phytools 2.0: An updated R ecosystem for phylogenetic comparative methods (and other things)

Liam J. Revell[1, 2]

[1]**Department of Biology, University of Massachusetts Boston, Boston, MA, USA**
[2]**Facultad de Ciencias, Universidad Católica de la Santísima Concepción, Concepción, Chile**

Corresponding author:
Liam J. Revell[1,2]

Email address: liam.revell@umb.edu

ABSTRACT

Phylogenetic comparative methods comprise the general endeavor of using an estimated phylogenetic tree (or set of trees) to make secondary inferences: about trait evolution, diversification dynamics, biogeography, community ecology, and a wide range of other phenomena or processes. Over the past ten years or so, the *phytools* R package (Revell 2012) has grown to become one of the most important research tools for phylogenetic comparative analysis. *phytools* is a diverse contributed R library now consisting of hundreds of different functions covering a variety of methods and purposes in phylogenetic biology. As of the time of writing, *phytools* included functionality for fitting models of trait evolution, for reconstructing ancestral states, for studying diversification on trees, and for visualizing phylogenies, comparative data, and fitted models, as well numerous other tasks related to phylogenetic biology. Here, I describe some significant features of and recent updates to *phytools*, while also illustrating several popular workflows of the *phytools* computational software.

1 INTRODUCTION

Phylogenetic trees are the directed graphs used to represent historical relationships among a set of operational taxa that are thought to have arisen via a process of descent with modification and branching (Felsenstein 2004; Harmon 2019). Operational taxa in a reconstructed phylogenetic tree might be gene copies, paralogous and orthologous members of a gene family, viral sequences, whole genomes, human cultural groups, or biological species (Yang 2014). According to its broadest definition, the phylogenetic comparative method corresponds to the general activity of using a known or (most often) estimated phylogenetic tree to learn something *else* (apart from the relationships indicated by the tree) about the evolutionary process or past, the contemporary ecology, the biogeographic history, or the origins via diversification, of the particular taxa of our estimated tree (Harvey and Pagel 1991; Felsenstein 2004; Nunn 2011; Harmon 2019; Revell and Harmon 2022).

Modern phylogenetic comparative methods are not new. Perhaps the most important article in the development of the phylogenetic approach to comparative biology (Felsenstein 1985) was first authored nearly 40 years ago, and was even the subject of a recent retrospective (Huey et al. 2019). Nonetheless, it's fair to say that phylogenetic comparative methods have seen a relatively impressive expansion and diversification over the past two decades (e.g., Butler and King 2004; Felsenstein 2005; O'Meara et al. 2006; Maddison et al. 2007; Hohenlohe and Arnold 2008; Revell and Collar 2009; Morlon et al. 2010; Stadler 2011; Etienne and Haegeman 2012; Goldberg and Igić 2012; Beaulieu et al. 2013; Rabosky 2014; Uyeda and Harmon 2014; Beaulieu and O'Meara 2016; Revell 2021, and many others; reviewed in Garamszegi 2014; Harmon 2019; Revell and Harmon 2022). This has included the development of new approaches for studying the generating processes of trees (that is, speciation and extinction), the relationship between phenotypic traits and species diversification, and a range of techniques for investigating heterogeneity in the evolutionary process across the branches and clades of the tree of life

(Harmon 2019; Revell and Harmon 2022).

Phylogenetic comparative methods have also begun to be applied extensively outside of their traditional domain of evolutionary research. In particular, phylogenies and the comparative method have made recent appearances in studies on infectious disease epidemiology, virology, cancer biology, sociolinguistics, biological anthropology, molecular genomics, and community ecology, among other disciplines (e.g., Moura et al. 2016; Baele et al. 2018; Bentz et al. 2018; Beale et al. 2019; Bushman et al. 2019; Sánchez-Busó et al. 2019; Valles-Colomer et al. 2019; Freitas et al. 2020; Jezovit et al. 2020; Blinkhorn and Grove 2021; McLaughlin et al. 2022; Pepke and Eisenberg 2022; Pozzi et al. 2022; Compton et al. 2023; Mifsud et al. 2023; Van Borm et al. 2023, and many others).

The scientific computing environment R (R Core Team 2022) is widely-used in biological research. One of the major advantages that R provides is that it empowers computational scientists and independent developers to build functionality on top of the basic R platform. This functionality often takes the form of what are called contributed R packages: libraries of related functions built by individuals or research collaboratives not part of the core R development team. The growth of importance of R in phylogenetic biology stems entirely from contributed R package. Among these, the most important core function libraries are *ape* (Paradis et al. 2004; Popescu et al. 2012; Paradis and Schliep 2019), *geiger* (Harmon et al. 2008; Pennell et al. 2014), *phangorn* (Schliep 2011), and my package, *phytools* (Revell 2012).

phytools is an R function library dedicated primarily to phylogenetic comparative analysis, but also including approaches and methodologies in a range of other domains of phylogenetic biology – not restricted to, but especially, visualization. The original article describing *phytools* is now more than 10 years old, and though I recently published a more comprehensive book on the subject of phylogenetic comparative methods in the R environment (Revell and Harmon 2022), I nonetheless felt that it was time to provide a briefer (although this article is by no means brief) update of *phytools* specifically – for the primary scientific literature. This is the main purpose of the current article.

The *phytools* library has now grown to be very large – consisting of hundreds of functions, a documentation manual that's over 200 pages in length, and tens of thousands of lines of computer code. As such, I thought it would be most useful to compactly summarize some of the functionality of the *phytools* R package in a few different areas, but each time provide a small set of more detailed example analytical workflows for the current "2.0" version of the *phytools* package.

2 OVERVIEW

The *phytools* R package contains functionality in a diversity of different research areas of phylogenetics and phylogenetic biology. Rather than attempt a comprehensive survey of this functionality here, what I've elected to do instead is briefly review a smaller number of methodological areas, and then illustrate each of these with multiple analysis workflows – including the corresponding R code that can be used to reproduce the analysis and results presented.

My hope is that this article will serve as more than the typical software note placeholder for *phytools*, and may instead aid R phylogenetic users, both new and old, to be inspired to apply some of the methodologies illustrated herein to their own questions and data.

3 INSTALLING AND LOADING *PHYTOOLS*

This article assumes that readers already have some familiarity with the R computing environment, and have previously installed contributed R packages. Nonetheless, to get started using *phytools*, the easiest way to install the package locally is by using the R base function called `install.packages`, which will download and install *phytools* from its CRAN page. (CRAN is an acronym for Comprehensive R Archive Network: a network of mirror repositories used both to archive and distribute R and contributed R packages.)

After opening a new interactive R session, this can be done as follows.

```
install.packages("phytools")
```

Readers undertaking phylogenetic analysis in the R environment for the first time will note that when we ask R to install *phytools*, several other R packages are also downloaded and installed automatically. These are packages upon which *phytools* depends – meaning that *phytools* uses one or multiple functions

exported by each of these packages in its own internal R code. More will be said later about the dependency relationship between *phytools* and other packages of the R and R phylogenetic ecosystems.

Having installed *phytools*, if we'd like to proceed and use it in an interactive R session, we'd normally load it. (Loading an R package simply makes the names of the functions of that package visible and available in our current R session.) This can be done using the base R function `library`.

```
library(phytools)
```

```
## Loading required package: ape
```

```
## Loading required package: maps
```

```
packageVersion("phytools")
```

```
## [1] '1.5.12'
```

(I expect that this version number with be updated to `phytools 2.0` by the time this article is published!) The *phytools* package is now loaded.

4 DISCRETE CHARACTERS

The *phytools* R library now contains a wide range of different methods and models for the analysis of *discrete character evolution* on trees. For example, *phytools* can be used to fit and plot an extended M*k* model (*phytools* function `fitMk`, Lewis 2001; Harmon 2019), it can fit Pagel's correlational binary trait evolution model (`fitPagel`, Pagel 1994), it can be used to perform stochastic character mapping and reconstruct ancestral states under the M*k* and threshold models (`make.simmap`, `simmap`, `ancThresh`, and `rerootingMethod`, Huelsenbeck et al. 2003; Felsenstein 2005, 2012; Revell 2014), it can fit a polymorphic trait evolution model (`fitpolyMk`, Revell and Harmon 2022), it can fit a hidden-rates model (`fitHRM`, Beaulieu et al. 2013), it can compare the rate of discrete character evolution between clades and trees (`fitmultiMk` and `ratebytree`, Revell et al. 2018; Revell and Harmon 2022), and it can simulate discrete character data under multiple models (e.g., `sim.Mk`, `sim.history`, `sim.multiMk`).

In this section, I'll illustrate the use of just a few of the different discrete character methods that have been implemented in the *phytools* software.

4.1 Stochastic character mapping

Perhaps the most important and widely-used discrete character method of *phytools* is a technique referred to as "stochastic character mapping" (Huelsenbeck et al. 2003). Stochastic character mapping is a method in which we randomly sample discrete character histories ("stochastic maps") of our trait on the tree under a specified model.

This can be undertaken in more than one way. An example of a stochastic character mapping analysis could be to first fit (e.g., using Maximum Likelihood) the character transition model (a variant of the M*k* discrete character evolution model of Lewis 2001; also see, Harmon 2019; Revell and Harmon 2022), and then proceed to randomly sample a set of 1,000 (perhaps) random character histories under this model that are consistent with the phenotypic trait observations that we have for the terminal taxa of our tree.

(Other workflows are also popular and possible to undertake within R. For instance, rather than use a single, fixed model of character evolution that's been optimized using Maximum Likelihood, one might instead sample parameters of the evolutionary process from their joint posterior probability distribution using Bayesian MCMC. See Revell and Harmon 2022 for more details.)

To illustrate this approach here, I'll use a discretely-valued, ecological trait for a small phylogeny of centrarchid fishes from Revell and Collar (2009). Since the trait (which we'll refer to as "feeding mode") is binary, meaning that it only takes two levels, there are a total of four possible discrete character (extended M*k*, see Harmon 2019) models: equal back-and-forth transitions between the two character values; different rates; and then the two different irreversible trait evolution models.

phytools now allows us to fit each of these four different models, compare them, and pass the model weights and fitted models directly to our stochastic mapping function. Our function (called `simmap`) will

then proceed to automatically sample stochastic character histories with probabilities that are proportional to each model weight. (Experience *phytools* users may figure out that `simmap` is just a sophisticated wrapper function of `make.simmap` – the traditional method used for undertaking stochastic character mapping in *phytools*.)

For this example, and all subsequent examples of the article, our data have been packaged with the *phytools* library – so we can easily load them in an interactive R session using the base R function `data` as follows.

```
data(sunfish.tree)
data(sunfish.data)
```

For our M*k* model-fitter (which here will be the *phytools* function `fitMk`), and for the other discrete character methods of the *phytools* R package, our input phenotypic trait data will typically takes the form of a character or factor vector. Personally, I prefer to use factors, because in that case we can more easily access the levels assumed by the character through a call of the base R function `levels`.

(In this example our input data consists of a data frame in which the `feeding.mode` column is already coded as a factor. In general, however, had we read this data from an input text file in, for example, comma-separated-value format, R would've created a character, rather than factor, formatted column by default. To adjust this we can set the argument `stringsAsFactors=TRUE` in our file-reading function, which, in that case, might be the base R function `read.csv`.)

```
feeding_mode<-setNames(sunfish.data$feeding.mode,
  rownames(sunfish.data))
levels(feeding_mode)
```

```
## [1] "non"  "pisc"
```

Here we see that our factor vector has two levels: `"non"` and `"pisc"`. These two character levels refer to non-piscivorous and piscivorous fish species in this group. Since R factors have no particular character limit on their levels, let's update our data to use these more descriptive names: once again using the function `levels`.

(`levels` is an odd R method in that it can serve both as an *extractor* function, that pulls out the levels of a factor – as well as acting as an assignment or *replacement* function, in which the levels of the factor are updated. When we adjust our factor levels for `feeding_mode`, we're using `levels` in this latter fashion.)

```
levels(feeding_mode)<-c("non-piscivorous",
  "piscivorous")
levels(feeding_mode)
```

```
## [1] "non-piscivorous" "piscivorous"
```

Now we're ready to proceed and fit our models. To do so, I'll use the *phytools* function `fitMk` and fit a total of four models, as previously indicated: `"ER"`, the equal rates model; `"ARD"`, the all-rates-different model; and, lastly, the two different irreversible models – one in which non-piscivory can evolve to piscivory, but not the reverse; and a second in which precisely the opposite is true.

For our two irreversible models, we'll tell `fitMk` how to construct the model by creating and supplying what I'll refer to as a "design matrix" for each model that we want to fit. This design matrix should be of dimensions $k \times k$, for k levels of the trait, with integer values in the positions of the matrix corresponding to allowed transitions, and zeros elsewhere. (We use different non-zero integer value for each rate that we want to permit to assume a different value in our fitted model.)

Since our $k = 2$, this is very easy; however, the same principle would apply to any value of k (see Revell and Harmon 2022 for more examples).

```
ER.model<-fitMk(sunfish.tree,feeding_mode,
  model="ER")
ARD.model<-fitMk(sunfish.tree,feeding_mode,
  model="ARD")
Irr1.model<-fitMk(sunfish.tree,feeding_mode,
  model=matrix(c(0,1,0,0),2,2,byrow=TRUE))
Irr2.model<-fitMk(sunfish.tree,feeding_mode,
  model=matrix(c(0,0,1,0),2,2,byrow=TRUE))
```

Having fit our four models, we can also compare them to see which is best-supported by our data. To do so, I'll use a generic `anova` function call as follows. `anova` will print the results of our model comparison; however, it's important that we also assign the value returned by `anova` to a new object. In my example, I'll call this object `sunfish.aov`, but the name is arbitrary.

```
sunfish.aov<-anova(ER.model,Irr1.model,
  Irr2.model,ARD.model)
```

```
##                log(L) d.f.      AIC    weight
## ER.model    -13.07453    1 28.14906 0.3486479
## Irr1.model  -12.98820    1 27.97640 0.3800846
## Irr2.model  -14.20032    1 30.40064 0.1130998
## ARD.model   -12.86494    2 29.72987 0.1581677
```

This table shows each of our fitted model names, their log-likelihoods, the number of parameters estimated, a value of the Akaike Information Criterion (AIC), and the Akaike model weights. Smaller values of AIC indicate better support for the corresponding model – taking into account its parameter complexity. Model weights can be interpreted as the "weight of evidence" favoring each of our four trait evolution hypotheses.

Based on this analysis, we might conclude that the first irreversible model (`Irr1.model`), in which non-piscivory can evolve to piscivory, but not the reverse, is best supported; however, we have a very similar weight of evidence favoring the equal-rates model (`ER.model`).

With the result of our `anova` call in hand (as the `sunfish.aov` object), we're ready to pass it on directly to *phytools*' new generic `simmap` method. By design, doing so will tell `simmap` to generate stochastic character maps under each of our four models with relative frequencies that are equal to the weight of evidence supporting of each model. (If we preferred, we could've generated stochastic character maps for just the best-supported of our four models. Using the `simmap` generic method, this would be done either by supplying our `anova` result and setting the argument `weighted=FALSE` – or simply by passing our favored M*k* model directly to the function!)

```
sunfish.simmap<-simmap(sunfish.aov,nsim=1000)
sunfish.simmap
```

```
## 1000 phylogenetic trees with mapped discrete characters
```

In spite of the significant number of stochastic simulations involved, this analysis should run fairly quickly (obviously, depending on the speed of our computer). In part this is because we saved computation time by circumventing the need to re-estimate our M*k* transition matrix, $\mathbf{Q}$, separately for each sampled model. An additional advantage of this approach is that it's also allowed us to (partly) account for variation in our modeled process of evolution that's due to uncertainty in model selection.

Figure 1 shows a set of six, randomly chosen stochastic character histories for our trait (feeding mode) on our input tree. Readers should see that each of these are consistent with our observed value of the binary trait at the tips of the tree, but that each differs one from the other in the specific hypothesis of trait evolution that it represents.

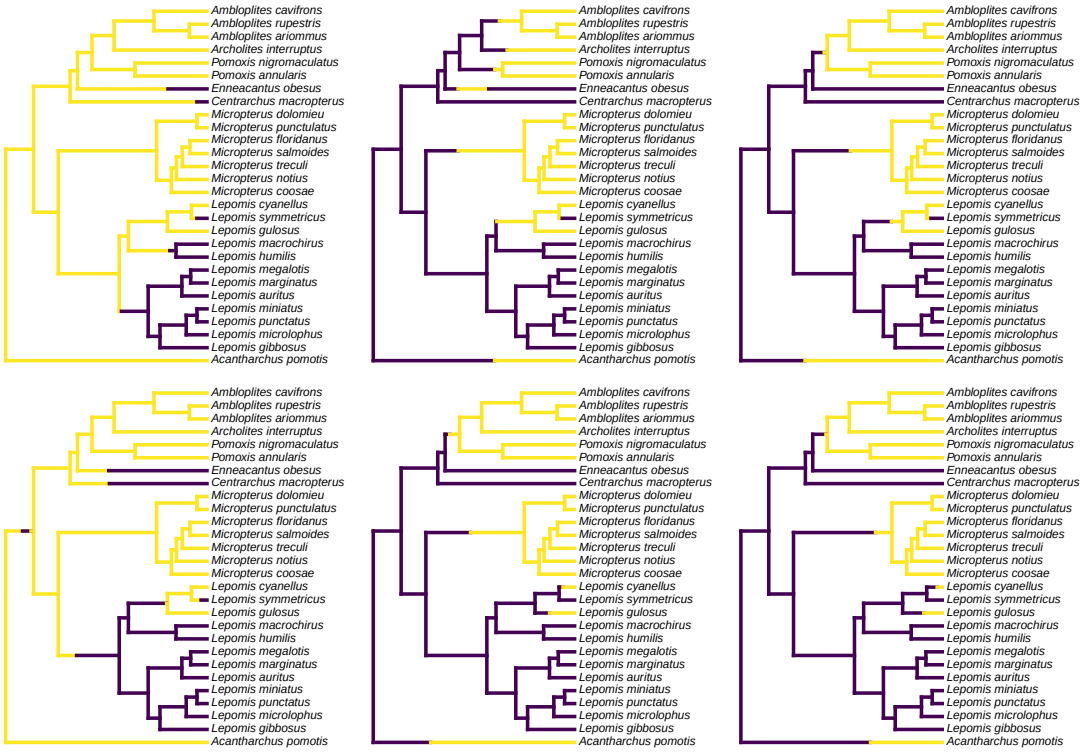

Figure 1. Six randomly chosen stochastic character maps of feeding mode (non-piscivorous, in dark blue, vs. piscivorous) on a phylogeny of 28 centrarchid fish species. Stochastic character mapping involves randomly sampling character histories that are consistent with our tip data in proportion to their probability under a model. In this case, histories were sampled under the set of four alternative M*k* models of a binary trait, with relative frequencies proportional to the weight of evidence supporting each model. The tree and data are modified from Revell and Collar (2009). See main text for additional details.

```
cols<-setNames(viridisLite::viridis(n=2),
  levels(feeding_mode))
par(mfrow=c(2,3))
plot(sample(sunfish.simmap,6),ftype="i",
  fsize=0.6,colors=cols,offset=0.2)
```

To create my color palette for plotting I used another contributed R package that we haven't seen yet called *viridisLite* by Garnier et al. (2022). To replicate Figure 1 exactly, users should first install *viridisLite* from CRAN by running `install.packages("viridisLite")` – but they do not need to load it. (Calling the contributed package function using the double colon syntax, `::`, takes care of that – i.e., `viridisLite::viridis`.)

Although Figure 1 already gives us a general sense of the uncertainty of our ancestral character history on the tree for our trait, most commonly we don't want to simply graph a subset (or all) of our stochastically mapped trees. Typically, instead, we'd first summarize our stochastic character maps (in multiple ways), and then proceed to plot or analyze these summarized findings.

Perhaps most often, *phytools* users undertaking stochastic character mapping will compute the posterior probabilities of each value of the character trait at each internal node of the tree. In *phytools*, these values can be obtained using the generic `summary` method for our object class, which is then easily plotted as follows.

```
plot(summary(sunfish.simmap),ftype="i",
  fsize=0.7,colors=cols,cex=c(0.6,0.3))
legend("topleft",levels(feeding_mode),pch=21,
```

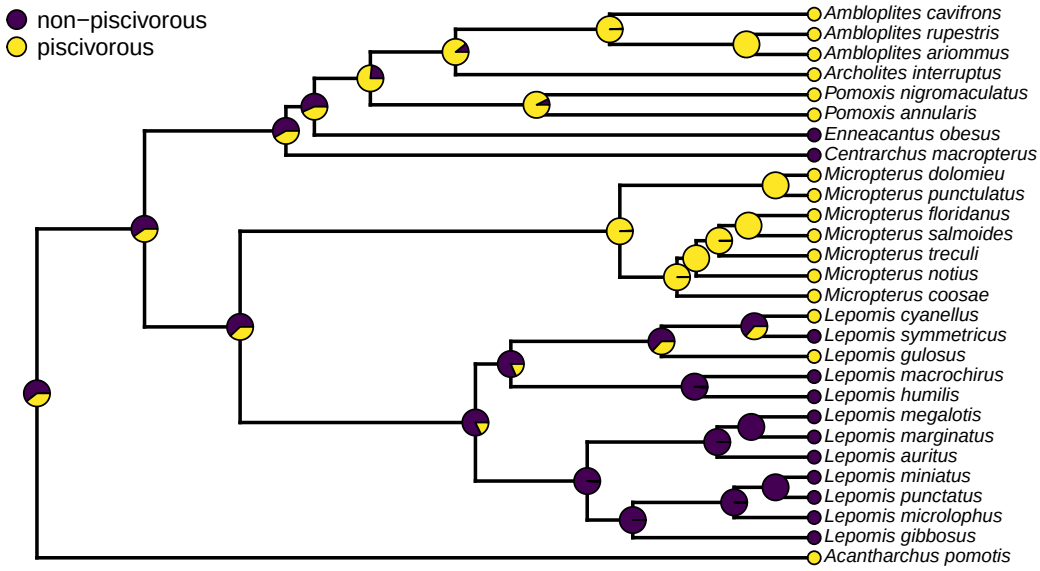

Figure 2. Posterior probabilities at each ancestral node of the centrarchid tree of Figure 1 from stochastic character mapping using model weights to sample across four different extended M*k* trait evolution models. See main text for more details.

```
  pt.cex=1.5,pt.bg=cols,bty="n",cex=0.8)
```

A correct interpretation of the graph of Figure 2 is that it shows the observed discrete character states (at the tips of the tree) and the posterior probabilities from stochastic mapping that each internal node is in each state – all while integrating over our four different transition models in proportion to the weight of evidence in support of each model!

In addition to node probabilities, *phytools* users undertaking a stochastic character mapping analysis are often interested in the number of changes of each type that are implied by the evolutionary process and our data. The procedure of stochastic mapping samples full character histories (not just states or probabilities at nodes), and can thus be deployed to produce estimates of the posterior probability distribution of the character changes of each type on specific edges, in specific clades, or on the entire tree, conditioning on our sampled model or models.

To obtain these distributions, we'll first call the generic method `density` which (when given an object from stochastic mapping) computes the relative frequency distribution of changes of each type over the whole tree. We can then graph these distributions (remember, our character is binary, so there are only two types of character state change: non-piscivorous → piscivorous, and the reverse) using a different generic `plot` method, as follows.

```
sunfish.density<-density(sunfish.simmap)
sunfish.density
```

```
##
## Distribution of changes from stochastic mapping:
##  non-piscivorous->piscivorous        piscivorous->non-piscivorous
##  Min.   :0   Min.   :0
##  Median :5   Median :2
##  Mean   :4.13    Mean   :2.24
##  Max.   :10  Max.   :9
##
## 95% HPD interval(non-piscivorous->piscivorous): [0, 8]
## 95% HPD interval(piscivorous->non-piscivorous): [0, 6]
```

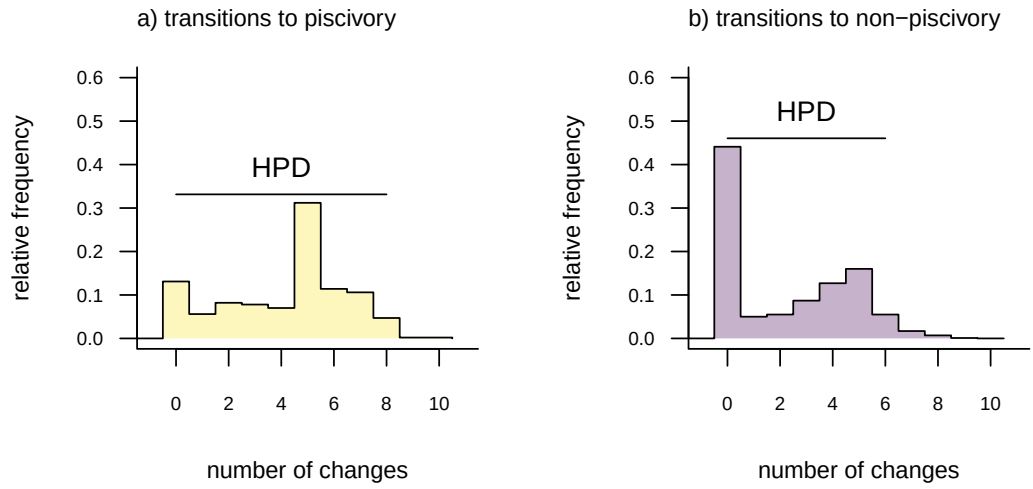

Figure 3. Posterior probability distributions of changes from either a) non-piscivory to piscivory, or b) piscivory to non-piscivory, obtained from an analysis of stochastic mapping. HPD indicates the 95% high probability density interval for changes of each type. See main text for additional details.

```
par(mfrow=c(1,2),las=1,cex.axis=0.7,
  cex.lab=0.8)
COLS<-setNames(cols[2:1],sunfish.density$trans)
plot(sunfish.density,ylim=c(0,0.6),
  transition=names(COLS)[1],colors=COLS[1],
  main="")
mtext("a) transitions to piscivory",line=1,
  adj=0,cex=0.8)
plot(sunfish.density,ylim=c(0,0.6),
  transition=names(COLS)[2],colors=COLS[2],
  main="")
mtext("b) transitions to non-piscivory",
  line=1,adj=0,cex=0.8)
```

The distributions shown in Figure 3 give the relative frequencies of changes of each type across our set of mapped histories, as well as Bayesian 95% high probability density (HPD) intervals calculated using the R pakage *coda* (Plummer et al. 2006).

An interesting attribute of the character state change distributions for our centrarchid feeding mode data is that they are both markedly bi-modal. This is due, in part, to our specific procedure of model-averaging in which we sampled both reversible and irreversible character evolution models in proportion to their weights, and isn't something we would've seen had we chosen to analyze just one model or the other. (Recall that the weight of evidence was highly similar between our equal-rates model and the irreversible model in which piscivory is acquired from non-piscivory, but never the reverse.)

Lastly, in addition to these analyses, *phytools* also makes it quite straightforward to visualize the posterior probabilities of each of the two trait conditions not only at nodes, but also also along the branches of the phylogeny. This is accomplished using the *phytools* function `densityMap` (Revell 2013), which creates a graph showing the probability density of stochastic histories in each of our mapped states. By design, in *phytools* this object can be first created (using the `densityMap` function), updated (using the method `setMap` to adjust our color palette for plotting), and then graphed (using a generic `plot` method that was created for this specific object class).

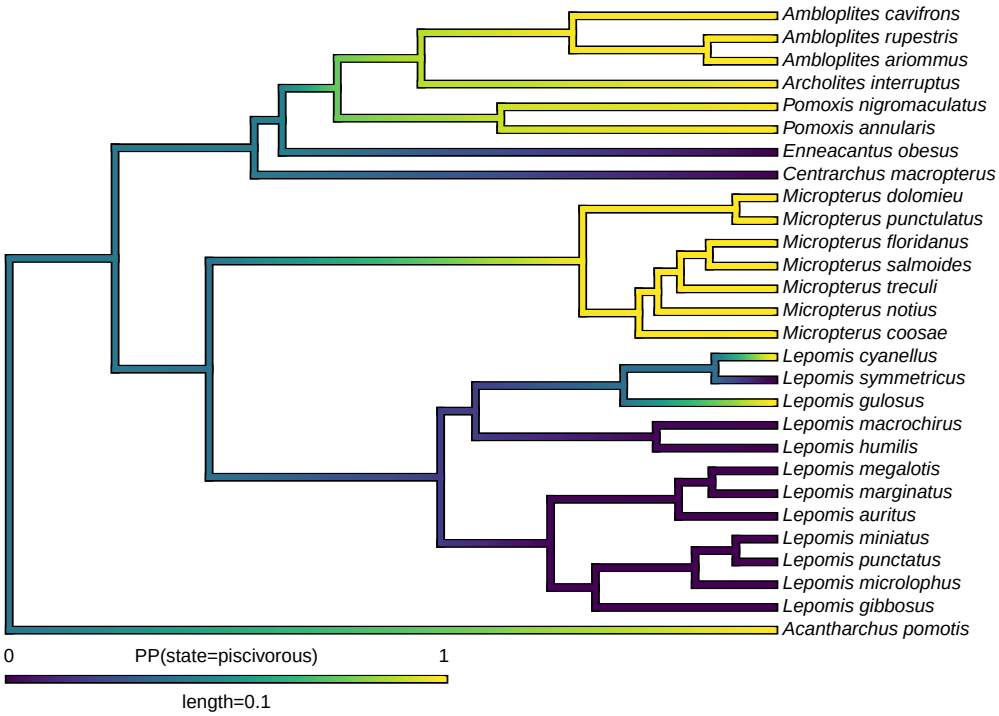

Figure 4. Posterior probability density of each of the two character levels, piscivory and non-piscivory, mapped along the edges of a tree of centrarchid fishes. See main text for more details.

```
sunfish.densityMap<-densityMap(sunfish.simmap,
  plot=FALSE,res=1000)
sunfish.densityMap
```

```
## Object of class "densityMap" containing:
##
## (1) A phylogenetic tree with 28 tips and 27 internal nodes.
##
## (2) The mapped posterior density of a discrete binary character
##     with states (non-piscivorous, piscivorous).
```

```
sunfish.densityMap<-setMap(sunfish.densityMap,
  viridisLite::viridis(n=10))
plot(sunfish.densityMap,lwd=3,outline=TRUE,fsize=0.7,
  legend=0.1)
```

4.2 The polymorphic trait evolution model

Another important, but much more recently-added, tool in the *phytools* R package is a method (called `fitpolyMk`) that's designed to fit a discrete character evolution model to trait data containing intraspecific polymorphism (Revell and Harmon 2022). In this case, our model is one in which an evolutionary transition from (say) character state a to character state b must first pass through the intermediate polymorphic condition of $a + b$. This model starts off very simply – but will become increasingly complicated for increasing numbers of monomorphic conditions of our trait. Not only that, but as soon as we have more than two monomorphic states, we must also consider whether our character is evolving in an ordered (Figure 5a) or unordered (Figure 5b) fashion (Revell and Harmon 2022).

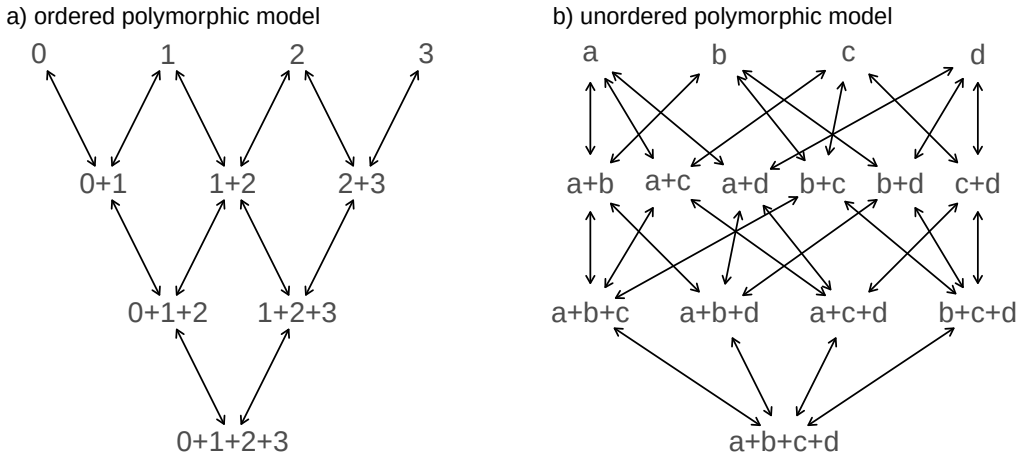

Figure 5. Example structures of two alternative polymorphic trait evolution models for characters with four monomorphic conditions: a) an ordered model with states 0 to 3; b) an unordered model, with states *a*, *b*, *c*, and *d*. See main text for additional details.

Figure 5 shows the general structure of an ordered and unordered polymorphic trait evolution model – both for the same number of monomorphic conditions of our trait (four).

```
par(mfrow=c(1,2))
graph.polyMk(k=4,ordered=TRUE,states=0:3,
  mar=rep(0.1,4))
mtext("a) ordered polymorphic model",line=-1,
  adj=0.2,cex=0.8)
graph.polyMk(k=4,ordered=FALSE,
  states=letters[1:4],mar=rep(0.1,4),
  spacer=0.15)
mtext("b) unordered polymorphic model",
  line=-1,adj=0.2,cex=0.8)
```

Obviously, the potential parameter complexity of the unordered polymorphic trait evolution model is higher than the ordered model. Since there exists an unordered model that also has all ordered models as a special case, ordered and unordered models can be compared using likelihood-ratio tests (if nested) or information criteria.

To try out our polymorphic trait evolution model, let's use an excellent, recently-published dataset from Halali et al. (2020) consisting of a phylogenetic tree containing 287 Mycalesina butterfly species and data for butterfly habitat use. Halali et al. (2020) coded habitat as a polymorphic trait in which, for example, a species using both "forest" and forest "fringe" habitat would be coded as `"forest+fringe"`. In this case, our polymorphic trait evolution model will assume that to evolve from forest specialization to fringe specialization, a species must first (at least transiently) evolve through the polymorphic condition of using both habitats at once. This seems logical.

The Halali et al. (2020) dataset and tree now come packaged with the *phytools* library, so both can be loaded using the `data` function, just as we've seen already for other datasets of this article.

```
data(butterfly.tree)
data(butterfly.data)
```

Let's begin by inspecting our data.

```
head(butterfly.data)
```

```
##                                    habitat
## Myc_francisca_formosana? forest+fringe+open
## Bic_cooksoni                          open
## Bic_brunnea                         forest
## Bic_jefferyi                  fringe+open
## Bic_auricruda_fulgida               forest
## Bic_smithi_smithi           forest+fringe
```

`fitpolyMk` requires us to separate the different states in each polymorphic condition using the `+` symbol, but does not demand that our states be ordered in a consistent manner. In other words, `a+b` and `b+a` would be considered (properly) to be same polymorphic condition!

As a first preliminary step in our analysis, we can proceed to extract the column of habitat use data (`habitat` in our data frame) as a vector, and then print the different levels that it takes.

```
butterfly.habitat<-setNames(
  butterfly.data$habitat,
  rownames(butterfly.data))
print(levels(butterfly.habitat))
```

```
## [1] "forest"             "forest+fringe"      "forest+fringe+open"
## [4] "fringe"             "fringe+open"        "open"
```

Now, let's proceed to fit our polymorphic trait evolution model to these data.

In this instance, I'll fit a grand total of six different models. (This isn't a comprehensive set of the conceivable models for polymorphic data with these levels, but it seemed like a reasonable selection for illustrative purposes.)

The first three of these models all suppose that the evolution of my discrete character is totally unordered. Among this set, we'll imagine, first, equal transition rates between all monomorphic states or polymorphic conditions. For our second model, we'll permit all possible transition rates between states or state combinations to assume different values. Finally, for our third model we'll assume that the acquisition of polymorphism (or its increase) occurs with one rate, whereas the loss (or decrease) of polymorphism occurs with another, separate rate.

We refer to this last scenario as the "transient model" following Revell and Harmon (2022). The name for this model comes from the general notion that if the rate of loss exceeds the rate of gain, then polymorphism will typically be relatively transient in nature. Since polymorphism tends to be less frequently observed in the types of data that typify many phylogenetic comparative studies, including this model in our set seems like a reasonable idea.

To get our remaining three models, and reach the six total models that I promised at the outset of this section – for each of the three listed above in which character evolution is unordered, we'll simply add a second *ordered* model in which we assume that character evolution for our three monomorphic conditions tends to proceed as follows: *forest* ↔ *fringe* ↔ *open* – not forgetting, of course, about the intermediate polymorphic conditions that occur in between each of these monomorphic states!

To fit our first three models in R, we'll use the function `fitpolyMk` from the *phytools* package as follows.

```
butterfly.ER_unordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="ER")
```

```
##
## This is the design matrix of the fitted model.
## Does it make sense?
##
```

```
##                      forest fringe open
## forest                    0      0    0
## fringe                    0      0    0
## open                      0      0    0
## forest+fringe             1      1    0
## forest+open               1      0    1
## fringe+open               0      1    1
## forest+fringe+open        0      0    0
##                      forest+fringe forest+open fringe+open
## forest                           1           1           0
## fringe                           1           0           1
## open                             0           1           1
## forest+fringe                    0           0           0
## forest+open                      0           0           0
## fringe+open                      0           0           0
## forest+fringe+open               1           1           1
##                      forest+fringe+open
## forest                               0
## fringe                               0
## open                                 0
## forest+fringe                        1
## forest+open                          1
## fringe+open                          1
## forest+fringe+open                   0
```

By default, `fitpolyMk` prints out the design matrix of the model for us to verify. (The design matrix is of dimensions dictated by the number of states and polymorphic conditions of our character, with integers populating the different types of transitions, from row to column, that should be permitted under our model, and zeros indicating unallowed transition types. The specific integer values don't mean anything; however, different integer values imply that the corresponding transitions will be allowed to take place with different rates under our model.)

This can be helpful, because we should find that it corresponds with the design matrix that was discussed under the simpler M*k* model of the previous section – as well as with the graphed models of Figure 5. If we don't want the design matrix to print, though, we can turn off this behavior simply by setting the optional argument `quiet=TRUE`.

Let's do that for our remaining two unordered models.

```
butterfly.ARD_unordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="ARD",quiet=TRUE,
  opt.method="optimParallel",rand_start=TRUE)
butterfly.transient_unordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="transient",quiet=TRUE,
  opt.method="optimParallel",rand_start=TRUE)
```

Astute readers may notice that I added two additional arguments that didn't feature in my previous `fitpolyMk` function call: `opt.method="optimParallel"` and `rand_start=TRUE`. The former tells my optimizer to use the *optimParallel* package (Gerber and Furrer 2019) for optimization. The latter says "choose random starting values." Both of these, and sometimes multiple optimization replicates, may be required to find our Maximum Likelihood solution for these complex models. (In fact, I virtually guarantee it!)

We're not done fitting models yet, but to see how things are going so far, why don't we compare our three fitted models using the generic `anova` method of their object class, as follows.

```
anova(butterfly.ER_unordered,
  butterfly.ARD_unordered,
  butterfly.transient_unordered)
```

```
##                                    log(L) d.f.      AIC       weight
## butterfly.ER_unordered         -355.8122    1 713.6244 1.017751e-19
## butterfly.ARD_unordered        -295.0807   18 626.1614 1.000000e+00
## butterfly.transient_unordered -353.4496    2 710.8991 3.975914e-19
```

This tells us that, just among the three models that we've considered to this point in our analysis, the best-supported by a wide margin is our parameter-rich all-rates-different (`"ARD"`) model.

Now we can proceed to do the same thing, but this time updating the argument value `ordered` to `ordered=TRUE`; however, when we switch from fitting an unordered polymorphic trait evolution model to our ordered model, it suddenly becomes critical that we specify the order levels using the optional function argument `order`. If `order` isn't indicated, `fitpolyMk` will simply assume that our characters are ordered alphanumerically – but this is very rarely likely to be correct! (By chance, it happens to be true of our butterfly dataset. I assigned the argument `order` anyway, just to be safe!)

```
levs<-c("forest","fringe","open")
levs
```

```
## [1] "forest" "fringe" "open"
```

```
butterfly.ER_ordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="ER",ordered=TRUE,order=levs,
  quiet=TRUE)
butterfly.ARD_ordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="ARD",ordered=TRUE,order=levs,
  quiet=TRUE,opt.method="optimParallel",
  rand_start=TRUE)
butterfly.transient_ordered<-fitpolyMk(
  butterfly.tree,butterfly.habitat,
  model="transient",ordered=TRUE,
  order=levs,quiet=TRUE,
  opt.method="optimParallel",
  rand_start=TRUE)
```

Now, with all six models in hand, let's compare them using a second `anova` call as follows. This time I'll save my results from our model comparison to the object `butterfly.aov`.

```
butterfly.aov<-anova(butterfly.ER_ordered,
  butterfly.ER_unordered,
  butterfly.transient_ordered,
  butterfly.transient_unordered,
  butterfly.ARD_ordered,
  butterfly.ARD_unordered)
```

```
##                                    log(L) d.f.      AIC       weight
## butterfly.ER_ordered           -329.0390    1 660.0779 1.472873e-09
## butterfly.ER_unordered         -355.8122    1 713.6244 3.472845e-21
## butterfly.transient_ordered    -329.0205    2 662.0409 5.519508e-10
## butterfly.transient_unordered -353.4496    2 710.8991 1.356691e-20
## butterfly.ARD_ordered          -297.7376   12 619.4753 9.658773e-01
## butterfly.ARD_unordered        -295.0807   18 626.1614 3.412273e-02
```

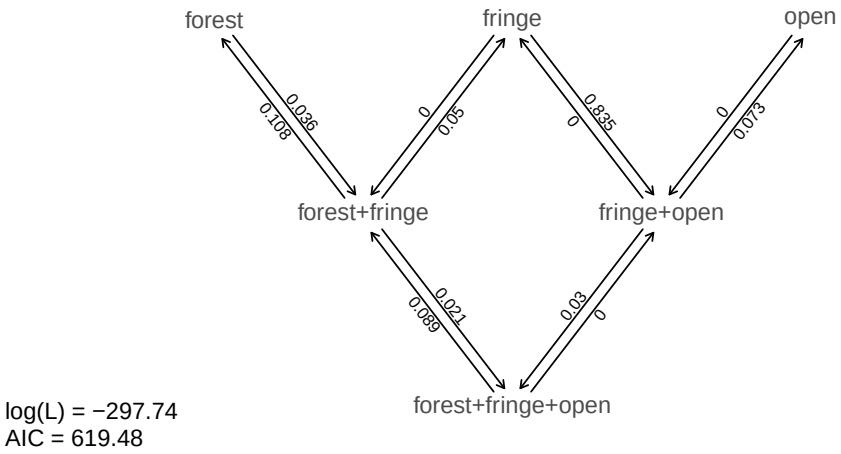

Figure 6. Best-fitting polymorphic trait evolution model for the evolution of habitat use in Mycalesina butterflies.

A quick word of caution to readers is probably merited. These models can be quite difficult to optimize, meaning that it's not inconceivable to imagine that (in spite of our best efforts) `fitpolyMk` hasn't converged on the true Maximum Likelihood solution for one model or another. Although the true best solution may be unknowable (this is why we use numerical optimization to try and ascertain it), common sense can be a valuable defense against very obvious failures of optimization. For instance, had we found that the most complex model (in our case, `butterfly.ARD_unordered`) had a lower likelihood than its nested counterparts (for instance, `butterfly.ARD_ordered`), this would give us very strong cause to believe that one or both models hadn't converged, and that we should perhaps try different random starts or alternative optimization routines to try to find better solutions!

Nonetheless, taking our fitted models at face value, model comparison shows that (among the models in our set) the best supported by far is the ordered, all-rates-different model. *phytools* has a function to graph this model, so let's go ahead and use it!

```
plot(butterfly.ARD_ordered,asp=0.65,
  mar=rep(0.1,4),cex.traits=0.8)
legend("bottomleft",legend=c(
  paste("log(L) =",
  round(logLik(butterfly.ARD_ordered),2)),
  paste("AIC =",
  round(AIC(butterfly.ARD_ordered),2))),
  bty="n",cex=0.8)
```

Just as with our fitted M*k* models from the prior section, we can also pass this model object to our generic stochastic character mapping method, `simmap`. When we do, `simmap` will automatically generate a set of 100 stochastic character maps under our fitted model. (We could've likewise passed `simmap` our `anova` results, just as we did with our `"fitMk"` objects in the centrarchid example, above. In this case, however, nearly all the weight of evidence fell on one model: our ordered, all-rates-different model.)

```
butterfly.simmap<-simmap(butterfly.ARD_ordered)
butterfly.simmap
```

```
## 100 phylogenetic trees with mapped discrete characters
```

Now that we have our stochastically mapped trees, let's compute a summary, just as we did in the prior section.

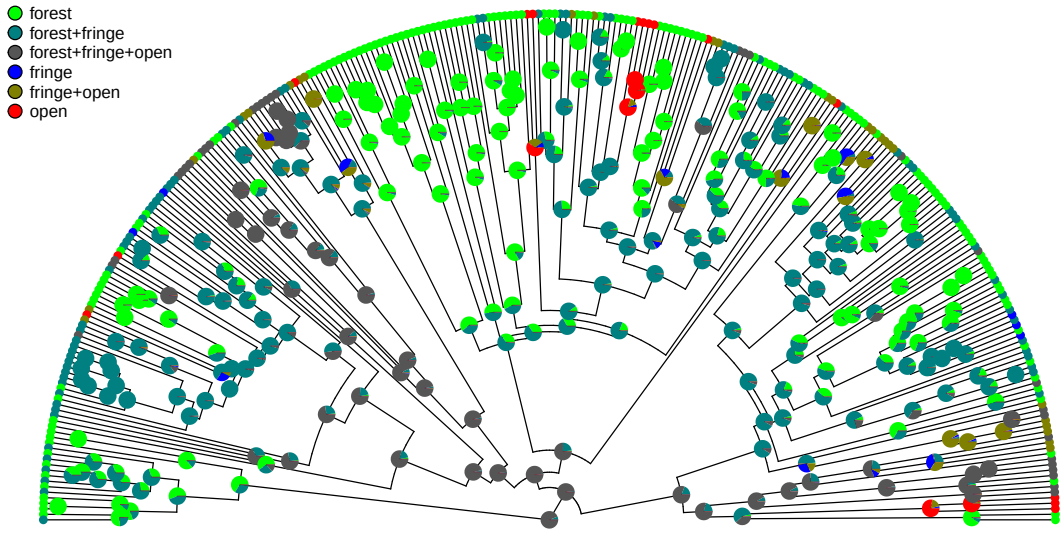

Figure 7. Posterior probabilities of monomorphic or polymorphic conditions at internal nodes from stochastic mapping under an ordered, ARD model of trait evolution. See main text for additional details.

```
butterfly.summary<-summary(butterfly.simmap)
```

Much as we saw earlier, the object from our generic `summary` call can be conveniently plotted using *phytools*. Here I'll use the base graphics function `rgb` to attempt to select colors for plotting that are evenly spaced in a red-green-blue color space in which the "corners" (red, green, and blue) correspond to the three monomorphic states of our data. Does that make sense? (I'm colorblind, so it's hard for me to be sure how the `rgb` color space captures the "intermediacy" of the polymorphic conditions between the corresponding monomorphic states.)

```
hab.cols<-setNames(c(rgb(0,1,0),
  rgb(0,0.5,0.5),rgb(1/3,1/3,1/3),
  rgb(0,0,1),rgb(0.5,0.5,0),
  rgb(1,0,0)),levels(butterfly.habitat))
par(fg="transparent")
plot(butterfly.summary,type="fan",
  ftype="off",colors=hab.cols,
  cex=c(0.4,0.2),part=0.5,lwd=1)
par(fg="black")
legend("topleft",names(hab.cols),pch=21,
  pt.bg=hab.cols,pt.cex=1.5,
  cex=0.8,bty="n")
```

Excellent! Figure 7 shows both the observed (at the tips) and reconstructed (at the internal nodes) marginal posterior probabilities for each of our states and polymorphic conditions.

Lastly, let's graph the posterior distribution of the accumulation of lineages in each state over time, using the *phytools* function `ltt` as follows. (We'll learn more about `ltt` in a subsequent section.) We can even do this while retaining the same color palette as we used for Figure 7.

```
butterfly.ltt<-ltt(butterfly.simmap)
par(mar=c(5.1,4.1,1.1,1.1))
plot(butterfly.ltt,show.total=FALSE,
  bty="n",las=1,cex.axis=0.7,
  cex.lab=0.8,colors=hab.cols)
```

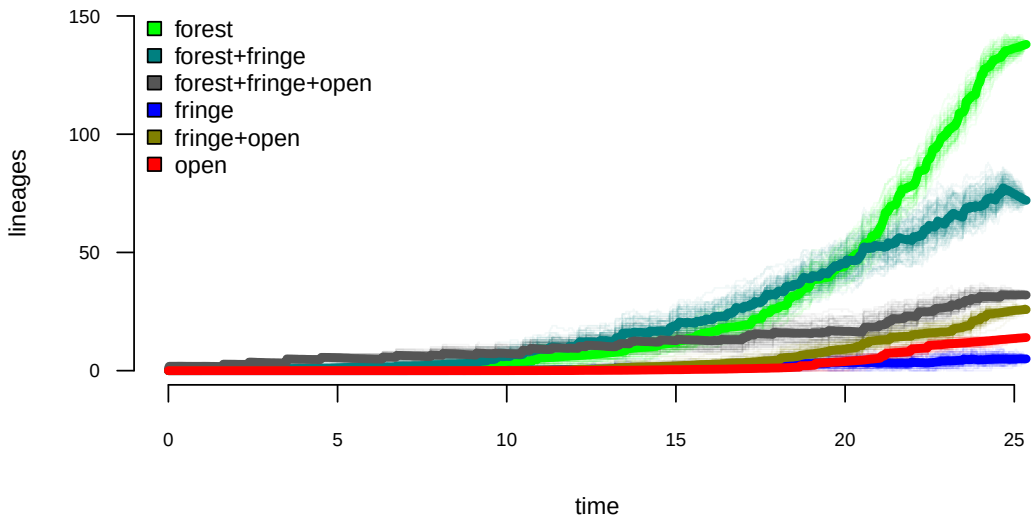

Figure 8. Lineage-through-time plot showing the reconstructed accumulation of lineages in each polymorphic condition or monomorphic state over time, from 100 stochastic character maps. See main text for additional details.

The plot of Figure 8 simultaneously shows the accumulation of lineages in each mono- or polymorphic state, but also the variation attributable to uncertainty in the evolutionary history of our group from our stochastic character maps! (Even though Figure 8 looks very cool – to be fair, this type of graph is only especially meaningful for the situation in which the taxa of our phylogeny have been completely sampled. Though this is unlikely to be true here, I felt that it was nonetheless interesting to demonstrate!)

5 CONTINUOUS CHARACTERS

Numerous continuous trait methods exist in the *phytools* package. For example, *phytools* can be used to measure phylogenetic signal (`phylosig`, Pagel 1999; Blomberg et al. 2003; Revell et al. 2008), it can fit multi-rate Brownian evolution models (`brownie.lite`, `brownieREML`, `evol.rate.mcmc`, `multirateBM`, `ratebytree`, and `rateshift`, O'Meara et al. 2006; Revell et al. 2012, 2018; Revell 2021; Revell and Harmon 2022), it can perform phylogenetic canonical correlation and principal components analysis (`phyl.cca` and `phyl.pca`, Revell and Harrison 2008; Revell 2009), it can reconstruct ancestral states under multiple evolutionary models (`anc.Bayes`, `anc.ML`, `anc.trend`, and `fastAnc`, Schluter et al. 1997; Revell and Harmon 2022), it can use continuous trait data to place a fossil or missing lineage into a reconstructed tree (`locate.fossil` and `locate.yeti`, Felsenstein 2002; Revell et al. 2015), it can fit a multivariate Brownian model with multiple evolutionary correlations on the tree (`evol.vcv` and `evolvcv.lite`, Revell and Collar 2009; Revell et al. 2022), and it can perform various types of continuous character numerical simulation on phylogenies (e.g., `branching.diffusion`, `fastBM`, `sim.corrs`, `sim.rates`).

Here I'll start by illustrating the measurement of phylogenetic signal (`phylosig`), then I'll demonstrate Bayesian ancestral state estimation (`anc.Bayes`). I'll show how to fit a variable-correlation multivariate Brownian trait evolution model (`evolvcv.lite`), and, finally, I'll demonstrate a relatively new multi-rate trait evolution model that uses the estimation technique of penalized likelihood (`multirateBM`).

5.1 Phylogenetic signal

Perhaps the simplest phylogenetic comparative analysis that we could choose to undertake for a continuous trait data in R is the measurement of phylogenetic signal (Pagel 1999; Blomberg et al. 2003; Revell et al. 2008).

Phylogenetic signal has been defined in a number of different ways, but could be considered to be the simple tendency of more closely related species to bear more similarity (one to another) than they do to more distant taxa. Phylogenetic signal can be quantified in various manners, but undoubtedly the two most popular metrics are Blomberg et al.'s (2003) K statistic, and Pagel's (1999) λ. Conveniently, both of these can be calculated using the *phytools* package.

To get started in this undertaking, let's load some data from *phytools* consisting of a phylogenetic tree of elopomorph eels and a data frame of phenotypic traits. Both tree and data were obtained from an article by Collar et al. (2014).

```
data(eel.tree)
data(eel.data)
head(eel.data)
```

```
##                    feed_mode Max_TL_cm
## Albula_vulpes        suction       104
## Anguilla_anguilla    suction        50
## Anguilla_bicolor     suction       120
## Anguilla_japonica    suction       150
## Anguilla_rostrata    suction       152
## Ariosoma_anago       suction        60
```

Having loaded these data, we can next extract one variable from our data array. Phylogenetic signal can be measured for any continuous trait, so we'll use maximum total length: here represented by the column of our data frame called `"Max_TL_cm"`. As is often the case, we'll transform our data to a log scale. (There are multiple reasons log transformations are favored by comparative biologists working on interspecies data. One is that it makes a, say, 10% change equal, regardless of whether it occurs in an elephant or a mouse! See Revell and Harmon 2022 for more details.)

```
eel.lnTL<-setNames(log(eel.data$Max_TL_cm),
  rownames(eel.data))
```

Next, we'll compute a value of the K statistic of Blomberg et al. (2003) using the *phytools* function `phylosig`. `phylosig` calculates K by default (that is, without specifying an argument for `method`), but if I add the argument value `test=TRUE`, `phylosig` will also conduct a statistical test of the measured value of K by comparing it to a null distribution of K obtained by permuting our observed trait values randomly across the tips of the phylogeny.

```
eel.Blomberg_K<-phylosig(eel.tree,eel.lnTL,
  test=TRUE)
eel.Blomberg_K
```

```
##
## Phylogenetic signal K : 0.362879
## P-value (based on 1000 randomizations) : 0.034
```

K has an expected value of 1.0 under Brownian motion (Blomberg et al. 2003). The lower value that we observe here thus indicates less phylogenetic signal than expected under Brownian evolution; whereas a value higher than 1.0 would've indicated more. Our significance test shows us that this value of K, though modest, is nonetheless significantly greater than we'd expect to find in data that were entirely random with respect to the tree!

In addition to Blomberg et al.'s K, *phytools* also can be used to estimate Pagel's (1999) λ statistic. λ measures phylogenetic signal as a scalar multiplier of the correlations of related taxa in our tree (Revell and Harmon 2022). That is to say, if λ has a value less than 1.0, this would indicate that related species in our phylogeny have a lower degree of "autocorrelation" than expected under Brownian evolution. In fact, a value of λ close to zero could be taken to indicate that related species are not phenotypically correlated at all!

We use Maximum Likelihood to find the value of λ that makes our observed data most probable. Since it's possible to compute a likelihood for any allowable value of λ, including $\lambda = 0$, we can very easily proceed to test a null hypothesis of no phylogenetic signal in our data by simply calculating a likelihood ratio in which we compare $\lambda = 0$ to our Maximum Likelihood estimate. Indeed, this is the test performed by *phytools* if `method="lambda"` and `test=TRUE`!

```
eel.Pagel_lambda<-phylosig(eel.tree,eel.lnTL,
  method="lambda",test=TRUE)
eel.Pagel_lambda
```

```
##
## Phylogenetic signal lambda : 0.673735
## logL(lambda) : -54.3016
## LR(lambda=0) : 5.18173
## P-value (based on LR test) : 0.0228256
```

This result tells us that we've found significant phylogenetic signal in our trait by both measures. Although K and λ tend to be correlated, it's entirely possible that we could've found significant K and non-significant λ, or vice versa. This is not a contradiction. The concept of phylogenetic signal is one of phenotypic similarity among related species – but K and λ measure this concept via two entirely different procedures!

Along with the simple calculation of phylogenetic signal, *phytools* also contains several methods to visualize our results. In particular, for Blomberg et al.'s K we can plot the permutation distribution of K alongside our observed measure. For Pagel's λ, we can plot the likelihood surface, our Maximum Likelihood solution, and the likelihood of $\lambda = 0$: the null hypothesis of our statistical tests. Both of these plots are illustrated in Figure 9 for our eel body length data.

```
par(mfrow=c(1,2),cex=0.9)
plot(eel.Blomberg_K,las=1,cex.axis=0.9)
mtext("a)",adj=0,line=1)
plot(eel.Pagel_lambda,bty="n",las=1,
  cex.axis=0.9,xlim=c(0,1.1))
mtext("b)",adj=0,line=1)
```

5.2 Bayesian ancestral state estimation

The *phytools* package contains several different functions for discrete and continuous character ancestral state estimation under multiple models. Earlier, we reviewed the method of stochastic character mapping (Huelsenbeck et al. 2003) which can be an important tool for ancestral state reconstruction of discrete characters.

Among the variety of approaches for ancestral character estimation of continuous characters that are implemented in the *phytools* package is the function `anc.Bayes`. As its name suggests, `anc.Bayes` performs ancestral state estimation using Bayesian MCMC. Just as any proper Bayesian approach should, the implementation of this method allows us to include prior information about the states at internal nodes. Here, I'll illustrate the simplest type of analysis that we can undertake with the function in which I'll simply accept the default node priors and MCMC conditions. `anc.Bayes`, however, will be most useful when we intend to explicitly incorporate prior knowledge about internal nodes of the tree – based on, for instance, observations from the fossil record.

To demonstrate the method, I'll load a dataset (now packaged with *phytools*) that consists of a phylogeny and phenotypic trait information for a set of lizards from the family Cordylidae, originally published by Broeckhoven et al. (2016).

```
data(cordylid.tree)
data(cordylid.data)
head(cordylid.data)
```

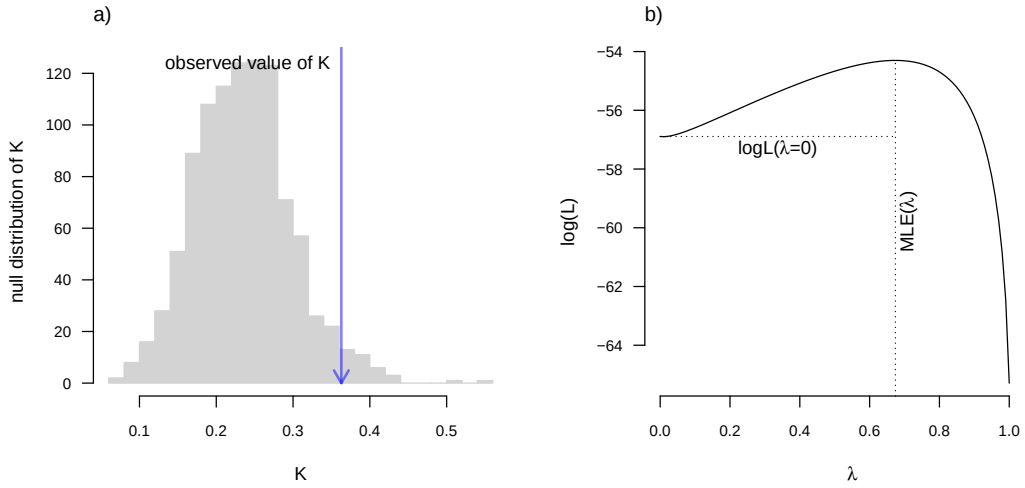

Figure 9. a) Blomberg et al. (2003) measured value of the K statistic for phylogenetic signal, compared to a null distribution of K obtained via randomization. b) Pagel's (1999) λ statistic for phylogenetic signal, also showing the likelihood surface. Data consist of maximum body length (on a log scale) from 61 species of elopomorph eels. See main text for additional details.

```
##                    pPC1     pPC2     pPC3
## C._aridus       0.59441 -0.40209  0.57109
## C._minor        0.65171 -0.32732  0.55692
## C._imkeae       0.19958 -0.08978  0.56671
## C._mclachlani   0.62065  0.03746  0.86721
## C._macropholis  0.44875 -0.75942  0.09737
## C._cordylus    -0.07267  0.48294 -0.54394
```

Our trait data in this case are species scores for three different principal component (PC) axes from a phylogenetic principal components analysis undertaken using the *phytools* `phyl.pca` function (Revell 2009). Cordylid lizards are known for their body and tail armor, consisting of large, rectangular scales called osteoderms. Principal component 1 in Broeckhoven et al. (2016) separated the most lightly armored cordylids (large negative values), from those cordylids with the heaviest body armor (large positive values of PC 1). Why don't we extract this principal component from our data frame and rename it, as follows?

```
cordylid.armor_score<-setNames(
  cordylid.data$pPC1,rownames(cordylid.data))
```

With this named trait vector at the ready, we're prepared to undertake our Bayesian MCMC. As noted above, we'll use the default conditions but update the number of generations that we want our MCMC to run to `ngen=500000`. Depending on the size of our phylogenetic tree, we may want to run more (or fewer) generations in a genuine empirical study.

```
cordylid.mcmc<-anc.Bayes(cordylid.tree,
  cordylid.armor_score,ngen=500000)
```

```
## List of 7
##  $ sig2   : num 0.713
##  $ a      : num [1, 1] 0.000422
##  $ y      : num [1:26] 0.000422 0.000422 0.000422 0.000422 ...
##  $ pr.mean: num [1:28] 1000 0 0 0 0 0 0 0 0 0 ...
##  $ pr.var : num [1:28] 1e+06 1e+03 1e+03 1e+03 1e+03 ...
```

```
##  $ prop   : num [1:28] 0.00713 0.00713 0.00713 0.00713 ...
##  $ sample : num 100

## Starting MCMC...

## Done MCMC.
```

We can see that the method starts by printing out a summary of the "control parameters" of the MCMC. These include information about our prior probability distributions, the variances of the proposal distributions, and, finally, the interval that we'll use to sample from our posterior. All of these parameters can be adjusted by the *phytools* user.

The object class that results from this function call (`"anc.Bayes"`) has a `summary` method in *phytools* that prints the mean from the posterior distribution, automatically excluding the first 20% of our samples as burn-in (though we can adjust this percentage if we'd like). It also passes the estimates (normally invisibly, but we can save them to a new variable in our workspace as we've done here) back to the user.

```
cordylid.ace<-summary(cordylid.mcmc)
```

```
##
## Object of class "anc.Bayes" consisting of a posterior
##    sample from a Bayesian ancestral state analysis:
##
## Mean ancestral states from posterior distribution:
##         29         30         31         32         33         34
##  0.059277 -0.099027 -0.106452  0.057220  0.153671  0.201722
##         35         36         37         38         39         40
##  0.225081  0.297659  0.392992  0.493316  0.015503 -0.006053
##         41         42         43         44         45         46
##  0.435309  0.392526  0.300532  0.210391 -1.505181 -1.857682
##         47         48         49         50         51         52
## -0.136014 -0.520322 -0.829181 -0.985510 -1.040208  0.385293
##         53         54         55
##  0.511646  0.159943  0.028358
##
## Based on a burn-in of 1e+05 generations.
```

Now that we've obtained our estimated Bayesian ancestral states for internal nodes, it's a straightforward task to visualize them on the branches and nodes of the tree. For this undertaking we'll use the popular *phytools* plotting function `contMap` (Revell 2013). By default, `contMap` uses Maximum Likelihood to compute ancestral states at all of the internal nodes of the tree – but it can also be supplied with user-specified values. Since we want to use our Bayesian estimates from `anc.Bayes`, that's what we'll do here.

```
cordylid.contMap<-contMap(cordylid.tree,
  cordylid.armor_score,anc.states=cordylid.ace,
  plot=FALSE)
cordylid.contMap<-setMap(cordylid.contMap,
  viridisLite::viridis(n=10,direction=-1))
plot(cordylid.contMap,ftype="i",fsize=c(0.6,0.8),
  leg.txt="PC 1 (increasing armor)",lwd=3)
nodelabels(frame="circle",bg="white",cex=0.6)
```

(For fun, compare Figure 10 to Figure 2 of Broeckhoven et al. in which estimated ancestral state values were assigned to each branch using a similar color gradient!)

In addition to this simple analysis, we can (naturally) extract and plot posterior probability densities from any of our internal nodes of the tree. To see this, let's focus on the node labeled "49" in Figure 10

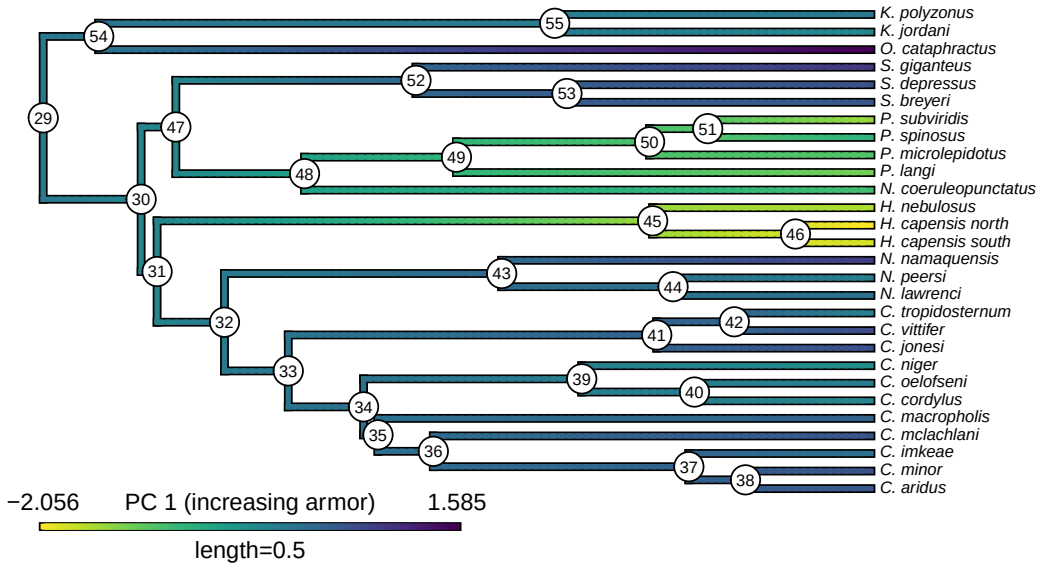

Figure 10. Reconstructed ancestral values from Bayesian MCMC projected onto the nodes and edges of the tree. Numerical values at internal nodes are node indices from our input phylogeny. Data consist of PC 1 from a phylogenetic principal components analysis of cordylid morphological trait, and separate highly armored (high values) from lightly armored (low values) lizards. See main text for more details.

and do exactly that. Node 49 corresponds to the common ancestor of the *Pseudocordylus* clade, which is among the least heavily armored of all cordylids in our tree, so we'd expect our posterior distribution for this node to be centered on a relatively low value for our armor score.

```
cordylid.node49<-density(cordylid.mcmc,
  what=49)
cordylid.node49
```

```
##
## Call:
##  density.anc.Bayes(x = cordylid.mcmc, what = 49)
##
## Data: node 49 (4001 obs.);   Bandwidth 'bw' = 0.05679
##
##        x                  y
##  Min.   :-2.1658   Min.   :0.000023
##  1st Qu.:-1.5270   1st Qu.:0.022415
##  Median :-0.8883   Median :0.187932
##  Mean   :-0.8883   Mean   :0.391002
##  3rd Qu.:-0.2495   3rd Qu.:0.784591
##  Max.   : 0.3893   Max.   :1.187968
```

```
par(mar=c(5.1,4.1,1.1,2.1))
plot(cordylid.node49,las=1,bty="n",main="",
  cex.lab=0.9,cex.axis=0.8,
  xlab="PC 1 (increasing armor)",
  ylab="Posterior density",
  xlim=range(cordylid.armor_score))
```

Figure 11 shows our estimate of the posterior probability distribution of the ancestral node 49 state, and should be centered precisely on the value we projected onto the tree of Figure 10!

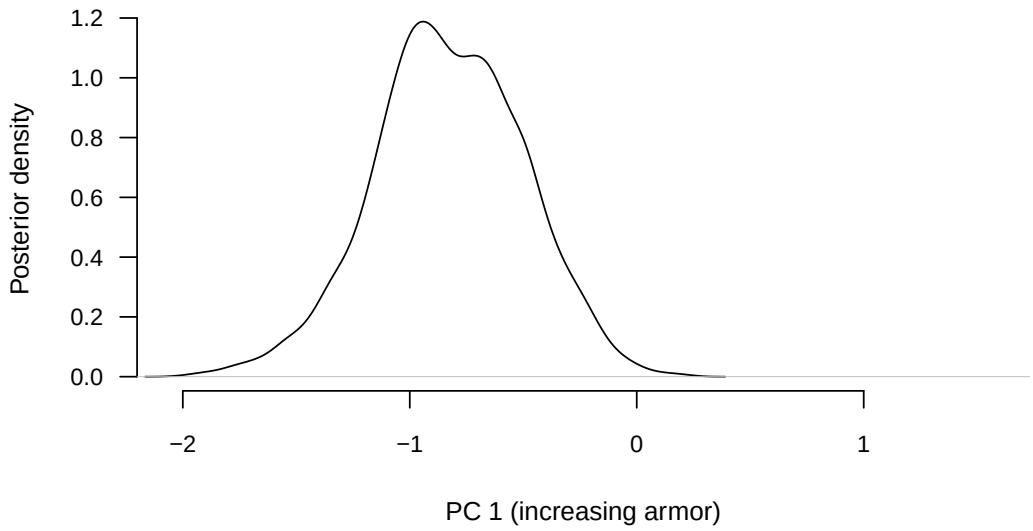

Figure 11. Posterior probability density at node 49 of Figure 10 from Bayesian MCMC ancestral state reconstruction of PC 1 from a morphological analysis on a phylogenetic tree of cordylid lizards. Node 49 corresponds to the common ancestor of the *Pseudocordylus*: a relatively lightly armored cordylid clade. See main text for more details.

5.3 Multivariate trait evolution

Along with the various univariate methods we've seen so far, *phytools* also contains a handful of different multivariate trait evolution models, designed for both continuous and discrete characters.

One of these is an interesting model (described in Revell and Collar 2009; Revell et al. 2022) in which the rates and evolutionary correlations between traits are allowed to vary as a function of a set of mapped regimes on the tree. (Similar to O'Meara et al. 2006, but for more than one trait at a time.) The underlying motivation of this method is to test hypotheses about phylogenetic heterogeneity in the evolutionary relationship (i.e., correlation) between different traits on our phylogeny.

To illustrate the method, I'll use a phylogenetic tree and dataset of tropidurid lizard species from Revell et al. (2022).

```
data(tropidurid.tree)
data(tropidurid.data)
```

In this case, our phylogeny is already a tree with mapped regimes. We can see this by merely printing the model object that we loaded. (In an empirical study we might imagine using a set of such trees sampled in proportion to their probabilities using stochastic mapping – and then averaging the result. E.g., see Revell et al. 2022.)

```
print(tropidurid.tree,printlen=2)
```

```
##
## Phylogenetic tree with 76 tips and 75 internal nodes.
##
## Tip labels:
##  Leiocephalus_raviceps, Leiocephalus_carinatus, ...
##
## The tree includes a mapped, 2-state discrete character
## with states:
##  n_rock, rock
```

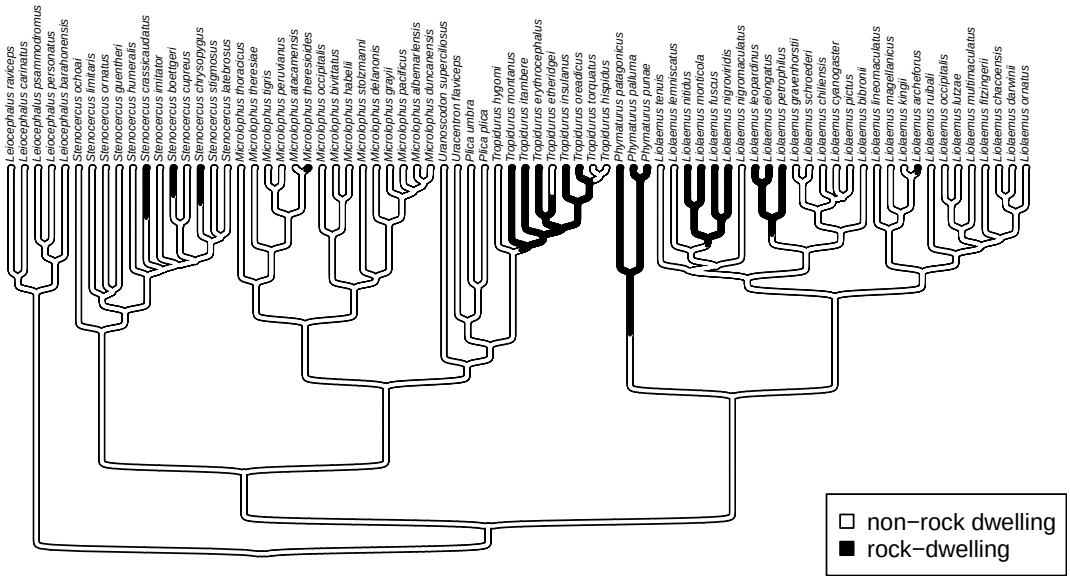

Figure 12. Phylogenetic tree of rock- and non-rock dwelling tropidurid lizard species from Revell et al. (2022). See main text for more details.

```
##
## Rooted; includes branch lengths.
```

This tells us that our phylogenetic tree contains 76 taxa and a mapped regime with two states: `"n_rock"` (non-rock dwelling) and `"rock"` (rock-dwelling). Since *phytools* permits mapped regimes to have arbitrary names, let's rename these levels in a more informative way. To do so, I'll use the *phytools* function `mergeMappedTraits`. `mergeMappedTraits`, as readers can probably guess, is designed to merge the mappings of two or more traits into one – but can also be employed to simply substitute one mapping name for another.

```
tropidurid.tree<-mergeMappedStates(
  tropidurid.tree,"n_rock","non-rock dwelling")
tropidurid.tree<-mergeMappedStates(
  tropidurid.tree,"rock","rock-dwelling")
```

Let's plot this updated tree. To do so, I'm going to use the recent *phytools* function `sigmoidPhylogram` that will plot our tree using curved ("sigmoidal") linking lines. (*phytools* contains lots of cool functions like this one!)

```
cols<-setNames(c("white","black"),c("non-rock dwelling",
  "rock-dwelling"))
sigmoidPhylogram(tropidurid.tree,
  direction="upwards",outline=TRUE,
  colors=cols,direction="upwards",
  outline=TRUE,lwd=2,fsize=0.4,
  ftype="i",offset=1)
legend("bottomright",c("non-rock dwelling",
  "rock-dwelling"),pch=22,pt.bg=cols,
  cex=0.8,pt.cex=1.2)
```

The phenotypic trait data in `tropidurid.data` consist of a single measure of overall body size (as trait 1, `"new_size"`), and a second metric trait measuring dorsoventral depth (vs. flattening, `"body_height"`).

```
head(tropidurid.data)
```

```
##                             newsize body_height
## Leiocephalus_raviceps      2.358317  0.05768818
## Leiocephalus_carinatus     2.931721  0.30216220
## Leiocephalus_psammodromus 2.700397  0.19667083
## Leiocephalus_personatus    2.535315  0.32216983
## Leiocephalus_barahonensis 2.473666  0.30266917
## Stenocercus_ochoai         2.549010  0.29067644
```

Our hypothesis of multivariate trait evolution in this clade is that these two traits (size and body depth) should generally scale together in non-rock dwelling lizard species: bigger lizards also tend to have larger body depths. We hypothesize, however, that this general relationship may become decoupled in rock-dwelling lineages where the force of selection is predicted to favor increased flattening, relative to their non-rock dwelling kin. (There are biomechanical and behavioral reasons to suspect this could be so. For more information, see Revell et al. 2007; Revell et al. 2022.)

To test this hypothesis, we'll use the *phytools* function `evolvcv.lite` which fits a heirarchical set of models for the evolutionary rates (of each character) and evolutionary correlations (between them, Revell et al. 2022).

```
tropidurid.fits<-evolvcv.lite(tropidurid.tree,
  tropidurid.data)
```

```
## Fitting model 1: common rates, common correlation...
## Best log(L) from model 1: 52.3056.
## Fitting model 2: different rates, common correlation...
## Best log(L) from model 2: 54.3968.
## Fitting model 3: common rates, different correlation...
## Best log(L) from model 3: 55.1105.
## Fitting model 4: no common structure...
## Best log(L) from model 4: 56.2877.
```

Having fit each of four models (in this case: `evolvcv.lite` actually includes several additional models that we won't review here, see Revell et al. 2022 for more details), we can most easily compare all of the models in our set using a generic `anova` function call as follows.

```
anova(tropidurid.fits)
```

```
##           log(L) d.f.       AIC     weight
## model 1 52.30560    5 -94.61119 0.09221085
## model 2 54.39681    7 -94.79362 0.10101737
## model 3 55.11048    6 -98.22096 0.56057433
## model 4 56.28765    8 -96.57530 0.24619744
```

This comparison shows us that "`model 3`" is the best-supported explanation of our data in this set. Indeed, this model is one in which the evolutionary covariance between overall body size and dorsoventral flattening is negative among rock-dwelling lineages – compared to the positive evolutionary covariance in non-rock species and across all other models, just as we'd predicted. Size and body depth are indeed evolutionarily decoupled in rock-dwelling forms!

```
tropidurid.fits
```

```
## Model 1: common rates, common correlation
##  R[1,1]  R[1,2]  R[2,2]  k   log(L)  AIC
## fitted   0.2224  0.0154  0.0589  5   52.3056 -94.6112
```

```
##
## (R thinks it has found the ML solution for model 1.)
##
## Model 2: different rates, common correlation
##  R[1,1]  R[1,2]  R[2,2]  k   log(L)  AIC
## non-rock dwelling    0.2025  0.0187  0.0456  7   54.3968 -94.7936
## rock-dwelling    0.3043  0.0382  0.1263
##
## (R thinks it has found the ML solution for model 2.)
##
## Model 3: common rates, different correlation
##  R[1,1]  R[1,2]  R[2,2]  k   log(L)  AIC
## non-rock dwelling    0.2256  0.0394  0.0588  6   55.1105 -98.221
## rock-dwelling    0.2256  -0.0354 0.0588
##
## (R thinks it has found the ML solution for model 3.)
##
## Model 4: no common structure
##  R[1,1]  R[1,2]  R[2,2]  k   log(L)  AIC
## non-rock dwelling    0.2108  0.0325  0.0485  8   56.2877 -96.5753
## rock-dwelling    0.2794  -0.0564 0.101
##
## (R thinks it has found the ML solution for model 4.)
```

5.4 Variable rate Brownian motion

Lastly, I recently added a function to *phytools* that permits us to fit a variable-rate Brownian evolution model using penalized likelihood (Revell 2021). Under this model we assume that our trait evolves via a standard Brownian motion process – but that the rate of evolution (σ^2) itself also changes through time and among the clades of our tree… via a process of geometric Brownian motion. As one might expect for a penalized likelihood method, when we go ahead and fit this model to data, the degree to which the evolutionary rate is permitted to vary from edge to edge in the tree is controlled by our λ penalty or "smoothing" coefficient (Revell 2021).

This method has already been used to, for example, investigate rate heterogeneity differences in body size evolution between cetaceans and plesiosaurs (Sander et al. 2021). Here, I'll apply it to the analysis of skull size evolution in a phylogenetic tree of primates. My data for this example (now packaged with *phytools*) come from a book chapter by Kirk and Kay (2004).

```
data(primate.tree)
data(primate.data)
```

Our data frame, `primate.data`, contains a number of different variables. Let's pull out just one of these, `Skull_length`, and (as we do) convert it to a logarithmic scale.

```
primate.lnSkull<-setNames(
  log(primate.data$Skull_length),
  rownames(primate.data))
head(primate.lnSkull)
```

```
## Allenopithecus_nigroviridis           Alouatta_palliata
##                    4.590057                    4.698661
##          Alouatta_seniculus           Aotus_trivirgatus
##                    4.682131                    4.102643
##           Arctocebus_aureus     Arctocebus_calabarensis
##                    3.901973                    3.985273
```

With just this input data vector and our tree, we're already ready to run our penalized likelihood analysis. Invariably, however, penalized likelihood requires the user to specify a smoothing parameter –

normally denominated λ. λ determines the weight that's assigned to the penalty term of the fitted model, in our case a measure of how much (or how little) the evolutionary rate evolves from edge to edge in the tree (Revell 2021). A large value of λ will penalize high rate variation between edges and thus cause us to fit a model with relatively low rate heterogeneity across the tree. Smaller values of λ, on the other hand, should allow the rate of evolution to vary with relatively little smoothing cost.

A number of approaches, such as cross-validation (e.g., Efron and Gong 1983), have been recommended to help us identify suitable values of λ for our data and question – however, I would minimally suggest testing multiple values of λ and comparing the results! Let's do exactly that for our analysis of primate skull length: first using $\lambda = 1.0$, and then swapping it for a much smaller $\lambda = 0.1$ and much larger $\lambda = 10$. This will allow us to pretty quickly see how these different values of our smoothing parameter affect our findings, and thus how sensitive any inference we draw might be to the specific value of λ we assigned!

Before continuing, however, we might get a better sense of our data by creating a simple projection of our phenotypic trait (log skull length) onto the tree. In this case, I'll use two different plotting methods.

First, I'll use the *phytools* function `edge.widthMap` which sizes the thickness of our plotted branches in proportion to the observed or reconstructed trait values. (This is one of my favorite *phytools* functions, but, compared to `contMap`, it's been used very little in published literature!) We can see the result in Figure 13a.

In addition to this visualization, I'll also undertake a simple project our phylogeny into the trait space. This is done using a popular *phytools* plotting method called `phenogram` (Evans et al. 2009; Revell 2013). In the typical style of this kind of plot, our phylogeny is graphed in a space defined by time since the root of the tree (on our horizontal axis), and the observed or reconstructed values of our phenotypic trait (on the vertical, Revell 2013). The result of this projection is shown in Figure 13b.

```
par(mfrow=c(1,2))
primate.widthMap<-edge.widthMap(primate.tree,
  primate.lnSkull)
plot(primate.widthMap,color=palette()[2],
  legend="log(skull length)",border=TRUE,
  fsize=0.4,mar=c(4.1,4.1,2.1,0.1))
mtext("a)",adj=0,line=0,cex=1.4)
phenogram(primate.tree,primate.lnSkull,fsize=0.4,ftype="i",
  spread.cost=c(1,0),mar=c(4.1,4.1,2.1,0.1),quiet=TRUE,
  las=1,cex.axis=0.8,ylab="log(skull length)")
mtext("b)",adj=0,line=0,cex=1.4)
```

Visual inspection of Figure 13 may give us a preliminary sense of where in our tree our penalized likelihood method could end up showing the rate of primate skull length evolution to vary the most – and the least.

The function we'll use to fit our rate-variable model, `multirateBM`, performs a computationally intensive optimization. Setting the optional argument `parallel=TRUE` will help distribute this burden across multiple processors of our computer, if possible.

Let's start with $\lambda = 1.0$.

```
primate.mBM_1<-multirateBM(primate.tree,
  primate.lnSkull,lamba=1,parallel=TRUE)
```

```
## Beginning optimization....
## Using socket cluster with 16 nodes on host 'localhost'.
## Optimization iteration 1. Using "L-BFGS-B" (parallel)
## optimization method.
## Best (penalized) log-likelihood so far: -267.108
## Done optimization.
```

Now we can do the same with $\lambda = 0.1$ and 10. This time I'll turn off printing by updating the optional argument to `quiet=TRUE`.

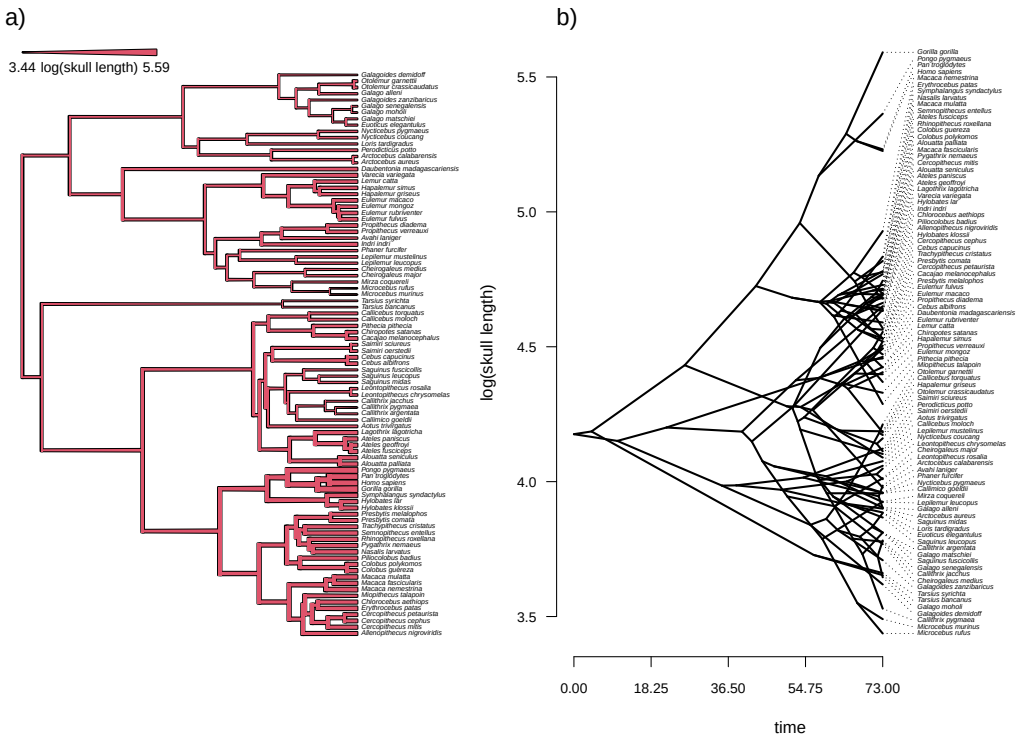

Figure 13. a) Primate skull lengths (on a log scale) projected onto the edges and nodes of the phylogeny. The width of each edge of the tree is proportional to the observed or reconstructed value of the trait. See main text for more details. b) A projection of the phylogeny of a) into a space defined by time since the root (on the horizontal axis) and log skull length.

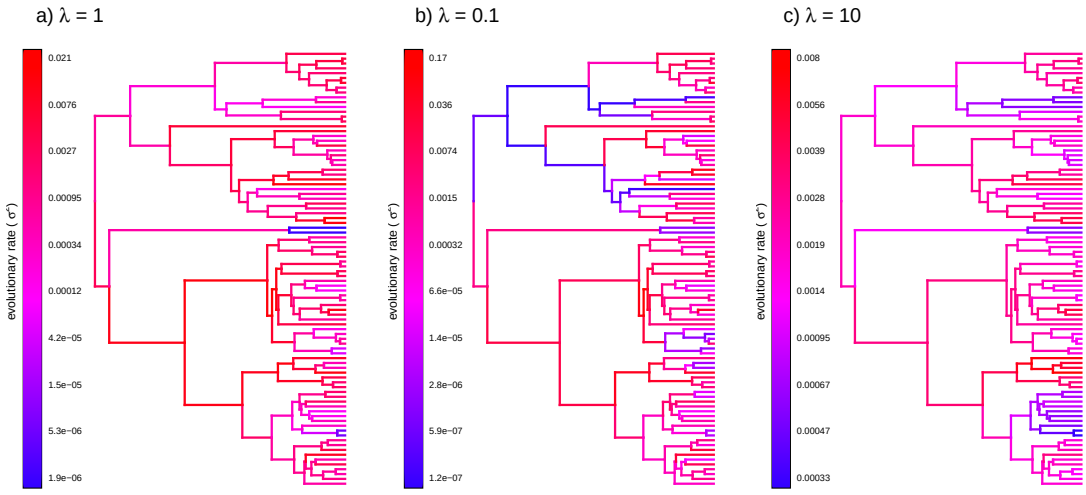

Figure 14. Estimated rates of log(skull length) evolution in primates under a variable-rate Brownian evolution model for different values of the smoothing parameter, λ. Increasing values of λ should correspond to less variation in the rate of evolution across the tree. See main text for additional details.

```
primate.mBM_0.1<-multirateBM(primate.tree,
  primate.lnSkull,lambda=0.1,parallel=TRUE,
  quiet=TRUE)
primate.mBM_10<-multirateBM(primate.tree,
  primate.lnSkull,lambda=10,parallel=TRUE,
  quiet=TRUE)
```

(Readers should take special care to note that the specific values of the penalized log likelihoods are not comparable between analyses with different values of the penalty coefficient, λ!)

Finally, let's visualize the differences and similarities between each of our three fitted models.

```
par(mfrow=c(1,3))
plot(primate.mBM_1,ftype="off",lwd=2,
  mar=c(0.1,0.1,2.1,0.1))
mtext(expression(paste("a) ",lambda," = 1")),
  adj=0.1,line=0.5,cex=1.1)
plot(primate.mBM_0.1,ftype="off",lwd=2,
  mar=c(0.1,1.1,2.1,0.1))
mtext(expression(paste("b) ",lambda," = 0.1")),
  adj=0.1,line=0.5,cex=1.1)
plot(primate.mBM_10,ftype="off",lwd=2,
  mar=c(0.1,1.1,2.1,0.1))
mtext(expression(paste("c) ",lambda," = 10")),
  adj=0.1,line=0.5,cex=1.1)
```

We can see from the plot of Figure 14 that even though the specific range of rate variation depends strongly on our specified values of λ, the pattern from clade to clade on the tree is relatively robust. This should give us some measure of confidence that the our inferred rate heterogeneity may be a product of real variability in the evolutionary rate for our character on the phylogeny!

6 DIVERSIFICATION

In addition to the methods that we've seen so far, *phytools* also contains a handful of different techniques for investigating diversification on reconstructed phylogenies. Diversification has never been the primary

focus of the *phytools* R package (to that end, I'd recommend the powerful *diversitree* package, FitzJohn 2012), but these methods are popular, and the *phytools* implementations can be relatively easy to use.

phytools contains methods to compute and visualize the accumulation of lineages through time, including with extinction (`ltt`), to calculate and test the γ statistic (`gammatest`, `mccr`, Pybus and Harvey 2000), to fit pure-birth and birth-death models, including with random missing taxa (`fit.yule` and `fit.bd`, Nee et al. 1994; Stadler 2013), to compare diversification rates between trees (`ratebytree`, Revell 2018), and to simulate stochastic trees under various conditions (`pbtree`).

6.1 Lineage through time plots

One of the most rudimentary phylogenetic methods for studying diversification is to simply graph the accumulation of new lineages in our reconstructed phylogeny over time since the global root of the tree. This visualization method is called a lineage-through-time plot.

One of the great appeals of this visualization is that if we graph the number of lineages through time in a fully-sampled pure-birth (that is, constant-rate speciation, but no extinction) phylogenetic tree, the accumulation curve should be exponential – or exactly linear on a semi-logarithmic scale. This means that the lineage-through-time plot gives us a handy tool that we can use to compare the real lineage accumulation in our reconstructed tree to this simple, neutral expectation (Pybus and Harvey 2000; Revell and Harmon 2022).

To see how the number of lineages through time are calculated and graphed using *phytools*, let's load a phylogenetic tree of lizards from the diverse South American family Liolaemidae. This phylogeny is packaged with *phytools* but was published by Esquerré et al. (2019).

```
data(liolaemid.tree)
print(liolaemid.tree,printlen=2)
```

```
##
## Phylogenetic tree with 257 tips and 256 internal nodes.
##
## Tip labels:
##   Liolaemus_abaucan, Liolaemus_koslowskyi, ...
##
## Rooted; includes branch lengths.
```

We're going to create our lineage-through-time graph with *phytools* over two steps.

First, we'll use the *phytools* function `ltt` to compute an object of class `"ltt"` containing our tree and a count of the number of lineages through time from the root of the tree to the tips.

```
liolaemid.ltt<-ltt(liolaemid.tree,plot=FALSE)
liolaemid.ltt
```

```
## Object of class "ltt" containing:
##
## (1) A phylogenetic tree with 257 tips and 256 internal
##     nodes.
##
## (2) Vectors containing the number of lineages (ltt) and
##     branching times (times) on the tree.
##
## (3) A value for Pybus & Harvey's "gamma" statistic of
##     gamma = 1.8129, p-value = 0.0698.
```

From the print-out we see that in addition to the tree and the lineages through time, our object also contains a value of (and a P-value for) Pybus and Harvey's (2000) γ statistic.

γ is a numerical value used to describe the general shape of the lineage through time curve. If the curve is straight (on a semi-log scale), then γ should have a value close to zero. This is what we expect under a pure-birth (speciation only) diversification process. On the other hand, significantly positive or

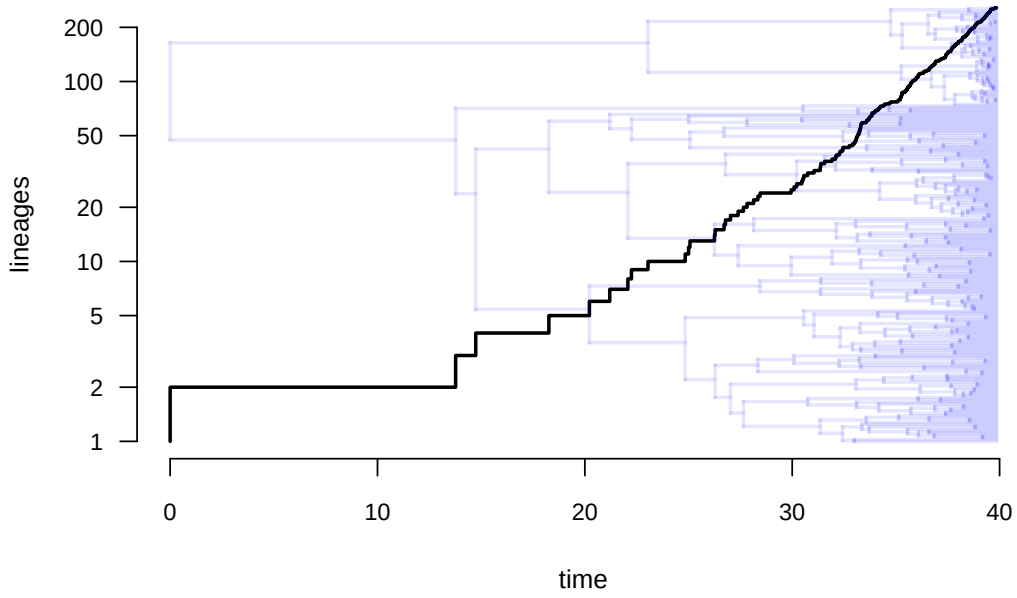

Figure 15. Lineage through time plot for phylogeny of lizards from the South American family Liolaemidae. See main text for more details.

significantly negative γ mean that the lineage through time graph curves upward or downward towards the present day (Pybus and Harvey 2000). This could mean that the rate of diversification has changed over time, but it could also be due to past extinction or incomplete taxon sampling (Revell and Harmon 2022).

We can see that for our liolaemid lizards, γ is slightly *positive*, though not significantly so.

```
par(mar=c(5.1,4.1,1.1,2.1))
plot(liolaemid.ltt,show.tree=TRUE,lwd=2,
  log.lineages=FALSE,log="y",bty="n",las=1,
  cex.axis=0.8,cex.lab=0.9,transparency=0.1)
```

In general, accounting for incomplete taxon sampling in the measurement of the γ statistic is important because missing taxa will tend to pull our lineage-through-time curve downwards as we approach the tips of the tree – in other words, towards more negative values of γ.

Fortunately, there's a simple way to address this bias. If we know the true species richness of our clade of interest, we can simply simulate trees that match this richness under pure-birth, randomly prune taxa to the level of "missingness" in our reconstructed tree, and then use the distribution of γ values across this set of simulated (and then randomly pruned) trees as our null distribution for hypothesis testing! This exact procedure is called the "Monte Carlo constant rates" (or MCCR, Pybus and Harvey 2000) test and is implemented in the *phytools* function `mccr`.

Of course, since the MCCR test accounts for randomly missing taxa from our tree, we must know or hypothesize a true species richness of our clade. In this instance, we're not too preoccupied about the precise value for Liolaemidae, but the *Reptile Database* puts the species richness of this diverse South American family at 340. For illustrative purposes only, let's just go with this number!

```
liolaemid.mccr<-mccr(liolaemid.ltt,
  rho=Ntip(liolaemid.tree)/340,nsim=1000)
liolaemid.mccr
```

```
## Object of class "mccr" consisting of:
##
```

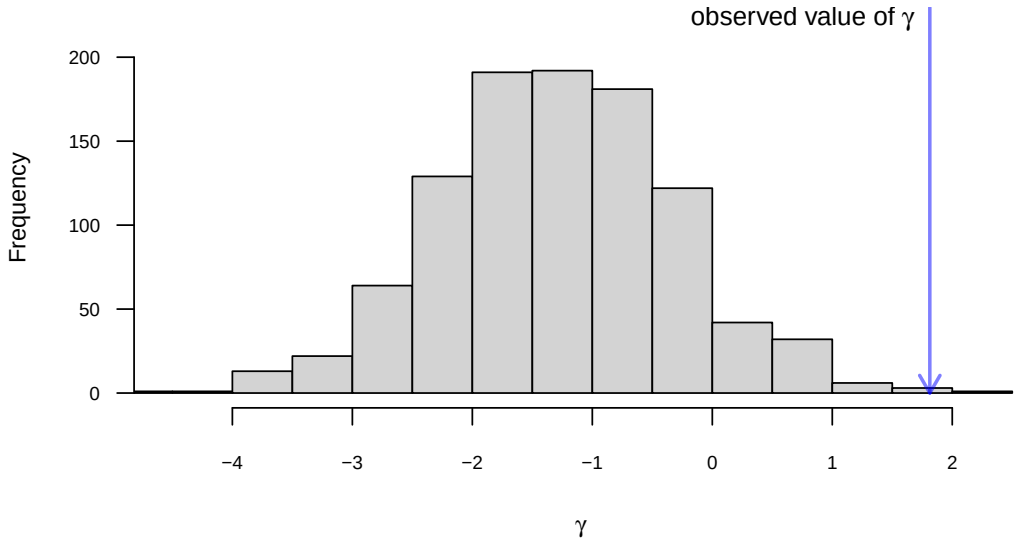

Figure 16. Lineage through time plot for phylogeny of lizards from the South American family Liolaemidae.

```
## (1) A value for Pybus & Harvey's "gamma" statistic of
##     gamma = 1.8129.
##
## (2) A two-tailed p-value from the MCCR test of 0.002.
##
## (3) A simulated null-distribution of gamma from 1000
##     simulations.
```

This tells us that, having accounted for missing taxa, our observed value of γ becomes highly significantly different from that expected under pure-birth. We can plot our results to see what I mean.

```
par(mar=c(5.1,4.1,0.6,2.1))
plot(liolaemid.mccr,las=1,cex.lab=0.8,
  cex.axis=0.7,main="")
```

Figure 16 shows that the measured value of γ by this test is strongly significantly positive – in other words (accounting for missing taxa), much larger than expected by random chance under a constant-rate pure birth speciation process!

6.2 Modeling speciation and extinction

In addition to these analysis, *phytools* can also fit simple speciation and extinction models following Nee et al. (1994; Stadler 2013; Harmon 2019). This is done primarily using the function `fit.bd`, which also allows us to take into account an incomplete taxonomic sampling fraction (Stadler 2013).

Just as with γ, incomplete sampling has the potentially to substantially distort our estimated rates of speciation (normally given as λ – a different λ from before!) and extinction (μ). In this case, ignoring (or underestimating) the missing lineages in our tree will tend to cause us to underestimate the rate of extinction, as nearly all of the information we have about extinction comes from the most recent parts of our phylogeny! (See Harmon 2019; Revell and Harmon 2022 for more details.)

Fitting a birth-death model using *phytools* is very easy. We'll pass our liolaemid tree to the `fit.bd` function, and the only additional argument to be assigned is `rho` (for ρ), the sampling fraction, which we'll set to have a value equal to the number of tips in our tree divided by the quantity (340, see above) that we hypothesize represents the true species richness of Liolaemidae.

```
liolaemid.bd<-fit.bd(liolaemid.tree,
  rho=Ntip(liolaemid.tree)/340)
liolaemid.bd
```

```
##
## Fitted birth-death model:
##
## ML(b/lambda) = 0.351
## ML(d/mu) = 0.1771
## log(L) = 526.451
##
## Assumed sampling fraction (rho) = 0.7559
##
## R thinks it has converged.
```

Other R packages (such as the aforementioned *diversitree*) might allow us to compare our fitted birth-death model to a range of other hypotheses about diversification, such as that the speciation and extinction rates change through time or as a function of our phenotypic traits (e.g., Maddison et al. 2007; FitzJohn 2010; Morlon et al. 2010; Revell and Harmon 2022). In *phytools* we can compare our fitted birth-death model to only one alternative model: the simpler, pure-birth model – also called a 'Yule' model.

```
liolaemid.yule<-fit.yule(liolaemid.tree,
  rho=Ntip(liolaemid.tree)/340)
liolaemid.yule
```

```
##
## Fitted Yule model:
##
## ML(b/lambda) = 0.2499
## log(L) = 521.2295
##
## Assumed sampling fraction (rho) = 0.7559
##
## R thinks it has converged.
```

```
anova(liolaemid.yule,liolaemid.bd)
```

```
##                  log(L) d.f.       AIC    weight
## liolaemid.yule 521.2295    1 -1040.459 0.0144644
## liolaemid.bd   526.4510    2 -1048.902 0.9855356
```

This result tells us that, in the context of the two very simple models that we've fit to our reconstructed tree, a two-parameter birth-death (speciation and extinction) model is much better supported than our simpler Yule model!

Lastly, the *phytools* function `fit.bd` exports a likelihood function as part of the fitted model object. This, in turn, makes it very straightforward for *phytools* users to (for example) compute and graph the likelihood surface. Here, I'll illustrate this using the base R graphics function `persp`. (But R and contributed R packages contain lots of even fancier 3D plotting methods that readers might be more interested in trying!)

```
ngrid<-40
b<-seq(0.25,0.45,length.out=ngrid)
d<-seq(0.10,0.25,length.out=ngrid)
logL<-matrix(NA,ngrid,ngrid)
```

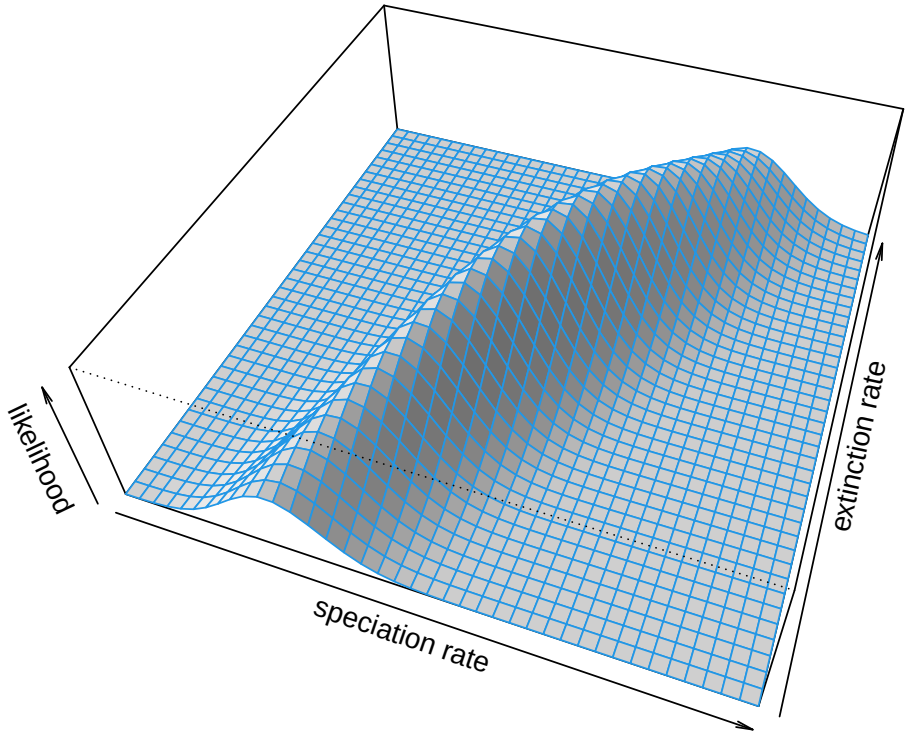

Figure 17. Visualization of the likelihood surface for speciation and extinction rates estimated for a phylogenetic tree of Liolaemidae. The ridge of values with similar likelihoods is typical of this class of model. See main text for more details.

```
for(i in 1:ngrid) for(j in 1:ngrid)
  logL[i,j]<-liolaemid.bd$lik(c(b[i],d[j]))
logL[is.nan(logL)]<-min(logL[!is.nan(logL)])
par(mar=rep(0.1,4))
persp(b,d,exp(logL),shade=0.3,
  phi=45,theta=20,
  xlab="speciation rate",
  ylab="extinction rate",
  zlab="likelihood",
  border=palette()[4],expand=0.3)
```

(Some astute readers will notice the line `logL[is.nan(logL)] <- min(...)` (etc.) in my script of above. This is because during our grid evaluation of the likelihood function, sometimes the function was being evaluated in parameter space where the likelihood is not defined. To account for this I set all parts of the likelihood surface that could not be computed to the numerical minimum of the graph!)

Figure 17 shows the very strong *ridge* in the likelihood surface (from low λ and low μ, to high λ and high μ) that almost invariably tends to characterize the likelihood surfaces of birth-death models.

7 VISUALIZATION

After phylogenetic comparative analysis, *phytools* is perhaps best known for its phylogeny visualization methods.

We've seen a number of these approaches already deployed throughout this article. For example, in Figures 1, 2, 3, 4, 7, and 12 I illustrated custom *phytools* plotting methods for stochastic character mapping and the analysis of stochastically mapped trees. Likewise, in Figures 5 and 6 I demonstrated *phytools* plotting methods for fitted discrete character evolution models. In Figures 9, 10, 13, and 14

I showed a variety of custom methods for visualizing continuous trait evolution. Finally, in Figures 8, 15, and 16 I illustrated several different approaches for graphing diversification or the results from an analysis of diversification on the tree. This is a sparse sample of the variety of plotting methods for phylogenies, phylogenetic comparative data, and the results of phylogenetic analysis that are implemented in the *phytools* package.

In this final section, I'll illustrate just a few more popular plotting methods of the package that we haven't already seen in prior bits of the present article.

7.1 Co-phylogenetic plotting

Among the most popular plotting method of the *phytools* package is the function `cophylo`, which creates co-phylogenetic plots (often referred to as "tanglegrams," Page 1993).

The purpose of tanglegrams varies widely from study to study. Classically, for instance, tanglegrams are used to visually illustrate the topological similarity between two groups that are hypothesized to co-speciate: for instance, an animal host and its parasites, or a plant and its pollinators (e.g., Page 1993; Medina and Langmore 2016; Endara et al. 2018; Caraballo 2022).

Equally often, however, tanglegrams are put to different purposes. For instance, tanglegrams are frequently employed to show the similarity or differences between alternative phylogenetic hypotheses (e.g., Amarasinghe et al. 2021), to identify incongruence among gene trees (e.g., Stull et al. 2020), and even to compare a phylogenetic history to a non-phylogenetic cluster dendogram based on phenotypic or ecological data (e.g., Atkinson et al. 2020; Huie et al. 2021).

To illustrate the *phytools* tanglegram method, I'll use a phylogenetic tree of bat species and another of their betacoronaviruses – both based on Caraballo (2022).

```
data(bat.tree)
data(betaCoV.tree)
```

Assuming that our tip labels differ between our different trees (and they do in this instance), we need more than just two phylogenies to create a tanglegram – we also need a table of associations linking the tip labels of one tree to those of the other! Again, based on Caraballo (2022), our association information for the two trees that we've loaded is contained in the *phytools* data object `bat_virus.data`. Let's load and review it.

```
data(bat_virus.data)
head(bat_virus.data)
```

```
##                     Bats betaCoVs
## 1    Artibeus lituratus KT717381
## 2 Artibeus planirostris MN872692
## 3 Artibeus planirostris MN872690
## 4 Artibeus planirostris MN872691
## 5 Artibeus planirostris MN872689
## 6 Artibeus planirostris MN872688
```

Inspecting just the first part of this object reveals its general structure. We can see that it consists of two columns: one for each of our two trees. The elements of the first column should match the labels of our first tree, and those of the second column the labels of our second tree. There's no problem at all if one or the other column has repeating names: a host can (of course) be associated with more than one parasite, and vice versa!

Now let's run our co-phylogenetic analysis. This will create, not a plot, but a `"cophylo"` object in which the node rotation has been optimized to maximize the tip alignment of the two trees.

```
bat.cophylo<-cophylo(bat.tree,betaCoV.tree,
  assoc=bat_virus.data)
```

```
## Rotating nodes to optimize matching...
## Done.
```

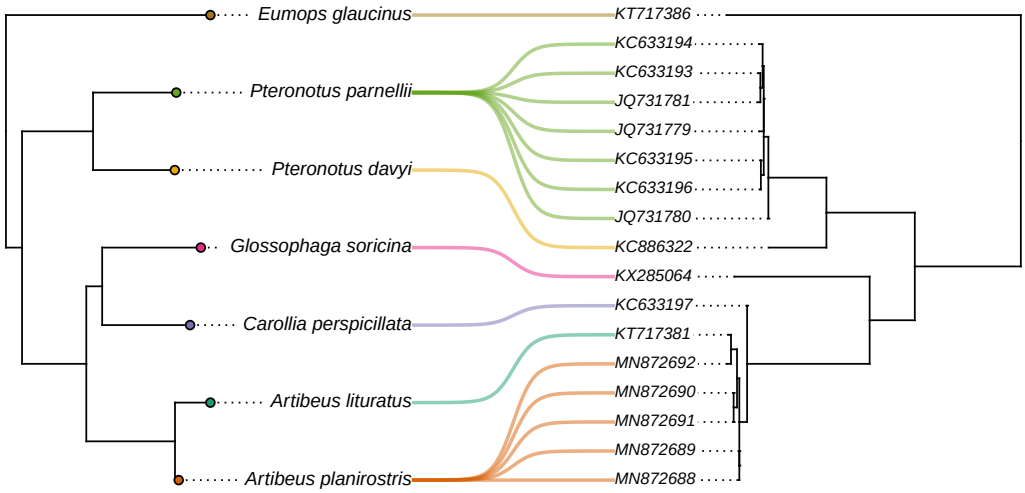

Figure 18. Co-phylogenetic plot of bat species and their associated betacoronaviruses. Associations and GenBank accession numbers from Caraballo (2022). See main text for more details.

We can print this object, as follows.

```
bat.cophylo
```

```
## Object of class "cophylo" containing:
##
## (1) 2 (possibly rotated) phylogenetic trees in an object of class
##     "multiPhylo".
##
## (2) A table of associations between the tips of both trees.
```

To plot it, we'll use the a generic *phytools* `plot` method for the object class. I'll go ahead and adjust a few settings of the method to make our graph look nice – and I'll use species-specific linking line colors so that we can more easily visualize all the different virus sequences that are associated with each bat host! (My color palette comes from the *RColorBrewer* function `brewer.pal`, Neuwirth 2022.)

```
cols<-setNames(RColorBrewer::brewer.pal(
  n=7,name="Dark2"),bat.tree$tip.label)
par(lend=3)
plot(bat.cophylo,link.type="curved",
  fsize=c(0.7,0.6),link.lwd=2,
  link.lty="solid",pts=FALSE,
  link.col=make.transparent(cols[
  bat_virus.data[,1]],0.5))
pies<-diag(1,Ntip(bat.tree))
colnames(pies)<-rownames(pies)<-names(cols)
tiplabels.cophylo(pie=pies,
  piecol=cols[
  bat.cophylo$trees[[1]]$tip.label],
  which="left",cex=0.2)
```

In general, our plot reveals a surprisingly strong association between the topology of the phylogenies of the bats and their viruses – a pattern that Caraballo (2022) also reported (and that happened to contrast with what Caraballo found for alphacoronaviruses, for what it's worth).

7.2 Projecting a tree onto a geographic map

phytools can also be used to project a phylogenetic tree onto a geographic map, a visualization technique that's been used in numerous published studies since it was added to the package (e.g., Csosz and Fisher 2016; Quach et al. 2019; Hermanson et al. 2020; Huang and Morgan 2021)

To see how this is done in R, we'll load two datasets that come with the *phytools* package. The first is a phylogenetic tree (`mammal.tree`) from Garland et al. (1992). The second is a corresponding geographic dataset that I obtained (i.e., built laboriously by hand) by sampling 1-3 records at random from the online citizen science resource *iNaturalist*.

```
data(mammal.tree)
data(mammal.geog)
```

Our data (`mammal.geog`) should come in the form of a matrix. This is important distinction to note because our plotting function allows more than one geographic record per species and requires that our taxon labels be supplied as row names. The most common ways to read data into R (for instance, using `read.table` or `read.csv`) create a data frames ,rather than a matrices – and R data frames don't permit repeating row names!

The easiest way to resolve this issue when working with a dataset that isn't pre-loaded, might be to read our data in with species as a separate data column rather than as row names, and then use our input data to create a matrix object within R.

Let's take a look at the data matrix `mammal.geog` to understand how it's structured

```
head(mammal.geog,10)
```

```
##                      lat       long
## A._alces       61.077601 -149.82472
## A._alces       45.655049  -69.65062
## A._americana   43.608278 -105.15943
## A._americana   40.739628 -104.51432
## A._buselaphus -19.960335   15.56863
## A._cervicapra  13.006147   80.24190
## A._cervicapra  14.656469   75.68250
## A._jubatus     -2.859852   34.90434
## A._melampus     1.307618   37.13768
## B._bison       44.974469 -110.70387
```

We should see that it consists of species names as row names, as promised, and geographic locality points in the form of decimal latitude and longitude coordinates.

I happen to know that there are some mismatches between our input tree and data. (This is because I didn't include geographic records for all species when I was creating `mammal.geog` – but it's a very common problem with comparative data!) To identify this discordance, I'm going to use the handy function `name.check` from the *geiger* comparative methods package (Pennell et al. 2014).

geiger is not a dependent package of *phytools*, but I suspect that most readers (particularly any that have followed along all the way to this point) will already have it installed. If not, it can be downloaded and installed from CRAN in the typical fashion.

```
library(geiger)
chk<-name.check(mammal.tree,mammal.geog)
summary(chk)
```

```
## 2 taxa are present in the tree but not the data:
##     G._granti,
##     V._fulva
##
## To see complete list of mis-matched taxa, print object.
```

This tells us that there are two taxa (*G. granti* and *V. fulva*) in our tree that are missing from our geographic data, so we can prune them out of our phylogeny using the *ape* function `drop.tip`. (There's no need to load *ape* – loading *phytools* took care of that.)

```
mammal.pruned<-drop.tip(mammal.tree,
  chk$tree_not_data)
name.check(mammal.pruned,mammal.geog)
```

```
## [1] "OK"
```

Our next step will be to build the map projection that we intend to plot. This is done using the *phytools* function `phylo.to.map`. In addition to combining our phylogenetic tree and map data, `phylo.to.map` also performs a series of node rotations designed to optimize the alignment of our phylogeny with the geographic coordinates of our tip data. As node rotation is arbitrary anyway, this can be helpful to facilitate a more convenient visualization.

```
mammal.map<-phylo.to.map(mammal.pruned,
  mammal.geog,plot=FALSE,quiet=TRUE)
mammal.map
```

```
## Object of class "phylo.to.map" containing:
##
## (1) A phylogenetic tree with 47 tips and 46 internal nodes.
##
## (2) A geographic map with range:
##      -85.19N, 83.6N
##      -180W, 180W.
##
## (3) A table containing 72 geographic coordinates (may include
##     more than one set per species).
##
## If optimized, tree nodes have been rotated to maximize alignment
## with the map when the tree is plotted in a downwards direction.
```

Our plotting function for this object class allows us to specify different colors for the different species being plotted. Let's do that here by choosing random colors from a divergent color palette using the CRAN package *randomcoloR* (Ammar 2019). (Readers lacking this package should install it from CRAN – or they can elect to use a different palette, or none at all.)

```
cols<-setNames(
  randomcoloR::distinctColorPalette(
  Ntip(mammal.pruned)),
  mammal.pruned$tip.label)
```

Finally, we're ready to plot our tree.

```
plot(mammal.map,fsize=0.5,ftype="i",
  colors=cols,asp=0.9,lwd=1,
  cex.points=c(0.4,0.8))
```

In this case, the lineages of our phylogeny in Figure 19 have a near-global distribution. For phylogenetic trees whose terminal taxa have a narrower geographic range, including phylogeographic studies, it's possible to restrict the plotted maps to countries or regions, or even to supply a custom map (e.g., Quach et al. 2019)!

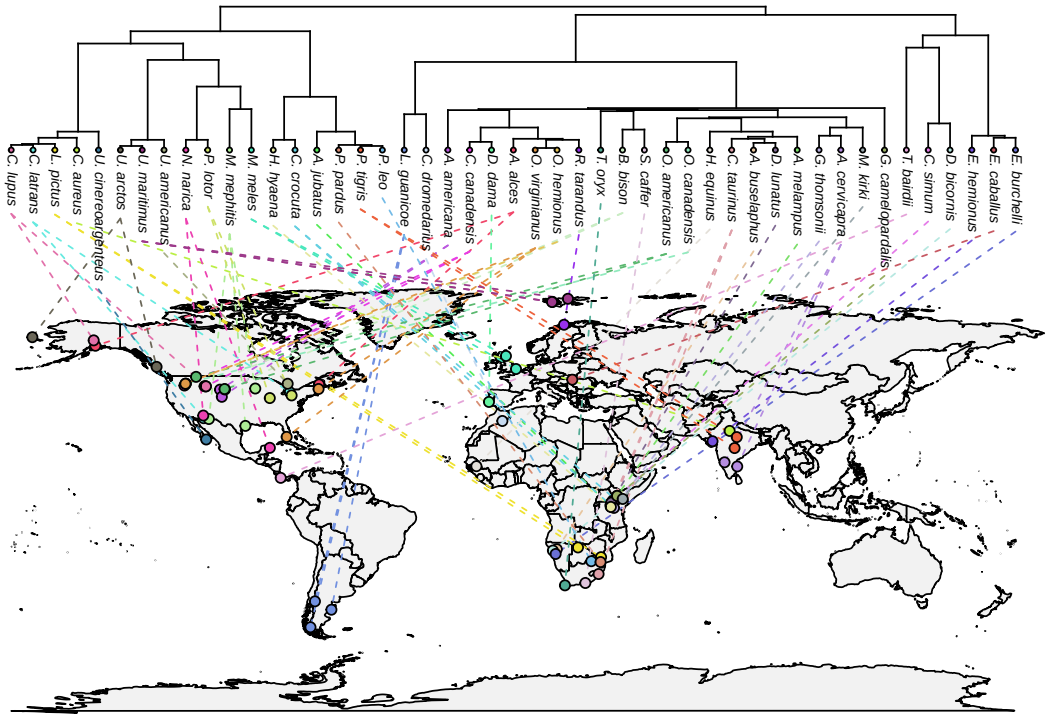

Figure 19. A phylogenetic tree of mammals projected onto a geographic map. Mammal phylogeny is from Garland et al. (1992), and the geographic coordinates were selected from citizen science records. See main text for more details.

7.3 Projecting trees into phenotypic space

Along with projecting a phylogenetic tree onto a geographic map (as we just saw), and projecting traits onto the edges and nodes of a plotted tree, the *phytools* package also contains multiple methods to project a tree into a space defined by our traits. Undoubtedly, the most popular of these are `phylomorphospace`, which projects a tree into a bivariate quantitative trait space (Sidlauskas 2008; e.g., Friedman et al. 2016; Martins et al. 2021), and `phenogram`, which projects a tree into a space defined by time since the root on the horizontal and phenotype on the vertical (typically called a "traitgram," see Evans et al. 2009; e.g., Martinez et al. 2020; Chazot et al. 2021). We saw how `phenogram` works in Figure 13b of any earlier section of this article. Here, I'll focus on the *phytools* method `phylomorphospace`.

For this example I'll use a time-calibrated phylogeny of 11 vertebrate species from the *TimeTree* website (Hedges et al. 2006), and a phenotypic trait dataset of body mass (in kg) and mean litter size. (The latter dataset was generated from *Wikipedia* and other sources: i.e., "Googling it.")

```
data(vertebrate.tree)
data(vertebrate.data)
head(vertebrate.data)
```

```
##                              Mass Length Litter_size
## Carcharodon_carcharias 2268.00  6.100        10.0
## Carassius_auratus         0.91  0.380       200.0
## Latimeria_chalumnae      80.00  2.000        15.0
## Iguana_iguana             8.00  2.000        50.5
## Turdus_migratorius        0.28  0.094         4.0
## Homo_sapiens             80.00  1.700         1.1
```

(We should see that our data frame actually has three columns – but henceforward I'll just use the first and the third of these.)

Normally, we could pass our data frame or matrix and phylogeny directly to the `phylomorphospace` function and obtain a plot. `phylomorphospace` would then undertake to project the tree, using Maximum Likelihood reconstructed ancestral values for both traits as the positions for internal nodes. I'd prefer, however, to first reconstruct ancestral states on a log scale, back-transform my estimated values to the original space, and then use these back-transformed reconstruction as my node positions. Fortunately, `phylomorphospace` allows that!

For the first step, I'll use the *phytools* ancestral state estimation function `fastAnc`. `fastAnc` computes Maximum Likelihood ancestral states for one input character vector at a time, so we just need to iterate across the two columns (of interest) in our data frame using an `apply` call as follows.

```
vertebrate.ace<-exp(apply(log(
  vertebrate.data[,c(1,3)]),2,
  fastAnc,tree=vertebrate.tree))
vertebrate.ace
```

```
##           Mass Litter_size
## 12  25.571594   15.552422
## 13  17.834793   16.114126
## 14  16.827622   14.481309
## 15   8.801275    8.791375
## 16  20.362215    1.653955
## 17  17.335294    1.442396
## 18  24.604666    1.610416
## 19 152.078728    1.737334
## 20 301.219076    1.582780
## 21   6.329908    9.613237
```

This gives us a set of reconstructed values on our original (linear) scale, but in which the reconstruction was performed on a *log* scale, and then back-transformed. Finally, let's create our phylomorphospace plot.

```
par(mar=c(5.1,4.1,0.6,2.1))
phylomorphospace(vertebrate.tree,
  vertebrate.data[,c(1,3)],
  A=vertebrate.ace,log="xy",
  xlim=c(1e-4,1e6),ylim=c(0.5,200),
  bty="n",label="off",axes=FALSE,
  xlab="Mass (kg)",ylab="Litter size",
  node.size=c(0,0))
axis(1,at=10^seq(-3,5,by=2),
  labels=prettyNum(10^seq(-3,5,by=2),
  big.mark=","),las=1,cex.axis=0.6)
axis(2,at=10^seq(0,2,by=1),
  labels=prettyNum(10^seq(0,2,by=1),
  big.mark=","),
  las=1,cex.axis=0.7)
cols<-setNames(RColorBrewer::brewer.pal(
  nrow(vertebrate.data),"Paired"),
  rownames(vertebrate.data))
points(vertebrate.data[,c(1,3)],pch=21,
  bg=cols,cex=1.2)
ind<-order(rownames(vertebrate.data))
legend("topleft",gsub("_"," ",
  rownames(vertebrate.data))[ind],
  pch=21,pt.cex=1.2,pt.bg=cols[ind],
  cex=0.6,bty="n",text.font=3)
```

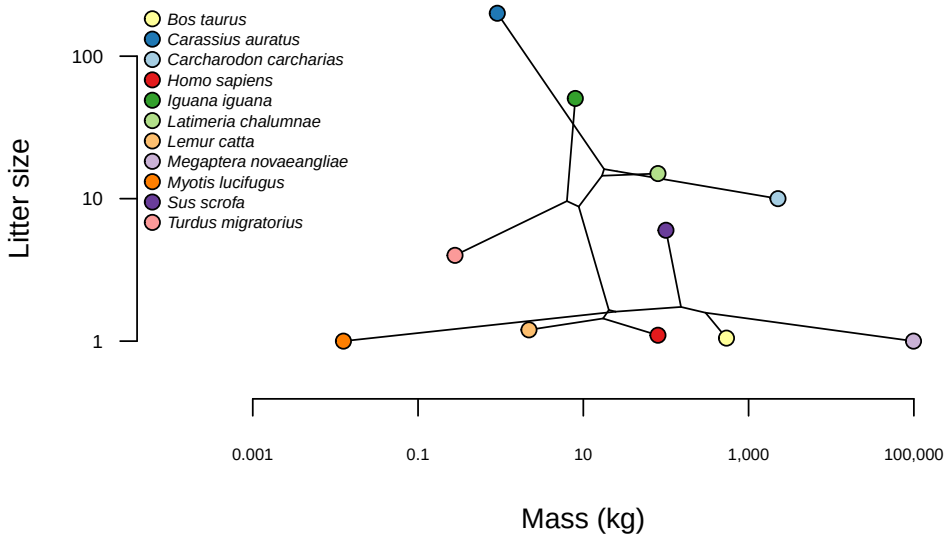

Figure 20. Phylomorphospace of body mass and litter size for a selection of vertebrate species. See main text for additional details.

In this code, I chose to graph the projection without taxon labels, then add different colored points and a legend to put the label information back on the plot (sorting my labels alphabetically as I did this). As readers can probably imagine, taxon labels on a phylomorphospace plot can easily become very messy – particularly for larger trees!

7.4 Plotting phenotypic data at the tips of the tree

In addition to projecting phylogenies into trait spaces, and plotting observed or reconstructed trait values on the tree, *phytools* can also help us undertake the (at least, conceptually) simple task of visualizing comparative trait data for species at the tips of the tree.

This might be done in various ways. For instance, we could graph the values of a quantitative trait adjacent to the tip labels using a bar or box, or we might plot the presence or absence of different lineages from a habitat type next to the tips of the tree. Numerous such approaches have been developed and implemented in the *phytools* package, and many of these are shown in my recent book (Revell and Harmon 2022).

Here, I'll illustrate just one such method in which a color gradient is used to visualize trait values for a set of quantitative characters at the tips of the tree. The *phytools* implementation of this plotting method is called `phylo.heatmap`, and it's been used in numerous published articles (e.g., Goelen et al. 2020; Hultgren et al. 2021; Molina-Mora et al. 2021; Huang et al. 2022). For this example, we'll use a phylogenetic tree and morphological trait dataset for *Anolis* lizards from (Mahler et al. 2010). To load these data, readers should run the following.

```
data(anoletree)
data(anole.data)
head(anole.data)
```

```
##                SVL      HL     HLL     FLL     LAM      TL
## ahli       4.03913 2.88266 3.96202 3.34498 2.86620 4.50400
## allogus    4.04014 2.86103 3.94018 3.33829 2.80827 4.52189
## rubribarbus 4.07847 2.89425 3.96135 3.35641 2.86751 4.56108
## imias      4.09969 2.85293 3.98565 3.41402 2.94375 4.65242
## sagrei     4.06716 2.83515 3.85786 3.24267 2.91872 4.77603
## bremeri    4.11337 2.86044 3.90039 3.30585 2.97009 4.72996
```

Our trait data object, `anole.data`, is a data frame with six trait columns for various phylogenetic traits.

We could visualize our data directly; however, the effect of overall size (data column `"SVL"`) would tend to obscure any interesting patterns of residual variation and covariation in body shape among the species in our tree. As such, primarily in an effort to control for overall size, I'll first run a phylogenetic principal components analysis (Revell 2009) using the *phytools* function `phyl.pca`. A phylogenetic principal components analysis (mentioned earlier with respect to the Broeckhoven et al. 2016 dataset) is similar to a regular PCA except that we account for non-independence of the information for different species in our data rotation (Revell 2009).

```
anole.ppca<-phyl.pca(anoletree,anole.data,
  mode="corr")
anole.ppca
```

```
## Phylogenetic pca
## Standard deviations:
##       PC1       PC2       PC3       PC4       PC5       PC6
## 2.2896942 0.6674345 0.4381067 0.2997973 0.1395612 0.1026573
## Loads:
##             PC1         PC2         PC3         PC4           PC5
## SVL -0.9782776 -0.01988115  0.14487425 -0.11332244  0.0781070110
## HL  -0.9736568 -0.03879982  0.13442473 -0.15596460 -0.0852979941
## HLL -0.9711545  0.14491400  0.02151524  0.17058611 -0.0588208480
## FLL -0.9759133 -0.02087140  0.14486273  0.14149988  0.0475205990
## LAM -0.8299594 -0.50437051 -0.23796010  0.01194704  0.0004983465
## TL  -0.8679195  0.40956428 -0.27350654 -0.05871034  0.0195584629
##              PC6
## SVL -0.051442939
## HL   0.028570939
## HLL -0.053257988
## FLL  0.062386141
## LAM -0.003133966
## TL   0.018373275
```

Our PC loadings show us the the first principal component dimension is strongly negatively correlated with all of the traits in our analysis. We could consider this the "size" axis. Principal component 2 is most strongly (negatively) correlated with the character `"LAM"`, number of adhesive toepad scales called lamellae; and most positively correlated with `"TL"`, tail length.

Let's compute the principal component scores for all of our species.

```
anole.pc_scores<-scores(anole.ppca)
head(anole.pc_scores)
```

```
##                     PC1       PC2         PC3       PC4         PC5
## ahli        -0.1747576 0.8697064  1.52379491 1.6029659 -0.23955421
## allogus      0.1646585 1.4017806  1.74506491 1.5358005 -0.08089546
## rubribarbus -0.4925001 1.0413268  1.45866163 1.4180850 -0.05716104
## imias       -1.1608049 0.7514380  0.75822327 1.7127381  0.35013533
## sagrei      -0.3486332 1.2632997 -0.05102313 0.7317455  0.37217463
## bremeri     -0.9714818 0.6943196  0.11689334 0.9290039  0.43486041
##                     PC6
## ahli        -0.1626049
## allogus     -0.1245855
## rubribarbus -0.1797060
## imias       -0.1340347
## sagrei      -0.1484157
## bremeri     -0.2304513
```

Since the sign of each principal component is arbitrary (principal components are vectors), we'll now "flip" the sign of PC 1 – so that it switches from negative size to simply "size."

```
anole.pc_scores[,1]<--anole.pc_scores[,1]
head(anole.pc_scores)
```

```
##                    PC1       PC2         PC3       PC4         PC5
## ahli         0.1747576 0.8697064  1.52379491 1.6029659 -0.23955421
## allogus     -0.1646585 1.4017806  1.74506491 1.5358005 -0.08089546
## rubribarbus  0.4925001 1.0413268  1.45866163 1.4180850 -0.05716104
## imias        1.1608049 0.7514380  0.75822327 1.7127381  0.35013533
## sagrei       0.3486332 1.2632997 -0.05102313 0.7317455  0.37217463
## bremeri      0.9714818 0.6943196  0.11689334 0.9290039  0.43486041
##                    PC6
## ahli        -0.1626049
## allogus     -0.1245855
## rubribarbus -0.1797060
## imias       -0.1340347
## sagrei      -0.1484157
## bremeri     -0.2304513
```

Finally, let's graph our results using the `phylo.heatmap` function. Seeing as the variance in our different principal component dimensions are quite different from PC to PC, we'll standardize them to have a constant variance using `standardize=TRUE`. As we've done in other exercises of this article, we can update the default color palette of the plot using the argument `colors`. Here, I'll use the colorblind-friendly `viridis` color palette from the *viridisLite* package (Garnier et al. 2022).

```
phylo.heatmap(anoletree,anole.pc_scores,
  standardize=TRUE,fsize=c(0.4,0.7,0.7),
  pts=FALSE,split=c(0.6,0.4),
  colors=viridisLite::viridis(n=40,
  direction=-1),mar=rep(0.1,4))
```

In Figure 21 we can already begin to discern some of the interesting ecomorphological phenotypic patterns of *Anolis* lizards (Losos 2009). For instance, the largest species (known as "crown-giants"), that is, those species with the highest values of PC 1, tend to have moderate or low values for PC 2: meaning they have larger lamellae and shorter tails, controlling for their body size. By contrast some of the smallest species (on PC 1) have among the highest values for PC 2 (tail length). These are the "grass-bush" anoles that perch on grass and bushes near the grown, and use their long tails to control body pitch while jumping. Neat!

8 RELATIONSHIP OF *PHYTOOLS* TO OTHER PACKAGES

The *phytools* package has grown to become (along with *ape*, *phangorn*, and *geiger*) among the most important core packages for phylogenetic analysis in the R environment. As of the time of writing, the original publication describing *phytools* (Revell, 2012) had been cited more than 6,700 times on *Google Scholar* and continues to be cited over 1,000 times per year.

In many respect, however, *phytools* owes its existence to a number of other packages making up the R phylogenetics ecosytem and from which it imports crtical functionality. In particular, *phytools* depends on the object classes and methods of the core R phylogenetics package, *ape* (Paradis et al. 2004; Popescu et al. 2012; Paradis and Schliep 2019). In addition, *phytools* relies on a number of different methods from the multifunctional phylogenetic inference package, *phangorn* (Schliep 2011). Finally, *phytools* is designed to interact with a variety of other function R phylogenetics libraries, especially the *geiger* package (Harmon et al. 2008; Pennell et al. 2014), which *phytools* "suggests" but does not import.

Outside phylogenetics as such, *phytools* also presently depends on or imports from a number of other R packages including *clusterGeneration* (Qiu and Joe. 2020), *coda* (Plummer et al. 2006), *combinat*

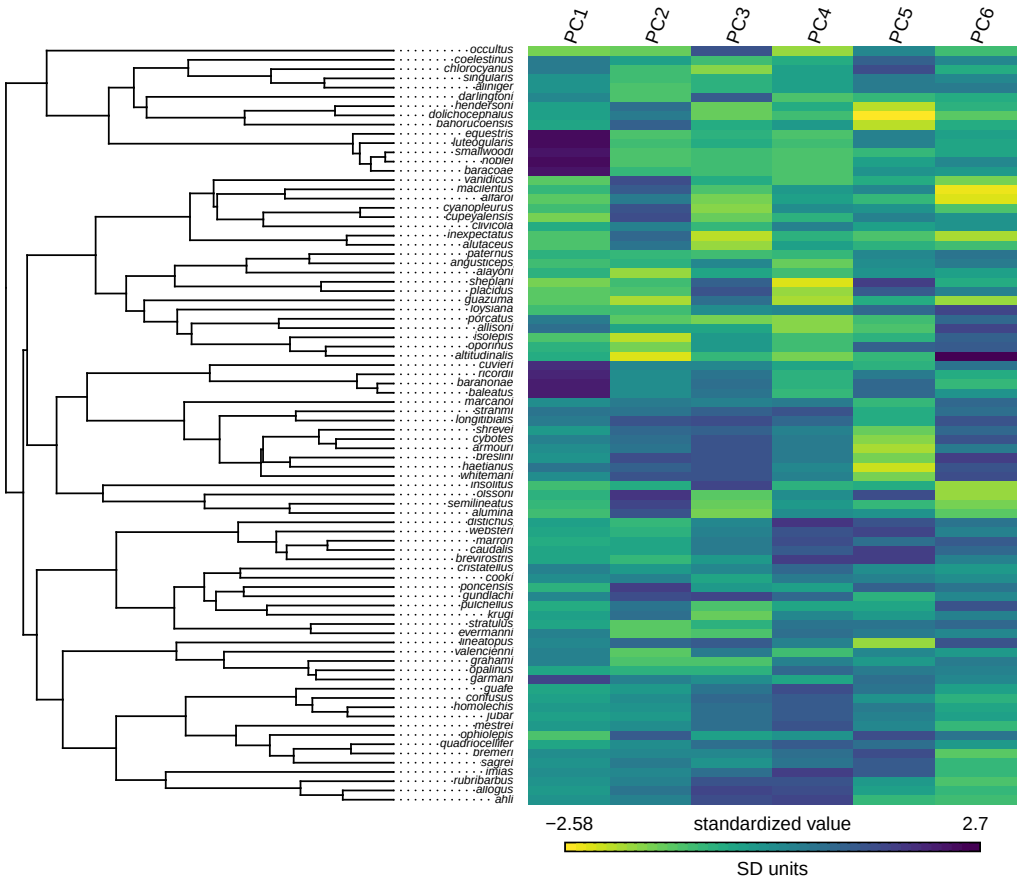

Figure 21. Phylogenetic heatmap showing principal components from a phylogenetic PCA of six morphological traits of *Anolis* lizards. Tree and data from Mahler et al. (2010). See main text for additional details.

(Chasalow 2012), *doParallel* (Corporation and Weston 2022), *expm* (Maechler et al. 2023), *foreach* (Microsoft and Weston 2022), *maps* (Becker et al. 2022), *MASS* (Venables and Ripley 2002), *mnormt* (Azzalini and Genz 2022), *nlme* (Pinheiro and Bates 2000; Pinheiro et al. 2022), *numDeriv* (Gilbert and Varadhan 2019), *optimParallel* (Gerber and Furrer 2019), *plotrix* (J 2006), and *scatterplot3d* (Ligges and Mächler 2003), although dependency relationships are dynamic as packages evolve and may change.

9 CONCLUSIONS

More than a decade has passed since the original and only article describing *phytools* was published (Revell 2012). Since that time, the *phytools* package has both evolved into one of the core function libraries of the R phylogenetics ecosystem, and expanded manyfold in size and scope. As such, I decided the literature reference for *phytools* was sorely in need of updating. In creating one, however, I was determined to make something that could serve as more than a placeholder to capture citations of the *phytools* package. I hope that what I've provided here will help guide new *phytools* users towards interesting analytical tools, as well as perhaps inspire experienced *phytools* and R phylogenetics researchers to generate new types of questions and data that will in turn help motivate continued development of the *phytools* package into the future.

10 SOFTWARE AND DATA AVAILABILITY

The *phytools* is free and open source, and can be downloaded from its CRAN (`https://CRAN.R-project.org/package=phytools`) or GitHub (`https://github.com/liamrevell/phytools`) pages.

This article was written in Rmarkdown (Xie et al. 2018, 2020; Allaire et al. 2023), and developed with the help of both *bookdown* (Xie 2016, 2023) and the posit Rstudio IDE (RStudio Team 2020).

Markdown code and data files necessary to exactly reproduce the analyses of this article are available at `https://github.com/liamrevell/Revell.phytools-v2/`.

REFERENCES

Allaire, J., Y. Xie, J. McPherson, J. Luraschi, K. Ushey, A. Atkins, H. Wickham, J. Cheng, W. Chang, and R. Iannone. 2023. Rmarkdown: Dynamic documents for r.

Amarasinghe, P., S. Joshi, N. Page, L. S. Wijedasa, M. Merello, H. Kathriarachchi, R. D. Stone, W. Judd, U. Kodandaramaiah, and N. Cellinese. 2021. Evolution and biogeography of memecylon. Am. J. Bot. 108:628–646. Wiley.

Ammar, R. 2019. randomcoloR: Generate attractive random colors.

Atkinson, C. L., B. C. Ee, and J. M. Pfeiffer. 2020. Evolutionary history drives aspects of stoichiometric niche variation and functional effects within a guild. Ecology 101. Wiley.

Azzalini, A., and A. Genz. 2022. The R package `mnormt`: The multivariate normal and *t* distributions (version 2.1.1).

Baele, G., S. Dellicour, M. A. Suchard, P. Lemey, and B. Vrancken. 2018. Recent advances in computational phylodynamics. Curr. Opin. Virol. 31:24–32. Elsevier BV.

Beale, M. A., M. Marks, S. K. Sahi, L. C. Tantalo, A. V. Nori, P. French, S. A. Lukehart, C. M. Marra, and N. R. Thomson. 2019. Genomic epidemiology of syphilis reveals independent emergence of macrolide resistance across multiple circulating lineages. Nat. Commun. 10:3255. Springer Science; Business Media LLC.

Beaulieu, J. M., and B. C. O'Meara. 2016. Detecting hidden diversification shifts in models of Trait-Dependent speciation and extinction. Syst. Biol. 65:583–601.

Beaulieu, J. M., B. C. O'Meara, and M. J. Donoghue. 2013. Identifying hidden rate changes in the evolution of a binary morphological character: The evolution of plant habit in campanulid angiosperms. Syst. Biol. 62:725–737.

Becker, R. A., A. R. Wilks, R. Brownrigg, T. P. Minka, and A. Deckmyn. 2022. Maps: Draw geographical maps.

Bentz, C., D. Dediu, A. Verkerk, and G. Jäger. 2018. The evolution of language families is shaped by the environment beyond neutral drift. Nat. Hum. Behav. 2:816–821. Springer Science; Business Media LLC.

Blinkhorn, J., and M. Grove. 2021. Explanations of variability in middle stone age stone tool assemblage composition and raw material use in eastern africa. Archaeol. Anthropol. Sci. 13. Springer Science; Business Media LLC.

Blomberg, S. P., T. Garland Jr, and A. R. Ives. 2003. Testing for phylogenetic signal in comparative data: Behavioral traits are more labile. Evolution 57:717–745.

Broeckhoven, C., G. Diedericks, C. Hui, B. G. Makhubo, and P. le Fras N. Mouton. 2016. Enemy at the gates: Rapid defensive trait diversification in an adaptive radiation of lizards. Evolution 70:2647–2656. Wiley.

Bushman, F. D., K. McCormick, and S. Sherrill-Mix. 2019. Virus structures constrain transmission modes. Nat. Microbiol. 4:1778–1780. Springer Science; Business Media LLC.

Butler, M. A., and A. A. King. 2004. Phylogenetic comparative analysis: A modeling approach for adaptive evolution. Am. Nat. 164:683–695.

Caraballo, D. A. 2022. Cross-species transmission of bat coronaviruses in the americas: Contrasting patterns between alphacoronavirus and betacoronavirus. Microbiol. Spectr. 10:e0141122. American Society for Microbiology.

Chasalow, S. 2012. Combinat: Combinatorics utilities.

Chazot, N., P. Blandin, V. Debat, M. Elias, and F. L. Condamine. 2021. Punctuational ecological changes rather than global factors drive species diversification and the evolution of wing phenotypes in morpho butterflies. J. Evol. Biol. 34:1592–1607. Wiley.

Collar, D. C., P. C. Wainwright, M. E. Alfaro, L. J. Revell, and R. S. Mehta. 2014. Biting disrupts integration to spur skull evolution in eels. Nature. nature.com.

Compton, Z., V. Harris, W. Mellon, S. Rupp, D. Mallo, S. E. Kapsetaki, M. Wilmot, R. Kennington, K. Noble, C. Baciu, L. Ramirez, A. Peraza, B. Martins, S. Sudhakar, S. Aksoy, G. Furukawa, O. Vincze, M. Giraudeau, E. G. Duke, S. Spiro, E. Flach, H. Davidson, A. Zehnder, T. A. Graham, B. Troan, T. M. Harrison, M. Tollis, J. D. Schiffman, A. Aktipis, L. M. Abegglen, C. C. Maley, and A. M. Boddy. 2023. Cancer prevalence across vertebrates. bioRxivorg.

Corporation, M., and S. Weston. 2022. doParallel: Foreach parallel adaptor for the 'parallel' package.

Csosz, S., and B. Fisher. 2016. Taxonomic revision of the malagasy nesomyrmex madecassus species-group using a quantitative morphometric approach. ZooKeys 603:105–130. Pensoft Publishers.

Efron, B., and G. Gong. 1983. A leisurely look at the bootstrap, the jackknife, and cross-validation. Am. Stat. 37:36–48. Informa UK Limited.

Endara, M.-J., J. A. Nicholls, P. D. Coley, D. L. Forrister, G. C. Younkin, K. G. Dexter, C. A. Kidner, R. T. Pennington, G. N. Stone, and T. A. Kursar. 2018. Tracking of host defenses and phylogeny during the radiation of neotropical inga-feeding sawflies (hymenoptera; argidae). Front. Plant Sci. 9:1237. Frontiers Media SA.

Esquerré, D., J. S. Keogh, D. Demangel, M. Morando, L. J. Avila, J. W. Sites Jr, F. Ferri-Yáñez, and A. D. Leaché. 2019. Speciation across mountains: Phylogenomics, species delimitation and taxonomy of the *Liolaemus leopardinus* clade (Squamata, Liolaemidae).

Etienne, R. S., and B. Haegeman. 2012. A conceptual and statistical framework for adaptive radiations with a key role for diversity dependence. Am. Nat. 180:E75–89.

Evans, M. E. K., S. A. Smith, R. S. Flynn, and M. J. Donoghue. 2009. Climate, niche evolution, and diversification of the "Bird-Cage" evening primroses (Oenothera, sections Anogra and Kleinia). Am. Nat. 173:225–240. The University of Chicago Press.

Felsenstein, J. 2012. A comparative method for both discrete and continuous characters using the threshold model. Am. Nat. 179:145–156.

Felsenstein, J. 2002. Graphical methods for visualizing comparative data on phylogenies. Pp. 27–44 *in* N. MacLeod and P. Forey, eds. Morphology, shape and phylogeny. Taylor; Francis, London, UK.

Felsenstein, J. 2004. Inferring phylogenies. Sinauer associates, Sunderland, MA.

Felsenstein, J. 1985. Phylogenies and the comparative method. Am. Nat. 125:1–15.

Felsenstein, J. 2005. Using the quantitative genetic threshold model for inferences between and within species. Philos. Trans. R. Soc. Lond. B Biol. Sci. 360:1427–1434.

FitzJohn, R. G. 2012. Diversitree : Comparative phylogenetic analyses of diversification in R. Methods Ecol. Evol. 3:1084–1092. Wiley.

FitzJohn, R. G. 2010. Quantitative traits and diversification. Syst. Biol. 59:619–633. Oxford University Press (OUP).

Freitas, A. R., A. P. Tedim, C. Novais, V. F. Lanza, and L. Peixe. 2020. Comparative genomics of global optrA-carrying enterococcus faecalis uncovers a common chromosomal hotspot for optrA acquisition within a diversity of core and accessory genomes. Microb. Genom. 6. Microbiology Society.

Friedman, S. T., S. A. Price, A. S. Hoey, and P. C. Wainwright. 2016. Ecomorphological convergence in planktivorous surgeonfishes. J. Evol. Biol. 29:965–978.

Garamszegi, L. Z. 2014. Modern phylogenetic comparative methods and their application in evolutionary biology: Concepts and practice. Springer.

Garland, T., P. H. Harvey, and A. R. Ives. 1992. Procedures for the analysis of comparative data using phylogenetically independent contrasts. Syst. Biol. 41:18–32. Oxford University Press.

Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer, and Cédric. 2022. viridis - colorblind-friendly color maps for r.

Gerber, F., and R. Furrer. 2019. optimParallel: An r package providing a parallel version of the l-BFGS-b optimization method. The R Journal 11:352–358.

Gilbert, P., and R. Varadhan. 2019. numDeriv: Accurate numerical derivatives.

Goelen, T., I. S. Sobhy, C. Vanderaa, F. Wäckers, H. Rediers, T. Wenseleers, H. Jacquemyn, and B. Lievens. 2020. Bacterial phylogeny predicts volatile organic compound composition and olfactory response of an aphid parasitoid. Oikos 129:1415–1428. Wiley.

Goldberg, E. E., and B. Igić. 2012. Tempo and mode in plant breeding system evolution. Evolution 66:3701–3709. Wiley Online Library.

Halali, S., E. Bergen, C. J. Breuker, P. M. Brakefield, and O. Brattström. 2020. Seasonal environments drive convergent evolution of a faster pace-of-life in tropical butterflies. Ecology Letters 24:102–112. Wiley.

Harmon, L. J. 2019. Phylogenetic comparative methods: Learning from trees. Ecoevorxiv.

Harmon, L. J., J. T. Weir, C. D. Brock, R. E. Glor, and W. Challenger. 2008. GEIGER: Investigating evolutionary radiations. Bioinformatics 24:129–131. Oxford University Press.

Harvey, P. H., and M. D. Pagel. 1991. The comparative method in evolutionary biology. Oxford University Press.

Hedges, S. B., J. Dudley, and S. Kumar. 2006. TimeTree: A public knowledge-base of divergence times among organisms. Bioinformatics 22:2971–2972. Oxford University Press (OUP).

Hermanson, G., F. V. Iori, S. W. Evers, M. C. Langer, and G. S. Ferreira. 2020. A small podocnemidoid (pleurodira, pelomedusoides) from the late cretaceous of brazil, and the innervation and carotid circulation of side-necked turtles. Pap. Palaeontol. 6:329–347. Wiley.

Hohenlohe, P. A., and S. J. Arnold. 2008. MIPoD: A hypothesis-testing framework for microevolutionary inference from patterns of divergence. Am. Nat. 171:366–385.

Huang, J.-P., and B. Morgan. 2021. Evolution of adult male horn developmental phenotypes and character displacement in xylotrupes beetles (scarabaeidae). Ecol. Evol. 11:5503–5510. Wiley.

Huang, X., Z. Chen, G. Yang, C. Xia, Q. Luo, X. Gao, and L. Dong. 2022. Assemblages of plasmodium and related parasites in birds with different migration statuses. Int. J. Mol. Sci. 23:10277. MDPI AG.

Huelsenbeck, J. P., R. Nielsen, and J. P. Bollback. 2003. Stochastic mapping of morphological characters. Syst. Biol. 52:131–158.

Huey, R. B., T. Garland Jr, and M. Turelli. 2019. Revisiting a key innovation in evolutionary biology: Felsenstein's "phylogenies and the comparative method." Am. Nat. 193:755–772. The University of Chicago Press Chicago, IL.

Huie, J. M., I. Prates, R. C. Bell, and K. de Queiroz. 2021. Convergent patterns of adaptive radiation between island and mainland anolis lizards. Biol. J. Linn. Soc. Lond. 134:85–110. Oxford University Press (OUP).

Hultgren, K. M., S. T. C. Chak, J. Bjelajac, and K. S. Macdonald. 2021. Correlated evolution of larval development, egg size and genome size across two genera of snapping shrimp. J. Evol. Biol. 34:1827–1839. Wiley.

J, L. 2006. Plotrix: A package in the red light district of r. R-News 6:8–12.

Jezovit, J. A., R. Rooke, J. Schneider, and J. D. Levine. 2020. Behavioral and environmental contributions to drosophilid social networks. Proc. Natl. Acad. Sci. U. S. A. 117:11573–11583. Proceedings of the National Academy of Sciences.

Kirk, E. C., and R. F. Kay. 2004. The evolution of high visual acuity in the anthropoidea. Pp. 539–602 *in* C. F. Ross and R. F. Kay, eds. Anthropoid origins: New visions. Springer US, Boston, MA.

Lewis, P. O. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. Syst. Biol. 50:913–925.

Ligges, U., and M. Mächler. 2003. Scatterplot3d - an r package for visualizing multivariate data. Journal of Statistical Software 8:1–20.

Losos, J. 2009. Lizards in an evolutionary tree: Ecology and adaptive radiation of anoles. University of California Press.

Maddison, W. P., P. E. Midford, and S. P. Otto. 2007. Estimating a binary character's effect on speciation and extinction. Syst. Biol. 56:701–710. sysbio.oxfordjournals.org.

Maechler, M., C. Dutang, and V. Goulet. 2023. Expm: Matrix exponential, log, 'etc'.

Mahler, D. L., L. J. Revell, R. E. Glor, and J. B. Losos. 2010. Ecological opportunity and the rate of morphological evolution in the diversification of greater antillean anoles. Evolution 64:2731–2745.

Martinez, Q., J. Clavel, J. A. Esselstyn, A. S. Achmadi, C. Grohe, N. Pirot, and P.-H. Fabre. 2020. Convergent evolution of olfactory and thermoregulatory capacities in small amphibious mammals. Proc. Natl. Acad. Sci. U. S. A. 117:8958–8965. Proceedings of the National Academy of Sciences.

Martins, B. O., L. Franco-Belussi, M. S. Siqueira, C. E. Fernandes, and D. B. Provete. 2021. The evolution of red blood cell shape in fishes. J. Evol. Biol. 34:537–548. Wiley.

McLaughlin, A., V. Montoya, R. L. Miller, G. J. Mordecai, Canadian COVID-19 Genomics Network (CanCOGen) Consortium, M. Worobey, A. F. Y. Poon, and J. B. Joy. 2022. Genomic epidemiology of the first two waves of SARS-CoV-2 in canada. Elife 11. eLife Sciences Publications, Ltd.

Medina, I., and N. E. Langmore. 2016. The evolution of host specialisation in avian brood parasites. Ecology Letters 19:1110–1118. Wiley.

Microsoft, and S. Weston. 2022. Foreach: Provides foreach looping construct.

Mifsud, J. C. O., V. A. Costa, M. E. Petrone, E. M. Marzinelli, E. C. Holmes, and E. Harvey. 2023. Transcriptome mining extends the host range of the flaviviridae to non-bilaterians. Virus Evol. 9:veac124. Oxford University Press (OUP).

Molina-Mora, J. A., D. Chinchilla-Montero, R. Garcia-Batan, and F. Garcia. 2021. Genomic context of the two integrons of ST-111 pseudomonas aeruginosa AG1: A VIM-2-carrying old-acquaintance and a novel IMP-18-carrying integron. Infect. Genet. Evol. 89:104740. Elsevier BV.

Morlon, H., M. D. Potts, and J. B. Plotkin. 2010. Inferring the dynamics of diversification: A coalescent approach. PLoS Biol. 8.

Moura, A., A. Criscuolo, H. Pouseele, M. M. Maury, A. Leclercq, C. Tarr, J. T. Björkman, T. Dallman, A. Reimer, V. Enouf, E. Larsonneur, H. Carleton, H. Bracq-Dieye, L. S. Katz, L. Jones, M. Touchon, M. Tourdjman, M. Walker, S. Stroika, T. Cantinelli, V. Chenal-Francisque, Z. Kucerova, E. P. C. Rocha, C. Nadon, K. Grant, E. M. Nielsen, B. Pot, P. Gerner-Smidt, M. Lecuit, and S. Brisse. 2016. Whole genome-based population biology and epidemiological surveillance of listeria monocytogenes. Nat. Microbiol. 2:16185.

Nee, S., R. M. May, and P. H. Harvey. 1994. The reconstructed evolutionary process. Philos. Trans. R. Soc. Lond. B Biol. Sci. 344:305–311.

Neuwirth, E. 2022. RColorBrewer: ColorBrewer palettes.

Nunn, C. L. 2011. The comparative approach in evolutionary anthropology and biology. University of Chicago Press.

O'Meara, B. C., C. Ané, M. J. Sanderson, and P. C. Wainwright. 2006. Testing for different rates of continuous trait evolution using likelihood. Evolution 60:922–933.

Page, R. D. M. 1993. Parasites, phylogeny and cospeciation. International Journal for Parasitology 23:499–506. Elsevier BV.

Pagel, M. 1994. Detecting correlated evolution on phylogenies: A general method for the comparative analysis of discrete characters. Proc. R. Soc. Lond. B Biol. Sci. 255:37–45. The Royal Society.

Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature 401:877–884.

Paradis, E., J. Claude, and K. Strimmer. 2004. APE: Analyses of phylogenetics and evolution in R language. Bioinformatics 20:289–290. academic.oup.com.

Paradis, E., and K. Schliep. 2019. Ape 5.0: An environment for modern phylogenetics and evolutionary analyses in R. Bioinformatics 35:526–528. Oxford University Press.

Pennell, M. W., J. M. Eastman, G. J. Slater, J. W. Brown, J. C. Uyeda, R. G. FitzJohn, M. E. Alfaro, and L. J. Harmon. 2014. Geiger v2. 0: An expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. Bioinformatics 30:2216–2218. Oxford University Press.

Pepke, M. L., and D. T. A. Eisenberg. 2022. On the comparative biology of mammalian telomeres: Telomere length co-evolves with body mass, lifespan and cancer risk. Mol. Ecol. 31:6286–6296. Wiley.

Pinheiro, J. C., and D. M. Bates. 2000. Mixed-effects models in s and s-PLUS. Springer, New York.

Pinheiro, J., D. Bates, and R Core Team. 2022. Nlme: Linear and nonlinear mixed effects models.

Plummer, M., N. Best, K. Cowles, and K. Vines. 2006. CODA: Convergence diagnosis and output analysis for MCMC. R News 6:7–11.

Popescu, A.-A., K. T. Huber, and E. Paradis. 2012. Ape 3.0: New tools for distance-based phylogenetics and evolutionary analysis in R. Bioinformatics 28:1536–1537. Oxford University Press (OUP).

Pozzi, L., M. Voskamp, and P. M. Kappeler. 2022. The effects of body size, activity, and phylogeny on primate sleeping ecology. Am. J. Biol. Anthropol. 179:598–608. Wiley.

Pybus, O. G., and P. H. Harvey. 2000. Testing macro-evolutionary models using incomplete molecular phylogenies. Proc. Biol. Sci. 267:2267–2272.

Qiu, W., and H. Joe. 2020. clusterGeneration: Random cluster generation (with specified degree of separation).

Quach, Q. N., R. G. Reynolds, and L. J. Revell. 2019. Historical allopatry and secondary contact or primary intergradation in the puerto rican crested anole, anolis cristatellus, on vieques island in the caribbean. Biol. J. Linn. Soc. Lond. Oxford University Press (OUP).

R Core Team. 2022. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Rabosky, D. L. 2014. Automatic detection of key innovations, rate shifts, anddiversity-dependence on phylogenetic trees. PLoS One 9:e89543. Public Library of Science.

Revell, L. J. 2021. A variable-rate quantitative trait evolution model using penalized-likelihood. PeerJ 9:e11997. PeerJ.

Revell, L. J. 2014. Ancestral character estimation under the threshold model from quantitative genetics. Evolution 68:743–759.

Revell, L. J. 2018. Comparing the rates of speciation and extinction between phylogenetic trees. Ecol. Evol. 8:5303–5312.

Revell, L. J. 2012. Phytools: An R package for phylogenetic comparative biology (and other things). Methods Ecol. Evol. 3:217–223. Wiley Online Library.

Revell, L. J. 2009. Size-correction and principal components for interspecific comparative studies. Evolution 63:3258–3268. Wiley Online Library.

Revell, L. J. 2013. Two new graphical methods for mapping trait evolution on phylogenies. Methods Ecol. Evol. 4:754–759. Wiley.

Revell, L. J., and D. C. Collar. 2009. Phylogenetic analysis of the evolutionary correlation using likelihood. Evolution 63:1090–1100.

Revell, L. J., L. E. Gonzalez-Valenzuela, A. Alfonso, L. A. Castellanos-Garcia, C. E. Guarnizo, and A. J. Crawford. 2018. Comparing evolutionary rates between trees, clades and traits. Methods Ecol. Evol. 9:994–1005. Wiley Online Library.

Revell, L. J., and L. J. Harmon. 2022. Phylogenetic comparative methods in R. Princeton University Press, Princeton, NJ.

Revell, L. J., L. J. Harmon, and D. C. Collar. 2008. Phylogenetic signal, evolutionary process, and rate. Syst. Biol. 57:591–601.

Revell, L. J., and A. S. Harrison. 2008. PCCA: A program for phylogenetic canonical correlation analysis. Bioinformatics 24:1018–1020. Oxford University Press (OUP).

Revell, L. J., M. A. Johnson, J. A. Schulte 2nd, J. J. Kolbe, and J. B. Losos. 2007. A phylogenetic test for adaptive convergence in rock-dwelling lizards. Evolution 61:2898–2912. Wiley.

Revell, L. J., D. L. Mahler, P. R. Peres-Neto, and B. D. Redelings. 2012. A new phylogenetic method for identifying exceptional phenotypic diversification. Evolution 66:135–146.

Revell, L. J., D. L. Mahler, R. G. Reynolds, and G. J. Slater. 2015. Placing cryptic, recently extinct, or hypothesized taxa into an ultrametric phylogeny using continuous character data: A case study with the lizard anolis roosevelti. Evolution 69:1027–1035. Wiley.

Revell, L. J., K. S. Toyama, and D. L. Mahler. 2022. A simple hierarchical model for heterogeneity in the evolutionary correlation on a phylogenetic tree. PeerJ 10:e13910. PeerJ.

RStudio Team. 2020. RStudio: Integrated development environment for r. RStudio, PBC., Boston, MA.

Sánchez-Busó, L., D. Golparian, J. Corander, Y. H. Grad, M. Ohnishi, R. Flemming, J. Parkhill, S. D. Bentley, M. Unemo, and S. R. Harris. 2019. The impact of antimicrobials on gonococcal evolution. Nat. Microbiol. 4:1941–1950. Springer Science; Business Media LLC.

Sander, P. M., E. M. Griebeler, N. Klein, J. V. Juarbe, T. Wintrich, L. J. Revell, and L. Schmitz. 2021. Early giant reveals faster evolution of large body size in ichthyosaurs than in cetaceans. Science 374:eabf5787. American Association for the Advancement of Science (AAAS).

Schliep, K. P. 2011. Phangorn: Phylogenetic analysis in R. Bioinformatics 27:592–593.

Schluter, D., T. Price, A. Ø. Mooers, and D. Ludwig. 1997. Likelihood of ancestor states in adaptive radiation. Evolution 51:1699–1711. Wiley.

Sidlauskas, B. 2008. Continuous and arrested morphological diversification in sister clades of characiform fishes: A phylomorphospace approach. Evolution 62:3135–3156. Wiley.

Stadler, T. 2013. How can we improve accuracy of macroevolutionary rate estimates? Syst. Biol. 62:321–329. Oxford University Press (OUP).

Stadler, T. 2011. Inferring speciation and extinction processes from extant species data. Proc. Natl. Acad. Sci. U. S. A. 108:16145–16146. National Acad Sciences.

Stull, G. W., P. S. Soltis, D. E. Soltis, M. A. Gitzendanner, and S. A. Smith. 2020. Nuclear phylogenomic analyses of asterids conflict with plastome trees and support novel relationships among major lineages. American Journal of Botany 107:790–805. Wiley.

Uyeda, J. C., and L. J. Harmon. 2014. A novel Bayesian method for inferring and interpreting the dynamics of adaptive landscapes from phylogenetic comparative data. Syst. Biol. 63:902–918.

Valles-Colomer, M., G. Falony, Y. Darzi, E. F. Tigchelaar, J. Wang, R. Y. Tito, C. Schiweck, A. Kurilshikov, M. Joossens, C. Wijmenga, S. Claes, L. Van Oudenhove, A. Zhernakova, S. Vieira-Silva, and J. Raes. 2019. The neuroactive potential of the human gut microbiota in quality of life and depression. Nat. Microbiol. 4:623–632. Springer Science; Business Media LLC.

Van Borm, S., G. Boseret, S. Dellicour, M. Steensels, V. Roupie, F. Vandenbussche, E. Mathijs, A. Vilain, M. Driesen, M. Dispas, A. W. Delcloo, P. Lemey, I. Mertens, M. Gilbert, B. Lambrecht, and T. van den Berg. 2023. Combined phylogeographic analyses and epidemiologic contact tracing to characterize atypically pathogenic avian influenza (H3N1) epidemic, belgium, 2019. Emerg. Infect. Dis. 29:351–359. Centers for Disease Control; Prevention (CDC).

Venables, W. N., and B. D. Ripley. 2002. Modern applied statistics with s. Fourth. Springer, New York.

Xie, Y. 2023. Bookdown: Authoring books and technical documents with r markdown.

Xie, Y. 2016. Bookdown: Authoring books and technical documents with R markdown. Chapman; Hall/CRC, Boca Raton, Florida.

Xie, Y., J. J. Allaire, and G. Grolemund. 2018. R markdown: The definitive guide. Chapman; Hall/CRC, Boca Raton, Florida.

Xie, Y., C. Dervieux, and E. Riederer. 2020. R markdown cookbook. Chapman; Hall/CRC, Boca Raton, Florida.

Yang, Z. 2014. Molecular evolution. Oxford University Press, London, England.