Methods in Macroecology and Macroevolution

Natalie Cooper (natalie.cooper@nhm.ac.uk)

February 2017

Contents

Chapter 1

Methods in Macroecology and Macroevolution

These practicals were designed for a module on the MSc Taxonomy and Biodiversity run by Natural History Museum and Imperial College London. Please feel free to use and share.

1.1 Module Aims

The aim of this module is to introduce you to some of the cutting-edge methods being used in macroecology and macroevolution, with particular focus on work being done right now at the Museum. Rather than concentrating on learning theory through lectures, the module will introduce you to various ways we collect, analyse and interpret macro-scale data with a large emphasis on practical work, mostly using R.

1.2 Datasets and practical handouts

All datasets and practical handouts are available on GitHub at [https://github.com/nhcooper123/TeachingMaterials/tree/master/MSc_Museum]. It can be tricky to download data from GitHub if you don't know what you're doing and/or don't want to clone the entire repository so the materials are also available at this DropBox link: [].

1.3 How to use these materials

It is possible to use these materials in a number of ways.

1. Follow this book, copy pasting code into an R script and then running it in the R console.
2. Open the .Rmd file for each practical in RStudio and use it as an RNotebook. This allows you to run chunks of code in the script and for the results to appear below the code. This is a good way to learn/teach these materials. See here for more details.

Some knowledge of R is required, but this is mostly covered in the first chapter of this book.

1.4 Module outline

These practicals fit into a broader module as follows.

- What are macroecology and macroevolution?
- Diversity indices in R. [practical]
- Visualising phylogenies in R. [practical]
- Using Big Data in conservation with the PREDICTS project. (Andy Purvis)
- Using the paleobiology database (PBDB) to get data on fossil species for macroecological and macroevolutionary analyses. (Terri Cleary)
- The comparative method, dealing with phylogenetic non-independence in comparative analyses.* PGLS in R. [practical]
- Macroevolutionary patterns.
- Models of evolution in R. [practical]
- Using museum specimen data in macroecological and macroevolutionary analyses.
- Geometric morphometrics. [practical]
- Case study using Foraminifera. (Isabel Fenton)
- Diversification rates.
- BAMM [practical]
- Critical thinking: Do you trust your methods?

Links to some of the lectures are also available here.

Chapter 2

What you need to be able to do in R before you start

Most people taking this module have used R a lot already, but it is possible you're a bit rusty, or you've found this course on GitHub and have no R experience. This isn't a problem, I will try and summarise what you need to be able to do to get these practicals running below. However, I'm not going to write a help guide to R here, if you can't work out how to open it and get started etc. I strongly recommend the book Getting Started With R or there are lots of great tutorials online.

Throughout, R code will be in shaded boxes:

```
library(ape)
```

R output will be preceded by ## and important comments will be in quote blocks:

Note that many things in R can be done in multiple ways. You should choose the methods you feel most comfortable with, and do not panic if someone is doing the same analyses as you in a different way!

2.1 Installing R (and RStudio)

- Install R from [https://cran.r-project.org]
- You can install RStudio from [http://www.rstudio.com/products/rstudio/download/]. I'd recommend trying this out if you're a beginner as it has a nicer interface.

2.2 Setting the working directory

To use the practicals you need to download all the files for each practical into a folder somewhere on your computer (I usually put mine on the Desktop). We will then tell R to look in this folder for all data etc. by **setting the working directory** to that folder.

To set the working directory you'll need to know what the **path** of the folder is. The path is really easy to find in a Windows machine, just click on the address bar of the folder and the whole path will appear. For example on my Windows machine, the path is:

```
C:/Users/Natalie/Desktop/RAnalyses
```

It's a bit trickier to find the path on a Mac, so use Google if you need help. On my Mac the path is:

```
~/Desktop/RAnalyses
```

Note that the tilde ~ is a shorthand for /Users/Natalie.

We can then set the working directory to your folder using `setwd`:

```
setwd("~/Desktop/RAnalyses")
```

Alternatively if using RStudio you use the menus to do this. Go to Session > Set Working Directory > Choose Directory.

Setting the working directory tells R which folder to look for data in (and which folder you'd like it to write results to). It saves a bit of typing when reading files into R. Now I can read in a file called `mydata.csv` as follows:

```
mydata <- read.csv("mydata.csv")
```

rather than having to specify the folder too:

```
mydata <- read.csv("~/Desktop/RAnalyses/mydata.csv")
```

> Remember if you move the data files, or the folder itself, you'll need to set the working directory again.

2.3 Using a script

Next, open a text editor. R has an inbuilt editor that works pretty well, but NotePad and TextEdit are fine too. However, I **highly** recommend using something that will highlight code for you. My personal favorite is Sublime Text 2, because you can also use it for any other kind of text editing like LaTeX, html etc. RStudio's editor is also very nice.

You should type (or copy and paste) your code into the text editor, edit it until you think it'll work, and then either paste it into R's console window, or you can highlight the bit of code you want to run and press `ctrl` or `cmd` and `enter` or `R` (different computers seem to do this differently). This will automatically send it to the console.

Saving the script file lets you keep a record of the code you used, which can be a great time saver if you want to use it again, especially as you know this code will work!

You can cut and paste code from my handouts into your script. You don't need to retype everything!

If you want to add comments to the file (i.e., notes to remind yourself what the code is doing), put a hash/pound sign (#) in front of the comment.

```
# Comments are ignored by R but remind you what the code is doing.
# You need a # at the start of each line of a comment.
# Always make plenty of notes to help you remember what you did and why
```

2.4 Installing and loading extra packages in R

To run any specialised analysis in R, you need to download one or more additional packages from the basic R installation. For these problem sets you will need to install the following packages:

- `ape`
- `geiger`

- picante
- caper
- BAMMtools

To install the package ape:

```
install.packages("ape")
```

Pick the closest mirror to you if asked.

You've *installed* the packages but they don't automatically get loaded into your R session. Instead you need to tell R to load them **every time** you start a new R session and want to use functions from these packages. To load the package ape into your current R session:

```
library(ape)
```

You can think of install.packages like installing an app from the App Store on your smart phone - *you only do this once* - and library as being like pushing the app button on your phone - *you do this every time you want to use the app.*

2.5 Loading and viewing your data in R

R can read files in lots of formats, including comma-delimited and tab-delimited files. Excel (and many other applications) can output files in this format (it's an option in the Save As dialog box under the File menu). Mostly I will give you .csv files during these practicals. As an example, here is how you would read in the tab-delimited text file called Primatedata.csv which we are going to use in the PGLS practical. Load these data as follows, assuming you have set your working directory (see step 2 above).

```
primatedata <- read.csv("Primatedata.csv")
```

read.csv reads in comma delimited files.

This is a good point to note that unless you **tell** R you want to do something, it won't do it automatically. So here if you successfully entered the data, R won't give you any indication that it worked. Instead you need to specifically ask R to look at the data.

We can look at the data by typing:

```
str(primatedata)
```

```
## 'data.frame':    77 obs. of  9 variables:
##  $ Order          : Factor w/ 1 level "Primates": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Family         : Factor w/ 15 levels "Aotidae","Atelidae",..: 2 2 2 14 3 3 3 4 4 4 ...
##  $ Binomial       : Factor w/ 77 levels "Alouatta palliata",..: 5 6 7 8 9 10 11 15 16 17 ...
##  $ AdultBodyMass_g: num  6692 7582 8697 958 558 ...
##  $ GestationLen_d : num  138 226 228 164 154 ...
##  $ HomeRange_km2  : num  2.28 0.73 1.36 0.02 0.32 0.02 0.00212 0.51 0.16 0.24 ...
##  $ MaxLongevity_m : num  336 328 454 304 215 ...
##  $ SocialGroupSize: num  14.5 42 20 2.95 6.85 ...
##  $ SocialStatus   : int  2 2 2 2 2 2 2 2 2 2 ...
```

Always look at your data before beginning any analysis to check it read in correctly.

str shows the structure of the data frame (this can be a really useful command when you have a big data file). It also tells you what kind of variables R thinks you have (characters, integers, numeric, factors etc.). Some R functions need the data to be certain kinds of variables so it's useful to check this.

As you can see, the data contains the following variables: Order, Family, Binomial, AdultBodyMass_g, GestationLen_d, HomeRange_km2, MaxLongevity_m, and SocialGroupSize.

```
head(primatedata)
```

```
##      Order      Family           Binomial AdultBodyMass_g GestationLen_d
## 1 Primates    Atelidae   Ateles belzebuth         6692.42         138.20
## 2 Primates    Atelidae   Ateles geoffroyi         7582.40         226.37
## 3 Primates    Atelidae    Ateles paniscus         8697.25         228.18
## 4 Primates Pitheciidae  Callicebus moloch          958.13         164.00
## 5 Primates     Cebidae  Callimico goeldii          558.00         153.99
## 6 Primates     Cebidae Callithrix jacchus          290.21         144.00
##   HomeRange_km2 MaxLongevity_m SocialGroupSize SocialStatus
## 1          2.28          336.0           14.50            2
## 2          0.73          327.6           42.00            2
## 3          1.36          453.6           20.00            2
## 4          0.02          303.6            2.95            2
## 5          0.32          214.8            6.85            2
## 6          0.02          201.6            8.55            2
```

This gives you the first few rows of data along with the column headings.

```
names(primatedata)
```

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

This gives you the names of the columns.

```
primatedata
```

This will print out all of the data!

This should be everything you need to know to get the practicals that follow working. Let me know if you have any problems (natalie.cooper@nhm.ac.uk).

Chapter 3

Diversity Indices in R

The aims of this practical are to learn how to use R to estimate diversity indices and species accumulation curves. You will need some of these functions to complete your paleoecology assessment.

We will be using some made up data about Pokemon sightings within the Museum.

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `vegan`
- `picante`

3.1 What are diversity indices?

A diversity index is a measure of the "diversity" of an area. *Diversity* could be measured in terms of the numbers of species (or higher taxonomic groupings like genera, families, phyla), or other metrics such as number of haplotypes if you're interested in genetic diversity, or number of functional groups for studies of functional diversity.

Many diversity indices also account for how evenly spread these different types are. For example, they identify whether there are there five species with 10 individuals of each (even), or five species with one species with 96 individuals and four species with one individual each (uneven). These differences in evenness may be as important for ecosystem function as the number of species.

There are about a million different diversity indices (OK this is a slight exaggeration but there are a lot check out the vegan package vignette and search for "diversity"), and different people prefer different measures for different questions. Amusingly the `vegan` helpfile for the function `diversify` states "these indices are all very closely related (Hill 1973), and there is no reason to despise one more than others (but if you are a graduate student, don't drag me in, but obey your Professor's orders)".

3.1.1 α, β and γ diversity (Whittaker 1960, 1972)

- α diversity is the mean species diversity in sites or habitats at a local scale.

- β diversity is turnover in α diversity among different sites.
- γ gamma diversity is diversity at the landscape level.

For example, if we count the species in Hyde Park and Green Park, we'd have a measure of α diversity for each. β diversity would measure the difference between species found in Hyde Park and those found in Green Park. γ diversity would be all the species found across London.

β diversity is the hardest to measure, and there are far more metrics for measuring this than any others.

3.2 Practical example using Pokemon...

We're going to play around with some invented data on sampling sites within the Museum and the Pokemon we've managed to find there (don't complain about my unlikely Pokemon combinations, they're just made up data with Pokemon names rather than A, B, C etc!).

```
# Load picante and vegan
library(picante)
library(vegan)
```

Next read in the data and take a look at it.

```
pokemon <- read.csv("pokemon_communities.csv")
```

```
# Look at the data
pokemon
```

For the `vegan` functions to work you need a matrix where the columns are the species names, the rows are the communities, and the contents of the matrix are the numbers of each species found at each site (or presence absence data as 0s and 1s). We can use the `sample2matrix` function in `picante` to do this easily.

Note that this only works if your first variable is the name of the site, your second is abundance and your third is the names of the species present.

```
pokemon.matrix <- sample2matrix(pokemon)

# Look at the matrix
pokemon.matrix
```

```
##        Bulbasaur Caterpie Charmander Pidgey Pikachu Psyduck Rattata
## site01         5        0          3      0       0       0       0
## site02         0        0          0      6       0       0       6
## site03         1        0          1      1       1       0      10
## site04         0        0          2      0       1       0       0
## site05         0        2          0      4       0       0       3
## site06         0        0          0      0       0       1       0
## site07         3        3          3      3       0       0       3
## site08         5        5          0      0       0       0       0
## site09         0        0          0      0       0       0       0
## site10         0        0          6      3       0       0       0
##        Spearow Squirtle Zubat
## site01       0        0     0
## site02       1        0     0
## site03       1        0     0
## site04       0        0     0
## site05       0        3     0
## site06       0        0     0
```

Figure 3.1: Who doesn't love a practical handout being interrupted by an enormous Pikachu?

```
## site07       3        3     0
## site08       0        0     0
## site09       0       10     0
## site10       0        0     2
```

3.3 Species diversity indices

3.3.1 α diversity

The simplest measure of α diversity is just the number of species in each site. You can easily extract this as follows.

```
specnumber(pokemon.matrix)
```

```
## site01 site02 site03 site04 site05 site06 site07 site08 site09 site10
##      2      3      6      2      4      1      7      2      1      3
```

Simpson's and Shannon's diversity indices can be estimated using the function `diversity`.

```
# Simpson's index
diversity(pokemon.matrix, index = "simpson")
```

```
##    site01    site02    site03    site04    site05    site06    site07
## 0.4687500 0.5680473 0.5333333 0.4444444 0.7361111 0.0000000 0.8571429
##    site08    site09    site10
## 0.5000000 0.0000000 0.5950413
```

```
# Shannon's index
diversity(pokemon.matrix, index = "shannon")
```

```
##    site01    site02    site03    site04    site05    site06    site07
## 0.6615632 0.9110175 1.1729935 0.6365142 1.3579779 0.0000000 1.9459101
##    site08    site09    site10
## 0.6931472 0.0000000 0.9949236
```

3.3.2 β diversity

The function `betadiver` allows you to estimate all the β diversity indices mentioned in Koleff et al. 2003. For help on which indices are included type:

```
betadiver(help=TRUE)
```

```
## 1 "w" = (b+c)/(2*a+b+c)
## 2 "-1" = (b+c)/(2*a+b+c)
## 3 "c" = (b+c)/2
## 4 "wb" = b+c
## 5 "r" = 2*b*c/((a+b+c)^2-2*b*c)
## 6 "I" = log(2*a+b+c) - 2*a*log(2)/(2*a+b+c) - ((a+b)*log(a+b) +
## (a+c)*log(a+c)) / (2*a+b+c)
## 7 "e" = exp(log(2*a+b+c) - 2*a*log(2)/(2*a+b+c) - ((a+b)*log(a+b)
## + (a+c)*log(a+c)) / (2*a+b+c))-1
## 8 "t" = (b+c)/(2*a+b+c)
## 9 "me" = (b+c)/(2*a+b+c)
## 10 "j" = a/(a+b+c)
## 11 "sor" = 2*a/(2*a+b+c)
```

```
## 12 "m" = (2*a+b+c)*(b+c)/(a+b+c)
## 13 "-2" = pmin(b,c)/(pmax(b,c)+a)
## 14 "co" = (a*c+a*b+2*b*c)/(2*(a+b)*(a+c))
## 15 "cc" = (b+c)/(a+b+c)
## 16 "g" = (b+c)/(a+b+c)
## 17 "-3" = pmin(b,c)/(a+b+c)
## 18 "l" = (b+c)/2
## 19 "19" = 2*(b*c+1)/(a+b+c)/(a+b+c-1)
## 20 "hk" = (b+c)/(2*a+b+c)
## 21 "rlb" = a/(a+c)
## 22 "sim" = pmin(b,c)/(pmin(b,c)+a)
## 23 "gl" = 2*abs(b-c)/(2*a+b+c)
## 24 "z" = (log(2)-log(2*a+b+c)+log(a+b+c))/log(2)
```

Note that some of these are similarity indices, and some are dissimilarity indices. See Koleff et al. 2003 for more details. Two commonly used similarity indices are Jaccard's index and Sorenson's index which can be estimated as follows (note that completely different communities get a score of 0):

```
# Jaccard's index
betadiver(pokemon.matrix, method = "j")
```

```
##           site01    site02    site03    site04    site05    site06
## site02 0.0000000
## site03 0.3333333 0.5000000
## site04 0.3333333 0.0000000 0.3333333
## site05 0.0000000 0.4000000 0.2500000 0.0000000
## site06 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## site07 0.2857143 0.4285714 0.6250000 0.1250000 0.5714286 0.0000000
## site08 0.3333333 0.0000000 0.1428571 0.0000000 0.2000000 0.0000000
## site09 0.0000000 0.0000000 0.0000000 0.0000000 0.2500000 0.0000000
## site10 0.2500000 0.2000000 0.2857143 0.2500000 0.1666667 0.0000000
##           site07    site08    site09
## site02
## site03
## site04
## site05
## site06
## site07
## site08 0.2857143
## site09 0.1428571 0.0000000
## site10 0.2500000 0.0000000 0.0000000
```

```
# Shannon's index
betadiver(pokemon.matrix, method = "sor")
```

```
##           site01    site02    site03    site04    site05    site06
## site02 0.0000000
## site03 0.5000000 0.6666667
## site04 0.5000000 0.0000000 0.5000000
## site05 0.0000000 0.5714286 0.4000000 0.0000000
## site06 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## site07 0.4444444 0.6000000 0.7692308 0.2222222 0.7272727 0.0000000
## site08 0.5000000 0.0000000 0.2500000 0.0000000 0.3333333 0.0000000
## site09 0.0000000 0.0000000 0.0000000 0.0000000 0.4000000 0.0000000
## site10 0.4000000 0.3333333 0.4444444 0.4000000 0.2857143 0.0000000
##           site07    site08    site09
```

```
## site02
## site03
## site04
## site05
## site06
## site07
## site08 0.4444444
## site09 0.2500000 0.0000000
## site10 0.4000000 0.0000000 0.0000000
```

Note that the outputs here are pairwise matrices, as these indices measure the similarity among each pair of sites. You can estimate Whittaker's original using `method = "w"` (this is a dissimilarity method so completely different communities get a score of 1).

```
betadiver(pokemon.matrix, method = "w")
```

```
##           site01    site02    site03    site04    site05    site06
## site02 1.0000000
## site03 0.5000000 0.3333333
## site04 0.5000000 1.0000000 0.5000000
## site05 1.0000000 0.4285714 0.6000000 1.0000000
## site06 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## site07 0.5555556 0.4000000 0.2307692 0.7777778 0.2727273 1.0000000
## site08 0.5000000 1.0000000 0.7500000 1.0000000 0.6666667 1.0000000
## site09 1.0000000 1.0000000 1.0000000 1.0000000 0.6000000 1.0000000
## site10 0.6000000 0.6666667 0.5555556 0.6000000 0.7142857 1.0000000
##           site07    site08    site09
## site02
## site03
## site04
## site05
## site06
## site07
## site08 0.5555556
## site09 0.7500000 1.0000000
## site10 0.6000000 1.0000000 1.0000000
```

3.3.3 γ diversity

In this example, γ diversity is the total number of species found across all sites. We can very simply calculate this in R using the following code:

```
length(unique(pokemon$Species))
```

```
## [1] 10
```

```
# To view unique species
unique(pokemon$Species)
```

```
##  [1] Bulbasaur  Charmander Pidgey     Rattata    Spearow    Pikachu
##  [7] Squirtle   Caterpie   Psyduck    Zubat
## 10 Levels: Bulbasaur Caterpie Charmander Pidgey Pikachu ... Zubat
```

Of course in this example things are simple as we only included species we actually spotted at each site. If we had a more complex dataset, we might worry about species being listed in a site with an NA or a zero

for the abundance value. In this case we can ask R to only add up species where abundance is not NA (`!is.na(pokemon$Abundance)`) and (`&`) where it is not zero (`pokemon$Abundance != 0`).

```
length(unique(pokemon$Species[!is.na(pokemon$Abundance) & pokemon$Abundance != 0]))
```

```
## [1] 10
```

If you want to test this out, we can remove the Zubat (which only occurs in site 10) and check.

```
# Replace Zubat abundance with 0
# [31,2] means the 31st row and 2nd (Abundance) column
pokemon[31, 2] <- 0

length(unique(pokemon$Species[!is.na(pokemon$Abundance) & pokemon$Abundance != 0]))
```

```
## [1] 9
```

3.4 Species accumulation curves (Colwell & Coddington 1994)

Often when we talk about species diversity we're interested in the total diversity of an area or a site. For example, if we want to conserve a patch of woodland, we might need to know how many species in total live there. Sounds easy enough right? Just go out and sample the heck out of that woodland… The problem of course is that sampling is time consuming and expensive, and in conservation we don't have much time or money. In addition, completely sampling all species in an area is difficult, especially for small, rare, shy species. Instead we often estimate species richness by various means. Species accumulation curves are one way to do this.

Species accumulation curves are graphs showing the **cumulative number of species** recorded in an area or site as a function of the **cumulative sampling effort** taken to search for them. Sampling effort can be number of quadrats, number of hours of sampling etc. for α diversity, or number of sites if trying to get an estimate of γ diversity.

The idea is that as you sample more, you should get closer to discovering all the species in the area. The first few samples you take will probably have lots of species you haven't recorded yet, but this rate of discovery should slow down. Eventually you hope that you'll stop finding any new species to record so the curve will asymptote, but in reality sampling is rarely that thorough. Luckily we can use species accumulation curves to estimate where the curve would asymptote if we kept on sampling.

3.4.1 Practical example

Let's try this for our Pokemon, how many species might be hiding in the Museum if we sampled throughly?

We can use the `pokemon.matrix` we've already created and estimate the accumulation curve using the `vegan` function `specaccum`. There are lots of methods for estimating these curves but we will use `method = "random"`. This works as follows.

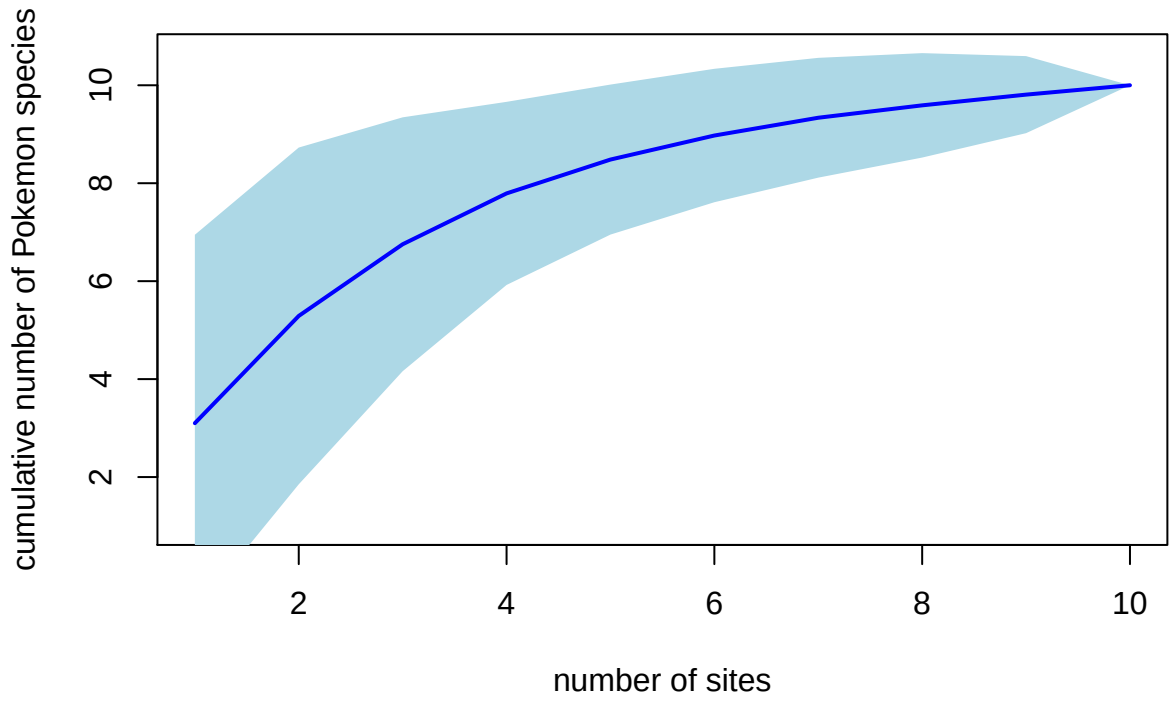
It randomly selects a site and calculates the initial richness, then randomly selects a second site and calculates the cumulative richness (i.e. the second site plus the first site), and repeats this until all the sites have been used. It then repeats this process 1000 times. You can change the number of times it does this using `permutations` but 1000 is about right - too few and the curves are not smooth, too many and it takes ages. The function outputs the mean cumulative richness and standard deviations for across all 1000 permutations. We do this because our curve will look different depending on which site we start with, and so will give a different total richness estimate. Randomising the order helps us get a better estimate of the total richness, and the standard error on that estimate.

To do this for our Pokemon:

```
pokemon.curve <- specaccum(pokemon.matrix, method = "random", permutations = 1000)

# Look at the results
pokemon.curve
```

```
## Species Accumulation Curve
## Accumulation method: random, with 1000 permutations
## Call: specaccum(comm = pokemon.matrix, method = "random", permutations = 1000)
##
##
## Sites    1.00000 2.000000 3.000000 4.000000 5.000000 6.00000 7.000000
## Richness 3.10000 5.290000 6.753000 7.792000 8.483000 8.97400 9.337000
## sd       1.92346 1.718969 1.296021 0.934671 0.765701 0.68102 0.611396
##
## Sites    8.000000 9.000000 10
## Richness 9.590000 9.809000 10
## sd       0.533089 0.393286  0
```

```
# Plot the curve
plot(pokemon.curve, ci.type = "poly", col = "blue", ci.col = "lightblue",
     lwd = 2, ci.lty = 0, xlab = "number of sites",
     ylab = "cumulative number of Pokemon species")
```

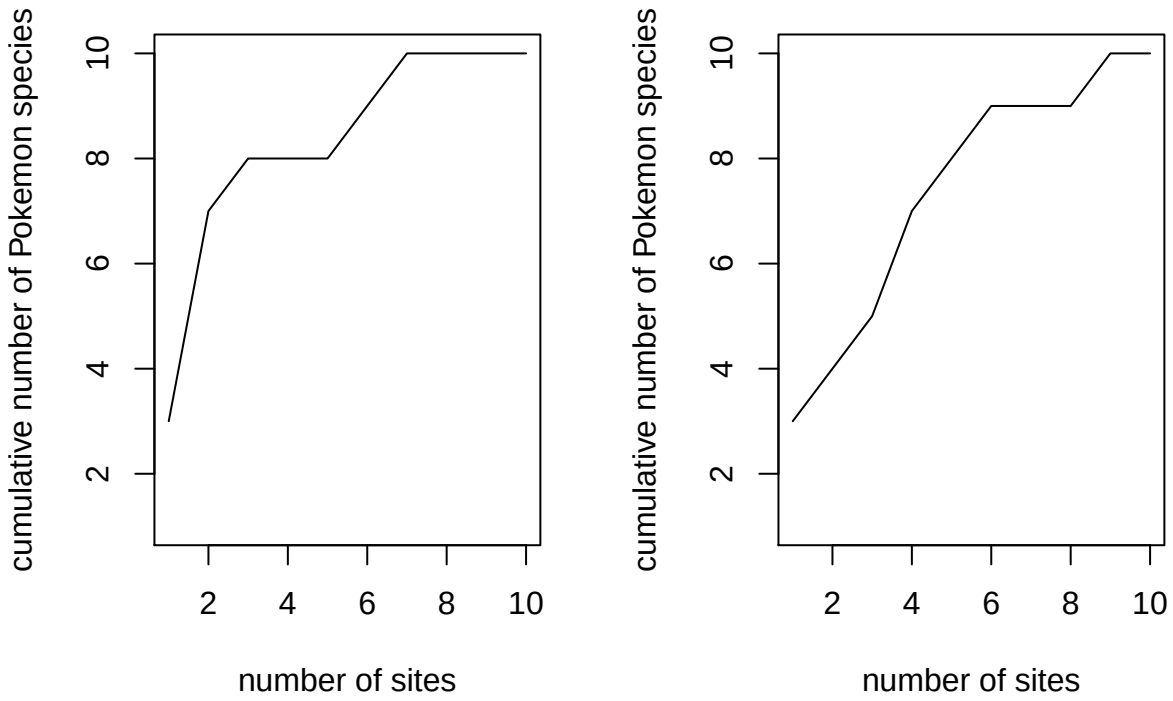


`"ci.type = "poly"` tells R that you want a shaded area showing the confidence intervals from your randomisations. You can play around with the colours etc. if you want to.

To demonstrate why we need the randomisations, look at two curves for just 1 permutation each.

```
pokemon.curve1 <- specaccum(pokemon.matrix, method = "random", permutations = 1)
pokemon.curve2 <- specaccum(pokemon.matrix, method = "random", permutations = 1)

par(mfrow = c(1,2))
```

```
plot(pokemon.curve1,
    xlab = "number of sites", ylab = "cumulative number of Pokemon species")

plot(pokemon.curve2,
    xlab = "number of sites", ylab = "cumulative number of Pokemon species")
```




```
par(mfrow = c(1,1))
```

Finally to estimate total species richness across all sites we can (again) use many different metrics. Some common ones include Chao 2 (Chao 1987), Jackknife and Bootstrapping approaches and these are easy to estimate using the `vegan` function `specpool`.

```
specpool(pokemon.matrix)
```

```
##     Species chao  chao.se jack1 jack1.se    jack2     boot   boot.se  n
## All      10 11.8 3.394113  11.8 1.272792 12.68889 10.90352 0.7502033 10
```

Estimates range from 10.9 ± 0.75 (bootstrap) to 11.8 ± 3.39 (Chao2). So we can be fairly confident there are between 10 and 15 Pokemon in the Museum.

3.5 References

- Chao, A. (1987). Estimating the population size for capture-recapture data with unequal catchability. Biometrics 43, 783–791.
- Colwell, R.K. & Coddington, J.A. (1994). Estimating terrestrial biodiversity through extrapolation. Phil. Trans. Roy. Soc. London B 345, 101–118.
- Hill, M. O. (1973) Diversity and evenness: a unifying notation and its consequences. Ecology, 54, 427–432
- Koleff, P., Gaston, K.J. and Lennon, J.J. (2003) Measuring beta diversity for presence-absence data. Journal of Animal Ecology 72, 367–382.

- Whittaker, R.H. (1960) Vegetation of Siskiyou mountains, Oregon and California. Ecological Monographs 30, 279–338.
- Whittaker, R. H. (1972). Evolution and Measurement of Species Diversity. Taxon, 21, 213-251. doi: 10.2307/1218190

3.5.1 Extra reading

- Whittaker, R. J. et al. (2001). Scale and species richness: towards a general, hierarchical theory of species diversity. Journal of Biogeography, 28, 453-470. doi:10.1046/j.1365-2699.2001.00563.x
- O'Hara, R.B. (2005). Species richness estimators: how many species can dance on the head of a pin? J. Anim. Ecol. 74, 375–386.

3.5.1.1 Nice series of papers on betadiversity

- Tuomisto, H. (2010) A diversity of beta diversities: straightening up a concept gone awry. Part 1. Defining beta diversity as a function of alpha and gamma diversity. Ecography, 33, 2-22.
- Tuomisto, H. (2010) A diversity of beta diversities: straightening up a concept gone awry. Part 2. Quantifying beta diversity and related phenomena. Ecography, 33, 23-45. doi:10.1111/j.1600-0587.2009.06148.x
- Tuomisto, H. 2010. A consistent terminology for quantifying species diversity? Yes, it does exist. Oecologia 4: 853–860. doi:10.1007/s00442-010-1812-0
- Tuomisto, H. (2011) Commentary: do we have a consistent terminology for species diversity? Yes, if we choose to use it. Oecologia, 167, 903-911.

3.5.1.2 Immensely cool new approach using methods developed by Alan Turing

- Chiu, C.H., Wang, Y.T., Walther, B.A. & Chao, A. (2014). Improved nonparametric lower bound of species richness via a modified Good-Turing frequency formula. Biometrics 70, 671–682.

Chapter 4

Visualising phylogenies in R

The aims of this practical are to introduce you to some of the fun ways you can visualise phylogenetic trees in R. The code and examples are based on Liam Revell's book chapter here.

Note that some of these plots won't work very well in RStudio. You may need to click the zoom button to see them, or resort to using the old R GUI instead.

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `ape`
- `phytools`

First load the packages we need and the data.

```
library(ape)
library(phytools)
```

Later we will use some Greater Antillean *Anolis* lizard data to add data to a phylogeny. Before we can add data to our tree, we need to load the data.

```
anoledata <- read.csv("anole.data.csv", header = TRUE)

# Look at the data
str(anoledata)
```

```
## 'data.frame':    100 obs. of  23 variables:
##  $ species             : Factor w/ 100 levels "ahli","alayoni",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ AVG.SVL               : num  4.04 3.82 3.53 4.04 4.38 ...
##  $ AVG.hl                : num  2.88 2.7 2.38 2.9 3.36 ...
##  $ AVG.hw                : num  2.36 1.99 1.56 2.37 2.69 ...
##  $ AVG.hh                : num  2.13 1.75 1.39 2.05 2.32 ...
##  $ AVG.ljl               : num  2.85 2.71 2.32 2.9 3.38 ...
##  $ AVG.outlever          : num  2.75 2.62 2.26 2.83 3.29 ...
##  $ AVG.jugal.to.symphysis: num  2.54 2.37 2.08 2.6 3.07 ...
##  $ AVG.femur             : num  2.74 2.07 2.17 2.48 2.8 ...
##  $ AVG.tibia             : num  2.69 2.02 2.09 2.34 2.69 ...
```

```
##  $ AVG.met               : num  2.25 1.54 1.55 1.87 2.18 ...
##  $ AVG.ltoe.IV           : num  2.55 1.88 1.73 2.26 2.53 ...
##  $ AVG.toe.IV.lam.width  : num  0.1795 0.0488 -0.5361 0.4904 0.8441 ...
##  $ AVG.humerus           : num  2.46 1.95 1.63 2.3 2.62 ...
##  $ AVG.radius            : num  2.27 1.69 1.4 2.09 2.34 ...
##  $ AVG.lfing.IV          : num  1.94 1.4 1.04 1.7 1.98 ...
##  $ AVG.fing.IV.lam.width : num  0.0754 -0.0739 -0.755 0.3155 0.6584 ...
##  $ AVG.pelv.ht           : num  1.99 1.51 1.19 1.87 2.1 ...
##  $ AVG.pelv.wd           : num  1.71 1.419 0.946 1.752 2.014 ...
##  $ Foot.Lam.num          : num  3.28 3.43 3.2 3.58 3.72 ...
##  $ Hand.Lam.num          : num  2.87 3.08 2.73 3.16 3.24 ...
##  $ Avg.lnSVL2            : num  3.94 3.74 3.48 3.93 4.36 ...
##  $ Avg.ln.t1             : num  4.41 4 4.37 4.44 5.04 ...
```

Now let's visualise some phylogenies!

4.1 Loading your phylogeny and data into R

4.1.1 Reading in a phylogeny from a file

To load a tree you need either the function `read.tree` or `read.nexus`. `read.tree` can deal with a number of different types of data (including DNA) whereas `read.nexus` reads NEXUS files. `elopomorph.tre` is not in NEXUS format so we use `read.tree`.

```
fishtree <- read.tree("elopomorph.tre")
```

4.1.2 Reading in a phylogeny that is already built into R

The bird and anole phylogenies are already built into R so we don't need to read them in using `read.tree`. Instead we just use:

```
data(bird.orders)
data(anoletree)
```

4.2 Basic tree viewing in R

We'll use the Elopomorpha (eels and similar fishes) tree to start as it is simple.

Let's examine the tree by typing:

```
fishtree
```

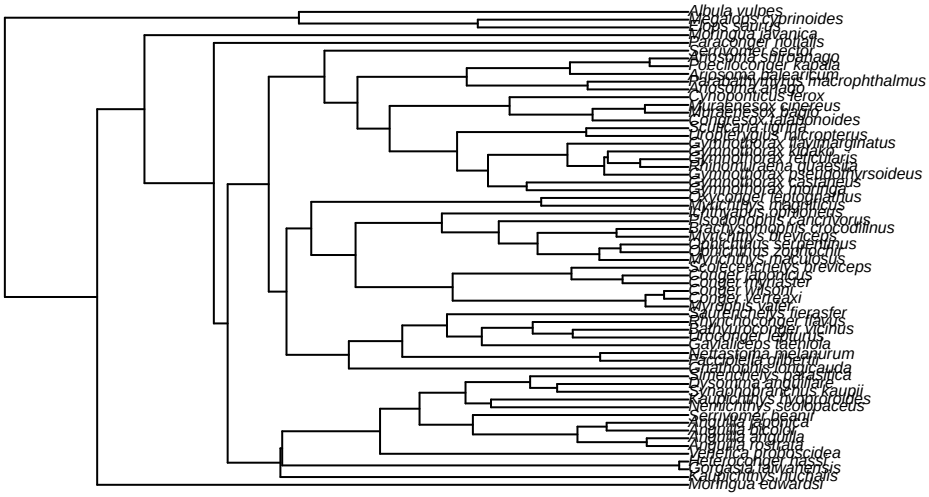
```
##
## Phylogenetic tree with 62 tips and 61 internal nodes.
##
## Tip labels:
##  Moringua_edwardsi, Kaupichthys_nuchalis, Gorgasia_taiwanensis, Heteroconger_hassi, Venefica_proboscid
##
## Rooted; includes branch lengths.
```

```
str(fishtree)
```

```
## List of 4
##  $ edge       : int [1:122, 1:2] 63 64 64 65 66 67 68 68 69 70 ...
##  $ Nnode      : int 61
##  $ tip.label  : chr [1:62] "Moringua_edwardsi" "Kaupichthys_nuchalis" "Gorgasia_taiwanensis" "Heterocong
##  $ edge.length: num [1:122] 0.0105 0.0672 0.00537 0.00789 0.00157 ...
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

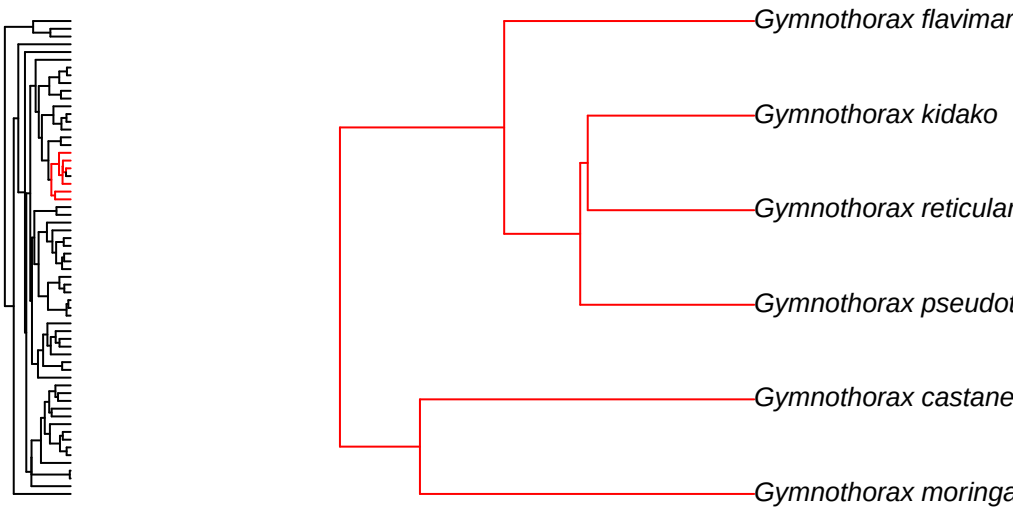
`fishtree` is a fully resolved tree with branch lengths. There are 62 species and 61 internal nodes. We can plot the tree by using the `plot.phylo` function of `ape`. Note that we can just use the function `plot` to do this as R knows if we ask it to plot a phylogeny to use `plot.phylo` instead!

```
plot(fishtree, cex = 0.5)
```



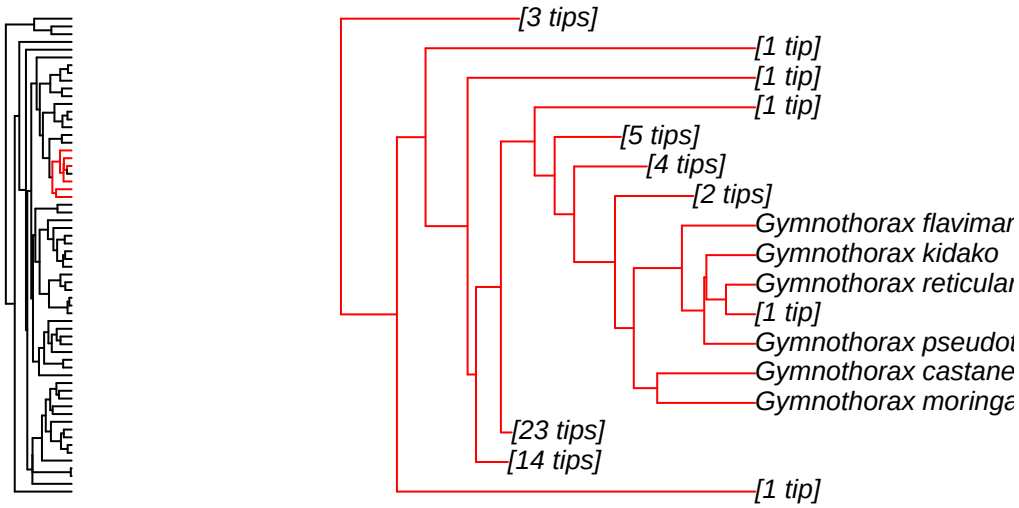
`cex = 0.5` reduces the size of the tip labels so we can read them. We can also zoom into different sections of the tree that you're interested in:

```
zoom(fishtree, grep("Gymnothorax", fishtree$tip.label), subtree = FALSE, cex = 0.8)
```



This just gives you the tree for *Micropterus* species but you can also see how the species fit into the rest of the tree using:

```
zoom(fishtree, grep("Gymnothorax", fishtree$tip.label), subtree = TRUE, cex = 0.8)
```



Note that `zoom` automatically sets the plotting window to display two plots at once. To reset this to one plot only use:

```
par(mfrow = c(1, 1))
```

To get further options for the plotting of phylogenies:

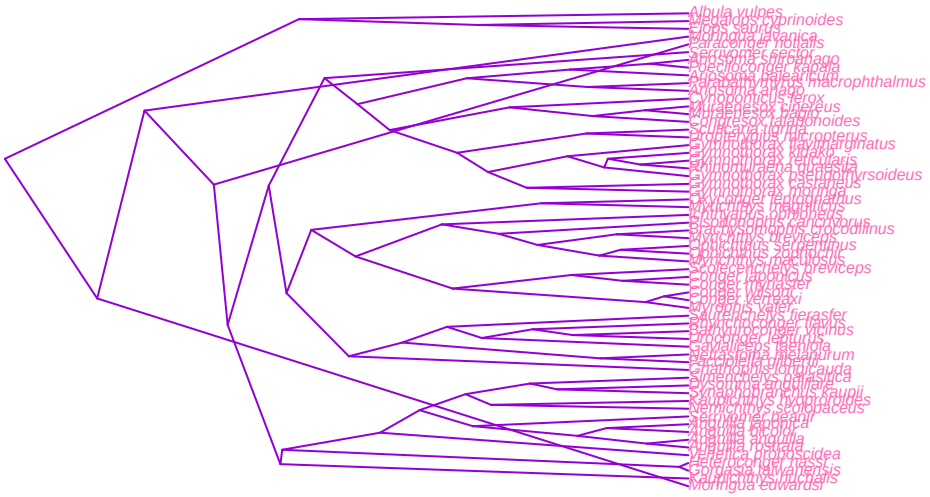
```
?plot.phylo
```

Note that although you can use `plot` to plot the phylogeny, you need to specify `plot.phylo` to find out the options for plotting trees. You can change the style of the tree (`type`), the color of the branches and tips (`edge.color`, `tip.color`), and the size of the tip labels (`cex`). Here's an fun/hideous example!

```
plot(fishtree, type = "fan", edge.color = "deeppink", tip.color = "springgreen",
     cex = 0.5)
```

Or try

```
plot(fishtree, type = "c", edge.color = "darkviolet", tip.color = "hotpink",
     cex = 0.5)
```



4.3 Adding trait data to trees in R

Note that the theory behind this is covered in more detail in the "Macroevolutionary Models in R" practicals part 1 and 2. ### Ancestral state reconstructions on discrete data We will use the bird data. Remember we already loaded the phylogeny and data as follows:

```
data(bird.orders)
```

First we will invent some data for each bird order that we can reconstruct along the tree. This creates three variables, 0, 1 and 2.

```
mydata <- factor(c(rep(0, 5), rep(1, 13), rep(2, 5)))
mydata
```

```
##  [1] 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
## Levels: 0 1 2
```

We can then use the `ape` function `ace` to reconstruct ancestral characters along the nodes of the tree. `type = d` means the character to be reconstructed is discrete.

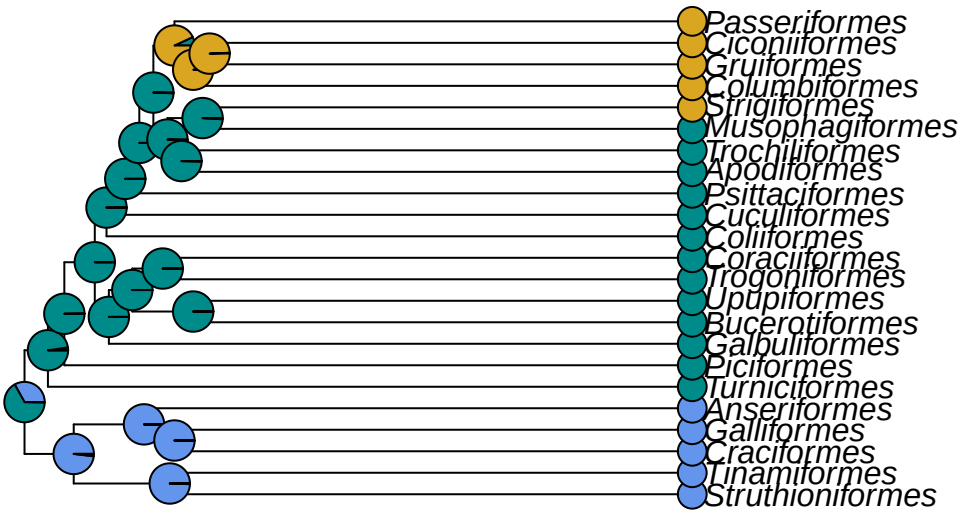
```
ancestors <- ace(mydata, bird.orders, type = "d")
```

Now we can plot this on a phylogeny. First decide which colours we'd like. To look at a list of colours in R type in `colors()`.

```
colours <- c("cornflowerblue", "cyan4", "goldenrod")
```

Now plot the tree and add labels to the tips and the nodes using the results in `ancestors`. We use `label.offset = 1` to move the labels to the right a bit so the pie charts will fit...

```
plot(bird.orders, label.offset = 1)
tiplabels(pch = 21, bg = colours[as.numeric(mydata)], cex = 2, adj = 1)
nodelabels(pie = ancestors$lik.anc, piecol = colours)
```



`pch = 21` sets the tip labels to be unfilled circles, `bg` defines the colours of the circles using the list of colours we provided, and ordering them based on what the species value was for mydata (i.e. 0, 1 or 2). `cex = 2` doubles the point size, and `adj = 1` moves the tip labels sideways so they don't obscure the ends of the branches. `pie` makes pie charts coloured using the ancestral state reconstructions in `ancestors`, and `piecol` tells it to use the colours we have defined.

4.3.1 Ancestral state reconstructions on continuous data

We are going to use the *Anolis* data to create a phylogeny with different colours for different observed and reconstructed body sizes (snout-to-vent length, SVL). Remember we already loaded the phylogeny and data as follows:

```
data(anoletree)
anoledata <- read.csv("anole.data.csv", header = TRUE, row.names = 1)
```

Note the names in `anoledata` are the species names without the Genus. In the phylogeny the species names are `Anolis_species`. So to get the two to match we need to add `Anolis_` to each name.

```
rownames(anoledata) <- paste("Anolis", rownames(anoledata), sep = "_")
```

`paste` just sticks together `Anolis` with the names in `anoles` already with an underscore (`_`) separating them (`sep = "_"`)

We then need to make sure the order of the species in the data matches that of the phylogeny.

```
anoledata <- anoledata[anoletree$tip.label, ]
```

Next we make a matrix containing only the SVL values for each *Anolis* species:

```
SVL <- as.matrix(anoledata)[,"AVG.SVL"]
```

This code selects only the variable `AVG.SVL` from `anoledata` (square brackets subset in R in the form [rows, columns]), and then uses `as.matrix` to make this into a matrix.

Take a look at `SVL`

```
head(SVL)
```

```
##        Anolis_ahli     Anolis_allogus Anolis_rubribarbus
##           4.039125           4.040138           4.078469
##       Anolis_imias      Anolis_sagrei     Anolis_bremeri
```

```
##           4.099687           4.067162           4.113371
```

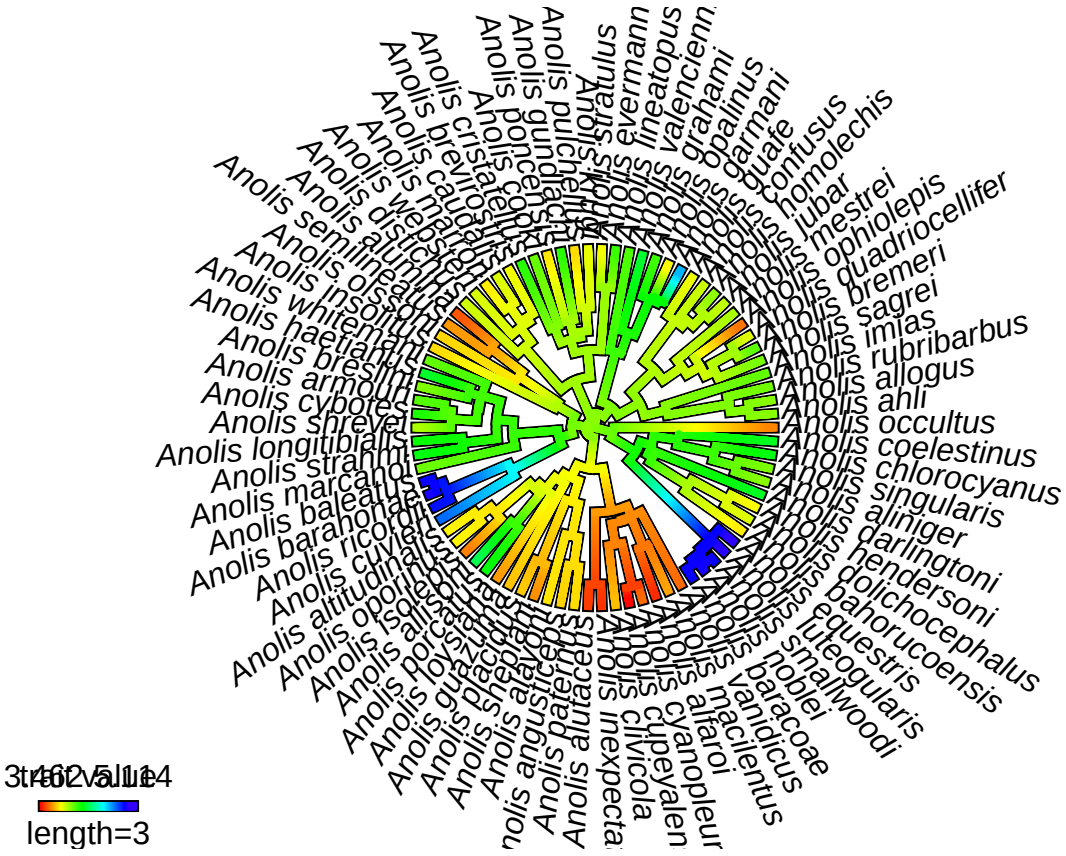
We then use the function `contMap`. `contMap` creates a tree with a mapped continuous character, i.e. where the value of the character is known at the tips, and estimated along the tree. The estimating of the character along the tree uses a Maximum Likelihood estimation procedure. Here we will tell `contMap` not to automatically plot the tree (using `plot = FALSE`) so we can make some modifications.

```
SVLplot <- contMap(anoletree, SVL, plot = FALSE)
```

Finally let's plot the tree as a fan (`legend = 10` just spreads the legend out so it is readable).

```
plot(SVLplot, type = "fan", legend = 10)
```

```
## legend scale cannot be longer than total tree length; resetting
```



4.4 Phylomorphospace plots in R

We are going to use the *Anolis* data to create a phylomorphospace plot. These plots show where species fall within a morphospace (usually based on principal components analyses of morphological variables), and then reconstructs values for the nodes of the phylogeny so the whole phylogeny can be projected into the morphospace.

We already loaded the phylogeny and data as follows:

```
data(anoletree)
anoledata <- read.csv("anole.data.csv", header = TRUE, row.names = 1)
```

As we saw above, the names of `anoles` are the species names without the Genus. In the phylogeny the species names are `Anolis_species`. So to get the two to match we need to add `Anolis_` to each name.

```
rownames(anoledata) <- paste("Anolis", rownames(anoledata), sep = "_")
```

`paste` just sticks together `Anolis` with the names in `anoles` already with an underscore (`_`) separating them (`sep = "_"`)

Next we again need to make sure the order of the species in the data matches that of the phylogeny.

```
anoledata <- anoledata[anoletree$tip.label, ]
```

We can now run a phylogenetic principal components analysis on our morphological trait data and extract the PC scores for each PC axis using `$S`.

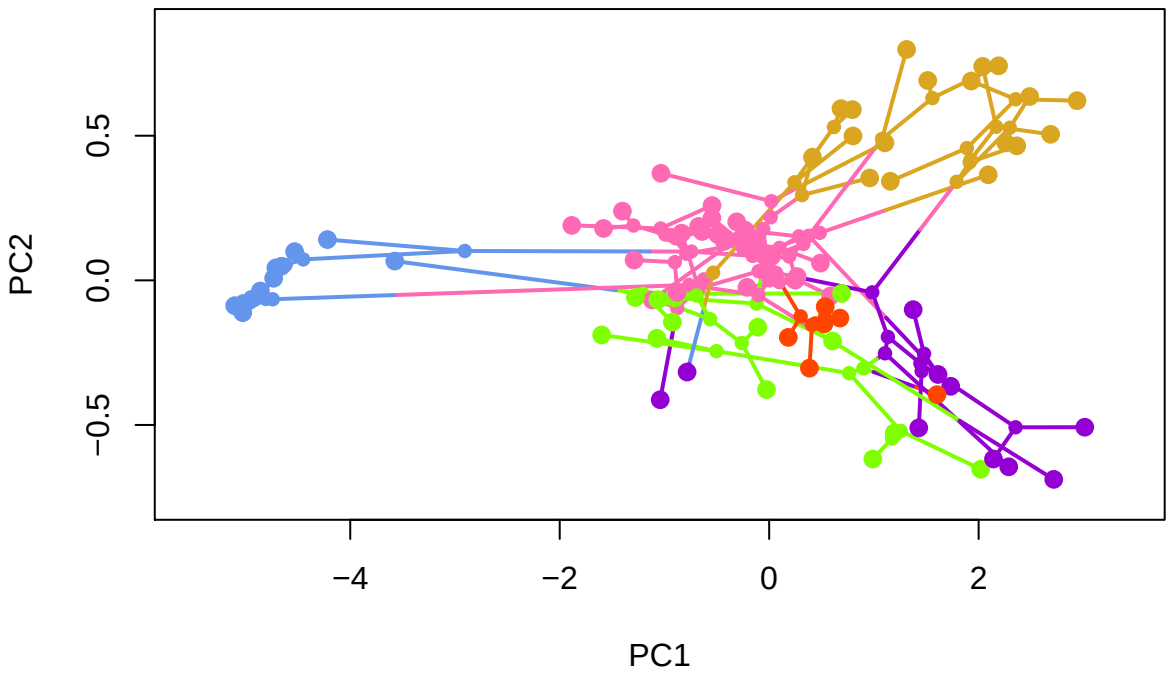
```
PC <- phyl.pca(anoletree, anoledata)$S
```

To plot the morphospace with colours, we need to define them. Here we choose six colours, and matched them with the ecomorph types listed here. These were built into `anoletree`.

```
colors <- setNames(c("cornflowerblue", "goldenrod", "chartreuse", "hotpink",
                     "orangered", "darkviolet"),
                     sort(unique(getStates(anoletree, "tips"))))
```

Finally we can make the plot, and colour by ecomorph. `label = "off"` suppresses the printing of tip labels which keeps things a bit tidier.

```
phylomorphospace(anoletree, PC[,1:2], label = "off", node.by.map = TRUE, colors = colors)
```



4.5 Phylogenies and maps!

We can also plot phylogenies attached to maps showing where species come from. We will use the bird orders tree again as it's small.

```
data(bird.orders)
```

We don't know where the birds come from, so we will simulate some latitude and longitude data as follows. This uses a function that simulates traits along the phylogeny, so close relatives should end up with more similar latitude and longitude values. I've used high rates of evolution (`sig2`) to force the points to be spread out. Of course these values will be nonsensical for birds!

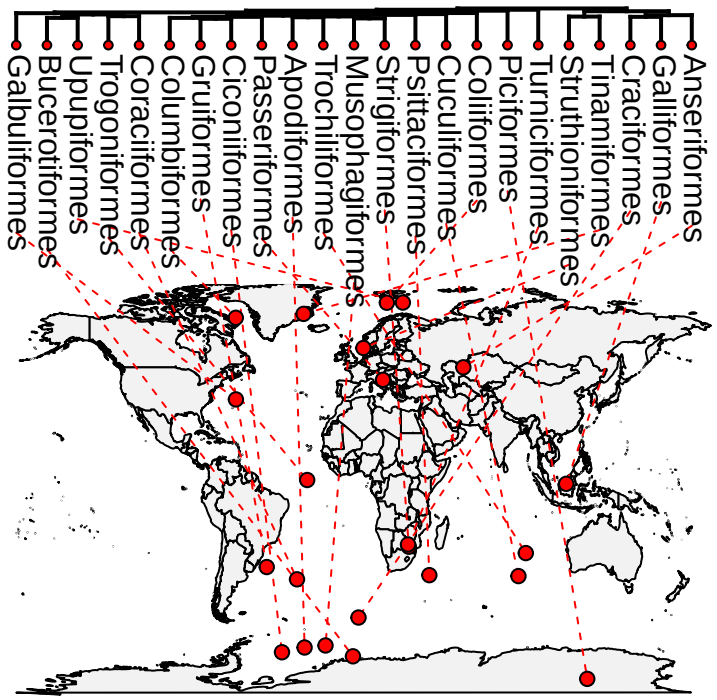
```
lat <- fastBM(bird.orders, sig2 = 100, bounds = c(-90, 90))
long <- fastBM(bird.orders, sig2 = 150, bounds = c(-180, 180))
```

Then we use the function `phylo.to.map` to match the locations with the world map. This produces some output in the form `objective 98` etc. (I have suppressed it here).

```
phylomap <- phylo.to.map(bird.orders, cbind(lat,long), plot = FALSE)
```

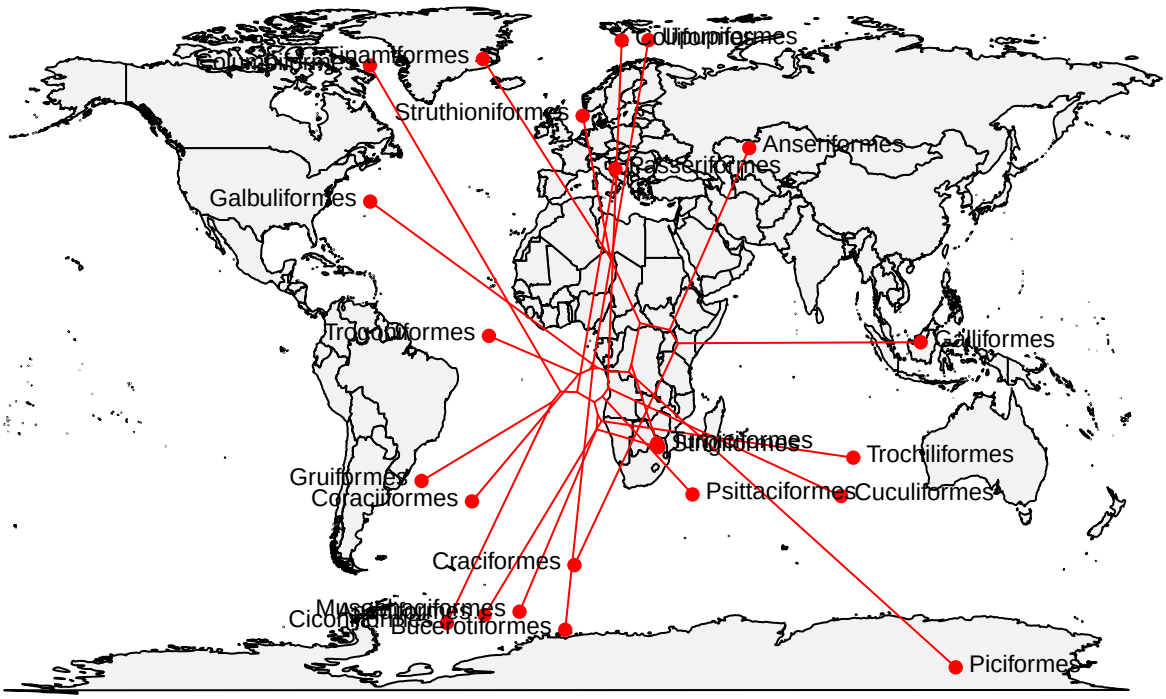
We can either put the phylogeny next to the map and draw lines to the places they come from using `type = "phylogram"`…

```
plot(phylomap,type = "phylogram", asp = 1.3, mar = c(0.1, 0.5, 3.1, 0.1))
```



…or plot the phylogeny directly onto the map using `type = "direct"`

```
plot(phylomap,type = "direct", asp = 1.3, mar = c(0.1, 0.5, 3.1, 0.1))
```



4.6 A new package for visualising trees: ggtree

`ggtree` is a newly released package that works with the popular `ggplot` framework in R. `ggplot` works by adding layers of different features together. A standard `ggplot` includes a layer that assigns the x and y variables then can have lots of other layers of points, lines, shapes and now with `ggtree` phylogenies.

Unfortunately this is broken in the most recent version of R. I've left this in, just in case they fix it soon

To use `ggtree` you will have to install it, then load it using:

```
library(ggtree)
```

Let's replicate some of the stuff we did above but using `ggtree`. As always choose whichever method you prefer when you're doing this with your own data.

```
ggtree(fishtree) +
geom_text(aes(label=label), size = 1)
```

Note that the layers here are the tree, and then the tip labels. Layers are separated by `+`

We can also zoom in on sections like with `ape`.

```
gzoom(fishtree, grep("Gymnothorax", fishtree$tip.label))
```

And we can have different shapes of phylogeny, different colours and even different line types.

```
ggtree(fishtree, color = "steelblue")
ggtree(fishtree, layout = "circular")
ggtree(fishtree, linetype = "dotted")
```

`ggtree` is really great for highlighting sections of a tree. First look at the nodes on your tree and pick an interesting one…

```
ggtree(fishtree) +
  geom_text(aes(label = node))
```

Then highlight and/or annotate…

```
ggtree(fishtree) +
geom_hilight(node = 73, fill = "steelblue", alpha = 0.6) +
geom_hilight(node = 97, fill = "darkgreen", alpha = 0.6)
```

This package has a huge amount of promise, but still a few bugs and weird features (it's designed by geneticists who are used to dealing with short species labels, hence the issues getting our species names to actually fit on the plot). The help files are also currently a bit sparse and some of the vignette examples don't work. But if you're likely to need to do any of this kind of stuff I recommend checking out the vignette and the paper:

- Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, 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* 2016, doi:10.1111/2041-210X.12628.

Chapter 5

Phylogenetic Generalised Least Squares (PGLS) in R

The aims of this practical are to learn how to use R to perform Phylogenetic Generalised Least Squares (PGLS) analyses, and to estimate phylogenetic signal.

We will be using the evolution of primate life-history variables as an example. These data come from the PanTHERIA database (Jones et al. 2009) and 10kTrees (Arnold et al. 2010). Note that this is an old version of 10kTrees, so if you want to use it in your research please download the newest version.

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `ape`
- `geiger`
- `picante`
- `caper`

5.1 Preparing for the analysis

5.1.1 Load the required packages

To begin we need to load the packages for this practical.

```
library(ape)
library(geiger)
library(picante)
library(caper)
```

5.1.2 Reading and checking your data in R

The data are in a comma-delimited text file called `Primatedata.csv`. Load these data as follows. I am assuming you have set your working directory.

```
primatedata <- read.csv("Primatedata.csv")
```

Check everything loaded correctly:

```
str(primatedata)
```

```
## 'data.frame':    77 obs. of  9 variables:
##  $ Order          : Factor w/ 1 level "Primates": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Family         : Factor w/ 15 levels "Aotidae","Atelidae",..: 2 2 2 14 3 3 3 4 4 4 ...
##  $ Binomial       : Factor w/ 77 levels "Alouatta palliata",..: 5 6 7 8 9 10 11 15 16 17 ...
##  $ AdultBodyMass_g: num  6692 7582 8697 958 558 ...
##  $ GestationLen_d : num  138 226 228 164 154 ...
##  $ HomeRange_km2  : num  2.28 0.73 1.36 0.02 0.32 0.02 0.00212 0.51 0.16 0.24 ...
##  $ MaxLongevity_m : num  336 328 454 304 215 ...
##  $ SocialGroupSize: num  14.5 42 20 2.95 6.85 ...
##  $ SocialStatus   : int  2 2 2 2 2 2 2 2 2 2 ...
```

As you can see, the data contains the following variables: Order, Family, Binomial, AdultBodyMass_g, GestationLen_d, HomeRange_km2, MaxLongevity_m, and SocialGroupSize.

5.1.3 Reading and checking your phylogeny in R

To load a tree you need either the function `read.tree` or `read.nexus`. `read.tree` can deal with a number of different types of data (including DNA) whereas `read.nexus` reads NEXUS files. We will use a NEXUS file of the consensus tree from 10kTrees.

```
primatetree <-read.nexus("consensusTree_10kTrees_Version2.nex")
```

Let's examine the tree by typing:

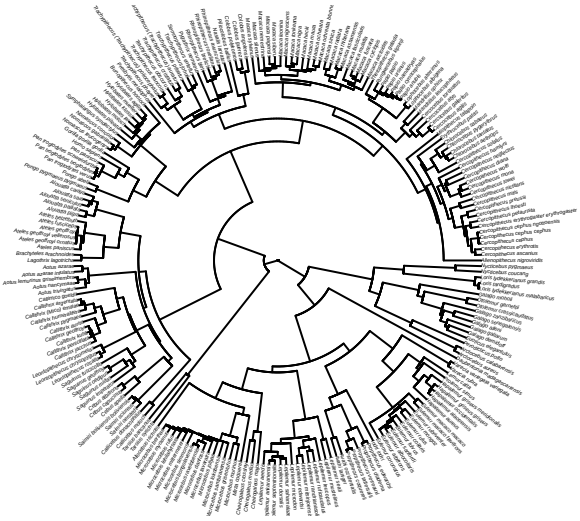
```
primatetree
```

```
##
## Phylogenetic tree with 226 tips and 221 internal nodes.
##
## Tip labels:
##  Allenopithecus_nigroviridis, Cercopithecus_ascanius, Cercopithecus_cephus, Cercopithecus_cephus_cephu
##
## Rooted; includes branch lengths.
```

```
str(primatetree)
```

```
## List of 4
##  $ edge       : int [1:446, 1:2] 227 228 229 230 231 232 233 234 234 235 ...
##  $ edge.length: num [1:446] 4.95 17.69 19.65 8.12 4.82 ...
##  $ Nnode      : int 221
##  $ tip.label  : chr [1:226] "Allenopithecus_nigroviridis" "Cercopithecus_ascanius" "Cercopithecus_cephus
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

```
# It's usually a good idea to quickly plot the tree too
plot(primatetree, cex = 0.2, typ = "fan")
```



`primatetree` is a fully resolved tree with branch lengths. There are 226 species and 221 internal nodes.

Most R functions require your tree to be dichotomous, i.e. to have no polytomies. To check whether your tree is dichotomous use `is.binary.tree`. If this is FALSE, use `multi2di` to make the tree dichotomous. This function works by randomly resolving polytomies with zero-length branches.

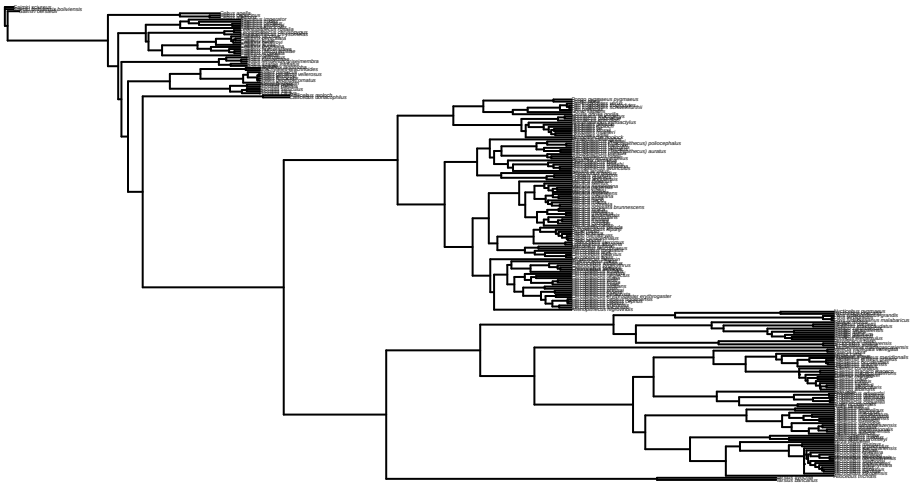
```
is.binary.tree(primatetree) # We want this to be TRUE
```

```
## [1] FALSE
```

```
primatetree <- multi2di(primatetree)
```

Most functions also require the tree to be rooted, i.e., to have one taxon designated as the outgroup. Our tree is rooted but if you wanted to change the root, or root an unrooted tree use `root`. Note that here I've just chosen a random species *Saimiri sciureus* to be the root.

```
primatetree.reroot <- root(primatetree, "Saimiri_sciureus")
plot(primatetree.reroot, cex = 0.2)
```



5.1.4 Manipulating your data and phylogeny in R

5.1.4.1 Species names with spaces

Species names in the tree cannot contain spaces so they are generally written as Genus_species (the gap between the genus name and species name replaced by an underscore _). If the species names in the data are written as Genus species with a space, then you will have to replace the spaces with _ so that they match up with the species names in the tree. You can do this as follows:

```
primatedata$Binomial <- gsub(" ", "_", primatedata$Binomial)
```

`gsub` means general substitution. It replaces any instance of the first item (here it's a space) with the second item (_) but only in the variable you tell it to (`primatedata$Binomial`).

5.1.4.2 Mismatches between species in your data and phylogeny

Often you will have data for species which are not in your phylogeny and/or species in your phylogeny which are not in your data. Some functions in R can deal with this, others will produce an error telling you the tree and data do not match (e.g. most `ape` functions). It's useful to know how to deal with this so I have provided code below.

Note that the `caper` function `comparative.data` (see below) matches up species names in the tree and data for you before you run any analyses. All `geiger` functions match the tree and the data too. However, these functions are only as good as their inputs. If you have even slightly misspelled a species name in the tree or the data it will automatically be dropped from the analyses. It is therefore **very important** to check this before running an analysis.

First we can use the function `name.check` to find out which names do not match.

```
check <- name.check(phy = primatetree, data = primatedata, data.names = primatedata$Binomial)
```

You can look at `check` by printing it, I won't do this here as it produces a lot of output but take a look on your computer.

```
check
```

`check` has two parts, `tree_not_data` for species in the tree but not in the dataset, and `data_not_tree` for species in the dataset but not in the tree.

We can remove species missing from the tree easily using `drop.tip` as we did above. You need to list the species which you do **not** want to select and then drop them from the tree instead of selecting the species you do want.

```
primatetree <- drop.tip(primatetree, check$tree_not_data)
```

In this case we don't have any species in the tree missing from the data, `data_not_tree` contains no species. However, if you do, to remove species from the data which are not in the tree you can use `match` and `subset` as follows:

```
matches <- match(primatedata$Binomial, check$data_not_tree, nomatch = 0)
primatedata <- subset(primatedata, matches == 0)
```

`==` means *equals*. So this line of code selects species which do appear in the `data_not_tree` list of species, i.e. their value from `matches` is 0.

Always check this has worked as expected by checking the data and the phylogeny. In the first instance you can just use `str` to make sure you have the expected number of species in each:

```
str(primatedata)
```

```
## 'data.frame':    77 obs. of  9 variables:
##  $ Order          : Factor w/ 1 level "Primates": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Family         : Factor w/ 15 levels "Aotidae","Atelidae",..: 2 2 2 14 3 3 3 4 4 4 ...
##  $ Binomial       : chr  "Ateles_belzebuth" "Ateles_geoffroyi" "Ateles_paniscus" "Callicebus_moloch" ...
##  $ AdultBodyMass_g: num  6692 7582 8697 958 558 ...
##  $ GestationLen_d : num  138 226 228 164 154 ...
##  $ HomeRange_km2  : num  2.28 0.73 1.36 0.02 0.32 0.02 0.00212 0.51 0.16 0.24 ...
##  $ MaxLongevity_m : num  336 328 454 304 215 ...
##  $ SocialGroupSize: num  14.5 42 20 2.95 6.85 ...
##  $ SocialStatus   : int  2 2 2 2 2 2 2 2 2 2 ...
```

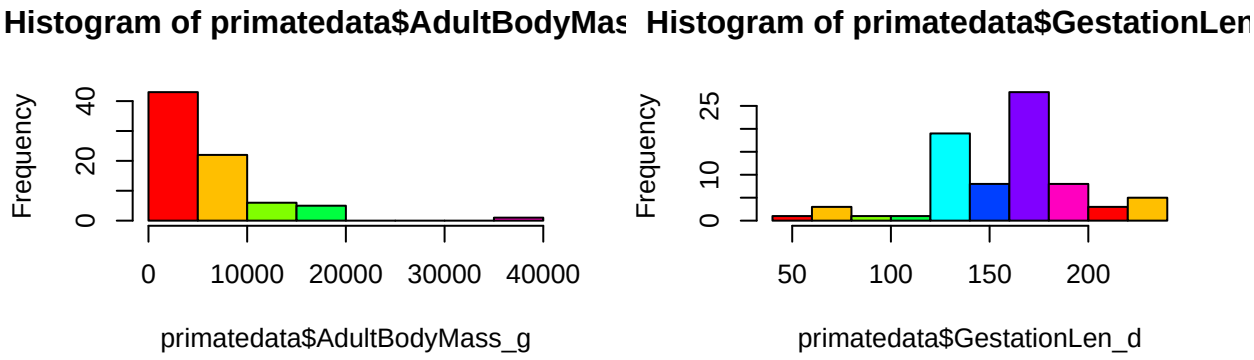
```
str(primatetree)
```

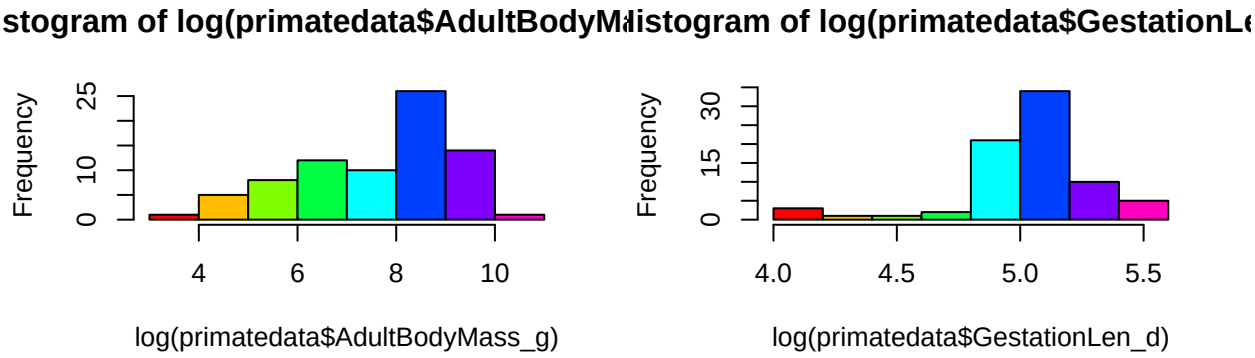
```
## List of 4
##  $ edge       : int [1:152, 1:2] 78 79 80 81 82 83 84 85 86 87 ...
##  $ edge.length: num [1:152] 4.95 17.69 19.65 8.12 4.82 ...
##  $ Nnode      : int 76
##  $ tip.label  : chr [1:77] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_mitis" "Cercop
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

5.2 Ordinary least squares (OLS) regression

Before we use phylogenetic comparative methods, we will quickly do an ordinary least squares (OLS) regression. Let's assume that we're interested in the relationship between primate body mass and gestation length. First we should look at the data. Note that both variables are highly skewed so we need to log transform them. Also note that the function `log` in R is actually natural log, not log10. And for fun let's make everything rainbow colored!

```
par(mfrow = c(2,2)) # So we can see 4 plots at once

# Raw data
hist(primatedata$AdultBodyMass_g, col = rainbow(8))
hist(primatedata$GestationLen_d, col = rainbow(8))

# Log transformed
hist(log(primatedata$AdultBodyMass_g), col = rainbow(8))
hist(log(primatedata$GestationLen_d), col = rainbow(8))
```






```
par(mfrow = c(1,1)) # So we can go back to 1 plot
```

OK let's do a linear regression of log gestation length against log adult body mass:

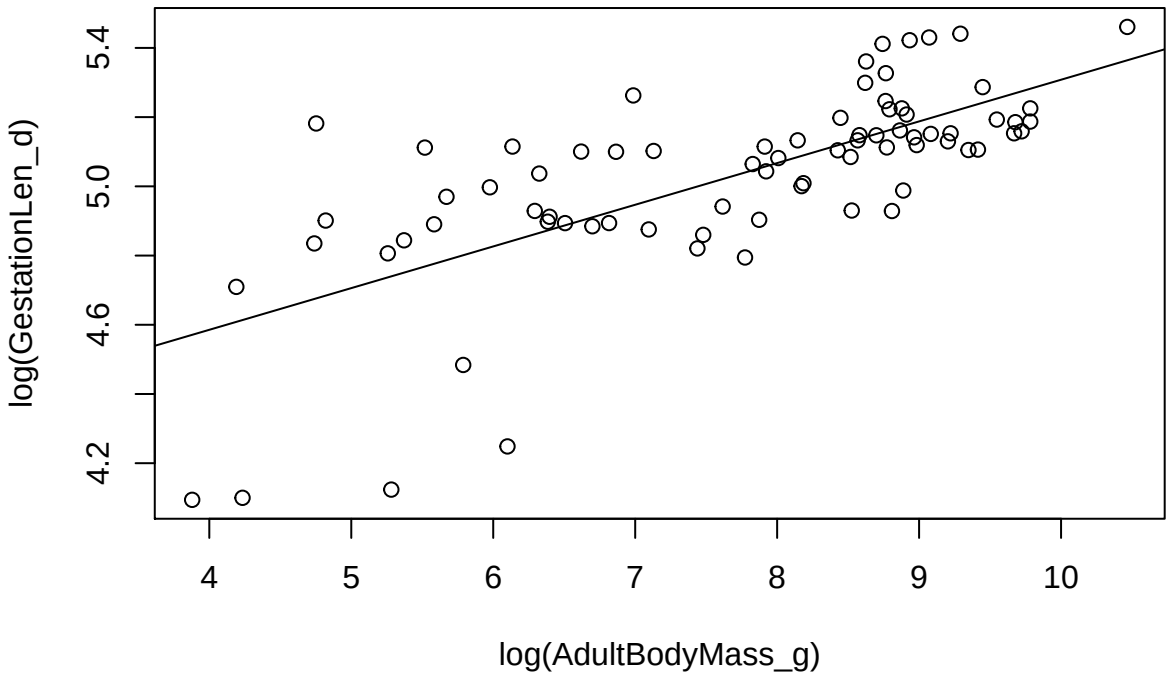
```
model.ols <- lm(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata)
summary(model.ols)
```

```
##
## Call:
## lm(formula = log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61614 -0.08279  0.00646  0.11414  0.50558
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.1037     0.1108  37.042  < 2e-16 ***
## log(AdultBodyMass_g)    0.1204     0.0141   8.544  1.1e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1977 on 75 degrees of freedom
## Multiple R-squared:  0.4933, Adjusted R-squared:  0.4865
## F-statistic: 73.01 on 1 and 75 DF,  p-value: 1.097e-12
```

The slope is positive (`0.1204 ± 0.0141`), and very significant (note that R can't display values lower than <2e-16 which is why it shows up so often).

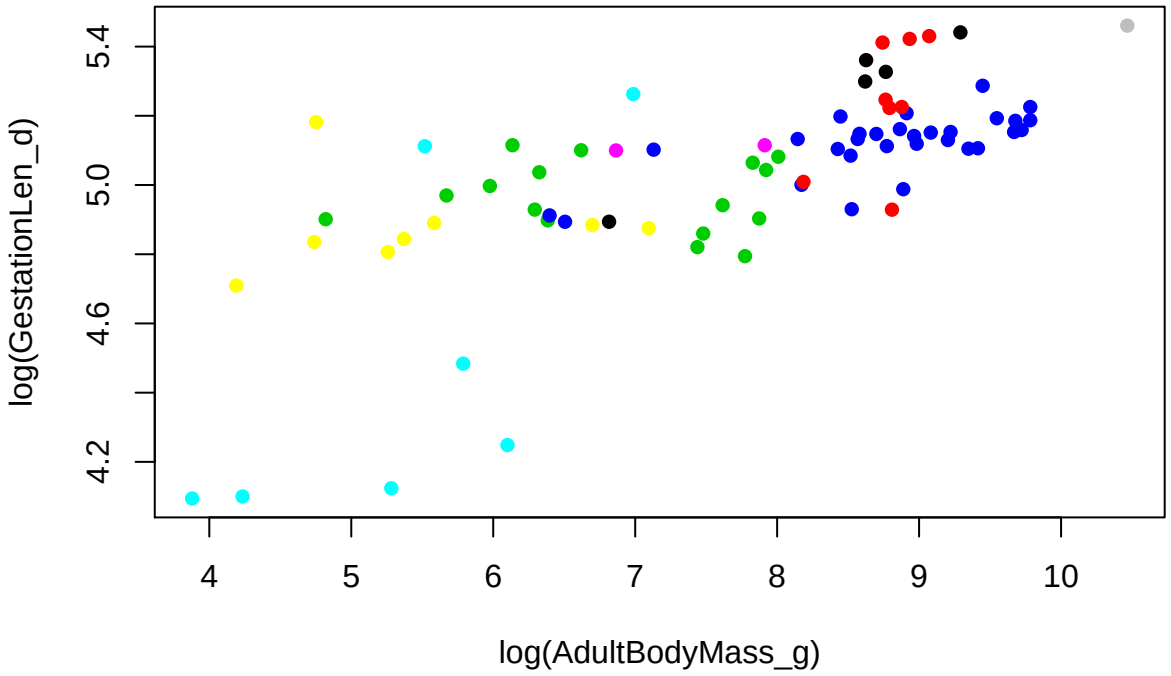
We can also plot the regression line on a scatter plot:

```
plot(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata)
abline(model.ols)
```



We can look at the phylogenetic pseudoreplication problem on the graph by colouring the points by family.

```
plot(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata, col = primatedata$Family, pch = 16)
```



5.3 Phylogenetic generalized least squares models (PGLS)

There are several ways of accounting for phylogenetic non independence in your analyses. Here we will just cover one - phylogenetic generalized least squares (PGLS). Another popular earlier method is independent contrasts. This method is really similar to PGLS, in fact it is just a special kind of PGLS where λ is equal to 1. PGLS offers some important advantages over independent contrasts. The model of trait evolution can be more flexible i.e., it can depart from a strict Brownian motion process (λ or K = 1). Different scaling parameters (λ, κ, and δ) can be incorporated in the analysis, which can significantly improve the fit of the data to the model and thus also improve the estimation of the trait correlation. Another advantage of PGLS is that the intercept of the regression is not forced to be zero.

To perform PGLS models in R, `caper` requires you to first combine the phylogeny and data into one object using the function `comparative.data`.

Note that `vcv = TRUE` stores a variance covariance matrix of your tree (you will need this for the `pgls` function). `na.omit = FALSE` stops the function from removing species without data for all variables. `warn.dropped = TRUE` will tell you if any species are not in both the tree and the data and are therefore dropped from the comparative data object. Here we won't drop any species because we already did this using `name.check`.

```
primate <- comparative.data(phy = primatetree, data = primatedata,
                            names.col = Binomial, vcv = TRUE,
                            na.omit = FALSE, warn.dropped = TRUE)
```

If you do need to drop species, this function will give a warning telling you that some species have been dropped. You can view the dropped species using:

```
primate$dropped$tips
```

```
## character(0)
```

```
primate$dropped$unmatched.rows
```

```
## character(0)
```

Always make sure you check the list of dropped species is what you expected, it often reveals typos in your species names, or mismatches in taxonomies used etc.

The function for PGLS analyses in `caper` is `pgls`. To fit a model which uses the Maximum Likelihood (ML) estimate of λ we use the following code:

```
model.pgls <- pgls(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primate, lambda = "ML")
summary(model.pgls)
```

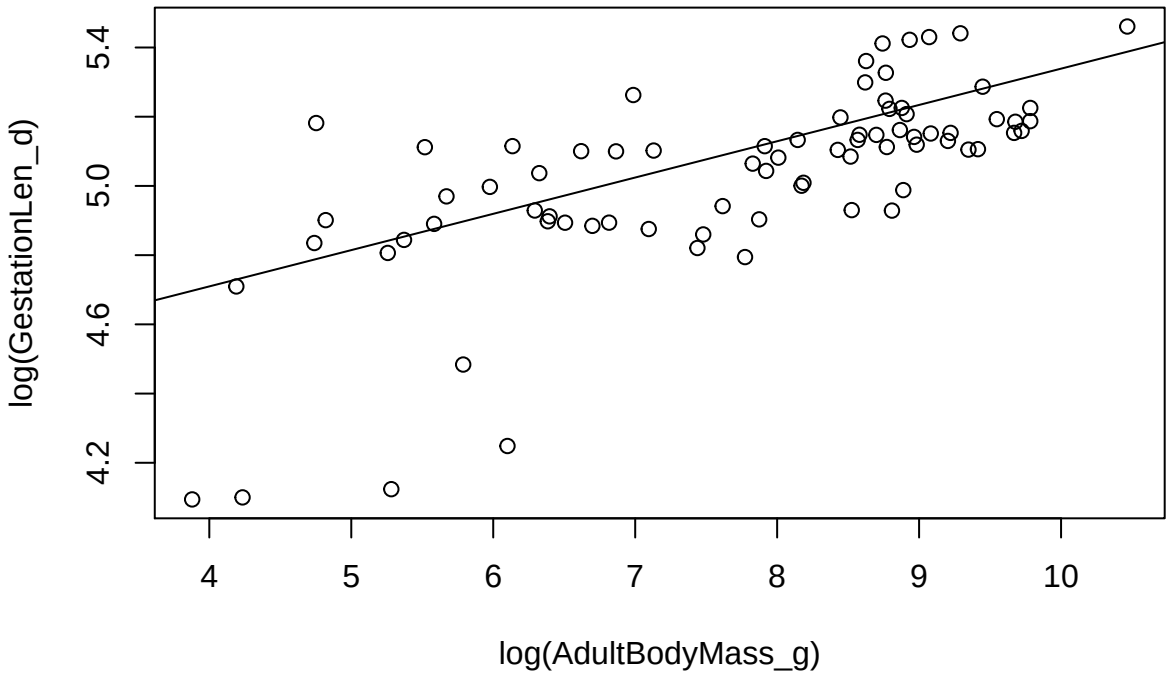
```
##
## Call:
## pgls(formula = log(GestationLen_d) ~ log(AdultBodyMass_g), data = primate,
##     lambda = "ML")
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.098899 -0.011661  0.003082  0.017758  0.075133
##
## Branch length transformations:
##
## kappa  [Fix]  : 1.000
## lambda [ ML]  : 0.892
##    lower bound : 0.000, p = 1.1435e-14
```

```
##    upper bound : 1.000, p = 0.00046393
##    95.0% CI   : (0.753, 0.967)
## delta  [Fix]  : 1.000
##
## Coefficients:
##                       Estimate Std. Error t value  Pr(>|t|)
## (Intercept)          4.290229   0.160355 26.7546 < 2.2e-16 ***
## log(AdultBodyMass_g) 0.104864   0.019628  5.3426 9.479e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0261 on 75 degrees of freedom
## Multiple R-squared: 0.2757,  Adjusted R-squared: 0.266
## F-statistic: 28.54 on 1 and 75 DF,  p-value: 9.479e-07
```

As well as the standard regression outputs, the output includes the estimated ML value of λ (0.892) and p values from likelihood ratio tests showing whether the ML λ is significantly different from 0 or 1. κ and δ are also tree transformations which can improve the fit of the data to the tree. It is also possible to use `pgls` to optimise κ or δ (using kappa = "ML" or delta = "ML" instead of lambda = "ML" in the code above). We will not cover this today. Note that optimizing more than one of these parameters at the same time is not advisable because it would be impossible to interpret the results!

We can also plot the results as follows:

```
plot(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata)
abline(model.pgls)
```



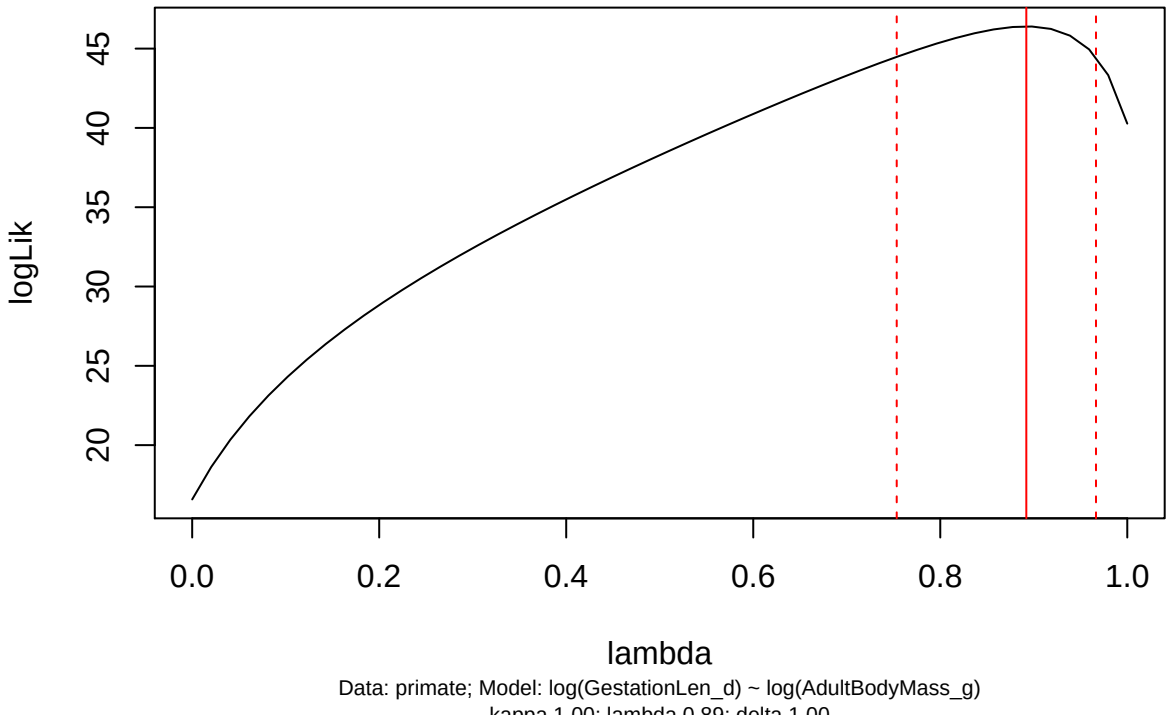
Sometimes you will find that `pgls` will not work and you get an `optim error`. This is much more common when using a Mac. To fix it all you need to do is change the bounds (upper and lower values) on the parameter being optimized, in this case λ. Just change the lower bound of λ to something a little bigger than 1e-6 until it works. For example:

```
model.pgls2 <- pgls(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primate, lambda = "ML", bounds =
```

5.3.1 Likelihood profiles for λ in PGLS models

You can look at the likelihood profiles for branch length transformations in PGLS models using `pgls.profile`:

```
lambda.profile <- pgls.profile(model.pgls, "lambda")
plot(lambda.profile)
```



This graph shows the likelihood profile of λ in our model. Ideally you want a line with an obvious peak/optimum like this, rather than a flat line which would suggest λ could be anything. You can see that the optimum (the peak of the curve) is at 0.892 as estimated in our PGLS model. The dotted red lines are the 95% confidence intervals on λ for our model. `pgls.confint` prints out these numbers in `$ci.val`

```
pgls.confint(model.pgls, "lambda")$ci.val
```

```
## [1] 0.753434 0.966543
```

Big problems with small datasets You will often find strange λ profiles when you don't have a lot of species in your data, because λ (and Blomberg's K - see below) has very low power to detect phylogenetic signal for less than 20-30 data points (see Freckleton et al. 2002 Am Nat). This means that using PGLS on small datasets is tricky - you almost always get ML λ of zero but the λ profile will show a pretty flat likelihood surface. Unfortunately people often forget to look at the λ profile so erroneously conclude that there is no phylogenetic autocorrelation in their data.

Generally I'd say don't use small datasets, however, this seems unavoidable in some fields. Therefore my advice is to (only in this situation!) ignore one of Freckleton's deadly sins (2009, JEB) and report the results from an OLS model (equivalent of PGLS with $\lambda = 0$) and also report the results from a PGLS model with λ set to 1 (equivalent to independent contrasts). This problem comes up every year and current consensus among the PCM community is that this is best solution at present, if collecting more data is really not an option!

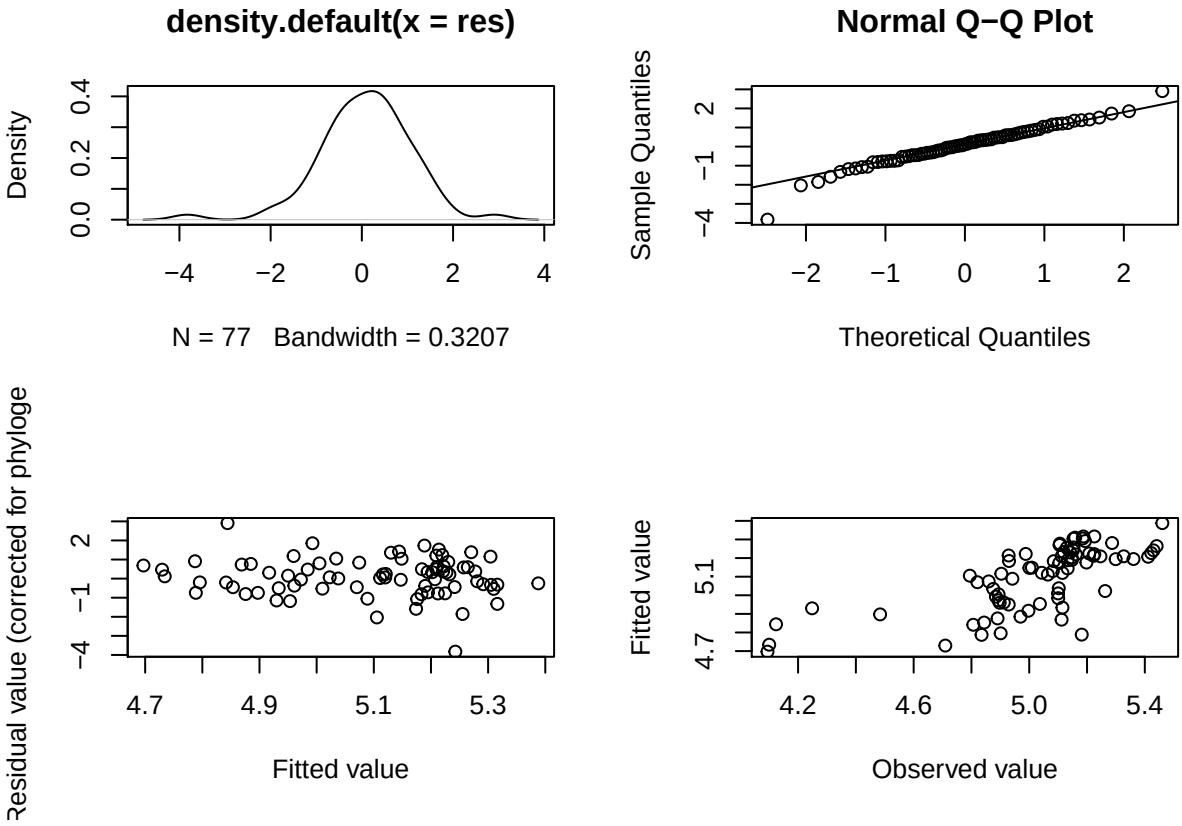
To set λ to 1 you just replace "ML" with 1

```
model.pgls3 <- pgls(log(GestationLen_d) ~ log(AdultBodyMass_g), data = primate, lambda = 1)
```

5.3.2 Model diagnostics for PGLS models

You should always check model diagnostic plots whenever you fit a model in R to check that your data meet the assumptions of the model. The method for this in PGLS is the same for OLS, independent contrasts and PGLS models (though the graphs are slightly different). To get model diagnostic plots for PGLS:

```
par(mfrow = c(2, 2))
plot(model.pgls)
```




```
par(mfrow = c(1, 1))
```

Without going into the statistical details (and if you've no idea what these plots are for I suggest looking this up online or in a stats textbook), what you are looking for in these plots is:

1. In plot one you shouldn't see any data with a studentized residual $> \pm 3$. Any species with such large residuals should be removed as these outliers may overly influence the results of the regression (see Jones and Purvis 1997). Often these are the result of measurement error associated with species pairs joined by very short branches. You should generally report results with and without outliers unless the results remain qualitatively the same.
2. The points of the Q-Q plot (plot 2) should approximately fall on the line. This tests for normality of residuals, one of the assumptions of linear models
3. Plots 3 and 4 should show a fairly random scattering of points. You want to avoid any clear patterns. The first is related to the systematic component of the model - any pattern here suggests that the model has not been correctly specified. The second is to test the assumption that variances are equal (homoscedascity).

It takes practice to know what is "good", "bad" and "acceptable" with these plots. I would say the plots above are fine, but there appear to be a couple of data points with studentized residuals $> \pm 3$ in plot 1 that should be removed, or at least checked for errors.

5.4 Estimating phylogenetic signal for one variable

5.4.1 Pagel's λ (Pagel 1997/1999, Freckleton et al 2002)

Phylogenetic signal is merely the pattern where close relatives have more similar trait values than more distant relatives (see Kamilar and Cooper 2013). Often people will mention that they "corrected for phylogeny" because of the phylogenetic signal in their variables. However, we **do not** correct for phylogeny because our variables show phylogenetic signal. We account for phylogenetic non-independence because the **residuals** from our models show phylogenetic signal (see Revell 2010). λ shown in PGLS models above is the λ for the model residuals **not** the individual variables.

Sometimes however, you might be interested in the phylogenetic signal of just one trait. λ is really easy to estimate using `caper`. To do this for log GestationLen_d:

```
lambdaGL <- pgls(log(GestationLen_d) ~ 1, data = primate, lambda = "ML")
summary(lambdaGL)
```

```
##
## Call:
## pgls(formula = log(GestationLen_d) ~ 1, data = primate, lambda = "ML")
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.128354 -0.014460  0.001389  0.017572  0.074329
##
## Branch length transformations:
##
## kappa  [Fix]  : 1.000
## lambda [ ML]  : 0.948
##    lower bound : 0.000, p = < 2.22e-16
##    upper bound : 1.000, p = 0.030039
##    95.0% CI   : (0.859, 0.996)
## delta  [Fix]  : 1.000
##
## Coefficients:
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  5.00514    0.11723  42.694 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0334 on 76 degrees of freedom
## Multiple R-squared:     0,   Adjusted R-squared:     0
## F-statistic:   NaN on 0 and 76 DF,  p-value: NA
```

Note that by replacing the explanatory variables with 1 you are just investigating the relationship between log gestation length and the phylogeny. Thus the λ value is the λ estimate for log GestationLen_d and the p values are from likelihood ratio tests showing whether the ML λ is significantly different from 0 (no phylogenetic signal) or 1 (the expectation under Brownian motion).

5.4.2 Blomberg's K (Blomberg et al 2003)

To estimate Blomberg's K we use the `Kcalc` function in `picante`. First you need to set up a new vector with the values for the variable you are interested in, here the log of gestation length, and with a names attribute with the names of your species. Here I'll call this vector `lngest`.

```
lngest <- log(primatedata$GestationLen_d)
names(lngest) <- primatedata$Binomial
```

We can then calculate K for log gestation length:

```
Kcalc(lngest[primatetree$tip.label], primatetree)
```

```
##           [,1]
## [1,] 0.7757771
```

`Kcalc` (and `phylosignal`) require the trait data to be in the same order as the tree tip labels. `lngest[primatetree$tip.label]` selects gestation lengths from the species in the same order as they are in the tree (square brackets `[ ]` are used in R to subset data).

K for log gestation length is 0.7758. As with λ above, we're interested in whether the value of K is significantly different from what we'd expect by chance. We can do this using the function `phylosignal`. This function randomly assigns the trait values to the species and then calculates K. This is repeated 1000 times (if `reps = 1000`) and the observed value of K is then compared to the randomized values to determine its significance.

```
phylosignal(lngest[primatetree$tip.label], primatetree, reps = 1000)
```

```
##           K PIC.variance.obs PIC.variance.rnd.mean PIC.variance.P
## 1 0.7757771      0.001660764            0.01192317    0.000999001
##   PIC.variance.Z
## 1      -2.403106
```

In this case the observed K is significantly higher than the random values of K (`PIC.variance.P` < 0.001).

Note this is a randomization test so your output here will not be identical to mine.

5.5 References

- Arnold, C., L. J. Matthews, and C. L. Nunn. 2010. The 10ktrees website: a new online resource for primate phylogeny. Evolutionary Anthropology: Issues, News, and Reviews 19:114–118.
- Blomberg, S. P., T. Garland, and A. R. Ives. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. Evolution 57:717–745.
- Freckleton, R. P., P. H. Harvey, and M. Pagel. 2002. Phylogenetic analysis and comparative data: a test and review of evidence. The American Naturalist 160:712-726.
- Freckleton, R. P. (2009) The seven deadly sins of comparative analysis. Journal of Evolutionary Biology, 22, 1367–1375.
- Jones, K. E., J. Bielby, M. Cardillo, S. A. Fritz, J. O'Dell, C. D. L. Orme, K. Safi, W. Sechrest, E. H. Boakes, C. Carbone, et al. 2009. Pantheria: a species-level database of life history, ecology, and geography of extant and recently extinct mammals: Ecological archives e090-184. Ecology 90:2648–2648.
- Kamilar, J. M., & Cooper, N. 2013. Phylogenetic signal in primate behaviour, ecology and life history. Phil. Trans. R. Soc. B, 368(1618), 20120341.

- Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature, 401(6756), 877-884.
- Revell, L. J. 2010. Phylogenetic signal and linear regression on species data. Methods in Ecology and Evolution, 1, 319-329.

5.5.1 Extra Reading

- Losos, J.B. (2011) Seeing the forest for the trees: the limitations of phylogenies in comparative biology. The American Naturalist, 177, 709–727.

Chapter 6

Macroevolutionary models in R: Part 1 - continuous traits

The aims of this practical are to learn how to use R to fit macroevolutionary models in R to continuous traits.

We will be using the evolution of magical creature life-history variables as an example. The data includes body mass (average adult size at rest) in kg, social status (1 = solitary, 2 = social), habitat (1 = terrestrial, 2 = aquatic, 3 = volant) and magical power (in thaum - with thanks to Terry Pratchett for the units). These data are invented, so please don't get too upset if I've misclassified anything!

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `ape`
- `geiger`
- `OUwie`

This practical is in two parts, Part 2 deals with discrete traits.

6.1 Preparing for the analysis

6.1.1 Load packages, read in the data and the tree

This is the same as we did in the PGLS practical, so I won't give detailed instructions here.

```
# Load packages
library(ape)
library(geiger)
library(OUwie)
```

```
# Read in data
magicaldata <- read.csv("magicalcreatures.csv")
```

```
# Check data is loaded correctly
str(magicaldata)
```

```
## 'data.frame':    30 obs. of  5 variables:
##  $ Species     : Factor w/ 30 levels "Acromantula",..: 22 24 19 5 7 16 21 25 30 29 ...
##  $ BodySize_kg : num  1.5 50 0.5 100 8000 1 2.5 6 800 600 ...
##  $ SocialStatus: int  2 2 2 1 1 2 2 1 1 2 ...
##  $ Habitat     : int  1 1 2 1 3 1 1 3 3 3 ...
##  $ Magic_thaum : num  106.3 86.7 98.6 57 131.9 ...
```

```
# Read in tree
magicaltree <- read.nexus("magicaltree.nex")
# Check tree is loaded correctly
str(magicaltree)
```

```
## List of 4
##  $ edge       : int [1:52, 1:2] 28 29 30 31 31 30 32 33 33 34 ...
##  $ edge.length: num [1:52] 136.86 65.69 97 2.92 2.92 ...
##  $ Nnode      : int 26
##  $ tip.label  : chr [1:27] "Doxie" "Bowtruckle" "Hinkypuff" "Grindylow" ...
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

6.1.2 Modify the tree and data so they can be used in the analyses.

Again we did this in the PGLS practical. Please remind yourself of what these steps are needed for.

```
# Ensure tree is fully bifurcating
magicaltree <- multi2di(magicaltree)

# Replace spaces with underscores in the species names
magicaldata$Species <- gsub(" ", "_", magicaldata$Species)

# Add species names to row names
row.names(magicaldata) <- magicaldata$Species
```

For some weird reason the `geiger` function we need (`treedata` see below) won't work if you input a dataset with variables that are characters i.e. words or letters. Our taxonomic variable Species is a character so we need to exclude it from the data. Note for your own data you'd need to remove all character variables (or recode them as 0,12 etc.). We will do this by making a new dataset called magicaldata2.

```
magicaldata2 <- magicaldata[, 2:5]
```

Here the [ ] tells R we want to subset the dataset. R data frames are always described by [X,Y] where X is rows and Y is columns. So [1, 1] will select the entry in the first column and the first row of the data frame. [, 2:2] selects all rows but only columns 2 to 5. These are the columns containing our numeric variables.

We then need to match the species in tree to those in the dataset as in the PGLS practical. **Note that we are using `magicaldata2` here.**

```
match.species <- treedata(magicaltree, magicaldata2)

mytree <- match.species$phy
mydata <- match.species$data
```

6.2 Models of continuous trait evolution

For fitting models of evolution to continuous data we will use the `fitContinuous` function in the R package `geiger`. `fitContinuous` is a likelihood based method, so the output will give the maximum likelihood (ML) estimates of the parameters. Bayesian methods are becoming preferred for these kinds of analyses and `fitContinuousMCMC` will perform these analyses. However, due to time constraints we will not cover this function. As an example, let's look at the evolution of `log(body size)` in magical creatures. We'll fit three evolutionary models – the Brownian motion (BM) model, the Ornstein-Uhlenbeck (OU) model and the Early Burst (EB) model. `fitContinuous` can also fit several other models. For more details look at the help file by typing: `?fitContinuous`

6.2.1 The Brownian motion (BM model)

The Brownian motion model (Cavalli-Sforza and Edwards, 1967; Felsenstein, 1973) is assumed to be the underlying mode of evolution in the majority of phylogenetic comparative methods (though this assumption is rarely tested; Freckleton and Harvey 2006). In the model, a trait X evolves at random at a rate σ:

$dX(t) = \sigma dW(t)$

where $W(t)$ is a white noise function and is a random variate drawn from a normal distribution with mean 0 and variance σ^2. This model assumes that there is no overall drift in the direction of evolution (hence the expectation of $W(t)$ is zero) and that the rate of evolution is constant. Because the direction of change in trait values at each step is random, Brownian motion is often described as a "random walk" (note that you can also fit models where $W(t)$ is not zero and there is drift in the direction of evolution. This is known as the drift model, or Brownian motion with a trend. We will not cover this here).

The model assumes the correlation structure among trait values is proportional to the extent of shared ancestry for pairs of species. This means that close relatives will be more similar in their trait values than more distant relatives. It also means that variance in the trait will increase (linearly) in proportion to time. The model has two parameters, the Brownian rate parameter, σ^2 and the state of the root at time zero, $X(0)$. `fitContinuous` estimates σ^2 and $X(0)$. Note that σ^2 can be used as a measure of the rate of trait evolution (Cooper and Purvis, 2010; Cooper et al., 2011).

6.2.2 The Ornstein-Uhlenbeck (OU) model

The Ornstein-Uhlenbeck (OU) model (Hansen, 1997; Butler and King, 2004) is a random walk where trait values are pulled back towards some "optimal" value with an attraction strength proportional to the parameter α. The model has the following form:

$dX(t) = -\alpha(X(t) - \mu) + \sigma dW(t)$

Note that this model has two parameters in addition to those of the Brownian model, α and μ. The parameter m is a long-term mean, and it is assumed that species evolve around this value. α is the strength of evolutionary force that returns traits back towards the long-term mean if they evolve away from it. α is sometimes referred to as the "rubber band" parameter because of the way it forces traits back towards μ. The OU model was introduced to population genetics by Lande (1976) to model stabilizing selection in which the mean was recast as a fitness optimum on an adaptive landscape. The process operating in comparative data is analogous, although clearly is not stabilizing selection (despite being sometimes referred to as such). The model has four parameters, the Brownian rate parameter, σ^2, the state of the root at time zero, $X(0)$, the attraction strength or "rubber band" parameter, α, and the long-term mean, μ. `fitContinuous` estimates σ^2, $X(0)$, and α. It does not estimate μ but in this implementation of the model, μ is equivalent to $X(0)$. Note that if α is close to zero then evolution is approximately Brownian.

6.2.3 The Early Burst (EB) model

The Early Burst (EB) model (Harmon et al. 2010, also called the ACDC model; Blomberg et al. 2003) is a Brownian motion/random walk model where the rate of evolution decreases exponentially through time under the model:

$r(t) = \sigma^2 e(at)$

Where $r(t)$ is the rate of evolution at time t, σ^2 is the initial value of the Brownian rate parameter, i.e. the initial rate of evolution, a is the rate change parameter, and t is time. The value of a is generally less than or equal to 0 (note that you can force a to be greater than zero by changing the bounds - see section 3.4 - however, this will only work if you have fossil species in your data; Slater et al. 2012). When a is negative, rates of evolution decrease through time. The model fits traits where diversification occurs most rapidly early in a lineage and slows as the lineage approaches the present, so that subclades tend to retain their differences through time. This is consistent with a clade radiating adaptively into a fixed set of niches and has been used as evidence of niche-filling modes of evolution (Harmon et al., 2010; Cooper and Purvis, 2010). The model has three parameters, the Brownian rate parameter, σ^2, the state of the root at time zero, $X(0)$, and the rate of change parameter, a. `fitContinuous` estimates σ^2, $X(0)$ and a. Note that if a is close to zero then evolution is approximately Brownian. Note that although many people report a values when reporting the results of fitting an Early Burst model, it is often more intuitive to report the rate half-life, $t_{\frac{1}{2}}$ (Slater and Pennell, 2014). This is calculated as:

$t_{\frac{1}{2}} = \frac{log(2)}{|a|}$

It can be interpreted as the time it takes for the rate of evolution of the trait to halve.

6.3 Fitting the models using `fitContinuous`

First we will fit a Brownian motion (BM) model to log(body mass):

```
BM <- fitContinuous(mytree, log(mydata[,"BodySize_kg"]), model = c("BM"))
```

To look at the output type:

```
BM
```

```
## GEIGER-fitted comparative model of continuous data
##  fitted 'BM' model parameters:
##  sigsq = 2.207510
##  z0 = 4.304110
##
##  model summary:
##  log-likelihood = -98.541855
##  AIC = 201.083711
##  AICc = 201.583711
##  free parameters = 2
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
```

```
##  'opt' -- maximum likelihood parameter estimates
```

The maximum likelihood estimates (lnL) of the model parameters are found near the top of the output. In a Brownian motion (BM) model we estimate the Brownian rate parameter, σ^2 or sigsq in the output above, which is 2.208 and the value of the trait at the root of the tree, $X(0)$ or z0 in the output above, which is 4.304. Other useful things in the output are the maximum-likelihood estimate (lnL) of the model (log-likelihood), the Akaike Information Criterion (AIC), sample-size corrected AIC (AICc) and the number of model parameters (free parameters) also known as k in the literature. We will return to the AIC values below.

To fit an Ornstein-Uhlenbeck model to log(body mass) we only need to change the model in the formula we used above:

```
OU <- fitContinuous(mytree, log(mydata[,"BodySize_kg"]), model = c("OU"))
```

To look at the output type:

```
OU
```

```
## GEIGER-fitted comparative model of continuous data
##  fitted 'OU' model parameters:
##  alpha = 2.718282
##  sigsq = 60.982369
##  z0 = 3.203383
##
##  model summary:
##  log-likelihood = -70.843002
##  AIC = 147.686004
##  AICc = 148.729482
##  free parameters = 3
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 0.14
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

As above, the maximum likelihood estimates (lnL) of the model parameters are found near the top of the output. In an Ornstein-Uhlenbeck (OU) model we estimate the Brownian rate parameter, σ^2 or sigsq in the output above, the value of the trait at the root of the tree, $X(0)$ or z0 in the output above, and the selection strength parameter, α or alpha in the output above. As alpha = 2.718 here, it suggests that there is evolution towards a body mass optimum.

Finally, to fit an Early Burst (EB) model to log(body mass):

```
EB <- fitContinuous(mytree, log(mydata[,"BodySize_kg"]), model = c("EB"))
```

To look at the output type:

```
EB
```

```
## GEIGER-fitted comparative model of continuous data
##  fitted 'EB' model parameters:
##  a = -0.000001
```

```
##  sigsq = 2.208170
##  z0 = 4.304129
##
##  model summary:
##  log-likelihood = -98.542390
##  AIC = 203.084780
##  AICc = 204.128258
##  free parameters = 3
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 0.55
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

As above, the maximum likelihood estimates (`lnL`) of the model parameters are found near the top of the output. In an Early Burst (EB) model we estimate the Brownian rate parameter, σ^2 or `sigsq` in the output above, the value of the trait at the root of the tree, $X(0)$ or `z0` in the output above, and the rate of change parameter, a. Here `a` is very close to 0 indicating that the rate of log(body mass) evolution in magical creatures has not decreased through time.

We can also extract the rate half-life, $t_{\frac{1}{2}}$, for this model as follows:

```
log(2)/abs(EB$opt$a)
```

```
## [1] 693144.5
```

For these data, the rate half-life is almost 700,000 time units, much greater than the total time represented on the tree! This means over the course of magical creature evolution, body size rate has not halved (yet).

6.3.1 Comparing models of evolution

Often we want to know which of the models fits our variable best. We can use `fitContinuous` to fit the models we are interested in and can then compare them using AIC. We can extract the AICs from the models we fitted above as follows:

```
BM$opt$aic
```

```
## [1] 201.0837
```

```
OU$opt$aic
```

```
## [1] 147.686
```

```
EB$opt$aic
```

```
## [1] 203.0848
```

The "best" model is the one with the smallest AIC, in this case the OU model. There is much debate about how big of a difference in AIC values can be classed as substantial improvement to a model fit (it usually ranges from 2-10 AIC units). Generally we use 4 units, so EB probably doesn't fit substantially better than BM, but OU is substantially better than BM and EB.

Alternatively we can use ΔAIC or AIC weights to compare our models using the following code and the `geiger` function `aicw`:

```
aic.scores <- setNames(c(BM$opt$aic, OU$opt$aic, EB$opt$aic), c("BM","OU","EB"))
aicw(aic.scores)
```

```
##         fit    delta           w
## BM 201.0837 53.39771 2.540009e-12
## OU 147.6860  0.00000 1.000000e+00
## EB 203.0848 55.39878 9.339176e-13
```

`aicw` outputs the AIC (`fit`), ΔAIC (`delta`) and AIC weights (`w`) for each of the models we fitted. The best model is the model with ΔAIC = 0 or with AICw closest to 1. Using ΔAIC we can conclude that the OU model is the best fit to the data.

6.3.2 Problems with convergence and bounds

Above we have mentioned the default bounds on each parameter. Sometimes these need to be changed because the model will not converge. This happens when the likelihood surface has long flat ridges that cause the likelihood search to get "stuck" (this is particularly common under the OU model). You can change bounds with the bounds argument in `fitContinuous`. Several bounds can be given at a time e.g. `bounds = list(sigsq = c(0, 0.1), alpha = c(0, 1))` would constrain both the σ^2 and α parameters.

For example, if an OU model keeps getting stuck you could try changing the lower bound on α:

```
OU <- fitContinuous(phy = mytree, dat = log(mydata[,"BodySize_kg"]), model = c("OU"),
                    bounds = list(alpha = c(0.01, 1)))
```

This example gives a warning message because `alpha` is over 1, so when we make the upper bound smaller the method ends up giving us this value instead because it's as close to 1 as we are allowing the model to go.

6.4 References

- Blomberg, S. P., T. Garland, and A. R. Ives. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. Evolution 57:717–745.
- Butler, M. A. and A. A. King. 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. The American Naturalist 164:683–695.
- Cavalli-Sforza, L. L. and A. W. Edwards. 1967. Phylogenetic analysis. models and estimation procedures. American Journal of Human Genetics 19:233.
- Cooper, N., R. P. Freckleton, and W. Jetz. 2011. Phylogenetic conservatism of environmental niches in mammals. Proceedings of the Royal Society B: Biological Sciences 278:2384–2391.
- Cooper, N. and A. Purvis. 2010. Body size evolution in mammals: complexity in tempo and mode.The American Naturalist 175:727–738.
- Cooper, N., Thomas, G.H., Venditti, C., Meade, A. & Freckleton, R.P. (2016b) A cautionary note on the use of ornstein-uhlenbeck models in macroevolutionary studies. Biological Journal of the Linnaean Society
- Felsenstein, J. 1973. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. Systematic Biology 22:240–249.
- Freckleton, R. P. and P. H. Harvey. 2006. Detecting non-brownian trait evolution in adaptive radiations. PLoS Biology 4:e373.
- Hansen, T. F. 1997. Stabilizing selection and the comparative analysis of adaptation. Evolution Pages 1341–1351.

- Harmon, L. J., J. B. Losos, T. Jonathan Davies, R. G. Gillespie, J. L. Gittleman, W. Bryan Jennings, K. H. Kozak, M. A. McPeek, F. Moreno-Roark, T. J. Near, et al. 2010. Early bursts of body size and shape evolution are rare in comparative data. Evolution 64:2385–2396.
- Lande, R. 1976. Natural selection and random genetic drift in phenotypic evolution. Evolution 30:314–334.
- Slater, G. J., L. J. Harmon, and M. E. Alfaro. 2012. Integrating fossils with molecular phylogenies improves inference of trait evolution. Evolution 66:3931–3944.
- Slater, G. J. and M.W. Pennell. 2014. Robust regression and posterior predictive simulation increase power to detect early bursts of trait evolution. Systematic Biology 63:293–308.

Chapter 7

Macroevolutionary models in R: Part 2 - discrete traits

The aims of this practical are to learn how to use R to fit macroevolutionary models in R to discrete traits.

We will be using the evolution of magical creature life-history variables as an example. The data includes body mass (average adult size at rest) in kg, social status (1 = solitary, 2 = social), habitat (1 = terrestrial, 2 = aquatic, 3 = volant) and magical power (in thaum - with thanks to Terry Pratchett for the units). These data are invented, so please don't get too upset if I've misclassified anything!

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `ape`
- `geiger`
- `OUwie`

This is Part 2 of the "Macroevolutionary models in R" practical, so you can skip through the set up if you're just carrying on from that.

This handout borrows heavily from a Linnaean Society workshop I ran with Graham Slater in 2014. Many thanks to Graham for his invaluable input.

7.1 Preparing for the analysis

7.1.1 Load packages, read in the data and the tree

This is the same as we did in the PGLS practical, so I won't give detailed instructions here.

```
# Load packages
library(ape)
library(geiger)
library(OUwie)
```

```
# Read in data
magicaldata <- read.csv("magicalcreatures.csv")
# Check data is loaded correctly
str(magicaldata)
```

```
## 'data.frame':    30 obs. of  5 variables:
##  $ Species     : Factor w/ 30 levels "Acromantula",..: 22 24 19 5 7 16 21 25 30 29 ...
##  $ BodySize_kg : num  1.5 50 0.5 100 8000 1 2.5 6 800 600 ...
##  $ SocialStatus: int  2 2 2 1 1 2 2 1 1 2 ...
##  $ Habitat     : int  1 1 2 1 3 1 1 3 3 3 ...
##  $ Magic_thaum : num  106.3 86.7 98.6 57 131.9 ...
```

```
# Read in tree
magicaltree <- read.nexus("magicaltree.nex")
# Check tree is loaded correctly
str(magicaltree)
```

```
## List of 4
##  $ edge       : int [1:52, 1:2] 28 29 30 31 31 30 32 33 33 34 ...
##  $ edge.length: num [1:52] 136.86 65.69 97 2.92 2.92 ...
##  $ Nnode      : int 26
##  $ tip.label  : chr [1:27] "Doxie" "Bowtruckle" "Hinkypuff" "Grindylow" ...
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

7.1.2 Modify the tree and data so they can be used in the analyses.

Again we did this in the PGLS practical. Please remind yourself of what these steps are needed for.

```
# Ensure tree is fully bifurcating
magicaltree <- multi2di(magicaltree)

# Replace spaces with underscores in the species names
magicaldata$Species <- gsub(" ", "_", magicaldata$Species)

# Add species names to row names
row.names(magicaldata) <- magicaldata$Species
```

For some weird reason the `geiger` function we need (`treedata` see below) won't work if you input a dataset with variables that are characters i.e. words or letters. Our taxonomic variable Species is a character so we need to exclude it from the data. Note for your own data you'd need to remove all character variables (or recode them as 0,12 etc.). We will do this by making a new dataset called magicaldata2.

```
magicaldata2 <- magicaldata[, 2:5]
```

Here the [ ] tells R we want to subset the dataset. R data frames are always described by [X,Y] where X is rows and Y is columns. So [1, 1] will select the entry in the first column and the first row of the data frame. [, 2:2] selects all rows but only columns 2 to 5. These are the columns containing our numeric variables.

We then need to match the species in tree to those in the dataset as in the PGLS practical. **Note that we are using `magicaldata2` here.**

```
match.species <- treedata(magicaltree, magicaldata2)

mytree <- match.species$phy
mydata <- match.species$data
```

7.2 Fitting models of evolution to discrete data (regime dependent evolution)

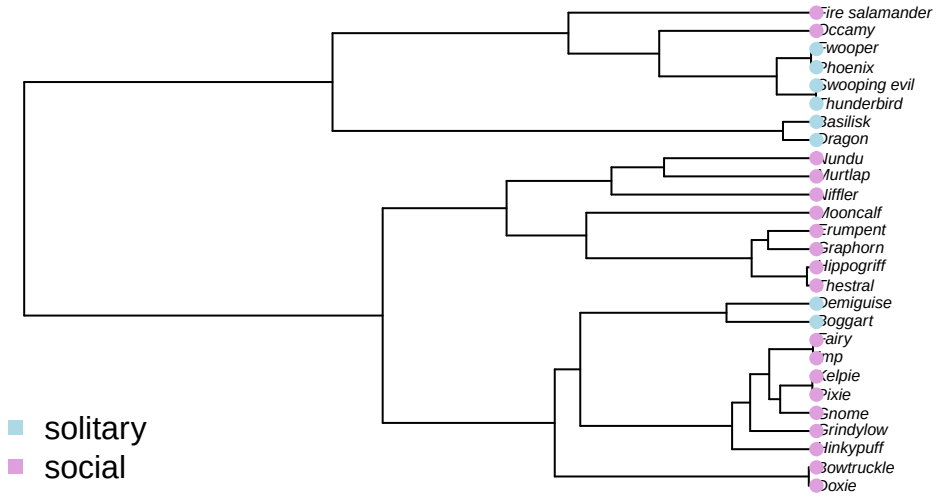
In the previous section we saw how to fit three models of trait evolution to continuous variables. Although the evolutionary modes seem quite different, all are similar in that the evolutionary process is constant over the entire clade, i.e., all branches are evolving at the same rate (in BM), drawn to the same trait value with the same strength (OU) or decline in rate at the same time-dependent pace (EB). Often, we want to relax this assumption. Members of our clade might belong to one of a set of ecological regimes, for example dietary niches or locomotor modes, and we might hypothesize that there are different evolutionary rates or different optimal trait values for each of these regimes. In this section, we'll look at how to fit these kinds of models using the `OUwie` package.

First we will look at how to reconstruct the evolutionary history of a discrete trait. Then we will use `OUwie` to allow rates and optimal trait values for a continuously valued trait, like body size, to vary, based on that discrete trait.

Here we will use the discrete variable `SocialStatus` from our magical creature dataset. Any species with a social group size of 1 is solitary (SocialStatus = 1), while any species with a group size greater than 1 is social (SocialStatus = 2).

We can visualize these variables on our tree by plotting them with colors. We'll use light blue for solitary and plum for social. Because our states are coded as "1" and "2", we can use a little trick to get the appropriate colors by indexing a vector of "lightblue" and "plum". We first need to make sure teh species in our data are in the same order as those in the tree.

```
# Reorder the data so it's the same order as in the tree
mydata <- mydata[match(mytree$tip.label,rownames(mydata)),]

# Create a vector of two colours
social.colors <- c("lightblue", "plum")

# Plot the tree, add coloured tip labels and a legend
plot(mytree, cex = 0.5)
tiplabels(pch = 16, col = social.colors[mydata[,"SocialStatus"]])
legend("bottomleft",legend=c("solitary", "social"),
       pch = 15, col=c("lightblue", "plum"), bty = "n")
```



You'll see that solitary behavior seems to be more restricted in its distribution. Most solitary magicals are bird or reptile like. From the distribution on the phylogeny, we might guess that solitary behavior is the ancestral state. This is exactly what we need to know in order to test whether evolutionary modes for body size vary for solitary vs social magical creatures. But before we can infer ancestral states, we need to chose the most appropriate model of social evolution.

There are several ways of mapping social status onto the tree but probably the most straightforward is to use an ancestral state estimation. We will estimate ancestral states for each node under a Markov model, pick the state with the highest marginal likelihood, and then assign that as the node state.

Unfortunately, just like rates can vary for continuous traits, so they can vary for discrete traits, and this can impact our ancestral state estimation. Fortunately, it's straightforward to test for this kind of heterogeneity using the `fitDiscrete` function in the `geiger` package.

7.3 Models of discrete trait evolution

7.3.1 Mk1 – all rates equal (ER)

The simplest Markov model we can fit to comparative data is an Mk1 model – M for Markov, k1 for $k = 1$ or 1 parameter. The single parameter of this model is a transition rate – the rate at which states change. Because we only have one rate, transitions between any pair of states occur at the same rate and are therefore equally probable. We can visualize the Q (rate) matrix for an Mk1 model like this:

-	1	2	3
1	-	1	1
2	1	-	1
3	1	1	-

The off-diagonals are the transition rates from state 1 to 2, 1 to 3, 2 to 1 and so on (read rows then columns). We typically designate individual transition rates in the form q_ij, which means the rate of going from state i to state j. Here, the 1 in all off-diagonal elements represents the fact that rates are the same regardless of what state i and j are, and the direction of that change. The diagonal elements q_ii give the rate of not changing and are computed as the negative sum of the non-diagonal row elements. This is so that the rows sum to zero.

If you're familiar with models of molecular evolution, you might know this model better as the **Jukes-Cantor model**, where transition rates = transversion rates.

7.3.2 Mk – symmetric rates (SYM)

We could add some complexity, and perhaps realism, by imagining that the rate of change between any pair of states is the same regardless of direction, but that the rate of change differs among states. Such a model is referred to as a symmetric model, and has a Q matrix of the form:

-	1	2	3
1	-	1	2
2	1	-	3
3	2	3	-

Here, $q_{12} = q_{21}$ but this rate is allowed to be different from q_{13}/q_{31} and from q_{23}/q_{32}. The number of different

rates is three for $S = 3$. However, if we were to add a fourth state, the new number of rates would not be 4 but rather would be 6. This is because there would now be 6 distinct off-diagonal elements present in the upper or lower diagonals.

If you're more familiar with molecular models, this is how we get the 6-rate GTR (General Time Reversible) model:

-	A	C	T	G
A	-	1	2	3
C	1	-	4	5
T	2	4	-	6
G	3	5	6	-

Obviously, a symmetric model with only 2 states becomes an equal rates (Mk1) model.

7.3.3 Mk – All Rates Different (ARD)

Finally, we can go crazy and allow all rates to be different. For S-states, this would generate $S^2 - S$ rates which might be crazy, depending on how many states you have. For completeness, the Q matrix for an All Rates Different model would look like this:

-	1	2	3
1	-	1	2
2	3	-	4
3	5	6	-

If you're more familiar with molecular models, then you'll be aware that molecular folks don't do this because of potential over-fitting. This is a very important point to consider. However, in comparative data, there are three situations in which this model might realistically be a good fit.

1. An All Rates Different model might be a good fit if you have an especially large datasets that spans a variety of different states. For example, if you had a tree of all 64,000 plus vertebrates and wanted to examine transition rates among different dietary strategies, this model would be worth examining.
2. If you have strong reasons to suppose character states are not reversible, this is worth using. For example, complex structures like eyes or teeth tend not to reappear once lost so asymmetric models might be a better fit for these kinds of characters.
3. This model is also a good option when you only have two states but think rates back and forth might be different. In this latter situation, the All Rates Different model simply gives you two rates, which isn't really over-fitting.

7.4 Fitting the models using `fitDiscrete`

Let's try these models with our magical creatures dataset using `fitDiscrete`. We will investigate rates of change in social status.

```
equal <- fitDiscrete(mytree, mydata[ , "SocialStatus"], model = "ER")
#sym <- fitDiscrete(mytree, mydata[ , "SocialStatus"], model = "SYM")
ard <- fitDiscrete(mytree, mydata[ , "SocialStatus"], model = "ARD")
```

Before moving on, note that the commented out fitting of the symmetric model is on purpose. Why?

Remember that we only have 2 states, solitary and social. A symmetric model with 2 states is just an equal rates model, so we can conveniently ignore it here. Let's look at the output for the equal rates model:

```
equal
```

```
## GEIGER-fitted comparative model of discrete data
##  fitted Q matrix:
##                  1            2
##     1 -0.002247321  0.002247321
##     2  0.002247321 -0.002247321
##
##  model summary:
##  log-likelihood = -8.815776
##  AIC = 19.631552
##  AICc = 19.791552
##  free parameters = 1
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

This looks very similar to the output from `fitContinuous`. We've got a model summary, with log likelihoods and AIC scores, convergence diagnostics and an object summary. The only difference is the first part, which gives us a fitted Q matrix, rather than a summary of model parameters (the Q matrix **is** the model parameters). This is an equal rates model, so the two off-diagonal elements are all same, and the diagonals are the negative values of the rates (so rows sum to zero).

By typing `ard`, we can look at the output for the all-rates-different model:

```
ard
```

```
## GEIGER-fitted comparative model of discrete data
##  fitted Q matrix:
##                  1            2
##     1 -0.005617678  0.005617678
##     2  0.002368874 -0.002368874
##
##  model summary:
##  log-likelihood = -8.634626
##  AIC = 21.269252
##  AICc = 21.769252
##  free parameters = 2
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 0.40
##
##  object summary:
##  'lik' -- likelihood function
```

```
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

It seems as though the rate of moving from solitary to social (0.005) is slightly higher than the rate of going from social to solitary (0.002). Based on our color-coordinated plot from earlier, we might have predicted this to be the case.

Because the output of equal and ard are just like fitContinuous, we can pull AICc values out and use them to perform model selection:

```
aic.discrete <- setNames(c(equal$opt$aic, ard$opt$aic), c("equal", "different"))
weights <- aicw(aic.discrete)
weights
```

```
##                fit  delta         w
## equal     19.63155 0.0000 0.6939922
## different 21.26925 1.6377 0.3060078
```

Based on AIC weights, which model should we prefer?

The all-rates-different model is less strongly supported (AICcW = 0.306) than the equal rates model (AICcW = 0.694). Now we can move forward with reconstructing ancestral states under the preferred equal rates transition rate model.

7.5 Ancestral state reconstructions

Ancestral state reconstruction is probably one of the most over-used and uninformative methods in the phylogenetic comparative methods toolkit. There are many reasons to be highly skeptical of ancestral state estimates and interpretations of macroevolutionary patterns and process that are based on them. However, if you want to know if evolutionary tempo or mode have varied over clade history based on the state of a discrete trait, you'll need to do it. We'll use ape's ace function here. There are other options out there, for example in the phytools package. But ace will work for our purposes.

To perform ancestral state estimation under the all-rates-different model:

```
asr <- ace(mydata[ , "SocialStatus"], mytree, type = "discrete", model = "ER")
```

You might see an error message appear saying NaNs produced.

Don't worry about this – it happens when rates for one transition are particularly low but doesn't really affect our node state estimates. One point to note here is that ace now defaults to a joint estimation procedure, where the ancestral states are optimized based on all information in the tree, not just the states at descendant tips.

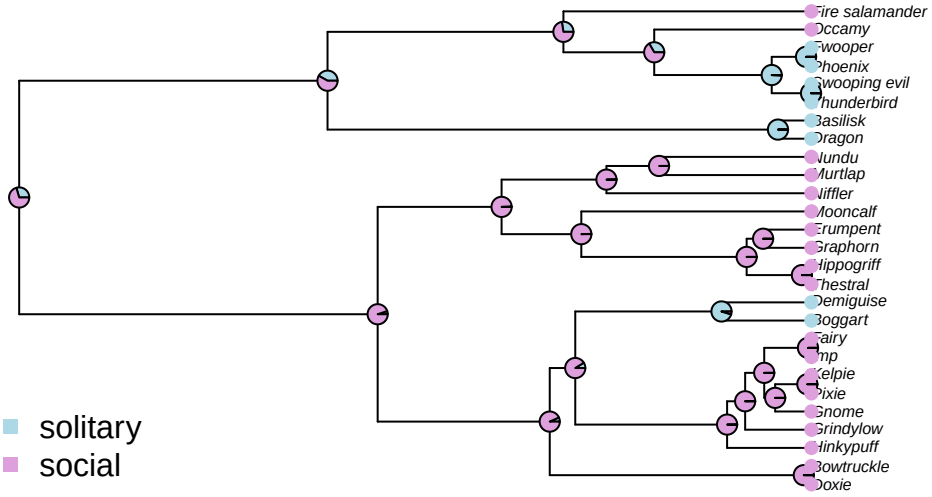
We can access our ancestral states by typing:

```
asr$lik.anc
```

In this matrix, the columns correspond to nodes in the tree (the numbering is off, as we'll see in a sec) and the two columns give the scaled likelihoods that the node is in states 1 or 2. The scaled likelihoods are like probabilities, so for the first node, we reconstruct state 1 with probability = 0.3 and state 2 with probability = 0.7 , but for node 2 the probability of state 1 = 0.05 while that of state 2 = 0.95. Scaled likelihoods lend themselves very well to graphical display, so we can visualize these states with pie charts on the tree we plotted earlier.

```
plot(mytree, cex = 0.5)
nodelabels(pie = asr$lik.anc, piecol = social.colors, cex = 0.5)
tiplabels(pch = 16, col = social.colors[mydata[,"SocialStatus"]])
```

```
legend("bottomleft",legend=c("solitary", "social"),
       pch = 15, col=c("lightblue", "plum"), bty = "n")
```



We reconstruct mostly social as the ancestral state for magical creatures, but with multiple transitions to solitary behaviour. For our next analyses though, we want to be able to extract the "best" state for each node. We can do this quite easily with the data structure `ace` gives us. First, we need to assign row names to our ancestral states that actual correspond to node numbers. phylo–format trees number nodes from `n+1` onwards, where `n` is the number of taxa in the tree. So if there are 10 taxa, the root node is 11. Recall also that a fully bifurcating tree has `n - 1` nodes. We can pull out the scaled likelihoods and number the rows appropriately with two simple lines of code:

```
node.states <- asr$lik.anc
rownames(node.states) <- seq(1:nrow(node.states)) + length(mytree$tip.label)
node.states
```

```
##                1            2
## 28 2.951598e-01 7.048402e-01
## 29 5.176500e-02 9.482350e-01
## 30 6.723783e-02 9.327622e-01
## 31 2.788813e-05 9.999721e-01
## 32 8.812542e-02 9.118746e-01
## 33 1.121960e-03 9.988780e-01
## 34 9.481369e-05 9.999052e-01
## 35 1.180908e-05 9.999882e-01
## 36 1.291077e-05 9.999871e-01
## 37 5.096106e-07 9.999995e-01
## 38 4.892579e-07 9.999995e-01
## 39 9.532305e-01 4.676946e-02
## 40 6.559008e-03 9.934410e-01
## 41 4.773548e-03 9.952265e-01
## 42 1.850043e-04 9.998150e-01
## 43 4.923981e-06 9.999951e-01
## 44 5.791449e-05 9.999421e-01
## 45 2.257465e-03 9.977425e-01
## 46 1.779187e-03 9.982208e-01
## 47 4.183958e-01 5.816042e-01
## 48 9.983242e-01 1.675751e-03
## 49 2.750360e-01 7.249640e-01
## 50 3.333531e-01 6.666469e-01
```

```
## 51 9.925342e-01 7.465840e-03
## 52 1.000000e+00 3.595570e-08
## 53 9.999929e-01 7.068412e-06
```

Now the rownumbers correspond to node numbers. This is useful. Now we'll use a simple trick to extract the most likely states and assign them as node values on our tree.

```
best <- apply(node.states, 1, which.max)
best
```

```
## 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
##  2  2  2  2  2  2  2  2  2  2  2  1  2  2  2  2  2  2  2  2  1  2  2  1  1
## 53
##  1
```

Now we have a named vector of "best" estimates of the node state. We can assign these to the tree using the following line of code.

```
mytree$node.label <- best
mytree
```

```
##
## Phylogenetic tree with 27 tips and 26 internal nodes.
##
## Tip labels:
##  Doxie, Bowtruckle, Hinkypuff, Grindylow, Gnome, Pixie, ...
## Node labels:
##  2, 2, 2, 2, 2, 2, ...
##
## Rooted; includes branch lengths.
```

Now we have node labels associated with our magical tree that specify which social regime each branch is evolving under. We're now ready to move on to modeling state specific rate variation and adaptive optima, using OUwie.

7.6 Fitting models using `OUwie`

`OUwie` (pronounced Ow-EE) is a package written by Brian O'Meara and Jeremy Beaulieu that performs maximum likelihood optimization of Brownian motion and Ornstein-Uhlenbeck models. The models implemented include the simple versions introduced earlier, but also include more complex versions that allow model parameters (rates, selection, optima) to vary among evolutionary regimes.

There are three things we need to fit an `OUwie` model. 1. A phylogeny with internal nodes labeled with the ancestral selective regimes - which we just did. 2. A dataset containing column entries in the following order: i) species names, ii) current selective regime, and iii) the continuous trait of interest. 3. The model we want to fit.

For the dataset all we need to do is make a dataframe with the relevant information. Let's make it and look at the first few lines:

```
ouwie.data <- data.frame(species = rownames(mydata), regime = mydata[ , "SocialStatus"],
trait = log(mydata[ , "BodySize_kg"]))
head(ouwie.data)
```

```
##                species regime      trait
## Doxie            Doxie      2  0.4054651
## Bowtruckle Bowtruckle      2 -1.0498221
```

```
## Hinkypuff   Hinkypuff      2  0.0000000
## Grindylow   Grindylow      2 -0.5978370
## Gnome           Gnome      2  3.2188758
## Pixie           Pixie      2 -0.9162907
```

Finally, we want to decide which model we're going to fit.

7.6.1 Multi-rate Brownian motion (BMS)

In `OUwie` a multi-rate BM model is coded as "BMS".

```
BMvariable <- OUwie(mytree, ouwie.data, model = "BMS")
```

```
## Initializing...
## Finished. Begin thorough search...
## Finished. Summarizing results.
```

Let's look at the results:

```
BMvariable
```

```
##
## Fit
##        lnL      AIC     AICc model ntax
##  -72.14716 152.2943 154.1125   BMS   27
##
## Rates
##                  1          2
## alpha           NA         NA
## sigma.sq 7.269257 0.09645345
##
## Optima
##                  1        2
## estimate 18.71153 2.902904
## se       65.09904 3.209815
##
## Arrived at a reliable solution
```

The final line tells us that we arrived at a reliable solution – that is that the optimizer converged on a reliable set of parameter estimates. The rest of the output includes log likelihood (`LnL`), AIC (`AIC`) etc, the rates (including the alpha parameters), and the optima. There are three main things to notice:

- You'll see here that there are `NAs` for alpha because this is a BM model, so there is no α parameter to estimate.
- The optima here correspond to the root state for species with state 1 (solitary) or state 2 (social). It makes sense that the root state is higher for solitary species, as we know these can be huge (basilisks, dragons etc.) For OU models, these optima would be the adaptive optimal values of body mass for each of the two states.
- The rates (`sigma.sq`) differ for regimes 1 and 2. Specifically, the rate for regime 1 (solitary) appears to be much higher that of regime 2 (social). So rates of body size evolution seem to be faster for solitary magical animals than for social magical animals, again this fits with our understanding of the data.

To know whether this difference is great enough for us to prefer this model, we'd need to compare AIC scores, or something similar. We'll come back to this in a minute.

7.6.2 Multi-peak OU models

If you look at the help file for `?OUwie` you'll see the following options are available to us.

1. single-rate Brownian motion (model=BM1) [equivalent to "BM" in `fitContinuous`]
2. Brownian motion with different rate parameters for each state on a tree (model=BMS)
3. Ornstein-Uhlenbeck model with a single optimum for all species (model=OU1) [equivalent to "OU" in `fitContinuous`]
4. Ornstein-Uhlenbeck model with different state means and a single α and σ^2 acting on all selective regimes (model=OUM)
5. Ornstein-Uhlenbeck models that assume different state means as well as either multiple σ^2 (model=OUMV), multiple α (model=OUMA), or multiple α and σ^2 for each selective regime (model=OUMVA).

We have quite a few options when it comes to OU models; we can allow the optima to vary, the rates, the alphas and any combination of these. I'd encourage you to play around with these options with your own data, but for now, we'll focus on the different optima models (OUM). Be aware too that these methods are very data hungry. I wouldn't recommend fitting an OUMVA model to a tree with 50 tips – you'd need closer to 200 and ideally more to get good fits for this complex model. Of course here we only have 26 species so I wouldn't trust these results (apart from the fact they are made up data about made up animals!).

```
OUmulti <- OUwie(mytree, ouwie.data, model = "OUM")
```

```
## Initializing...
## Finished. Begin thorough search...
## Finished. Summarizing results.
```

```
# Look at the results
OUmulti
```

```
##
## Fit
##        lnL      AIC     AICc model ntax
##  -69.75334 147.5067 149.3249   OUM   27
##
##
## Rates
##                   1         2
## alpha     1.169449  1.169449
## sigma.sq 24.617820 24.617820
##
## Optima
##                  1         2
## estimate 4.946723 2.5550174
## se       1.212267 0.7470149
##
## Arrived at a reliable solution
```

What is different here?

We now have parameter estimates for `alpha`, as well as `sigmasq`. We can use the optima to infer an optimal size for solitary magical creatures of `4.947` log(body mass) units and an optimal mass of `2.555` log(body mass) units for social magical creatures. `alpha` is greater than zero suggesting that body size of magical creatures is evolving towards these optima.

To find out which model best fits our data, we'll need to compute AIC weights again.

```
aic.scores <- setNames(c(BM$opt$aicc, OU$opt$aicc, EB$opt$aicc, BMvariable$AICc, OUmulti$AICc),
                       c("BM", "OU", "EB", "BMvariable", "OUmulti"))
aicw(aic.scores)
```

```
##                 fit       delta            w
## BM         201.5837 52.25885489 2.160766e-12
## OU         149.3528  0.02789568 4.746969e-01
## EB         204.1283 54.80340245 6.054332e-13
## BMvariable 154.1125  4.78764935 4.393888e-02
## OUmulti    149.3249  0.00000000 4.813643e-01
```

Which is the best model overall?

7.7 References

- Blomberg, S. P., T. Garland, and A. R. Ives. 2003. Testing for phylogenetic signal in comparative data: behavioral traits are more labile. Evolution 57:717–745.
- Butler, M. A. and A. A. King. 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. The American Naturalist 164:683–695.
- Cavalli-Sforza, L. L. and A. W. Edwards. 1967. Phylogenetic analysis. models and estimation procedures. American Journal of Human Genetics 19:233.
- Cooper, N., R. P. Freckleton, and W. Jetz. 2011. Phylogenetic conservatism of environmental niches in mammals. Proceedings of the Royal Society B: Biological Sciences 278:2384–2391.
- Cooper, N. and A. Purvis. 2010. Body size evolution in mammals: complexity in tempo and mode.The American Naturalist 175:727–738.
- Cooper, N., Thomas, G.H., Venditti, C., Meade, A. & Freckleton, R.P. (2016b) A cautionary note on the use of ornstein-uhlenbeck models in macroevolutionary studies. Biological Journal of the Linnaean Society
- Felsenstein, J. 1973. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. Systematic Biology 22:240–249.
- Freckleton, R. P. and P. H. Harvey. 2006. Detecting non-brownian trait evolution in adaptive radiations. PLoS Biology 4:e373.
- Hansen, T. F. 1997. Stabilizing selection and the comparative analysis of adaptation. Evolution Pages 1341–1351.
- Harmon, L. J., J. B. Losos, T. Jonathan Davies, R. G. Gillespie, J. L. Gittleman, W. Bryan Jennings, K. H. Kozak, M. A. McPeek, F. Moreno-Roark, T. J. Near, et al. 2010. Early bursts of body size and shape evolution are rare in comparative data. Evolution 64:2385–2396.
- Lande, R. 1976. Natural selection and random genetic drift in phenotypic evolution. Evolution 30:314–334.
- Slater, G. J., L. J. Harmon, and M. E. Alfaro. 2012. Integrating fossils with molecular phylogenies improves inference of trait evolution. Evolution 66:3931–3944.
- Slater, G. J. and M.W. Pennell. 2014. Robust regression and posterior predictive simulation increase power to detect early bursts of trait evolution. Systematic Biology 63:293–308.

Chapter 8

Geometric Morphometrics in R

The aims of this practical are to learn how to use R to perform simple geometric morphometrics analyses.

We will use a data set of ventral skull views for eight species of toothed whales (Odontoceti) taken by a previous Masters student (Dan Bell). Download the eight photographs, and place them into your working directory.

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `geomorph`

8.1 A quick introduction to geometric morphometrics

8.1.1 What is morphometrics?

Morphometrics is the study of shape and size and their relationships with other variables. Shape is generally defined as **the property of an object invariant under scaling, rotation, or translation**.

To compare shapes, we need to define which bits of the shape to compare, for example if comparing the shapes of two cups, we might compare the width of their handles, or their diameters. In biological objects, structures that are recognizable and comparable among specimens are said to be **homologous**. Homologous points include things like the points that two bones join together. We need homologous points to compare the shapes of biological specimens. In morphometrics these points are referred to as **landmarks**.

8.1.2 Landmarks

There are three types of landmarks (defined by Bookstein 1991) **Type I** landmarks are truly homologous features that can be defined by a single point, for example where bone plates join, or small knobs on the bone. **Type II** landmarks include things like maximum of curvature of a feature, such as the most extreme part of the curve of the skull. **Type III** landmarks can also be referred to as **pseudolandmarks** or **semi-landmarks**. These are constructed geometrically, rather than being identifiable as unique points on the

structure. For example, the centre of a series of points, or the intersection of a line joining up several landmarks.

Type I landmarks are generally favoured as they are easier to put in the right place and give more information about the development of the feature. However, often we also use some Type II and Type III landmarks. Choosing landmarks carefully is really important - they need to be informative for the question you are trying to answer and capture the variation in shape that you are interested in. They should also be easy to identify and repeat to reduce measurement error.

8.1.3 Collecting data and adding landmarks

You can collect morphometric data in lots of ways - sometimes using measurements with calipers or rulers, but more often these days by taking digital photographs or 3D scans of specimens. Traditional morphometrics uses measurements, whereas **geometric morphometrics** uses geometric coordinates (x and y for 2D; x, y and z for 3D) instead. Geometric morphometrics have become a really popular way to investigate morphology (size and shape), and are a particularly useful tool when using museum specimen data. So we'll be using geometric morphometric tools in this practical.

Once you have your data (photos or scans), you need to add landmarks to them - the relationships between the landmarks will then be used in your shape analyses. You can do this in lots of different programs including R (see practical example below).

8.1.4 Measurement error

As with all analyses there are lots of sources of error in the collection of morphometric data. Think about what some of these might be, and see if you can think of ways to test whether they are a problem with your data. See Claude *pages 63-65* and Fruciano 2016 for some ideas. We will not cover this today but it is very important to consider if you use these methods in your projects.

8.2 A practical example using R

For this practical we are going to use the package `geomorph`. You will need to **install it in R** for this example to work. Load the package using `library`.

```
library(geomorph)
```

You will also need to **set the working directory** before you start.

8.3 Gathering the data

We are going to use a data set of ventral skull views for eight species of toothed whales (Odontoceti) taken by a previous Masters student (Dan Bell). Download the eight photographs, and place them into your working directory.

We will need to make a list of the files so we can tell `geomorph` what we want to work with. For this we can use the helpful R function `list.files`. This is better than manually typing a list as we avoid typos.

```
myphotos <- list.files(pattern = ".jpg")
# This tells R to look in your working directory for files with .jpg in their names
```

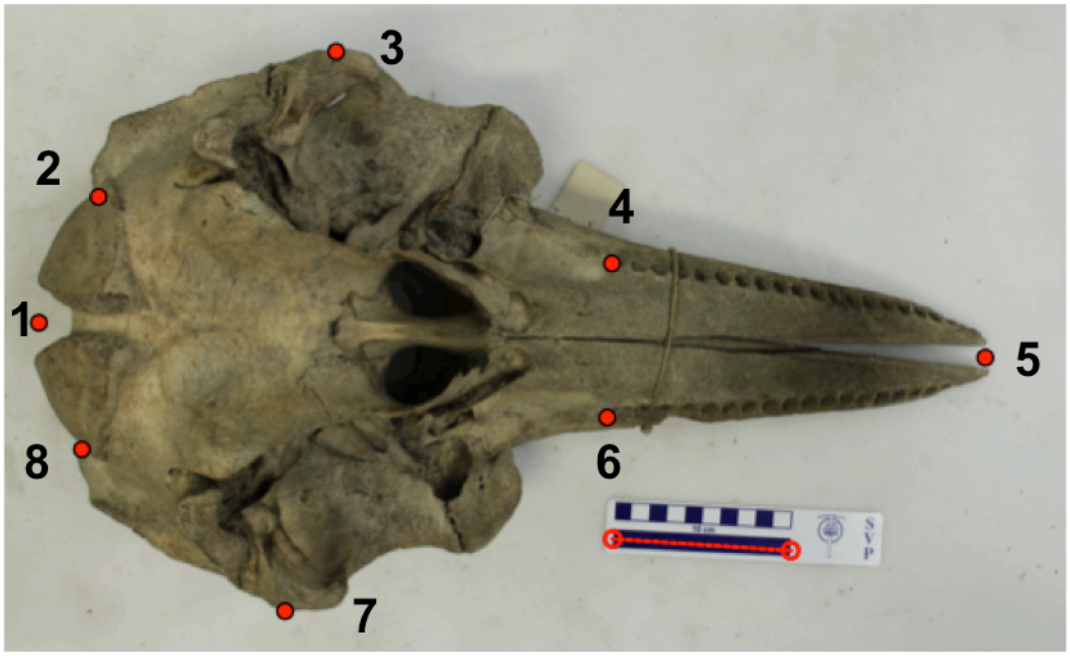


Figure 8.1: Whale landmarks example

```
# Look at the output
myphotos
```

This is just so you can see what `list.files` does. We will do this within the digitising function below rather than using `myphotos`.

8.4 Digitising (adding landmarks and curves)

You'll hear people use the word **digitise** to mean a bunch of different things (especially within museums!). In geometric morphometrics we mean the process of going from photos/scans to having coordinate data we can play around with, i.e. adding landmarks to our photos/scans. We will use the following landmarks for this practical:

1. Midway point between the occipital condyles.
2. Suture where lacrimojugal and basoccipital bones join on left hand side (indentation).
3. Most extreme point of the zygomatic arch on left hand side.
4. Anterior edge of on rearmost tooth on left hand side.
5. Tip of the rostrum.
6. Same as 4 but right hand side.
7. Same as 3 but right hand side.
8. Same as 2 but right hand side.

Digitizing is pretty simple to do in `geomorph`, though some people prefer to use package ImageJ or tpsdig which are more user friendly. Either way, you use the mouse to click on the photos at the point you want each landmark to be. Curves/semi-landmarks are added after the initial digitisation step. Today we will use `geomorph` and we won't add any curves.

Remember you need to add landmarks in the **same order** for each specimen, and don't forget to add the **scale** before you start. This step can be a time consuming process in a real analysis.

You will need to use the normal R Console, NOT RStudio for this stage to work

We are going to use the function `digitize2d` to add eight landmarks (`nlandmarks = 8`) to our list of photos (`list.files(pattern = ".jpg")`) and output these into a TPS (thin plate spline) file called `whale_landmarks.tps`. The scale bars in each photo are 10cm long so we add `scale = 10`.

```
digitize2d(filelist = list.files(pattern = ".jpg"),
           nlandmarks = 8, tpsfile = "whale_landmarks.tps", scale = 10)
```

Note that if you stop this procedure and then start again you may get the following error message: `Filelist does not contain the same specimens as TPS file.` To fix this you just need to delete `whale_landmarks.tps` from your working directory and then start again.

If you don't get an error you should find that a picture of the first specimen appears in your plotting window, and the message:

```
Only 1 scale measure provided. Will use scale = 10  for all specimens. Digitizing specimen
1 in filelist  Set scale = 10
```

To digitize proceed as follows. You will need to repeat this for *each* photo.

1. Set the scale by clicking on the start and end of the black scale bar.
2. The console will then display `Keep scale (y/n)?`. If you're happy press `y`, if not press `n` and try again. The function will check you are happy after every landmark you place. This gets a bit tedious but means you don't have to start over if you click somewhere by accident.
3. Select landmark 1, click `y` when happy, then select landmark 2 and so on…
4. Once you've finished one specimen, you'll be asked if you want to continue to the next specimen. Again you need to press `y` to continue.
5. Continue until all eight are digitised

Each specimen should end up looking something like this once the scale and landmarks have been added.

Don't get too worried about digitising accurately today. This is just an example so you get a chance to try this out. You'll notice that some landmarks are a lot harder than others to place accurately each time, bringing us back to thinking about potential sources of error.

If you're having trouble with this step you can use the TPS file I made earlier.

8.5 Reading in your data and plotting it

We can now read in our TPS file to look at our landmarks using `readland.tps`. I've asked it to add the specimen identities from the file names so we know which species is which by using `specID = "ID"`.

```
landmarks <- readland.tps("whale_landmarks.tps", specID = "ID")
```

```
## [1] "Specimen names extracted from line ID="
```

We can look at all the landmarks by typing:

```
landmarks
```

To save printing them all out I'll just look at the landmarks for the first specimen (*Pseudorca crassidens*).

```
landmarks[, , 1]
```

```
##           [,1]      [,2]
## [1,]  4.917176 31.136595
## [2,] 16.863048 49.815389
## [3,] 27.883473 60.925151
## [4,] 58.264038 42.370924
```

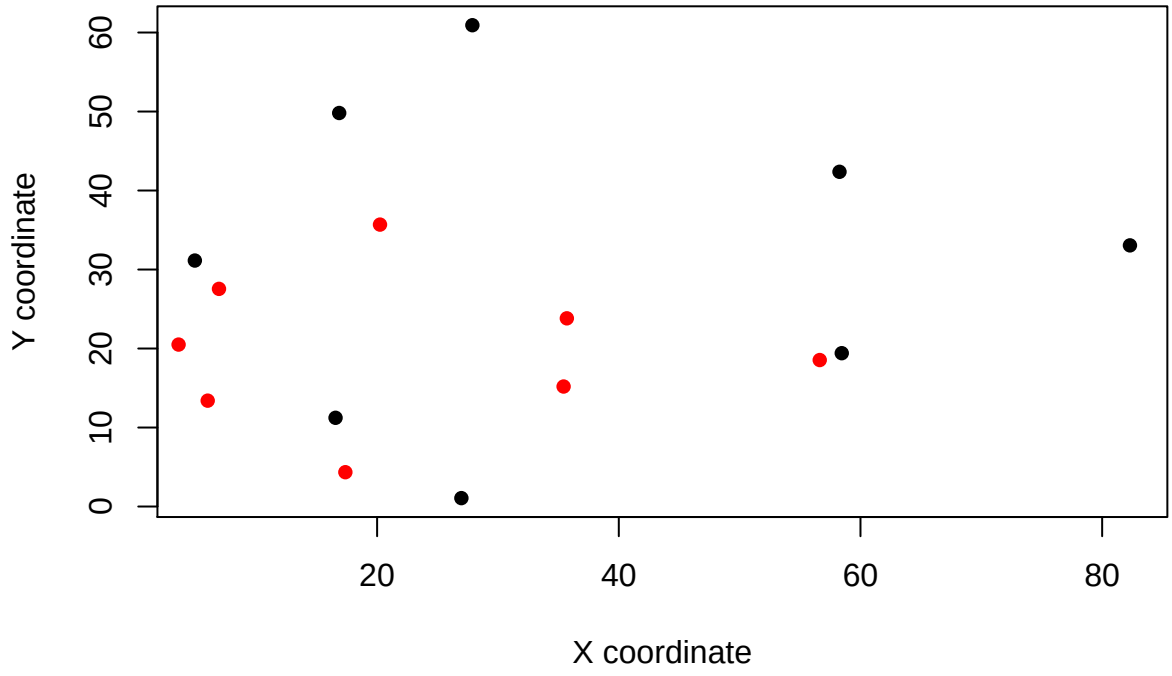
Figure 8.2: What your photos should look like post digitising

```
## [5,] 82.299223 33.057950
## [6,] 58.449630 19.410289
## [7,] 26.973126  1.064303
## [8,] 16.559808 11.230374
```

These are the Y and X coordinates of each point *after scaling*. They are scaled in the units of your scale bar (in the case mm). If you look at the TPS file itself in a text editor you'll see the numbers are different and there is information on the length of the scale bars.

We can plot the coordinates for the *Pseudorca* (specimen 1) and the *Tursiops truncatus* (specimen 8) specimens as follows.

```
plot(landmarks[, , 1], pch = 16, xlab = "X coordinate", ylab = "Y coordinate")
points(landmarks[, , 8], col = "red", pch = 16)
```

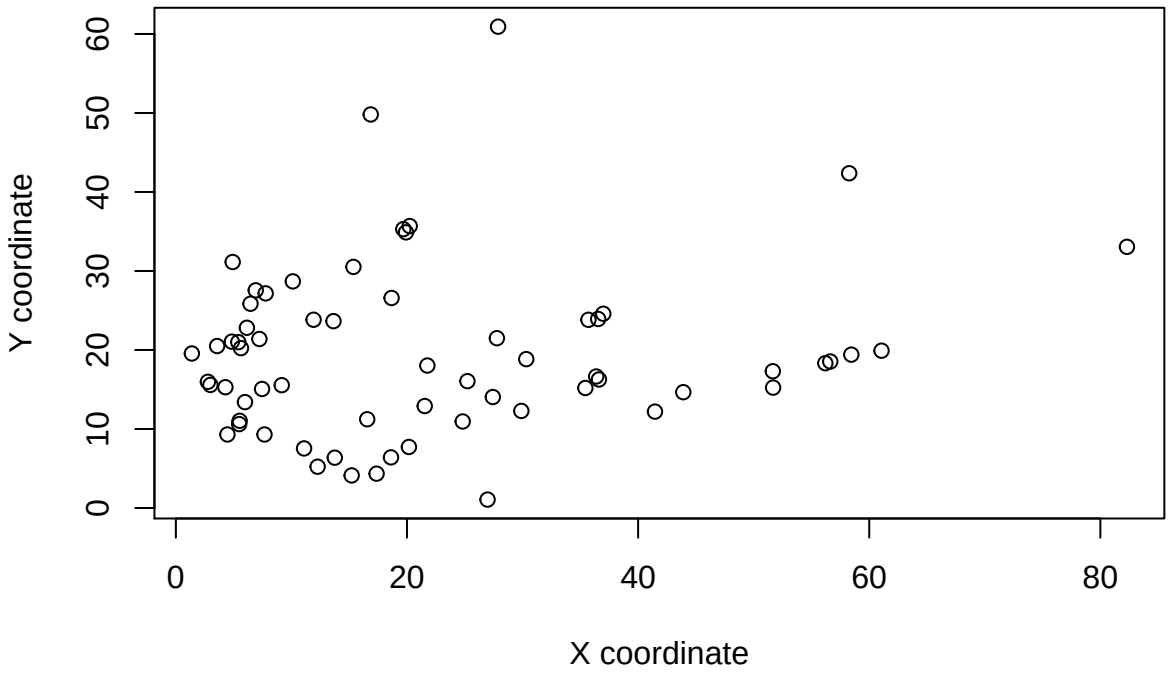


You can see that although both filled the screen when you were digitising, *Tursiops* (in red) is much smaller. This is why this scaling step is so important.

8.6 Generalised Procrustes Superimposition (GPA)

If you plot all the landmarks at the same time you'll notice it's a bit of a mess.

```
plot(landmarks[, 2, ] ~ landmarks[, 1, ], xlab = "X coordinate", ylab = "Y coordinate")
```



The skulls vary in their placement in each photo - in some they cover the whole area, in others they are further forward or sideways, some are at a slight angle (rotation). Also as we already noted, there is a difference in the *size* of the skulls. Remember that we are interested in comparing shapes and shape is defined as **the property of an object invariant under scaling, rotation, or translation**. So if we want to compare shapes, we need to remove these differences.

Luckily a number of solutions exist. We are going to use Generalised Procrustes Superimposition/Analysis (GPA). GPA is a way to remove rotation, translation and scaling differences among specimens so they can be compared. GPA translates all specimens to the origin (0,0,0), scales them to unit-centroid size, and optimally rotates them (using a least-squares criterion) until the coordinates of corresponding points align as closely as possible. The resulting aligned Procrustes coordinates represent the shape of each specimen.

This is hard to explain in words, but there are a number of excellent graphical explanations in the references below.

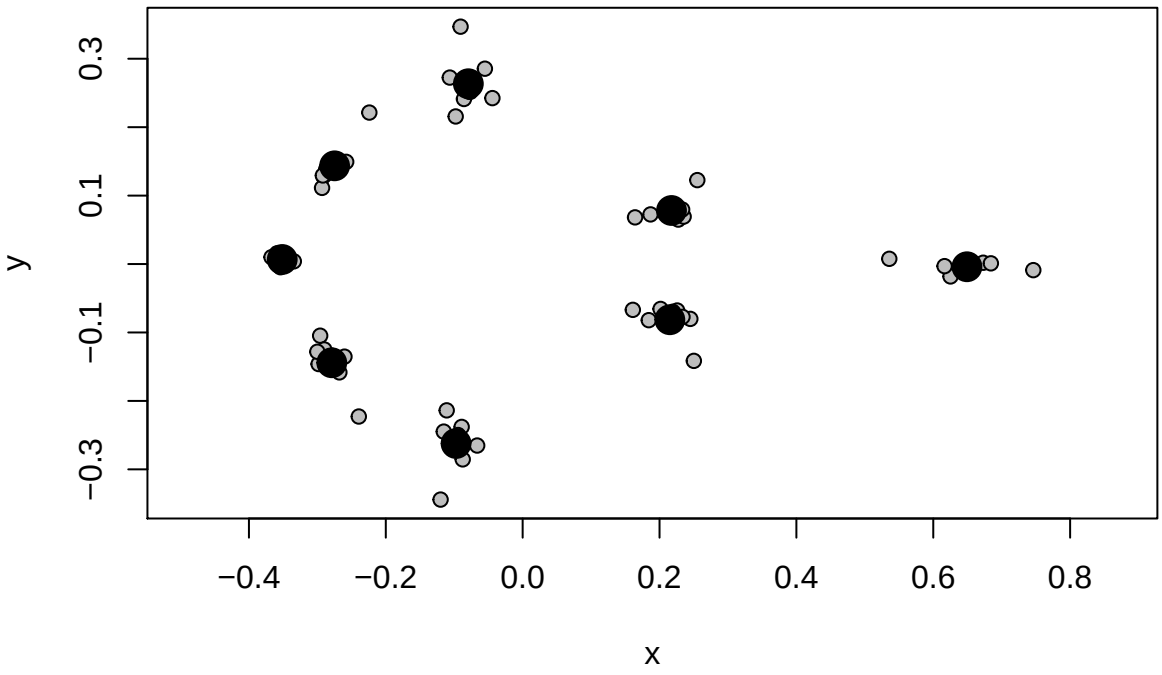
Remember that you *never* do a morphometrics analysis on the raw landmarks, they must always be aligned first or the results are meaningless. Also remember that you need to align **all** the specimens you want to use in your analysis at the same time, or again your analysis will be meaningless.

To do this in R we just need one line of code and the function `gpagen`.

```
gpa.landmarks <- gpagen(landmarks)
```

```
##
  |
  |                                                                      |   0%
  |
  |=============                                                         |  20%
  |
  |==========================                                            |  40%
  |
  |======================================================================| 100%
```

```
# Plot the Procrusted coordinates
plot(gpa.landmarks)
```



Note that now we have the average for each landmark as a large black point (the **centroid** - centroid just means the centre point of a sample of points/shapes), and the aligned landmarks for each specimen scattered around these in grey. You can see some landmarks are very variable, while others are more constant across our specimens.

`gpagen` not only outputs a nice plot of the specimens and their mean shape, but also the superimposed coordinates (`$coords`), the shape variables as principal components (`$pc.scores) and the centroid size of the specimens` (Csize‘).

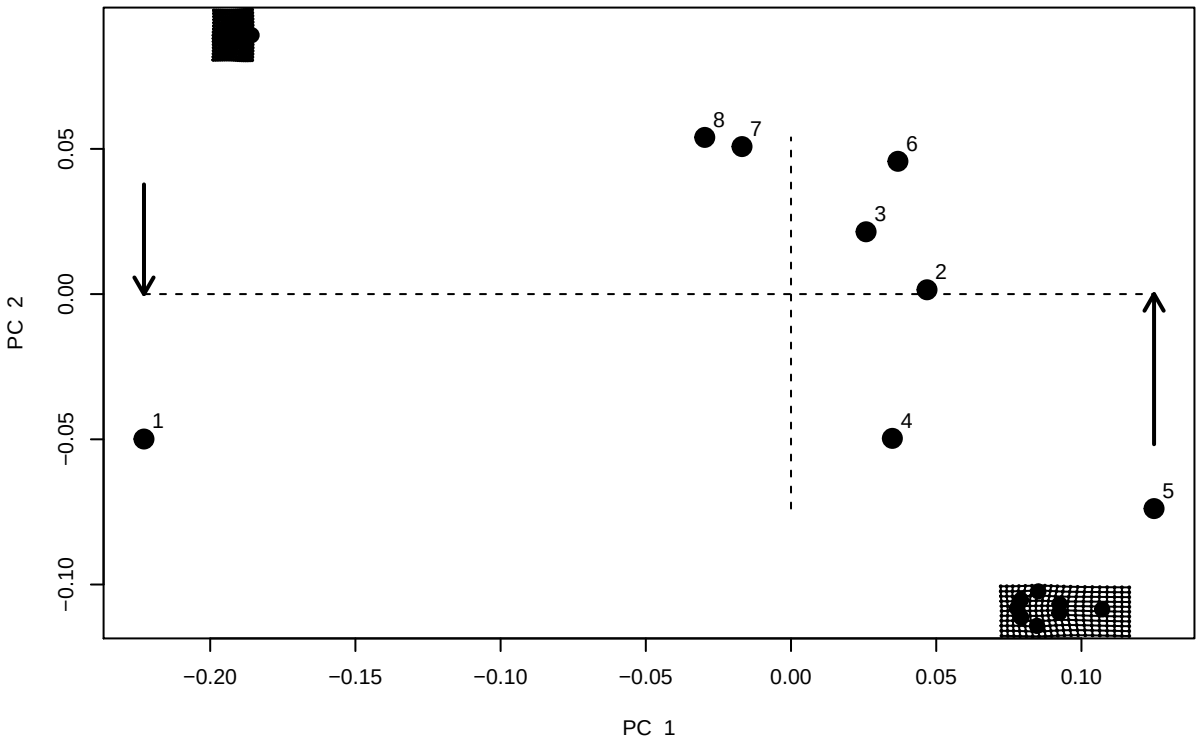
8.7 Principal components and plotting

We now have superimposed coordinates we can use to further investigate shape variation among our specimens. But there's a bit of a problem. If you think back to the assumptions we make when we do statistics, we often talk about how the variables should not be correlated. Here our points are highly correlated, e.g. the tip of the rostrum can only be in front of the back of the tooth row, so where the tooth row is in each specimen will be closely related to where the rostrum tip is. It can also be a bit hard to interpret analyses with all the landmarks included - how do we interpret a result that suggests a small increase in rostrum tip and a decrease in occipital condyle? To solve these issues we can use principal components analysis (PCA).

PCA finds the axes of greatest variation in a dataset and removes correlations among variables. It does this while still preserving the distances between data points - i.e. it doesn't distort variation in the data. The outputs of PCA are principal components scores, which we can think of as "shape variables". These PC scores tend to be used in any further analysis and are **independent components of shape variation**. Note that PCA is essentially just a rotation of the data, nothing more complicated than that. Again I recommend checking out the graphical examples in the texts below to help with your understanding.

Let's extract principal components scores for our cetacean dataset using `plotTangentSpace`.

```
pca.landmarks <- plotTangentSpace(gpa.landmarks$coords, label = TRUE)
```

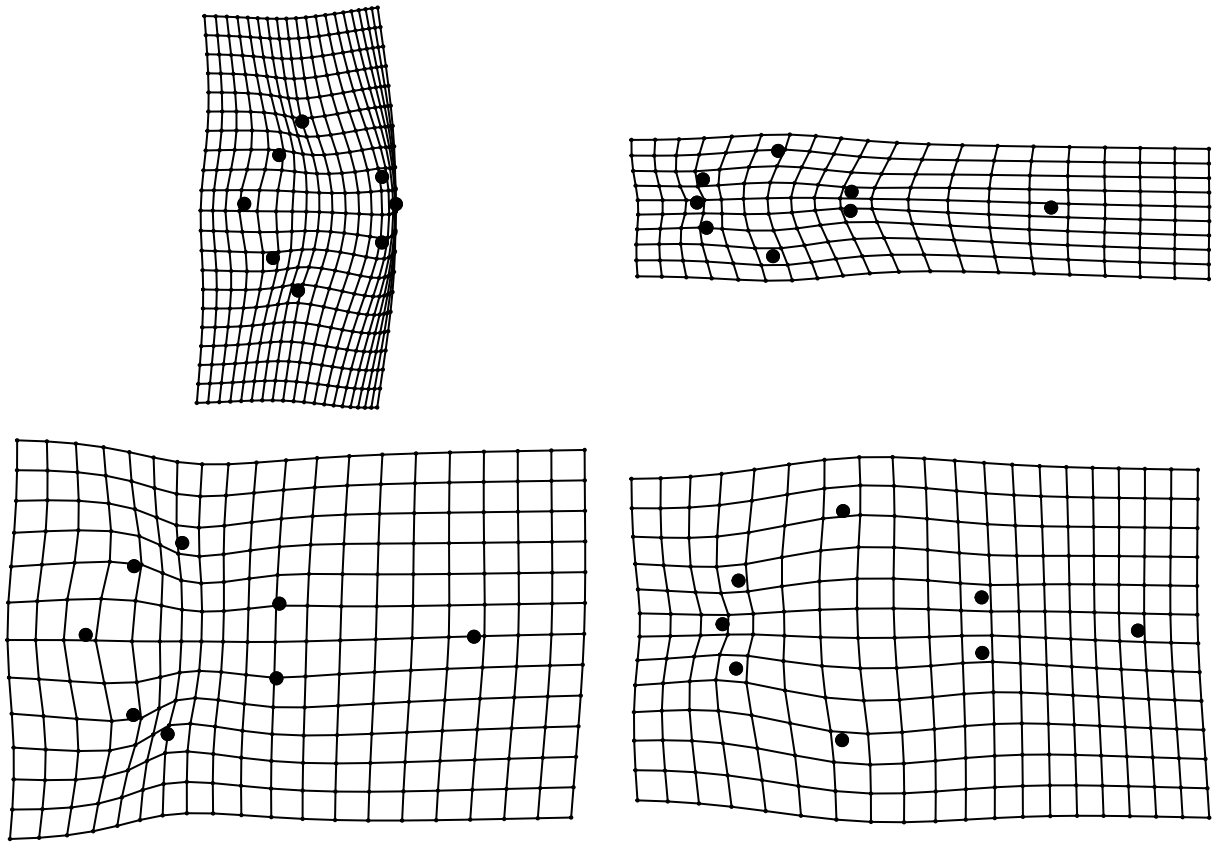


This produces a plot of the first two principal components (PC1 and PC2), with dotted lines at the origin and each specimen represented by a point. Point 1 is *Pseudorca*. Note that what we are now looking at is called a "shape space" or "morphospace". **Each point represents a shape** not an individual landmark.

To help with interpretation, two grids are also plotted. These represent the shape at the points indicated by the arrows. These are called thin plate splines, because we imagine the landmarks for the average/centroid shape were engraved on a thin plate of metal, and then deformed to get to the shapes at the indicated points. The grid to the left shows lots of deformation, with a widening of the back of the skull, and a shortening of the rostrum. The grid to the right shows deformation with the tooth rows being closer together and some stretching at the front to enlarge the rostrum. Specimens with high PC1 scores look more like the right hand grid, specimens with low PC1 scores look more like the left hand grid.

We can look at these grids individually, and for the other PC axes. The code below will show us the grids for the min and max PC1 and PC2. Note that I have included `mag = 2` which magnifies the differences two fold to make them easier to see.

```
# Make plotting window into 2 x 2 grid
par(mfrow = c(2, 2))
# Reduce margins around the plots
par(mar = c(0, 0, 0, 0))

# Select reference shape - the centroid of our Procrustes aligned coordinates
ref <- mshape(gpa.landmarks$coords)

# Plot each min/max PC scores in comparison to the reference shape (ref)
# with two fold magnification (mag = 2)
plotRefToTarget(ref, pca.landmarks$pc.shapes$PC1min, mag = 2)
plotRefToTarget(ref, pca.landmarks$pc.shapes$PC1max, mag = 2)
plotRefToTarget(ref, pca.landmarks$pc.shapes$PC2min, mag = 2)
plotRefToTarget(ref, pca.landmarks$pc.shapes$PC2max, mag = 2)
```



```
# Put graphics parameters back to default settings
par(mfrow = c(2, 2))
par(mar = c(5, 4, 4, 2))
```

Another way of interpreting PC scores is to extract the **loadings** for each PC using `$rotation`.

```
pca.landmarks$rotation
```

```
##               PC1         PC2          PC3         PC4         PC5
##  [1,]  0.07782421  0.01145886  0.010978946 -0.12118065  0.32956627
##  [2,] -0.01685081 -0.02976881  0.237243510  0.15913484 -0.24917185
##  [3,] -0.15165444 -0.35228327  0.131087366 -0.10282307 -0.11148058
##  [4,] -0.26942003 -0.35780565  0.109069813  0.24333031  0.05325473
##  [5,]  0.02198347  0.09012823 -0.784843674  0.20706328 -0.32689443
##  [6,] -0.38619090  0.02990721 -0.053598578 -0.37241810 -0.21857694
##  [7,] -0.23106590  0.32574319  0.026774344  0.21696513  0.37957940
##  [8,] -0.16544042 -0.11557074 -0.153305579 -0.15718351  0.22318585
##  [9,]  0.57879930 -0.30554975  0.024395464 -0.13850038  0.01307704
## [10,] -0.03727420 -0.06838202 -0.212667343  0.08987931 -0.04242125
## [11,] -0.23534726  0.36264877  0.184749068  0.19886767  0.17186361
## [12,]  0.23024040  0.16755749 -0.072168034  0.17366534 -0.03227954
## [13,]  0.02898251  0.23494160  0.398172339 -0.04393459 -0.61107235
## [14,]  0.37752351 -0.03750461  0.165149456  0.42261250  0.12908818
## [15,] -0.08952190 -0.36708762  0.008686147 -0.21645738  0.15536103
## [16,]  0.26741245  0.41156713 -0.019723245 -0.55902070  0.13692083
##                PC6         PC7         PC8
##  [1,]  0.732980006 -0.08204786 -0.14042273
##  [2,] -0.034715583  0.36122405 -0.05381482
```

```
##  [3,]  0.007281487  0.43550615  0.06763819
##  [4,]  0.158610967 -0.27638033  0.52594604
##  [5,] -0.041646343 -0.07954720  0.05088173
##  [6,]  0.149383964 -0.19916231 -0.56878480
##  [7,] -0.128169667  0.36715675 -0.20297068
##  [8,] -0.023489409 -0.04840285  0.04503906
##  [9,]  0.054086438  0.03069426 -0.12319514
## [10,]  0.037088383  0.41908435 -0.10530240
## [11,] -0.117978102 -0.24845750  0.03602596
## [12,]  0.071908482 -0.21206921 -0.03395509
## [13,]  0.045527030 -0.13851486  0.01994870
## [14,] -0.161022728 -0.14265102 -0.34854652
## [15,] -0.552080849 -0.28478974 -0.20316407
## [16,] -0.197764076  0.09835732  0.36740417
```

This shows how each of our landmarks contributes to each PC. Larger numbers mean the landmark has more influence on the PC, negative numbers show a negative effect. These can be hard to interpret but worth looking at. Remember each landmark has an X and a Y coordinate, so `[1,]` is the X coordinate of the occipital condyle landmark, and `[2,]` is the Y coordinate.

Here for example I'd suggest that PC1 is most heavily influenced by `[9,]`, the X coordinate of landmark 5, the tip of the rostrum. It's a positive number so it means species with high values for PC1 have an elongated rostrum. See if you can interpret these results further. It is often best to look at the loadings and the plots above to help with this.

Another important output to look at is the summary of the PC axes:

```
summary(pca.landmarks)
```

```
##
## PC Summary
##
## Importance of components:
##                             PC1     PC2     PC3     PC4     PC5      PC6
## Standard deviation     0.1013 0.05135 0.02514 0.01454 0.01177 0.006117
## Proportion of Variance 0.7361 0.18906 0.04531 0.01517 0.00994 0.002680
## Cumulative Proportion  0.7361 0.92521 0.97052 0.98568 0.99562 0.998300
##                               PC7       PC8
## Standard deviation     0.004865 2.022e-17
## Proportion of Variance 0.001700 0.000e+00
## Cumulative Proportion  1.000000 1.000e+00
```

This shows the variance on each PC axis (eigenvalues). Note that the first PC has the highest proportion of the variance (73.61%). PCs will always decrease in the amount of variance explained because of the way PCA works - it takes the axis that explains most variation first. Often people will only use PCs in their later analyses that sum up to 95% or 99% of the cumulative variance, because the later PCs are not really explaining much of the variation in shapes. In this example we'd probably use PC1, PC2 and PC3.

Finally, to extract the PC scores for each specimen we use `$pc.scores`. We'll use these for all further analysis.

```
pca.landmarks$pc.scores
```

```
##                                            PC1         PC2
## Pseudorca_crassidens_1961.6.14.3.jpg -0.22279912 -0.04990209
## Sousa_plumbea_70.1506.jpg             0.04682774  0.00149044
## Stenella_attenuata_1960.6.24.1.jpg    0.02583791  0.02145984
## Stenella_coeruleoalba_1938.2.5.1.jpg  0.03489668 -0.04967601
```

```
## Stenella_longirostris_1965.8.25.2.jpg  0.12500666 -0.07385256
## Steno_bredanensis_345c.jpg              0.03680834  0.04572532
## Tursiops_aduncus_1903.9.12.1.jpg       -0.01687868  0.05079845
## Tursiops_truncatus_1920.8.14.1.jpg     -0.02969952  0.05395661
##                                                PC3          PC4
## Pseudorca_crassidens_1961.6.14.3.jpg  -0.0006377733 -0.001028733
## Sousa_plumbea_70.1506.jpg             -0.0417508568  0.009805947
## Stenella_attenuata_1960.6.24.1.jpg     0.0169424424  0.030573421
## Stenella_coeruleoalba_1938.2.5.1.jpg   0.0008958266  0.001192135
## Stenella_longirostris_1965.8.25.2.jpg  0.0118958174 -0.010838289
## Steno_bredanensis_345c.jpg            -0.0043676504 -0.014536311
## Tursiops_aduncus_1903.9.12.1.jpg       0.0408120527 -0.006280211
## Tursiops_truncatus_1920.8.14.1.jpg    -0.0237898587 -0.008887959
##                                                PC5           PC6
## Pseudorca_crassidens_1961.6.14.3.jpg   0.006797670 -7.155469e-05
## Sousa_plumbea_70.1506.jpg              0.004444909 -8.639956e-03
## Stenella_attenuata_1960.6.24.1.jpg     0.004876528  5.166614e-03
## Stenella_coeruleoalba_1938.2.5.1.jpg  -0.023181651 -2.208889e-03
## Stenella_longirostris_1965.8.25.2.jpg  0.009562895  4.327237e-03
## Steno_bredanensis_345c.jpg             0.011629308 -1.185434e-03
## Tursiops_aduncus_1903.9.12.1.jpg      -0.004195661 -6.823983e-03
## Tursiops_truncatus_1920.8.14.1.jpg    -0.009933998  9.435965e-03
##                                                PC7           PC8
## Pseudorca_crassidens_1961.6.14.3.jpg   0.0003870485 0.000000e+00
## Sousa_plumbea_70.1506.jpg             -0.0036414119 2.081668e-17
## Stenella_attenuata_1960.6.24.1.jpg     0.0017329304 2.059984e-17
## Stenella_coeruleoalba_1938.2.5.1.jpg   0.0049957672 1.387779e-17
## Stenella_longirostris_1965.8.25.2.jpg -0.0037522008 2.992398e-17
## Steno_bredanensis_345c.jpg             0.0086445940 3.122502e-17
## Tursiops_aduncus_1903.9.12.1.jpg      -0.0046789921 1.908196e-17
## Tursiops_truncatus_1920.8.14.1.jpg    -0.0036877353 2.341877e-17
```

8.8 Statistical analyses of geometric morphometric datasets

This is the point at which geometric morphometrics gets really exciting - it's the point that you get to answer whatever question you started out with! It's also the point at which the number of different options becomes very large. Two common analyses are regressions and multivariate ANOVA (MANOVA). Regressions are used when you have continuous explanatory variables, for example if you want to see if body size is correlated with shape. MANOVA (or ANOVA) is used when you have categorical explanatory variables, for example if you want to see if habitat type is correlated with shape.

At this stage of the analysis we often add a new dataset containing our variables of interest. For simplicity we will just invent some data as follows.

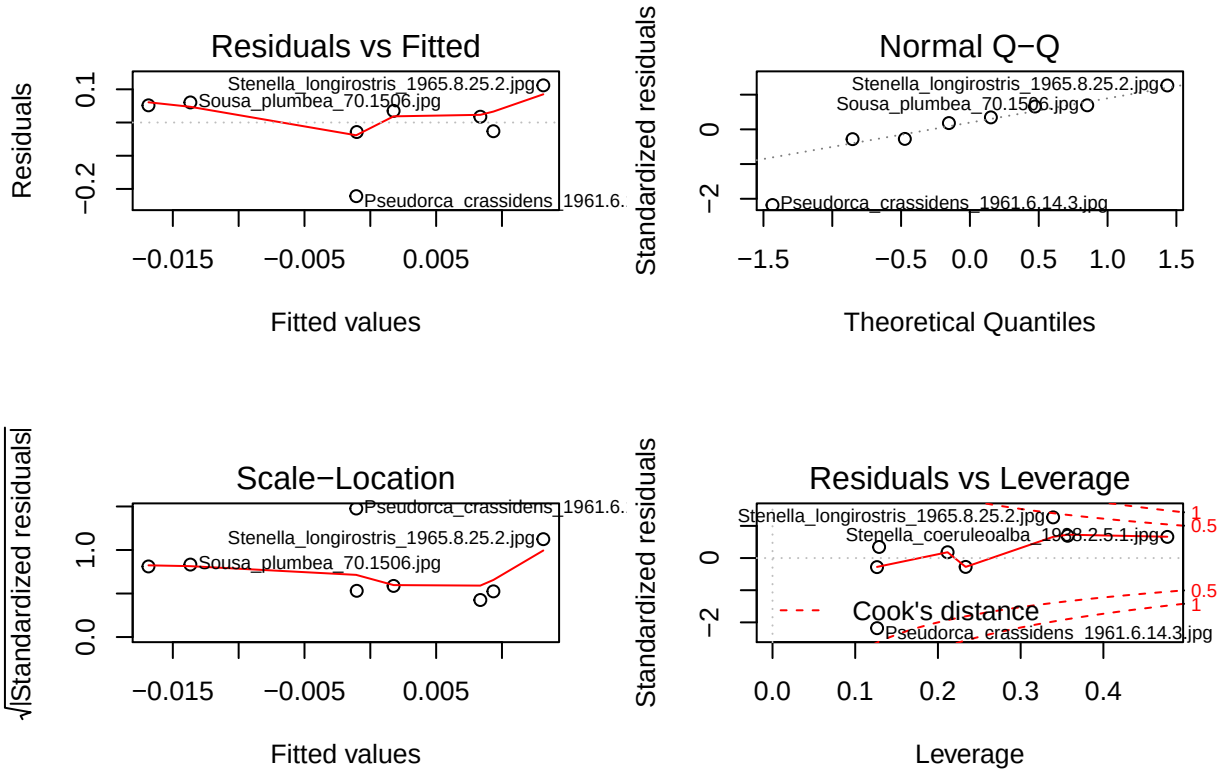
```
# Create new dataframe with PC scores, body size and diet.
# Body sizes from a random normal distribution with mean of 100 and sd of 25.
# Diet assigned as squid to first two specimens and fish to others.
mydata <- data.frame(pca.landmarks$pc.scores,
                     body.size = rnorm(n = 8, mean = 100, sd = 25),
                     diet = c(rep("squid",2), rep("fish", 6))
                     )
# Look at the data
str(mydata)
```

```
## 'data.frame':    8 obs. of  10 variables:
##  $ PC1      : num  -0.2228 0.0468 0.0258 0.0349 0.125 ...
##  $ PC2      : num  -0.0499 0.00149 0.02146 -0.04968 -0.07385 ...
##  $ PC3      : num  -0.000638 -0.041751 0.016942 0.000896 0.011896 ...
##  $ PC4      : num  -0.00103 0.00981 0.03057 0.00119 -0.01084 ...
##  $ PC5      : num  0.0068 0.00444 0.00488 -0.02318 0.00956 ...
##  $ PC6      : num  -7.16e-05 -8.64e-03 5.17e-03 -2.21e-03 4.33e-03 ...
##  $ PC7      : num  0.000387 -0.003641 0.001733 0.004996 -0.003752 ...
##  $ PC8      : num  0.00 2.08e-17 2.06e-17 1.39e-17 2.99e-17 ...
##  $ body.size: num  104.8 131.9 84.5 138.8 74.2 ...
##  $ diet     : Factor w/ 2 levels "fish","squid": 2 2 1 1 1 1 1 1
```

8.9 Regression

Regressions are done in the same way as usual.

```
# Fit model
model1 <- lm(PC1 ~ body.size, data = mydata)

# Assess assumptions
par(mfrow = c(2,2))
plot(model1)
```




```
par(mfrow = c(1,1))

# Look at overall model significance
anova(model1)
```

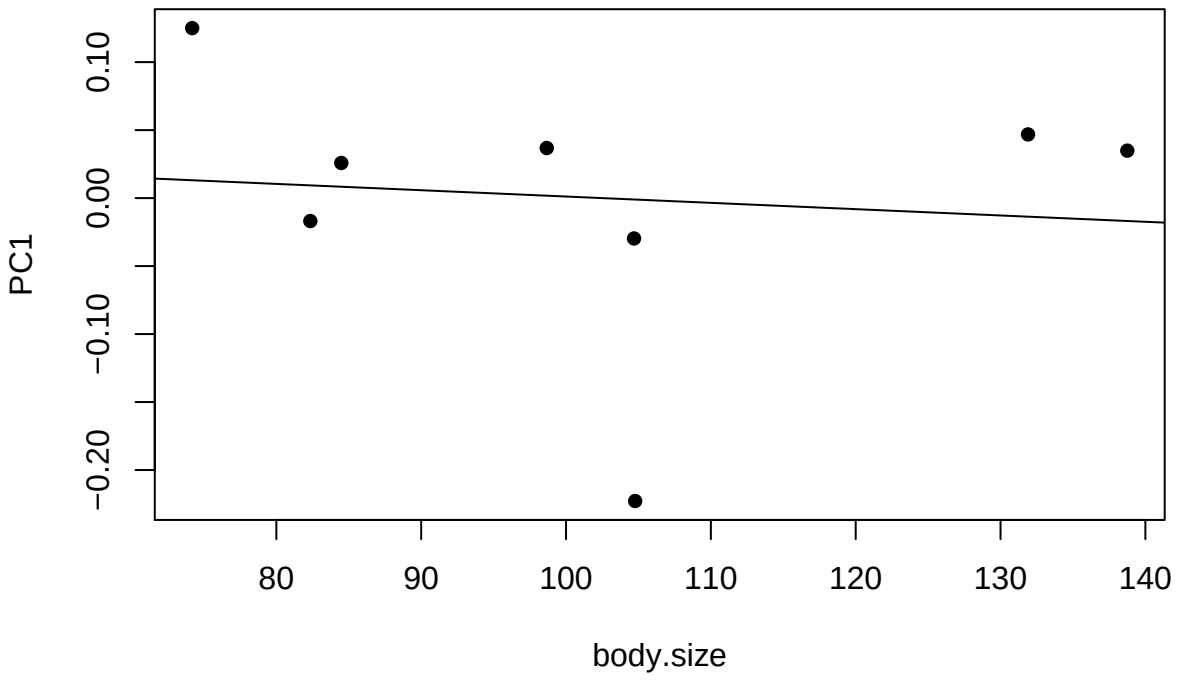
```
## Analysis of Variance Table
```

```
##
## Response: PC1
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## body.size  1 0.000806 0.0008059   0.068 0.8029
## Residuals  6 0.071060 0.0118434
```

```
# Look at parameters and their significance
summary(model1)
```

```
##
## Call:
## lm(formula = PC1 ~ body.size, data = mydata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22173 -0.02684  0.02626  0.05393  0.11187
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0475990  0.1864811   0.255    0.807
## body.size   -0.0004645  0.0017806  -0.261    0.803
##
## Residual standard error: 0.1088 on 6 degrees of freedom
## Multiple R-squared:  0.01121,    Adjusted R-squared:  -0.1536
## F-statistic: 0.06805 on 1 and 6 DF,  p-value: 0.8029
```

```
# Plot
plot(PC1 ~ body.size, data = mydata, pch = 16)
abline(model1)
```



- Can you remember how to interpret these results?
- Why will your results be different to mine?

8.10 Multivariate regression

Multivariate regression is the same as univariate regression but is used where you have multiple response variables, i.e. if we wanted to include PC1 and PC2 in the analysis. It's quite easy to implement, we just use `cbind` to bind together the different variables we want to include. Note that assumption checking isn't possible, nor is plotting in the usual way. Additionally, `summary(model)` presents the parameters for each response variable separately (e.g. `PC1 ~ body.size` then `PC2 ~ body.size`) so is not particularly useful here.

```
# Fit model
model2 <- lm(cbind(mydata$PC1,mydata$PC2) ~ body.size, data = mydata)

# Look at overall model significance
anova(model2)
```

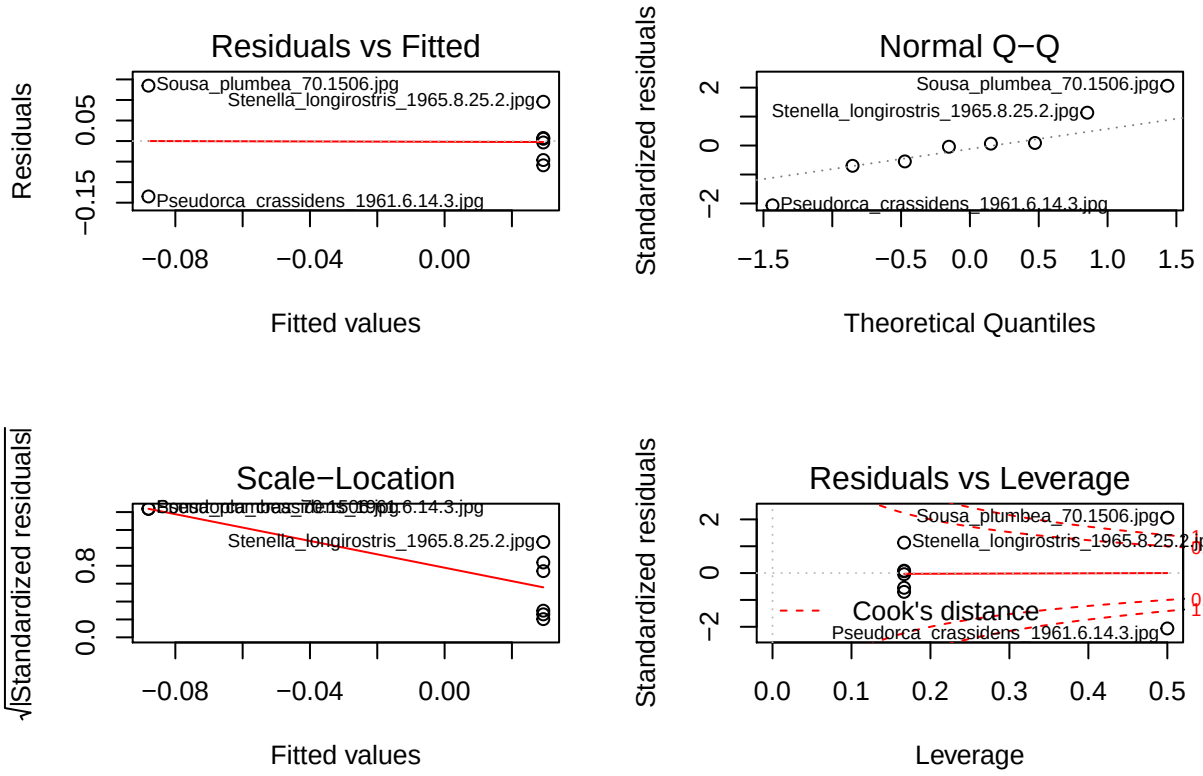
```
## Analysis of Variance Table
##
##             Df   Pillai approx F num Df den Df Pr(>F)
## (Intercept)  1 0.000000 0.000000      2      5 1.0000
## body.size    1 0.033752 0.087327      2      5 0.9177
## Residuals    6
```

- Can you guess how to interpret these results given what you know about regression?

8.11 ANOVA

ANOVAs are done in the same way as usual.

```
# Fit model
model3 <- lm(PC1 ~ diet, data = mydata)

# Assess assumptions
par(mfrow = c(2,2))
plot(model3)
```




```
par(mfrow = c(1,1))

# Look at overall model significance
anova(model3)
```

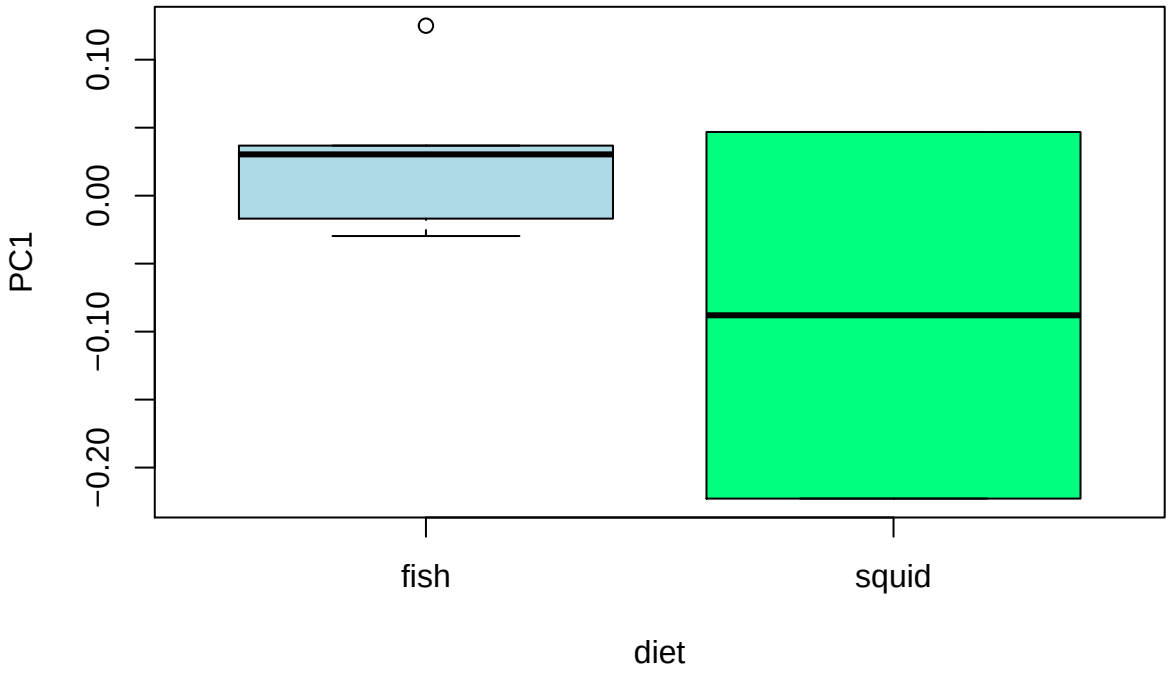
```
## Analysis of Variance Table
##
## Response: PC1
##           Df   Sum Sq  Mean Sq F value Pr(>F)
## diet       1 0.020644 0.020644  2.4182 0.1709
## Residuals  6 0.051222 0.008537
```

```
# Look at parameters and their significance
summary(model3)
```

```
##
## Call:
## lm(formula = PC1 ~ diet, data = mydata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.134813 -0.049412  0.001039  0.029529  0.134813
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02933    0.03772   0.778    0.466
## dietsquid   -0.11731    0.07544  -1.555    0.171
##
## Residual standard error: 0.0924 on 6 degrees of freedom
## Multiple R-squared:  0.2873, Adjusted R-squared:  0.1685
```

```
## F-statistic: 2.418 on 1 and 6 DF,  p-value: 0.1709
```

```
# Plot
plot(PC1 ~ diet, data = mydata, col = c("lightblue", "springgreen"))
```



- Can you remember how to interpret these results?

8.12 MANOVA

MANOVA is the same as ANOVA but is used where you have multiple response variables, i.e. if we wanted to include PC1 and PC2 in the analysis. It's quite easy to implement, we just use `cbind` to bind together the different variables we want to include. Note that assumption checking isn't possible for MANOVA in R at this time.

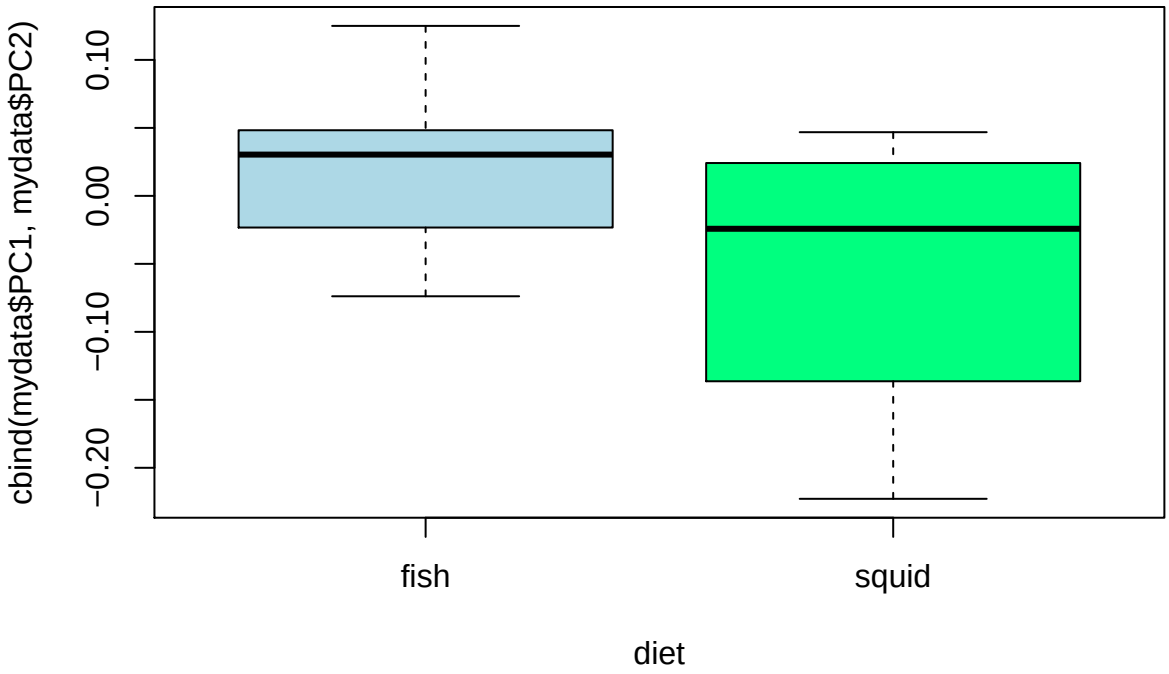
```
# Fit model
model4 <- manova(cbind(mydata$PC1,mydata$PC2) ~ diet, data = mydata)

# Look at overall model significance
anova(model4)
```

```
## Analysis of Variance Table
##
##             Df  Pillai approx F num Df den Df Pr(>F)
## (Intercept)  1 0.00000   0.0000      2      5 1.0000
## diet         1 0.37191   1.4803      2      5 0.3127
## Residuals    6
```

```
# Look at parameters and their significance
summary(model4)
```

```
##           Df  Pillai approx F num Df den Df Pr(>F)
## diet       1 0.37191   1.4803      2      5 0.3127
## Residuals  6
```

```
# Plot
plot(cbind(mydata$PC1,mydata$PC2) ~ diet, data = mydata, col = c("lightblue", "springgreen"))
```



- Can you guess how to interpret these results given what you know about ANOVA?

Of course the results here are nonsensical as we made up the data, but hopefully this gives you an idea of how we might use geometric morphometrics data in analyses. Other common analyses look at disparity of groups, convergence and divergence, evolution of shape and shape space etc.

8.13 Resources for learning geometric morphometrics

1. I highly advise getting hold of this book (Zelditch et al. 2012) from the library and reading the first few chapters, plus any chapters later in the book that are relevant to the analyses you will be doing. Don't panic too much about the maths or the equations, just try to get a general understanding of what each method is doing, especially GPA and PCA.

2. Another useful book is Morphometrics with R by Julien Claude. It is a little harder to read than Zelditch, but more focused on practical analysis in R. Note that I'd generally advise using the geomorph package (see links below) to do these analyses in R, but many of the principles are the same in this book which uses other methods. It's available as a PDF.

3. David Polly has an excellent set of lectures about all basic topics in geometric morphometrics including PCA and GPA. Note that these use Mathematica not R. There are also slides here from an R based course with a basic workflow for an analysis using geomorph. I'd recommend starting any project with a quick flick through these intro slides

4. Emma Sherratt has put together an excellent tutorial/vignette for geomorph from inputting landmarks to complex analyses.

5. The geomorph vignette may also be helpful for more complex analyses.

6. For error checking, take a look at Fruciano 2016, a review of the subject in Evolution. Also take a look at Claude pages 63-65 for ideas on sources of error.

8.14 References

- Zelditch, M.L., Swiderski, D.L., and Sheets, H.D.. 2012. Geometric Morphometrics for Biologists: A Primer. Academic Press.
- Claude, Julien. Morphometrics with R. Springer Science & Business Media, 2008.
- Fruciano, C. Measurement error in geometric morphometrics. 2016. Development genes and evolution 226:139-158.

Chapter 9

BAMM: Bayesian Analysis of Macroevolutionary Mixtures

The aims of this practical are to learn how to use R to use results obtained from BAMM to investigate rate shift regimes across phylogenies.

We will be using a dataset of body sizes from marsupials and monotremes. These data come from the PanTHERIA database (Jones et al. 2009) and the mammal supertree (Bininda-Emonds et al. 2008).

This practical is based on the Rabosky lab's excellent resource for using BAMM found here. I highly recommend reading it, especially if you want to use BAMM on your own data.

REMEMBER

- Download all of the data for the practical into a folder somewhere on your computer.
- Set your working directory to this folder.
- Start a new script for this practical.

You will also need to install the following packages:

- `BAMMtools`
- `coda`
- `viridis`

9.1 A quick intro to BAMM

Often in evolutionary biology we are interested in how clades (and their traits) diversify. The simplest models for looking at this are birth death models, where there is a rate of speciation (birth) and a rate of extinction (death) and these combine to get a rate of diversification (speciation - extinction). If diversification rates are high we tend to find phylogenies with lots of species.

The problem with these simple models is that they estimate a single rate of speciation and extinction across a whole tree. Obviously this is a huge oversimplification; we expect lots of changes in rates (or **rate heterogeneity**) through time, especially across fairly large trees. For example, we expect to see increases in speciation at points in time where a clade experiences an adaptive radiation, increases in extinction where the suddenly climate changes. BAMM (Bayesian Analysis of Macroevolutionary Mixtures) deals with this issue by allowing rates to vary across the tree (Rabosky et al 2014ab, Rabosky 2014).

More precisely, BAMM identifies **discrete shifts in rate at nodes of a tree** (either of speciation or of trait evolution, but we will focus on speciation here), i.e places where rates speed up or slow down. BAMM looks for rate shifts across the whole tree, so it can find 1, 2, 3 or more shifts. It does this using *reversible jump Markov Chain Monte Carlo* (i.e. Bayesian) methods to automatically explore a vast universe of possible models. It is biased towards simpler models (a common tactic in most evolutionary models - remember parsimony?) so rarely results in lots of rate shifts.

For more information it is worth checking out the references at the end of this practical.

9.2 A brief aside: how do I run BAMM?

BAMM is a command line program written in C++. Analysis and visualization of the results are performed using the R package `BAMMtools` (Rabosky et al. 2014a).

BAMM is pretty easy to run which makes it a slightly dangerous! It's easy to run with no idea of what you are doing or why. If you want to use it for your projects you'll need to do a lot of reading and thinking first. Here I am presenting a very simplified version so you can at least get a taste of what it does. **We will not run BAMM itself in this practical, just `BAMMtools` in R. This is just for interest.**

To run BAMM, you need to download the program first, then you run it via the command line (Terminal for Macs or something like cygwin in Windows). Your command line window will need to be opened from a folder containing the following files:

1. Your phylogeny
2. A control file (there is an example with the datasets)
3. Your data - if doing trait evolution analyses

The control file specifies lots of information about how the BAMM model should be run, including MCMC settings (e.g. number of generations, number of chains, acceptance rates), the priors (including your expectation for the number of shifts, rates of speciation and extinction) and the initial values of the parameters. It is **crucial** that these settings are carefully thought through, and that multiple different combinations are tried to make sure models are converging etc.

Running BAMM is then as simple as running this in the command line:

```
bamm -c control.txt
```

(assuming your control file is called `control.txt`, and your command line is running this in the folder containing the control file and your tree).

BAMM can take a long time to run for large datasets, and outputs several files with lots of different output. We will spend the rest of the practical focusing on some of these outputs.

9.3 What does a BAMM analysis produce?

Remember that BAMM is a Bayesian method, so rather than giving the one most likely result (like we get in maximum likelihood analyses), BAMM outputs a distribution of results, often containing thousands of results, that we call the posterior distribution (or just the posterior).

9.3.1 Distinct shift configurations and how to summarise them

For each MCMC run (in our BAMM analysis we ran the models 10^9 times!), BAMM simulates speciation and extinction along the tree, extracts the number of rate shifts (increase or decrease in speciation rate), and then works out the probability of that particular combination of rate shifts occurring. The results BAMM

outputs (to the `marsupial_event_data.txt` file) are a sample of all possible combinations across all 10^9 models, in proportion to their posterior probability. The posterior, or distribution of results, from BAMM will thus contain lots of different combinations of rates and rate shifts, but will have those that occur more often across the 10^9 models appearing more often, than those that occur rarely appearing less often.

In the posterior, we call each of these possible combinations **distinct shift configurations**. These are the most probable configuration of shifts from **one model** from the posterior. For example, one shift configuration may be a speed up at node 34 and a slow down at node 22 on model 10000. Each model in the posterior might have a different distinct shift configuration, or they might all be very similar. It depends on the dataset.

9.3.1.1 How do we report these results?

The number of possible distinct shift configurations is huge. Eventually if ran BAMM for for long enough you'd find a shift on every branch in the tree (because the branches can show shifts due to the effect of the prior alone).

We know that all the distinct shift configurations are *possible* but they aren't equally *probable*. As mentioned above some may be common, and others rare. We need some way of summarising thousands of models, and taking this into account. There are two main approaches.

1. **Overall best shift configuration** You can get this by looking at the maximum a posteriori (MAP) probability shift configuration, i.e. the one that appeared the most often in the posterior. This is a bit like using a consensus tree in phylogenetics. However, for most real datasets, the best rate shift configuration is merely one of a large number of possible rate shift configurations that have similar probabilities. So this method is not preferred (also if you've bothered to fit 10^9 models it seems pointless to just get one result!).
2. **Credible shift sets** An alternative way to present the results is to summarise all the distinct shift configurations. However, not all distinct shift configurations are going to be significant. Therefore, BAMM splits shifts into "important" ones that help explain the data (core shifts) and ones that are less important (or likely just due to priors) using *marginal odds ratios*. Specifically, BAMM computes the marginal odds ratio for each rate shift for every branch in the phylogeny. It then excludes all shifts that are unimportant using a pre-determined threshold value (usually 5). The remaining shifts are the credible shift set. These are usually reported in papers using BAMM.

9.4 A practical example looking at BAMM results using `BAMMtools`

This should all become clearer with an example. I have created a control file for the marsupial data (`marsupial_control.txt`) and run this with BAMM. The output files we will use are:

- marsupial_chain_swap.txt"
- marsupial_event_data.txt"
- marsupial_mcmc_out.txt"
- marsupial_run_info.txt"

As usual we'll first need to load the packages we need.

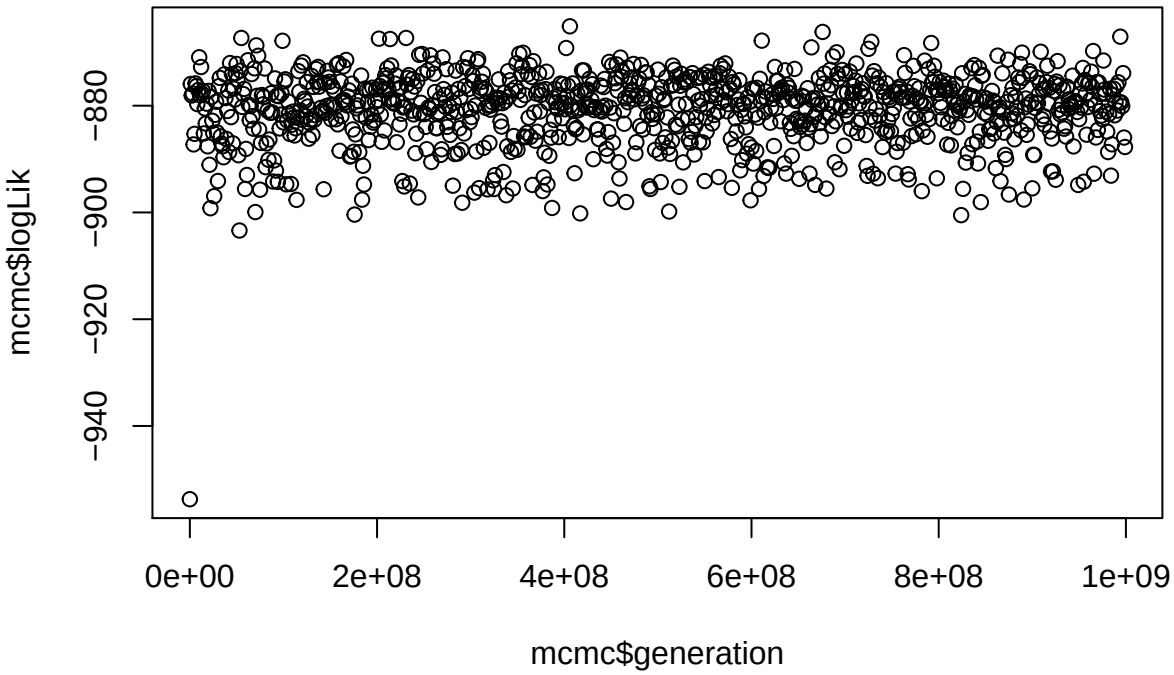
```
library(BAMMtools)
library(coda)
```

Note I'm assuming you've **set the working directory** and all the data files are in this folder.

9.5 Assessing MCMC convergence

Before we look at any of the exciting results, we need to check if our MCMC run actually converged. You should be familiar with this from building Bayesian phylogenies. Luckily BAMM is really helpful in what it outputs, so we can do this quite easily by looking at the `marsupial_mcmc_out.txt` file.

```
# Read in the file
mcmc <- read.csv("marsupial_mcmc_out.txt")
# Take a look at the plot
plot(mcmc$logLik ~ mcmc$generation)
```



This gives us a vague idea of whether your run has converged. SPOILER alert, it has because I played around with the parameters in BAMM for ages until it did!

Again, as with Bayesian phylogenies, we next want to discard some runs as burnin. Let's discard the first 10% of samples.

```
burnstart <- floor(0.1 * nrow(mcmc))
postburn <- mcmc[burnstart:nrow(mcmc), ]
```

We can also use the `coda` library to check the effective sample sizes (ESS) of the log-likelihood and the number of shift events present in each sample.

```
effectiveSize(postburn$N_shifts)
```

```
##     var1
## 1077.517
```

```
effectiveSize(postburn$logLik)
```

```
## var1
##  901
```

In general, we want these to be at least 200, and larger for small datasets. Note that these are standard procedures for any Bayesian analysis, not anything magical invented just for BAMM.

OK great now we know that our analysis converged we can look at the cool stuff.

9.6 Analysis of rate shifts in marsupial phylogeny

We first need to load our phylogeny and the output of BAMM called the event data file. This contains most of the outputs we've mentioned above, such as the branch-specific speciation rates, positions of rate shifts etc.

Note that we again exclude the first 10% of samples as burn-in using `burnin = 0.1`.

```
tree <- read.tree("marsupialtree.tre")

# This step may take some time...
edata <- getEventData(tree, eventdata = "marsupial_event_data.txt", burnin = 0.1)
```

```
## Reading event datafile:  marsupial_event_data.txt
##      ..........
## Read a total of 1000 samples from posterior
##
## Discarded as burnin: GENERATIONS <  99000000
## Analyzing  901  samples from posterior
##
## Setting recursive sequence on tree...
##
## Done with recursive sequence
```

The message this produces tells you how many samples were read and how many excluded as burnin.

9.6.1 How many rate shifts?

The first thing we might like to know is how many rate shifts occur in our tree? We can do this using our event data (`edata`) to look at the posterior probabilities of models sampled by BAMM.

```
shifts <- summary(edata)
shifts
```

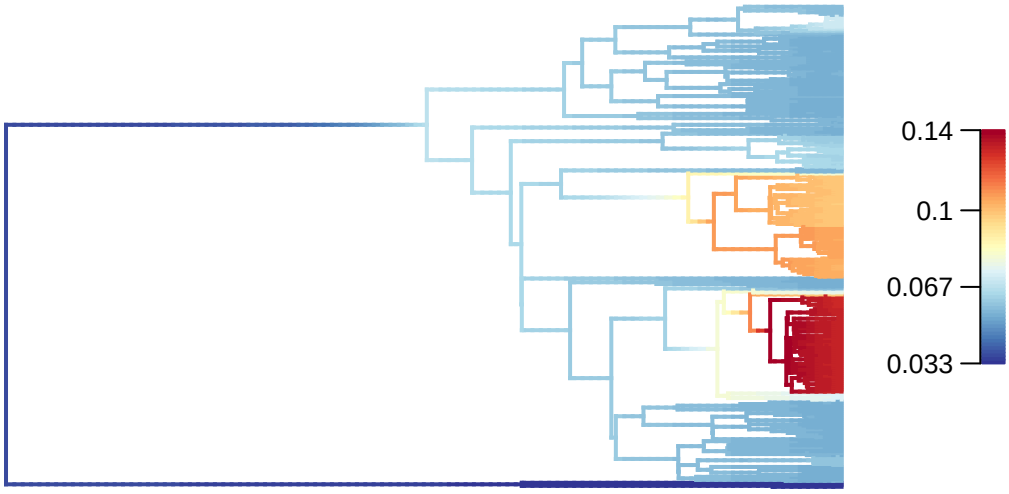
This should show the posterior probabilities of each number of rate shifts observed during simulation of the posterior. This doesn't tell us *where* they occur on the tree, just how many there are, and what their probability was.

Here it looks like 4 shifts occurred most often, but with a range in number of shifts from 0 to 11.

9.7 Mean phylorate plot

Next we might want to visualise these results on our tree. Mean phylorate plots are a way of visualizing the mean, model-averaged diversification rates along every branch of the tree. We can plot one using `plot.bammdata`.

```
plot.bammdata(edata, lwd = 2, legend = TRUE, cex = 0.5)
```

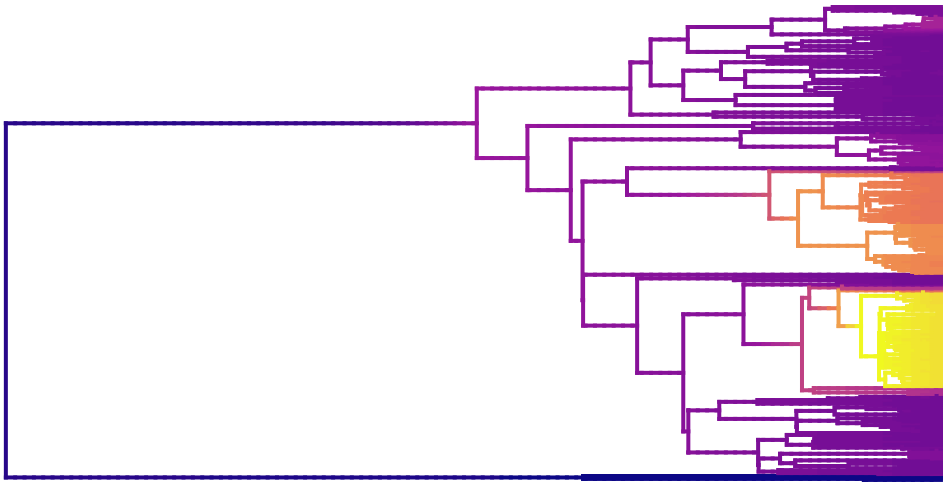


High rates are in red or yellow, and the low rates in blue. You can add the option `labels = TRUE` to get tip labels, but they're really hard to read so you might need to look at a normal tree plot `plot(tree, cex = 0.5)` to see which species are involved here. I've provided a large PDF version of the tree in the folder. Also check out [https://en.wikipedia.org/wiki/Marsupial] to learn more about marsupials!

In our data we see low rates in the monotremes (the three species in dark blue at the bottom) which is not surprising given their separation from the rest of the tree and very long branches. The two clades that appear to evolve more quickly are the Dasyuromorphia, the marsupial carnivores (including quolls and Tasmanian devils - though these are missing from this phylogeny), and the Macropodidae, the kangaroos and wallabies. Again this makes sense as these groups are very species rich. American possums, and other Australasian marsupials have much lower rates of evolution.

We can also plot with slightly nicer colour-blind friendly colours from the package `viridis` (have a look at the options and choose your favourite).

```
library(viridis)

plot.bammdata(edata, lwd = 2, pal = plasma(4))
```



9.8 Best single distinct shift configuration

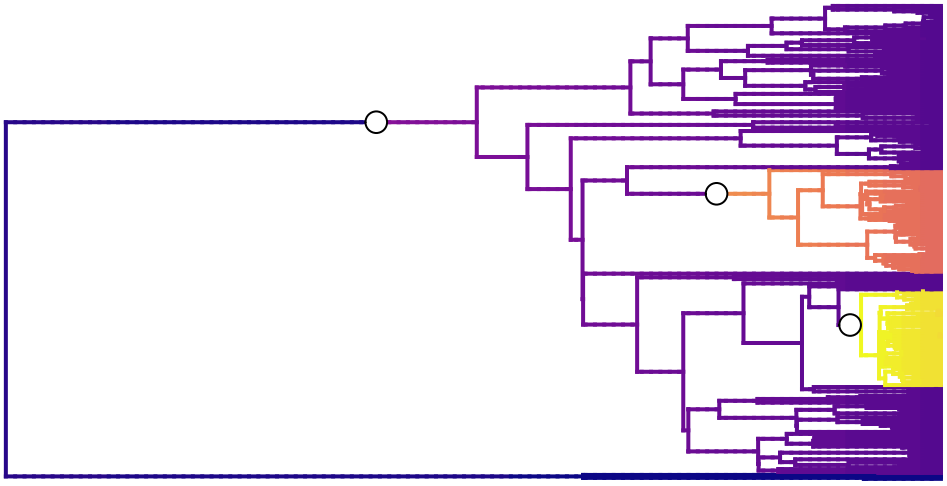
To extract the overall best distinct shift configuration, we use the maximum a posteriori (MAP) probability shift configuration, i.e. the one that appeared the most often in the posterior, as follows.

```
best <- getBestShiftConfiguration(edata, expectedNumberOfShifts = 1)
```

```
## Processing event data from data.frame
##
## Discarded as burnin: GENERATIONS <  0
## Analyzing  1  samples from posterior
##
## Setting recursive sequence on tree...
##
## Done with recursive sequence
```

We can plot this as follows:

```
plot.bammdata(best, lwd = 2, pal = plasma(4))
addBAMMshifts(best, cex = 1.5, bg = "white")
```



The best single distinct shift configuration consists of three shifts. One on the branch leading to the marsupials, perhaps unsurprising given the low rates at the root with monotremes, and another two at the clades we already identified as having high rates.

9.9 Credible sets of shifts

Using just one answer is not very desirable, so let's look at the shifts from across the posterior. As described above, the number of possible distinct shift configurations is huge, but not all these shifts are going to be significant. BAMM computes the marginal odds ratio for each a rate shift for every branch in the phylogeny. It then excludes all shifts that are unimportant using a pre-determined threshold value (usually 5). The remaining shifts are the credible shift set.

To extract the credible shift set for our data, we can use the `BAMMtools` function `credibleShiftSet`.

```
css <- credibleShiftSet(edata, expectedNumberOfShifts = 1, threshold = 5, set.limit = 0.95)
summary(css)
```

```
##
##  95 % credible set of rate shift configurations sampled with BAMM
```

```
##
## Distinct shift configurations in credible set:  113
##
## Frequency of 9 shift configurations with highest posterior probability:
##
##
##    rank     probability cumulative  Core_shifts
##          1 0.09433962 0.09433962          3
##          2 0.08768036 0.18201998          3
##          3 0.08102109 0.26304107          3
##          4 0.06104329 0.32408435          1
##          5 0.05771365 0.38179800          3
##          6 0.05660377 0.43840178          3
##          7 0.04883463 0.48723640          3
##          8 0.03884573 0.52608213          2
##          9 0.03884573 0.56492786          2
##
## ...omitted 104 additional distinct shift configurations
## from the credible set. You can access the full set from your
## credibleshiftset object
```

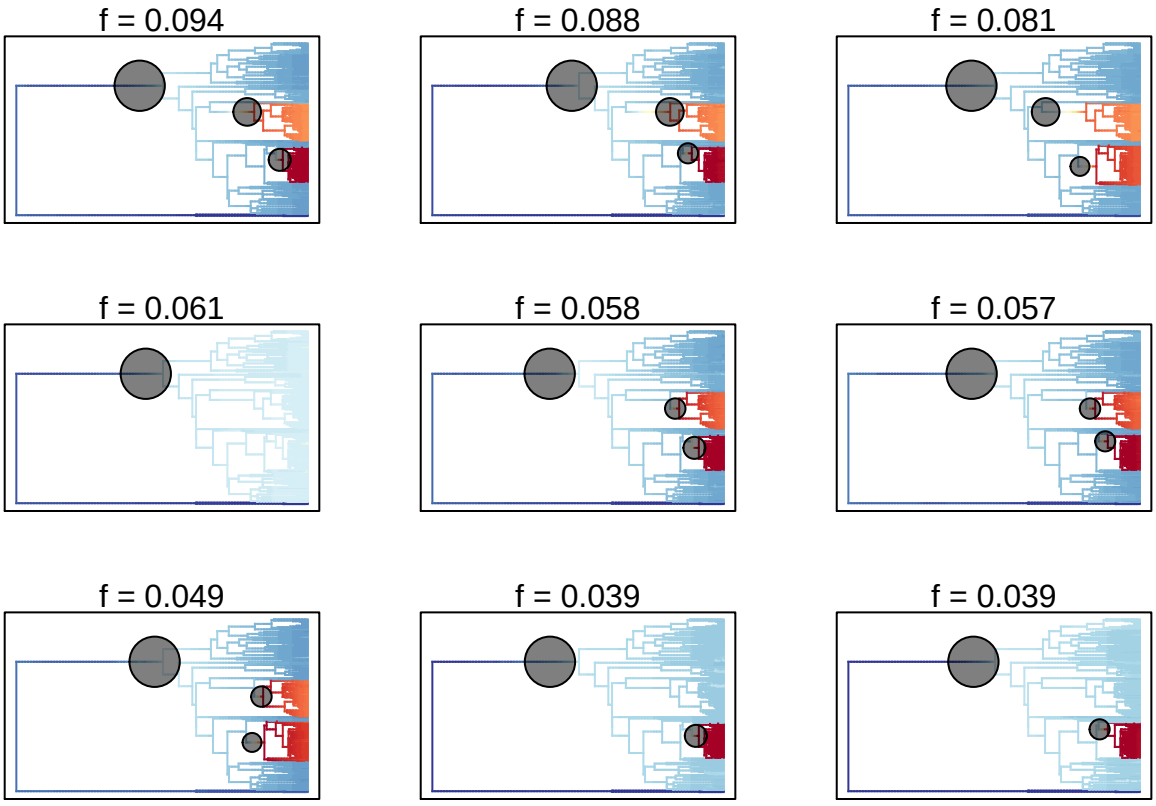
Here we see that there are 113 distinct shift configurations in our posterior. This highlights why using BAMM is a good idea - we can incorporate the distribution of results rather than giving just one best answer.

We also see that even though there are 113 distinct configurations in the 95% credible set, 9 of these account for most of the probability of the data.

We can generate phylorate plots for each of these shift configurations using `plot.credibleshiftset`

```
plot.credibleshiftset(css)
```

```
## Omitted 104 plots
```



The text above each phylorate plot gives the posterior probability of each shift configuration. Because many samples from the posterior can be assigned to each distinct shift configuration, the phylorate plots generated by `plot.credibleshiftset` are model-averaged mean rate parameters across all samples assignable to a given configuration. The shifts themselves are indicated with circles on branches, with the circle size being related to the probability of that particular shift.

Note: this will set the plotting window to plot 9 plots, so we need to take it back to the default of one plot using:

```
par(mfrow = c(1,1))
```

9.10 Clade specific evolutionary rates

We might also be interested in seeing if particular clades have different rates of speciation or extinction. This can be done in `BAMMtools` using the function `getCladeRates`, which computes the average rate for the chosen clade.

Here we might be interested in seeing how different the rates are in our fastest clade, the macropodids, compared to the rest of our species.

First we can look at the average rate across the whole tree using `getCladeRates`.

```
allrates <- getCladeRates(edata)
```

`allrates` is a list with speciation and extinction rates, with the mean rate across all marsupials for each sample in the posterior. We can extract the mean speciation rate for our tree and estimate the 90% highest posterior density (HPD) as follows.

```
mean(allrates$lambda)
```

```
## [1] 0.06926659
```

```
quantile(allrates$lambda, c(0.05, 0.95))
```

```
##         5%        95%
## 0.06160736 0.07829082
```

To get the rates for a specific clade, we just specify the node leading to that clade.

In the our marsupial/monotreme example, node 301 is the node number of the macropodid clade (you can find identify node numbers using `plot.phylo` and `nodelabels` from the `ape` package). We can estimate the mean speciation rates for macropodids as follows:

```
kangaroorates <- getCladeRates(edata, node = 301)
mean(kangaroorates$lambda)
```

```
## [1] 0.1323322
```

```
quantile(kangaroorates$lambda, c(0.05, 0.95))
```

```
##         5%        95%
## 0.07387751 0.18106431
```

To get the rates for everything but the macropodids, we do the same thing, but add the argument `nodetype = "exclude"`

```
nonkangaroorate <- getCladeRates(edata, node = 301, nodetype = "exclude")
mean(nonkangaroorate$lambda)
```

```
## [1] 0.06298746
```

```
quantile(nonkangaroorate$lambda, c(0.05, 0.95))
```

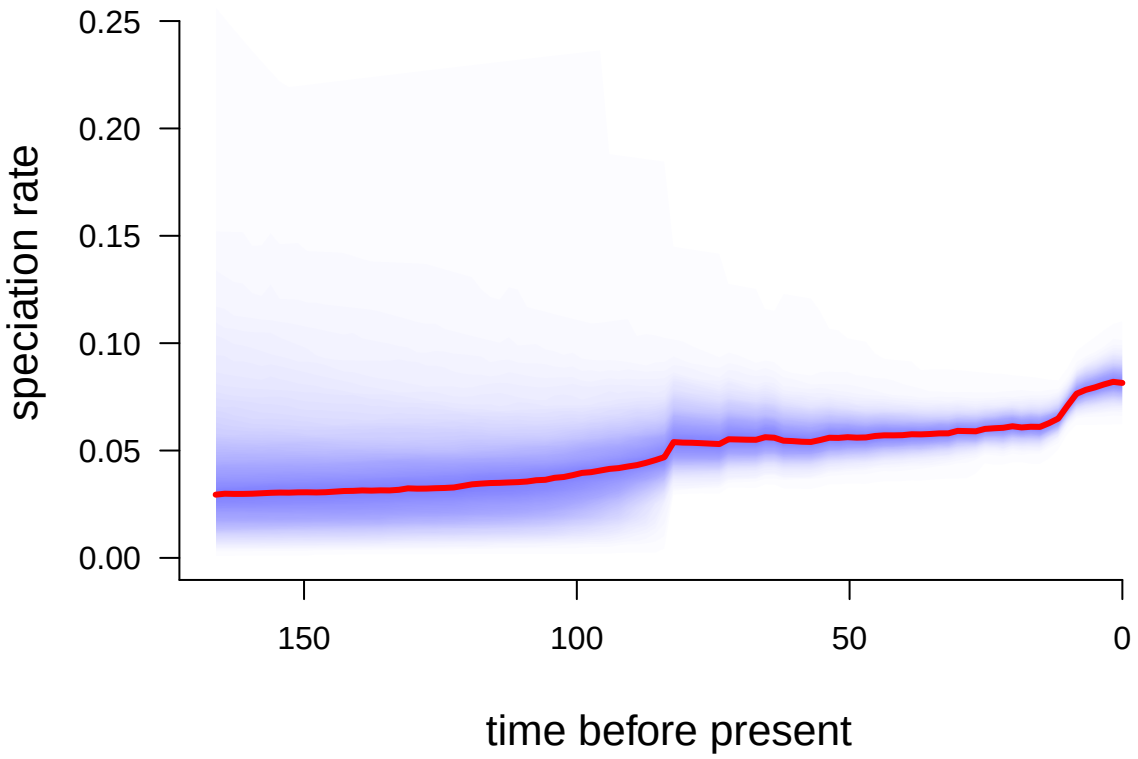
```
##         5%        95%
## 0.05438734 0.07304822
```

Note that these are *mean time-averaged clade-specific rates*. If diversification rates have changed dramatically through the history of a particular clade, a single overall mean rate might not be particularly informative.

9.11 Rate-through-time analysis

We may also want to visualise how rates of speciation change through time on our tree. We can do this by plotting a **rate-through-time curve** using the `plotRateThroughTime` function.

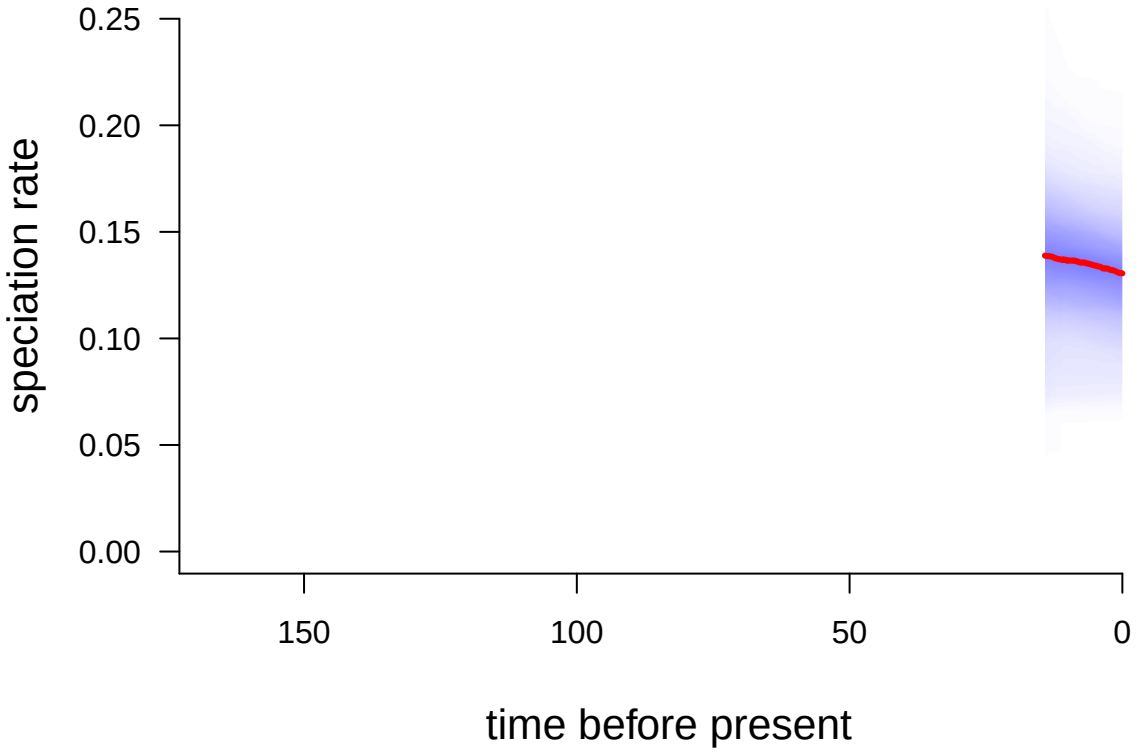
```
# This may take some time...
plotRateThroughTime(edata, ratetype = "speciation")
```



The red line is the average speciation rate, with density shading showing the confidence intervals.

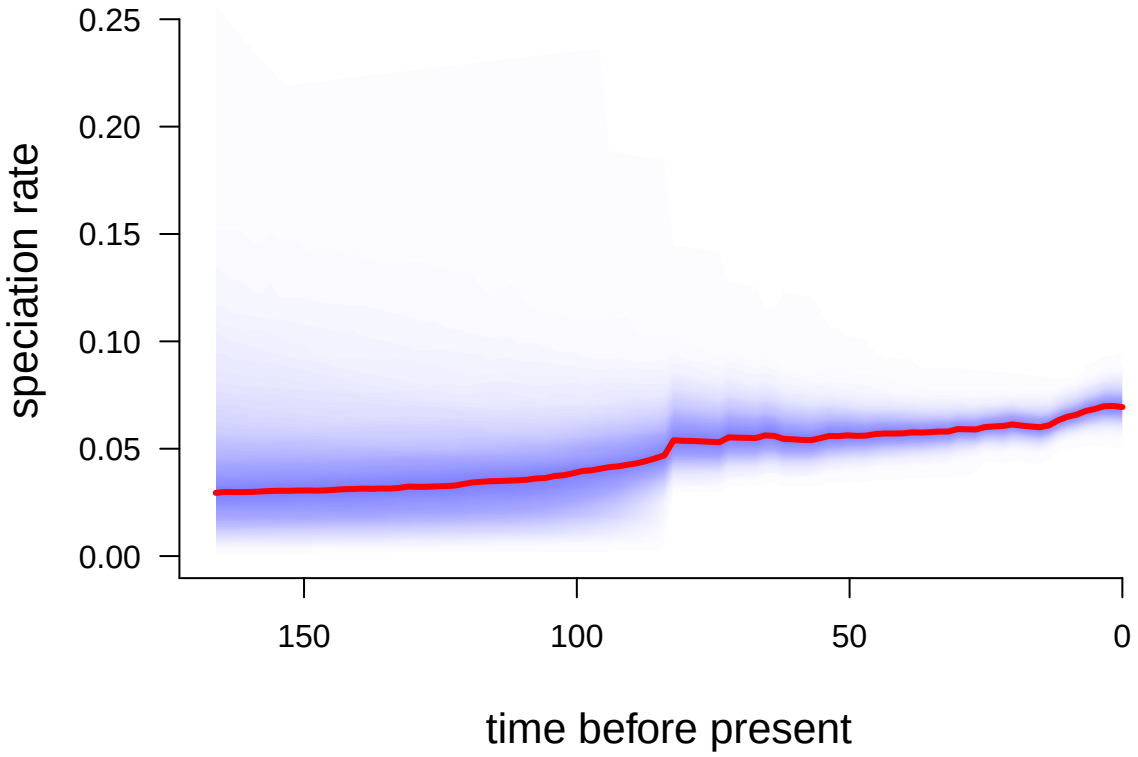
You can also use `plotRateThroughTime` to plot speciation through time curves for just a portion of your phylogeny. Here we can just look at the macropodids (node 301):

```
plotRateThroughTime(edata, node = 301, nodetype="include")
```



And compare these to the phylogeny minus the macropodids:

```
plotRateThroughTime(edata, node = 301, nodetype="exclude")
```

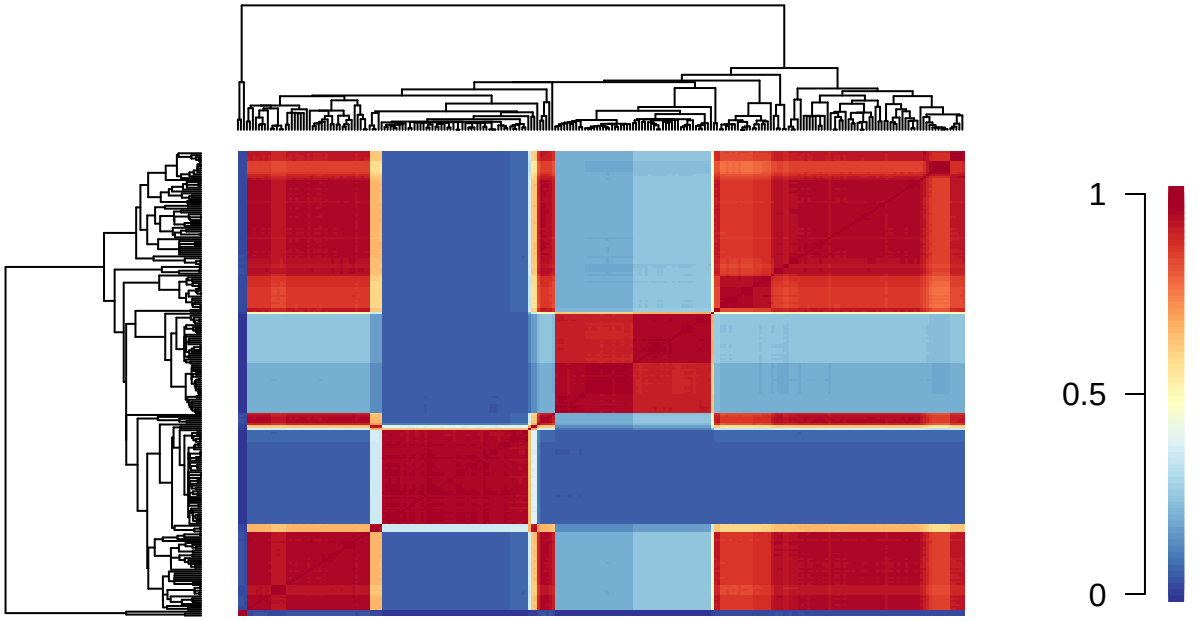


9.12 Macroevolutionary cohort analysis

A final nice feature of `BAMMtools` that we'll play with today (there are others too) is the ability to easily perform a macroevolutionary cohort analysis (Rabosky et al 2014b). These are a good way of summarising which species/clades share correlated macroevolutionary dynamics i.e. similar rates of speciation etc. The basic idea is to visualize the pairwise probabilities that any two species share a common macroevolutionary rate regime.

First we generate a cohort matrix, which contains the pairwise probabilities of shared macroevolutionary dynamics. We then pass thsi to the `cohorts` function, which generates the plot.

```
cmat <- getCohortMatrix(edata)
cohorts(cmat, edata)
```



Each cell of the cohort matrix corresponds to a pair of tip taxa from the phylogeny. You read these plots by drawing an imaginary line between any two tips of the tree - the color of the matrix at their intersection is the probability that those taxa share a common rate dynamic. Species pairs that share a common rate dynamic are red, species with different macroevolutionary dynamics are blue.

Here there appear to be a very complicated set of macroevolutionary dynamics (again I wonder why I decided to use a completely untested real example for this practical!). Dasyurids and macropodids appear to be similar, as do the American possums and the Australasian possums. There are also small pockets of similarity across the other marsupial groups, for example bettongs have a similarly slow rate to the possums.

Check out this example with whales for an easier to interpret plot!

9.13 Assumptions and issues with BAMM

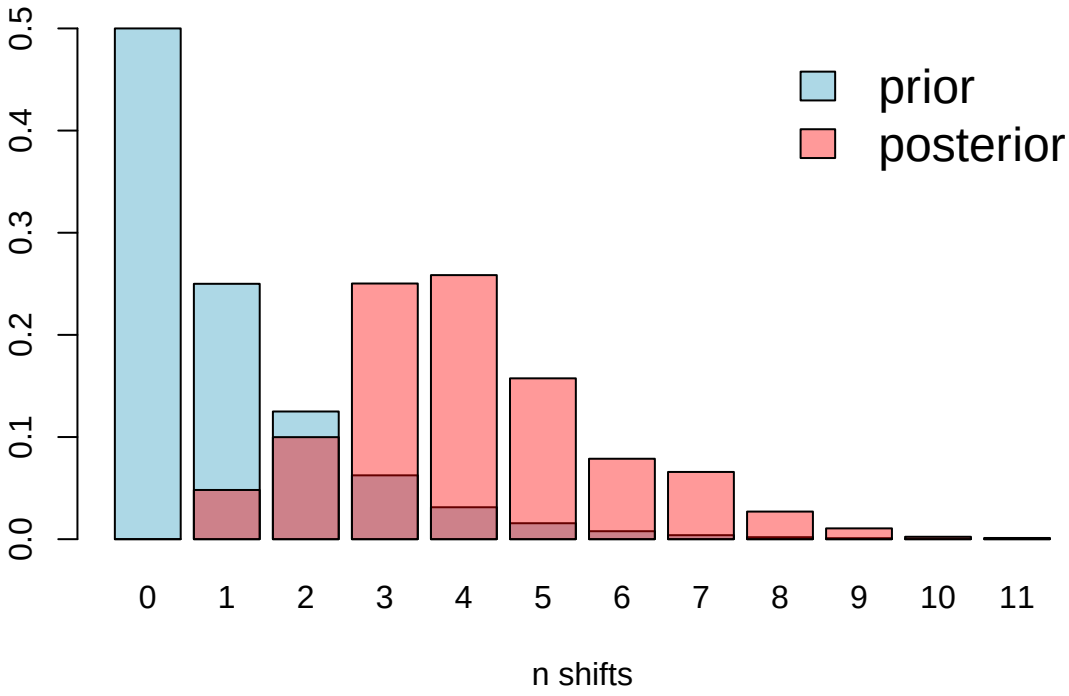
Like all methods, BAMM has a number of important assumptions and issues.

First, it assumes that evolutionary dynamics are described by *discrete* shifts at nodes. It could equally be gradual changes along branches. BAMM cannot detect this, but neither can any other method. However it is worth remembering this when interpreting results, especially on long branches.

Second, the prior for the number of expected shifts will have a large effect on how many shifts are detected, particularly for long branches as the probability of seeing a shift due to the prior alone increases with branch length. To solve this BAMM estimates marginal odds ratios, scaling each marginal shift probability by the prior and branch length.

You can (and should) check for this problem using the code below.

```
postfile <- "marsupial_mcmc_out.txt"
bfmat <- computeBayesFactors(postfile, expectedNumberOfShifts = 1, burnin=0.1)

plotPrior(mcmc, expectedNumberOfShifts = 1)
```



There is some overlap, but it's clear that the prior is different to the posterior, so this issue is not affecting our analyses.

Thirdly, BAMM (and all other similar methods) gives inaccurate results for phylogenies with incomplete sampling, i.e. where you don't have every species in your phylogeny. This may be quite likely if you're working with invertebrates or plants or pretty much anything other than mammals and birds! If the sampling is non random, for example, you're missing a whole clade, the results will be even weirder. BAMM has ways of dealing with this, but it's best to try and avoid the problem in the first place by only using BAMM where you've got a fairly complete phylogeny, or at least one where the missing species are missing at random. Rabosky and colleagues recommend that if <10% of the species are in your phylogeny, you should add them in yourself. See [http://bamm-project.org/advanced.html#incompsampling] for more help with this.

Additionally, there has been a quite a lot of debate about the validity of BAMM in the literature recently. Moore et al. 2016 proposed several serious issues with BAMM, which basically boil down to two main points:

- BAMM outputs are highly sensitive the priors
- The likelihood function BAMM uses is wrong

Rabosky and colleagues have refuted these criticisms here: [http://bamm-project.org/prior.html], [http://bamm-project.org/replication.html], [http://bamm-project.org/developertoggle.html], [http://bamm-project.org/mea_likelihood.html], and [http://ift.tt/2m7qv6T]. There's no consensus just yet as to whether BAMM is usable or not. I would say use with caution, but be prepared to defend your choice!

9.14 References

- BAMM website [http://bamm-project.org/documentation.html]
- Moore et al 2016. Critically evaluating the theory and performance of Bayesian analysis of macroevolutionary mixtures. PNAS, 113:9569-9574. [http://www.pnas.org/content/113/34/9569.full]
- Rabosky, 2014. Automatic Detection of Key Innovations, Rate Shifts, and Diversity-Dependence on Phylogenetic Trees. PLoS ONE. [http://dx.doi.org/10.1371/journal.pone.0089543]

- Rabosky et al. 2014a. BAMMtools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. Methods Ecol Evol, 5: 701-707 [10.1111/2041-210X.12199].
- Raboksy et al. 2014b. Analysis and Visualization of Complex Macroevolutionary Dynamics: An Example from Australian Scincid Lizards. Syst Biol (2014) 63 (4): 610-627. [https://doi.org/10.1093/sysbio/syu025].
- Rabosky et al 2017. Is BAMM flawed? Theoretical and practical concerns in the analysis of multi-rate diversification models. Syst Biol [http://ift.tt/2m7qv6T]
- Shi & Rabosky 2015. Speciation dynamics during the global radiation of extant bats. Evolution, 69: 1528-1545 [10.1111/evo.12681].

Chapter 10

Critical thinking about methods and analyses

Often it is tempting accept the results of papers at face value. They were published so they *must* be correct, right? Sadly no. Even the best papers have flaws. There may be problems with the data, method or interpretation of the results. Some of these are unavoidable, some reflect misunderstandings of the methods used, and others are just mistakes. Learning to critically read the scientific literature (or indeed any literature in this age of fake news!) is therefore a key skill to develop.

As practice we will split into groups and critically evaluate recent papers using some of the methods we've worked with in this module. I'll assign papers to everyone on the first day. Read the paper before class, and make notes of things you don't understand or disagree with. I have provided some guidance of things to look for below. The question you should keep asking yourself throughout is **given the data and methods, do I trust the conclusions of the paper?**

10.1 How to critically evaluate a paper

10.1.1 Logic/interpretation

- What questions does the paper address?
- Do the analyses/data actually answer the questions the paper is meant to be asking, or do they answer a different question?
- What are the conclusions? Do the analyses/data support the conclusions?
- Have the authors exaggerated the importance of their conclusions (e.g. evidence of one species shifting range in the ice age and the conclusions are that climate change is going to be fine?)
- Is the logic of the paper clear and justifiable, given the assumptions?
- Are there any flaws in the authors reasoning?
- Do you agree with how the authors interpret their results?

10.1.2 Data

- What's the sample size? Is it large enough to support the conclusions of the paper?

- How many species are missing from the analysis? Does this worry you?
  - Is two species missing from a clade of 50 species a problem?
  - Can 50 species be used to make conclusions about a clade of 1000s of species?
- Are species missing in a way which might influence the results?
  - Would you be concerned if all species from one clade were missing?
  - Are the species present well distributed across the phylogeny?
- Are fossil/extinct species considered? Would this influence the results/conclusions?
- How were the data collected? Could this bias the results at all?
- Are there biases in the age, sex, geographic locality etc. of species included?
- Do you think the data quality is high enough?
- Would other data have been better to answer this question?

10.1.3 Methods

- Check the text carefully for caveats. These may appear in the introduction, methods, results or discussion. Did the authors deal with them or just mention them?
- What are the assumptions/limitations of the method being used? These may be mentioned in the text, or you may need to dig into the literature to find them (don't worry about this for the class though do check the handouts from practicals for some pointers).
- Are the assumptions the authors make reasonable? For example, a big assumption underlying all phylogenetic methods is that the phylogeny is correct. Do you agree?
- Be aware that some older methods may have been superseded by better methods.
- Be aware that sometimes there is debate in a community about the best method to use (e.g. the BAMM debate).

10.1.4 Moving forwards

- What are the good things in this paper? Make sure that you don't ignore the positive in your hunt for the negative!
- Do these ideas have other applications or extensions that the authors might not have thought of?
- How would you fix the flaws in this paper?
- How might the paper be useful to you? For example, as a paper to cite in your thesis, a method to use, or a cautionary tale of what not to do?!

10.2 References

These papers involve critiques/reviews of some of the methods we've been learning about in this module. They may be helpful for some of the papers.

- Cooper et al. 2016a. Shedding light on the "Dark Side" of phylogenetic comparative methods. Methods Ecology and Evolution.
- Freckleton 2009. The seven deadly sins of comparative analysis. Journal of Evolutionary Biology.
- Losos 2011. Seeing the forest for the trees: the limitations of phylogenies in comparative biology. American Naturalist.

- Cooper et al. 2016. A cautionary note on the use of Ornstein Uhlenbeck models in macroevolutionary studies. Biological Journal of the Linnaean Society.
- Kamilar and Cooper 2013. Phylogenetic signal in primate behaviour, ecology, and life history. Phil Trans Roy Soc B.
- Moore et al. 2016. Critically evaluating the theory and performance of Bayesian analysis of macroevolutionary mixtures. PNAS.

Chapter 11

Practice Questions

11.1 Diversity indices in R

1. Using `?diversity` find out how to calculate **inverse Simpson's index**, then calculate it for the Pokemon dataset. What is the inverse Simpson's index for site 10?
2. Try out three other betadiversity indices with the pokemon data. Do they all agree? How would you interpret the results from the betadiversity analyses?
3. Create a new dataset of Pokemon sightings (or anything else you like). Calculate species richness, Simpson's index, Jaccard's index, Chao2 estimate, and plot a species accumulation curve in colours of your choice.

11.2 PGLS in R

1. What is λ for primate social group size?
2. Make a plot of social group size against home range size.
3. Does home range size differ significantly among the different primate genera?
4. What is Blomberg's K for adult body mass?
5. Run a PGLS analysis to determine the relationship between ln(longevity) (Y variable) and ln(body mass) (X variable). 1. How does this differ from the result obtained with an OLS regression? 1. Plot the result. 1. Plot the lambda profile 1. Look at the confidence intervals on the estimate of lambda. How would you interpret these?

11.3 Macroevolutionary models

1. Fit Brownian, OU and Early burst models to the magical power variable. Which model fits best according to AIC weights? How could you interpret this result biologically?
2. Fit equal rates, symmetric rates and all rates different models to the habitat variable. Which model fits best according to AIC weights? How could you interpret this result biologically?
3. Reconstruct ancestral states for habitat on the phylogeny and plot the result (with node and tip labels) using colours of your choice.
4. Fit a multi-rate Brownian motion model using `OUwie`, with habitat types as regimes, and body size as the evolving trait. What would be your next step in trying to find the best fitting model for these variables? How would you interpret the results?
5. Invent your own variable to add to the dataset and fit models of evolution (continuous or discrete) to it as appropriate.

11.4 Geometric morphometrics

1. Plot the min/max for PC3 scores in comparison to the reference shape with three fold magnification.
2. In one plotting window, plot PC1 against PC2, PC2 against PC3, and PC1 against PC3 (clue `par(mfrow = ???)`). With the help of `pca.landmarks$rotation` and the whale photgraphs, can you interpret the plots in relation to the morphology of the species?
3. Perform a multivariate regression using PC1, PC2 and PC3 as response variables, and body size as the explanatory variable.
4. Perform a MANOVA using PC1, PC2 and PC3 as response variables, and diet as the explanatory variable.
5. Invent your own variable to add to the dataset and fit a regression or MANOVA to it as appropriate.