Comparative Methods Workshops

Chris Mitchell

2020-06-23

Contents

1 Preamble 5

2 Introduction 7

3 Workshop 1: Plotting phylogenies in R 9

3.1 Phylogenetics in R 9
3.2 Importing your tree 10
3.3 ggtree 11
3.4 Example Phylogeny 20

4 Workshop 2: PGLS 29

4.1 Data 29
4.2 Plotting with ggplot2 29
4.3 Linear Regression 34
4.4 Trees and Phylogenetic Signal 36
4.5 Phylogenetic Regression 40
4.6 Conclusion 47

Chapter 1

Preamble

This book contains a series of workshops on the use of phylogenetically controlled comparative methods. The workshops assume a basic familiarity with R. You should revisit the material from LIFE223 if you need a quick refresher on R.

Chapter 2

Introduction

Chapter 3

Workshop 1: Plotting phylogenies in R

3.1 Phylogenetics in R

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

3.1.1 Packages used

In this tutorial we'll mostly be using a package called ggtree. To install it, we need another package called BiocManager.

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

We will also need to use phylobase, ggimage and it would help to have the tidyverse packages loaded since we'll be using the syntax of ggplot2. If you get an error message, make sure the packages are installed first.

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

3.2 Importing your tree

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

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

If we take a look at the structure of our tree object using the **str** function. The tree is stored as an object of class **phylo**.

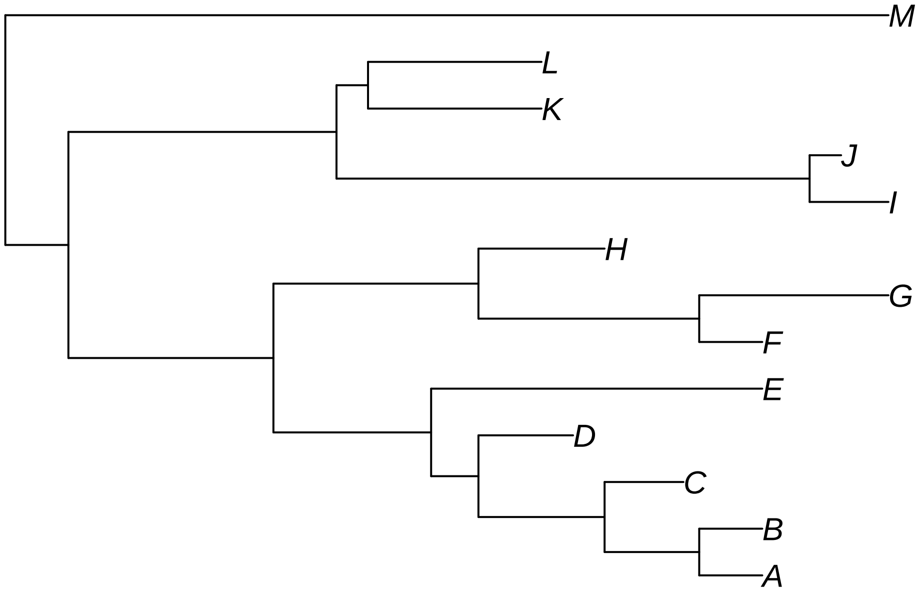
```r
str(tree)
```

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

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

Things become clearer when we plot the tree. We can do this with the **plot** function in base R.

```r
plot(tree)
```



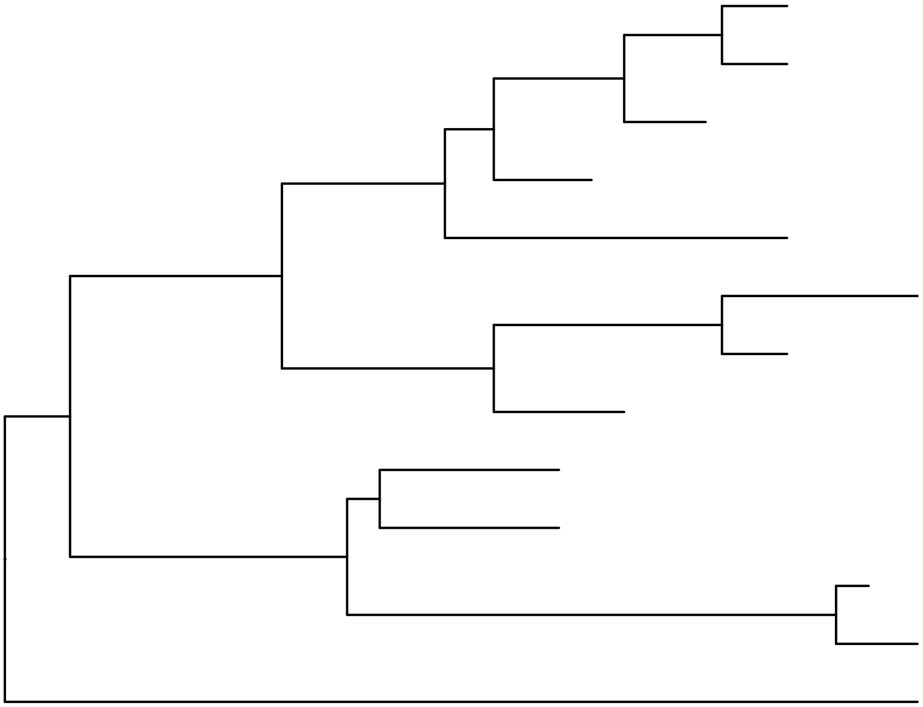
This plot is fine for a quick check to make sure the tree looks as we expected it to. Let's look at making a more attractive plot with ggtree.

3.3 ggtree

The syntax we'll be using here is a little different that what you may be used to so don't get intimidated. **ggtree** uses the same syntax as a package called **ggplot2**. This works by creating layers (known as **geoms**) and plotting them over each other.

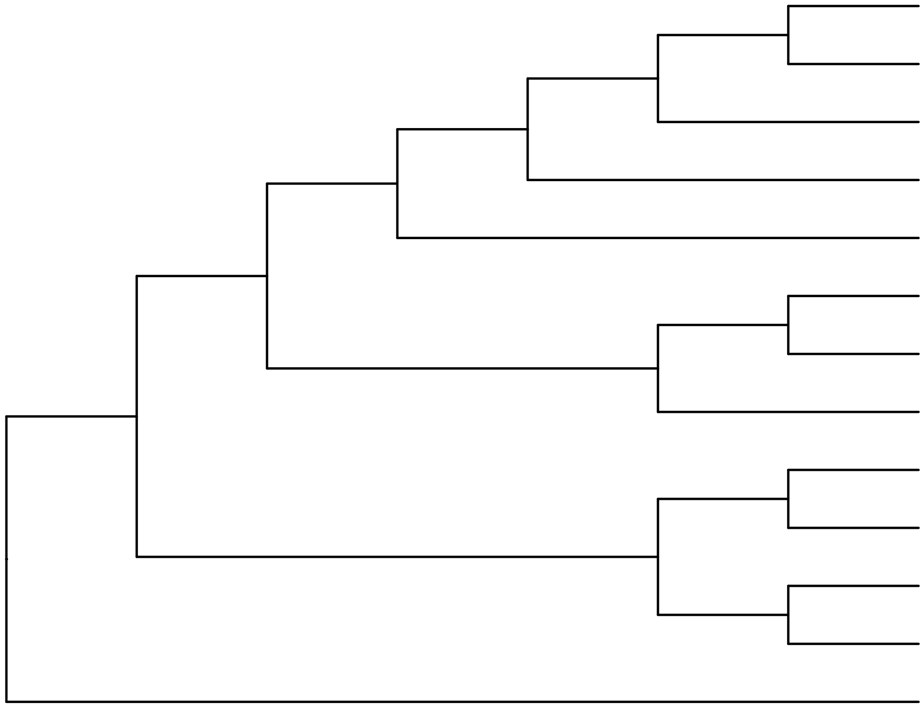
We'll start by using ggtree to plot our tree. This is the base layer of the plot.

```
ggtree(tree)
```



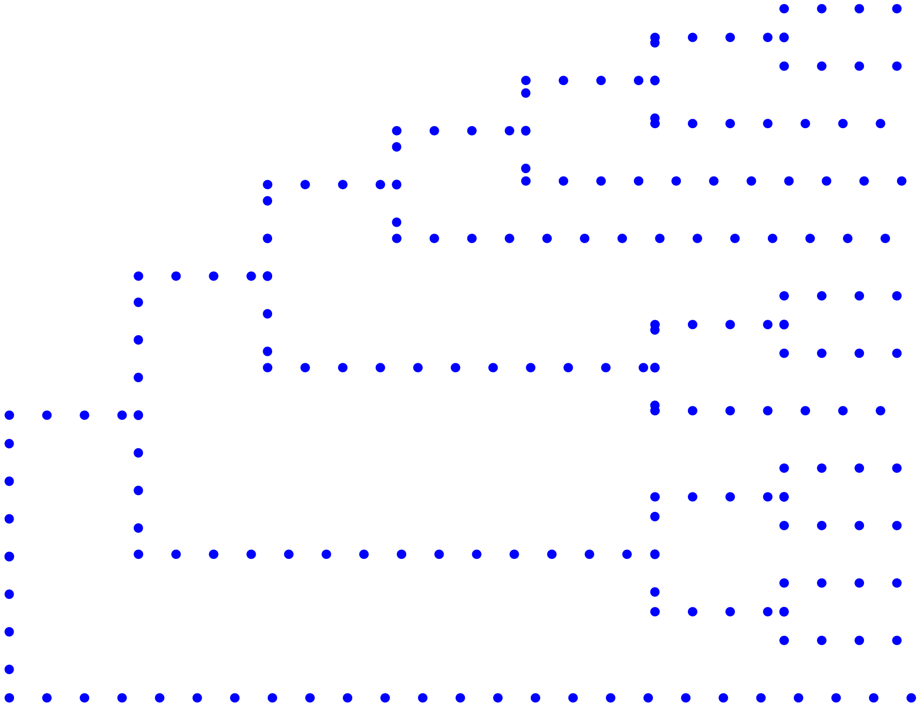
You may want to plot the same tree as a **cladogram**. To do this, disable branch lengths.

```r
ggtree(tree, branch.length = "none")
```



There are many other options we can include to customise our tree.

```
ggtree(tree,
       branch.length="none",
       color="blue",
       size=2,
       linetype=3)
```



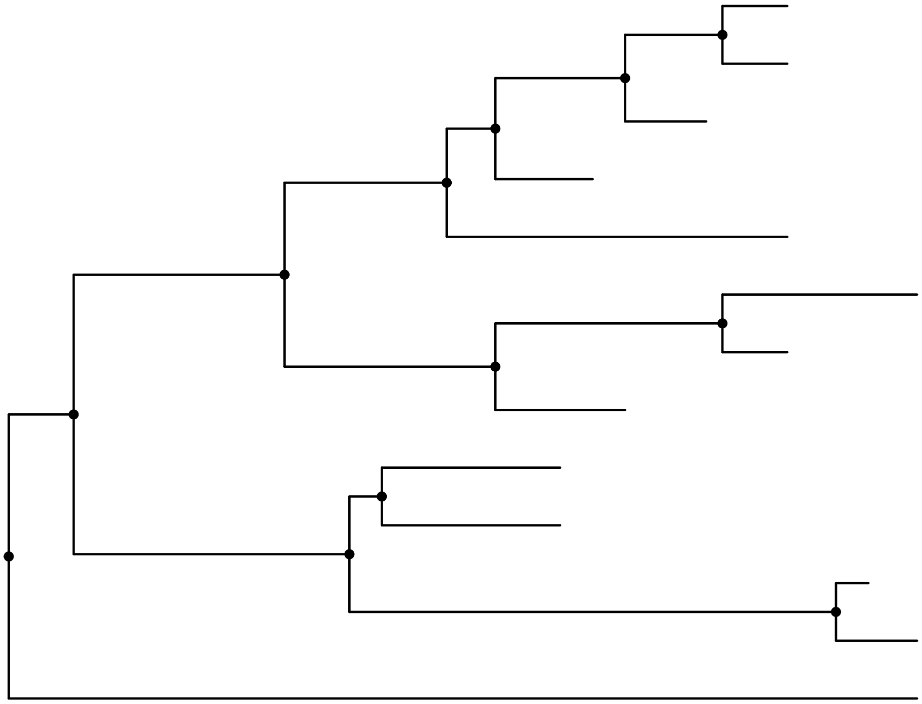
3.3.1 Geoms

Geoms are new layers to plot on or alongside your tree. Here I'm creating the plot as an object in R. You should see "p" appear in your environment but no plot will appear.

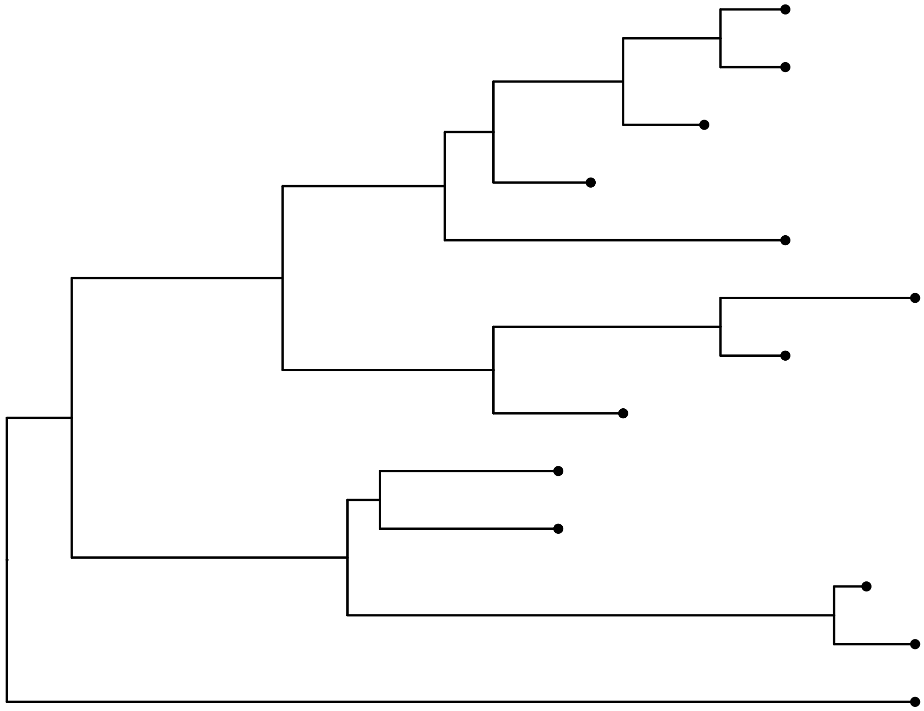
```r
p <- ggtree(tree)
```

Now let's plot it whilst adding new layers. Note that the hash denotes text not to be interpreted by R. This is a great way to annotate your code so that you can recall what it does!

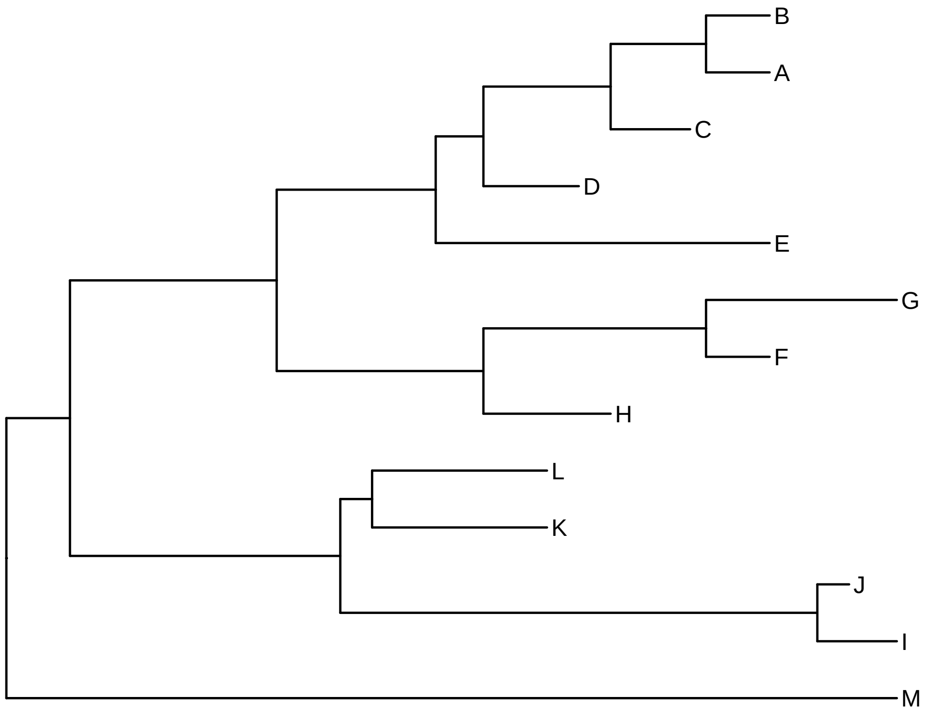
```r
p + geom_nodepoint() #Add node points
```



```
p + geom_tippoint() # add tip points
```



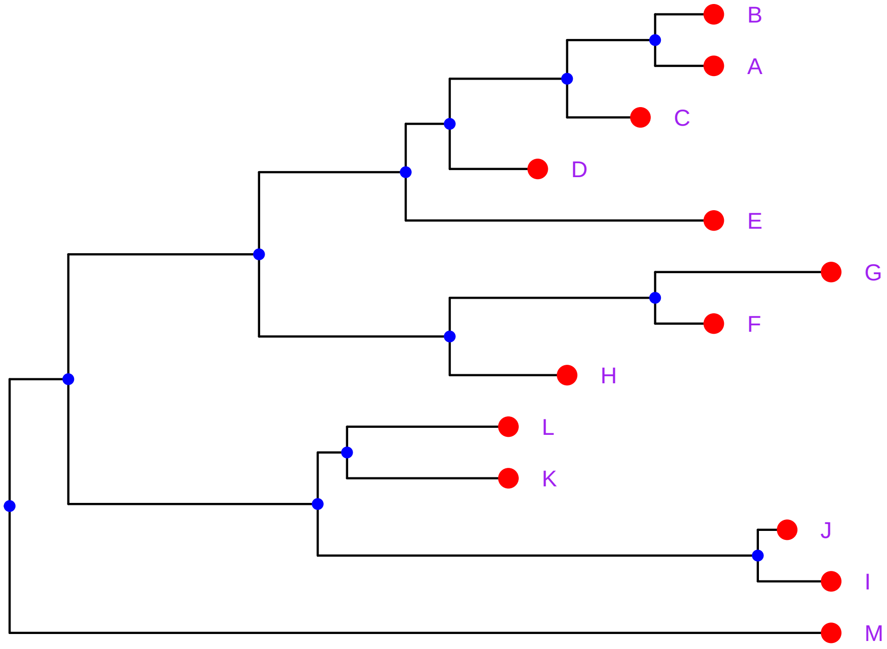
```
p + geom_tiplab() # Label the tips
```



These geoms can be combined as you see fit. This gives you a lot of flexibility in how you plot your trees.

```
p +
  geom_tiplab(offset = 2, color = "purple") +
  geom_nodepoint(color = "blue", size = 2) +
  geom_tippoint(color = "red", size = 4) +
  ggtitle("A phylogeny of letters. For some reason...")
```

A phylogeny of letters. For some reason...



3.3.2 Labelling clades

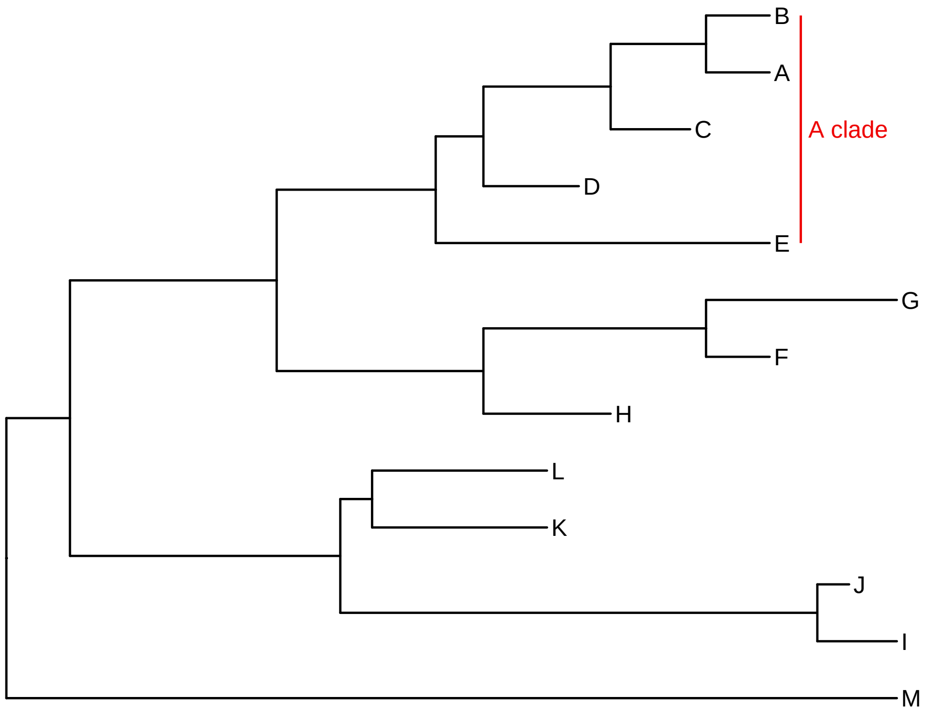
To label clades, we need to be able to identify the node of the most recent common ancestor. The function **MRCA** tells us that the common ancestor of C and E is node 17.

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

```
[1] 17
```

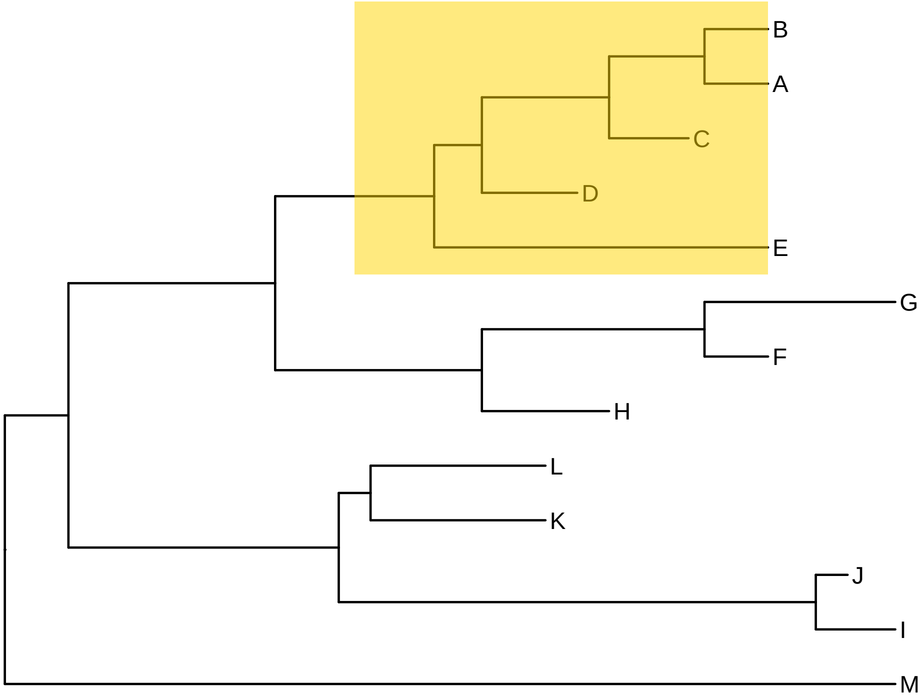
Let's use a new geom to label the clade.

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



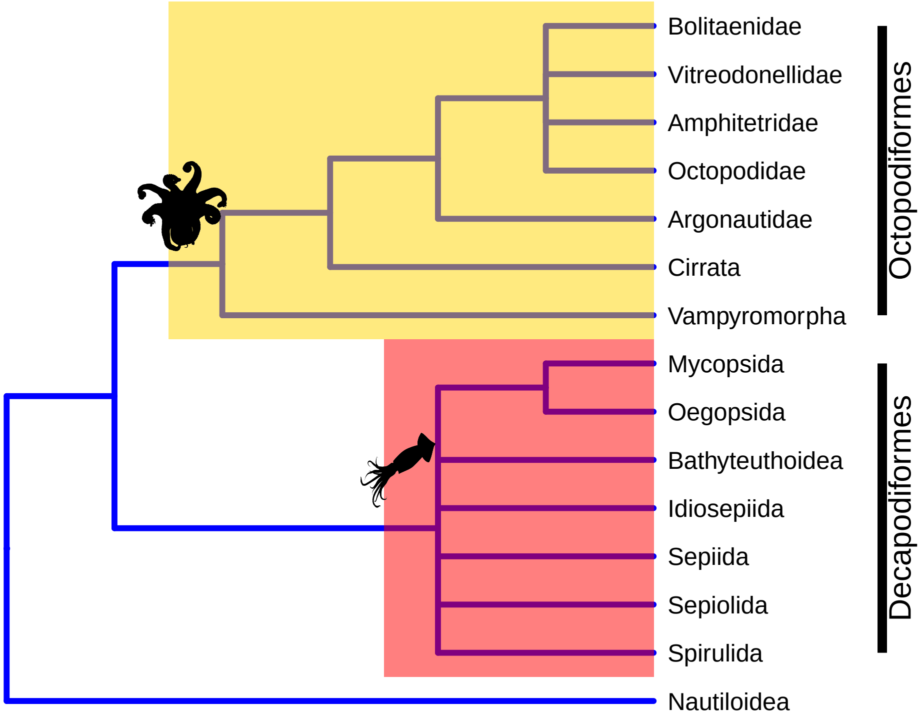
Pretty good. But there are other options. Again it's a matter of personal preference. You may prefer to overlay a translucent rectangle over your clade of interest.

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



3.4 Example Phylogeny

Let's now have a look at how we can include images on our plots. Using images is a great way to annotate a phylogeny. Here's the kind of thing I mean.


Bolitaenidae
Vitreodonellidae
Amphitetridae
Octopodidae
Argonautidae
Cirrata
Vampyromorpha
Mycopsida
Oegopsida
Bathyteuthoidea
Idiosepiida
Sepiida
Sepiolida
Spirulida
Nautiloidea
Octopodiformes
Decapodiformes

This phylogeny is annotated in a number of useful ways. The tip labels are the most common type you probably recognise and in this case, describe cephalopod families. The superorders (octopodiformes and decapodiformes) are highlighted by gold and red rectangles as well as a bar across the tips.

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

3.4.1 Phylopic

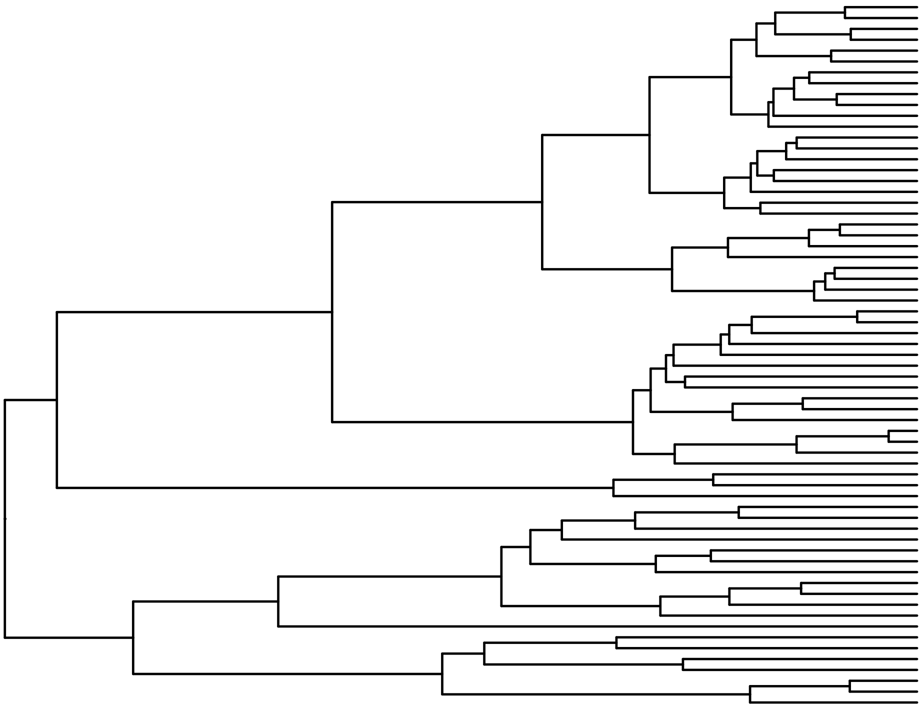
The silhouettes I used for that plot are from a website (http://phylopic.org/). Phylopic provides open source biological silhouettes that are free to use. We're now going to look at how to do this.

Let's start with loading an example tree. This one is a primate tree courtesy of Randi Griffin. You'll notice that I'm loading this tree using a url. This is because I'm loading a file directly from GitHub, a sort of social network of coding and the host of this site! Randi (and many other coders) make some of the things they produce freely available through GitHub. This can be data, files or code.

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

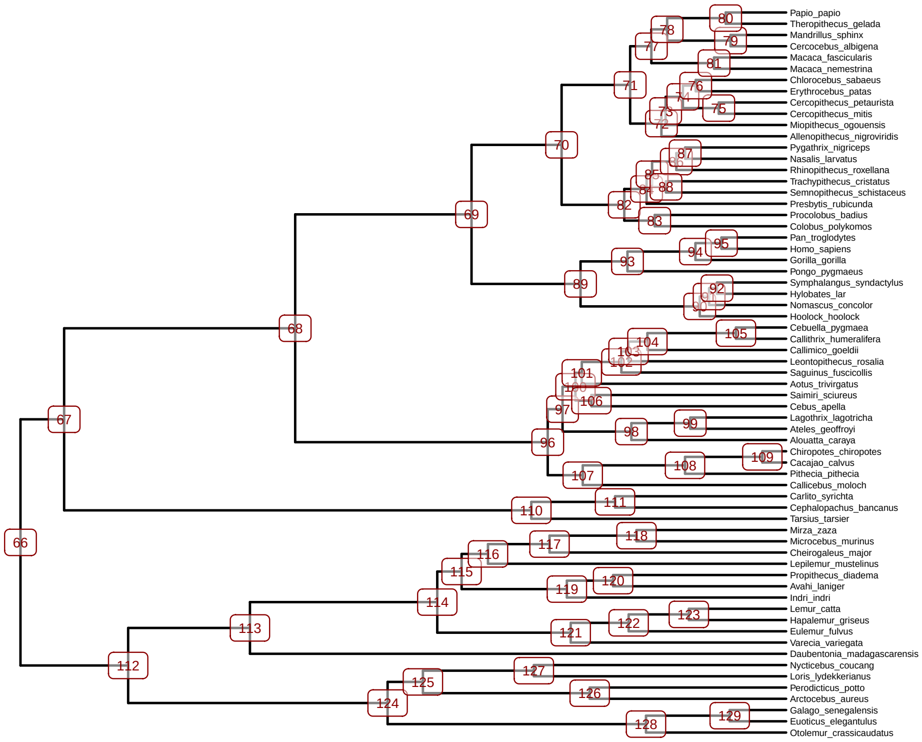
Let's plot the new tree first.

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



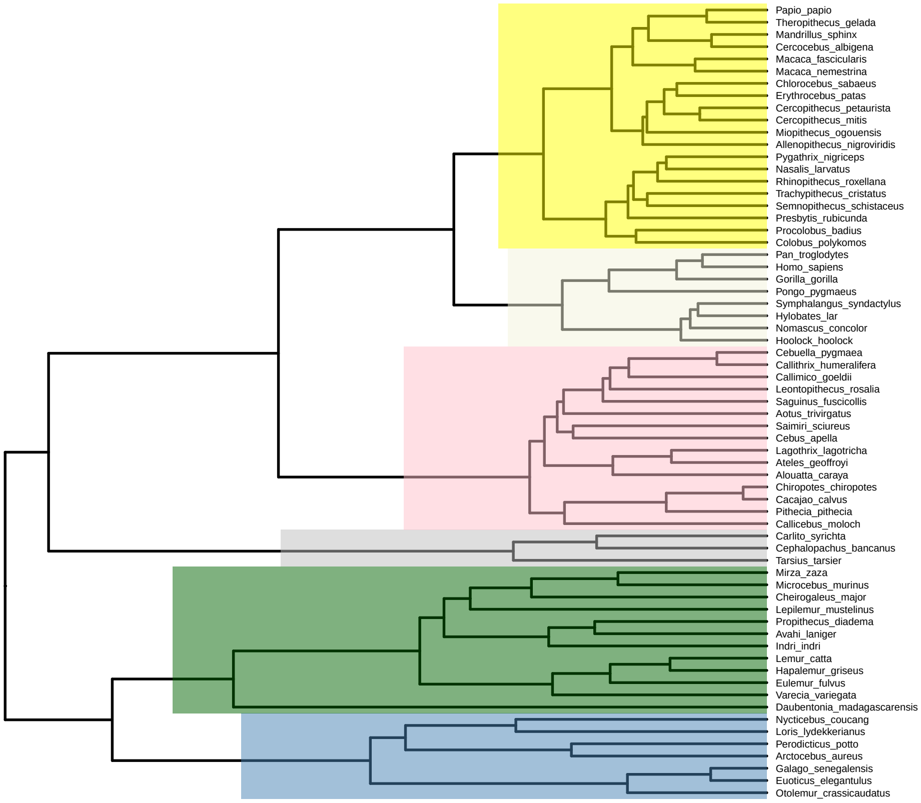
Let's use what we know about ggtree to customise this plot into something more useful. In particular, this plot is quite useful because it tells us the numbers of each node and we will need that later on.

```
p2 <- ggtree(primates) +
  xlim(0,90) +
  geom_tiplab(size=1.5) +
  geom_label2(aes(subset=!isTip, label=node), size=2, color="darkred", alpha=0.5)
p2
```


Papio_papio
Theropithecus_gelada
Mandrillus_sphinx
Cercocebus_albigena
Macaca_fascicularis
Macaca_nemestrina
Chlorocebus_sabaeus
Erythrocebus_patas
Cercopithecus_petaurista
Cercopithecus_mitis
Miopithecus_ogouensis
Allenopithecus_nigroviridis
Pygathrix_nigriceps
Nasalis_larvatus
Rhinopithecus_roxellana
Trachypithecus_cristatus
Semnopithecus_schistaceus
Presbytis_rubicunda
Procolobus_badius
Colobus_polykomos
Pan_troglodytes
Homo_sapiens
Gorilla_gorilla
Pongo_pygmaeus
Symphalangus_syndactylus
Hylobates_lar
Nomascus_concolor
Hoolock_hoolock
Cebuella_pygmaea
Callithrix_humeralifera
Callimico_goeldii
Leontopithecus_rosalia
Saguinus_fuscicollis
Aotus_trivirgatus
Saimiri_sciureus
Cebus_apella
Lagothrix_lagotricha
Ateles_geoffroyi
Alouatta_caraya
Chiropotes_chiropotes
Cacajao_calvus
Pithecia_pithecia
Callicebus_moloch
Carlito_syrichta
Cephalopachus_bancanus
Tarsius_tarsier
Mirza_zaza
Microcebus_murinus
Cheirogaleus_major
Lepilemur_mustelinus
Propithecus_diadema
Avahi_laniger
Indri_indri
Lemur_catta
Hapalemur_griseus
Eulemur_fulvus
Varecia_variegata
Daubentonia_madagascarensis
Nycticebus_coucang
Loris_lydekkerianus
Perodicticus_potto
Arctocebus_aureus
Galago_senegalensis
Euoticus_elegantulus
Otolemur_crassicaudatus
80
78
79
77
81
71
76
74
75
73
72
87
85
88
84
82
83
70
69
95
94
93
92
90
89
68
105
104
103
102
101
106
97
99
98
96
109
108
107
111
110
67
118
117
116
115
120
119
114
123
122
121
113
66
112
127
125
126
124
129
128

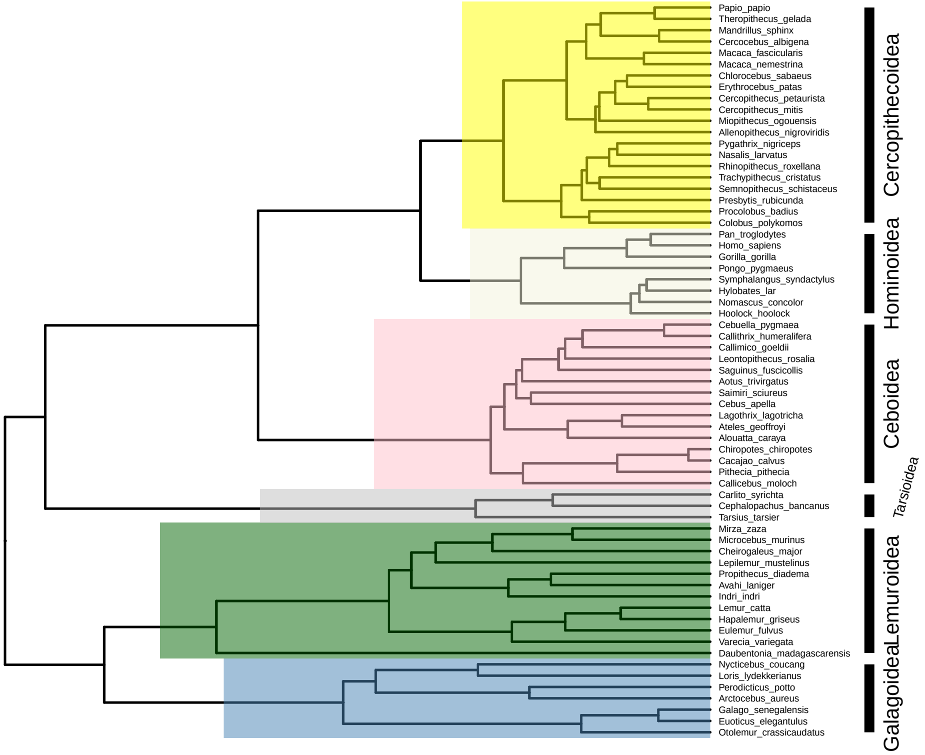
Let's label the 6 primate superfamilies.

```r
p3 <- ggtree(primates) +
  xlim(0,100) +
  geom_tiplab(size=1.5, offset=0.5) +
  geom_hilight(node=124, fill="steelblue", alpha=0.5) +
  geom_hilight(node=113, fill="darkgreen", alpha=0.5) +
  geom_hilight(node=110, fill="gray", alpha=0.5) +
  geom_hilight(node=96, fill="pink", alpha=0.5) +
  geom_hilight(node=89, fill="beige", alpha=0.5) +
  geom_hilight(node=70, fill="yellow", alpha=0.5)
p3
```


Papio_papio
Theropithecus_gelada
Mandrillus_sphinx
Cercocebus_albigena
Macaca_fascicularis
Macaca_nemestrina
Chlorocebus_sabaeus
Erythrocebus_patas
Cercopithecus_petaurista
Cercopithecus_mitis
Miopithecus_ogouensis
Allenopithecus_nigroviridis
Pygathrix_nigriceps
Nasalis_larvatus
Rhinopithecus_roxellana
Trachypithecus_cristatus
Semnopithecus_schistaceus
Presbytis_rubicunda
Procolobus_badius
Colobus_polykomos
Pan_troglodytes
Homo_sapiens
Gorilla_gorilla
Pongo_pygmaeus
Symphalangus_syndactylus
Hylobates_lar
Nomascus_concolor
Hoolock_hoolock
Cebuella_pygmaea
Callithrix_humeralifera
Callimico_goeldii
Leontopithecus_rosalia
Saguinus_fuscicollis
Aotus_trivirgatus
Saimiri_sciureus
Cebus_apella
Lagothrix_lagotricha
Ateles_geoffroyi
Alouatta_caraya
Chiropotes_chiropotes
Cacajao_calvus
Pithecia_pithecia
Callicebus_moloch
Carlito_syrichta
Cephalopachus_bancanus
Tarsius_tarsier
Mirza_zaza
Microcebus_murinus
Cheirogaleus_major
Lepilemur_mustelinus
Propithecus_diadema
Avahi_laniger
Indri_indri
Lemur_catta
Hapalemur_griseus
Eulemur_fulvus
Varecia_variegata
Daubentonia_madagascarensis
Nycticebus_coucang
Loris_lydekkerianus
Perodicticus_potto
Arctocebus_aureus
Galago_senegalensis
Euoticus_elegantulus
Otolemur_crassicaudatus

So far so good. Let's add on bars like I did for the cephalopod version. This time, I'll add the new details to the object p3 to save some typing.

```r
p4 <- p3 +
  geom_cladelabel(124, "Galagoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(113, "Lemuroidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(110, "Tarsioidea", offset=15, barsize=2, angle=75,
                  offset.text=2.5, hjust=0.2, fontsize=2) +
  geom_cladelabel(96, "Ceboidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(89, "Hominoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3) +
  geom_cladelabel(70, "Cercopithecoidea", offset=15, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=3)
p4
```


Papio_papio
Theropithecus_gelada
Mandrillus_sphinx
Cercocebus_albigena
Macaca_fascicularis
Macaca_nemestrina
Chlorocebus_sabaeus
Erythrocebus_patas
Cercopithecus_petaurista
Cercopithecus_mitis
Miopithecus_ogouensis
Allenopithecus_nigroviridis
Pygathrix_nigriceps
Nasalis_larvatus
Rhinopithecus_roxellana
Trachypithecus_cristatus
Semnopithecus_schistaceus
Presbytis_rubicunda
Procolobus_badius
Colobus_polykomos
Pan_troglodytes
Homo_sapiens
Gorilla_gorilla
Pongo_pygmaeus
Symphalangus_syndactylus
Hylobates_lar
Nomascus_concolor
Hoolock_hoolock
Cebuella_pygmaea
Callithrix_humeralifera
Callimico_goeldii
Leontopithecus_rosalia
Saguinus_fuscicollis
Aotus_trivirgatus
Saimiri_sciureus
Cebus_apella
Lagothrix_lagotricha
Ateles_geoffroyi
Alouatta_caraya
Chiropotes_chiropotes
Cacajao_calvus
Pithecia_pithecia
Callicebus_moloch
Carlito_syrichta
Cephalopachus_bancanus
Tarsius_tarsier
Mirza_zaza
Microcebus_murinus
Cheirogaleus_major
Lepilemur_mustelinus
Propithecus_diadema
Avahi_laniger
Indri_indri
Lemur_catta
Hapalemur_griseus
Eulemur_fulvus
Varecia_variegata
Daubentonia_madagascarensis
Nycticebus_coucang
Loris_lydekkerianus
Perodicticus_potto
Arctocebus_aureus
Galago_senegalensis
Euoticus_elegantulus
Otolemur_crassicaudatus
Cercopithecoidea
Hominoidea
Ceboidea
Tarsioidea
Lemuroidea
Galagoidea

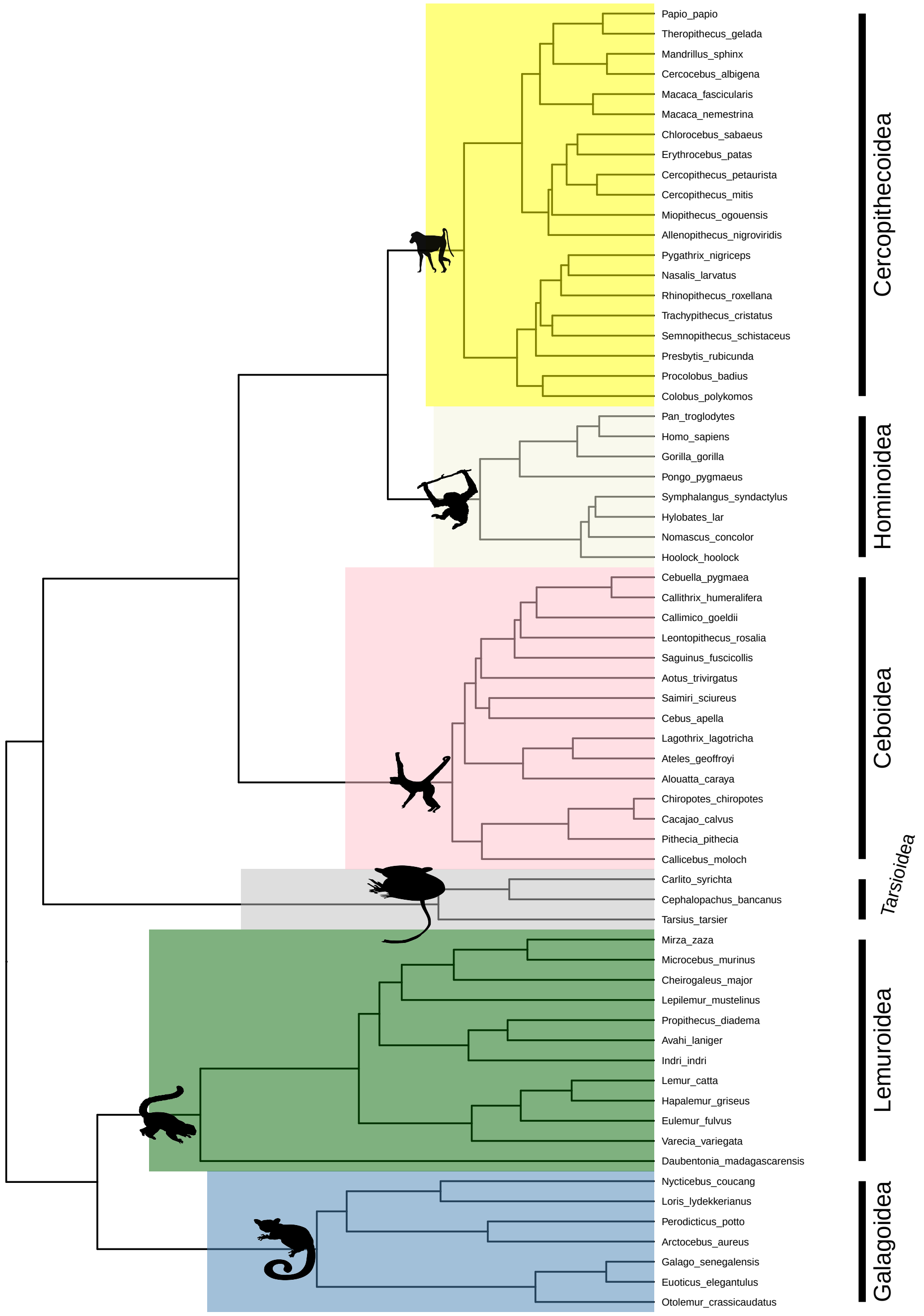
There are some helpful details here, such as the fact that the label for Tarsioidea is off at an angle to avoid overlapping with other labels. The extra arguments in these options demonstrate how much control you can exercise over each geom.

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

```
images <- data.frame(node = c(124,113,110,96,89,70),
                     phylopic = c("http://phylopic.org/assets/images/submissions/
                                  7fb9bea8-e758-4986-afb2-95a2c3bf983d.512.png",
                                  "http://phylopic.org/assets/images/submissions/
                                  bac25f49-97a4-4aec-beb6-f542158ebd23.512.png",
                                  "http://phylopic.org/assets/images/submissions/
                                  f598fb39-facf-43ea-a576-1861304b2fe4.512.png",
                                  "http://phylopic.org/assets/images/submissions/
                                  aceb287d-84cf-46f1-868c-4797c4ac54a8.512.png",
                                  "http://phylopic.org/assets/images/submissions/
                                  0174801d-15a6-4668-bfe0-4c421fbe51e8.512.png",
                                  "http://phylopic.org/assets/images/submissions/
                                  72f2f854-f3cd-4666-887c-35d5c256ab0f.512.png"),
                     species = c("Galagoidea","Lemuroidea","Tarsioidea",
                                 "Ceboidea","Hominoidea","Cercopithecoidea"))
```

This is a way of plotting them all at once with all the code included to build the plot from scratch.

```
p5 <- ggtree(primates) +
  xlim(0,110) +
  geom_tiplab(size=2, offset=0.5) +
  geom_hilight(node=124, fill="steelblue", alpha=0.5) +
  geom_hilight(node=113, fill="darkgreen", alpha=0.5) +
  geom_hilight(node=110, fill="gray", alpha=0.5) +
  geom_hilight(node=96, fill="pink", alpha=0.5) +
  geom_hilight(node=89, fill="beige", alpha=0.5) +
  geom_hilight(node=70, fill="yellow", alpha=0.5) +
  geom_cladelabel(124, "Galagoidea", offset=22, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=5) +
  geom_cladelabel(113, "Lemuroidea", offset=22, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=5) +
  geom_cladelabel(110, "Tarsioidea", offset=22, barsize=2, angle=75,
                  offset.text=2.5, hjust=0.2, fontsize=4) +
  geom_cladelabel(96, "Ceboidea", offset=22, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=5) +
  geom_cladelabel(89, "Hominoidea", offset=22, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=5) +
  geom_cladelabel(70, "Cercopithecoidea", offset=22, barsize=2, angle=90,
                  offset.text=1.5, hjust=0.5, fontsize=5)

p5 %<+% images +
  geom_nodelab(aes(image = phylopic), geom = "image",
               size = .04, nudge_x = -4)
```


Papio_papio
Theropithecus_gelada
Mandrillus_sphinx
Cercocebus_albigena
Macaca_fascicularis
Macaca_nemestrina
Chlorocebus_sabaeus
Erythrocebus_patas
Cercopithecus_petaurista
Cercopithecus_mitis
Miopithecus_ogouensis
Allenopithecus_nigroviridis
Pygathrix_nigriceps
Nasalis_larvatus
Rhinopithecus_roxellana
Trachypithecus_cristatus
Semnopithecus_schistaceus
Presbytis_rubicunda
Procolobus_badius
Colobus_polykomos
Pan_troglodytes
Homo_sapiens
Gorilla_gorilla
Pongo_pygmaeus
Symphalangus_syndactylus
Hylobates_lar
Nomascus_concolor
Hoolock_hoolock
Cebuella_pygmaea
Callithrix_humeralifera
Callimico_goeldii
Leontopithecus_rosalia
Saguinus_fuscicollis
Aotus_trivirgatus
Saimiri_sciureus
Cebus_apella
Lagothrix_lagotricha
Ateles_geoffroyi
Alouatta_caraya
Chiropotes_chiropotes
Cacajao_calvus
Pithecia_pithecia
Callicebus_moloch
Carlito_syrichta
Cephalopachus_bancanus
Tarsius_tarsier
Mirza_zaza
Microcebus_murinus
Cheirogaleus_major
Lepilemur_mustelinus
Propithecus_diadema
Avahi_laniger
Indri_indri
Lemur_catta
Hapalemur_griseus
Eulemur_fulvus
Varecia_variegata
Daubentonia_madagascarensis
Nycticebus_coucang
Loris_lydekkerianus
Perodicticus_potto
Arctocebus_aureus
Galago_senegalensis
Euoticus_elegantulus
Otolemur_crassicaudatus
Cercopithecoidea
Hominoidea
Ceboidea
Tarsioidea
Lemuroidea
Galagoidea

Chapter 4

Workshop 2: PGLS

This tutorial will show you how to perform phylogenetically correct regression analyses on continuous data in R.

As usual, remember to set your working directory to wherever you have saved the necessary files. Other than that, this tutorial assumes you have already installed the packages ape and caper.

4.1 Data

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

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

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

4.2 Plotting with ggplot2

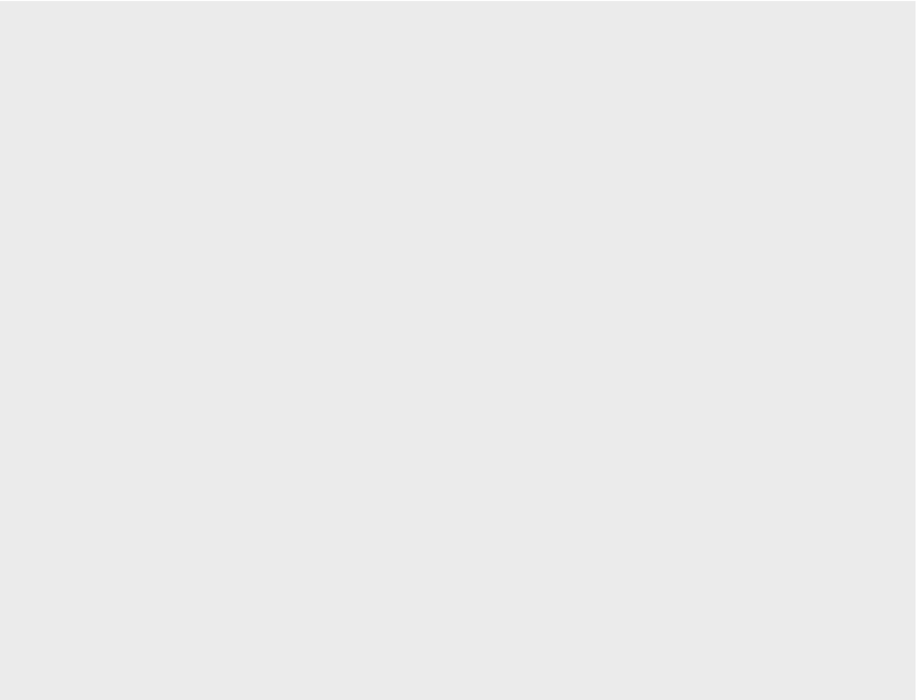
A nice way of investigating data is by plotting it. For a quick visualisation I usually use the base graphics in R. They get the job done and are realtively simple to edit once you understand the syntax of R.

However, most R users seem to agree that the package "ggplot2" gives better plots. This might be useful to you when you want to prepare your plots for reports. So here, I'm going to use ggplot2 just to show you what it can do.

```r
install.packages("ggplot2")
library(ggplot2)
```

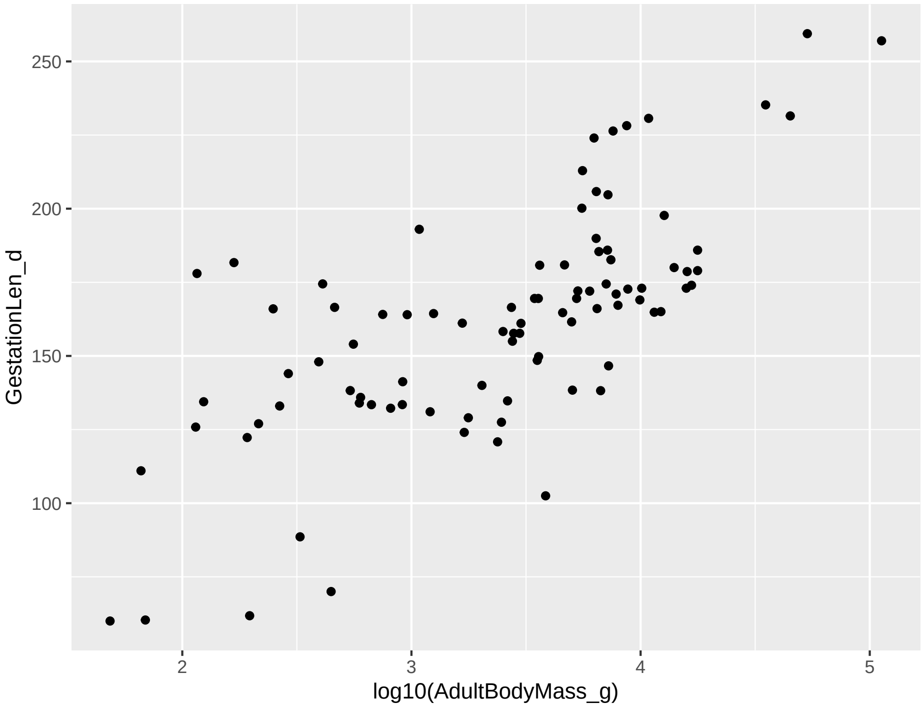
ggplot2 builds plots in layers similar to ggtree which you have met before in workshop 1. In fact, the ggtree package was built based upon ggplot2. We start with the function ggplot which creates a coordinate system to which we can add layers. This function alone just creates a blank plot

```r
ggplot(data = primate.data)
```



Let's add a layer with the points. We can add a layer of points with the function geom_point. This creates our basic scatterplot. Each geom function (the ones that add layers) takes a mapping argument which controls how the layer is mapped onto the plot. This argument usually takes the form seen below. Remember that body mass should be log transformed before we plot it!

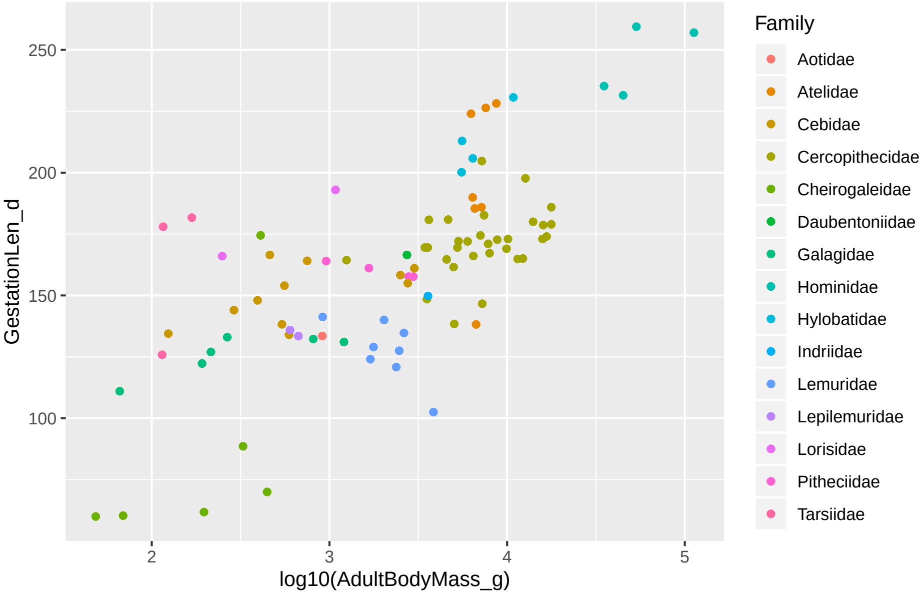
```r
ggplot(data = primate.data) +
  geom_point(mapping = aes(x = log10(AdultBodyMass_g), y = GestationLen_d))
```



In practice, exactly how you add geoms and layers seems to be affected by exactly what you want to do with the plot. As we'll see later, if you want to add regression lines or other statistical features to your plot, you may want to reorganise this code. My advice is to google what you want to do and work backwards from someone else's code. That's what I do.

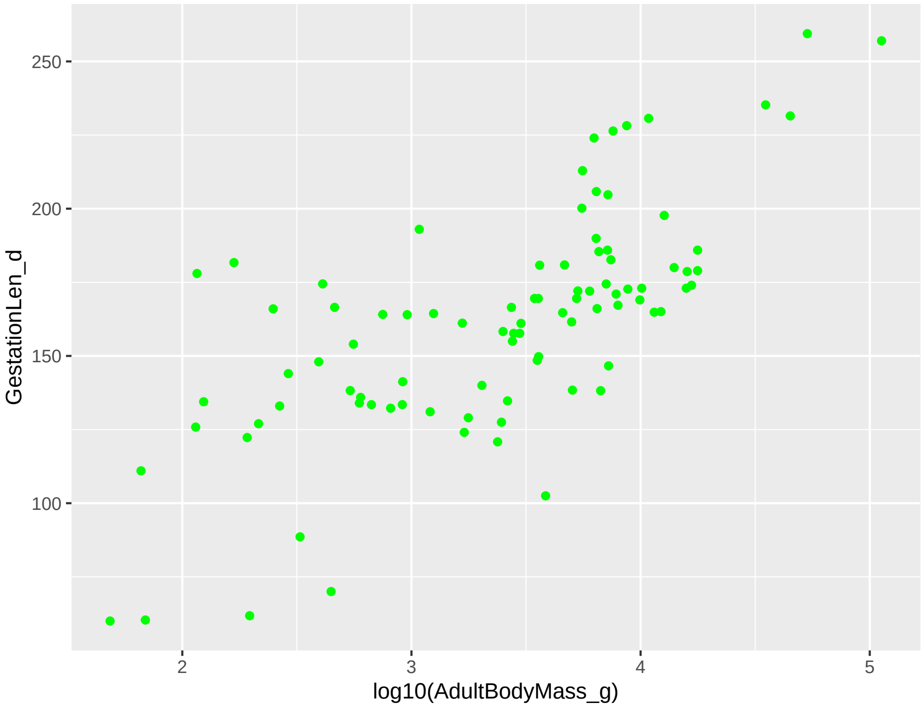
The plot isn't done yet. We can display a bit more information by adding another mapping argument. Here, I've coloured the points according to their family. The unique colours are assigned auotmatically and the legend is also created automatically.

```
ggplot(data = primate.data) +
  geom_point(mapping = aes(x = log10(AdultBodyMass_g), y = GestationLen_d,
                           colour = Family))
```



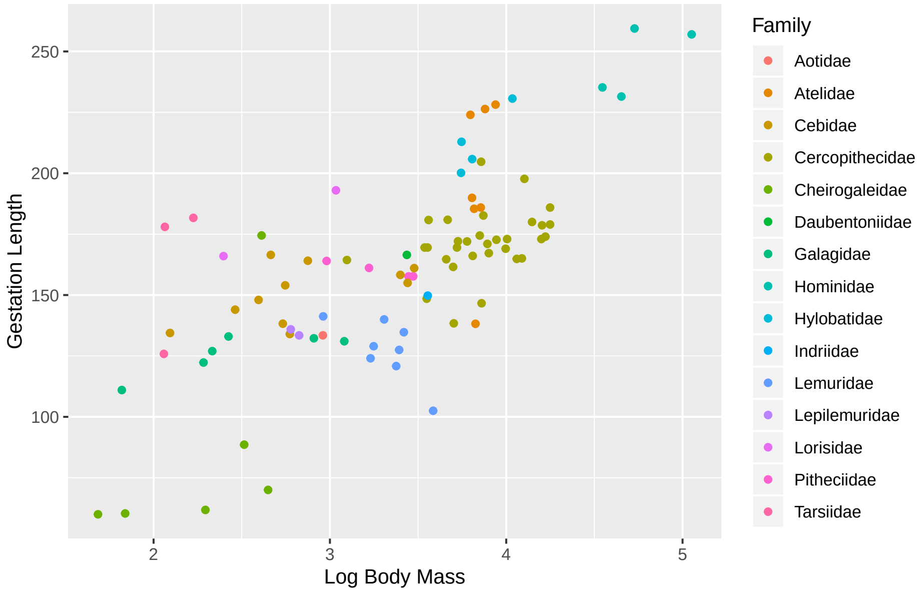
If you prefer, you can manually set the colour for all points. Be aware of the difference here! In the code below, the difference is that colour here is outside of the mapping argument. This means that the colour doesn't display any real information but does change the aesthetic of the plot.

```
ggplot(data = primate.data) +
  geom_point(mapping = aes(x = log10(AdultBodyMass_g), y = GestationLen_d),
             colour = "green")
```



Let's clean up the plot with some axis labels.

```r
ggplot(data = primate.data) +
  geom_point(mapping = aes(x = log10(AdultBodyMass_g), y = GestationLen_d,
                           colour = Family)) +
  labs(x = "Log Body Mass", y = "Gestation Length")
```



4.3 Linear Regression

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

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

You should now see a new object (m1) in your environment. This contains the output of your regression analysis and to view it, we can use the summary function.

```
summary(m1)
```

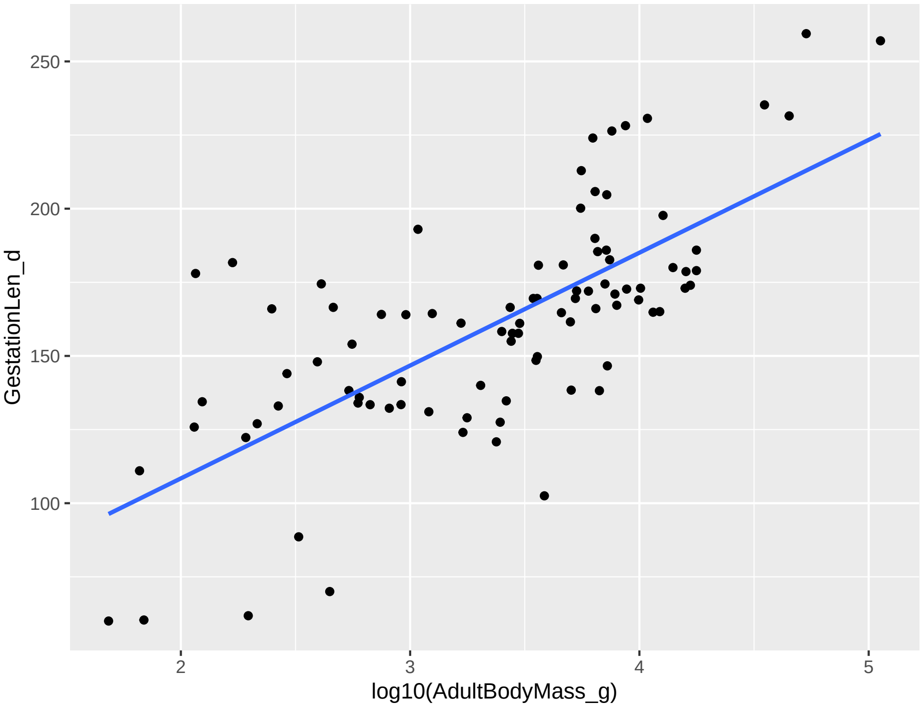
```

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

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

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)              31.775     13.927   2.281   0.0249 *  
log10(AdultBodyMass_g)   38.319      4.031   9.505 3.37e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

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

We can also plot this line with ggplot with the following code. There are some key differences here. Most importantly, I've specified x and y in ggplot rather than geom_point. I've then added the function geom_smooth amd used the method "lm" to specify that I want a linear model plotted. The reason I did this is quite simple; I Googled it and the hive-mind of R users on the internet told me to.

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



4.4 Trees and Phylogenetic Signal

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

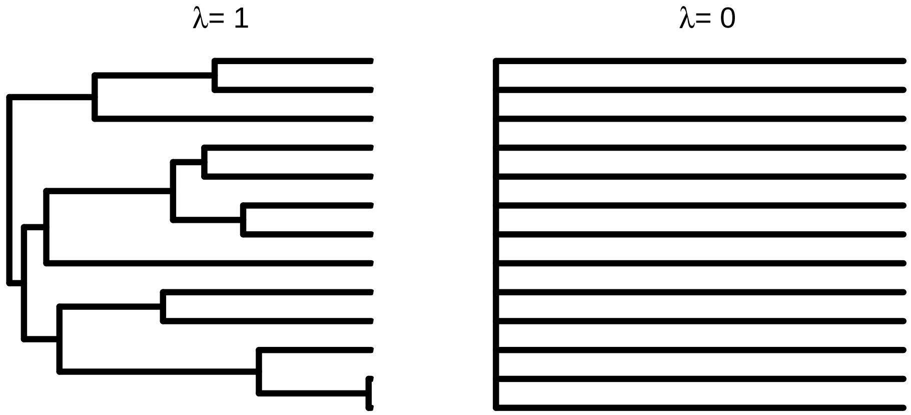
4.4.1 Phylogenetic signal

Importantly, you need to understand the concept of phylogenetic signal which is defined as *the tendency for closely related species to resemble each other more than distantly related species.*

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

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

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



4.4.2 caper

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

```
library(caper)
```

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

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

The regression command in caper (along with some other functions) requires the data and tree to be combined in a comparative data object. This type of objectis simply a tree and comparative data set concatenated and is created using the function "comparative.data". We need to specify the tree object, data object, column name in the data where species names are stored and whether we want a variance-covariance matrix included (we do).

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

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

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

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

```
str(primates)
```

```
List of 7
 $ phy      :List of 5
  ..$ edge       : int [1:163, 1:2] 84 85 86 87 88 89 90 91 92 93 ...
  ..$ edge.length: num [1:163] 4.95 17.69 19.65 8.12 4.82 ...
  ..$ Nnode      : int 81
  ..$ tip.label  : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopi
  ..$ node.label : int [1:81] 84 85 86 87 88 89 90 91 92 93 ...
  ..- attr(*, "class")= chr "phylo"
  ..- attr(*, "order")= chr "cladewise"
 $ data     :'data.frame':  83 obs. of  7 variables:
  ..$ Order          : Factor w/ 1 level "Primates": 1 1 1 1 1 1 1 1 1 1 ...
  ..$ Family         : Factor w/ 15 levels "Aotidae","Atelidae",..: 4 4 4 4 4 4 4 5 5
  ..$ AdultBodyMass_g: num [1:83] 3540 3445 5041 5325 5257 ...
  ..$ GestationLen_d : num [1:83] 148 170 138 172 170 ...
  ..$ HomeRange_km2  : num [1:83] 0.16 0.24 0.1 0.06 1.15 ...
  ..$ MaxLongevity_m : num [1:83] 340 276 325 316 276 ...
  ..$ SocialGroupSize: num [1:83] 26.3 11 16 4.5 16 28 91.2 1 1 1 ...
 $ data.name: chr "primate.data"
 $ phy.name : chr "primate.tree"
 $ dropped  :List of 2
  ..$ tips          : chr [1:143] "Allenopithecus_nigroviridis" "Cercopithecus_cephus_
  ..$ unmatched.rows: chr [1:8] "Cercopithecus_campbelli" "Cercopithecus_pogonias" "Ch
 $ vcv      : 'VCV.array' num [1:83, 1:83] 72.3 69 64 62.4 64 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_m
  .. ..$ : chr [1:83] "Cercopithecus_ascanius" "Cercopithecus_cephus" "Cercopithecus_m
 $ vcv.dim  : num 2
 - attr(*, "class")= chr "comparative.data"
```

4.4.3 Estimating Phylogenetic Signal

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

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

The object "signal" should have appeared in your environment now. We can inspect the object using the ever-useful summary function.

```
summary(signal)
```

```

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

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

Branch length transformations:

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

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

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

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

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

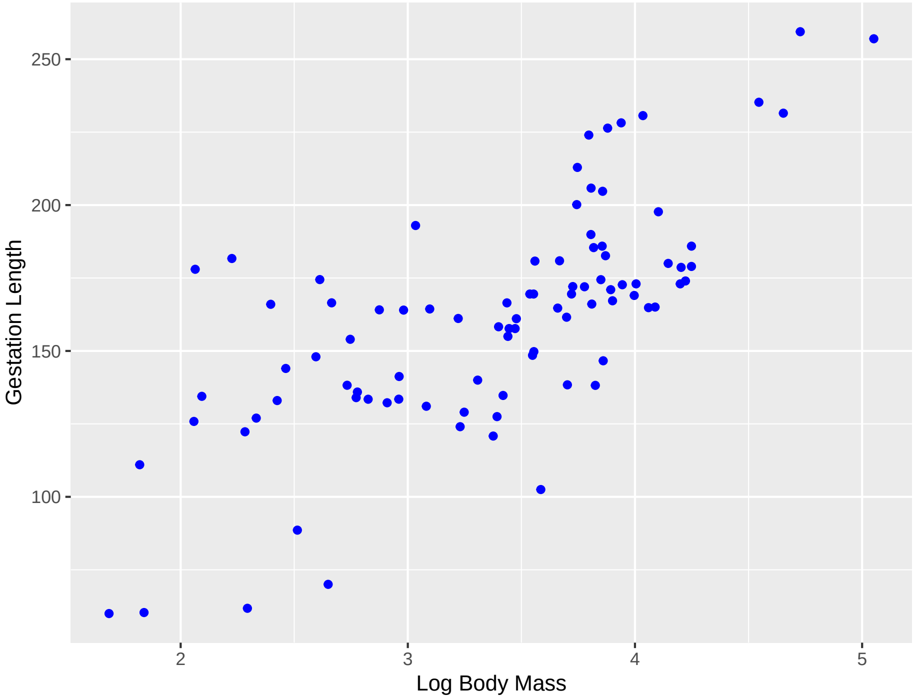
We also have lower bound and upper bound tests. We can see that λ is significantly different from the lower bound of 0 ($p < 2.2 \times 10^{-16}$). This isn't surprising at all!

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

4.5 Phylogenetic Regression

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

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



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

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

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

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

The summary function gets the key details we need.

```
summary(m2)
```

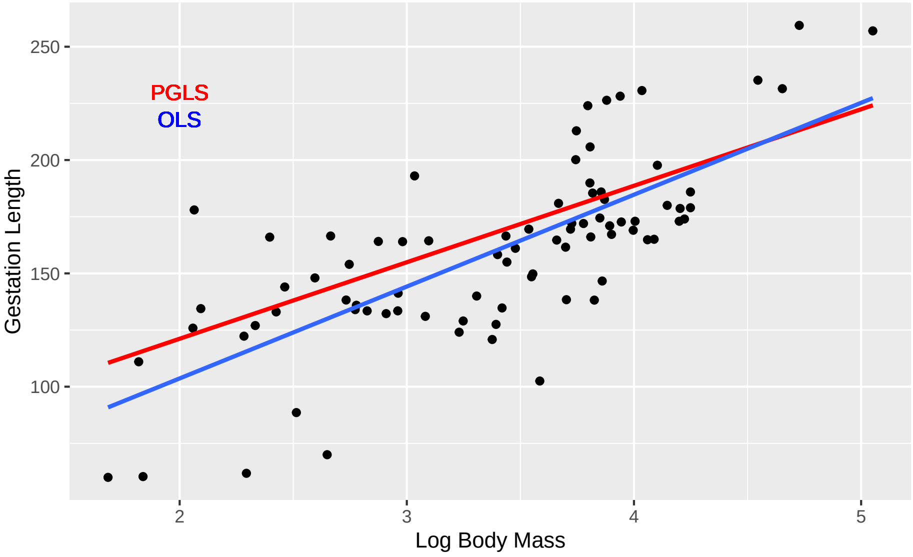
```

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

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

Branch length transformations:

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

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

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

As you can see, our model is a significant fit to the data ($F = 33.3$, $R^2 = 0.29$, p

= 1.39 x 10^{-7}). More importantly, We've confirmed that body size has a positive effect on gestation length ($\beta = 33.75$, s.e. = 5.85, p = 1.39 x 10^{-7}). Time to plot!

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

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

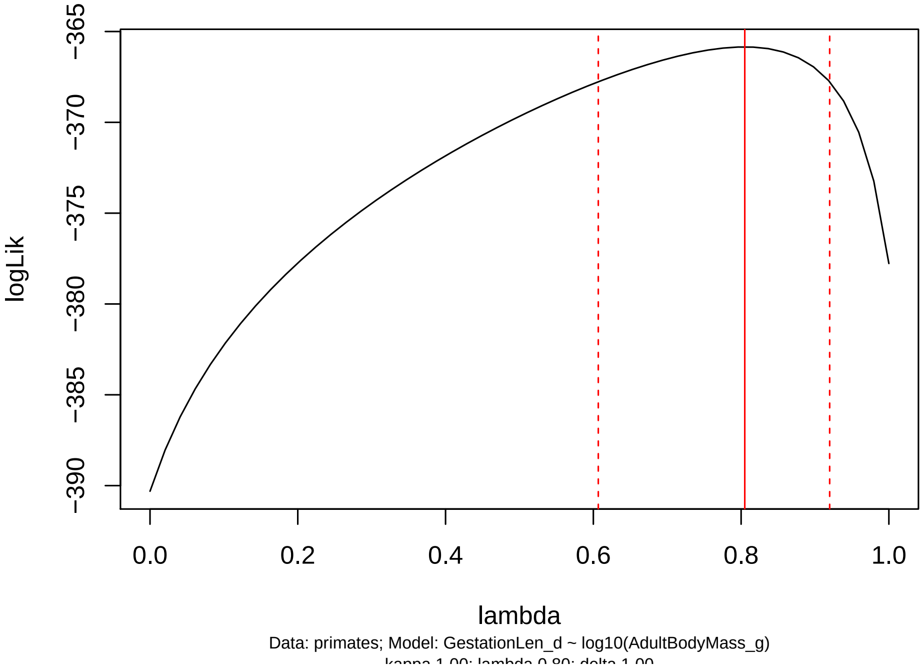


4.5.1 Model Checking

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

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

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



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

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

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

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

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

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

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

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

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

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

```

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

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

Branch length transformations:

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

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

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

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

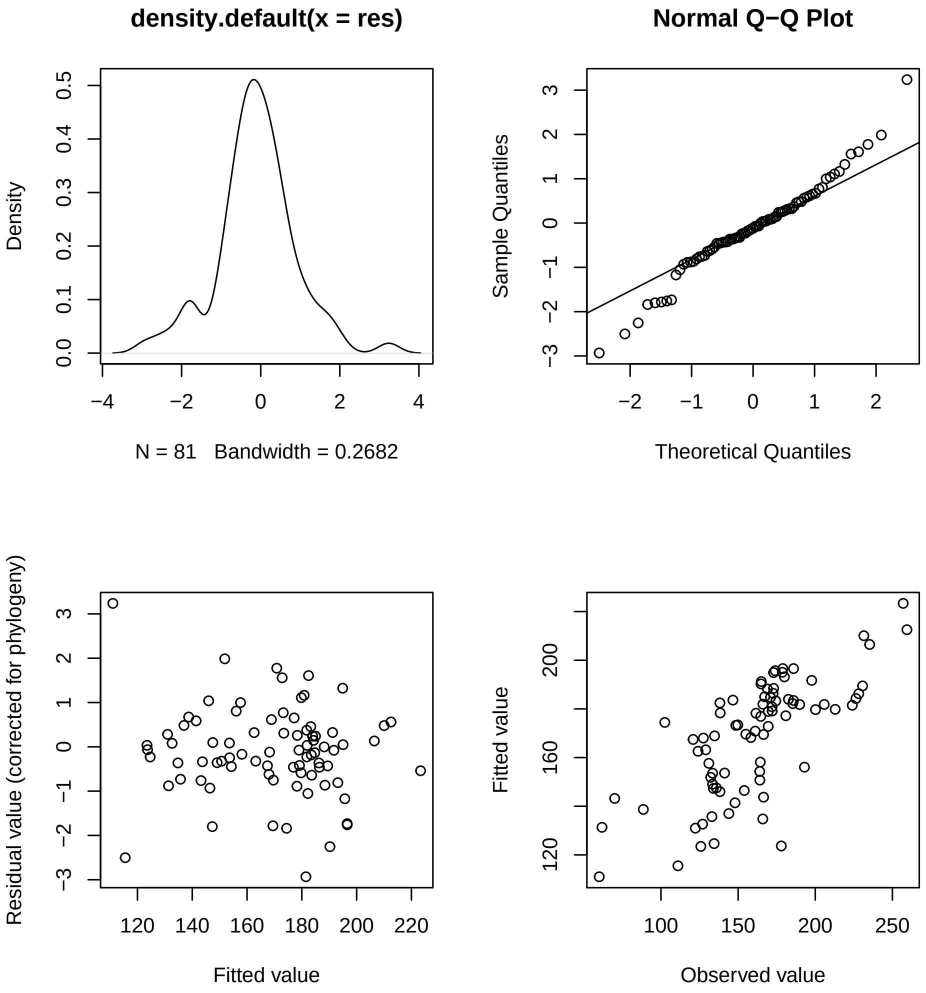
```
res <- residuals(m3, phylo = TRUE)
res <- res/sqrt(var(res))[1]
rownames(res) <- rownames(m3$residuals)
rownames(res)[abs(res)>3]
```

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

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

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

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

plot(m3)
```



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

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

4.6 Conclusion

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

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

Drive/University of Liverpool/GitHub Stuff/bookdownCRG/Images/mouse lemur.bb