer inspection that the differences are down to the rate shift in Delphinidae we have identified multiple times throughout this chapter.

```
d <- data.frame("lambda" = tip.rates$lambda.avg,
                "group" = c(rep("non-dolphins", 52), rep("dolphins", 35)))
ggplot(d, aes(y = lambda, x = group, colour = group)) +
  geom_boxplot() +
  scale_color_manual(values = c("blue", "red"))
```



9.6 Further info

There's an interesting post-script here. A recent study claimed to show that making inferences about diversification patterns using extant time-trees is a deeply flawed approach [Louca and Pennell, 2020]. This article claims that there are actually an infinite number of diversification scenarios that are equally likely to be the product of these types of analysis.

If correct, this is a *huge* problem for the study of diversification. The authors do suggest a way through however. In particular, they emphasise that the inclusion of paleontological data is crucially important to improve these methods. The methods covered in this chapter explicitly deal exclusively with ultrametric trees (no fossil data) and so we are a little stuck with this new criticism.

However, developers of comparative methods have been aware for some time that including fossil taxa in these analyses is important. For example, look back to the chapters on ancestral state reconstruction for a demonstration of how fossil data improves an analysis. Versions of these analyses that can deal with non-ultrametric trees are bound to help address the concerns of Louca and Pennell [2020].

Chapter 10

Trait Dependent Diversification

The previous chapter used a variety of methods to estimate the rate of diversification in a phylogeny. This is an interesting and contentious field of study which allows us to ask marcoevolutionary questions. In this chapter, we will extend this approach to model the effects of trait evolution on diversification.

10.1 Binary traits: BiSSE

BiSSE stands for Binary State Speciation and Extinction model.

Let's start by simulating a tree and binary character using the **tree.bisse** function in the package **diversitree** [FitzJohn, 2012]. We need to specify parameters for the simulation. These parameters are (in order);

- The speciation rate with the trait in state 0: λ_0
- The speciation rate with the trait in state 1: λ_1
- The extinction rate with the trait in state 0: μ_0
- The extinction rate with the trait in state 1: μ_1
- The probability of transitioning from state 0 to 1: q_{01}
- The probability of transitioning from state 1 to 0: q_{10}

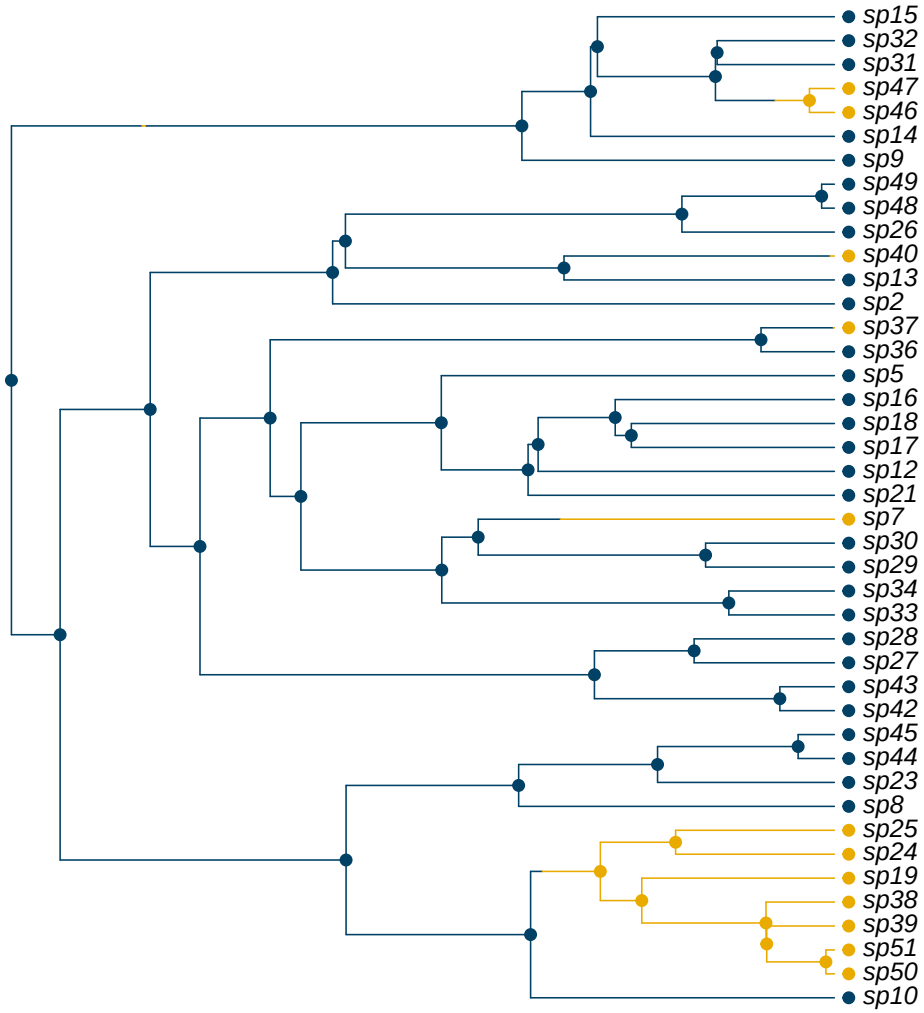
For this example, we will imagine a scenario in which lineages are twice as likely to speciate with the trait in state 1 than in state 0 ($\lambda_1 = 2 \times \lambda_0$). Extinction rates will not be different in either state ($\mu_0 = \mu_1$) and the probability of transitioning between states is the same in either direction ($q_{01} = q_{10}$).

```
library(diversitree)
pars <- c(.1,.2,.03,.03,.01,.01)
```

```
set.seed(42)
phy <- tree.bisse(pars, max.t = 30, x0 = 0)
states <- phy$tip.state
```

We can extract the history of the trait's evolution using **history.from.sim.discrete**. Note that although this looks like an ancestral state reconstruction, it isn't. This is the *actual* history of the trait we have simulated.

```
plot(history.from.sim.discrete(phy, 0:1),
     phy, col=c("#004165", "#eaab00"), no.margin = TRUE)
```



Next we use the function **make.bisse** with the tree and the named vector of states. This creates a likelihood function that we will go on to use in our analysis.

```
lik <- make.bisse(phy, states)
```

To perform a maximum likelihood search, we will also need to select a starting point. The function **starting.point.bisse** produces a guess for a starting point based on the *character-independent* birth-death fit.

```
p <- starting.point.bisse(phy)
p
```

```
   lambda0    lambda1        mu0        mu1        q01        q10
0.09380372 0.09380372 0.00000000 0.00000000 0.01876074 0.01876074
```

Now that we have our likelihood function and our starting point for each parameter, we can start the likelihood search using **find.mle**. Depending on the size and complexity of your tree and data, this step may take a while.

```
fit <- find.mle(lik, p)
```

The resulting object contains the log likelihood of the model (**lnLik**) and the estimates for all of the parameters.

```
fit$lnLik
```

```
[1] -152.5654
```

```
round(coef(fit),3)
```

```
lambda0 lambda1     mu0     mu1     q01     q10
  0.089   0.150   0.000   0.052   0.018   0.022
```

In this case we can compare these parameters to the known values we set for the simulation. The estimates are not exactly the same and perhaps a better set of starting estimates would be helpful. However, we can see that $\lambda_1 > \lambda_0$ as we know it should be.

Let's look at testing the hypothesis that speciation rates are different when the trait we are studying is in different states. We can do this by constraining the speciation rates to be equal in a new model using **constrain**.

```
lik1 <- constrain(lik, lambda1~lambda0)
fit1 <- find.mle(lik1, p[argnames(lik1)])
fit1$lnLik
```

```
[1] -153.0321
```

We can put the estimated parameters together in a table for ease of comparison.

```
knitr::kable(round(rbind(full=coef(fit), equal.l=coef(fit1, TRUE)), 3))
```

	lambda0	lambda1	mu0	mu1	q01	q10
full	0.089	0.150	0	0.052	0.018	0.022
equal.l	0.094	0.094	0	0.000	0.016	0.030

The final thing to do is test the differnece between the new and old models.

```
anova(fit, fit1)
```

```
        Df   lnLik    AIC   ChiSq Pr(>|Chi|)
full     6 -152.56 317.13
model 1  5 -153.03 316.06 0.93342      0.334
```

The ANOVA table tells us that there is no significant difference in fit between the two models ($\chi^2 = 0.93, p = 0.33$).

If you're not a fan of maximum likelihood estimation you can run the same analysis by MCMC. First we need to set a prior as the starting point for our analysis. The function **make.prior.exponential** will allow us to set an exponential prior of $\frac{1}{2r}$. Remember that $r = \lambda - \mu$.

```
prior <- make.prior.exponential(1/(2*(p[1] - p[3])))
```

In our final *mcmc* analysis we will need to provide an argument called simply **w**. This is a so-called tuning parameter for the sampling that the **mcmc** function will perform. The function uses a process called slice sampling [Neal, 2003]. In slice sampling, the parameter **w** affects how many function evaluations are required between sample updates. According to the documentation for the function, the optimal value for **w** is equal to the width of the high probability region we are searching through. The easiest way to work this out is to run a short chain and use the obseved range in the results.

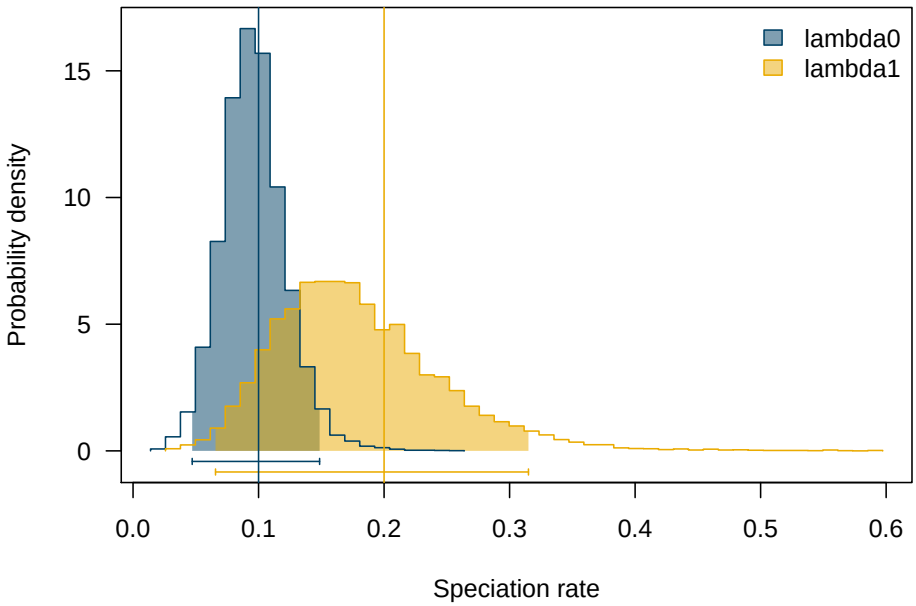
```
set.seed(42)
tmp <- mcmc(lik, fit$par, nsteps=100, prior=prior,
            lower=0, w=rep(1, 6), print.every=0)
w <- diff(sapply(tmp[2:7], range))
```

To run our full analysis, we will need to use a much longer chain. I've gone for 10,000 here. Be aware that this will take a little while.

```
samples <- mcmc(lik, fit$par, nsteps=10000, w=w,
                lower=0, prior=prior, print.every=0)
```

We can plot the distributions of λ_0 and λ_1 using the function **profiles.plot**. Here we can see the true values as lines. The shaded areas and bars along the x axis both represent the 95% confidence intervals of the sample.

```
col <- c("#004165", "#eaab00")
profiles.plot(samples[c("lambda0", "lambda1")], col.line=col, las=1,
              xlab="Speciation rate", legend="topright")
abline(v=c(.1, .2), col=col)
```



10.1.1 REALDATA EXAMPLE. SHOREBIRDS?

10.2 Multi-state traits: MuSSE

BiSSE works well with binary traits but what about categrorical traits with more than two states? For this we can use an extension of the BiSSE model called **MuSSE** (Multiple State Speciation and Extinction).

Let's start with a simulated example. The traits we simulate will have three states (coded numerically) and transitions are only possible between adjacent states (you can't go from state 1 to 3 or 3 to 1). All other transition rates will be equal. In this case we need to specify the parameters in the following order:

$$\lambda_1 \ \lambda_2 \ \lambda_3 \ \mu_1 \ \mu_2 \ \mu_3 \ q_{12} \ q_{13} \ q_{21} \ q_{23} \ q_{31} \ q_{32}$$

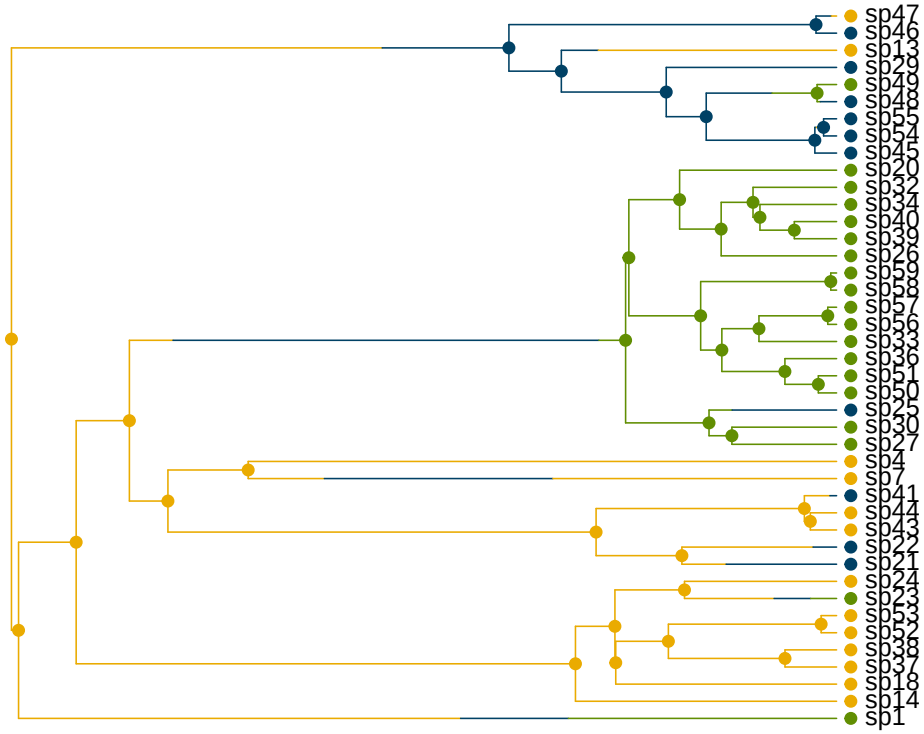
```
pars <- c(.1,.15, .2,        #lambda
          .03, .045, .06,    #mu
          .05, 0,            #q12 q13
          .05, .05,          #q21 q23
          0, 0.05)           #q31 q32
```

Now we can simulate our tree and data using **tree.musse**.

```
set.seed(2)
phy <- tree.musse(pars, 42, x0 = 1)
```

We can extract the history of the simulated trait with **history.from.sim.discrete** and plot it over the tree.

```
col <- c("#eaab00", "#004165", "#618e02")
h <- history.from.sim.discrete(phy, 1:3)
plot(h, phy, cex=1, col=col, no.margin=TRUE, font=1)
```



Next we must make the likelihood function in much the same way as we did for BiSSE but this time using the function **make.musse**.

```
states <- phy$tip.state
lik <- make.musse(phy, states, 3)
```

For our analysis, we'll start with a very simple model. The simplest model is one in which all three speciation rates are the same ($\lambda_1 = \lambda_2 = \lambda_3$), all 3 extinction rates are the same ($\mu_1 = \mu_2 = \mu_3$) and all the non-zero transition rates are the same ($q_{12} = q_{21} = q_{23} = q_{32}$). Also, remember that q_{13} and q_{31} are still both 0.

```
lik.base <- constrain(lik, lambda2 ~ lambda1, lambda3 ~ lambda1,
                      mu2 ~ mu1, mu3 ~ mu1,
                      q13 ~ 0, q21 ~ q12,
                      q23 ~ q12, q31 ~ 0, q32 ~ q12)
```

Just as with BiSSE, we need to set a starting point for the likelihood search, this time with the function **starting.point.musse**.

```
p <- starting.point.musse(phy, 3)
```

Now we can start the maximum likelihood search with **find.mle**.

```
fit.base <- find.mle(lik.base, p[argnames(lik.base)])
fit.base$lnLik
```

```
[1] -153.4905
```

```
round(coef(fit.base),3)
```

```
lambda1     mu1     q12
  0.201   0.139   0.055
```

To test if the speciation rate varies when the traits is in different states, we can run another model in which the values of λ_1, λ_2 and λ_3 are unconstrained.

```
lik.lambda <- constrain(lik, mu2 ~ mu1, mu3 ~ mu1,
                        q13 ~ 0, q21 ~ q12,
                        q23 ~ q12, q31 ~ 0, q32 ~ q12)
fit.lambda <- find.mle(lik.lambda, p[argnames(lik.lambda)])
```

In this case the new model is not significantly better than the minimal model ($\chi^2 = 2.67,\ p = 0.263$). Keep in mind that when we simulated the data, we specified that λ was indeed different between different states. It's probable that we haven't been able to detect this on such a small tree (n = 42).

```
anova(fit.base, free.lambda=fit.lambda)
```

```
            Df   lnLik    AIC  ChiSq Pr(>|Chi|)
minimal      3 -153.49 312.98
free.lambda  5 -152.16 314.32 2.6651     0.2638
```

Small trees are not good fits for these methods. Generally, the larger the tree the more power you have. Here is the model comparison for the same analysis with the same parameters but simulated on a tree with 1000 tips. As you can see, the analysis shows very clearly that the model in which λ is allowed to vary is a much better fit ($\chi^2 = 33.35,\ p < 0.001$).

	Df	lnLik	AIC	ChiSq	Pr(>|Chi|)
minimal	3	-3694.195	7394.390	NA	NA
free.lambda	5	-3677.520	7365.041	33.34883	1e-07

10.3 Quantitative traits: QuaSSE

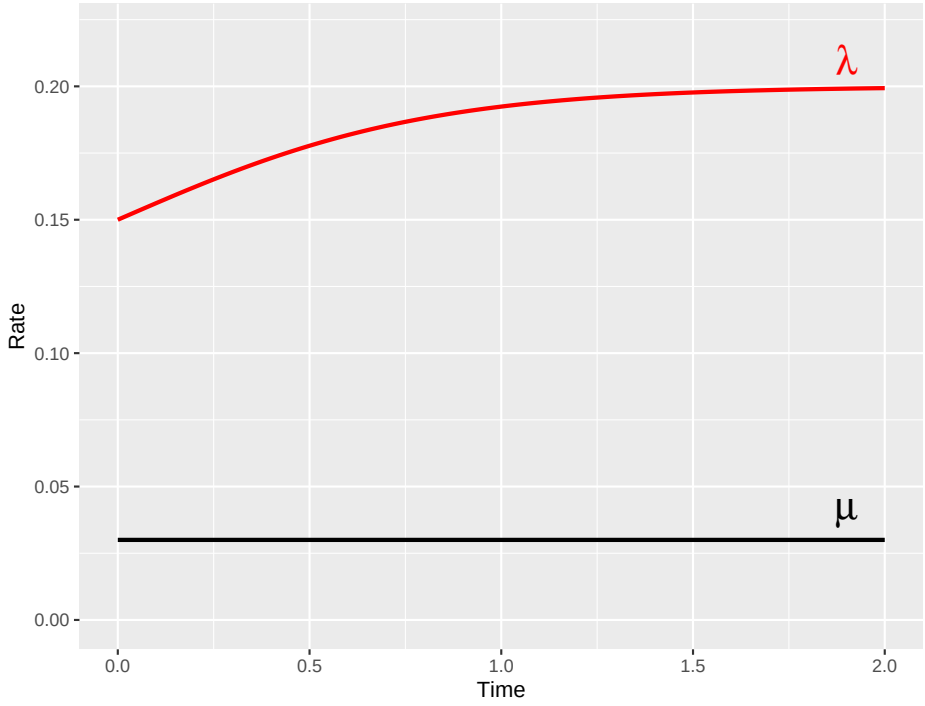
Dealing with quantitative traits in models like these is lsightly less straightforward. Categorical traits are well defined and we can model the transition rates

between states. Deciding when something like body mass has undergone significant evolutionary change is a little more difficult. This is the same problem we faced with continuous characters in ancestral state reconstruction and it will come up again when we look at convergence.

For diversification, we have the **QuaSSE** (Quantitative State Speciation and Extinction) model.

First we need to specify some functions to use. For lambda we will use a sigmoidal function with an inflection at x = 0. This means we will have an increasing rate of speciation reaching a plateau. We can specify this with the function **sigmoid.x**.

```
lambda <- function(x) sigmoid.x(x, 0.1, 0.2, 0, 2.5)
mu <- function(x) constant.x(x, 0.03)
```



Next we can specify the model of character evolution for our trait simulation. We will be going with Brownian motion and we will set the diffusion parameter to 0.025. To specify all this, we can use the function **make.brownian.with.drift**.

```
char <- make.brownian.with.drift(0, 0.025)
```

Now as with BiSSE and MuSSE, we can simulate the tree and data using **tree.quasse**.

```
set.seed(1)
phy <- tree.quasse(c(lambda, mu, char), max.taxa=15, x0=0, single.lineage=FALSE)
```

Extract the trait states and specify the standard deviation. For this example, assume the standard deviation for all tips is 1/200.

```
states <- phy$tip.state
states.sd <- 1/200
```

Next we create the likelihood function, this time specifying the speciation and extinction functions.

```
lik <- make.quasse(phy, states, states.sd, sigmoid.x, constant.x)
```

Now to calculate the starting point. This is a little more involved than BiSSE and MuSSE! **starting.point.quasse** gives us constant rates for each parameter.

```
p <- starting.point.quasse(phy, states)
p
```

```
    lambda         mu  diffusion
0.16107838 0.02569057 0.03164062
```

Let's ignore drift for our first model (**drift** ~ **0**). The function **argnames** will return the names of parameters we will need to supply.

```
lik.nodrift <- constrain(lik, drift ~ 0)
argnames(lik.nodrift)
```

```
[1] "l.y0"      "l.y1"      "l.xmid"    "l.r"       "m.c"       "diffusion"
```

Next, we select the starting point values. For **l.y0** and **l.y1**, we will take the suggested value of λ from **starting.point.quasse** (**p[1]**). For **l.xmid** we will take the mean of the state values (**mean(states)**). **l.r** is set at 1. **m.c** (μ) and **diffusion** are taken straight form **p** (**p[2:3]**)

```
p.start <- c(p[1], p[1], mean(states), 1, p[2:3])
names(p.start) <- argnames(lik.nodrift)
p.start
```

```
      l.y0       l.y1     l.xmid        l.r        m.c  diffusion
0.16107838 0.16107838 0.57097062 1.00000000 0.02569057 0.03164062
```

One final thing before we run the search. We now need to specify the lower bounds for the search for each parameter..

```
lower <- c(0, 0, min(states), -Inf, 0, 0)
```

Finally we are ready to run our analysis! As with BiSSE and MuSSE, we will do so with **find.mle**. This will take some time.

```
fit <- find.mle(lik.nodrift, p.start,
                control=list(parscale=.1),
                lower=lower, verbose=0)
round(coef(fit),3)
```

```
    l.y0     l.y1    l.xmid      l.r      m.c diffusion
   0.000    0.220     0.238 28961.248    0.000     0.029
```

Let's compare this model to one in which λ is constant.

```
lik.constant <- constrain(lik.nodrift,
                          l.y1 ~ l.y0,
                          l.xmid ~ 0,
                          l.r ~ 1)
fit.constant <- find.mle(lik.constant,
                         p.start[argnames(lik.constant)],
                         control=list(parscale=.1),
                         lower=0, verbose=0)
knitr::kable(anova(fit, constant=fit.constant))
```

	Df	lnLik	AIC	ChiSq	Pr(>|Chi|)
full	6	-51.73395	115.4679	NA	NA
constant	3	-55.15222	116.3044	6.836543	0.0772943

Once again we see no significant difference ($\chi^2 = 6.84$, $p = 0.08$). Again this probably due to the small tree and a larger one would give us more power.

10.3.1 QuaSSE primate example

To demonstrate how QuaSSE can be used in research, let's investigate trait dependent diversification in primates using some existing data on primate body mass [Fitzjohn, 2010].

```
phy <- read.nexus("Vos-2006.nex")
d <- read.csv("Redding-2010.csv")
```

We will need to log-transform mass and we will also assume the standard deviation is $\frac{1}{50}$.

```
mass <- log(d$mass)
names(mass) <- d$tip.label
mass.sd <- 1/50
```

Starting point parameter estimates.

```
p <- starting.point.quasse(phy, mass)
p
```

```
    lambda         mu  diffusion
```

```
0.19072726 0.11034810 0.03251953
```

Now we will create a piecewise linear function using **make.linear.x**. The function is linear between **xr**[**1**] and **xr**[**2**] and flat outside this range.

```
xr <- range(mass) + c(-1,1) * 20 * p["diffusion"]
linear.x <- make.linear.x(xr[1], xr[2])
```

Now let's create a shortcut because we will be analysing several models. First we will create a function that will take our speciation and extinction functions and buld our model for us using **make.quasse**. The second function simply constrains drift to zero.

```
make.primates <- function(lambda, mu)
  make.quasse(phy, mass, mass.sd, lambda, mu)
nodrift <- function(f)
  constrain(f, drift ~ 0)
```

Now we can use these functions to build our likelihood functions. We are keeping μ constant in each case.

```
f.c <- make.primates(constant.x, constant.x)
f.l <- make.primates(linear.x, constant.x)
f.s <- make.primates(sigmoid.x, constant.x)
f.h <- make.primates(noroptimal.x, constant.x)
```

We will start by fitting the constant model (**f.c**) with no drift.

```
control <- list(parscale=.1, reltol=0.001)
mle.c <- find.mle(nodrift(f.c), p, lower=0, control=control, verbose=0)
```

Next we calculate starting points for our other models.

```
p.c <- mle.c$par
p.l <- c(p.c[1], l.m=0, p.c[2:3])
p.s <- p.h <- c(p.c[1], p.c[1], mean(xr), 1, p.c[2:3])
names(p.s) <- argnames(nodrift(f.s))
names(p.h) <- argnames(nodrift(f.h))
```

Once we have our starting points, we are ready to fit each of the models. Each of these lines may take a while to run.

```
mle.l <- find.mle(nodrift(f.l), p.l, control=control, verbose=0)
mle.s <- find.mle(nodrift(f.s), p.s, control=control, verbose=0)
mle.h <- find.mle(nodrift(f.h), p.h, control=control, verbose=0)
anova(mle.c, linear=mle.l, sigmoidal=mle.s, hump=mle.h)
```

	Df	lnLik	AIC	ChiSq	Pr(>|Chi|)
minimal	3	-841.4029	1688.806	NA	NA
linear	4	-836.0697	1680.139	10.66629	0.0010911
sigmoidal	6	-832.6038	1677.208	17.59805	0.0005323
hump	6	-829.0036	1670.007	24.79861	0.0000170

The ANOVA table shows us that the best fit is the *hump* model where speciation rate follows a hump shaped fit.

The next lines (which will again take some time) will start with parameters from the previous constrained models and add the drift parameter.

```
mle.d.l <- find.mle(f.l, coef(mle.l, TRUE), control=control, verbose=0)
mle.d.s <- find.mle(f.s, coef(mle.s, TRUE), control=control, verbose=0)
mle.d.h <- find.mle(f.h, coef(mle.h, TRUE), control=control, verbose=0)
```

We should add these new models to the ANOVA table to compare them all. We can see that in all cases the fit of the model is improved by the addition of drift.

```
knitr::kable(anova(mle.c, linear=mle.l, sigmoidal=mle.s,
                   hump=mle.h, drift.linear=mle.d.l,
                   drift.sigmoidal=mle.d.s, drift.hump=mle.d.h))
```

	Df	lnLik	AIC	ChiSq	Pr(>|Chi|)
minimal	3	-841.4029	1688.806	NA	NA
linear	4	-836.0697	1680.139	10.66629	0.0010911
sigmoidal	6	-832.6038	1677.208	17.59805	0.0005323
hump	6	-829.0036	1670.007	24.79861	0.0000170
drift.linear	5	-832.5001	1675.000	17.80563	0.0001360
drift.sigmoidal	7	-830.2691	1674.538	22.26745	0.0001773
drift.hump	7	-825.3671	1664.734	32.07150	0.0000018

When we extract the drift parameters, we can see that in all cases they are positive, indicating that mass is increasing over the tree on average.

```
c(linear=coef(mle.d.l)[["drift"]],
  sigmoidal=coef(mle.d.s)[["drift"]],
  hump=coef(mle.d.h)[["drift"]])
```

```
    linear  sigmoidal       hump
0.10458477 0.06776915 0.07929087
```

So the hump model with drift is the best fit to the data. The parameters of this model tell us that λ peaks around a log body mass value of 8.43 (**l.xmid**) and a variance of 0.12 (**l.s2**).

```
coef(mle.d.h)
```

When we plot the data alongside the tree, we can see that many of the species that fall within 2 standard deviations of the value of **l.xmid** are within the Cercopithecidae.

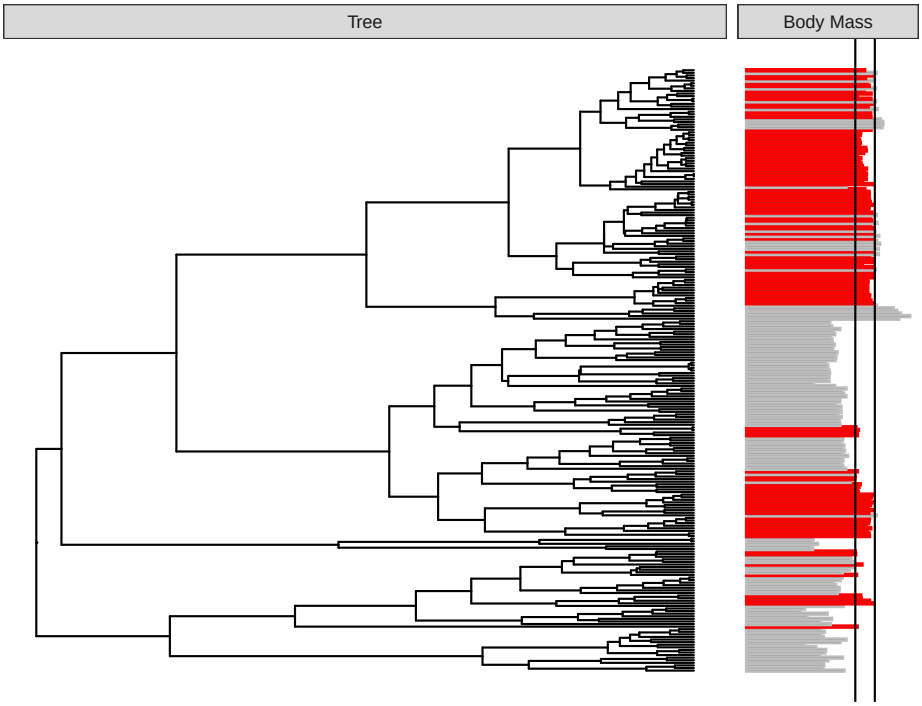


Figure 10.1: Body mass plotted alongside the primate phylogenetic tree. Species with a body mass in the range identified as having higher speciation rates are plotted in red

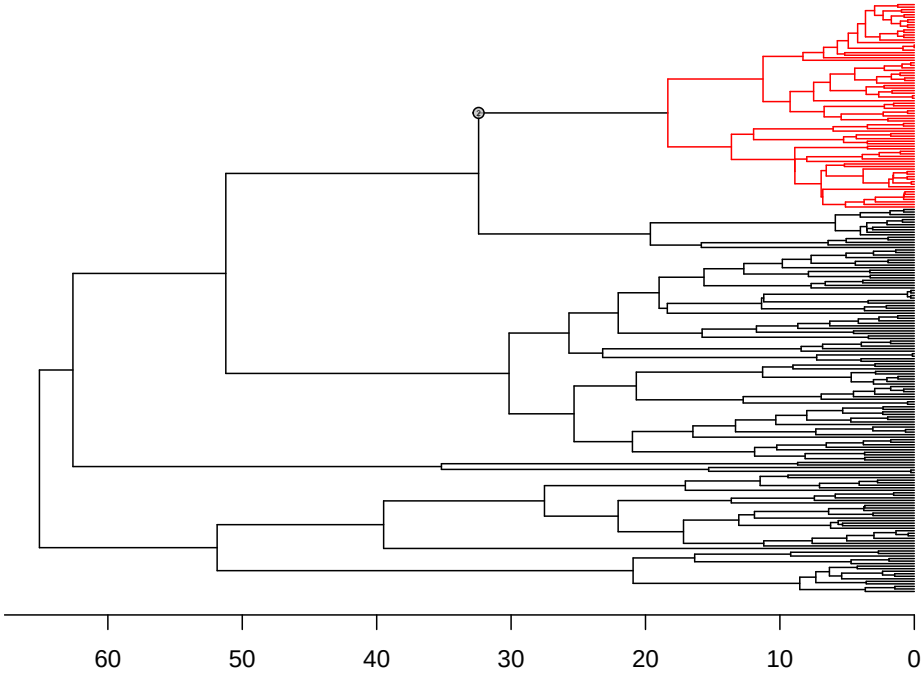
This suggests we might want to split the tree here and evaluate different models for each section of the tree much like we did with cetaceans in the previous chapter. In fact, the output of analysis with MEDUSA supports the idea that there has been a shift in diversification in the Cercopithecinae.

```
library(geiger)
m1 <- medusa(phy, model = "bd")
```

```
Appropriate  aicc-threshold for a tree of 233 tips is: 6.687066.

Step 1: lnLik=-691.8978; aicc=1387.821; model=bd
Step 2: lnLik=-678.3882; aicc=1366.907; shift at node 386; model=bd; cut=node; # shifts=1

No significant increase in aicc score. Disregarding subsequent piecewise models.

Calculating profile likelihoods on parameter values.

  Model.ID Shift.Node Cut.At Model Ln.Lik.part         r   epsilon     r.low
1        1        234   node    bd   -491.4083 0.0756587 0.4497380 0.0627243
```

```
2        2        386   node    bd   -186.9799 0.2446490 0.0745145 0.1937557
     r.high   eps.low  eps.high
1 0.0903268 0.2837926 0.5764245
2 0.3039223 0.0000000 0.3621622
```

```
par(mar = c(3.1, 0, 0.1, 0))
plot(m1, show.tip.label = F)
```



Before we proceed, it might be best to label the nodes so we can call the right one by name.

```
phy$node.label <- paste("nd", 1:phy$Nnode, sep = "")
```

Now we can make split QuaSSE objects using **make.quasse.split**. The node we want here is node 153. In this case the speciation and extinction functions are both constant and the functions are the same on both sides of the split. For cases where functions may differ either side of the split, we can pass lists of functions. The **Inf** places the split at the base of the branch protuding from Node 153 (as in the MEDUSA plot above). A value of **0** would place the split at the node itself.

```
f.cc <- make.quasse.split(phy, mass, mass.sd,
                          constant.x, constant.x,
                          "nd153", Inf)
argnames(f.cc)
```

```
[1] "l.c.1"       "m.c.1"       "drift.1"     "diffusion.1" "l.c.2"
[6] "m.c.2"       "drift.2"     "diffusion.2"
```

The first set of parameters refer to the "background" tree and the second set refer to the "foreground" tree rooted at node 153.

Let's constrain drift to be 0 and assume diffusion is the same in both trees.

```
g.cc <- constrain(f.cc, drift.1 ~ 0, drift.2 ~ 0,
                  diffusion.2 ~ diffusion.1)
argnames(g.cc)
```

```
[1] "l.c.1"       "m.c.1"       "diffusion.1" "l.c.2"       "m.c.2"
```

Next we generate a starting point using the starting points from the earlier model (**p.c**).

```
p.cc <- c(p.c, p.c[1:2])
names(p.cc) <- argnames(g.cc)
```

Let's run the ML search.

```
mle.cc <- find.mle(g.cc, p.cc, control=control, lower=0, verbose=0)
```

We need to repeat this process for linear speciation functions and calculate starting points for these as well.

```
f.ll <- make.quasse.split(phy, mass, mass.sd, linear.x, constant.x, "nd153", Inf)
g.ll <- constrain(f.ll, drift.1 ~ 0, drift.2 ~ 0, diffusion.2 ~ diffusion.1)
g.lc <- constrain(g.ll, l.m.2 ~ 0)
g.cl <- constrain(g.ll, l.m.1 ~ 0)
p.cc <- coef(mle.cc)
p.ll <- c(p.cc[1], 0, p.cc[2:4], 0, p.cc[5])
names(p.ll) <- argnames(g.ll)
```

```
mle.ll <- find.mle(g.ll, p.ll, control=control, verbose=0)
```

```
p.lc <- c(coef(mle.ll)[1:3], p.ll[c(4, 5, 7)])
p.cl <- c(p.ll[c(1, 3, 4)], coef(mle.ll)[5:7])
```

```
mle.lc <- find.mle(g.lc, p.lc, control=control, verbose=0)
mle.cl <- find.mle(g.cl, p.cl, control=control, verbose=0)
```

```
knitr::kable(anova(mle.c, linear=mle.l, sigmoidal=mle.s, hump=mle.h,
      part.constant=mle.cc, part.linear.bg=mle.lc,
      part.linear.fg=mle.cl, part.linear=mle.ll))
```

	Df	lnLik	AIC	ChiSq	Pr(>|Chi|)
minimal	3	-841.4029	1688.806	NA	NA
linear	4	-836.0697	1680.139	10.66629	0.0010911
sigmoidal	6	-832.6038	1677.208	17.59805	0.0005323
hump	6	-829.0036	1670.007	24.79861	0.0000170
part.constant	5	-828.5921	1667.184	25.62147	0.0000027
part.linear.bg	6	-828.2577	1668.515	26.29041	0.0000083
part.linear.fg	6	-826.0173	1664.035	30.77116	0.0000009
part.linear	7	-828.3231	1670.646	26.15951	0.0000294

10.4

10.5 Further info

Chapter 11

Phylogenetic Regression

This chapter will show you how to perform phylogenetically correct regression analyses on continuous data in R. As usual, remember to set your working directory to wherever you have saved the necessary files.

11.1 Data

Let's load up some example primate data. You should see the dataframe appear in your environment. If you inspect the object, you will find a number of continuous variables in there for us to investigate.

```
primate.data <- read.table("primates_data.txt", header = T)
names(primate.data)
```

```
[1] "Order"           "Family"          "Binomial"        "AdultBodyMass_g"
[5] "GestationLen_d"  "HomeRange_km2"   "MaxLongevity_m"  "SocialGroupSize"
```

11.2 Linear Regression

Simple linear regression will be familiar to you from LIFE223. The principle is to find out what the relationship is between two or more variables.

To see if body mass and gestation length are related in primate, the best way to go would seem to be a traditional linear regression. The funtion to perform an ordinary least squares linear regression is **lm()**. The first argument is our model, stating in this case that body mass predicts gestation length. Then we specify the data object to tell R where to find the data.

```
m1 <- lm(GestationLen_d ~ log10(AdultBodyMass_g), data = primate.data)
```

You can use the summary function to see the output of your regression.

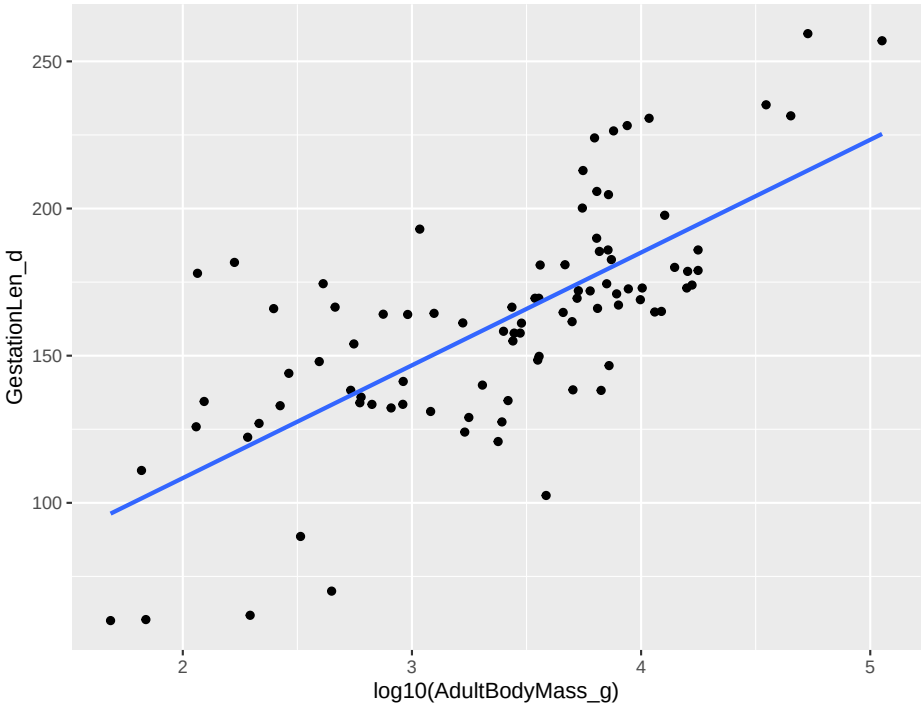
```
summary(m1)
```

```
Call:
lm(formula = GestationLen_d ~ log10(AdultBodyMass_g), data = primate.data)

Residuals:
    Min      1Q  Median      3Q     Max
-66.665 -15.762  -3.987  16.869  67.121

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             31.775     13.927   2.281   0.0249 *
log10(AdultBodyMass_g)  38.319      4.031   9.505 3.37e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 27.31 on 89 degrees of freedom
Multiple R-squared:  0.5038,    Adjusted R-squared:  0.4982
F-statistic: 90.35 on 1 and 89 DF,  p-value: 3.374e-15
```

The key parts of our output are the coefficients table and the three lines of output below which contain the R^2 value. Here, it's telling us that our model is a significant fit to the data as we might expect. Also, the mid-range R^2 (0.50) is what we'd expect given the spread of data in the plot.

We can also plot this line with **ggplot** with the following code. To plot the regression line, add the function geom_smooth amd used the method "lm" to specify that I want a linear model plotted.

```
library(ggplot2)
ggplot(data = primate.data, aes(x = log10(AdultBodyMass_g), y = GestationLen_d)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



11.3 Phylogenetic Signal

As you know, the fact that comparative data points are not statistically independent is a problem for these kind of analyses. Therefore we need to run a phylogenetically corrected analysis.

Phylogenetic regression dates back a while and there have been many different ways to do it [Grafen, 1989, Nunn, 2011]. To understand the logic behind the method, we will first consider the concept of phylogenetic signal.

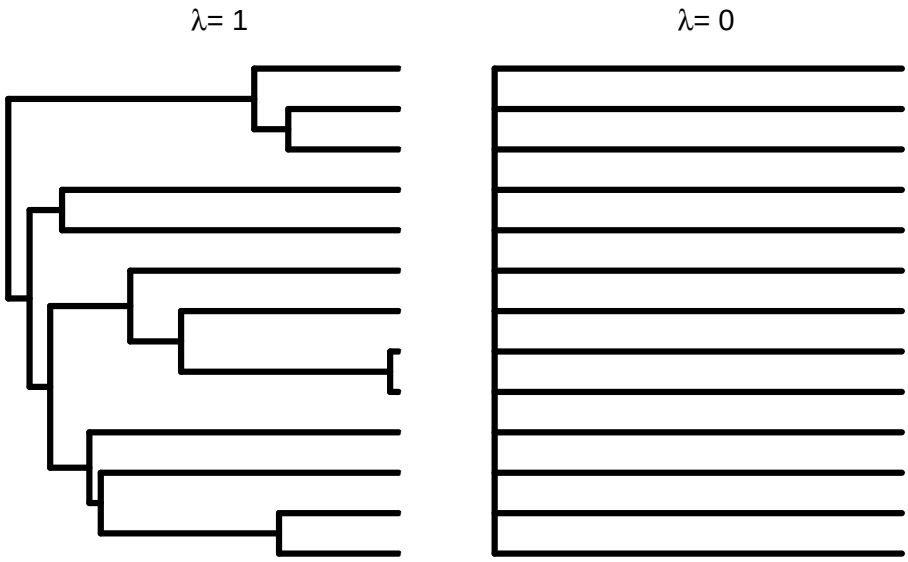
11.3.1 Phylogenetic signal

Phylogenetic signal is defined as *the tendency for closely related species to resemble each other more than distantly related species.*

For example, body mass is (usually) a trait with a strong phylogenetic signal. What this means in primates is that although there is a broad range of body sizes from a few tens of grams up to around 200kg, the distribution of body masses closely follows the pattern of relatedness. Large primates like orangutan, gorillas, chimps and humans are all closely related for example.

The degree of phylogenetic signal in a trait is often described using the scaling parameter λ. λ varies between 0 and 1 and is used to multiply the internal branch lengths so that the tree describes the pattern of variation in the trait.

For example, take the case on the left, where $\lambda = 1$. In this case the tree is untransformed because the variation in the trait follows the structure of the tree. On the right, where $\lambda = 0$, all the internal branch lengths have been multiplied by 0 and therefore collapsed. This "star phylogeny" describes a pattern of variation in which the trait varies at random with respect to the phylogeny. The trait is not equal across the tree but rather the variation in the trait does not correlate to the pattern of relatedness.



11.3.2 caper

Let's run through some examples. There are a few packages that can run phylogenetic regressions in R but the one I usually go with is called caper (Comparative Analysis of Phylogenetics and Evolution in R) [Orme et al., 2018]. So first we'll need to load caper.

```
library(caper)
```

Now we can load up our phylogeny using read.nexus from the ape package.

```
library(ape)
primate.tree <- read.nexus("primate_tree.nex")
```

The regression command in **caper** (along with some other functions) requires the data and tree to be combined in a **comparative data object**. This type

of object is simply a tree and comparative data set concatenated and is created using the function **comparative.data**. We need to specify the tree object, data object, column name in the data where species names are stored and whether we want a variance-covariance matrix included (we do).

```
primates <- comparative.data(phy = primate.tree,      #Our tree
                             data = primate.data,     #Our data
                             names.col = Binomial,    #Data column with the species names
                             vcv = TRUE,              #Variance-covariance matrix
                             na.omit = FALSE,         #We don't want to drop missing data
                             warn.dropped = TRUE)
```

```
Warning in comparative.data(phy = primate.tree, data = primate.data, names.col =
Binomial, : Data dropped in compiling comparative data object
```

This warning message isn't really a problem. If you look at the tree and data I provided, you'll see that the tree has about 200 species but the datafile contains data for only 91. Therefore we expected R to drop some species when compiling the comparative data object. In fact, we asked it warn us if it did so!

We can inspect the structure of the comparative data object using **str** if necessary. You should see that the object contains both the tree and the data. Either one of these (pruned from the larger objects we specified) can be extracted again if needed.

```
str(primates)
```

```
List of 7
 $ phy      :List of 5
  ..$ edge       : int [1:163, 1:2] 84 85 86 87 88 89 90 91 92 93 ...
  ..$ edge.length: num [1:163] 4.95 17.69 19.65 8.12 4.82 ...
  ..$ Nnode      : int 81
  ..$ tip.label  : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_miti
  ..$ node.label : int [1:81] 84 85 86 87 88 89 90 91 92 93 ...
  ..- attr(*, "class")= chr "phylo"
  ..- attr(*, "order")= chr "cladewise"
 $ data     :'data.frame':  83 obs. of  7 variables:
  ..$ Order          : Factor w/ 1 level "Primates": 1 1 1 1 1 1 1 1 1 1 ...
  ..$ Family         : Factor w/ 15 levels "Aotidae","Atelidae",..: 4 4 4 4 4 4 4 5 5 6 ...
  ..$ AdultBodyMass_g: num [1:83] 3540 3445 5041 5325 5257 ...
  ..$ GestationLen_d : num [1:83] 148 170 138 172 170 ...
  ..$ HomeRange_km2  : num [1:83] 0.16 0.24 0.1 0.06 1.15 ...
  ..$ MaxLongevity_m : num [1:83] 340 276 325 316 276 ...
  ..$ SocialGroupSize: num [1:83] 26.3 11 16 4.5 16 28 91.2 1 1 1 ...
 $ data.name: chr "primate.data"
 $ phy.name : chr "primate.tree"
 $ dropped  :List of 2
  ..$ tips          : chr [1:143] "Allenopithecus_nigroviridis" "Cercopithecus_cephus_cephus" "Ce
```

```
  ..$ unmatched.rows: chr [1:8] "Cercopithecus_campbelli" "Cercopithecus_pogonias" "Ch
 $ vcv      : 'VCV.array' num [1:83, 1:83] 72.3 69 64 62.4 64 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_m
  .. ..$ : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_m
 $ vcv.dim  : num 2
 - attr(*, "class")= chr "comparative.data"
```

11.3.3 Estimating Phylogenetic Signal

Let's estimate the phylogenetic signal of gestation length in primates. The key is to remember that we need to call our comparative data object and not the data file we loaded up at the start. We're running the trait on its own (hence the "~ 1") and estimating lambda by maximum likelihood.

```
signal <- pgls(log10(GestationLen_d) ~ 1,
               data = primates,
               lambda = "ML")
summary(signal)
```

```
Call:
pgls(formula = log10(GestationLen_d) ~ 1, data = primates, lambda = "ML")

Residuals:
      Min        1Q    Median        3Q       Max
-0.035946 -0.007060 -0.001217  0.008039  0.049662

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.957
   lower bound : 0.000, p = < 2.22e-16
   upper bound : 1.000, p = 0.050633
   95.0% CI   : (0.879, NA)
delta  [Fix]  : 1.000

Coefficients:
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 2.175175   0.051457  42.272 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01462 on 82 degrees of freedom
Multiple R-squared:     0,  Adjusted R-squared:     0
F-statistic:   NaN on 0 and 82 DF,  p-value: NA
```

This output has a lot in common with a basic regression output. That's because it is one! We used the **pgls** function which performs a regression with phylogenetic correction. Because we included no predictors, the value of λ we estimate here corresponds only to this one trait.

The key part for us is the **Branch length transformations** section of the output. κ and δ are fixed at 1 and so we aren't concerned with those for now. λ is estimated at 0.957. That's a pretty strong phylogenetic signal.

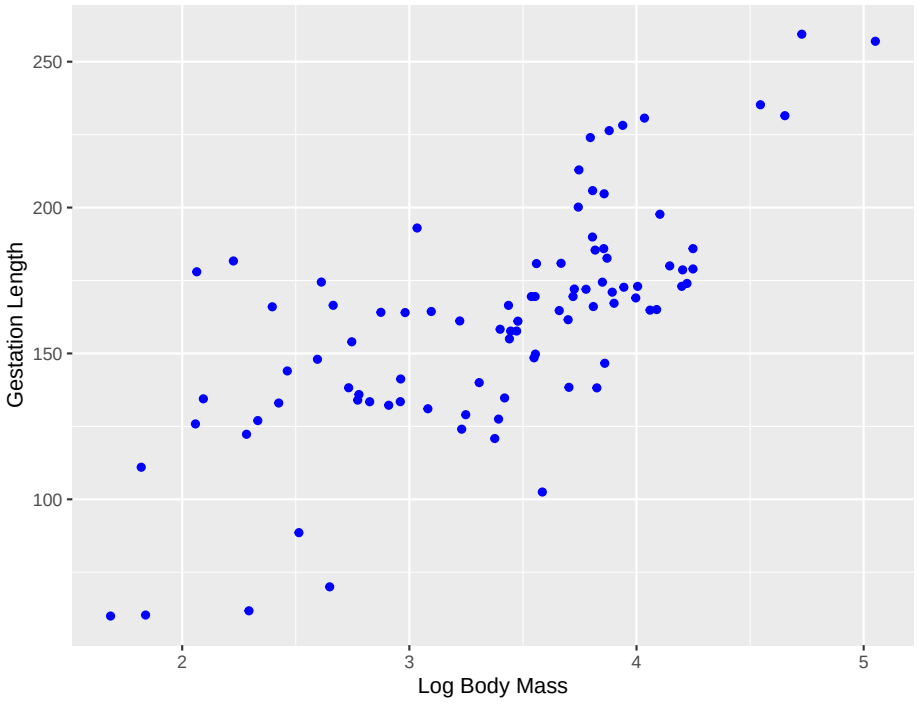
We also have lower bound and upper bound tests. We can see that λ is significantly different from the lower bound of 0 ($p < 2.2 \times 10^{-16}$).

The upper bound test shows us that λ is (narrowly) not significantly different from 1 ($p = 0.051$). This means that we can assume that gestation length has evolved by Brownian motion, in which case λ would equal 1 and the variation in trait would simply reflect the pattern of relatedness amongst species.

11.4 Phylogenetic Regression

Now let's have a go at performing a PGLS regression!

Let's say we have an idea that larger species of primate have longer gestations. Our plot seems to back this up but how strong is this relationship?



We found earlier that there does seem to be a relationship but ordinary least squares linear regression can't be relied upon in this situation. This is because of the statistical non-independence of data points due to shared evolutionary history!

A **phylogenetic generalised least squares regression** (PGLS) uses a covariance matrix to correct the analysis for this statistical non-independence. Put simply, the PGLS assumes the residuals are more similar in more closely related species rather than being randomly distributed as in linear regression.

As you've already seen, the function we need here is **pgls**. The model is constructed exactly as before but this time, we need to construct a full model. We'll be estimating λ by maximum likelihood again.

```
m2 <- pgls(GestationLen_d ~ log10(AdultBodyMass_g), data = primates, lambda = "ML")
summary(m2)
```

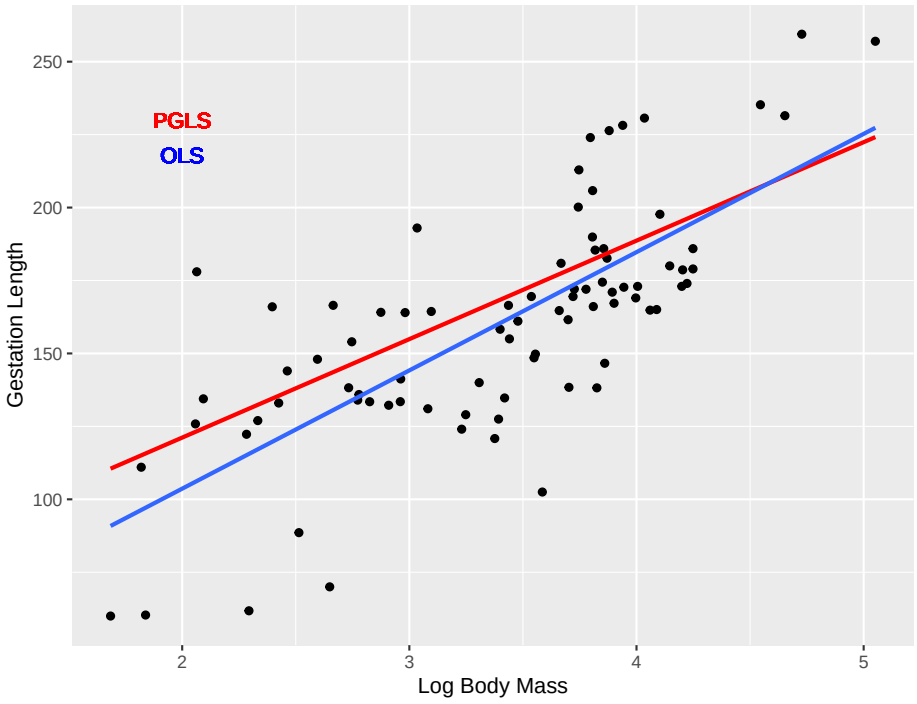
```
Call:
pgls(formula = GestationLen_d ~ log10(AdultBodyMass_g), data = primates,
    lambda = "ML")

Residuals:
     Min       1Q   Median       3Q      Max
-13.0979  -2.4301  -0.8407   1.7269   6.9863

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.805
   lower bound : 0.000, p = 2.6579e-12
   upper bound : 1.000, p = 1.042e-06
   95.0% CI   : (0.607, 0.920)
delta  [Fix]  : 1.000

Coefficients:
                      Estimate Std. Error t value  Pr(>|t|)
(Intercept)            53.6444    20.9100  2.5655   0.01215 *
log10(AdultBodyMass_g) 33.7532     5.8487  5.7710 1.394e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.513 on 81 degrees of freedom
Multiple R-squared: 0.2914, Adjusted R-squared: 0.2826
F-statistic:  33.3 on 1 and 81 DF,  p-value: 1.394e-07
```

As you can see, our model is a significant fit to the data ($F = 33.3$, $R^2 = 0.29$, $p = 1.39 \times 10^{-7}$). More importantly, We've confirmed that body size has a positive effect on gestation length ($\beta = 33.75$, s.e. $= 5.85$, $p = 1.39 \times 10^{-7}$). Time to

plot!

A brief note here. The syntax to get **ggplot** to do this is a little more complex than base graphics (where we can just use abline(m2)!). Here I've plotted both the OLS (blue) and PGLS (red) lines so you can see how they differ.

```
library(dplyr)
primates$data %>%
  mutate(my_model = predict(m2)) %>%
  ggplot() +
  geom_point(aes(log10(AdultBodyMass_g), GestationLen_d)) +
  geom_line(aes(log10(AdultBodyMass_g), my_model),
            colour = "red", lwd = 1) +
  geom_smooth(aes(log10(AdultBodyMass_g), GestationLen_d),
              method = 'lm', se = FALSE) +
  labs(x = "Log Body Mass", y = "Gestation Length") +
  geom_text(x = 2, y = 230, label = "PGLS", colour = "red", size = 4) +
  geom_text(x = 2, y = 218, label = "OLS", colour = "blue", size = 4)
```

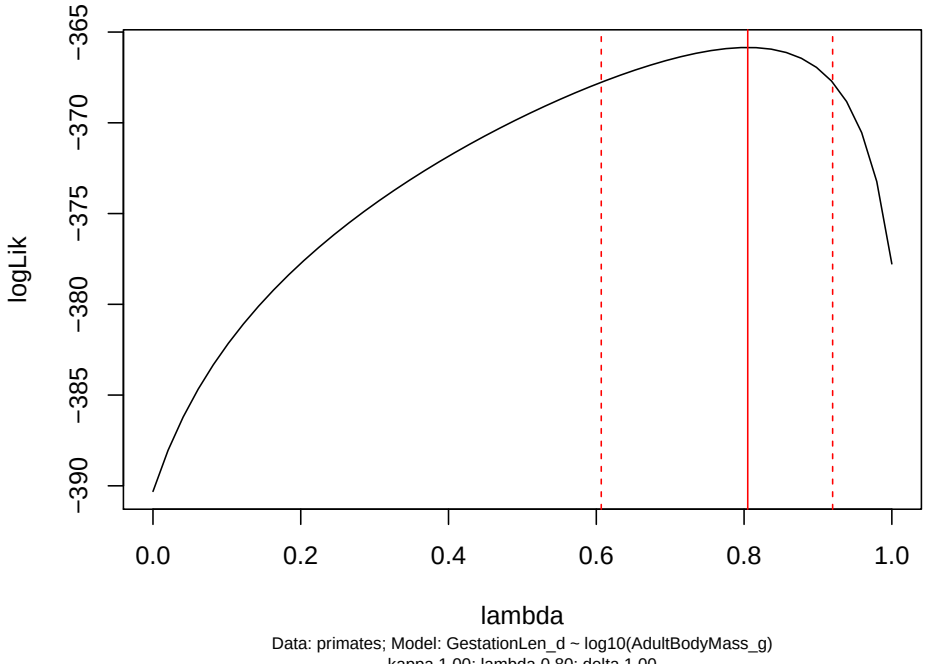


11.4.1 Model Checking

That's the basic model run nicely. Now, we need to run some diagnostic checks. We should start with the likelihood surface of λ since we estimated it by maximum likelihood.

We begin by using the **pgls.profile** function to extract the likelihoods and then simply plot them. What we are looking for is a single peak around our estimated value. If we get a flat surface or multiple peaks, there might be an issue somewhere.

```
lambda.profile <- pgls.profile(m2, which = "lambda")
plot(lambda.profile)
```



This plot describes the log likelihood of λ across its possible range of values (0 - 1). We can clearly see that the likelihood is highest around a single point around 0.8. Check back against the model output earlier to see if this is what we would expect.

Next we need to identify any outliers in the model residuals. The first step here is to extract the residuals from the model, making sure to tell R that we want the phylogenetic residuals. The model output of pgls actually stores both phylogenetic and non-phylogenetic residuals. We can then standardise the residuals by dividing through by the square root of the variance.

```
res <- residuals(m2, phylo = TRUE)
res <- res/sqrt(var(res))[1]
```

The general rule of thumb is that any standardised residual with an absolute value greater than 3 is an outlier and needs to be removed from the analysis. Here, I'm just assigning the species names to the **res** object so we can tell which species are the outliers (if any).

```
rownames(res) <- rownames(m2$residuals)
rownames(res)[abs(res)>3]
```

```
[1] "Microcebus_murinus" "Prolemur_simus"
```

Outliers! Maybe they're causing problems and maybe they aren't. We need to check that by re-running our analysis without them. A simple line of code will take our existing comparative data object and drop out the named outliers.

```
primates.nooutliers <- primates[-which(abs(res)>3),]
```

Now simply re-run the model, remembering to direct R to the new data object.

```
m3 <- pgls(GestationLen_d ~ log10(AdultBodyMass_g),
           data = primates.nooutliers, lambda = "ML")
summary(m3)
```

```
Call:
pgls(formula = GestationLen_d ~ log10(AdultBodyMass_g), data = primates.nooutliers,
    lambda = "ML")

Residuals:
     Min       1Q   Median       3Q      Max
-10.2217  -2.0473  -0.4166   1.3041  11.2824

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.800
   lower bound : 0.000, p = 1.6399e-11
   upper bound : 1.000, p = 1.2064e-06
   95.0% CI   : (0.595, 0.918)
delta  [Fix]  : 1.000

Coefficients:
                      Estimate Std. Error t value  Pr(>|t|)
(Intercept)            54.8023    21.1816  2.5873   0.01151 *
log10(AdultBodyMass_g) 33.3672     5.9418  5.6156 2.815e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 3.53 on 79 degrees of freedom
Multiple R-squared: 0.2853, Adjusted R-squared: 0.2762
F-statistic: 31.54 on 1 and 79 DF,  p-value: 2.815e-07
```

The results have barely changed. So it seems that although those two lemurs were outliers, they weren't effecting the analysis too much. Let's check for outliers in this new model.

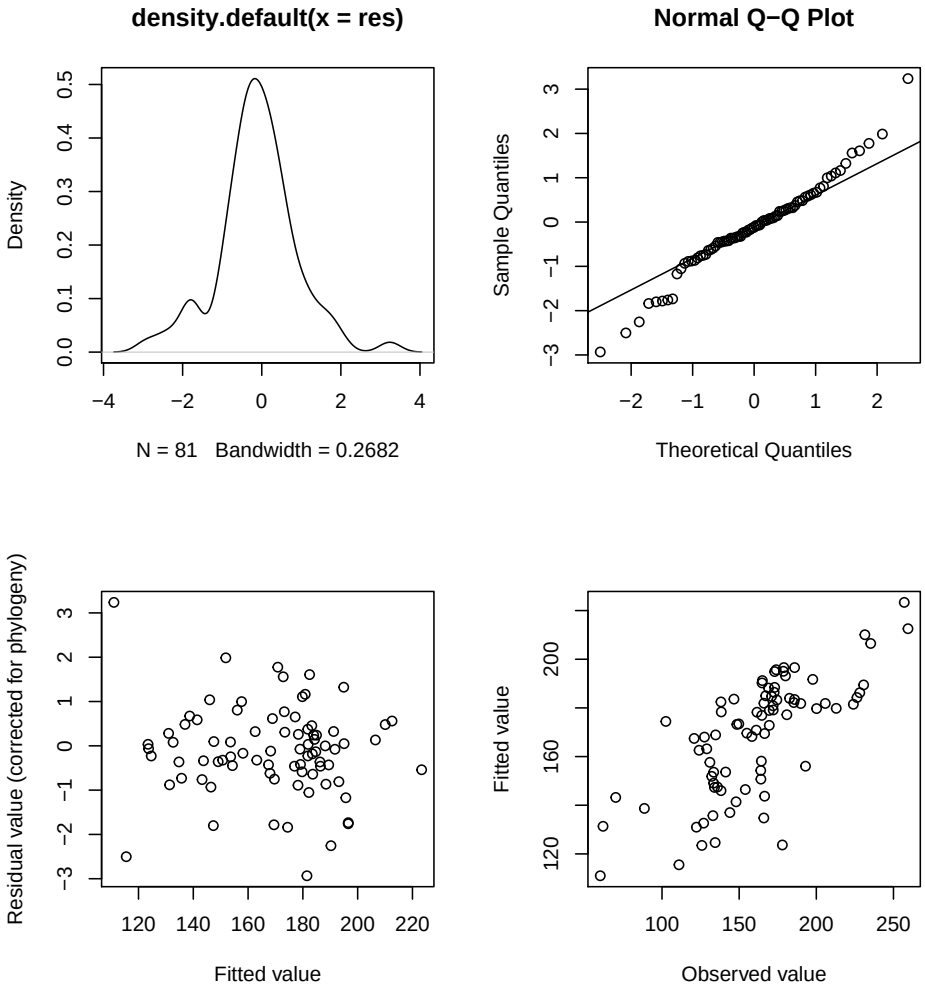
```
res <- residuals(m3, phylo = TRUE)
res <- res/sqrt(var(res))[1]
rownames(res) <- rownames(m3$residuals)
rownames(res)[abs(res)>3]
```

```
[1] "Microcebus_rufus"
```

Another one! Don't worry. This can happen. We need to drop the new outlier again to run the same checks.

Finally, we can check the diagnostic plots of the model. I've included some lines to help arrange the plots. To view the plots for model diagnostics, we can simply plot the model object!

```
par.default <- par(no.readonly = T) #Save default plotting parameters
par(mfrow=c(2,2)) #Set the plot window to show 4 different plots
plot(m3)
```




```
par(par.default) #Reset plot window to default
```

The top left panel shows the distribution of our residuals. We can see a bump near +3. That will be our outlier that needs to be dropped before we proceed any further. The top right plot closely approximates a straight line so that's good. The bottom left shows no real pattern which is also good. The bottom right graph should show a correlation (and it seems to) with the points more or less equally scattered above and below the 45° diagonal. Along that line, the observed and fitted values would be exactly equal.

11.5 Conclusion

So that's how to perform a simple PGLS analysis. This kind of analysis is great for attempting make causal connections between traits of extant species, thus inferring a connection over evolutionary history. For example, we hypothesised that the reason some primates have longer gestation periods is that they have larger body sizes and the PGLS confirmed our suspicion. More complex regressions can include multpile predictors and that's what we'll look at next.

By the way, always make sure to check your models for outliers! In this analysis the gray mouse lemur was an outlier and we had to drop it. Outliers like this can throw off your analysis. If we hadn't checked, we would have presented the analysis in a paper and then had it invalidated when someone checked up on it. Fortunately in this case, the outliers didn't really change the outcome so the gray mouse lemur is off the hook. Look how relieved he is!

Chapter 12

Path Analysis

12.1 PGLS regression

Let's have a closer look at using PGLS regression in statistical analyses. We'll be using the same data and tree from the previous chapter.

If we have a look into the data file provided, we notice a number of continuous variables describing primates. Let's say we're interested in what makes some primates live such long lives. The natural thing to do, is run a PGLS regression with longevity as the outcome.

```
names(primate.data)
```

```
[1] "Order"           "Family"          "Binomial"        "AdultBodyMass_g"
[5] "GestationLen_d"  "HomeRange_km2"   "MaxLongevity_m"  "SocialGroupSize"
```

First, we need a comparative data object combining our data and tree.

```
library(caper)
primates<-comparative.data(phy = primate.tree, data = primate.data,
                           names.col = Binomial, vcv = TRUE,
                           na.omit = FALSE, warn.dropped = FALSE)
```

Now we can assemble and run our model. We want to predict longevity. Many things might influence longevity so how about something like this.

$$Longevity = BodyMass + Gestation + HomeRange + GroupSize$$

Pretty much all of those will need to be log transformed in our model. We can do this ahead of time to make the code easier.

```
primates$data$log.mass <- log10(primates$data$AdultBodyMass_g)
primates$data$log.gestation <- log10(primates$data$GestationLen_d)
primates$data$log.range <- log10(primates$data$HomeRange_km2)
primates$data$log.group <- log10(primates$data$SocialGroupSize)
primates$data$log.lifespan <- log10(primates$data$MaxLongevity_m)
```

Now we can run the model just as we did in the previous chapter.

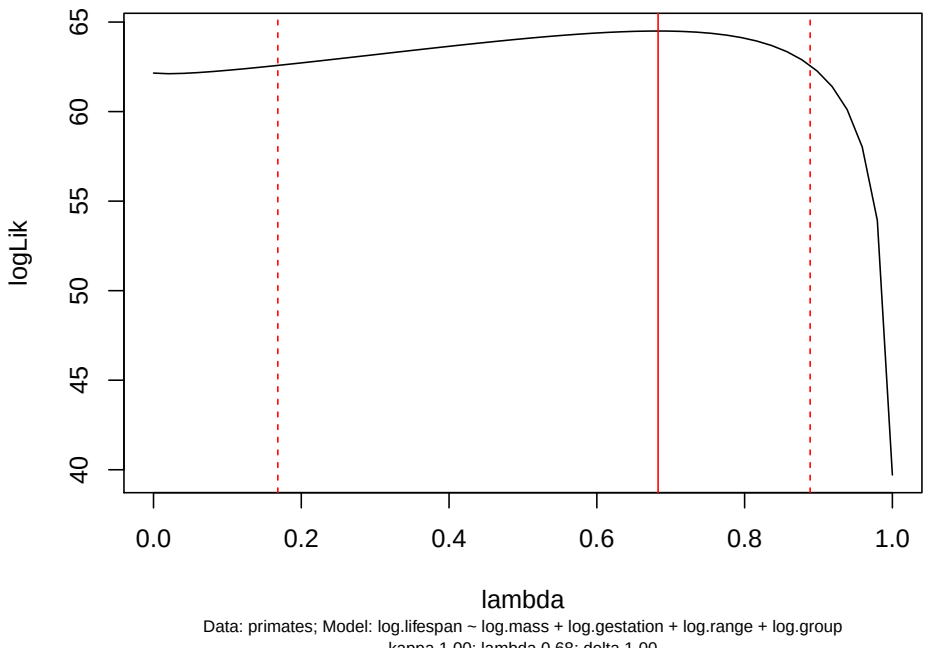
```
m1 <- pgls(log.lifespan ~ log.mass + log.gestation + log.range + log.group,
           data = primates, lambda = "ML")
summary(m1)
```

```
Call:
pgls(formula = log.lifespan ~ log.mass + log.gestation + log.range +
    log.group, data = primates, lambda = "ML")

Residuals:
      Min        1Q    Median        3Q       Max
-0.045758 -0.011997  0.001693  0.008869  0.042009

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.683
   lower bound : 0.000, p = 0.030195
   upper bound : 1.000, p = 1.9055e-12
   95.0% CI   : (0.168, 0.889)
delta  [Fix]  : 1.000

Coefficients:
               Estimate Std. Error t value  Pr(>|t|)
(Intercept)    2.119950   0.404224  5.2445 1.307e-06 ***
log.mass       0.126652   0.045236  2.7998  0.006443 **
log.gestation -0.037853   0.212621 -0.1780  0.859162
log.range      0.027879   0.023855  1.1687  0.246091
log.group      0.028718   0.038070  0.7544  0.452910
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01793 on 78 degrees of freedom
Multiple R-squared: 0.2713, Adjusted R-squared: 0.234
F-statistic: 7.262 on 4 and 78 DF,  p-value: 5.036e-05
```

We have a λ of 0.683. This is **not** the phylogenetic signal of lifespan or any other variable in the model. Rather, it is the phylogenetic signal of the residuals of the model. As you can see, there clearly is a signal here and it's different from 0

($p = 0.03$) and 1 ($p < 0.001$). However, as you can see from the lambda profile below, the CIs are very wide.

```
plot(pgls.profile(m1, which = "lambda"))
```



Not to worry! On we go. The coefficients table in the results output is what we really want to look at here. We can see that the overall fit of the model is significant ($R^2 = 0.23$, $p < 0.001$) but a little underwhelming. The only significant predictor of longevity in our model is body mass ($\beta = 0.13$, se = 0.05, $p = 0.006$). Our other variables do not significantly predict lifespan.

However, we're not done yet! This is where we can employ deletion testing. It is pretty clear that gestation is the most useless component of our model (β = -0.04, se = 0.21, $p = 0.86$). An effect size around 0 with a large standard error and p-value is something we don't need in our lives. So we can re-run the model dropping this variable.

IMPORTANT: ONLY DROP ONE VARIABLE AT A TIME!

```
m2 <- pgls(log.lifespan ~ log.mass + log.range + log.group,
           data = primates, lambda = "ML")
summary(m2)
```

```
Call:
pgls(formula = log.lifespan ~ log.mass + log.range + log.group,
    data = primates, lambda = "ML")
```

```
Residuals:
      Min        1Q    Median        3Q       Max
-0.052816 -0.013804 -0.000923  0.007778  0.037524

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.681
   lower bound : 0.000, p = 0.018281
   upper bound : 1.000, p = 1.9111e-12
   95.0% CI   : (0.178, 0.889)
delta  [Fix]  : 1.000

Coefficients:
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 2.052722   0.142450 14.4101 < 2.2e-16 ***
log.mass    0.122130   0.037140  3.2884  0.001506 **
log.range   0.028703   0.023321  1.2308  0.222061
log.group   0.027939   0.037602  0.7430  0.459670
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0178 on 79 degrees of freedom
Multiple R-squared: 0.2716, Adjusted R-squared: 0.2439
F-statistic: 9.817 on 3 and 79 DF,  p-value: 1.412e-05
```

Our new output presents a similar picture. Body mass is significantly predicting longevity. This time we might want to look at dropping group size ($\beta = 0.03$, se $= 0.04$, p $= 0.46$).

```
m3 <- pgls(log.lifespan ~ log.mass + log.range,
           data = primates, lambda = "ML")
summary(m3)
```

```
Call:
pgls(formula = log.lifespan ~ log.mass + log.range, data = primates,
    lambda = "ML")

Residuals:
      Min        1Q    Median        3Q       Max
-0.038916 -0.012031  0.000161  0.010589  0.040507

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.669
```

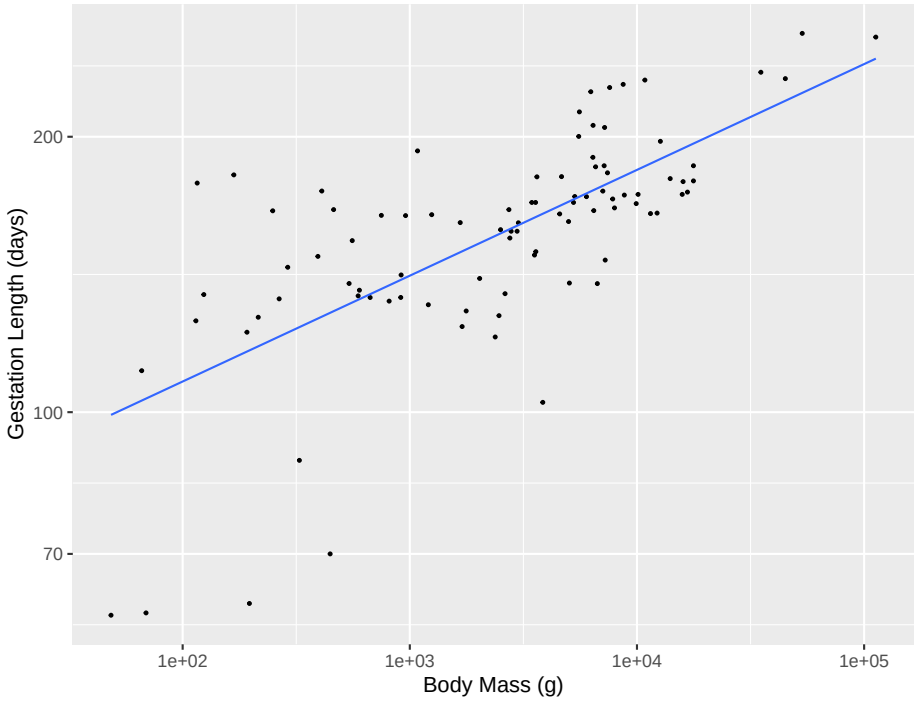
```
   lower bound : 0.000, p = 0.020906
   upper bound : 1.000, p = 2.0745e-12
   95.0% CI    : (0.143, 0.886)
delta  [Fix]  : 1.000

Coefficients:
            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 2.077904   0.137147 15.1510 < 2.2e-16 ***
log.mass    0.120965   0.036786  3.2883  0.001499 **
log.range   0.035873   0.021436  1.6735  0.098141 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01758 on 80 degrees of freedom
Multiple R-squared: 0.271,  Adjusted R-squared: 0.2528
F-statistic: 14.87 on 2 and 80 DF,  p-value: 3.224e-06
```

Now we're getting somewhere. We have reduced our maximal model to something a little more descriptive. We can keep going with this until we have eliminated all non-significant variables. However, at each stage, we should make sure that dropping variables doesn't reduce our model fit! At each stage so far, the model fit has got better (see the bottom line of each output). What would happen if you also dropped range?

Also keep in mind that the model we've built here doesn't use any interaction terms. These can be included just as in any other model and treated in the same way!

12.2 Variance Inflation

In analyses like the one above, it is not uncommon to run into problems. When constructing a regression model, it is important to make sure your predictor variables don't correlate.



As we can see here, this is clearly not the case. Exactly how much of a problem this can be is uncertain. Strong correlations are hugely problematic but weak ones are probably not that big a deal.

Imagine if we had two strongly correlated predictor variables in a model. How would we determine which one was causing the variation in our outcome? The more strongly correlated these predictors are, the more difficult it becomes to use them both to predict an outcome. This is called **variance inflation** (or multicollinearity).

The function **vif** in the package **car** [Fox and Weisberg, 2019] calculates the variance inflation factor of a model. Here we can see that every variable in our model has a VIF greater than 1, indicating we do have some multicollinearity. However, there is no critical value beyond which variance inflation becomes a problem. As a general rule of thumb, a value that exceeds 5 is considered enough of a problem that you shouldn't trust your results.

```
m1.1 <- lm(primates$data$log.lifespan ~ primates$data$log.mass +
             primates$data$log.gestation + primates$data$log.range +
             primates$data$log.group)
library(car)
vif(m1.1)
```

```
     primates$data$log.mass primates$data$log.gestation
                   3.724550                     2.109221
```

```
primates$data$log.range     primates$data$log.group
               3.433493                    2.045929
```

So it may be that we don't have a problem here and we're probably ok to proceed. Note however that I had to calculate VIF for the uncorrected model (without phylogenetic correction) so the true values for the pgls may differ slightly.

12.3 Path Analysis

In comparative analyses, multicollinearity is to be expected. Many traits of a given animal are influenced by other traits. To solve this problem, we have path analysis [von Hardenberg and Gonzalez-Voyer, 2013, Gonzalez-Voyer and Von Hardenberg, 2014].

In path analysis, we take a number of variables and define a **causal model**. This model consists of **vertices** (our variables) and **arrows** which define the relationships between the variables. This is an example of such a model.

Figure 12.1: Predicted pathway model of primate brain evolution taken from Dunbar and Shultz (2007).

Models like this one can be complex with many arrows showing a complex network of relationships. This is fairly typical of biological systems! If you study it closely, you'll be able to see that it describes the interactions between variables. For example, *body size* directly influences *lifespan*, *home range*, *activity*, *BMR* and *diet*. The relationships described by these arrows represent a causal hypothesis.

Confirmatory path analysis was developed to test causal hypotheses that can be represented by **Directed Acyclic Graphs** (DAGs) [Shipley, 2000]. DAG is the term given to the graphical representation of the variables and the hypothesised relationships between them. We can test if a DAG is a reasonable hypothesis using a principle called **distance-sepearation** or d-sep for short [Shipley, 2000].

D-sep tests **conditional independencies** in your DAG. If two vertices are not connected by an arrow, then the model implies that these variables are not causally linked.

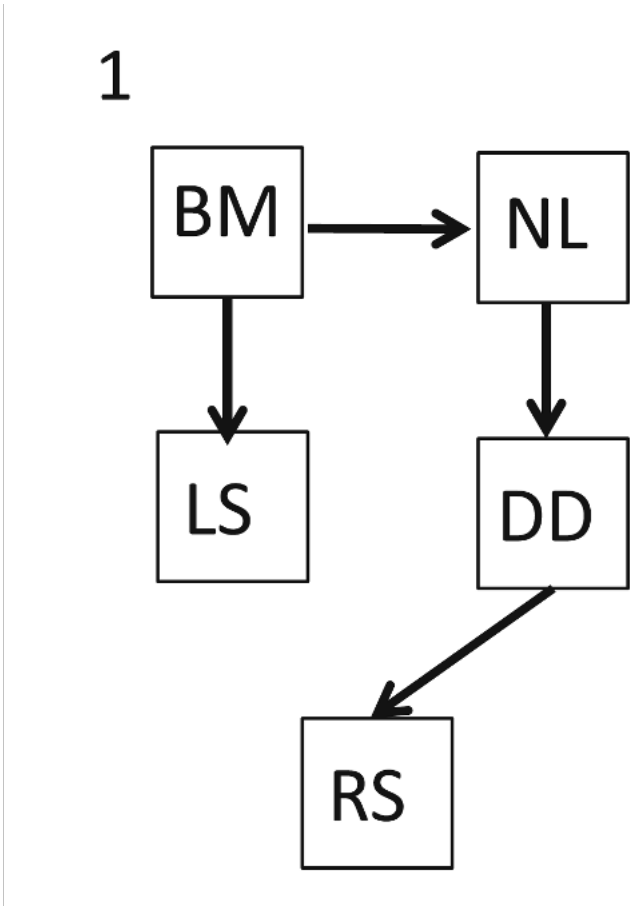


Figure 12.2: Example of a simple directed acyclic graph from Gonzalez-Voyer and Von Hardenberg 2014.

This example DAG contains a number of independencies. **BM** and **DD** are unconnected and therefore should be independent if this model is reasonable. We can actually build a model to test that indepedency as follows:

$$DD = BM + NL$$

Note that **NL** must be included as a predictor of **DD** as it is in the model. If **BM** and **DD** are truly independent, then this model would return no statistically significant relationship between them.

This is not the only independency to test for. **LS** and **RS**, **DD** and **LS**, **NL** and **LS**, and **BM** and **RS** are all also independent under the DAG and so need

to be tested in the same way. The probabilities of all these tests are combined using Fisher's C test [Shipley, 2016].

To calculate C, we add the natural logs of all the p-values describing the supposed independencies using the formula below where k is the number of indepedence tests.

$$C = -2\sum_{i=1}^{k} ln(p_i)$$

Using this approach for numerous models, we can begin to compare competeing hypotheses and select the most appropriate model. To do so, we can calculate the C-statistic Information Criterion with a modification for small sample sizes (CICc) with the following formula where n is the sample size, q is the number of parameters and C is the C-statistic.

$$CICc = C + 2q \times \frac{n}{(n-q-1)}$$

CICc is an information theory based approach. Essentially, the smaller the value of CICc the better the fit of the model to the data. There's no hard limit for distinguishing between two models but the general rule of thumb is that a difference in CICc of more than 2 mean that one model is a better fit than the other. Within 2 and we can't reliably distinguish between the models.

So far, I've taken you through confirmatory path analysis but I haven't mentioned phylogentic control at all. Path analysis without phylogenetic control is performed using generalised linear models (glms)

Let's see how this works with an example.

12.3.1 Example

For this demonstration I'm loading in some new data [von Hardenberg and Gonzalez-Voyer, 2013, Gonzalez-Voyer and Von Hardenberg, 2014]. The data contains information on five continuous variables for 100 species from a group of mammals called the *Rhinogradentia* [Stumpke, 1957].

```
rhino.dat <- read.csv("rhino.csv")
rownames(rhino.dat) <- rhino.dat$SP
rhino.tree <- read.tree("rhino.tree")
```

Path analysis is something I used to do manually, model by model. However, as is usual in comparative studies, someone has written a package called **phylopath** that does it all quite simply [van der Bijl, 2018]. We'll also need **ggforce** to help with plotting later on [Pedersen, 2019].

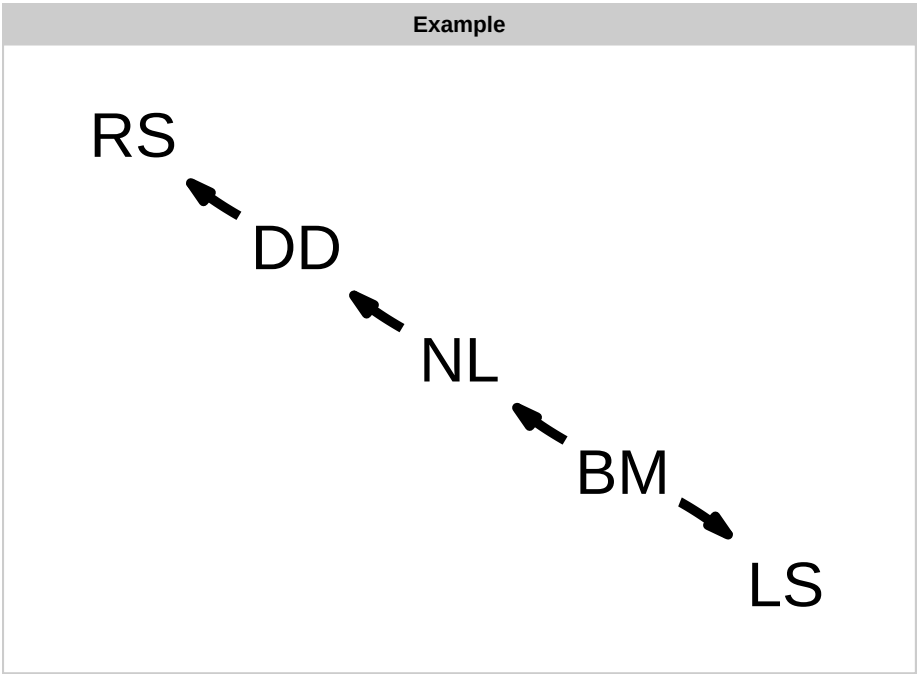
```
install.packages("phylopath")
install.packages("ggforce")
require(phylopath)
require(ggforce)
```

The first step is to define the models using the function **define_model_set**. Take your time over this! Typos will be very important. In this instance, let's reproduce the example DAG from above. The formulae are passed in the same style as any formula in R.

```
m <- define_model_set(
  Example = c(RS ~ DD, LS ~ BM, NL ~ BM, DD ~ NL)
)
```

We can plot the model using the function **plot_model_set**. The extra arguments here clean up the plot a little. **box_x** and **box_y** create a box around the vertices to force the arrows not to overlap with the text. **text_size** and **edge_width** set the font size and arrow size respectively.

```
plot_model_set(m, text_size = 10,
               box_x = 18, box_y = 16,
               edge_width = 2)
```

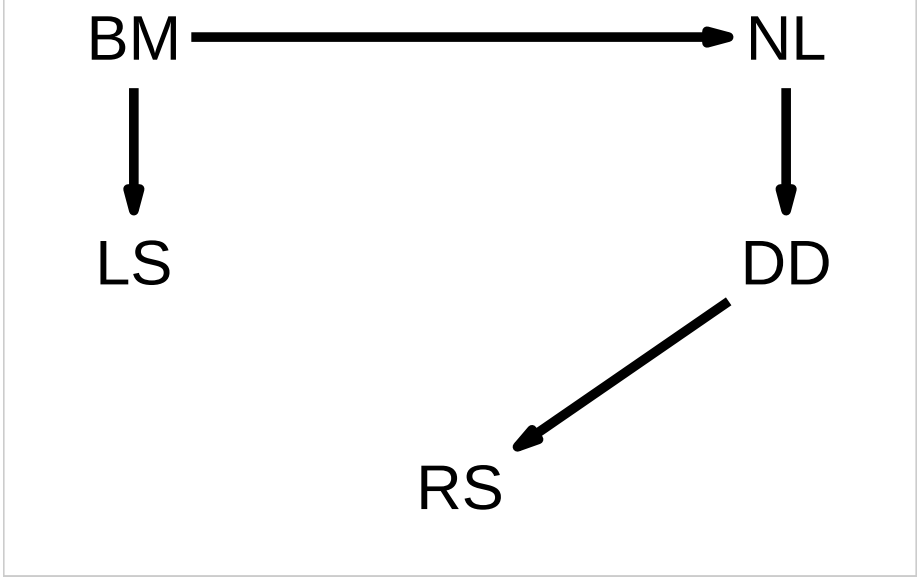


The layout isn't exactly the same as the previous diagram but we can manipulate

that with the argument **manual_layout** if we wish. We need to create a dataframe with x and y coordinates for the final plots. We can also set the arrows to be straight by setting **curvature** to be 0.

```
pos <- data.frame(names = c("RS", "DD", "NL", "BM", "LS"),
                  x = c(1,1.5,1.5,.5,.5),
                  y = c(0,.5,1,1,.5))
plot_model_set(m, text_size = 10, box_x = 18, box_y = 16,
               edge_width = 2, manual_layout = pos, curvature = 0)
```

Example



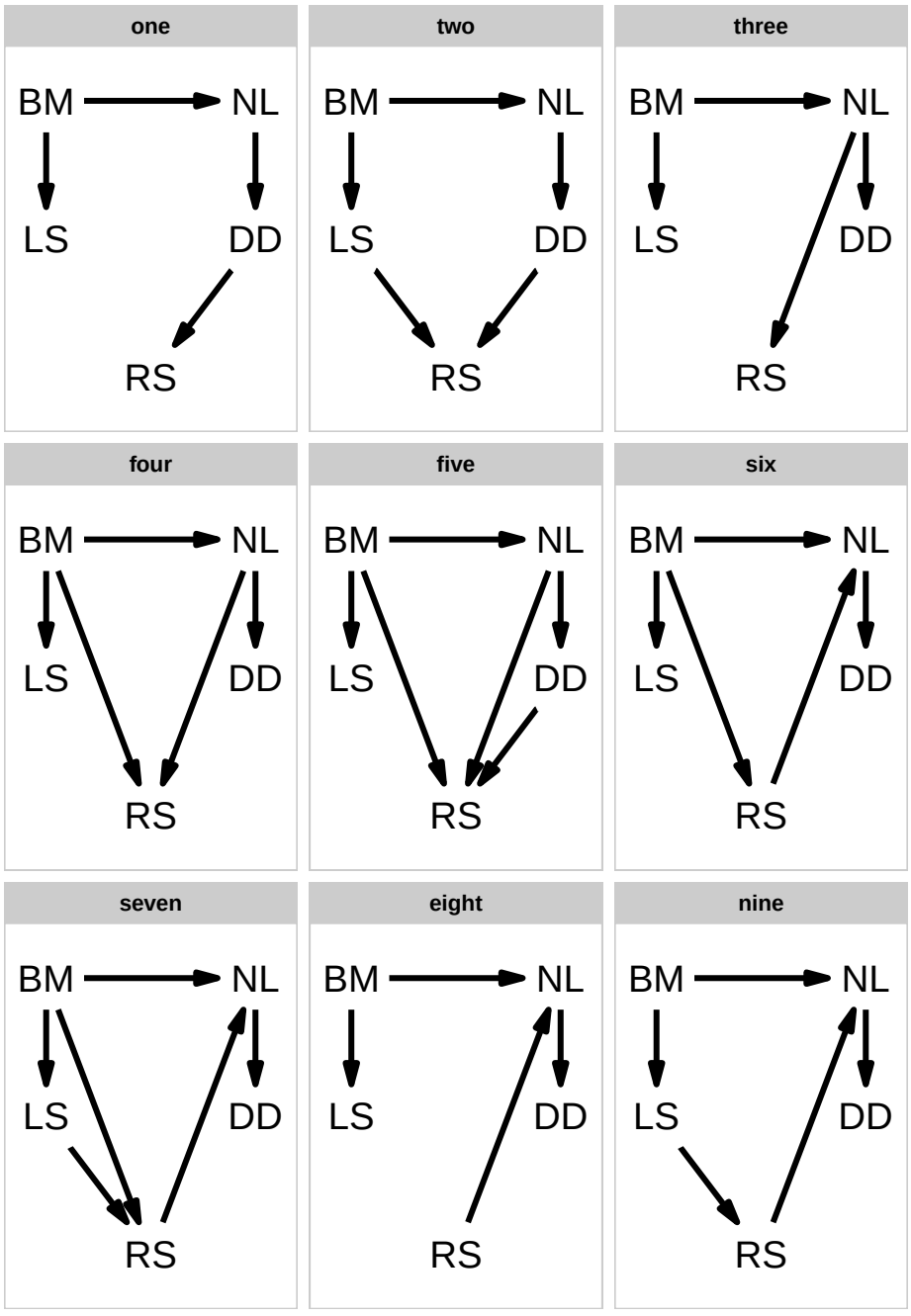
For the full analysis, let's define a number of possible models and try and see which best describes the data. We start with **define_model_set** and pass lists of formulae to each model which here I have named numerically 1 to 9. We've also passed some formulae to the **.common** option as these formulae are present in every model. This saves a fair bit of repetition.

```
m <- define_model_set(
  one   = c(RS ~ DD),
  two   = c(DD ~ NL, RS ~ LS + DD),
  three = c(RS ~ NL),
  four  = c(RS ~ BM + NL),
  five  = c(RS ~ BM + NL + DD),
  six   = c(NL ~ RS, RS ~ BM),
  seven = c(NL ~ RS, RS ~ LS + BM),
```

```
  eight = c(NL ~ RS),
  nine  = c(NL ~ RS, RS ~ LS),
  .common = c(LS ~ BM, NL ~ BM, DD ~ NL)
)
```

Because this is a complex step, ALWAYS check your **Directed Acyclic Graphs** (DAGs) describing your models look as you expect them to.

```
plot_model_set(m, text_size = 6, box_x = 12, box_y = 10,
               edge_width = 1.25, manual_layout = pos, curvature = 0)
```



If we are doing this by hand, now we need to identify d-separation statements and test conditional independencies. **Phylopath** does it all for us. To interpret the final results, you really need to understand what happens in this step so take

it slow and revisit the explanations above. You can also refer to the methods papers which explain things very well [von Hardenberg and Gonzalez-Voyer, 2013, Gonzalez-Voyer and Von Hardenberg, 2014].

```
results <- phylo_path(m, data = rhino.dat, tree = rhino.tree, model = "lambda")
```

If we call up the results, we get some basic information. Here our results contain the models and the PGLS analyses used to evaluate each model.

```
results
```

```
A phylogenetic path analysis, on the variables:
    Continuous:  BM NL DD RS LS
    Binary:

 Evaluated for these models: one two three four five six seven eight nine

 Containing 46 phylogenetic regressions, of which 22 unique
```

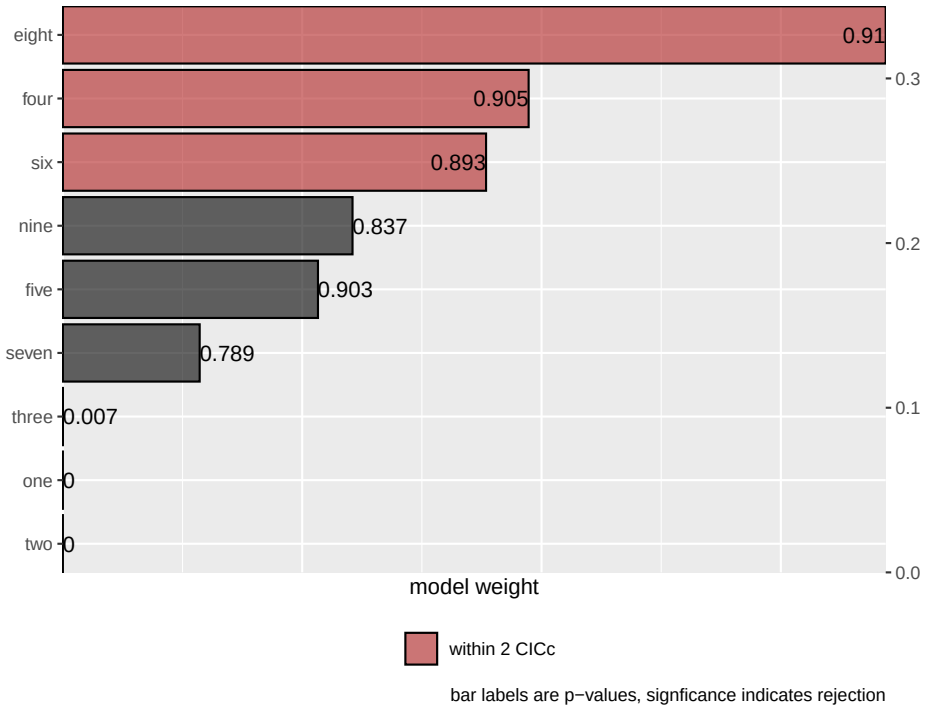
The summary of the results object gives us the table for model comparisons. For each model we have all the key stats. We can see that the top six models are not rejected based on d-separation ($p > 0.05$). Pay particular attention to the CICc (C-statistic Information Criterion) and ΔCICc. You'll notice that the table is helpfully sorted for us by ΔCICc.

```
summary(results)
```

```
      model k  q     C        p CICc delta_CICc        l        w
eight eight 6  9  6.11 9.10e-01 26.1       0.00 1.00e+00 3.44e-01
four   four 5 10  4.78 9.05e-01 27.3       1.14 5.66e-01 1.95e-01
six     six 5 10  4.97 8.93e-01 27.4       1.33 5.14e-01 1.77e-01
nine   nine 5 10  5.73 8.37e-01 28.2       2.09 3.52e-01 1.21e-01
five   five 4 11  3.46 9.03e-01 28.5       2.34 3.10e-01 1.07e-01
seven seven 4 11  4.70 7.89e-01 29.7       3.59 1.66e-01 5.72e-02
three three 6  9 27.17 7.30e-03 47.2      21.06 2.68e-05 9.20e-06
one     one 6  9 62.01 9.70e-09 82.0      55.89 7.29e-13 2.51e-13
two     two 5 10 60.97 2.38e-09 83.4      57.33 3.56e-13 1.23e-13
```

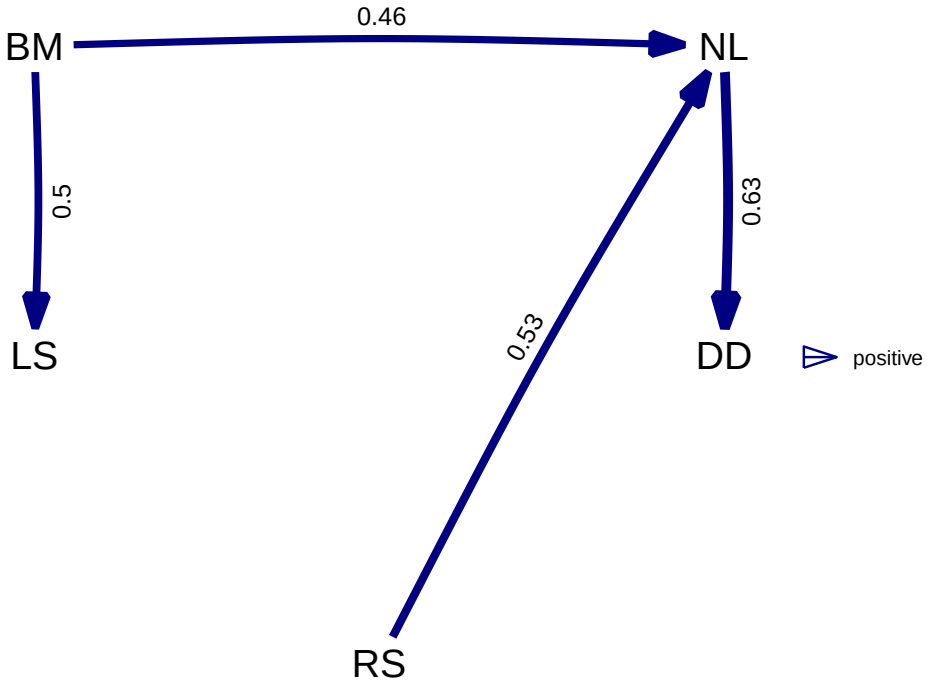
We can plot the summary of results and we are presented with a very useful plot, showing us the degree of support for each model. In this case, the top 3 (models eight, six and four) are within 2 CICc of each other and therefore have roughly equal statistical support.

```
plot(summary(results))
```



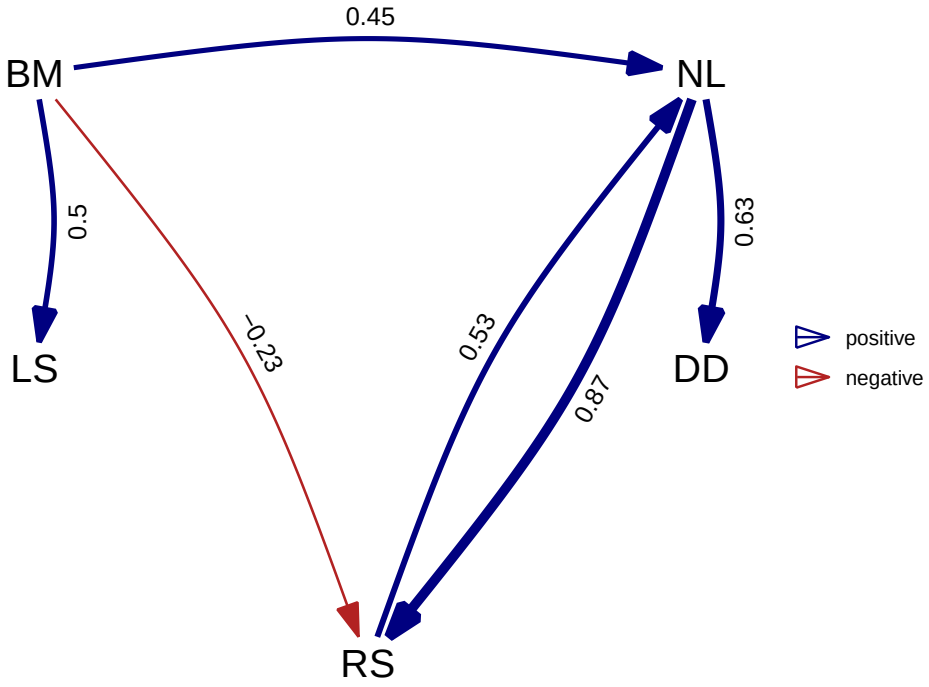
For now, let's just plot the best model.

```
best_model <- best(results)
plot(best_model, manual_layout = pos)
```



The numbers and width of each arrow represent the path coefficents (effect sizes). The blue colour represents a positive relationship. A negative relationship would be red. Remember here that we have a few models with roughly equivalent support. We can average these models if this happens. The **cut_off** argument tells us where to place the limit of support. By default this is 2 CICc and so the function **average** will only average the models that fall within 2 CICc of the best supported model.

```
av.model <- average(results, cut_off = 2)
plot(av.model, algorithm = "mds", curvature = 0.1, manual_layout = pos)
```



12.4 Conclusion

So that's a path analysis. I've skirted over some details here because this particular method is complex and the package used here strips out a lot of the manual effort needed to do a path analysis. This is a good thing. It cuts down the computing time and means you don't have to write as much code. However, it does mean you can run the analysis without really understanding it. This could be a problem when you need to write it up or present it so be sure you know what's happening.

For getting to the end, here's a slightly confused looking monkey.

Chapter 13

Studying Convergence

13.1 Indentifying convergent evolution

The first challenge in the study of convergent evolution is finding a way to systematically identify when it has occurred. For some traits this is easy. For example we can say with some confidence that three groups of mammals (cetaceans, pinnipeds and sirenians) have convergently evolved an aquatic lifestyle.

If we want some support for an assertion of convergent evolution, we can look to ancestral state reconstruction. As shown in chapters 8 and 10, ancestral state reconstruction allows us to look into the evolutionary history trait and potentially make a determination about if a trait arose independently or not.

As we saw in chapter 8, there has been and independent loss of conspicuous sexual swelling in macaques. Namely, *Macaca arctoides* has evolved this trait independently of the other 4 species (*M. sinica, M. radiata, M. thibetana* and *M. assamensis*) that exhibit it.

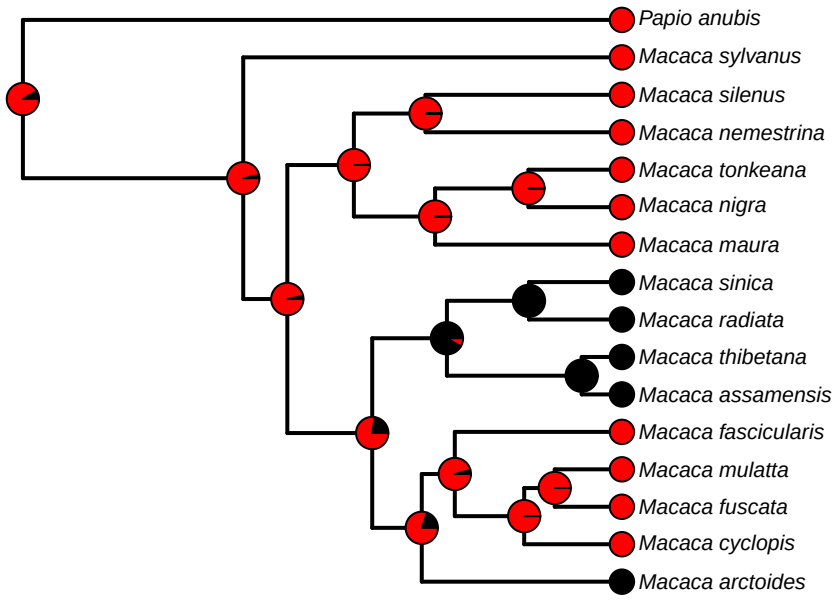


Figure 13.1: Maximum likelihood reconstruction of the evolution of conspicuous sexual swellings in macaques using an equal rates model of evolution.

As you can probably tell, it can be easy to assign convergent evolution to categorical traits and less so to continuous traits. To identify convergence in continuous characters, we need to identify selective regimes.

13.1.1 SURFACE

We can identify selective regimes by reconstructing the trait on the tree and fitting models to the reconstruction. One method to do exactly this is called SURFACE [Ingram and Mahler, 2013].

SURFACE stands for *SURFACE Uses Regime Fitting with Aikaike information Criterion to model Convergent Evolution*. The method has two phases, known as **forward** and **backward**.

The **forward** phase begins at the root of the tree and fits models to the ancestral state reconstruction of the trait or traits in question. At the root, the trait is only experiencing one selective regime (because the root represents one lineage and therefore one trait value). As we proceed forward in time we reach branching points (nodes). At each node we add a new regime, giving us two models. The first model has no change and the new model as a regime shift at the node. This new model is then tested against the no-change model and if the new regime improves the fit of the model to the data, then we accept the addition of the regime.

To summarise, the **forward** phase begins at the root and identifies rgeime shifts in the evolution of the trait going forward through the tree.

The **backward** phase is very similar but starts at the tips of the tree. In this phase we go backwards in time and each time we reach a regime shift, we collapse the regime and combine it with others across the tree. Each time regimes are combined, this produces a new model to be compared against the older model. If the combination of two regimes improves the fit of the model, the regimes are combined and we move on. This means that this phase of the analysis will collapse similar regimes together across the tree.

The end product of SURFACE is a list of parameters describing the number of distinct regimes *k'*, the number of shifts towards convergent regimes *c* and other useful parameters like the number of convergent regimes reached by multiple shifts and the proportion of shifts towards convergent regimes (see [Ingram and Mahler, 2013] for details).

13.1.2 A Super Example

Let's see how SURFACE works. First we need to install the **surface** package [Ingram and Mahler, 2013]. Unfortunately this is no longer on the CRAN archive so we need to look elsewhere. Fortunately, MRAN (Microsoft R Application Network) have a handy tool that takes snapshots of the CRAN R packages. This means we can go to MRAN and get a version of **surface** from before it was scrubbed from CRAN. What heroes!

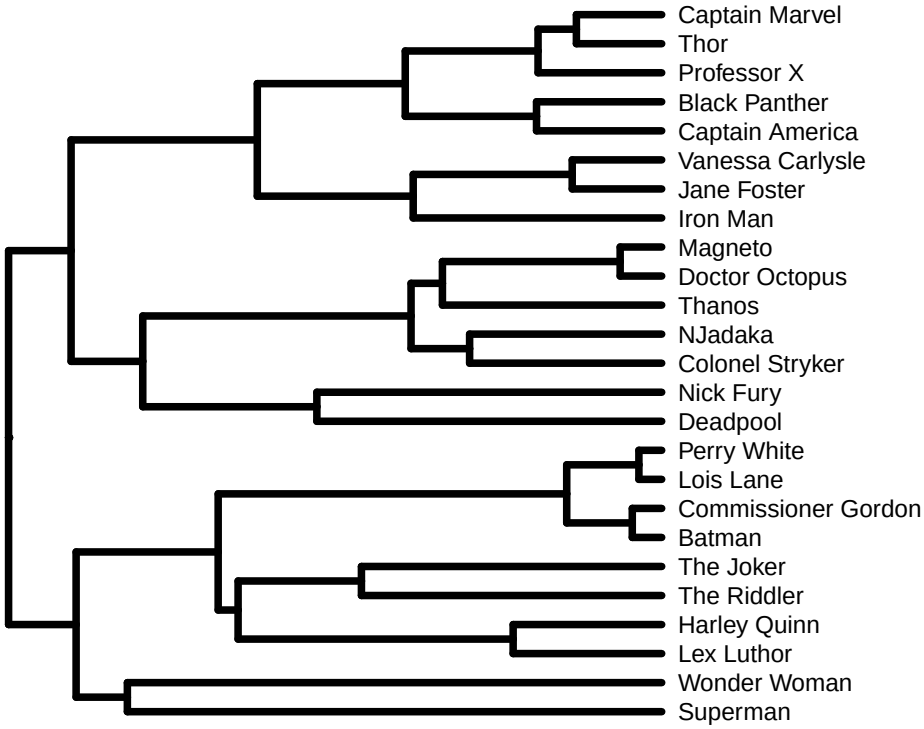
```
install.packages("surface", repos="https://mran.microsoft.com/snapshot/2015-06-30")
library(surface)
```

Next we can load some example data.

```
tree <- read.nexus("Superheroes.nex")
data <- read.csv("Superhero data.csv")
rownames(data) <- data$X
geiger::name.check(tree, data)
```

```
[1] "OK"
```

The data we have here describe three personality traits of 25 characters from the Marvel and DC cinematic universes. There is also a phylogeny describing the pattern of relatedness between characters. For this analysis, we are going to use all 3 personality traits to try and identify any convergence across universes.



13.1.2.1 Forward Phase

Before we run the analysis, we need to do a little preparation. These functions convert the tree and data into formats that can be analysed by functions called by the **surface** analysis. **nameNodes** adds unique node labels to the tree to aid the conversion.

```
tree<-nameNodes(tree)
olist<-convertTreeData(tree,data[,c(2,3,4)])
otree<-olist[[1]]
odata<-olist[[2]]
```

The function **surfaceForward** runs the forward phase of the analysis as described above. The more complex your tree and data, the longer this will take.

```
fwd<-surfaceForward(otree, odata, aic_threshold = 0, exclude = 0,
                    verbose = FALSE, plotaic = FALSE)
```

From this analysis, we can extract the value of k (the number of distinct regimes).

```
k<-length(fwd)
k
```

```
[1] 6
```

For some more details, we can look into the summary of the analysis using **surfaceSummary**. There are a number of useful components we could investigate further.

```
fsum<-surfaceSummary(fwd)
names(fsum)
```

```
 [1] "n_steps"       "lnls"          "n_regimes_seq" "aics"
 [5] "shifts"        "n_regimes"     "alpha"         "phylhalflife"
 [9] "sigma_squared" "theta"
```

13.1.2.2 Backward Phase

Next we can collapse the regimes by running the backwards analysis with **surfaceBackward**.

```
bwd<-surfaceBackward(otree, odata, starting_model = fwd[[k]],
                     aic_threshold = 0, only_best = TRUE,
                     verbose = FALSE, plotaic = FALSE)
```

We can use **surfaceSummary** again to extract the key parameters.

```
bsum<-surfaceSummary(bwd)
kk<-length(bwd)
kk
```

```
[1] 4
```

```
bsum$n_regimes
```

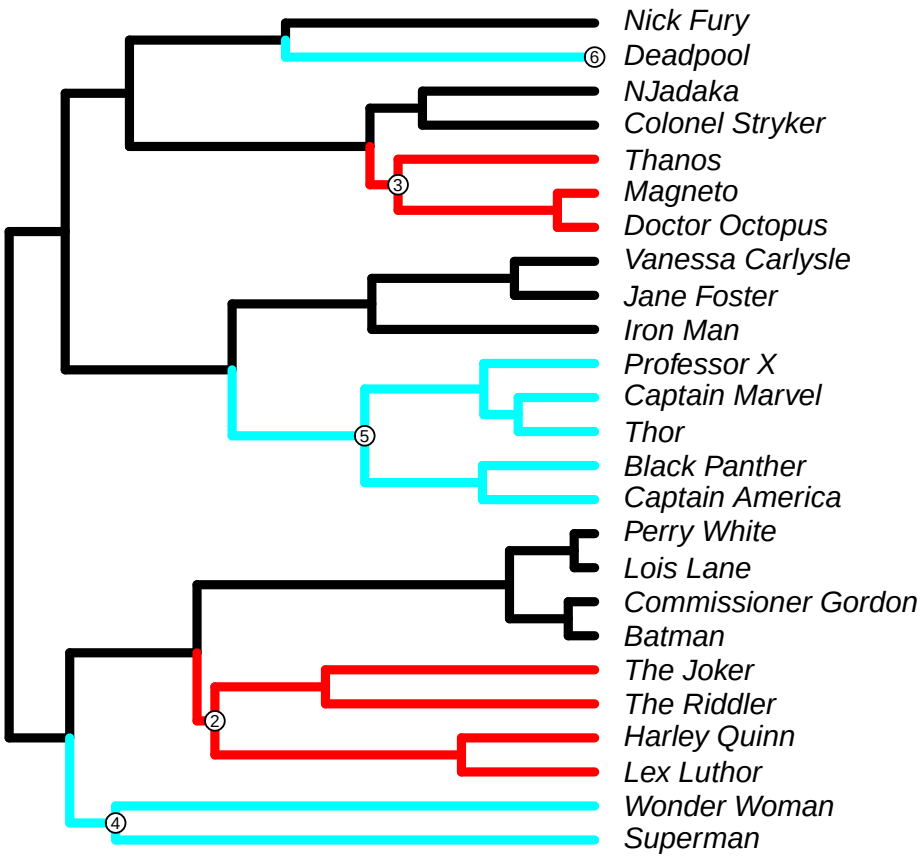
```
             k         kprime         deltak              c    kprime_conv
             6              3              3              5              2
kprime_nonconv
             1
```

So we have 6 regime shifts from **surfaceForward** (k). There are 3 distinct regimes ($kprime$), 2 of which are convergent ($kprime\_conv$) and 1 is non-convergent ($kprime\_nonconv$).

13.1.2.3 Visualisation

The best thing to do with an analysis like this is visualise the results. We can paint the regimes onto the tree using **surfaceTreePlot**.

```
surfaceTreePlot(tree, bwd[[kk]], labelshifts = T, label.offset = .5,
                no.margin = T, edge.width = 5, cex = 1)
```

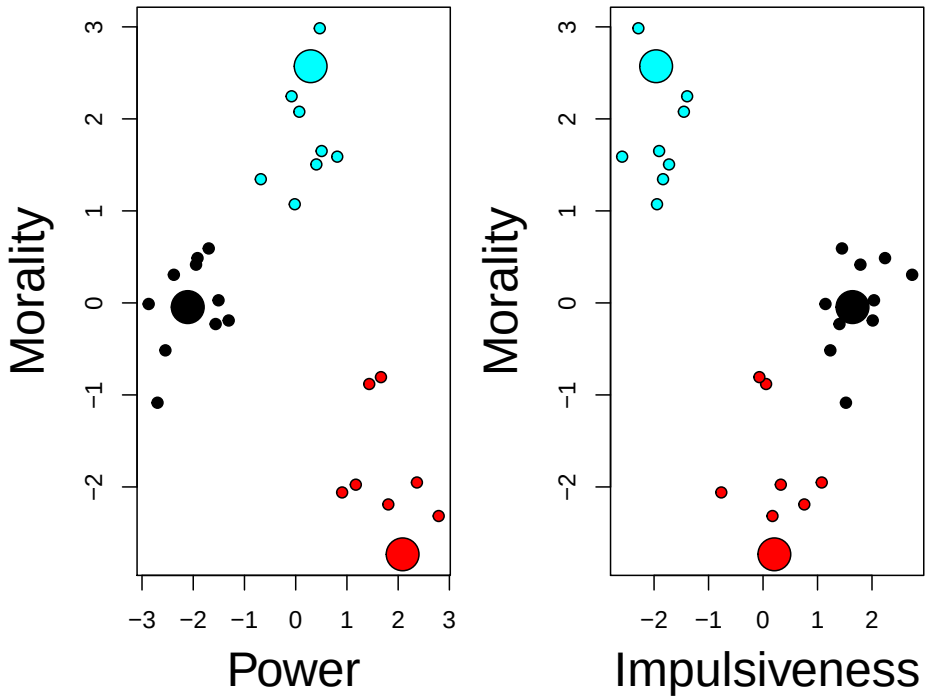


Using the tree like this allows us to spot some patterns. The blue regime appears to be mostly heroes such as Wonder Woman, Captain Marvel and Black Panther. The selective regime that produces heroes occurs 3 times here. Once leading to Wonder Woman and Superman in the DC universe, once leading to a cluster of Avengers plus Professor X in the Marvel universe and a third time leading to Deadpool.

The red regime appears to produce villains such as The Joker and Harley Quinn in DC and Thanos and Magneto in Marvel. Finally the black regime is mostly made up of human beings whose moral leanings may go one way or the other (eg. Lois lane or Colonel Stryker) but are nevertheless grouped separately from heroes and villains.

There are some surprising placements. Batman and Iron Man are both in the same regime (black) which is unsurprising given that they are similar but they are oddly not grouped alongside heroes. This is most likely because of the traits we have used. Let's plot them out as scatterplots using **surfaceTraitPlot**.

```
par(mfrow=c(1,2), mai=c(0.9,0.9,0.2,0.2))
surfaceTraitPlot(data[,c(2,3,4)], bwd[[kk]], whattraits = c(1,2),
                 cex.opt = 3, cex = 1, cex.lab = 2)
surfaceTraitPlot(data[,c(2,3,4)], bwd[[kk]], whattraits = c(3,2),
                 cex.opt = 3, cex = 1,cex.lab = 2)
```



These plots show how the traits cluster along with the average value for each regime (larger point).

13.2 Quantifying convergent evolution

SURFACE and other methods identify convergent evolution but sometimes we might be more interested in finding out how strong the convergence is. Taking our superhero example a little further, we might expect that there would be particularly strong convergence between similar characters. We could lump together characters like Iron Man and Batman who are essentially humans with access to lots of technology. Alternatively, we could look at superpowered aliens such as Thanos, Superman and Thor or perhaps characters with military backgrounds such as Deadpool, Colonel Stryker, Captain Marvel and Captain America.

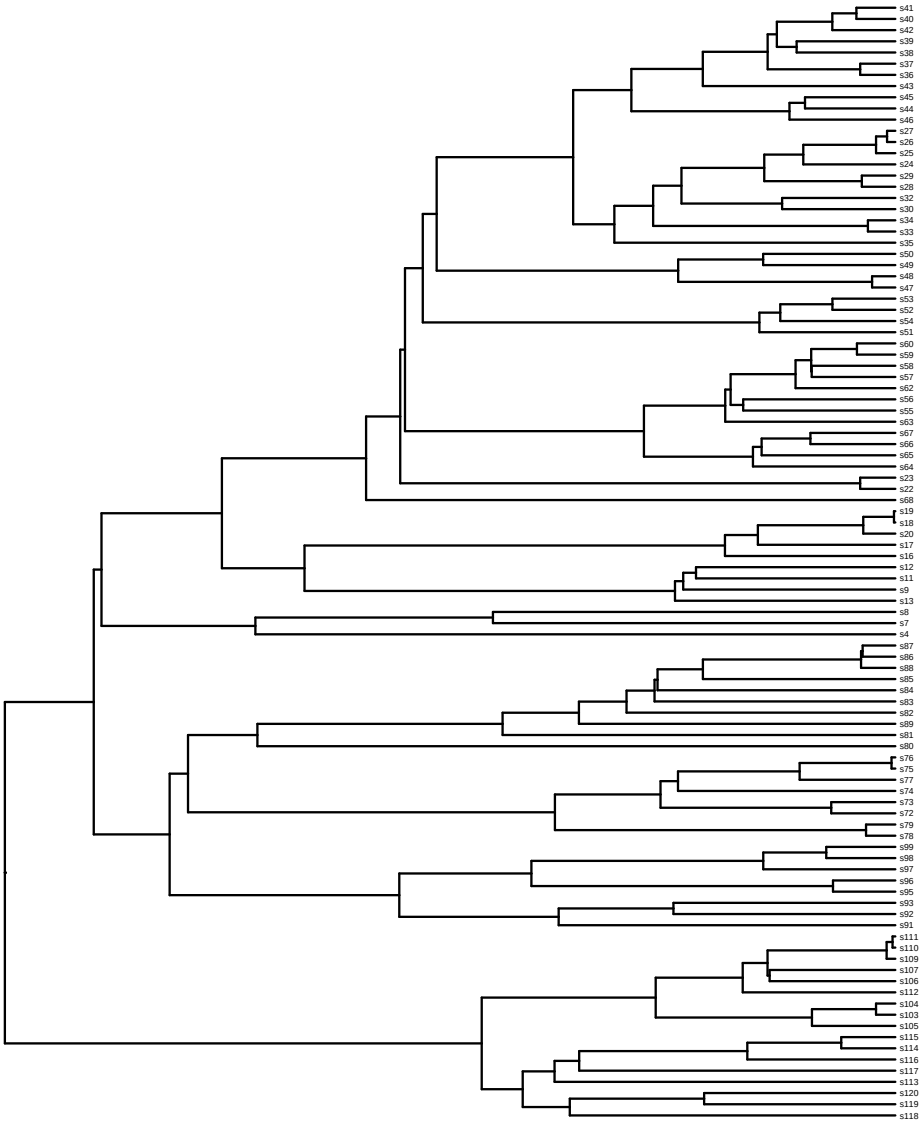
Groupings such as these with a convergent characteristic are called **focal groupings**. We can quantify the strength of convergence by measuring how similar the focal group members are to each other relative to the group as a whole. The way to do this is to calculate the **Wheatsheaf Index** [Arbuckle et al., 2014].

13.2.1 The Wheatsheaf Index

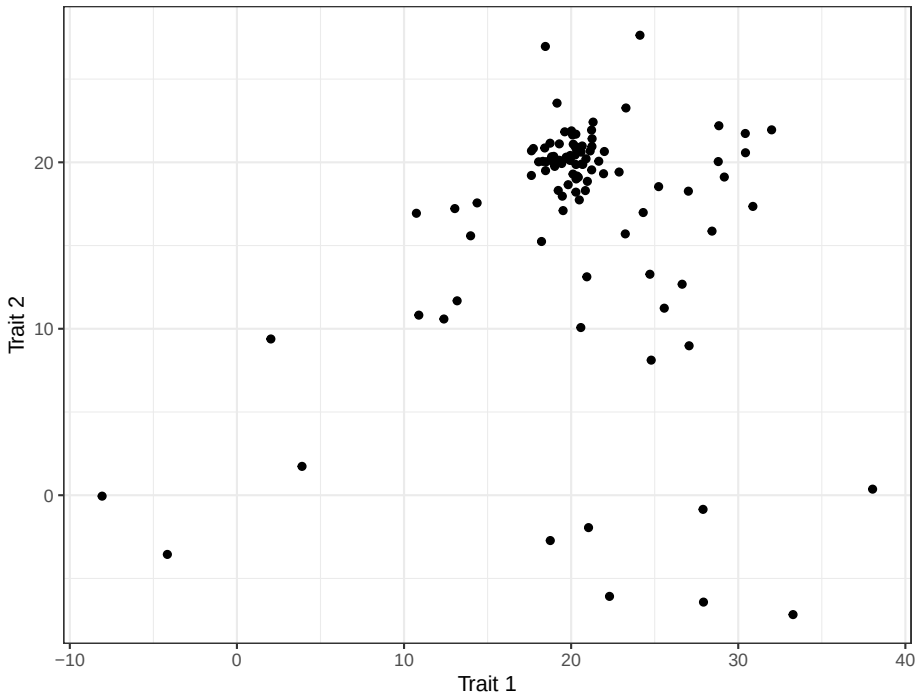
The Wheatsheaf Index (W) requires us to calculate the Euclidean distance between species in phenotypic space. First we calculate the mean distance between all the species in the sample and then the mean distance between the focal group species. Finally, we calculate W by dividing the total distance by the focal group distance. If we get a value greater than 1, then we can say that the focal group are more similar to each other than to the group as a whole and thus, quantify the strength of convergence.

The package to perform a Wheatsheaf analysis is called **windex** [Arbuckle and Minter, 2014]. The package contains some example data for us to use.

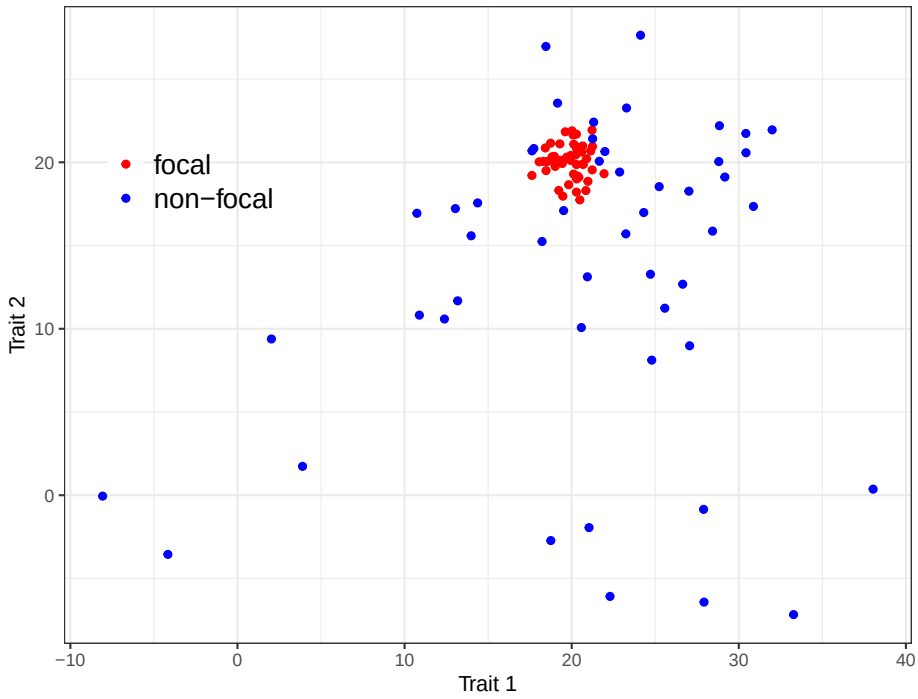
```
library(windex)
data("sample.data")
data("sample.tree")
```



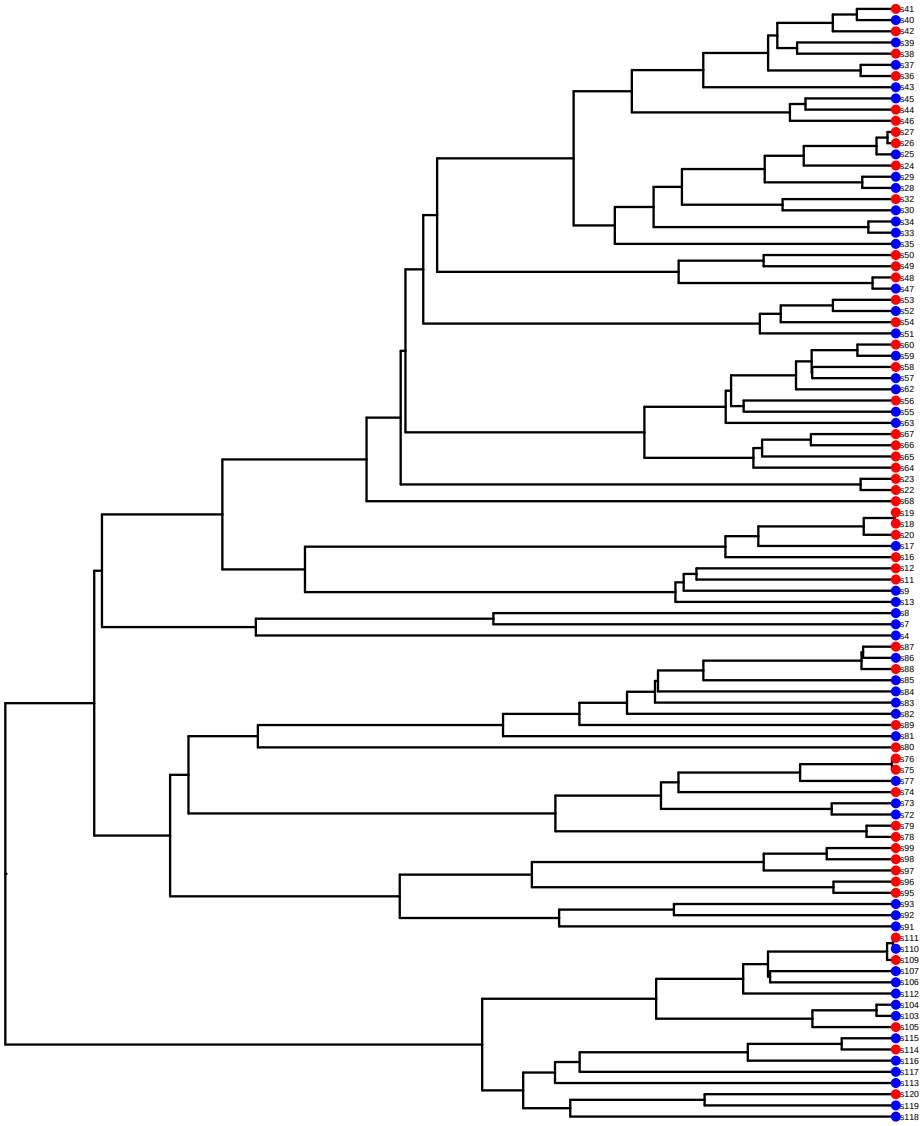
This scatterplot represents the **phenotypic space** described by the two traits. We can see there is a clear clustering of species but the question we have to ask is, does this pattern mean convergence and if yes, how strong is that convergence?



To make things clearer, let's highlight our focal group species in phenotypic space by plotting them in red.



It's clear that the focal group species cluster very closely. The tree shows us the taxonomic distribution of the focal group. so it seems that this clustering cannot be explained by phylogeny.



To calculate W, we can use the **windex** function. Remember that a value of W greater than 1 indicates the focal group is more similar to each other than to the group as a whole.

```
windex(sample.data, sample.tree,
       traits = c("ou1", "ou2"),
       focal = sample.data[,2],
       SE = FALSE)
```

```
$`Wheatsheaf Index`
[1] 5.289446
```

```
$`Lower 95% CI`
[1] 5.121161

$`Upper 95% CI`
[1] 5.413457
```

Here the value of W of 5.29 indicates very strong convergence as we would have predicted by using the traitspace.

The question we have to answer now is whether the value of W is higher than expected given the shape of the tree. This can be done using the function **test.windex** which performs a number of bootstrap replicates of the analysis, resampling the tree and data as many times as we ask (1000 is standard but it takes a while!).

```
t <- test.windex(sample.data, sample.tree,
       traits = c("ou1", "ou2"),
       focal = sample.data[,2], reps = 100,
       SE = FALSE, plot = TRUE)
```

The plot here shows the distribution of the bootstrap replicates. We can see from the plotted line that the calculated value and the accompanying confidence intervals are far separate from the distribution of the replicates. This indicates a hih degree of confidence in some unusually storng convergence here. As we can see, the p-value backs this up.

```
t
```

```
$`P-value=`
[1] 0
```

13.3 Further info

Chapter 14

Bibliography

Bibliography

Michael E. Alfaro, Francesco Santini, Chad Brock, Hugo Alamillo, Alex Dornburg, Daniel L. Rabosky, Giorgio Carnevale, and Luke J. Harmon. Nine exceptional radiations plus high turnover explain species diversity in jawed vertebrates. *Proceedings of the National Academy of Sciences*, 106(32): 13410–13414, 2009. ISSN 0027-8424. doi: 10.1073/pnas.0811087106. URL https://www.pnas.org/content/106/32/13410.

Kevin Arbuckle and Amanda Minter. *windex: windex: Analysing convergent evolution using the Wheatsheaf index*, 2014. URL https://CRAN.R-project.org/package=windex. R package version 1.0.

Kevin Arbuckle, Cheryl M Bennett, and Michael P Speed. A simple measure of the strength of convergent evolution. *Methods in Ecology and Evolution*, 5 (7):685–693, 2014.

Christian Arnold, Luke J. Matthews, and Charles L. Nunn. The 10ktrees website: A new online resource for primate phylogeny. *Evolutionary Anthropology: Issues, News, and Reviews*, 19(3):114–118, 2010. doi: 10.1002/evan.20251. URL https://doi.org/10.1002/evan.20251.

D.A. Baum and S.D. Smith. *Tree Thinking: An Introduction to Phylogenetic Biology*. Macmillan Learning, 2012. ISBN 9781936221165. URL https://books.google.co.uk/books?id=zW_ApwAACAAJ.

Silvia Castiglione, Gianmarco Tesone, Martina Piccolo, Marina Melchionna, Alessandro Mondanaro, Carmela Serio, Mirko Di Febbraro, and Pasquale Raia. A new method for testing evolutionary rate variation and shifts in phenotypic evolution. *Methods in Ecology and Evolution*, 9:974–983, 2018. URL https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.12954.

Silvia Castiglione, Carmela Serio, Alessandro Mondanaro, Marina Melchionna, Francesco Carotenuto, Mirko Di Febbraro, Antonio Profico, Davide Tamagnini, and Pasquale Raia. Ancestral state estimation with phylogenetic ridge regression. *Evolutionary Biology*, 2020. doi: 10.1007/s11692-020-09505-x. URL https://doi.org/10.1007/s11692-020-09505-x.

R. G. Fitzjohn. Quantitative traits and diversification. *Systematic*

Biology, 59(6):619–633, 2010. doi: 10.1093/sysbio/syq053. URL https://liverpool.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-78650666284&site=eds-live&scope=site.

R. G. FitzJohn. Diversitree: Comparative phylogenetic analyses of diversification in r. *Methods in Ecology and Evolution*, in press, 2012. doi: 10.1111/j.2041-210X.2012.00234.x.

John Fox and Sanford Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition, 2019. URL https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Jr. Garland, Theodore, Allan W. Dickerman, Christine M. Janis, and Jason A. Jones. Phylogenetic Analysis of Covariance by Computer Simulation. *Systematic Biology*, 42(3):265–292, 1993. ISSN 1063-5157. doi: 10.1093/sysbio/42.3.265. URL https://doi.org/10.1093/sysbio/42.3.265.

Olivier Gascuel and Mike Steel. Predicting the ancestral character changes in a tree is typically easier than predicting the root state. *Systematic Biology*, 63 (3):421–435, 2014. doi: 10.1093/sysbio/syu010. URL https://doi.org/10.1093/sysbio/syu010.

Alejandro Gonzalez-Voyer and Achaz Von Hardenberg. An introduction to phylogenetic path analysis. In *Modern phylogenetic comparative methods and their application in evolutionary biology*, pages 201–229. Springer, 2014.

Alan Grafen. The phylogenetic regression. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 326(1233):119–157, 1989. doi: 10.1098/rstb.1989.0106. URL https://doi.org/10.1098/rstb.1989.0106.

LJ Harmon, JT Weir, CD Brock, RE Glor, and W Challenger. Geiger: investigating evolutionary radiations. *Bioinformatics*, 24:129–131, 2008.

David W. E. Hone and Michael J. Benton. The evolution of large size: how does cope's rule work? *Trends in Ecology & Evolution*, 20(1):4–6, 2005. URL https://liverpool.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselp&AN=S0169534704003143&site=eds-live&scope=site.

Nizar Ibrahim, Simone Maganuco, Cristiano Dal Sasso, Matteo Fabbri, Marco Auditore, Gabriele Bindellini, David M. Martill, Samir Zouhri, Diego A. Mattarelli, David M. Unwin, Jasmina Wiemann, Davide Bonadonna, Ayoub Amane, Juliana Jakubczak, Ulrich Joger, George V. Lauder, and Stephanie E. Pierce. Tail-propelled aquatic locomotion in a theropod dinosaur. *Nature*, 581(7806):67–70, 2020. doi: 10.1038/s41586-020-2190-3. URL https://doi.org/10.1038/s41586-020-2190-3.

Travis Ingram and D. Luke Mahler. Surface: detecting convergent evolution from comparative data by fitting ornstein-uhlenbeck models with stepwise akaike information criterion. *Methods in Ecology and Evolution*, 4(5):416–425,

2013. doi: 10.1111/2041-210X.12034. URL https://doi.org/10.1111/2041-210X.12034.

Jeffrey B. Joy, Richard H. Liang, Rosemary M. McCloskey, T. Nguyen, and Art F. Y. Poon. Ancestral reconstruction. *PLOS Computational Biology*, 12(7): e1004763–, 2016. URL https://doi.org/10.1371/journal.pcbi.1004763.

Joel G. Kingsolver and David W. Pfennig. Individual-level selection as a cause of cope's rule of phyletic size increase. *Evolution*, 58(7):1608, 2004. URL https://liverpool.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.3449385&site=eds-live&scope=site.

Stilianos Louca and Matthew W. Pennell. Extant timetrees are consistent with a myriad of diversification histories. *Nature*, 580(7804):502–505, 2020. doi: 10.1038/s41586-020-2176-1. URL https://doi.org/10.1038/s41586-020-2176-1.

Michael R. McGowen, Stephen H. Montgomery, Clay Clark, and John Gatesy. Phylogeny and adaptive evolution of the brain-development gene microcephalin (mcph1) in cetaceans. *BMC Evolutionary Biology*, 11(1):98, 2011. doi: 10.1186/1471-2148-11-98. URL https://doi.org/10.1186/1471-2148-11-98.

Stephen H. Montgomery, Isabella Capellini, Robert A. Barton, and Nicholas I. Mundy. Reconstructing the ups and downs of primate brain evolution: implications for adaptive hypotheses and homo floresiensis. *BMC Biology*, 8(1):9, 2010. doi: 10.1186/1741-7007-8-9. URL https://doi.org/10.1186/1741-7007-8-9.

Stephen H. Montgomery, Jonathan H. Geisler, Michael R. McGowen, Charlotte Fox, Lori Marino, and John Gatesy. The evolutionary history of cetacean brain and body size. 67(11):3339–3353, 2013. URL www.jstor.org/stable/24032748.

Martin Morgan. *BiocManager: Access the Bioconductor Project Package Repository*, 2019. URL https://CRAN.R-project.org/package=BiocManager. R package version 1.30.9.

H Morlon, E Lewitus, FL Condamine, M Manceau, J Clavel, and J Drury. Rpanda: an r package for macroevolutionary analyses on phylogenetic trees. *Methods in Ecology and Evolution*, 7:589–597, 2016. URL https://CRAN.R-project.org/package=RPANDA. R package version 1.4.

R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–741, 2003. doi: 10.1214/aos/1056562461. URL https://liverpool.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-1642370803&site=eds-live&scope=site.

C.L. Nunn. *The Comparative Approach in Evolutionary Anthropology and*

Biology. University of Chicago Press, 2011. ISBN 9780226608983. URL https://books.google.co.uk/books?id=qj4cSzJGQJAC.

Katrin Nyakatura and Olaf RP Bininda-Emonds. Updating the evolutionary history of carnivora (mammalia): a new species-level supertree complete with divergence time estimates. *BMC Biology*, 10(1):12, 2012. doi: 10.1186/1741-7007-10-12. URL https://doi.org/10.1186/1741-7007-10-12.

David Orme, Rob Freckleton, Gavin Thomas, Thomas Petzoldt, Susanne Fritz, Nick Isaac, and Will Pearse. *caper: Comparative Analyses of Phylogenetics and Evolution in R*, 2018. URL https://CRAN.R-project.org/package=caper. R package version 1.0.1.

E. Paradis and K. Schliep. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics*, 35:526–528, 2018.

Heidi G. Parker, Dayna L. Dreger, Maud Rimbault, Brian W. Davis, Alexandra B. Mullen, Gretchen Carpintero-Ramirez, and Elaine A. Ostrander. Genomic analyses reveal the influence of geographic origin, migration, and hybridization on modern dog breed development. *Cell Reports*, 19(4):697 – 708, 2017. ISSN 2211-1247. doi: https://doi.org/10.1016/j.celrep.2017.03.079. URL http://www.sciencedirect.com/science/article/pii/S2211124717304564.

Thomas Lin Pedersen. *ggforce: Accelerating 'ggplot2'*, 2019. URL https://CRAN.R-project.org/package=ggforce. R package version 0.3.1.

Ralph S. Peters, Lars Krogmann, Christoph Mayer, Alexander Donath, Simon Gunkel, Karen Meusemann, Alexey Kozlov, Lars Podsiadlowski, Malte Petersen, Robert Lanfear, Patricia A. Diez, John Heraty, Karl M. Kjer, Seraina Klopfstein, Rudolf Meier, Carlo Polidori, Thomas Schmitt, Shanlin Liu, Xin Zhou, Torsten Wappler, Jes Rust, Bernhard Misof, and Oliver Niehuis. Evolutionary history of the hymenoptera. *Current Biology*, 27(7):1013–1018, 2017. doi: 10.1016/j.cub.2017.01.027. URL https://doi.org/10.1016/j.cub.2017.01.027.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

R Hackathon et al. *phylobase: Base Package for Phylogenetic Structures and Comparative Data*, 2019. URL https://CRAN.R-project.org/package=phylobase. R package version 0.8.6.

Daniel L. Rabosky. Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. *PLOS ONE*, 9(2):1–15, 2014. doi: 10.1371/journal.pone.0089543. URL https://doi.org/10.1371/journal.pone.0089543.

DL Rabosky, MC Grundler, CJ Anderson, PO Title, JJ Shi, JW Brown, H Huang, and JG Larson. Bammtools: an r package for the analysis of evolutionary dynamics on phylogenetic trees. *Methods in Ecology and Evolution*, 5:701–707, 2014.

Simon M. Reader, Yfke Hager, and Kevin N. Laland. The evolution of primate general and cultural intelligence. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 366(1567):1017–1027, 2011. doi: 10.1098/rstb.2010.0342. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2010.0342.

Liam J. Revell. phytools: An r package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, 3:217–223, 2012.

Carmela Serio, Silvia Castiglione, Gianmarco Tesone, Martina Piccolo, Marina Melchionna, Alessandro Mondanaro, Mirko Di Febbraro, and Pasquale Raia. Macroevolution of toothed whales exceptional relative brain size. *Evolutionary Biology*, 46:332–342, 2019. URL https://link.springer.com/article/10.1007/s11692-019-09485-7.

Bill Shipley. A new inferential test for path models based on directed acyclic graphs. *Structural Equation Modeling*, 7(2):206–218, 2000.

Bill Shipley. *Cause and correlation in biology: a user's guide to path analysis, structural equations and causal inference with R*. Cambridge University Press, 2016.

Graham J. Slater, Luke J. Harmon, and Michael E. Alfaro. Integrating fossils with molecular phylogenies improves inference of trait evolution. *Evolution*, 66(12):3931–3944, 2012. doi: 10.1111/j.1558-5646.2012.01723.x. URL https://doi.org/10.1111/j.1558-5646.2012.01723.x.

Tanja Stadler. *TreeSim: Simulating Phylogenetic Trees*, 2019. URL https://CRAN.R-project.org/package=TreeSim. R package version 2.4.

Theodore Stankowich, Tim Caro, and Matthew Cox. Bold coloration and the evolution of aposematism in terrestrial carnivores. *Evolution*, 65(11):3090, 2011. URL https://liverpool.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsjsr&AN=edsjsr.41317030&site=eds-live&scope=site.

Harald Stumpke. *The Snouters: Form and Life of the Rhinogrades*. Gustav Fischer Verlag, Stuttgart, 1957.

J. G. M. Thewissen, Lisa Noelle Cooper, John C. George, and Sunil Bajpai. From land to water: the origin of whales, dolphins, and porpoises. *Evolution: Education and Outreach*, 2(2):272–288, 2009. doi: 10.1007/s12052-009-0135-2. URL https://doi.org/10.1007/s12052-009-0135-2.

Wouter van der Bijl. phylopath: Easy phylogenetic path analysis in r. *PeerJ*, 6: e4718, 2018. doi: 10.7717/peerj.4718. R package version 1.1.2.

Achaz von Hardenberg and Alejandro Gonzalez-Voyer. Disentangling evolutionary cause-effect relationships with phylogenetic confirmatory path analysis. *Evolution*, 67 - 2:378–387, 2013. doi: 10.1111/j.1558-5646.2012.01790.x.

Andrea J. Webster and Andy Purvis. Testing the accuracy of methods for reconstructing ancestral states of continuous characters. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 269(1487):143–149, 2002. doi: 10.1098/rspb.2001.1873. URL https://doi.org/10.1098/rspb.2001.1873.

Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.2.1.

Guangchuang Yu. *ggimage: Use Image in 'ggplot2'*, 2019. URL https://CRAN.R-project.org/package=ggimage. R package version 0.2.4.

Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, and Tommy Tsan-Yuk Lam. ggtree: an r package for visualization and annotation of phylogenetic trees with their covariates and other associated data. *Methods in Ecology and Evolution*, 8:28–36, 2017. doi: 10.1111/2041-210X.12628. URL http://onlinelibrary.wiley.com/doi/10.1111/2041-210X.12628/abstract.

Guangchuang Yu, Tommy Tsan-Yuk Lam, Huachen Zhu, and Yi Guan. Two methods for mapping and visualizing associated data on phylogeny using ggtree. *Molecular Biology and Evolution*, 35:3041–3043, 2018. doi: 10.1093/molbev/msy194. URL https://doi.org/10.1093/molbev/msy194.