10kTrees - Exercise #4

Running Analyses Across a Tree Block – Phylogenetic Signal and Correlated Evolution

Once you have your block of trees, you are ready for running comparative analyses, such as reconstructing ancestral states, investigating the factors that influence speciation and extinction rates, or testing for correlated evolution.

How does one run a set of analyses across a tree block? You need to run your planned analysis across 100, or 1000, or even 10,000 trees. But you don't want to spend weeks doing this. Thus, the analysis needs to be automated in some way. Fortunately, a number of programs and packages already provide the kinds of automation to make this process feasible. Here, I focus on methods for correlated evolution involving independent contrasts and phylogenetic generalized least squares (PGLS) in R, and methods to study correlated evolution in the program BayesTraits.

For an example dataset, we will investigate the association between Trait Y and Trait X from Exercise 1, and the modified tree from Exercise 3. Recall that we simulated these data on the tree under a specific model of evolution and a specific branch length transformation. We will assess how well the methods reconstruct the "true" parameters that were simulated, and how phylogenetic uncertainty among the species in our hypothetical data affect the outcome of statistical results.

Before proceeding, please note: This exercise is only intended to give you a glimpse of how to run the analyses that follow. Further reading is necessary to implement the methods correctly, such as checking assumptions. I hope this opens up possibilities for you to make use of the tree blocks from *10kTrees*, but please cite the appropriate software program and package files for implementation, and be sure you understand all of the code that follows before using the results in published p*ape*rs.

1) Calculating Independent Contrasts Across a Tree Block in R

We will calculate independent contrasts using the *ape* package. Open R. To use *ape*, we must first download and install it. To load a package that has already been installed in your R system, you can type `library("ape");` otherwise, type `install.packages("ape")`. You can also use the graphical interfaces for downloading and installing via the menus.

You will also need the data for this exercise (edu_data.csv), and the tree from Exercise 3 (also available on the server if you skipped this exercise). Put the two files in a folder of your choice.

Once you have successfully installed *ape* and acquired the files, change the working directory to the directory where the files are located (e.g., using the Misc menu on a Mac, or the setwd() function; see previous exercises).

After loading the ape library, let's read in our tree file. Open the text file to take a quick look at the format and remind yourself of the Nexus format. Then, import the tree using "read.nexus" in the *ape* package.

```
tree_list <- read.nexus("treeblock_example.nex.ber.nex.coq.nex")
```

As you will recall, the tree block consists of 200 trees downloaded from the *10kTrees* website. Let's plot the second tree in the file by passing that tree to a new variable, "tree", as follows:

```
tree2 <- tree_list[[2]]
plot(tree2)
```

Now, let's read in the data:

```
data.yx <- read.csv("edu_data.csv", header=T)
```

And, to make our lives easiers, let's assign the values in the data frame to new variables for "`trait.Y`" and "`trait.X`", as follows:

```
trait.Y <- data.yx$Trait.Y
trait.X <- data.yx$Trait.X
```

Type these variables to see the vector of values, and compare to what is seen when you type "data.yx".

Now, let's assign names to characters, which is necessary for linking up the data to the tree.

```
names(trait.Y) = data.yx$Species
names(trait.X) = data.yx$Species
```

Now type the variable names again, e.g., "trait.Y". Do you see how the names are now linked to the data vectors? Does this match what you saw in the table for "data"?

Now that we have variables with names, we can use the "pic" command to calculate "phylogenetically independent contrasts" for each variable. Give it a try, using tree #1, and then we will apply what we learn to the tree block.

First, calculate the contrasts on tree 1 with the `pic` function:

```
pic.trait.Y <- pic(trait.Y, tree_list[[1]])
pic.trait.X <- pic(trait.X, tree_list[[1]])
```

To display the contrasts graphically:

```
plot(pic.trait.Y, pic.trait.X)
```

And on the actual tree, at the nodes:

```
plot(tree_list[[1]])
nodelabels(round(pic.trait.Y, 3), adj = c(0, -0.5),frame =
"n")
nodelabels(round(pic.trait.X, 3), adj = c(0, 1),frame = "n")
```

We can then run a linear model through the origin (don't forget that "-1" at the end of the model specification! This puts the regression through the origin.)

```
lmResult <- lm(pic.trait.Y ~ pic.trait.X - 1)
summary(lmResult)
```

Interestingly, these variables appear to show no association using independent contrasts calculated on tree 1; the regression coefficient is not significantly different from zero:

```
Call:
lm(formula = pic.trait.Y ~ pic.trait.X - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-5.4282 -1.1328 -0.3391  0.3865  4.9279

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
pic.trait.X   0.2181     0.2033   1.073    0.291

Residual standard error: 1.948 on 33 degrees of freedom
Multiple R-squared: 0.03372,  Adjusted R-squared: 0.004439
F-statistic: 1.152 on 1 and 33 DF,  p-value: 0.291
```

Now, let's build a simple loop to go through all the trees, and to store the results for each tree. We will use a for loop, which will go from tree 1 to tree 200, calculating contrasts and storing key results in vectors.

```
slope_list<- tstat_list <- pval_list <-matrix(0,
length(tree_list))

for (i in 1:length(tree_list))
{
     pic.trait.Y <- pic(trait.Y, tree_list[[i]])
     pic.trait.X <- pic(trait.X, tree_list[[i]])

     lmOut <- summary(lm(pic.trait.Y ~ pic.trait.X - 1))
```

```
        slope_list[i]=lmOut$coefficients[1]
        tstat_list[i]=lmOut$coefficients[3]
        pval_list[i]=lmOut$coefficients[4]
    }
```

Take a look at the code to try to understand it. And then copy and paste into R... it may take several minutes for all the calculations to complete.

Now, with the list of slopes, t-statistics and p-values, we can say a lot about the statistical results. For example, find the largest p-value and the mean p-value across the 200 calculations:

```
max(pval_list)
mean(pval_list)
min(pval_list)
```

The maximum is ~0.62, while the minimum is <0.0001. Thus, it appears that these results depend on the tree used! Typing `hist(pval_list, 100)`, however, reveals that most of the of the tests were not significant.

To use this approach, you would need to find that all statistics are above (or below) this cut-off. A more mixed result is harder to interpret, and suggests that you should use the Bayesian approach implemented in BayesTraits (see below).

Remember as well that you should be checking the assumptions of independent contrasts.

2) PGLS Across a Tree Block in R

Phylogenetic generalized least squares (PGLS) offers important advantages over independent contrasts (IC). The model of trait evolution can be more flexible, i.e. it can depart from a strict "Brownian motion" process. In particular, different scaling parameters (e.g., λ) can be estimated and incorporated into the analysis, which can significantly improve the fit of the data to the model and thus also improve the estimation of statistical parameters of interest, such as regression coefficients. Another advantage of PGLS is that the intercept of the regression model is not forced to equal 0; we can estimate the intercept much as one would in an ordinary regression, and this can be useful for testing certain hypotheses and for predicting traits values based on the model.

In fact, IC represents a special case of PGLS. Thus, if λ does not depart from 1.0 (which implies Brownian motion), GLS will return results that are identical to independent contrasts. If the assumption of Brownian motion is incorrect, however, then λ will depart from 1.0. Incorporating a scaling parameter such as λ into the analysis is interesting for

its own sake - e.g., as a measure of phylogenetic signal - and it can result in different parameter estimates than IC for parameters of the statistical model.

The PGLS approach that we will use is implemented in R and the R phylogenetics package *ape* and one of the R linear modeling packages *nlme*.

Open R and verify that you installed and loaded the *ape* and *nlme* packages:

```
install.packages("ape")  # if needed
library(ape)

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

Next change the working directory to the directory where the files for this example are located on your computer using, e.g., the "Misc" menu or the setwd function (see above).

If you haven't completed the first part of this exercise, acquire the necessary files and place them in a folder, as above. Then, import the phylogeny:

```
tree_list <- read.nexus("treeblock_example.nex.ber.nex.coq.nex")
```

Now, let's read in the data on group composition (unless done above in your currently open R session):

```
data.yx <- read.csv("edu_data.csv", header=T)
trait.Y <- data.yx$Trait.Y
trait.X <- data.yx$Trait.X
names(trait.Y) = data.yx$Species
names(trait.X) = data.yx$Species
```

To begin, let's run the analysis on tree 1, and then expand to the full tree block. First, let's declare a phylogenetic structure based on our tree and a Brownian motion model of evolution. We also need to create a data frame for our data, to provide to the *gls* function.

```
bm.tree1_lam1 <- corPagel(value = 1,phy = tree_list[[1]],
fixed=T)
DF <- data.frame(trait.Y, trait.X)
```

This last step might seem redundant, since we already have the data frame "data.yx". However, it ensures that the names are in the same order as tree tips and matches up names of columns.

We are now ready to run the GLS with our correlation structure set as the tree.

```
outTree1Lambda1 <- gls(trait.Y ~ trait.X, correlation =
bm.tree1_lam1,data = DF, method= "ML")
```

To get the results, let's use the handy "summary" function:

```
summary(outTree1Lambda1)
```

The following output should appear which, as expected based on λ=1, are nearly identical to the output from independent contrasts above:

```
Generalized least squares fit by maximum likelihood
  Model: trait.Y ~ trait.X
  Data: DF
       AIC      BIC    logLik
  252.1573 256.8233 -123.0787

Correlation Structure: corPagel
 Formula: ~1
 Parameter estimate(s):
lambda
     1

Coefficients:
                Value Std.Error   t-value p-value
(Intercept) 1.7012613  7.336423 0.2318925  0.8181
trait.X     0.2181325  0.203267 1.0731315  0.2910

 Correlation:
        (Intr)
trait.X 0.069

Standardized residuals:
        Min          Q1         Med          Q3         Max
-0.97764402 -0.40107201 -0.06518098  0.12961331  0.85326219

Residual standard error: 16.11462
Degrees of freedom: 35 total; 33 residual
```

R outputs intercept (1.70) and slope (0.22) for the regression model, along with standard errors, t-statistics and p-values for intercept and slope. We can also obtain the log-likelihood (-123.1) and other estimates of model fit, such as AIC. Finally, you can obtain residuals from the model, as follows:

```
outTree1Lambda1$residuals
```

The analysis in R should be identical to an IC analysis. However, we can also estimate λ and incorporate estimates of λ into the analysis. In other words, we can obtain a measure of phylogenetic signal, and then tailor the method to take into account that level of signal when running the GLS (R can also estimate and set kappa and delta with slightly different code, or the Ornstein-Uhlenbeck model for stabilizing selection).

To set λ to zero, which is equivalent to a non-phylogenetic analysis, modify the "value" to be zero (it can actually take any value you want, between 0 and 1):

```
bm.tree1_lam0 <- corPagel(value = 0,phy = tree_list[[1]],
fixed=T)
```

Re-execute the R code again, typing:

```
outTree1Lambda0 <- gls(trait.Y ~ trait.X, correlation =
bm.tree1_lam0,data = DF, method= "ML")
```

The PGLS results will be different - and interestingly, the likelihood is higher when λ is set to zero, suggesting that in fact a value of 1 - as is assumed in independent contrasts - is inappropriate. The question is, what should we set λ to equal? Maybe the maximum likelihood estimate of λ lies between 0 and 1.

You can estimate the maximum likelihood value of lambda. Re-type the code to read:

```
bm.tree1_lamEst <- corPagel(value = 0,phy = tree_list[[1]],
fixed=F)
```

The "fixed=F" part indicates that it is false (F) that we are setting lambda to 1 (T stand for true in R function parameter values). Thus, we are obtaining the maximum likelihood estimate of λ.

Now, run the gls model again, with this new corPagel structure.

```
outTree1LambdaEst <- gls(trait.Y ~ trait.X, correlation =
bm.tree1_lamEst,data = DF, method= "ML")
```

We can see that for this tree, the maximum likelihood estimate of λ is 0.58, which is very close to the λ transformation used to simulate the data (λ=0.7).

Now, let's run this analysis across the whole tree block, again using a loop as in the section for independent contrasts:

```
slope_list<- tstat_list <- pval_list <- lamb_list <-
likeli_list <-matrix(0, length(tree_list))

for (i in 1:length(tree_list))
{
     bm.tree1_lamEst <- corPagel(value = 0,phy =
     tree_list[[i]], fixed=F)


outTree1LambdaEst = summary( gls(trait.Y ~ trait.X,
correlation = bm.tree1_lamEst,data = DF, method= "ML"))

lamb_list[i]= attr(outTree1LambdaEst$apVar, "Pars")[1]
```

```
slope_list[i]=outTree1LambdaEst$coefficients[2]
pval_list[i]=outTree1LambdaEst$tTable[2,4]
tstat_list[i]=outTree1LambdaEst$tTable[2,3]
likeli_list[i]=outTree1LambdaEst$logLik
}
```

Let's check out the distribution of the various statistics using the hist() function; copy and paste these one at a time:

```
hist(lamb_list,50)
hist(slope_list,50)
hist(pval_list,50)
hist(tstat_list,50)
hist(likeli_list,50)
```

What can we interpret from these distributions? First of all, the distribution of lambda is fairly tight across the trees, but is not centered on the expected value of 0.7. This could indicate bias in estimating λ, but we can't determine that based on a single simulation; the data would need to be simulated multiple times, and ideally on multiple trees. An important conclusion, however, is that λ is clearly neither zero nor 1.

Second, the slopes all cluster above zero. Does this necessarily mean that the effect is significant? No. All we see here are variation in estimating the slope based on variation in the topology and branch lengths. The estimates could have wide errors despite a tight distribution of values in the context of phylogenetic uncertainty. To assess statistical significance, we need to examine the p-values or t-statistics.

Finally, results are in fact significant on all the trees, based on the distribution of p-values or t-statistics. This differs sharply from the finding using independent contrasts, where λ was, in effect, forced to equal 1. Failing to fit scaling parameters such as λ can potentially lead you to miss interesting findings! And we can say with confidence that phylogenetic uncertainty does not affect the conclusions from these analyses; these two variables are related to one another for each tree.

It is worth noting that in some cases, when λ is very close to 0 or 1, the scripts can crash due to an optimization error. If this happens across your tree block and you have good reason, based on the estimates of λ for the traits on a subset of trees, to expect that λ is close to 0 or 1, simply set it a fixed value in the analysis, rather than estimating it.

Correlation and Regression Models in BayesTraits

Start by downloading the program BayesTraits (http://www.evolution.reading.ac.uk/BayesTraits.html). An installation is not necessary.

BayesTraits is available for different operating systems; here, I describe the Mac version.

For this example, you will need two files. The first file is the tree file, which is the same as used above (treeblock_example.nex.ber.nex.coq.nex). The second file is a data file. We need to slightly alter the file used above, specifically by removing the headers and saving as a tab-delimited text file. Do that, and rename the file "edu_data_nh.txt".

Now, put both the BayesTraits executable and the two files that are needed into the same directory in order to run the analysis. You can either copy the data files into the BayesTraits directory or copy the BayesTraits executable into the folder where the data files are.

Open a command line (e.g. click on Start -> Run and type "cmd," if you use Windows or open the "terminal" window on a Mac). On a PC, navigate to the folder containing the necessary files for this example.

BayesTraits must be started by declaring the program file on the command line, followed by two additional files: the tree file and the data file (in that order). E.g., you might see something really hideous like this on your command line:

```
charlie-nunns-macbook-pro:  ./BayesTraits
./treeblock_example.nex.ber.nex.coq.nex ./edu_data_nh.txt
```

The "`./BayesTraits`" is the path to the program. The "`treeblock_example.nex.ber.nex.coq.nex`" is the path to the tree file. And the "`./edu_data_nh.txt`" is the path to the data file. Note that these paths are simply separated by spaces. On a mac, you can also simply drag the files directly to the command line (the paths will be much longer, but it works...). Here, on my Mac, I navigated to the folder with the 3 files using the `setwd()` command, and then used the "./" command in front of each file so that the terminal window "knows" to look in the folder to which I navigated.

If everything worked, you should now see a menu that consists of 6 items.

```
Rand Seed  1286833159
Please Select the model of evolution to use.
1)    MultiState.
2)    Discrete: Independent
3)    Discrete: Dependant
4)    Continuous: Random Walk (Model A)
5)    Continuous: Directional (Model B)
6)    Continuous: Regression
```

Ignore the "Rand Seed," which relates to the random number generator.

In this case, let's try a correlation model under a "random walk," which can be implemented with option 4 (you will need to consult the manual and additional sources to understand all the options). Type 4, and you will next see:

```
Please Select the analysis method to use.
1)    Maximum Likelihood.
2)    MCMC
```

This allows you to choose if you want to use maximum likelihood or Bayesian (MCMC) approaches for the regression analysis. With a tree block, it is more appropriate to use the Bayesian option; in this case, the model will use MCMC to select parameters of the statistical model, while also randomly selecting a tree from the 200 that you will provide to the program. You can then integrate the results across the MCMC samples.

So, type "2". The following text (or something like it) should appear on the screen, summarizing all parameters and settings:

```
Options:
Model:                          Continuous Random Walk
Tree File Name:                 /Users/charlienunn/Desktop/learn
r/treeblock_example.nex.ber.nex.coq.nex
Data File Name:                 /Users/charlienunn/Desktop/learn
r/edu_data_nh.txt
Log File Name:                  /Users/charlienunn/Desktop/learn
r/edu_data_nh.txt.log.txt
Summary:                        False
Analysis Type:                  MCMC
Sample Period:                  100
Iterations:                     5050000
Burn in:                        50000
Rate Dev:                       2.000000
No of Rates:                    2
Test for trait correlation:     True
Kappa                           Not in use
Delta                           Not in use
Lambda                          Not in use
Restrictions:
    alpha-1                     None
    alpha-2                     None
Prior Information:
    Prior Categories:           100
    alpha-1                     uniform -100.00 100.00
    alpha-2                     uniform -100.00 100.00
Tree Information
     Trees:                     1000
     Taxa:                      35
     Sites:                     2
>
```

The output will be written to a file called "`edu_data_nh.txt.log.txt`", which will be found in the folder with your input files. You can change the name of the output file using the log-file command "lf", followed by the name of the new file you'd like to use (it will create it, e.g. "`lf ./output`").

The three scaling parameters λ, δ and κ are also displayed, and they can be either fixed (e.g. by typing "`lambda 0.5`"), estimated (e.g., type "`lambda`" or "`la`"), or left at the default value 1 (printed as "`Not in use`", or type "`lambda 1`").

Whenever you change a setting, you can type “info” to confirm that the change is in effect. For example, after typing “`lambda 0.5`” and then “`info`”, the lambda line should look as follows.

```
Lambda                                  0.500000
```

Now, it is time to run the analyses.

Let’s estimate lambda, but leave the other scaling parameters as they are. To do this, type `lambda`. (Note: shortcuts are possible for each of the commands; see the appendix of the BayesTraits manual.) Type "`run`" to conduct the analysis, and after the numbers start scrolling down the screen, type control-C to stop the program before it scrolls too far.

The output should look something like this:

```
Iteration   Tree No     Lh    HMean Alpha Trait 1     Alpha Trait 2     Trait
1 Var Trait 2 Var Trait 1 2 Co-Var  Lambda      Acceptance
50000 866   -16.601041  -16.601041  0.493571    0.447726    0.009342
      0.010336    0.008437    1.000000    0.000000
50100 866   -16.601041  -16.601041  0.493571    0.447726    0.009342
      0.010336    0.008437    1.000000    0.000000
50200 866   -16.601041  -16.601041  0.493571    0.447726    0.009342
      0.010336    0.008437    1.000000    0.000000
50300 866   -16.601041  -16.601041  0.493571    0.447726    0.009342
      0.010336    0.008437    1.000000    0.000000
50400 866   -16.601041  -16.601041  0.493571    0.447726    0.009342
      0.010336    0.008437    1.000000    0.000000
50500 866   -20.497089  -18.284196  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.010000
50600 866   -20.497089  -19.096296  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000
50700 866   -20.497089  -19.438317  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000
50800 866   -20.497089  -19.638404  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000
50900 866   -20.497089  -19.772204  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000
51000 866   -20.497089  -19.868768  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000
51100 866   -20.497089  -19.942056  0.576454    1.127474    0.009515
      0.012360    0.009267    1.000000    0.000000

etc.
```

Pretty hard to follow, but we’ll clean it up. In particular, the text is not perfectly lined up under the headings. Let's open the “output” file in Excel to see the columns of output.

You will see a summary at the top, and then columns A to L will give results related several key output variables for each MCMC sample, including the tree number used (“`Tree No`”), likelihood (“`Lh`”), the variance of the traits, their covariance, and an estimate of phylogenetic signal (“`Lambda`”).

In the last column is a header called “Acceptance”, and before going any further, let’s take a close look at this number across MCMC samples (i.e., down the rows, where the iteration number is shown in the far left). You can see that it is very high, on the order of 0.8. According to the BayesTraits manual, we need to boost that number up to around 0.2 to 0.4. Ideally, averaged across that column, the average would come out somewhere in this range, say around 0.3. So, let’s restart the analysis, and learn how to tweak a parameter - `ratedev` - that will increase or decrease the acceptance rate.

Restart the program by using the up-arrow, retyping the code from above, or dragging the files - whatever works best for you:

```
charlie-nunns-macbook-pro:  ./BayesTraits
./treeblock_example.nex.ber.nex.coq.nex ./edu_data_nh.txt
```

When you get to the point of setting parameters, such as lambda, we will set ratedev, as follows:

```
ratedev 10
```

Moving `ratedev` up will decrease the acceptance rate.

Be sure to set lambda 1, and restart the program. Does the acceptance rate go down? Eyeballing the rightmost column of output in your terminal window, does it look to average about 0.25? If so, let the program run until it stops; if not, repeat, using larger or smaller values of `ratedev` until it appears to have an acceptance rate around 0.25 (just eyeball based on the last column of data in the window).

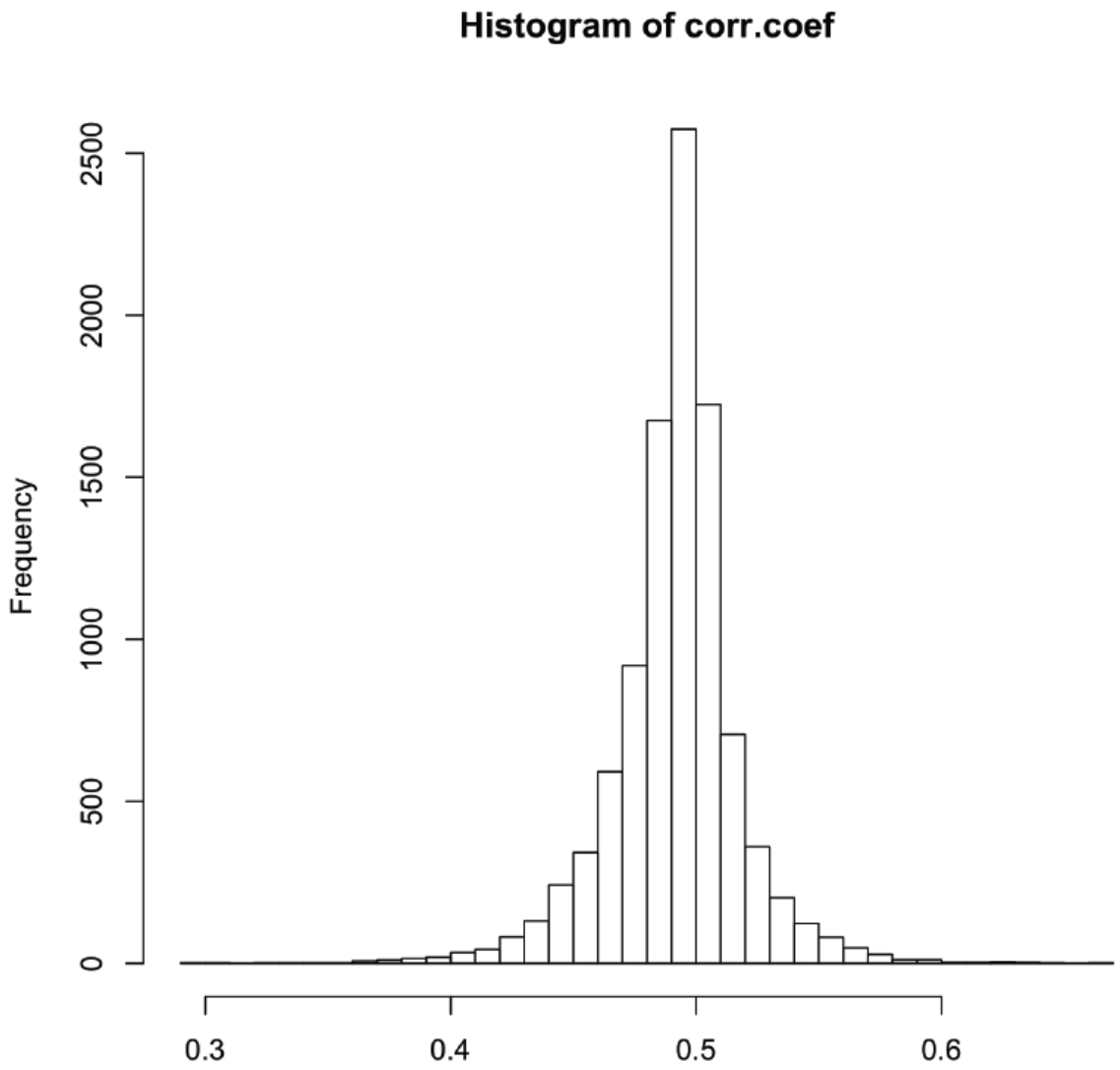
FYI, I found that ratedev of 20 seemed to work pretty well. The program will take a long time to run, so be patient... ~5 minutes for this dataset depending on the computer you are using. The default number of iterations it probably too high - you can reduce the run time by setting it to something like 1,050,000, e.g. type: `it 1050000`

When the program is finished, open it in Excel or some other spreadsheet program. You could also strip out the first summary of the text file in a text editor, and then open the file in R.

The next step is very easy. You can simply obtain means for the different parameters of interest, such as λ, or 95% “credible intervals,” or count the percentage of the time that a coefficient is positive – the choice is really up to you, depending on how you want to make your statistical inferences (there is a lot of flexibility with Bayesian output, such as this). It is also often useful to present distributions of values as histograms.

Let’s take an example with the correlation coefficient, which can be calculated across the MCMC sample in a spreadsheet program by, for each row, dividing the covariance by the square roots of each of the variances. Using R, I found that the mean correlation between the two simulated traits is 0.49. This estimate is slightly lower than the value

used in the simulation of the data (r=0.6), which can be seen graphically with the hist() function:



In this correlational model, the estimate of λ ranges from 0.09 to 0.97, with a mean of 0.61. This again suggests that phylogeny is important, and that independent contrasts, with its assumption of λ=1, is inappropriate for these data.

Good luck, and please remember that this lab is only intended as an initial introduction. There is much more to these tests, and many sources for starting to learn more, including:

Freckleton, R. P., P. H. Harvey, and M. Pagel. 2002. Phylogenetic analysis and comparative data: A test and review of evidence. American Naturalist 160:712-726.

Freckleton, R. 2009. The seven deadly sins of comparative analysis. Journal of Evolutionary Biology.

Garland, T., P. E. Midford, and A. R. Ives. 1999. An introduction to phylogenetically based statistical methods, with a new method for confidence intervals on ancestral values. American Zoologist 39:374-388.

Garland, T., A. F. Bennett, and E. L. Rezende. 2005. Phylogenetic approaches in comparative physiology. Journal of Experimental Biology 208:3015-3035.

Nunn, C. L., and R. A. Barton. 2001. Comparative methods for studying primate adaptation and allometry. Evolutionary Anthroplogy 10:81-98.

Nunn, C. L. 2011. The Comparative Approach in Evolutionary Anthropology and Biology, University of Chicago Press (in press).

Finally, a variety of wikis and websites - in addition to this one - provide advice on running phylogenetic analyses in R. Check out, for example, the "R-phylo wiki", or the wiki associated with the Bodega Bay Workshop for Applied Phylogenetics. Brian O'Meara has produced a great CRAN Task View on "Phylogenetics, Especially Comparative Methods." Or better yet, consider taking the Bodega workshop, or the one that I run - the AnthroTree Workshop.