# Phylogenetic Generalized Least Squares in R

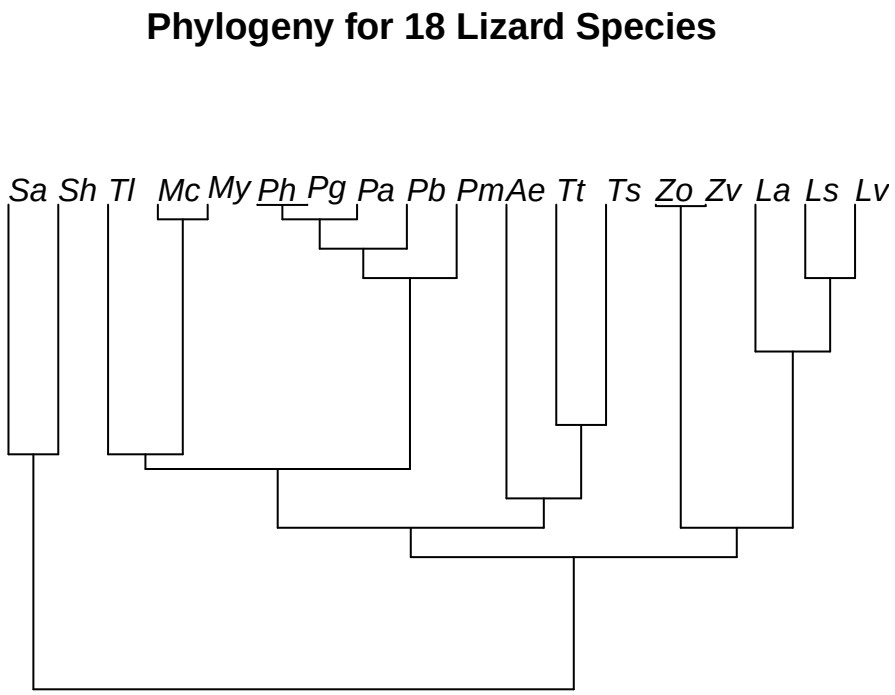
Here we give an example PGLS analysis using a tree and data from the **ade4** package. We analyse the data using functions from the **ape** and **nlme** packages. There are several other ways to conduct PGLS analyses in R, most notably using the **pgls** function in the **caper** package. We choose to present an analysis using **nlme** as this is mature, general purpose software with useful "helper" functions, and it ships with the base R distribution.

The first step in any analysis is to import your tree and data into R. For most purposes, the R functions **read.table** and **read.tree** from the **ape** package will suffice. Typical programming commands would be:

```
library(ape)
dat <- read.table("myDataFile.csv", header=TRUE, sep=",") # import a comma-delimited
                                                          # (csv) file with column headings
tree <- read.tree("MyNewickTree.phy") # import a Newick format tree
```

Where the data and tree files, **myDataFile.csv** and **MyNewickTree.phy**, are assumed to be in the user's working directory. Package **ape** also has a function for reading Nexus files, **read.nexus**, which is another popular tree file format. For demonstration purposes in this tutorial, we will use a tree and data from the **ade4** package. Consider the following life history data and tree (including branch lengths) for 18 species of lizards:

```
library(ade4)  # source of example data and tree
library(ape)  # tree handling
library(nlme)  # regression modelling
data(lizards)
tree <- read.tree(text = lizards$hprA)
dat <- lizards$traits[tree$tip.label, ]  # sort data according to tree
plot(tree, main = "Phylogeny for 18 Lizard Species", direction = "up", srt = -90,
    label.offset = 1)
```



```
head(dat)  # just the first few lines of data

##    mean.L matur.L max.L hatch.L hatch.m clutch.S age.mat clutch.F
## Sa   69.2      58    82    27.8   0.572      6.0      13      1.5
## Sh   48.4      42    56    22.9   0.310      3.2       5      2.0
## Tl  168.4     132   190    42.8   2.235     16.9      19      1.0
## Mc   66.1      56    72    25.0   0.441      7.2      11      1.5
## My   70.1      60    81    26.6   0.550      5.4      10      1.0
## Ph   49.6      39    57    23.8   0.310      2.1       8      2.0
```

Suppose we were interested in the regression of length at maturity on age at maturity. If we were to assume a Brownian motion model of evolution, then we could construct the phylogenetic correlation matrix implied by the tree, and then use that in a call to the `gls` function in the `nlme` package:

```
mat <- vcv(tree, corr=TRUE) # construct matrix
fit <- gls(matur.L ~ age.mat, correlation=corSymm(mat[lower.tri(mat)],fixed=TRUE), data=dat)
```

There are several things to note:

- The `vcv` function computes a covariance matrix by default, and a correlation matrix with argument `corr=TRUE`.
- The `gls` function works like the other regression functions in R, except that it has a `correlation` argument.
- The lower triangle of the correlation matrix is passed to the `corSymm` function, which specifies a general symmetric correlation matrix. Since we are not estimating any parameters in this matrix, we set `fixed=TRUE`, telling `gls` that the correlation matrix is fixed for the duration of the parameter optimization.
- This analysis assumes that the tree is ultrametric, meaning that the tips of the tree all line up. They do in this case, but it is easy to imagine cases where this might not be so. (e.g. inclusion of extinct species or certain branch length transformations). The solution is to use the `weights` argument to `gls`:

```
tip.heights <- diag(mat)
fit <- gls(matur.L ~ age.mat, correlation = corSymm(mat[lower.tri(mat)], fixed = TRUE),
    weights = varFixed(~tip.heights), data = dat)
```

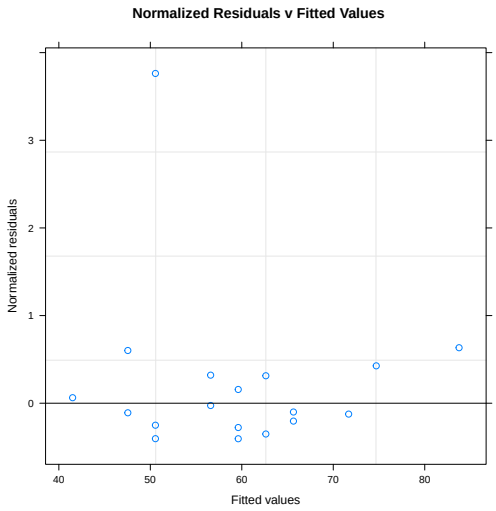
See this article for further explanation. The `ape` package has several other correlation structures for use with `gls`. The simplest is `corBrownian`, which fits the same model as above:

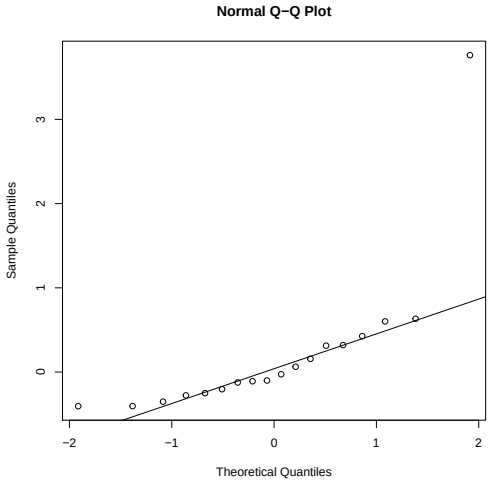
```
fit2 <- gls(matur.L ~ age.mat, correlation=corBrownian(phy=tree), data=dat)
anova(fit, fit2) # models are the same

##      Model df   AIC   BIC logLik
## fit      1  3 172.4 174.8 -83.22
## fit2     2  3 172.4 174.8 -83.22
```

Using `corBrownian` is simpler than using `corSymm` since all you need to tell it is the name of the tree. You do not need to mess with the correlation matrix directly.

We can do some basic diagnostic plots to see if the model departs from the usual regression assumptions. In particular, we can look at the residuals versus the fitted values to check for heteroscedasticity and/or curvature, and a quantile-quantile plot of the (normalized) residuals to check for departures from the assumption of Normally-distributed residuals:

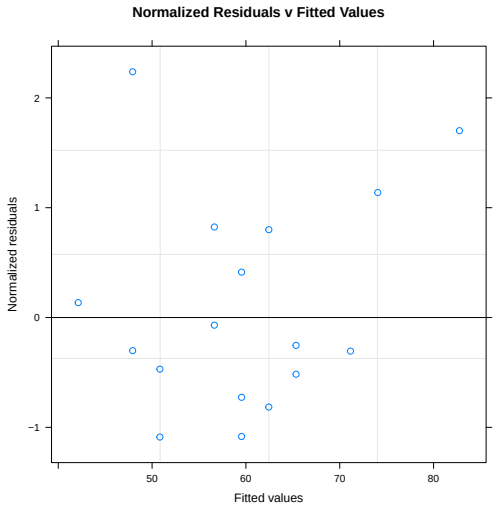
```
plot(fit2, resid(., type="n")~fitted(.), main="Normalized Residuals v Fitted Values",
     abline=c(0,0))
res <- resid(fit2, type="n")
qqnorm(res)
qqline(res)
```

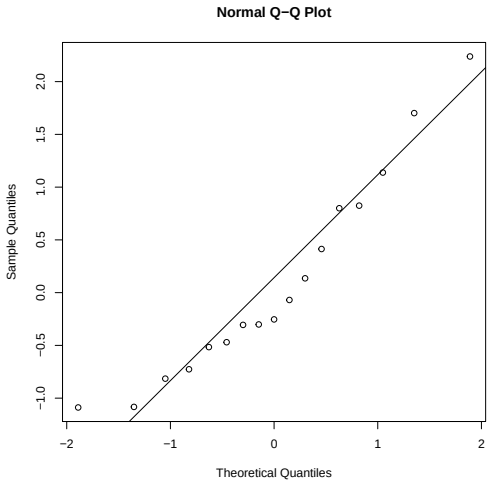



Note that the plots look pretty good, except for perhaps one outlier. We could re-examine the original data to see if there was a mistake or something unusual about that species. One option is then to exclude that species from the analysis. Re-running the analysis without the outlier (species "Pg"):

```
res[which.max(res)] # species with largest residual (outlier)

##    Pg
## 3.762

dat3 <- dat[-which(rownames(dat) == "Pg"),]
tree3 <- drop.tip(tree, "Pg")
fit3 <- gls(matur.L ~ age.mat, correlation=corBrownian(phy=tree3), data=dat3)
```



Both plots look a lot better. Now we can get some summary information about the model:

```
summary(fit3)

## Generalized least squares fit by REML
##   Model: matur.L ~ age.mat
##   Data: dat3
##     AIC   BIC logLik
##   138.6 140.7 -66.31
##
## Correlation Structure: corBrownian
##  Formula: ~1
##  Parameter estimate(s):
## numeric(0)
##
## Coefficients:
##              Value Std.Error t-value p-value
## (Intercept) 27.616    21.545   1.282  0.2194
## age.mat      2.902     1.436   2.021  0.0616
##
##  Correlation:
##         (Intr)
## age.mat -0.727
##
## Standardized residuals:
##     Min      Q1     Med      Q3     Max
## -0.5715 -0.3228 -0.1223  0.1508  1.7016
##
## Residual standard error: 28.94
## Degrees of freedom: 17 total; 15 residual
```

We see that the slope of the relationship between length at maturity and and age at maturity is 2.902, and this is not quite significant with $p = 0.0616$. This means that for every unit increase in age at maturity, there is an associated increase of 3 units in length at maturity. Later maturing species are longer at maturity.

Perhaps we are concerned about the amount of phylogenetic "signal" in our data. The observed data may not have evolved under a Brownian motion model. (NB: All models are wrong.) Instead, we may include an extra parameter in the correlation matrix to effectively account for error in branch-length estimation. Our new matrix is then a function of the tree and this new parameter: $NewMatrix = f(tree, parameter)$. Package `ape` supplies four single-parameter correlation structures for use with function `gls`. These are `corPagel`, `corGrafen`, `corMartins` and `corBlomberg`. Each correlation structure constructs the correlation matrix in a different way. The simplest is `corPagel`, where there is a parameter ($\lambda$) that multiplies the off-diagonals of the Brownian motion correlation matrix. $\lambda$ can range from zero (no phylogenetic signal, equivalent to a "star" phylogeny), to one (consistent with Brownian motion). Intermediate values imply that the data support a model that is somewhere between a "star" phylogeny and Brownian motion. $\lambda$ represents the amount of phylogenetic "signal" in the data, conditional on the tree. In a regression context, we can use the method of (restricted) maximum likelihood to simultaneously fit the regression model and estimate $\lambda$, which represents the phylogenetic signal in the residuals. We can also get a confidence interval for $\lambda$ and perform tests of the hypotheses $\lambda = 0$ and $\lambda = 1$.

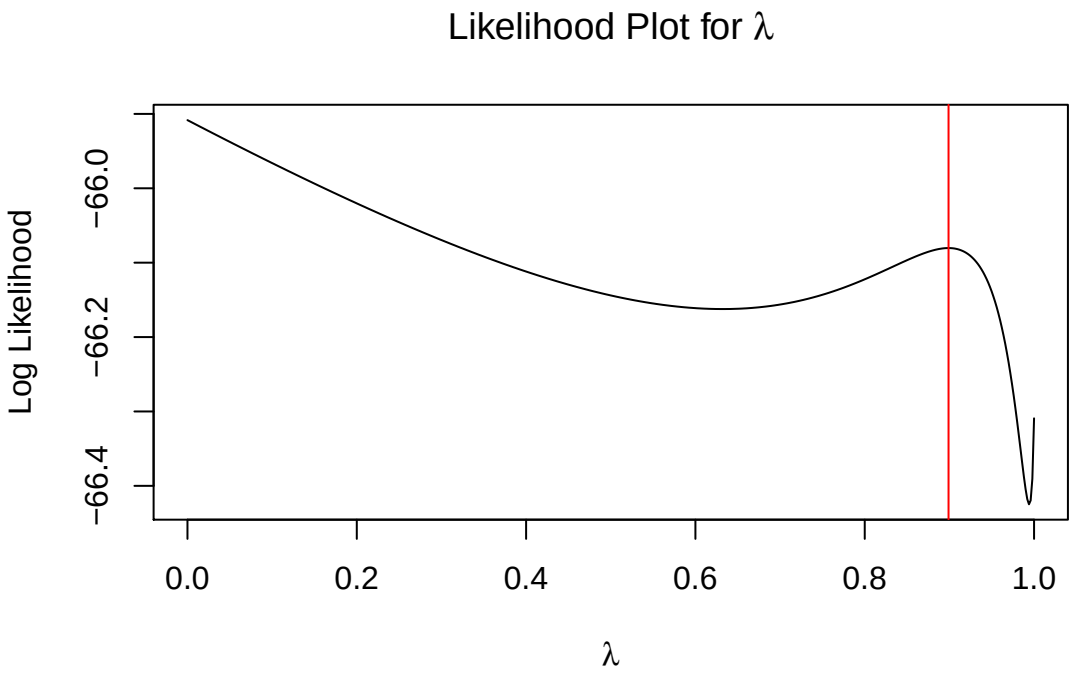
```
fitPagel <- gls(matur.L ~ age.mat, correlation=corPagel(value=0.8, phy=tree3), data=dat3)
intervals(fitPagel, which="var-cov")

## Approximate 95% confidence intervals
##
##  Correlation structure:
##        lower  est. upper
## lambda  0.49 0.899 1.308
## attr(,"label")
## [1] "Correlation structure:"
##
##  Residual standard error:
## lower  est. upper
## 11.76 21.88 40.72
```

We can get direct tests of the hypotheses $\lambda = 0$ (independence) and $\lambda = 1$ (Brownian motion) by fitting models that are forced to have $\lambda = 0$ (independence) and $\lambda = 1$ (Brownian motion) and then comparing them with a likelihood ratio $\chi^2$ test.

```
fitPagel0 <- gls(matur.L ~ age.mat, correlation = corPagel(value = 0, phy = tree3,
    fixed = TRUE), data = dat3)  # independence
fitPagel1 <- gls(matur.L ~ age.mat, correlation = corPagel(value = 1, phy = tree3,
    fixed = TRUE), data = dat3)  # Brownian motion
anova(fitPagel, fitPagel0)

##           Model df   AIC   BIC logLik   Test L.Ratio p-value
## fitPagel      1  4 140.2 143.0 -66.08
## fitPagel0     2  3 137.8 139.9 -65.91 1 vs 2  0.3439  0.5576

anova(fitPagel, fitPagel1)

##           Model df   AIC   BIC logLik   Test L.Ratio p-value
## fitPagel      1  4 140.2 143.0 -66.08
## fitPagel1     2  3 138.6 140.7 -66.31 1 vs 2  0.4578  0.4987
```

We see that the likelihood ratio $\chi^2$ tests comparing our "free-floating" $\lambda$ model with a model that has $\lambda$ fixed at zero or one have p-values of `0.5576` and `0.4987`, respectively. Thus, we cannot reject the null hypothesis of "independent residuals" or Brownian motion. Why then does the confidence interval on $\lambda$ not include zero? Correct estimation of $\lambda$ is difficult with such a small sample size. This can be seen by running the `gls` analysis with different starting values for $\lambda$. The optimal lambda converges to values that are either 0.899 or less than zero. One can get an impression of the estimation of $\lambda$ by plotting the model likelihood for various fixed values of $\lambda$:

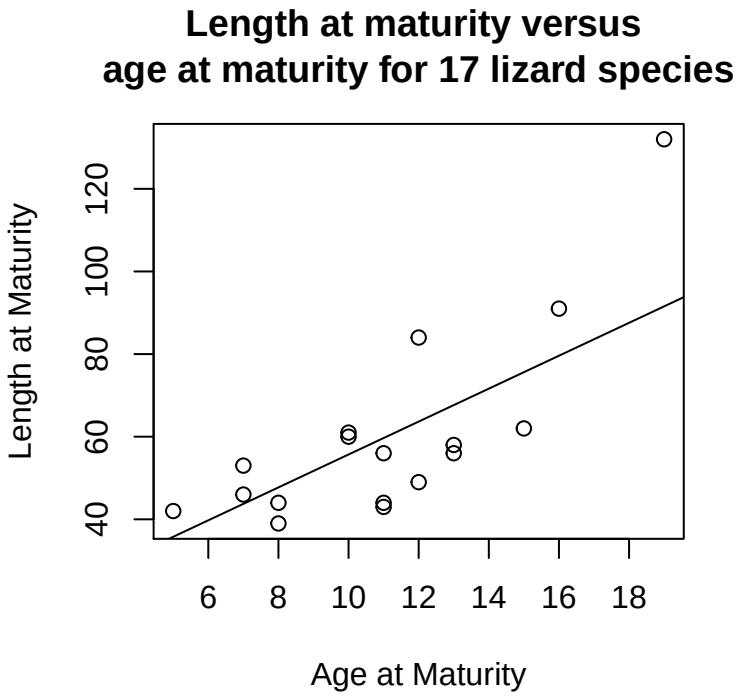
```
lambda <- seq(0, 1, length.out = 500)
lik <- sapply(lambda, function(lambda) logLik(gls(matur.L ~ age.mat,
    correlation = corPagel(value = lambda, phy = tree3, fixed = TRUE),
    data = dat3)))
plot(lik ~ lambda, type = "l", main = expression(paste("Likelihood Plot for ",
    lambda)), ylab = "Log Likelihood", xlab = expression(lambda))
abline(v = fitPagel$modelStruct, col = "red")
```




We can see that for starting values for $\lambda$ less than about 0.7, numerical optimization will be drawn towards $\lambda \leq 0$. Otherwise, the optimization will be drawn towards $\lambda = 0.899$. This value is indicated by the red line. With data for more species, we may be able to better estimate $\lambda$. This is a warning about the difficulty of parameter estimation with small sample sizes.

We would like to construct a plot of the relationship between length at maturity and age at maturity, and include the trend line:

```
with(dat3, plot(matur.L ~ age.mat, xlab = "Age at Maturity", ylab = "Length at Maturity",
    main = "Length at maturity versus \nage at maturity for 17 lizard species"))
abline(fitPagel)
```



Up until now we have concentrated on estimating $\lambda$ as part of a regression analysis. We may also want to know the value of $\lambda$ for a single trait. This is just a matter of fitting an intercept-only model. Here we do it for all the variables in the dataframe:

```
# set up matrix to hold the results
mat <- matrix(NA, ncol=3, nrow=(dim(dat3)[2]),
              dimnames=list(variable=names(dat3),
                            c("Lower" , "Estimate", "Upper")))

for (i in 1:(dim(dat)[2])) { # loop through each variable
  form <- formula(paste(names(dat)[i], " ~ 1")) # construct the model formula
  this.fit  <- gls(form, data=dat,
                   correlation=corPagel(0.1, phy=tree),
                   control=glsControl(opt="optim")) # fit the model
  ints <- try(intervals(this.fit)$corStruct) # get lambda and confidence interval
  # if there is an error, just get the point estimate and set the confidence limits to NA
if (inherits(ints, "try-error")) ints <- c(NA, coef(this.fit$modelStruct), NA)
  mat[i,] <- ints # save the results in the matrix
} # end loop
## Lambda for each variable:
mat

##
## variable    Lower Estimate Upper
##   mean.L   0.9502   0.9820 1.014
##   matur.L  0.9382   0.9779 1.018
##   max.L    0.9571   0.9853 1.014
##   hatch.L  0.8832   0.9582 1.033
##   hatch.m  0.9780   0.9929 1.008
##   clutch.S 0.8927   0.9637 1.035
##   age.mat      NA   1.0005    NA
##   clutch.F     NA   1.0000    NA
```

There are a few things to note:

- We constructed the model formula "on the fly" using the `formula` and `paste` functions. This is a useful technique if you need to fit many different regressions using different combinations of response and/or explanatory variables.
- We used a call to `glsControl` to change the optimizer for the fit. Sometimes this can improve convergence.
- We used the `try` function to catch errors. Sometimes a point estimate for $\lambda$ can be obtained, but if the associated variance-covariance matrix is not positive-definite, it will not be possible to estimate confidence limits (and there will be an error). Hence we told R that if there was an error, just report the point estimate for $\lambda$ and leave the confidence limits as `NA`, ie missing values.

Estimates of $\lambda$ for all the variables are very close to one, indicating considerable phylogenetic "signal" in these variables.

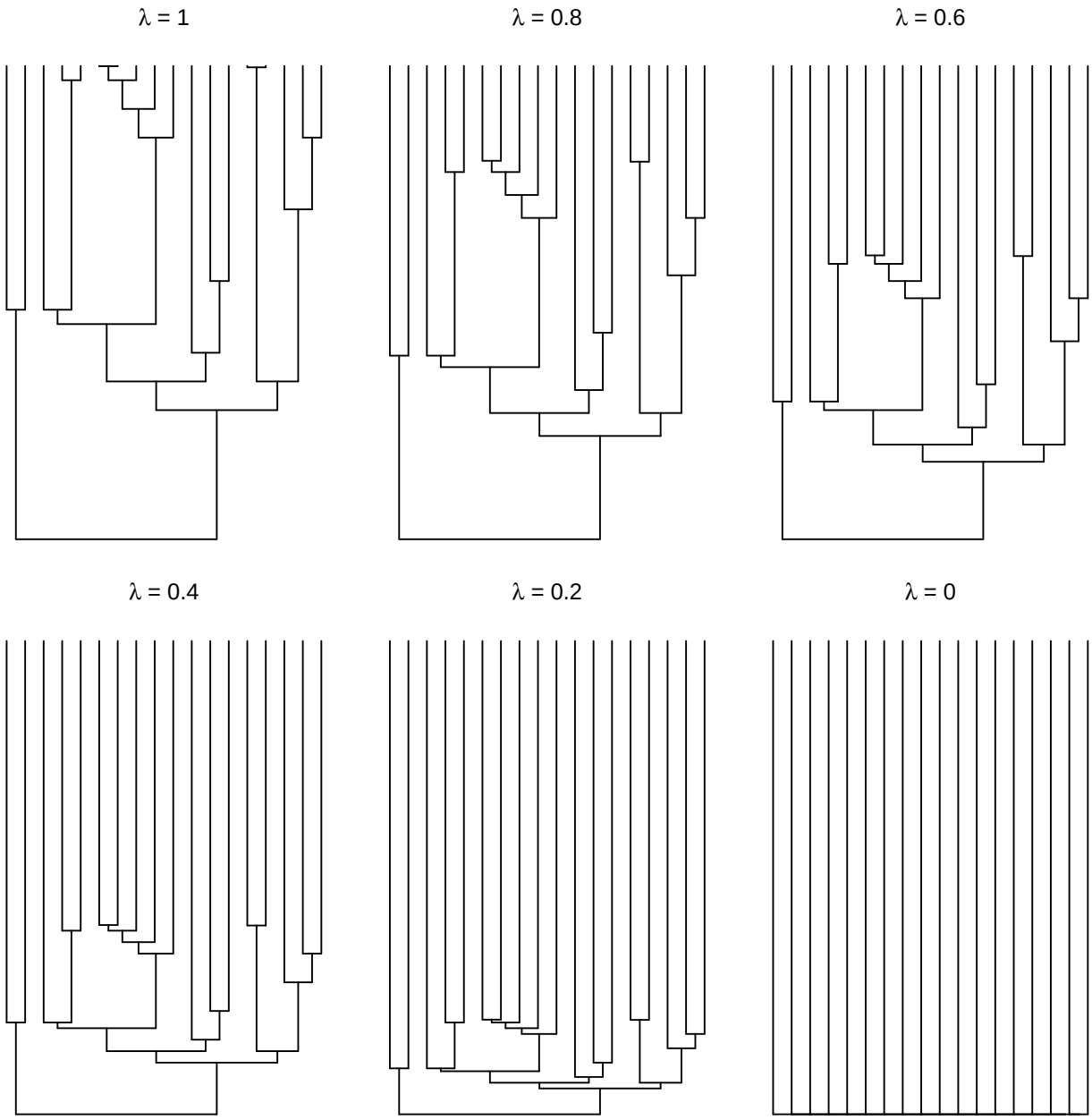
Rarely, there will be information external to the data relating to the value of $\lambda$. That is, $\lambda$ may be known. It is easy to use a pre-specified value for $\lambda$. Simply pass it to the `corPagel` function with the `value` argument, along with the argument `fixed=TRUE`:

```
new.lambda <- 0.7  # pre-specified lambda value
fitPagel4 <- gls(matur.L ~ age.mat, correlation = corPagel(value = new.lambda,
    fixed = TRUE, phy = tree3), data = dat3)
fitPagel4

## Generalized least squares fit by REML
##   Model: matur.L ~ age.mat
##   Data: dat3
##   Log-restricted-likelihood: -66.16
##
## Coefficients:
## (Intercept)     age.mat
##       9.886       4.532
##
## Correlation Structure: corPagel
##  Formula: ~1
##  Parameter estimate(s):
## lambda
##    0.7
## Degrees of freedom: 17 total; 15 residual
## Residual standard error: 18.73
```

Note that it is a mistake to first estimate $\lambda$ for a trait, and then use it in a subsequent analysis of the same data set. This is because the (restricted) maximum likelihood estimate of $\lambda$ for a single trait may be a poor estimate of $\lambda$ when applied to the residuals of a more complex model, such as a regression model. It makes sense to estimate $\lambda$ and the regression parameters together so that the $\lambda$ that is estimated is the (restricted) maximum likelihood estimator for the full data and model.

It is useful to know how the tree changes when different values of $\lambda$ are applied to it. Below are six versions of the original tree after transformation. The code uses the `vcv2phylo` function to convert back from a matrix to a tree, and requires the `matrixcalc` package:

```
library(matrixcalc)
source("vcv2phylo.R")  # load in the code for vcv2phylo
ll <- seq(1, 0, by = -0.2)
mats <- sapply(ll, function(x) corMatrix(Initialize(corPagel(x, phy = tree),
    data = dat)), simplify = FALSE)
trees <- lapply(mats, vcv2phylo)
class(trees) <- "multiPhylo"
par(mfrow = c(2, 3), mar = rep(1, 4))
for (i in 1:length(ll)) plot(trees[[i]], main = substitute(paste(lambda, " = ",
    ll), list(ll = ll[i])), direction = "upwards", show.tip.label = FALSE)
```




We can see that $\lambda = 1$ is just the original tree. Decreasing values of $\lambda$ lengthen the terminal branches and compress the internal branches. $\lambda = 0$ shrinks all the internal branches to length zero, producing a star phylogeny. It is left as an exercise for the reader to show the effect of other single-parameter transformations on the tree.