Chapter 9
Integrating R with Other Phylogenetic and Functional Trait Analytical Software

9.1 Objectives

The objectives of this chapter are to quickly cover how to use R to interface with analytical software that is written in other programming languages. The chapter will focus primarily on integrating R commands with the software Phylocom written in C [14]. I have chosen to focus on this program because it carries out many of the types of analyses covered in this book. Although I have focused on this particular program, the general principles of how to call other programs from R and integrate them into your R-based analyses remain the same.

9.2 Background: The Development of Eco-Informatics Tools for Phylogenetic- and Functional Trait-Based Ecology

In other places in this book, we have discussed the long history of performing phylogenetic and functional diversity analyses in ecology in some cases dating back nearly a century. Despite this long history, only recently has the ecological literature seen a dramatic increase in phylogenetically and functionally based analyses. Although conceptual advances can explain some of this, I would argue that the majority of this increase is due to the development of freely available analytical and eco-informatics tools. Indeed over the course of approximately 5 years, community ecology went from a discipline that rarely generated phylogenetic trees and had few analytical tools to analyze those trees where available to one where phylogenetic trees, though often crude, could be estimated for entire communities within seconds or minutes and analyzed with powerful software. I think it would be hard to underestimate the impact of this development on ecology in general and plant community ecology in particular.

Recently developed analytical and eco-informatics tools have had a large role in the increasing the number of phylogenetically and, to a lesser extent, functionally

N.G. Swenson, *Functional and Phylogenetic Ecology in R*, Use R!,
DOI 10.1007/978-1-4614-9542-0_9, © Springer Science+Business Media New York 2014

based analyses in ecology and the most widely utilized of these tools are coded in C. Although these C programs are open source, ecologists who are not familiar with C and perhaps more fluent in R have found it difficult to utilize or modify the analyses available in the C programs. For example, the two most widely used C programs in phylogenetic community ecology are Phylocom and Phylomatic [14, 56]. Ecologists not accustomed to using command lines have often found these programs difficult to understand or those ecologists not fluent in C have found it difficult to change the source code in these programs to run additional analyses that are not a "canned" function. Thus, when R packages such as *picante* were derived, many phylogenetic community ecology analyses became unlocked to the large number of R using ecologists. One issue though that these users have run into is that the original C program is faster for null model analyses for a series of reasons of which may be the use of `for()` loops in R. For this reason I frequently encounter researchers who say an analysis is "impossible" in R using the available packages when I know it is easily possible in the original C programs. One solution to this issue is writing faster R code, but an alternative option is to use R to communicate with the C program. This approach may increase speed, but it also has the benefit of perhaps allowing you to run analyses that you cannot code (yet) in R. Further, it may allow you to pull out additional information regarding your analyses that the original C program may not allow. For example, the C program Phylocom runs a series of null models and for phylogenetic alpha diversity metrics will produce a standardized effect size value and *P*-value and for phylogenetic beta diversity will only produce a standardized effect size value. In both cases, it is not easy or possible for the user to see the null distribution and in the case of phylogenetic beta diversity it isn't possible to obtain a *P*-value.

In the following sections I will describe how to use R to communicate with the Phylocom program written in C. The goal here will be to simply understand how to call another program from R. After we have mastered this task and seen how to run many phylogenetic and functional diversity analyses in Phylocom while in R, I will discuss how to use R to "unlock" parts of Phylocom to produce all null model analyses and how to seamlessly integrate your R-based analyses with Phylocom.

9.3 Phylocom

Many attribute the rise in phylogenetic analyses of communities to a paper written by Cam Webb in 2000 in The American Naturalist regarding the phylogenetic structure of tree assemblages in Borneo [28]. While this paper was very interesting and did provide the original formulations of the Net Relatedness Index and Nearest Taxon Index, the article in and of itself was not the reason for the tidal wave of phylogenetic community ecology that arrived in the subsequent decade. Rather, it was the eco-informatics tools developed by Webb and colleagues [28] to generate and analyze phylogenetic trees for the species in plant communities that were made freely available to the research community. The tools they developed were and are

available in the C programs Phylocom and Phylomatic. In this section we will discuss how to integrate Phylocom and R and we will cover Phylomatic in the next section. Phylocom is the main analytical software produced by Webb and colleagues to quantify the phylogenetic diversity of communities and to perform comparative methods such as independent contrasts and measures of phylogenetic signal. This program is still frequently utilized by researchers, but has been forgotten or avoided by researchers comfortable only in R. My goal here is to simply demonstrate how this powerful program can be accessed and utilized by calling it from R. We will begin with quantifying phylogenetic and functional diversity metrics with null models, follow this with quantifying independent contrasts and phylogenetic signal, and end with a demonstration of how to randomize data in R and provide that data to Phylocom to produce null distributions. Before we do any of this though, you will need to download and unzip the latest version of Phylocom from https://www.phylodiversity.net/phylocom.

9.3.1 *Quantifying Phylogenetic and Functional Diversity and Dispersion in Phylocom*

Now that we have downloaded Phylocom onto our computer and unzipped the directory, we are ready to begin interfacing it with R. First, for simplicity, let us set our working directory in R to the same directory on our computer that holds the Phylocom program. For Mac users this is the "mac" directory in the Phylocom directory. For PC users this is the "w32" directory in the Phylocom directory. Note that the PC version is 32 bit, which can cause memory issues for large datasets. I am a Mac user so I will set my working directory as follows:

```
> setwd("/Users/swenson/phylocom-4.2/mac")
```

We will now write some example files to these directories that we can use for analysis. We will use the example Phylocom phylogeny, community data, and trait files from the *picante* package.

```
> library(picante)

> data(phylocom)
```

First, we extract the example phylogeny and write it to our working directory in a Newick file format using `write.tree()`.

```
> my.phylo <- phylocom$phylo

> write.tree(my.phylo, "my.phylo.txt")
```

Second, we extract the example community data matrix and write it to our working directory in the Phylocom format where each row is a species in a particular

community with the first column being the community name, the second column being the abundance of that species in that community, and the third column being the species name. It is critical that this file is sorted by the first column, that a species does not occupy more than one row per community, and that the species names perfectly match the names of the species in the phylogeny. These first two requirements are handled by the `writesample()` function in *picante* and the third requirement is not a concern here given we are using an example dataset, but should be checked by you or using the `treedata()` function in *geiger*.

```
> my.sample <- phylocom$sample

> writesample(my.sample, "my.sample.txt")
```

Last, we extract the example trait data matrix and write it to our working directory. Again it is critical that the names in the trait data match those in the phylogeny and community data. The first row in Phylocom trait data matrices contains information regarding the type of trait data (e.g., continuous, categorical, etc.). In this instance our trait data is continuous so this row will include a "3" as the type with the word "type" in the first column. The second row provides the name of each trait in the dataset with the word "name" in the first column. The remaining rows in the trait data matrix contain species names in the first column and trait values in the remaining columns. This format can be produced by using the `writetraits()` function in *picante*.

```
> my.traits <- phylocom$traits

> writetraits(my.traits, "my.traits.txt")
```

If you now look in your working directory you will see these files have been written to your hard drive.

```
> list.files(getwd())
```

You can open these files with a text editor to examine their format and to quickly see if there are any obvious errors that may have occurred. If the files look fine, then we are ready to proceed with some simple analyses using Phylocom from R.

The following commands will differ slightly depending on whether you are using a Mac operating system or Windows. As a Mac user I will be using the `system()` function, but if you are a Windows/PC user, you can simply replace `system()` with the `shell()` function. These functions are used to call a program that can be run in your terminal or command line. For example, we can first use R and `system()` to call Phylocom.

```
> system("./phylocom")
```

The description of the Phylocom software is printed to my R console. Here I can see all of the functions available in this program and how to call them, but first a couple of quick points about how I used the `system()` function. First you will note that I

used quotes around the program that I am calling. So if you had a program called SuperProgram and the executable was in your working directory, you would replace "./phylocom" with "./SuperProgram." Second you will notice that I had a "./" in front of the program name. This is because I have not placed Phylocom in my path on my Mac and you likely have not so that the program is not universally accessibly by your computer. Thus, unless we set the program into our path (see Phylocom user manual) we can simply use "./" prior to the program name for Mac OSX and Unix operating systems. For Windows users the "./" is not required.

Now that we can call the Phylocom program from R, let us first calculate the Faith's Index to discover how to use Phylocom. Like many command line programs, Phylocom works by first telling the command line the program you want to use (i.e., phylocom), then the command or method you would like to use in that program, and finally perhaps some options may be selected. The program we want to use is phylocom and the method we would like to use is "pd" since that is the command for calculating Faith's Index in Phylocom. We follow these two commands by telling the program the name of our phylogeny file using the "-f" switch and the name of our community data file using the "-s" switch.

```
> system("./phylocom pd -f my.phylo.txt -s
my.sample.txt")
```

The program will look for your phylogeny and community data files in your working directory, so please make sure they are there. You should see a matrix print on your R console with five columns and seven rows. The first row gives the column names and each row is a community in your community data matrix. The species richness in each community is given in the "ntaxa" column, the Faith's Index is given in the "PD" column, the summed branch lengths for the entire phylogeny is given in the "treeBL" column, and the Faith's Index represented as a proportion of the summed branch lengths of the entire tree is given in the "propTreeBL" column. If you wanted to actually save this output and not have it only printed to your R console, you could tell Phylocom to write a .txt file to your working directory with the output using a ">" followed by the file name you want to assign the output.

```
> system("./phylocom pd -f my.phylo.txt -s
my.sample.txt > my.faiths.output.txt")
```

You should now see that file in your working directory and you can read it into R using `read.table()` if you wanted to do further analyses or plotting. Next we can calculate the mean pairwise phylogenetic distance and the mean nearest phylogenetic neighbor and perform a null modeling analysis simultaneously in Phylocom using the "comstruct" command. We will again provide the information regarding where to find the phylogeny and community data files, but we will also use the "-m" switch to inform the program what null model to utilize and the "-r" switch to tell the program how many randomizations to run.

```
> system("./phylocom comstruct -f my.phylo.txt -s
my.sample.txt -m 0 -r 99")
```

The output should print out on your R console, but if you wanted you could save it to your hard drive if you used a ">" followed by the file name. We only ran 99 randomizations but this could easily be switched to 999 or higher. We used "-m 0" to indicate we wanted to use null model type "0" which shuffles the names of species on the phylogeny. An independent swap of the community data matrix null model could be invoked by using "-m 3" instead. If we examine the output we see again, there is one row per community in our dataset. The species richness is the "ntaxa" column, the mean pairwise phylogenetic distance is the "MPD" column, the mean of the null distribution of MPD values is in the "MPD.rnd" column, and the standard deviation of that null distribution is in the "MPD.sd" column. The next column is named NRI for Net Relatedness Index. Though here we should pause to consider a small difference between Phylocom and R output that is of major importance. First recall that the NRI is a standardized effect size (S.E.S.) for the MPD metric. In the *picante* R package in the code and the equations we used in Chap. 6, we defined that the S.E.S. is the observed value minus the mean of the null distribution and that value divided by the standard deviation. Now if we consider the output from Phylocom we see that the NRI value is *not* the (MPD − MPD.rnd)/MPD.sd. Rather it is this value multiplied by negative one. Therefore in Phylocom when the observed MPD is larger than the mean of the null distribution, the NRI value is negative. In R when the observed MPD is larger than the mean of the null distribution, the NRI is positive. The multiplication by negative one in Phylocom is due to this calculation being used in the original work by Webb [28] that introduced the NRI and NTI metrics. Thus, it is critical to know whether a researcher has calculated his or her NRI or NTI value using Phylocom or the *picante* package because a negative value in one program has the opposite meaning in the other. It is not completely uncommon for me to see this issue come up in the papers of colleagues where the exact opposite inference has been made. So please be cautious. The best method is to simply look at the observed value and the mean of the null distribution and then consider the S.E.S. value. By doing this you can see whether a S.E.S. value is being multiplied by negative one or not. Now that we have discussed that issue we can return to the results on our screen. The next two columns indicate where the observed value lands in the null distribution. The first reports how many null values are greater and the second column reports how many null values are lower. These rank values can be used to estimate *P*-values. The next six columns are in a similar format except they concern the mean nearest taxon distance and the Nearest Taxon Index (NTI; again multiplied by negative one). The last column reports the number of randomizations used in the null model. This number and the type of null model are also reported in the very first row of the output.

The NRI and NTI analyses above only considered the species as present or absent, but the metrics can be weighted using abundance by using the "-a" switch.

```
> system("./phylocom comstruct -f my.phylo.txt -s
my.sample.txt -m 0 -r 99 -a")
```

Lastly, we can output all of the individual random MPD and MNTD by adding a "-v" to the end of this line of code. This permits the user to examine the null distributions themselves.

Now we are ready to compute phylogenetic beta diversity using Phylocom via R. Specifically, we can calculate the phylogenetic D_{pw}, D_{pw}', D_{nn}, and D_{nn}' metrics. We will start by not running any null model analyses and use the D_{pw} metric, which is calculated using the "comdist" command in Phylocom.

```
> system("./phylocom comdist -f my.phylo.txt -s
my.sample.txt")
```

The result is a matrix where the community names are on the rows and columns and the values are the pairwise phylogenetic dissimilarity between each set of communities. The diagonal values correspond to the MPD values. We can again weight this D_{pw} calculation by abundance to produce the D_{pw}' values by adding the "-a" switch.

```
> system("./phylocom comdist -f my.phylo.txt -s
my.sample.txt -a")
```

The D_{nn} and D_{nn}' values for our communities can be calculated using the "icomdist" function in Phylocom.

```
> system("./phylocom comdistnt -f my.phylo.txt -s
my.sample.txt")
> system("./phylocom comdistnt -f my.phylo.txt -s
my.sample.txt -a")
```

The result is a matrix where the community names are on the rows and columns and the values are the nearest neighbor phylogenetic dissimilarity between each set of communities. The diagonal values correspond to the MNTD values. Next we can perform a null modeling analysis for each of these metrics again by specifying the number of randomizations using the "-r" switch and the type of null model using the "-m" switch. Again we will choose 99 randomizations and the name shuffling null model "0." Lastly, we will use the "-n" switch to tell Phylocom to run a null model. If this is not included, the null model analyses will not be performed.

```
> system("./phylocom comdist -f my.phylo.txt -s
my.sample.txt -r 99 -m 0 -n")
```

We see that four matrices have been printed out to our R console. The first matrix contains the observed D_{pw} values. The second matrix contains the mean of the null distribution, the third matrix contains the standard deviation of the null distribution, and the fourth matrix contains the S.E.S. value for the D_{pw} metric again multiplied by negative one such that negative values indicate higher than expected phylogenetic beta diversity and positive values indicate lower than expected phylogenetic beta diversity. This code can be modified to use abundance using the "-a" switch and the "comdistnt" function for D_{nn}, but we will forgo those analyses for now. Also recall that you do not have to output the results to your R console. You can simply write them to your hard drive as text files using ">" followed by the new file name.

A final community phylogenetic function that we will address in Phylocom concerns whether there is more species in a community from a particular internal node in the phylogeny than expected from randomizing species names on the tips of the phylogeny. To my knowledge, this function is not yet available in R. So for the time being we can run this test in Phylocom using the "nodesigl" function.

```
> system("./phylocom nodesigl -f my.phylo.txt -s
my.sample.txt")
```

The output is a table with seven columns. The first column indicates the name of the community. The second and third columns indicate the number and name of each internal node in the phylogeny. The fourth column contains the number of taxa from that node found in that community. The fifth column indicates the median expected value and the sixth column gives the rank of the observed number of species in the null distribution of expected number of species. Note in some cases, such as the root, the null distribution contains only one possible value and therefore should not be considered. The final column contains a "+," "–," or no value to indicate the observed number of species from that node in that community is higher than expected, lower than expected, or no different from expected, respectively. This analysis can also be output as a series of Nexus phylogenies, one per community, with the node labels when there is more or less species than expected for that community. This Nexus file can be generated using the "nodesig" command and read into a program such as Mesquite for visualization.

```
> system("./phylocom nodesig -f my.phylo.txt -s
my.sample.txt > nodesig4mesquite.nex")
```

We now transition to the measurement of functional diversity using Phylocom via R. The command "comtrait" calculates the main measures of functional diversity found in Phylocom. The command requires a "-t" switch to indicate the trait file instead of the "-f" switch to indicate the phylogeny file. It also requires a "-x" switch to indicate what type of functional diversity you would like to calculate. A value of "-x 1" calculates the variance of each trait in each community, "-x 2" calculates the mean pairwise distance for each trait, "-x 3" calculates the mean nearest trait neighbor distances, and finally "-x 4" calculates the range of each trait in each community. The function also runs a null model using the same "-m" and "-r" switches we used above. To provide an example, we will calculate the variance of each trait in each community using a name shuffling null model with 99 randomizations.

```
> system("./phylocom comtrait -t my.traits.txt -s
my.sample.txt -x 1 -m 0 -r 99")
```

The output is a matrix with each row being a unique trait by community combination sorted by trait order in the traits file. The first column is the trait name, the second column is the community name, the third column is the species richness, the fourth column is the mean trait value, the fifth column is the functional diversity metric (variance in this case), the sixth column is the mean of the null distribution, the seventh column is the standard deviation of the null distribution, the eighth

column is the S.E.S. for the metric, the ninth and tenth columns are the ranks, and the final column is the number or randomizations. Unfortunately, in this function the S.E.S. value is *not* multiplied by negative one further adding to the confusion, so once again some diligence is needed when considering your results.

9.3.2 Comparative Analyses in Phylocom

Although Phylocom is primarily used for the analysis of phylogenetic diversity in communities, it also has a powerful comparative methods module that can implement the calculation of phylogenetic signal and phylogenetically independent contrasts (PICs). Conveniently these measures can be calculated simultaneously using one simple command called "aot" in Phylocom.

```
> system("./phylocom aot -f my.phylo.txt -t
my.traits.txt")
```

The first output printed to your R console is the phylogenetic signal for each trait. Here phylogenetic signal is being computed as the variance of the node-level contrasts compared to a null distribution of variances generated by shuffling names of species on the phylogeny 999 times. The first column of the output is the trait name, the second column is the number of taxa analyzed, the third column is the observed variance in contrast values, and the fourth and fifth columns are the rank values of the observed variance in the null distributions. For a one-tailed test, values in the fourth column less than or equal to 50 indicate significant phylogenetic signal when using 999 randomizations.

The second component of the output reports the PIC correlation values between trait one and the other traits with the first column being the second trait being compared to the first, the second column is the correlation, the third column is the number of positive contrasts, and the final column is the number of contrasts. In order to calculate a *P*-value for the correlation, one can use a Pearson's correlation coefficient statistical table with the degrees of freedom equaling the number of contrasts minus one.

9.3.3 Interfacing R and Phylocom for Null Modeling

In the previous two subsections, we have discussed how to run Phylocom via R, but this alone may not be a very powerful way to interface R and Phylocom. After all this approach doesn't really take advantage of R *per se*. Rather it just prevents you from having to open up a shell or terminal window and may allow you to do a few additional analyses not currently available in R. Thus we may want to quickly consider how R could be more usefully interfaced with a program like Phylocom. One potential approach that may be immediately useful is to output manipulated files

from R to be used in Phylocom. In particular, we have already seen that many of the null modeling functions currently available in R are quite slow, whereas they are generally much faster in Phylocom. Further, we have seen that at least in the case of beta diversity there are no "canned" R functions for null model analyses, whereas in Phylocom there are null model analyses but they do not provide the null distribution or the estimated *P*-value. Here I will demonstrate a methodology for solving this particular problem, but the general approach should be useful for solving a number of tasks where one generates or modifies files in R that could be input into another program on your computer.

We being by identifying the problem and the task we want accomplished. We want to conduct a null model analysis for the phylogenetic beta diversity metric D_{pw}, where we can obtain all null values and therefore calculate a *P*-value. We know we can conduct a null model analysis of D_{pw} in Phylocom using the "comdist" command, but we also know that we cannot obtain the null distribution or an estimated *P*-value. A potential solution is to provide Phylocom 999 random phylogenies one at a time and have it calculate the D_{pw} values using each random phylogeny. We could then use all of the output files to generate null distributions and estimate *P*-values. For brevity, we will take this approach, but using only nine random phylogenies. We start by reading into R our original phylogeny while making sure we have our working directory set to where our Phylocom program is located.

```
> my.phylo <- read.tree("my.phylo.txt")
```

We will keep our observed community data matrix in the working directory because we will only use R to manipulate the phylogenetic tree. Specifically, we will write a simple loop that randomizes the names on the phylogeny, writes the randomized phylogeny to our working directory, and tells Phylocom to calculate the D_{pw} values using the randomized phylogeny and repeats this process nine times.

```
> for(i in 1:9){

     ## Shuffling names on the phylogeny.
     my.phylo$tip.label <- sample(my.phylo$tip.label)

     ## Writing phylogeny to working directory.
     write.tree(my.phylo, "tmp.phylogeny.txt")

     ## We use the system() command to tell Phylocom
     ## to run the comdist function using the
     ## randomized phylogeny tmp.phylogeny.txt. We
     ## want to output a new file for each new
     ## iteration of the randomization and we
     ## therefore need a new name for each iteration.
     ## We therefore will a paste() function that
     ## provides the command to Phylocom but changes
     ## the 'i' value in the output file name to match
     ## the 'i' in the for() loop.
     system(paste("./phylocom comdist -s my.sample.txt
     -f tmp.phylogeny.txt > ", i, ".txt", sep = ""))

}
```

The result is a series of nine files written to your working directory named "1.txt," "2.txt,"…, "9.txt." These files contain the random D_{pw} values generated for each iteration of the null model. The goal now is to read those nine files that contain matrices into R as an array with nine levels in the *z*-dimension. This can be accomplished by first reading in the observed output file and then combining it with the remaining files with a loop.

```
> tmp <- read.table("1.txt", sep = "\t", row.names = 1,
header = T)

> library(abind)

> for(i in 2:9){

     ## Read in the next random Dpw output
     new.tmp <- read.table(paste(i,".txt",sep = ""),
     sep = "\t", row.names = 1,
     header = T)

     ## Bind the new.tmp output to the existing array
     tmp <- abind(tmp, new.tmp, along = 3)

}
```

We now have an array with the row and column numbers equaling the number of communities and the number of levels in the *z*-dimension equals the number of randomizations, which in this case is 9. We can now calculate the mean and standard deviation of the null distributions using an `apply()` function.

```
> nulls.mean <- apply(tmp, c(1:2), mean, na.rm = T)

> nulls.sd <- apply(tmp, c(1:2), sd, na.rm = T)
```

These values along with the observed value can be used to calculate a S.E.S. value as described in Chap. 6, but such values can be provided by Phylocom. What are not available in Phylocom are the actual null distribution and the estimated *P*-value. Using the above approach that integrates R with Phylocom, we can plot the null distributions for community comparisons. For example, we can plot a histogram of the null values for the community 1 versus community 3 comparison.

```
> hist(tmp[1, 3, ]
```

We can also calculate an estimated *P*-value for each comparison by first calculating the observed D_{pw} values and reading them into R.

```
> system("./phylocom comdist -f my.phylo.txt -s
my.sample.txt > observed.dpw.txt")

> obs <- read.table("observed.dpw.txt", sep = "\t",
row.names = 1, header = T)
```

We add the observed results to the top layer of the array of null values using the `abind()` function.

```
> obs.nulls <- abind(obs, tmp)
```

Using this we can now calculate the estimated *P*-values as we have done before in Chap. 6.

```
>array(dim = dim(obs.nulls),
t(apply(apply(obs.nulls,c(1, 2),rank), 3, t)))[ , , 1]
```

We now have the rank of the observed value in the null distribution from which we can estimate a *P*-value (see Chap. 6). This same approach could be used to calculate the D_{pw}', D_{nn}, and D_{nn}' metrics by using the "-a" switch in the above code calling Phylocom from R and/or using the "comdistnt" command in the above code calling Phylocom from R. As you can see, it can be fairly easy to integrate R with a program like Phylocom where you would like to sequentially feed the external program output from R. I have used a `for()` loop for this purpose, but the computational speed is not a problem because the task I am looping in R, randomizing names on a phylogeny, is a fairly simple process. Thus, when you are faced with potentially outputting a lot of files from R that you need to feed into an external program, the process can be automated using R freeing you to spend your time on other tasks.

9.4 Conclusions

In the fields of ecology and evolution we often run across interesting analytical approaches presented in an article that we think would be useful or our own research. For some that are comfortable with command line or are fluent in multiple programming languages, these published programs are easily utilized. For the majority of the potential users, though, using these programs is not an option or extremely difficult. Thankfully, the convergence of the ecological and evolutionary research communities on R has removed many of these obstacles since many of the published programs now in this literature are written in R and can be used by many researchers and this will only continue and become more common as we increasingly expect our students to become fluent in R. There are still cases, however, where the program of interest is written in another language making it difficult for a user to run or manipulate the analyses. The goal of this chapter was to demonstrate how such an obstacle can be overcome while remaining in R. I used one very popular C program, Phylocom, frequently used by ecologists for phylogenetic and functional diversity analyses and for comparative analyses, but the general concepts and approaches should apply to other programs you may be interested in written in C or other languages.

9.5 Exercises

1. Calculate the Faith's Index for your example data in Phylocom via R.
2. Noting that Phylocom does not have a randomization for Faith's Index at this time, write a phylogenetic null model shuffling species names that interfaces R with Phylocom to calculate 99 random Faith's Indices for each community and write the results of each iteration of the null model to your hard drive.
3. Take the randomization results from number 2 above, read them into R, and calculate a standardized effect size and *P*-value.
4. Repeat numbers 2 and 3, but this time randomize the community data matrix using an independent swap null model and do not manipulate the phylogenetic tree.
5. Simulate two trait datasets using your example phylogeny and the `fastBM()` function in the *phytools* package.
6. Write the two trait datasets in a single file to your hard drive assuring that it is formatted as required by Phylocom.
7. Via R, run the "aot" command in Phylocom using your example phylogeny and two simulated trait datasets.