Phylogenetic Comparative Methods in R
Using phylogeny as a statistical fix

Natalie Cooper (ncooper@tcd.ie)

The aims of this problem set are to learn how to use R to answer phylogenetic comparative questions without panicking! I am assuming that people have little or no experience with R so more experienced users may want to skip quickly through the first few sections.

By the end of this problem set you should be able to:

1. Read your data and phylogeny into R
2. View and manipulate your data and phylogeny
3. Match taxa in your data with those in your phylogeny
4. Perform PGLS analyses using `caper`
5. Perform model diagnostics for PGLS
6. Estimate phylogenetic signal using λ and K

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.

Throughout, R code will be in shaded boxes:

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

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

> 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! This workshop will be full of different ways to do things.

Preparations

Downloading the data and finding the path of your folder

First you need to download all the files for this problem set into a folder somewhere on your computer, let's call it "RAnalyses2014" and pop it on the Desktop. We will use this folder throughout the problem set. You'll need to know what the **path** of the folder is. For example on my Windows machine, the path is:

```
C:/Users/Natalie/Desktop/RAnalyses2014
```

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.

> In Windows, paths usually include \ but R can't read these. It's easy to fix in your R code, just change any \ in the path to / or \\.

On my Mac the path is:

```
~/Desktop/RAnalyses2014
```

It's a bit trickier to find the path on a Mac, so ask if you need help. Note that the tilde ~ is a shorthand for /Users/Natalie.

Using a text editor

Next, open a text editor. R has an inbuilt editor that works pretty well, but NotePad and TextEdit are fine too. However, in the future 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.

You should type (or copy and paste) your code into the text editor, edit it until you think it'll work, and then paste it into R's console window. Saving the text file lets you keep a record of the code you used, which can be a great timesaver if you want to use it again, especially as you know this code will work!

> You can cut and paste code from this handout into your text editor, or straight into R. 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 can remind you what the code is doing.
# You need a hash sign at the start of each line of a comment.
```

Installing extra packages in R

To run comparative analyses (or any specialized analysis) in R, you need to download one or more additional packages from the basic R installation. For this problem set you will need to install the following packages: `ape`, `picante` and `caper`. To install the package `ape`:

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

Now install `picante` and `caper`.

Loading packages in R

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)
```

Don't forget to load `picante` and `caper` too!

Reading your data and phylogeny into R

Reading data into R

Next we need to load the data we are going to use for the analysis. 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). To save time I have given you a tab-delimited text file called "Primatedata.txt" which we are going to use. Load these data as follows.

> You will need to replace MYPATH with the name of the path to the folder containing the data and tree on your computer.

```
primatedata <- read.table("MYPATH/Primatedata.txt", sep = "\t",
                          header = TRUE)
```

Note that `sep = "\t"` indicates that you have a tab-delimited file, `sep = ","` would indicate a comma-delimited csv file. You can also use `read.delim` for tab delimited files or `read.csv` for comma delimited files. `header = TRUE`, indicates that the first line of the data contains column headings.

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  8 variables:
##  $ Order          : Factor w/ 1 level "Primates": 1 1 1 ...
##  $ Family         : Factor w/ 15 levels "Aotidae","Atelidae",...
##  $ Binomial       : Factor w/ 77 levels "Alouatta palliata",...
##  $ 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 ...
##  $ MaxLongevity_m : num  336 328 454 304 215 ...
##  $ SocialGroupSize: num  14.5 42 20 2.95 6.85 ...
```

This 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          6692.4          138.2
## 2 Primates    Atelidae   Ateles geoffroyi          7582.4          226.4
## 3 Primates    Atelidae    Ateles paniscus          8697.2          228.2
## 4 Primates Pitheciidae  Callicebus moloch           958.1          164.0
## 5 Primates     Cebidae  Callimico goeldii           558.0          154.0
## 6 Primates     Cebidae Callithrix jacchus           290.2          144.0
##   HomeRange_km2 MaxLongevity_m SocialGroupSize
## 1          2.28          336.0           14.50
## 2          0.73          327.6           42.00
## 3          1.36          453.6           20.00
## 4          0.02          303.6            2.95
## 5          0.32          214.8            6.85
## 6          0.02          201.6            8.55
```

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"
```

This gives you the names of the columns.

```
primatedata
```

This will print out all of the data!

Reading and displaying 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.

> You will need to replace MYPATH with the name of the path to the folder containing the data and tree on your computer.

```
primatetree <- read.nexus("MYPATH/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_cephus, ...
##
## Rooted; includes branch lengths.
```

```
str(primatetree)
```

```
## List of 4
##  $ edge       : int [1:446, 1:2] 227 228 229 230 231 232 ...
##  $ 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" ...
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

`primatetree` is a fully resolved tree with branch lengths. There are 226 species and 221 internal nodes. We can plot the tree by using the `plot` function of `ape`:

```
plot(primatetree)
```

Note that the tree has too many species for us to read the species names. You can make the tip labels smaller, but that doesn't help much here:

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

Alternatively you can zoom into different sections of the tree that you're interested in:

```
zoom(primatetree, list(grep("Cercopithecus", primatetree$tip.label)),
      subtree = FALSE)
```

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

```
zoom(primatetree, list(grep("Cercopithecus", primatetree$tip.label)),
      subtree = TRUE)
```

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))
```

You can also remove species from the tree very easily using the `ape` function `drop.tip`:

```
primatetree2 <- drop.tip(primatetree,  "Aotus_azarae_infulatus")

str(primatetree2)
```

```
## List of 4
##  $ edge       : int [1:444, 1:2] 226 227 228 229 ...
##  $ edge.length: num [1:444] 4.95 17.69 19.65 8.12 ...
##  $ Nnode      : int 220
##  $ tip.label  : chr [1:225] "Allenopithecus_nigroviridis" ...
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

To remove more than one species see below. 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). Here's an fun example!

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

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

You can also add a timescale to your tree:

```
plot(primatetree)

axisPhylo()
```

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)
```

Manipulating your data and phylogeny in R

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 _). 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 **g**eneral **sub**stitution. 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).

Species names = row names

A number of R functions require that species names are the row names of the data. This is really easy to fix:

```
row.names(primatedata) <- primatedata$Binomial
```

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.

Species in the phylogeny but not in the data

These are easy to identify using setdiff:

```
setdiff(primatetree$tip.label, primatedata$Binomial)
```

```
##   [1] "Allenopithecus_nigroviridis"
##   [2] "Cercopithecus_cephus_cephus"
##   [3] "Cercopithecus_cephus_ngottoensis"
##   [4] "Cercopithecus_diana"
etc.
```

`setdiff` tells you what is found in the first list, but not in the second. Here, it has listed the species that are found in the tree but **not** in the data. We can then use `setdiff` with `drop.tip` to prune the tree to just the species we have data for.

```
primatetree2 <- drop.tip(primatetree, setdiff(primatetree$tip.label,
                          primatedata$Binomial))
```

Note that 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 want.

Species in the data but not in the phylogeny

Again these are easy to identify using `setdiff`, just swap the inputs around:

```
setdiff(primatedata$Binomial, primatetree$tip.label)
```

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

In this case we don't have any species in the tree missing from the data. However, if you do, to remove species from the data which are not in the tree you can use `match` and `subset`:

```
matches <- match(primatedata$Binomial, primatetree2$tip.label, nomatch = 0)

primatedata2 <- subset(primatedata, matches != 0)
```

Remember `!=` means "does not equal". So this line of code only selects species which do appear in the tree, i.e. their value from `matches` is not 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(primatedata2)

str(primatetree2)
```

Controlling for phylogenetic non-independence

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))

hist(primatedata$AdultBodyMass_g, col = rainbow(8))

hist(primatedata$GestationLen_d, col = rainbow(8))

hist(log(primatedata$AdultBodyMass_g), col = rainbow(8))

hist(log(primatedata$GestationLen_d), col = rainbow(8))

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

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)
```

The output should look like this:

```
##
## Call:
## lm(formula = log(GestationLen_d) ~ log(AdultBodyMass_g), data = primatedata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6161 -0.0828  0.0065  0.1141  0.5056
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.1038     0.1108   37.04  < 2e-16 ***
## log(AdultBodyMass_g)   0.1204     0.0141    8.54  1.1e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.198 on 75 degrees of freedom
## Multiple R-squared:  0.493,  Adjusted R-squared:  0.487
## F-statistic:   73 on 1 and 75 DF,  p-value: 1.1e-12
```

The slope is positive (0.1204 ± 0.0141), and very significant (note that R can't give 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 looking at just one family.

```
points(log(GestationLen_d[Family == "Cercopithecidae"]) ~
       log(AdultBodyMass_g[Family ==  "Cercopithecidae"]),
       data = primatedata, col = "blue", pch = 16)
```

Phylogenetic generalized least squares models (PGLS)

Phylogenetic generalized least squares (PGLS) models offer 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 will be dropping the 149 species from the tree that we don't have any data for, but we won't drop any species from the data.

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

This will give a warning telling you that some species have been dropped.

```
##  Warning message:
##  In comparative.data(phy = primatetree, data = primatedata, :
##    Data dropped in compiling comparative data object
```

You can view the dropped species using:

```
primate$dropped$tips

primate$dropped$unmatched.rows
```

> 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 estimate of λ we use the following code:

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

summary(model.pgls)
```

The output should look like this:

```
##
## Call:
## pgls(formula = log(GestationLen_d) ~ log(AdultBodyMass_g),
##      data = primate, lambda = "ML")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.09890 -0.01166  0.00308  0.01776  0.07513
##
## Branch length transformations:
##
## kappa  [Fix]  : 1.000
## lambda [ ML]  : 0.892
##    lower bound : 0.000, p = 1.1e-14
##    upper bound : 1.000, p = 0.00046
##    95.0% CI   : (0.753, 0.967)
## delta  [Fix]  : 1.000
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.2902     0.1604   26.75  < 2e-16 ***
## log(AdultBodyMass_g)   0.1049     0.0196    5.34  9.5e-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: 0.276,   Adjusted R-squared: 0.266
## F-statistic: 28.5 on 1 and 75 DF,  p-value: 9.48e-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 = primate$data)

abline(model.pgls)
```

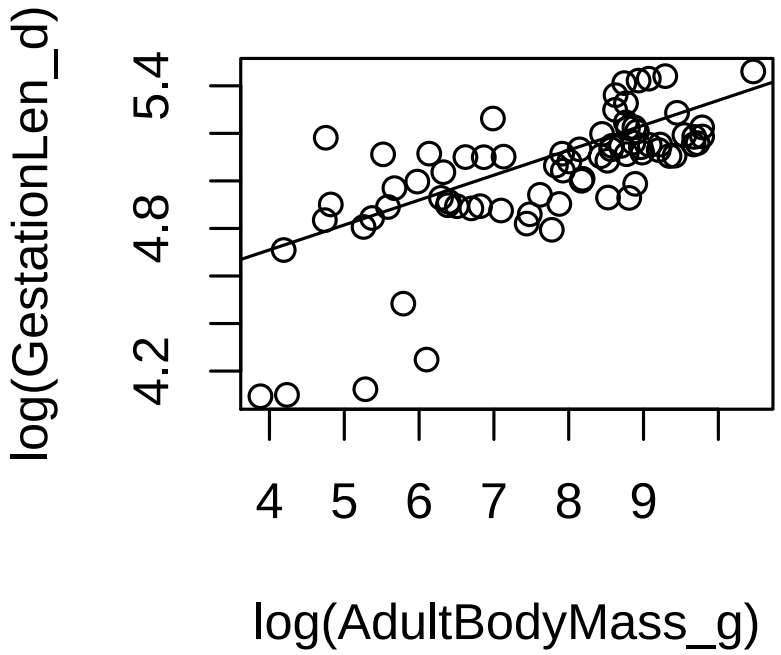

Figure 1: Plot of PGLS regression model

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 lambda. Just change the lower bound of lambda 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 = list(lambda = c(1e-05, 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)
```

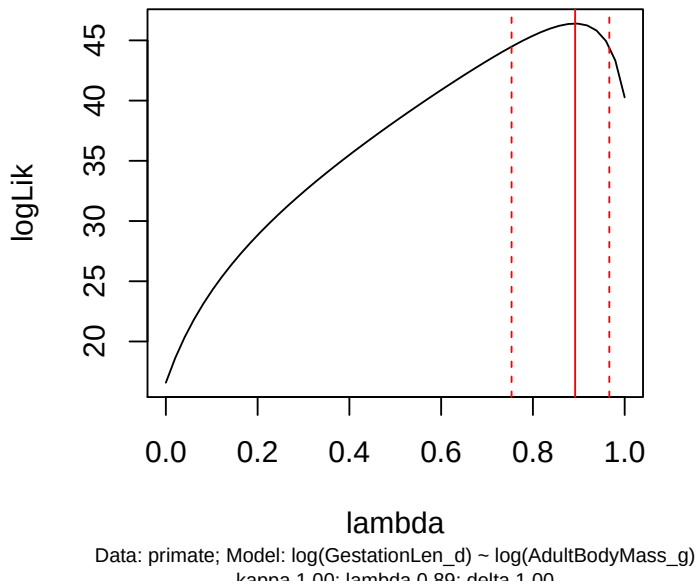

Figure 2: λ profile for PGLS model

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] 0.7534 0.9665
```

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 many areas of anthropology. 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 lambda set to 1 (equivalent to independent contrasts). This problem comes up every year at AnthroTree and consensus among the PCM community (well Rob Freckleton, Charlie Nunn and others!) 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.pgls <- pgls(log(GestationLen_d) ~ log(AdultBodyMass_g),
                   data = primate, lambda = 1)
```

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)
```

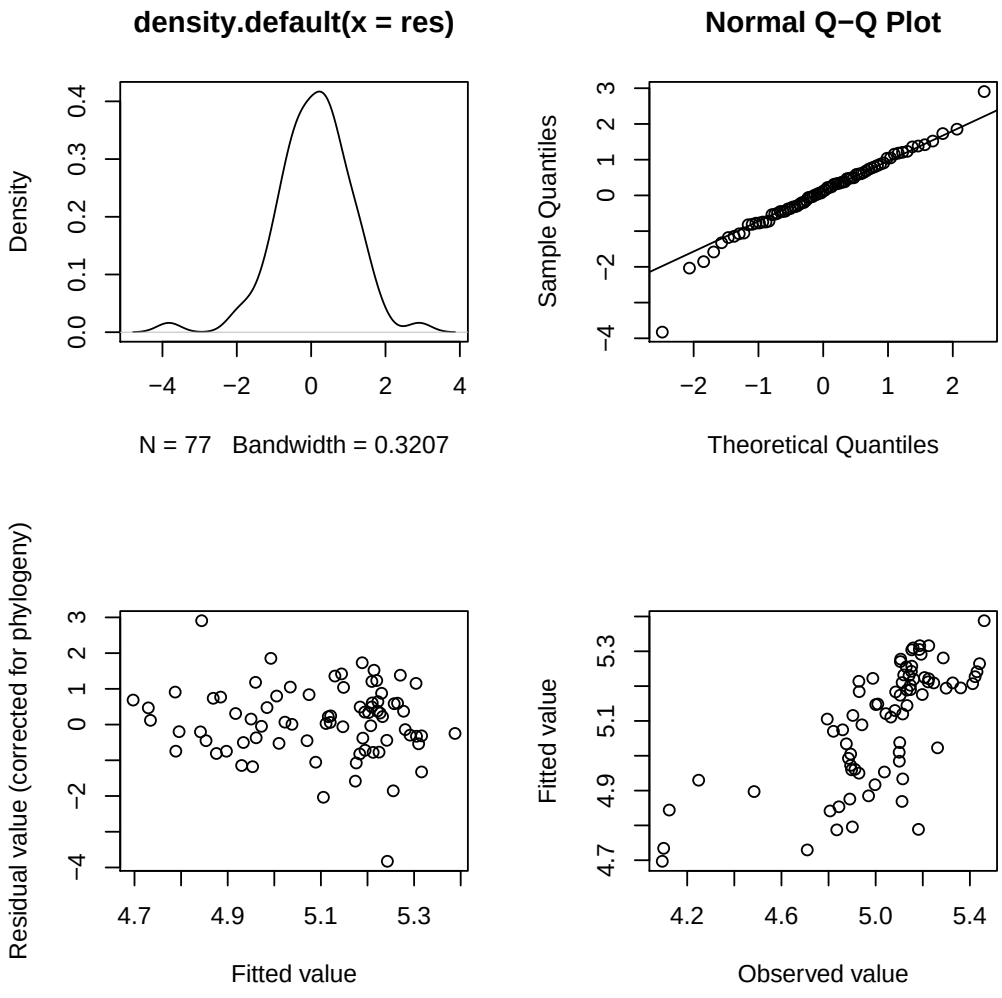

Figure 3: Model diagnostic plots for PGLS model

Without going into the statistical details, 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.
3. Plots 3 and 4 should show a fairly random scattering of points. You want to avoid any clear patterns.

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.

Estimating phylogenetic signal for one variable: λ

Phylogenetic signal is merely the pattern where close relatives have more similar trait values than more distant relatives. 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 correct for phylogeny because the **residuals** from our models show phylogenetic signal – see Revell 2010 Methods in Ecology and Evolution. λ shown in PGLS models above is the λ for the model residuals **not** the individual variables. Also see Kamilar and Cooper 2013 Phil Trans B.

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:

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

summary(est.lambda)
```

```
## Call:
## pgls(formula = log(GestationLen_d) ~ 1, data = primate, lambda = "ML")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12835 -0.01446  0.00139  0.01757  0.07433
##
## Branch length transformations:
##
## kappa  [Fix]  : 1.000
## lambda [ ML]  : 0.948
##    lower bound : 0.000, p = <2e-16
##    upper bound : 1.000, p = 0.03
##    95.0% CI   : (0.859, 0.996)
## delta  [Fix]  : 1.000
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.005      0.117    42.7   <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).

Estimating phylogenetic signal for one variable: Blomberg's K

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] "Dropping taxa from the data because they are not present in the phylogeny:"
## [1] NA
## [1] "Dropping tips from the tree because they are not present in the data:"
##   [1] "Allenopithecus_nigroviridis"
##   [2] "Cercopithecus_cephus_cephus"
##   [3] "Cercopithecus_cephus_ngottoensis"
##   [4] "Cercopithecus_diana"
etc.

##        [,1]
## [1,] 0.7758
```

Note that we are using the original data and tree here, not the pruned down versions, so lots of tips are being dropped.

`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)
```

After the warnings about which tips have been dropped, the output looks like this:

```
##        K PIC.variance.obs PIC.variance.rnd.mean PIC.variance.P
## 1 0.7758         0.001661               0.01212       0.000999
##   PIC.variance.Z
## 1         -2.281
```

In this case the observed K is significantly higher than the random values of K (Z = -2.281, $p < 0.001$, mean of the random values = 0.012).

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

Practice exercises

The code above only runs these analyses on a few variables. If you have time, try and run some analyses on the other variables. Here are some example questions to get you started (or you can head straight to testing it on your own data!).

1. What is λ for primate social group size?
2. Make a plot of social group size against home range size. Color the Cebidae in green.
3. Perform a PGLS of two variables of your choice.
4. Does home range size differ significantly among the different primate genera?
5. What is Blomberg's K for adult body mass?
6. Plot the primate tree and zoom in on the *Saimiri* genus.
7. Run a PGLS analysis to determine the relationship between social group size and gestation length. How does this differ from the result obtained with an OLS regression? Plot the result.