Introduction to phylogenetic comparative methods in R: models of continuous character evolution

Graham Slater (gslater@uchicago.edu)

This document aims to provide an overview of how continuous trait evolution is modeled on phylogenies and how to use R to answer phylogenetic comparative questions. By the end of this session you should :

1. Understand the basic math and probability theory underlying random walks and their application to traits on trees
2. Be able to fit simple Brownian motion models of evolution to continuous data
3. Understand what an Ornstein-Uhlenbeck process is and how the parameters influence the long term behavior of the process
4. know how to fit simple Ornstein-Uhlenbeck models of evolution to continuous data
5. Understand what an evolutionary rate matrix is, and how to estimate / visualize it.

1 Discrete Random Walks - a coin-tossing example.

Random walks form the basis for nearly all models of continuous trait evolution. The mathematics of random walks have been extensively studied in physics and can be complex. We need only a basic understanding of random walks to apply statistical models to our phylogenetic data. To introduce the theory of random walks, we will consider a discrete random walk using a coin-toss.

We'll generate a random walk by tossing a quarter. Starting at a value of 0, if we toss a heads, we'll move +1 steps and if we toss a tails, we'll move -1. After 15 flips, I see something like Figure 1

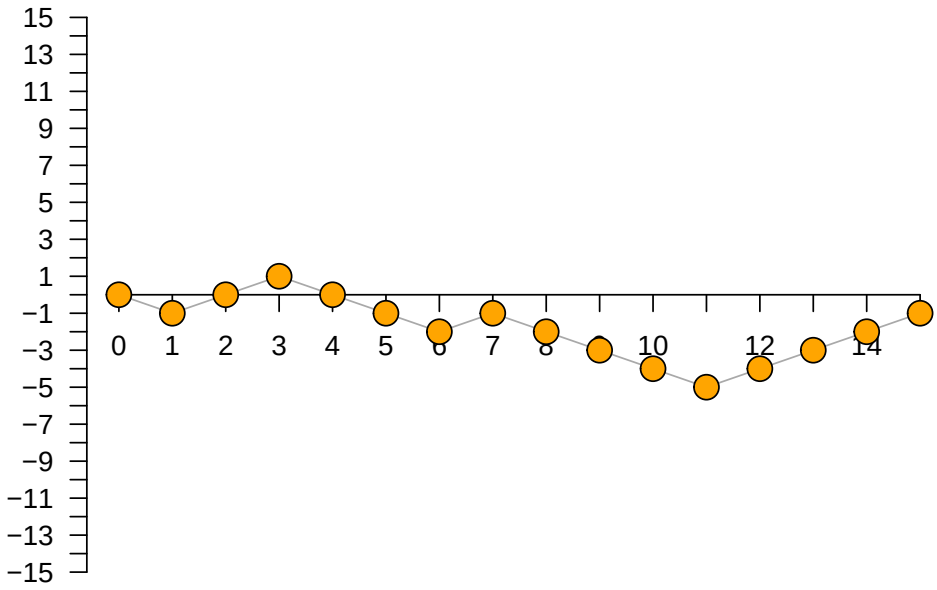


Figure 1: One realization of a discrete random walk with a step size of +/- 1.

You probably saw something different. And if you repeat the process, the outcome will be different again (try doing this to convince yourself). Why does this happen? The answer is that coin flipping is a game of chance and so multiple iterations are likely to generate different outcomes. But with that said, there must be some *expectation* for the process, right? In other words, if we repeat the 15 flips many, many times there must be one value that we see more frequently than any other. Indeed there is, and working out what that expectation and its variance is central to understanding how to fit random walk models to our trait data.

We know that for any given flip, there are only two possible outcomes; a heads, which yields a +1, and a tails which yields a -1. If we assume the coin is fair, each outcome occurs with probability of 0.5. We can write this formally as

$$\Pr(X = 1) = \Pr(X = -1) = 0.5. \tag{1}$$

Remember, this is just a fancy way of saying the probability of a heads is the same as the probability of a tails, which is equal to 0.5.

Because the flip can only take one of two discrete states, heads and tails (or +1 and -1 depending on which outcome you consider), X is what we call a Discrete Random Variable. The expected outcome or expected value ($\mathbb{E}$) for a discrete random variable is just a weighted average of its possible states, where the weights are the probabilities associated with those states. If there are j states, we would write this down as

$$\mathbb{E}[X] = \sum_{i=1}^{j} X_i Pr_i. \tag{2}$$

So the expected value can be obtained by multiplying each outcome by its probability and then summing those terms. For our coin example, we have two states (+1 and -1) and each has a probability of 0.5. Thus the expected value of a single flip is

$$\mathbb{E}[\text{X}] = (+1 * 0.5) + (\text{-}1 * 05)$$
$$\mathbb{E}[\text{X}] = 0.5 \text{ - } 0.5 = 0$$

The expected value for one flip is zero. How is that possible when zero isn't a possible outcome? Well, no one said the expected value had to be a possible outcome! But it is the average of the two outcomes. After two flips the expected value is 0 + 0. If the coin really is fair, a +1 and -1 *is* a reasonable outcome, right? For n flips, the expected value will be $n\mathbb{E}[X]$, which is still zero.

Now the $\mathbb{E}[X]$ gives us the expected value but each run of 15 flips generates a different final number. Can we characterize the spread? The variance of a discrete random variable is also calculated as a weighted average but this time it involves weighting the squared differences between the values the discrete random variable can take and the expectation (the sample variance) by the associated probability. We write this as

$$Var(X) = \sum_{i=1}^{j} (X_j - \mathbb{E}[X])^2 * Pr(X = X_j), \tag{3}$$

and for a single coin toss it would be

$$Var(X) = [(+1 - 0)^2 * 0.5] + [(-1 - 0)^2 * 0.5]$$
$$Var(X) = [1 * 0.5] + [1 * 0.5] = 1.$$

Again, this should make sense - for a single coin toss, the expectation is zero and the variance is 1 (giving -1 and +1 as the variance). For our 15 coin run, we can get a variance by taking advantage

of the fact that the variance of a series of indepedent events is just the sum of their individual variances. Thus, for 15 flips the expected outcome is 0 and the variance around that estimate is 15 (the standard deviation is 3.87).

What happens if we decrease the step size by half? Let's keep our fair coin but we'll move +0.5 for a heads and -0.5 for a tails. We can do the experiment but to understand what changes and why, we should do the math too. What does your intuition say should happen?

We know the expected value is a weighted sum of all possible steps, where the weights are the probabilities. Our steps are now +/- 0.5 but the probabilities stay the same, so

$$\mathbb{E}[X] = +0.5 * 0.5 + (-0.5 * 0.5) = 0.25 - 0.25 = 0.$$

If you predicted that the expected value wouldn't change, you're correct. It makes sense - the probability of positive or negative steps is the same and the step sizes are equal so they should cancel out. How does the variance change though? We calculated the variance as a weighted sum of the squared deviations from the expected value, so

$$Var(X) = [0.5 * (+0.5 - 0)^2] + [0.5 * (-0.5 - 0)^2] = 0.125 + 0.125 = 0.25$$

Did that result surprise you? Following the math, it shouldn't; for an unbiased random walk in one dimension, the expected variance is the square of the step size. This means that step sizes > 1 give $>>$ variances.

One more hypothetical - what happens if the coin is biased? For example, if the probability of heads is 0.7 and tails is 0.3? let's go back to our +1/-1 outcomes to keep things simple.

Again, we know how to calculate the expected value of a discrete random variable as a weighted sum of the possible outcomes,

$$\mathbb{E}[X] = +1 * 0.7 + (-1 * 0.3) = 0.7 - 0.3 = 0.4$$

What changed? Well, now the probability of a positive step is greater than the probability of a negative step, so our expectation changes accordingly. What would the expected outcome be after 15 steps?

Does the variance change too?

$$Var(X) = 0.7 * (+1 - 0.4)^2 + 0.3 * (-1 - 0.4)^2 = (0.7 * 0.36) + (0.3 * 1.96) = 0.84$$

Yes, it does a bit. After 15 reps, we'd have an expected value of 6 for this biased random walk and a variance of ~ 12.6.

2 Brownian motion as a continuous random walk

We began thinking about random walks as a way of modeling traits that take a range of continuously varying values. The general ideas we just discussed regarding discrete random variables can be applied to such quantitative traits, but we need to move away from to continuous random walks.

Let's imagine we are studying a population of organisms we are interested in their body size, which has a population mean of 1 (on a log scale) and a variance of 0.5. Let's imagine a slightly contrived scenario in which a random (wrt size) subset of the population breed, that size is perfectly heritable, and that all of the adults then die to be replaced by the offspring in the next generation. We'll aslo assume there's no selection. Can we predict the mean size for the population in the following generation? How about after 15 generations?

We can simulate this process, just like we did with the coin flipping. It'll be easier to do this by coding now, though. If we know the starting value and variance, we could get the value in the next generation by drawing from a random normal distribution with mean and sd based on the population parameters. That'll make it easy to produce the first generation but we'd need to know what the mean is in the next generation too. We can make the simulation process easier by fixing our starting value as 1, drawing random normal deviates from a distribution centered on 0, and then adding them all up. This is bascially what we did with the coin flipping.

```
> time <- seq(1,15,1) # define 15 time steps
> deviates<- rnorm(n= 15, mean=0, sd=sqrt(0.5))
> plot(c(0, time), cumsum(c(1, deviates)), type="b", xlab="time",
+      ylab="trait value", ylim=c(-5,5))
```

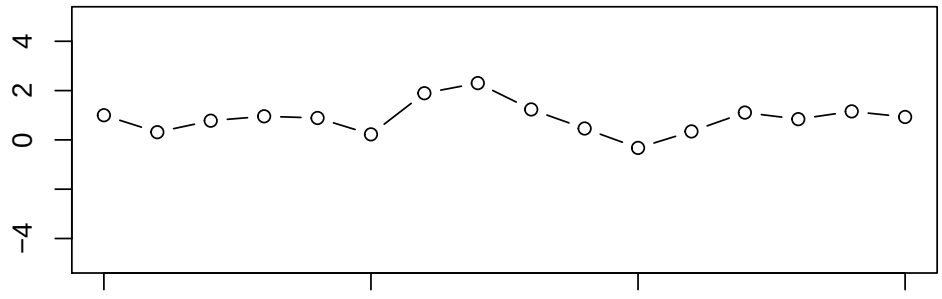


Figure 2: One realization of a continuous random walk from a starting value of 1.

As with the coin-flipping, your plot will possibly look quite different from mine. But the same set of outcomes emerge. Some of our walks increase, some decrease and some stay around the starting value. Based on the coin flipping, we might predict that the expected value after some number of generations should be the same as our starting value (i.e. no expected change) and that the variance of that estimate is proportional to time and the sample variance. We'd be correct, but calculating those values is a little less straightforward; we can't take all the possible sizes, multiply them by their probabilities, and add them up because size varies continuously, in theory between

-∞ and + ∞. But we can apply the same logic. The definite integral is the addition of all the little bits given by some function between two bounds, so we calculate a weighted integral

$$\mathbb{E}[X] = \int_{-\infty}^{+\infty} xf(x)dx \tag{4}$$

where $f(x)$ is the probability density of the tiny region dx around the value x. The logic of the discrete random walk tells us what this value will be without having to actually integrate the normal density function. A normal distribution is symmetric, which means that increases cancel decreases and the expectation must be 0. But remember we started this random walk at 1, so the expectation at any point in time t is 1.

Things get even more complex mathematically for the variance but we don't really need to do the math at all. $Var(x)$ has to be 0.5 because we specified it to be so. If you need to convince yourself...

```
> var(rnorm(n=10000, mean=0, sd=sqrt(0.5)))
```

```
[1] 0.5145617
```

will give you a number that converges on 0.5 as more reps are added. By using the additive property of independent variances. We can say that the variance of the mean after 15 time steps will be $15 * 0.5 = 7.5$. Again, we can convince ourselves that this is true by simulation

```
> foo <- function(x) sum(rnorm(n=15, mean=0, sd=sqrt(0.5)))
> var(sapply(seq(1,100000), FUN =  foo))
```

```
[1] 7.538896
```

We could repeat this exercise with different variances and with a bias (we'll come back to that later) but we need not do this now. What we've learned here already is sufficient to make three important points. First, For any unbiased random walk, the expected change after any number of time steps is zero and so the best estimate of the starting value will come from the sample mean. Second, in an unbiased random walk, increases in trait value are equally as probable as decreases of the same magnitude and so the outcome is normally distributed. Third, the variance associated with this outcome increases as the product of time and the sample variance - note that this means we could increase time for a given variance or increase variance for a given time to get the same range of results.

2.1 Random Walks and MLEs

Ok, so how does this all relate to models of trait evolution? Imagine now that we observe the trait values of 10 different species and that we know those species shared a common ancestor 10 million years ago. If we were willing to assume that the species evolved independently of one another the entire time (i.e. they evolved on a star phylogeny), then the **Maximum Likelihood** estimates

of the ancestral state and evolutionary rate can be found by finding the combination of mean and variance that maximize the likelihood of these data assuming a normal distribution.

Let's demonstrate this by simulating some data:

```
> root <- 2.4
> rate <- 0.1
> d <- rnorm(n=10, mean=root, sd = sqrt(10*rate))
```

and now we can write a function to evaluate the likelihood

```
> p <- c(0, log(1)) ## set up some initial values
> foo <- function(p) -sum(dnorm(x = d,mean = p[1], sd = sqrt(10*exp(p[2])), log = T))
> o <- optim(par = p,fn = foo)
> cat(paste("root MLE =", round(o$par[1],2)))

root MLE = 2.38

> cat(paste("rate MLE =", round(exp(o$par[2]),2)))

rate MLE = 0.07

> cat(paste("LogLikelihood =", round(-o$value,3)))

LogLikelihood = -12.132
```

So, we do ok but we're a bit off in our rate estimate. This intro to optimization might raise a few questions that need answers. For example, we optimize the negative log likelihood because optimization is normally a minimization procedure. And rates are easier to optimize on the log scale because they are bounded at zero and can sometimes get very small. But, of course, the bigger concern is that species aren't independent and so we need to use a different probability distribution that accounts for this co variation

3 Brownian motion on trees

For evolution along a single branch of a phylogeny, trait values are draws from a normal distribution. But on the complete phylogeny, trait values at the tips may not be independent of one another due to shared evolutionary history. Let's simulate a phylogeny and see what I mean.

```
> library(phytools);
> set.seed(1)
> phy <- pbtree(n=4)
> plot(phy) #plot the tree
> axisPhylo() #put up a scale bar
```

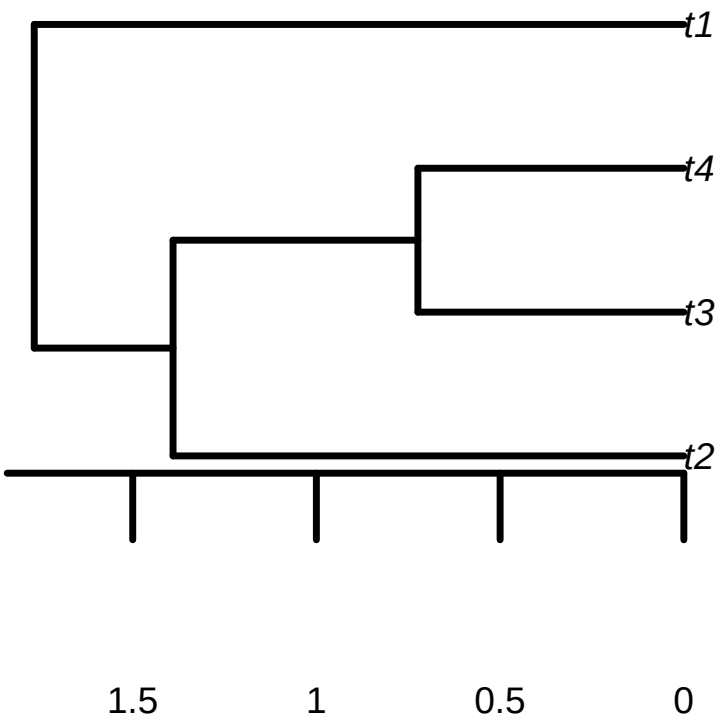


Figure 3: A simple phylogenetic tree.

We can represent this tree as a variance-covariance matrix where the off-diagonals of the matrix are the covariances between pairs of taxa (the shared evolutionary history) while diagonals are the variances (total amount of evolutionary history from origin for a pair of taxa). There are some functions in R that allow you to produce a variance-covariance matrix directly from a tree. Let's use ape's vcv function to get a vcv for our simulated tree

```
> vcv(phy);

          t2        t3        t4       t1
t2 1.7679163 0.3775909 0.3775909 0.000000
t3 0.3775909 1.7679163 1.0441742 0.000000
t4 0.3775909 1.0441742 1.7679163 0.000000
t1 0.0000000 0.0000000 0.0000000 1.767916
```

Does this make sense in comparison to the tree you plotted? It should - the diagonal elements

are all identical because this tree is ultrametric (all tips are equidistant from the root). The off-diagonals give the covariances; taxon 1 shares no common ancestry with the other taxa and so all of it's off-diagonal cells are 0. t3 and t4 share a very recent common ancestor, so they have the largest covariance (or most shared ancestry).

If the tree didn't exist, we could use the normal probability density function to calculate the likelihood of these data. But the tree does exist. The multivariate normal distribution is similar to the normal but allows for a covariance structure among observations and / or different mean values. remember this last point! Let's generate some simulated data on our little tree using the root and rate from above and then try to compute the likelihood of the data given possible rates and root states.

```
> root=0
> rate=0.1
> d <- fastBM(tree= phy, a = root, sig2=rate)
> d # take a look - we have four trait values

        t2         t3         t4         t1
-0.0594196  0.7689962  0.3486177 -0.4825507
```

As above, we could write a function to optimize the likelihood of the trait data given some combination of rate and root state. Functions like `fitContinuous` in the `geiger` package wrap this all up but let's write our own. We'll need to use the mvnorm likelihood function from the `mvtnorm` package.

```
> library(mvtnorm)
> p <- c(0, log(1)) ## set up some initial values
> foo <- function(p){
+   n <- length(d)
+   o <- - dmvnorm(x=d, mean=rep(p[1], n), sigma = exp(p[2]) * vcv(phy), log=TRUE)
+ }
> o <- optim(par = p,fn = foo)
> cat(paste("root MLE =", round(o$par[1],2)))

root MLE = 0.02

> cat(paste("rate MLE =", round(exp(o$par[2]),2)))

rate MLE = 0.13

> cat(paste("LogLikelihood =", round(-o$value,3)))

LogLikelihood = -2.419
```

Ok, that's great. but what does it actually tell us? We can look at the likelihood surface to see how peaky it is (or isn't) and compare our estimates to the true values. To do this, we will write a nested loop - for each row (root value), we'll compute the likelihood of every column (rate) and store it. This is super clunky but makes a point!

```
> root.tries <- seq(-2,2, 0.1)
> rate.tries = seq(0.01, 0.5, 0.01)
> res <- matrix(data = NA, nrow = length(root.tries), ncol = length(rate.tries))
> for(i in 1:nrow(res)){
+         for(j in 1:ncol(res)) {
+                 res[i,j] <- dmvnorm(x=d, mean = rep(root.tries[i], 4),
+                             sigma = vcv(phy) * rate.tries[j], log = T)
+         }
+ }
> contour.lines<-c(-1000, -500, -250, -100, -50, -25, -10, -5, -1, 0)
> contour(root.tries, rate.tries, res, levels = contour.lines, xlab = "root tries",
+         ylab= "rate tries")
> points(x=root, y= rate, pch = 21, bg = "red", cex=1); # plot the true values
> points(x= o$par[1], y= exp(o$par[2]), pch = 21, bg = "blue", cex=1); # plot the MLE values
>
```

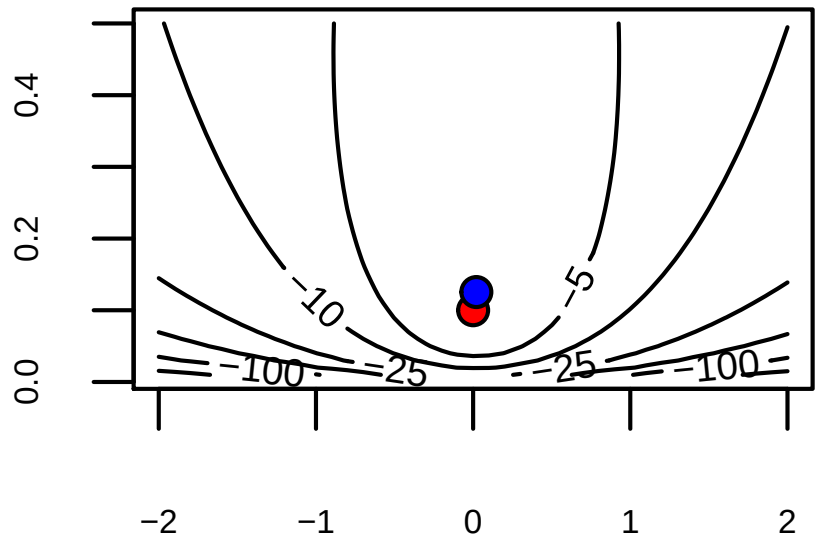


Figure 4: Likelihood surface

The likelihood surface shown in figure 4 tells us a few interesting things. We know that our estimate of the root state (blue circle) is pretty good compared to the true value (red circle) but our rate

estimate is off. The contours show how many log-likelihood units away from the optimum different combinations of values fall. These show that rates are just harder to estimate than the root state for this simple example; if both were estimated with comparable error, we'd see circles, but instead we see ellipses with the major axis oriented in the direction of rates. More data might fix this. But real data might not perform so well at estimating root states. Why?

3.1 Intermission - ancestral state estimation

A lot of people like to use phylogenies to reconstruct the ancestral morphology for their clade, either at the root node or at internal nodes of the tree. Reconstruction is a poor choice of term here; what is being obtained are estimates and have considerable uncertainty associated with them. Nonetheless, performing ancestral state estimation can be an interesting and somewhat informative exercise.

The methods we've used so far generate an estimation for one node in the tree - that of the root. Methods for inferring ancestral states are widely implemented. For example then `ape` package's `ace` function allows for ancestral state estimation using a variety of different algorithms (independent contrasts, rerooting etc). Furthermore, there are some neat tools in `phytools` for visualizing them.

Because PCMs are data hungry, there's a view that sampling more tips might lead to better ancestral state estimates. But that's not quite true. Cecile Ané showed in a 2008 paper that the effective sample size (n_e) when estimating the ancestral morphology at the root is given by

$$n_e \leq \frac{kT}{t}, \tag{5}$$

where k is the number of branches coming from the root node, T is the (mean) root to tip distance and t is the length of the shortest branch descended from the root node.

3.2 Extending Brownian motion I: time varying rates

. Note that the number of tips doesn't enter this expression at all! The way to best leverage data in estimating ancestral states is to break up the branches descended directly from the root by adding early diverging extant lineages or fossil taxa.

Now that we know how to fit a simple Brownian motion model to a tree and trait dataset, we might want to extend the model and make it more complicated. Scenarios we might consider are

1. whether the rate of evolution in one clade differs from the rate in the rest of the clade,
2. whether rates decline through time, consistent with traditional models of adaptive radiation, or
3. whether rates vary as a function of some other extrinsic variable, such as temperature.

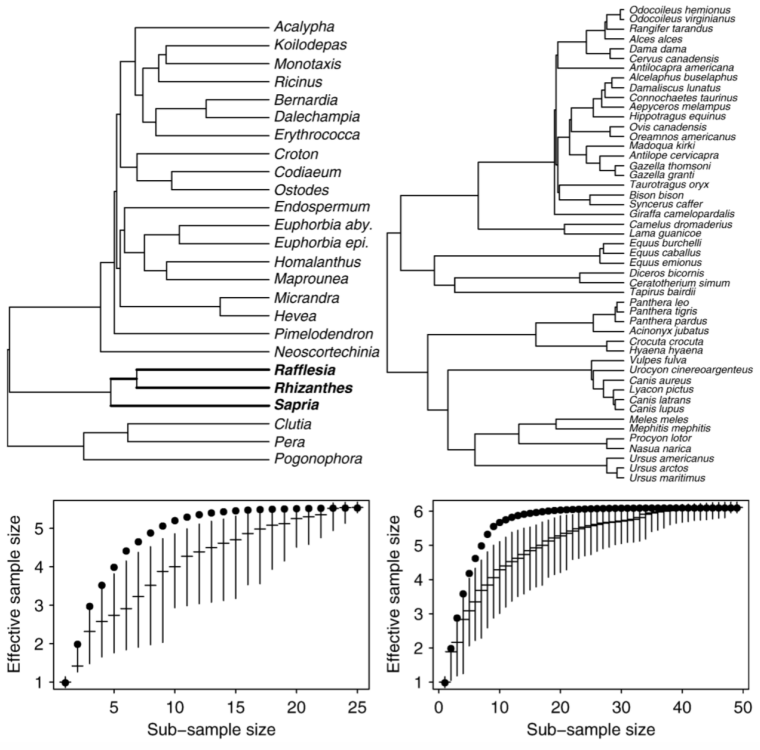
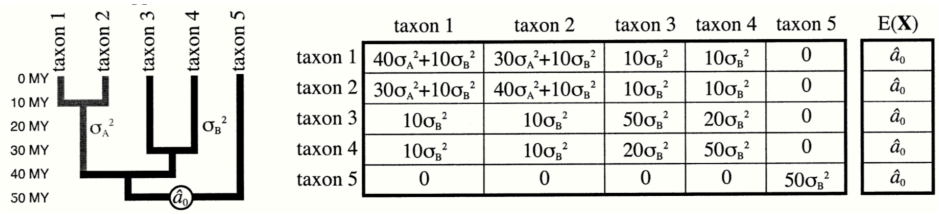


Figure 5: Effective sample sizes for ancestral state estimation are much lower than the number of tips in the tree. $n_e = 5.54$ for the tree of 25 taxa on the left and $n_e = 6.11$ for the tree of 49 taxa on the right. From Ané 2008.

Because rates (or step sizes) are just scalars - numbers that we use to adjust the covariance structure implied by the tree - it's straightforward to do all of these things. We just need to figure out how to do the appropriate adjustment.

Clade or regime specific shifts are something that many people are interested in. For example, O'Meara et al. (2006) considered a model in which genome size evolution was faster in Santalales than in other angiosperms. Alternatively, we might want to know if some ecological factor impacts the rate of evolution in another trait. For example Slater et al. (2010) considered a model in which diet influenced rates of body size evolution in cetaceans. Conceptually, these models are very simple. We know (but have not yet formalized) that the Brownian motion covariance structure ($\mathbf{V}$) for a clade is the product of its evolutionary rate σ^2 and phylogenetic covariance matrix $\mathbf{C}$. Adding clade or lineage specific rates is no more complicated than adding a new rate parameter that only applies to relevant branches.

Figure 6 shows an example where the clade comprising taxon 1, taxon 2, and the branch leading to them evolves under a distinct rate (σ^2_A) from the rest of the clade (σ^2_B). The elements of the model-specific covariance matrix reflect these differences. For example, the cell [taxon 1, taxon 2] multiplies the short, 10 myr branch from the root by the ancestral rate σ^2_B and the longer 30 Myr branch leading to their common ancestor by the new rate σ^2_A. Numerically, the problem now is to



	taxon 1	taxon 2	taxon 3	taxon 4	taxon 5	E(**X**)
taxon 1	$40\sigma_A^2+10\sigma_B^2$	$30\sigma_A^2+10\sigma_B^2$	$10\sigma_B^2$	$10\sigma_B^2$	0	$\hat{a}_0$
taxon 2	$30\sigma_A^2+10\sigma_B^2$	$40\sigma_A^2+10\sigma_B^2$	$10\sigma_B^2$	$10\sigma_B^2$	0	$\hat{a}_0$
taxon 3	$10\sigma_B^2$	$10\sigma_B^2$	$50\sigma_B^2$	$20\sigma_B^2$	0	$\hat{a}_0$
taxon 4	$10\sigma_B^2$	$10\sigma_B^2$	$20\sigma_B^2$	$50\sigma_B^2$	0	$\hat{a}_0$
taxon 5	0	0	0	0	$50\sigma_B^2$	$\hat{a}_0$

Figure 6: A two-rate regime where the clade (1,2) and its ancestral branch evolve under a different rate to the rest of the clade. From O'Meara et al. 2006

find the values of σ_A^2, σ_B^2, and the root state that maximize the likelihood of observing the trait data at the tips. The machinery we developed earlier is adequate to do this.

As an example, we will as a simple question - are rates of body size evolution faster in toothed or baleen whales. The whale data are in the geiger package. We'll need to clean them up first

```
> library(geiger)
> data(whales)
> #extract ln lengths and name them
> lnlength <- setNames(whales$dat[,2], rownames(whales$dat))
> # remove non-overlapping taxa from the tree and data
> td<-treedata(whales$phy, lnlength, sort=T)
> phy <- td$phy
> lnlength <- td$data
```

The baleen whales are tip numbers 64 through 75 so we'll create a vector of tooth types and then use stochastic mapping to generate a reconstruction of the trait on the tree

```
> clade <- setNames(object = c(rep("toothed", 63), rep("baleen", 12)), nm = phy$tip.label)
> whale.mapped<-make.simmap(tree=phy, x=clade)

make.simmap is sampling character histories conditioned on the transition matrix

Q =
              baleen      toothed
baleen  -0.001411352  0.001411352
toothed  0.001411352 -0.001411352
(estimated using likelihood);
and (mean) root node prior probabilities
pi =
 baleen toothed
    0.5     0.5

> # plotSimmap(whale.mapped, fsize=0.1)
```

We can now use the `phytools` function brownie.lite to fit the two rate model. What does the output tell you?

```
> brownie.lite(tree=whale.mapped, x=lnlength)

ML single-rate model:
        s^2        se         a         k        logL
value         0.012         0         1.7086         2         -23.1681

ML multi-rate model:
        s^2(baleen)         se(baleen)         s^2(toothed)         se(toothed)         a         k
value         0.0146         0         0.0115         0         1.7086         3         -23.0355

P-value (based on X^2): 0.6065

R thinks it has found the ML solution.
```

Sometimes we don't have an *a priori* hypothesis for where the shifts might be or what causes them, but we might want to ask whether there is evidence for rate shifts in the data. These kinds of ~~data dredging~~ exploratory methods require a different approach. By far the most common is to used a Bayesian approach called reversible jump MCMC (or rjMCMC) to explore the dimensionality of model space. The output of such models is typically a summarized in a way that allows the user to assess the posterior probability of a rate shift along a branch or at a node using the relative frequency with which that shift is sampled, as well as the average rates per branch. The `auteur` function in `geiger` is a rjMCMC approach for doing this. Standalone methods featuring rjMCMC trait inference are `BAMM` by Dan Rabosky and BayesTraits by Mark Pagel and colleagues. The `motmot` package for R by Gavin Thomas contains an approach called TraitMedusa that uses a stepwise-AIC procedure to identify rate shifts.

There's another kind of shift model that we might like to try. As a paleobiologist, I often think less about clade-specific shifts in rate and more about temporal changes. For example, we might predict that the Cenozoic radiation of mammals was facilitated by increased rates of evolution after the extinction of the dinosaurs that affected multiple lineages simultaneously.

Again, all this requires is that we can identify and scale elements of the variance–covariance matrix appropriately. This is easy to do. If we call our Mesozoic rate $\sigma^2_{mesozoic}$ and our Cenozoic rate $\sigma^2_{cenozoic}$ and the shift position is defined as the distance from the root (rather than back in time), the elements of the model matrix are just

$$\mathbf{V}_{ij} = min(shift, \mathbf{C}_{ij}) * \sigma^2_{mesozoic} + max(0, [\mathbf{C}_{ij} - shift]) * \sigma^2_{cenozoic} \quad (6)$$

All this is really saying is that for each cell in the VCV, we multiply the Mesozoic rate by either the value in the cell or the shift time (whichever is lowest because we want to max out at the shift point)

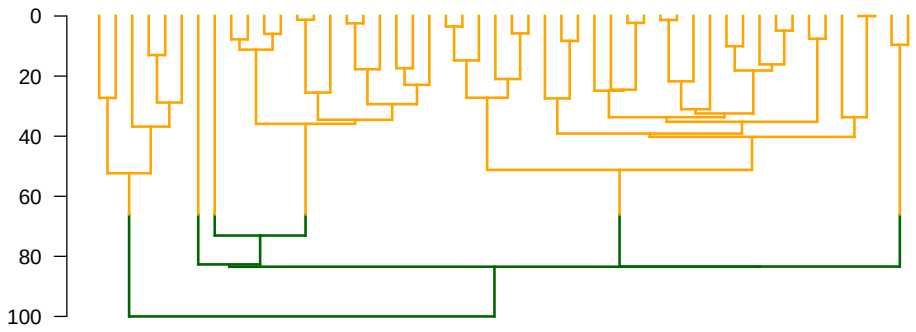


Figure 7: a tree in which Mesozoic lineages (green branches) are inferred to evolve at a different rate to Cenozoic lineages (orange).

and we add to that the product of the Cenozoic rate and the maximum of either the value in that cell minus the shift time (which will be positive if the branch extends in to the Cenozoic) or zero. This algorithm is suprisingly easy to code up yourself but can be implemented by using the `phytools` function `make.era.map` to generate a stochastically mapped tree and then using `brownie.lite` to fit the two rate model. A maximum likelihood approach that treats the shift point as a free parameter to be estimated from the data is also available in the `OUwie` function `OUwie.slice`.

The Early Burst model is a model of adaptive radiation in which rates decline exponentially as a function of time. If we start at time 0 with a rate of σ_0^2, the rate at time t is

$$\sigma_t^2 = \sigma_0^2 * e^{rt} \tag{7}$$

The new parameter here is r, which determines the strength of the change in rate with time. If $r = 0$, then e^{rt} is 1 and the rate does not change with time. If $r > 0$ then rates increase with time, while if $r < 0$ then rates decline. Figure 8 shows how rate declines for different values of r over the course of 100 myr. One thing to note here is that the timescale matters; if our clade were only 60 myr old, the implied differences between the bottom two curves would be greater. For this reason, its often more intuitive to transform your estimates of r into rate half-lives ($ln(2)/|r|$) which tell the amour of time it takes for the rate to fall to half of its initial value. If this number is large, relative to the age of your clade, the early burst cannot really be an "early burst".

In order to fit the early burst model, we can't just calculate a rate and multiply the covariance matrix by it. This is because the rate changes as a function of time and so we have to account for the fact that the rate is incrementally slowing (or speeding up). In other words, we need to integrate rate change along the branches.

This is easier than it sounds as we can leverage the fact that the exponential change in equation 7 is time–dependent and assumes that the process begins at time 0. Our phylogenetic covariance matrix also gives the time from the root to mrca of pairs of taxa in the off-diagonal elements and to the observation of the tip on the diagonals. Therefore, if we find the definite integral of equation 7, where the bounds are 0 and the age of the node or tip in question, then we can just multiple this value by σ_0^2, to get the early burst model covariance matrix.

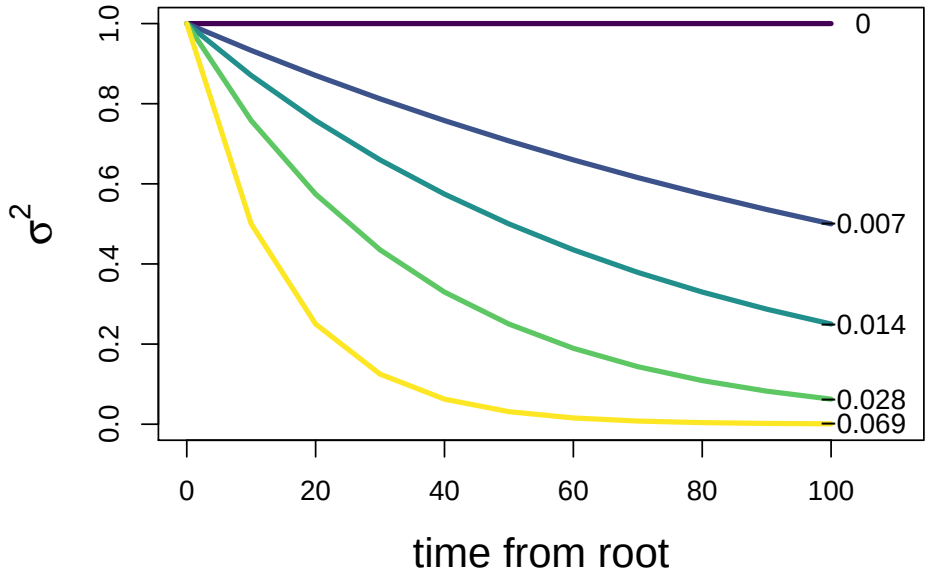


Figure 8: Rates change through time under an EB model

$$\mathbf{V}_{i,j} = \sigma_0^2 * \int_0^{\mathbf{C}_{i,j}} e^{rt} d(t) = \sigma_0^2 * \frac{e^{r\mathbf{C}_{i,j}} - 1}{r} \tag{8}$$

To fit an early burst model, we can use the `fitContinuous` functon in the `geiger` library. Let's try it with the whale dataset from before.

```
> whales.eb <- fitContinuous(phy=phy, dat=lnlength, model="EB")
> whales.eb$opt


$a
[1] -0.02270969

$sigsq
[1] 0.02386768

$z0
[1] 1.728077

$lnL
[1] -22.702

$method
[1] "subplex"
```

```
$k
[1] 3

$aic
[1] 51.404

$aicc
[1] 51.74203
```

The `opt` component of the output is where the fitted parameters and other model info are stored. Here, a is the r parameter, sigsq is the initial rate, and z0 is the root state. The MLE for a is negative, suggesting a slow down in rate but the rate half life is 30.1 Ma, which is about 80% the age of the clade. So maybe this isn't a great explanation for our data.

We could formulate other scenarios in which rate changes as a function of time. For example, maybe rate changes linearly, rather than exponentially. In this case, the rate at time t would be

$$\sigma_t^2 = \sigma_0^2 + bt, \tag{9}$$

where b is the rate of change in evolutionary rate with time. Again, we just need to integrate this function and take the definite integral to get the transformed branch lengths of the model covariance matrix,

$$\mathbf{V}_{i,j} = \sigma_0^2 + bt\, d(t) = \sigma_0^2 \mathbf{C}_{ij} + \frac{b\mathbf{C}_{ij}}{2}. \tag{10}$$

This model is implemented in `fitContinuous` as the trend model.

Because we're just integrating a function along branches of a tree, in theory we can come up with any model we like to explain rates of trait evolution. Clavel and Morlon (2017) implemented a model in which rates vary as a function of some extrinsic environmental factor, such as climate. In this kind of scenario, rates may increase and decrease over time, rather than follow a single monotonic process. To fit this model, you need a function that describes how your variable changes through time. Using the whale dataset we can model rates of body size evolution as a function of mean global temperature through time as extrapolated from the deep sea oxygen isotope record, which looks like this

The blue line in Figure 9 is a smooth spline, which is itself a function. Analytically integrating this is, well, gnarly but we let a computer do it numerically to help us figure out what the elements of the covariance matrix should be. The `RPANDA` package takes care of this for us.

```
> library(RPANDA)
> zd <- read.csv("~/Dropbox/MidWestPhylo/2008CompilationData.csv")
> head(zd)
```

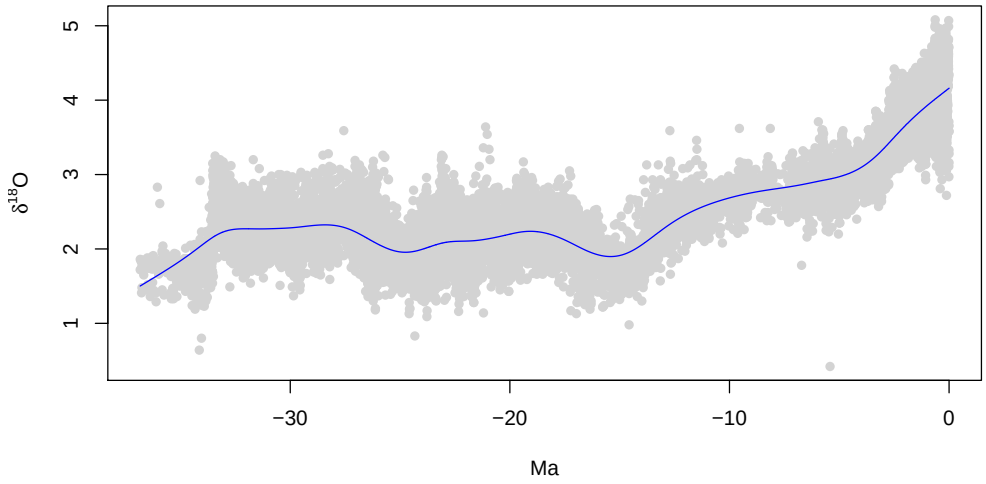


Figure 9: The oxygen isotope record from the beginning of the Oligocene to present.

```
  site   age genus oxygen carbon
1  607 0.000   CIB   3.28   0.88
2  849 0.000   CIB   3.58   0.15
3  607 0.002   CIB   3.16   0.96
4  659 0.002   CIB   2.91     NA
5  607 0.004   CIB   3.33   0.88
6  659 0.004   CIB   3.16     NA
```

```
> zd <- na.omit(zd) # remove rows with NA values
> root.age<- max(diag(vcv(phy))) # get root age from the vcv matrix
> zd<-subset(zd, age < root.age) # chuck out data older than the clade
> spl <- smooth.spline(zd$age, zd$oxygen, df = 15)
> xx <- cbind(spl$x, spl$y)
> plot(-zd$age, zd$oxygen, pch=16, col="lightgray")
> lines(-xx[,1],xx[,2], col="blue")
> temp.rate <- fit_t_env(phylo = phy, data = setNames(lnlength[,1], rownames(lnlength)), env_da
+                        method="Nelder-Mead", control=list(maxit=20000))
> temp.rate
```

```
successful convergence of the optimizer
a reliable solution has been reached

LogLikelihood:          -22.7663
AIC:           51.5326
AICc:           51.87063
3 parameters

Beta:
```

```
----------------------
0.01979781

Brownian rate (sigma):

----------------------
0.02324947

Estimated root states:

----------------------
1.712374
```

The results suggest that rates increase linearly with increases in the Oxygen isotope value, which corresponds to a decrease in temperature. So rates are inferred to be faster as temperatures cool.

This is all really exciting stuff, but you should think very carefully about whether these models are appropriate reflections of your hypotheses. From the coin flipping example, we know that a rate increase results in an increase in the expected variance but not the mean. So if rates speed up with cooling temperatures that means you should expect more variability in your traits. Maybe that's what you expect, in which case power to you. But often times people get rates and variances confused with trends and changes in the expected value. Just be prepared to defend your choice!

3.3 Extending Brownian motion II: trends

Paleontologists have long been interested in detecting trends. One prominent example is Cope's rule, the tendency of vertebrates to, on average, increase is size over their evolutionary history. The "rule" has been the subject of much debate, with some arguing it's a real phenomenon and others arguing that it is sampling artifact due to unceasing variance through time or that it is due to the fact that most clades originate at small size and so the mean must increase.

Our biased coin example suggests a way of testing this idea. We saw there that the expected value increased as a function of time, with the expected at time t being equal to the starting value + the number of time steps multiplied by the change over one time step. Formally,

$$\mathbb{E}[X_t] = x_0 + mt. \tag{11}$$

In this expression, x_0 is just our root state and m is the change in the expected value per unit time. Therefore, when calculating the likelihood of the data using the mulivariate normal density function, instead of using a vector of root states as the expected value for each tip, we use a vector in which the expected value for each is derived using the expression above. Importantly, the rate isn't changing here so we can find the maximum likelihood estimate of the rate in the normal way.

Is there a problem here? It depends on what kind of data you have. If you have fossils in your tree, you're fine. But if you have only extant taxa then there is a problem - all your ts are the same, which means all your expected values will be the same. The result is that the model is unidentifiable; Trends cannot be detected in clades of extant taxa because the maximum likelihood estimate of the trend parameter will be 0!

Is there a way around this? Kind of. If you have some idea what the value at the root should be, you could fix it or put a prior on it and use ML to estimate or Bayesian inference to sample from the posterior of the trend parameter. This approach works - but you need the information and you need to be right about where in the tree it goes.

Trends are worth being aware of in another capacity - ancestral state estimation. If trait evolution followed a trend but you only have extant taxa, your ancestral state will be affected by it (Figure 10). This is why the earliest work looking at the influence of fossil taxa in PCMs was focused on ancestral trait estimates rather than model inference.

```
> trend.tree <- pbtree(n=20)
> trend.tree<-rescale(trend.tree, "depth", 10)
> trend.data <- fastBM(tree = trend.tree, a = 0,mu =1, sig2 = 0.1,internal = T)
> phenogram(tree = trend.tree, x = trend.data[1:20],ylim=c(0,15),
+           fsize=0.1,spread.labels=FALSE) # plot based on tips only
> phenogram(tree=trend.tree, x = trend.data,ylim=c(0,15), add=T,
+           colors="red", lty=2, spread.labels=F) # add known states
```

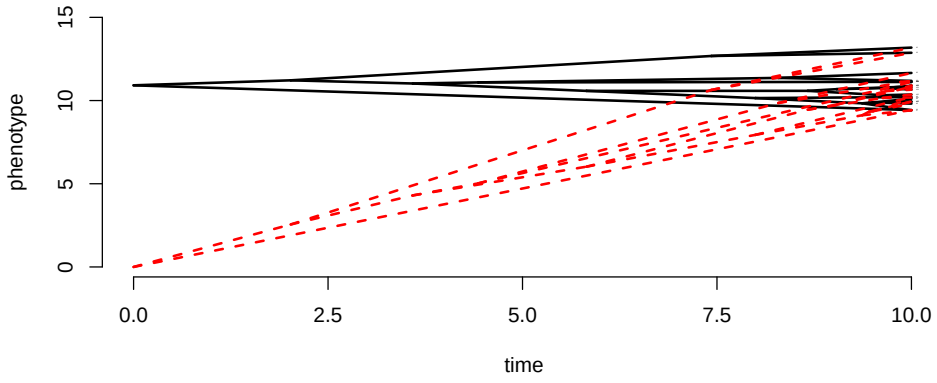


Figure 10: ancestral state estimation under BM (black) when the true model is a trend (red) is very wrong

4 Ornstein-Uhlenbeck processes - modeling adaptation (and maybe constraint)

When we introduced random walks, we talked about tracking the change in the mean trait value of a population through time and extended that idea up to phylogenetic levels. This makes it seem as though a Brownian motion model is a model of evolution by drift. That's true in part – drift will

result in brownian-like data – but there are other ways to get this kind of outcome. For example, if our population was actually tracking a fluctuating optimum (think about environmental change occurring rather randomly and the population's mean chasing it) then the result will also look Brownian. Support for a BM model therefore can't be interpreted as meaning your clade evolved by drift. But likewise, it can't be taken to mean that your clade was definitely chasing an optimum. To do that, we need to use models that explicitly consider how traits move towards optimal values through time and how this affects the associated variance of that expectation.

The models we use are based on the Ornstein–Uhlenbeck process. We can represent this model simply in terms of the expected change of trait over an infinitesimally small time unit

$$\mathbb{E}[\Delta X] = -\alpha(X - \theta), \tag{12}$$

Where θ is the "optimum" trait value and α is a strength of selection. To understand how this works, take a look at Figure 11. The upper purple dot is at 0.5 and the optimum is at 0. So the expected change is -α(0.5 - 0). Clearly this is a negative number and so the trait will move towards the optimum. The lower dot at -0.5 has an expected change of -α(-0.5 - 0), which will be a positive number (the product of two negatives). So this trait should move up towards the optimum. This behavior means that the Ornstein-Uhlenbeck process exhibits a behavior that we call "mean–reverting".

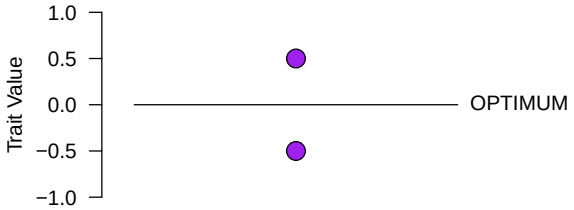


Figure 11: The expected direction of change in the purple traits in this space is towards the optimum. The amount they should move is based on the strength of α

This is just the expected change. If we assume a normally distributed noise associated with that expectation, then we could do a whole bunch of very complicated math to generate an expectation for the mean value and associated variance of a trait evolving under this process at some time $T = t$ given that the process started at time $T = 0$ at trait value X_0.

The expected value is

$$\mathbb{E}(X_t|X_0) = [1 - e^{-\alpha t}]\theta + e^{-\alpha t}X_0. \tag{13}$$

This is a complicated looking expression but it's surprisingly easy to understand. The first term on the right hand size tells us that must multiply the optimum by $[1 - e^{-\alpha t}]$. We know that α is

the strength of the pull back to the optimum and we also know that t is time from the root. So as time increases and / or as α gets bigger, the exponent gets more and more negative. Think of this as a natural-logged number; more negative logs correspond to values closer and closer to zero and e^0 is 1. So when time is small and / or α is small, the number be subtracted from 1 is closer to 1 and the number we multiply the optimum by is closer to zero, while as time increases or as α gets bigger, this number gets closer to zero and so the term on the right becomes equal to θ. Likewise, for the term on the right, as time increases and / or as α gets bigger, the exponentiated number goes closer to zero and we multiply the starting value by increasingly smaller numbers.

What we're doing should be familiar from our discrete random walk - we're just computed a weighted average of the starting value and the optimum, where our weights correspond to product of distance from the starting point and α. In other words, as we get further from the starting point, the weight of the starting value should decrease such that the optimum dominates. Or, if α is large enough, we should lose all influence of the starting point very quickly and go immediately to the optimum.

This process is often challenging to understand just by staring at an equation, so let's simulate it. We'll write a function to track the expected value of an OU process through time and then explore how changing the parameters affects the outcome.

```
> #### a function to generate expected values of an OU process ####
> expected.value.ou <- function(root, theta, alpha, time) {
+                 influence.of.optimum <- (1-exp(-alpha*time)) *theta
+                 influence.of.root <- exp(-alpha*time) * root
+                 influence.of.optimum - influence.of.root
+ }
> ###################################################################
>                 time.seq <- seq(0,10,0.1)
>                 a <- 1
>                 X0 <- 0
>                 optimum <- 1
>                 Eou<-expected.value.ou(X0, optimum, a, time.seq)
>                 plot(time.seq, Eou, type="b", ylim=c(0,1))
>
```

Try running the code above with the values specified. What do you see when you plot the results? Now try setting a (α) equal to 0.1 - what happens now? What if you set a = 0.01? or a = 10?

What you should find is that as α gets bigger, the rate at which the trait approaches the optimum increases. For $\alpha = 0.01$, the expected value at time 10 is nowhere near the optimum, but for $\alpha = 1$, the trait makes it and does so relatively quickly. For $\alpha = 10$, we get there even more quickly – almost instantly, infact.

Can we relate α's value to how quickly we approach the optimum? Indeed we can. Remember that the influence of the optimum (it's weight in the weighted average) increases as an exponential function of α and time. In the Early Burst model, we noted that rates declined as an exponential function of time and a parameter r and so we calculated the rate half–life, the time until the rate

fell to half its initial value as $\frac{ln2}{r}$. We can do the same thing in the OU model. Here, $\frac{ln2}{\alpha}$ is called the **phylogenetic half–life** and is the time until the trait reaches half way to the optimum. $\ln 2 \approx 0.69$, so values of α that are greater than 1 will give rapid approaches to the optimum, while values less than 1 will require longer time periods to get there.

Try setting $\alpha = 0$. What do you find now? The expected value should not change. If you plug 0 into the expression for the expected value, you'll see why - there is no pull to the optimum. In fact, an OU process with $\alpha = 0$ is equivalent to Brownian motion. We'll see why that's true next.

We need to consider the variance of a random draw from an OU process. It's defined as follows

$$Var(X_t|X_0) = \frac{\sigma^2}{2\alpha}[1 - e^{-2\alpha t}], \tag{14}$$

where the parameters are as before. The $1 - e^{-2\alpha t}$ term is familiar from the expected value equation; it tells us that whatever that term in front of it is, it's going to be small early on and become more important later. This is because when t and / or α are large, that term will drop out. The term in front is the rate divided by two times the strength of selection. This tells us that diffusion, which increases variance, is being constrained by the strength of selection; for a given rate of evolution, the actual variance is going to be limited by the strength of selection.

There's an important point to note here. In Brownian motion, the variance increases with time - $\mathbb{E}[X] = \sigma^2 t$. In the OU process, time only enters the variance expression in the exponent term and, as times goes on, it ultimately disappears. So after some period of time, the variance becomes constant. Think about this - once we reach that point, it doesn't matter whether the clade has been evolving for 10, 100, or a million times longer - the expected variance will be the same. For this reason, we call the term $\frac{\sigma^2}{2\alpha}$ the **equilibrium variance**.

We can prove this to ourselves by writing a function to calculate the expected variance through time and playing around with the values:

```
> variance.of.ou <- function(sigmasq, alpha, time) {
+                 (sigmasq / (2*alpha)) * (1-exp(-2*alpha*time))
+                 }
> a=2
> ssq=1
> time.seq <- seq(0,10,0.1)
> plot(time.seq, variance.of.ou(ssq, a, time.seq))
```

Try playing around with different values of α and σ^2 to see what happens. In particular see what happens when you make α *really, really, really* small (NOTE: you can't make α zero exactly because you'll get a divide by zero error, but you can make it [e.g.,] 0.0000001). What you should find is that

1. Increasing σ^2 increases the equilibrium variance
2. Increasing α increases the rate at which you get to the equilibrium variance, and

3. making α really really really small results in a time-increasing variance with no equilibrium point in sight.

Combining this with what we understand about how the expected value changes through time, it leads us to a verbal understanding of how the OU process works. We can see that the OU process operates as a kind of random walk but, unlike Brownian motion, it's an equilibrium process. If run long enough, or with strong enough attraction to where it's going, an OU random walk will converge on it's optimum and the wiggle around that optimum will reflect a balance of the rate at which it wants to walk away and the strength of selection pulling it back.

4.1 OU models on trees

Now we understand how the process works, we are nearly ready to figure out how to fit an OU model to our comparative data. But before we can do this, we need to get the expected covariance structure for the multivariate normal distribution. From the Early Burst example, you might remember that we need to integrate the function with respect to time to get the values for each element in the covariance matrix. We won't do that [it's gnarly] but we will take advantage of the fact that better math people than us have already done it. The elements of the OU model covariance matrix $\mathbf{V}$ are given by

$$\mathbf{V}_{ij} = \frac{\sigma^2}{2\alpha}[e^{-\alpha t_{ij}}][1 - e^{-2\alpha \mathbf{C}_{ij}}] \tag{15}$$

There's a lot going on here but let's unpack it. $\frac{\sigma^2}{2\alpha}$ we already know – it's the equilibrium variance. $1 - e^{-2\alpha \mathbf{C}_{ij}}$ is also familiar. It's scalar that regulates how close to the equilibrium variance we are, except that we've swapped the i, jth element of the phylogenetic covariance matrix for t. The middle expression $e^{-\alpha t_{ij}}$ is a new one and it introduces a new term t_{ij}. This is the total path length separating species i and j; in other words, the total amount of time they've been evolving independently since they split. We're multiplying this by $-\alpha$ and then taking the exponent, so if t_{ij} is small (the taxa split recently) then this number is small and it's exponential is close to 1. But it it's large (the species split a while ago) it makes the exponential close to zero.

This is the final important thing to realize about the OU process on a tree. We know that the longer the process goes on, we should converge on the equilibrium variance. So if a path length from the root to the common ancestor of a pair of taxa is really long, we should expect their covariance to converge on the equilibrium variance. But this last term adds a wrinkle. It tells us that as soon as a pair of species split, they start to erase any and all covariance that they previously accrued. So if a pair of species share 100 million years of evolutionary history but evolve independently for 20 million years after splitting, it's possible that they share absolutely no evolutionary covariance according to the OU process. Does that sound crazy? It shouldn't because we know the process is mean reverting. We can imagine those walks in trait space criss-crossing each other over and over again until we have no idea that they share a common ancestor based on their trait values. If this same thing happens for all tips in the tree, then the resulting covariance structure will essentially be a matrix with zeros on all the off-diagonal elements and the equilibrium variance on the diagonals. In other words, a star phylogeny. For this reason, the OU process is often refered to as a "low phylogenetic signal" model.

4.2 flavors of OU model for phylogenetic data

The most basic implementation of the OU model assumes that the process starts at its optimal value (i.e., the root state and optimum are the same). This is sometimes (not enough, in my opinion) referred to as a **single stationary peak** process. This model can be fitted to comparative data using functions in a number of packages. We'll try it out using `fitContinuous` in the `geiger` package. We'll use our whale dataset again.

```
> ssp.whales <- fitContinuous(phy, lnlength, model="OU")
>
> ## type ssp.whales$opt to see the parameter estimates
```

What did you find? In this case the Maximum Likelihood Estimate of α is very small. It seems almost pointless to do so, but we can calculate the phylogenetic half–life

```
> log(2) / ssp.whales$opt$alpha
```

```
[1] 3.45917e+15
```

which is 3.5 x 10^{15} million years, or a very long time! Clearly we do not even need to look at the AIC score for this model. The parameter estimates tell us that there is considerable phylogenetic signal in this dataset and that these data are not consistent with an SSP process.

Another way to use this process is to ask whether clades or lineages with different ecologies / geographic distributions / life histories etc might evolve towards different optimal trait values, possibly with different rates and with pulls of different strength. These models come naturally from the ideas we've already dicussed but require that we can map the "regimes" of interest onto our phylogeny first so that we can obtain appropriate expectations and covariance structures. We've already seen how this can be done using `make.simmap` to generate a map for testing for different rates using `brownie.lite`. We can use the same idea to fit multi-optimum OU models using functions in the OUwie package. Let's try this with the whale data.

OUwie requires that we put our data into a dataframe with species names in the first column, the trait of interest in the second and the regime in the third. Let's load OUwie and make the data

```
> library(OUwie)
> ouwie.dat <- data.frame(species=phy$tip.label, regime=clade, trait = lnlength)
> head(ouwie.dat)
```

```
                                                 species  regime     trait
Kogia_breviceps_KBU72040__         Kogia_breviceps_KBU72040__ toothed 1.2237754
Kogia_simus_AF304072                     Kogia_simus_AF304072 toothed 0.8878913
Physeter_catodon_X75589                Physeter_catodon_X75589 toothed 2.4006188
Platanista_gangetica_AF304070_ Platanista_gangetica_AF304070_ toothed 0.9242589
Platanista_minor_X92543                Platanista_minor_X92543 toothed 0.9242589
Tasmacetus_shepherdi_AF334484   Tasmacetus_shepherdi_AF334484 toothed 1.8718022
```

Look's good. Ok now we can fit the model. OUwie gives us a lot of options. We're going to use the option `model = "OUM"`, which fits a model in which the optima differ between our two regimes but the rate parameter (V) and α (A) are the same

```
> OUwie(phy = whale.mapped, data=ouwie.dat, model="OUM", simmap.tree=TRUE)

Initializing...
Finished. Begin thorough search...
Finished. Summarizing results.

Fit
       lnL     AIC     AICc model ntax
 -20.69565 49.3913 49.96273   OUM   75


Rates
              baleen     toothed
alpha    0.003753347 0.003753347
sigma.sq 0.011708577 0.011708577

Optima
            baleen   toothed
estimate 2.5387427 1.3137669
se       0.4161751 0.2827318

Arrived at a reliable solution
```

The output gives us a bunch of information. Perhaps first and most obviously, we are interested in the MLE's of the trait optima. These are quite different; 2.5 log meters for baleen whales and 1.3 log meters for toothed whales. To determine whether this model fits our data better than the other models we have already fitted, we could use AIC weights and see how much of the relative model support each gets. In this case, the difference is tiny but let's imagine that this model is actually the best by a long way. Would we be justified in claiming evidence for strong adaptation?

Well, maybe we should be cautious. Often times [*"pet peeve alert"*] people are so excited to find differences in their inferred optima that they forget the OU process also makes predictions about quickly those optima should be approached and how variance/covariances should differ from Brownian motion models. We know that the α parameter determines the strength of these pulls, and we also know we can gain an intuituve understanding of its meaning by transforming it to a phylogenetic half-life and comparing this value to the age of our clade.

Our MLE for α here is 0.0037 (lets be generous and round up to 0.004), so we can get the half–life by

```
> log(2) / 0.004
```

```
[1] 173.2868
```

Whoa! The crown age of cetaceans is 36 myr, but the time to get half way to a new optimum is 173 myr! What are we to do with this? Well, maybe we don't care. Maybe we're happy with the idea that the strength of the pull to the optimum is weak and the equilibrium variance is large. We might imagine a scenario in which there is some "adaptive" peak but it's not really that peaky; more of a table mountain than an Everest.

An alternative explanation - perhaps not that different – is that whale body size evolved by a random walk but that there was a rapid pulse of evolution along the branch leading to baleen whales that pushed their mean size to larger. It might have been a trend, it might have been a short–lived increase in rate, it might have been selection; we can't tell. But the outcome is Brownian-like rather than OU-like. We could fit this model by assuming Brownian motion rather than an OU process but with different phylogenetic means for our two regimes.

```
> OUwie(phy = whale.mapped, data=ouwie.dat, model="BMS", simmap.tree=TRUE)

Initializing...
Finished. Begin thorough search...
Finished. Summarizing results.

Fit
       lnL      AIC     AICc model ntax
 -20.72677 49.45354 50.02496   BMS   75

Rates
             baleen    toothed
alpha            NA         NA
sigma.sq 0.01138535 0.01119089

Optima
            baleen   toothed
estimate 2.5364766 1.3170172
se       0.4494891 0.3059534

Arrived at a reliable solution
```

What's different here? If you look at the output, you'll see that the parameter estimates all look similar except for one thing – there's no α. What we've just showed is that we dont really need to invoke an OU process to explain these dramatic size differences. If we just assume that rates along the branch leading to baleen whales pushed them to a mean size and that they then evolved from that starting value then we can equally well explain these data. This is a neat way to check whether an OU process really is a good explanation for your data.

There are all kinds of ways to do OU model fitting – more than I can cover in this document. But the following info might be helpful to point you in the right direction:

1. Single stationary peak models can be fitted using `geiger`

2. If you know what your regimes are and how they map to the tree, multi-peak models can be fitted in a bunch of different packages. `OUwie` is one of the most flexible because it can accomodate different rates and selection parameters, in addition to optima. The `mvMORPH` package can also do this and can accomodate multivariate data (see next section).

3. If you don't have regimes but want to identify them from the data, there are still options. The `bayou` package by Josef Uyeda uses a reversible jump MCMC to sample optimal trait values and OU model parameters, as well as regime configurations. The `l1ou` package by Mohammed Khabbazian and Cécile Ané uses a LASSO regression approach to obtain maximum likelihood estimates of the parameters while identify shifts.

4. `slouch` (stochastic linear OU comparative hypotheses) can do single and multi-regime models but adds a further details by allowing for interactions among covariates. For example, we might ask whether the evolution of tooth crown height favors different optima for ruminants with different diet and habitat preferences, rather than one or the other (Tolijagić et al. 2018).

5 Multivariate traits

Thus far we've only considered how to model the evolution of a single trait. Often times you are actually interested in modeling multiple traits at the same time. Intuition might suggest that you could assume all evolve at the same rate and find the estimates of model parameters that maximize the product of the likelihoods of all traits. That'd be good logic if your traits are independent of one another, but that typically doesnt happen. And for that reason, we need to account for covariances in traits as well as covariances in trees.

Let's imagine we have n taxa and for each of those we've measured r traits. The likelihood of an n x r matrix of multivariate data $\mathbf{X}$ is given by

$$\mathcal{L}(\mathbf{X}|\mathbf{R},\boldsymbol{\mu}) = \frac{e^{-0.5[\mathbf{X}-\boldsymbol{\mu}]^T\boldsymbol{\Sigma}^{-1}[\mathbf{X}-\boldsymbol{\mu}]}}{\sqrt{(2\pi^{nr})det(\boldsymbol{\Sigma})}} \tag{16}$$

This looks exactly the same as the likelihood for univariate data, and it is except for one small difference. Instead of calculating the likelihood of the data given a rate σ^2 and a starting value μ, we're now aksing for the likelihood given the starting value and something denoted $\mathbf{R}$. In linear algebra, a boldened upper case letter indicates a matrix, so instead of a rate, we're now estimating a matrix of some kind. In fact, this matrix does give us rates for each trait, which occur along the diagonal elements. But the off-diagonal elements give us the trait co–variances, or how change in the ith trait affects change in the jth. Let's look at an example.

```
> library(geiger)
> data("geospiza")
> td <- treedata(geospiza$geospiza.tree, geospiza$geospiza.data, sort=T)
> (R<-ratematrix(td$phy, td$data))
```

```
             wingL    tarsusL   culmenL     beakD    gonysW
wingL   0.07642449 0.05753586 0.1662145 0.2322144 0.1936422
tarsusL 0.05753586 0.05345056 0.1211433 0.1652403 0.1418082
culmenL 0.16621446 0.12114329 0.4455064 0.5097081 0.4265022
beakD   0.23221436 0.16524030 0.5097081 0.8117959 0.6699364
gonysW  0.19364216 0.14180823 0.4265022 0.6699364 0.5623170
```

The data here are measurements of beaks and limbs from Darwin's finch species. The function ratematrix allows us to directly estimate the evolutionary rates and the evolutionary covariances among traits. What you see by looking at this is that all the elements are positive. That makes sense for rates (rates have to be positive), but what do the positive covariances imply?

We can visualize this matrix to better understand what it is telling us. We'll need to convert it into a correlation matrix and then use a function from the `ellipse` package

```
> library(ellipse)
> cor <- cov2cor(R)
> colors <- rev(c("#A50F15","#DE2D26","#FB6A4A","#FCAE91","#FEE5D9"
+                 ,"white","#EFF3FF","#BDD7E7","#6BAED6","#3182BD","#08519C"))
>  plotcorr(cor, type="full", diag=T, cex.lab=1, col=colors[cor*10])
```

What this tells us is that all the evolutionary covariances between traits are positive (as one gets bigger, so does the other) but that the strength of these covariances differ among pairs of traits (different intensities of red and differently shaped ellipses). This makes sense - we estimated this from raw trait values and we would expect that as a bird gets bigger, it's beak, wings and feet should get bigger too. We could get a better handle on what is going on by doing some kind of size correction. Figure 13 shows that actually the traits change in slightly less predictable ways when allometry is accounted for, with some positively correlated changes (red) and some negative (blue).

Now we know what a rate matrix is. How does this factor in to the calculation of the likelihood. Well, when dealing with univariate traits, the evolutionary rate was just a scalar of the phylogenetic covariance matrix; we multiplied each element of the matrix by σ^2 and that gave us a Brownian motion matrix. But here our rates comes in matrix form, which means we have to multiply the matrix $\mathbf{R}$ by the matrix $\mathbf{C}$. Furthermore, these matrices are of different sizes ($\mathbf{R}$ is r x r and $\mathbf{C}$ is n x n). This means we have to calculate something called a kronecker product of the two.

$$\mathbf{V} = \mathbf{R} \otimes \mathbf{C} \tag{17}$$

The result is a very big matrix! We need to multiply each element of $\mathbf{R}$ by every element of $\mathbf{C}$, resulting in a matrix $\mathbf{V}$ that is rn x rn in size. That might not be much of an issue for the Darwin's

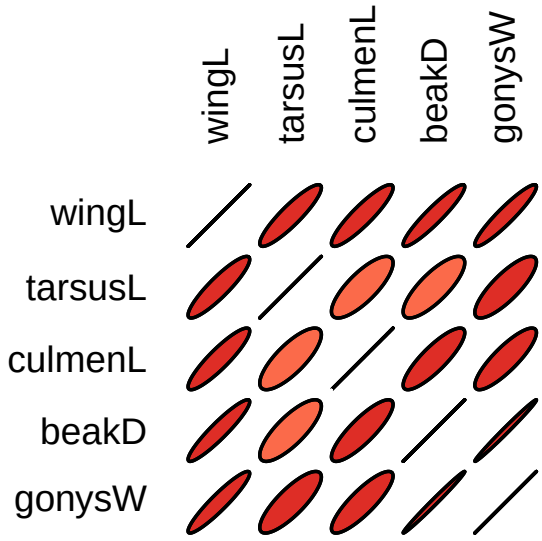


Figure 12: The evolutionary correlation matrix for the *Geospiza* data.

finch dataset but these numbers get very big very fast. As a result calculating the likelihood can get really computationally demanding. Nonetheless a number of packages implement methods to fit multivariate models and there are number of short cuts that can be used to get likelihoods quickly. Just be aware, if you have 150 3D landmarks digitized for 350 species and want to fit multivariate models to this dataset, you're going to be trying to work with 157500 x 157500 matrices and that's going to be tough.

Multivariate models (BM with or with rate shifts, EB, OU etc) etc can be fitted using functions in the `mvMORPH` package. Importantly all these models assume a common evolutionary rate matrix is shared among taxa. The `ratematrix` package by Daniel Caetano and Luke Harmon allows for testing the hypothesis that the evolutionary variances AND covariances differ among predefined lineages. So if, for example, you think that a switch in diet results in a change in morphological integration, you can test this idea by using mcmc to sample a posterior distribution of matrices. Multivariate trait modeling is an active area of development and new tools are likely to emerge quickly.

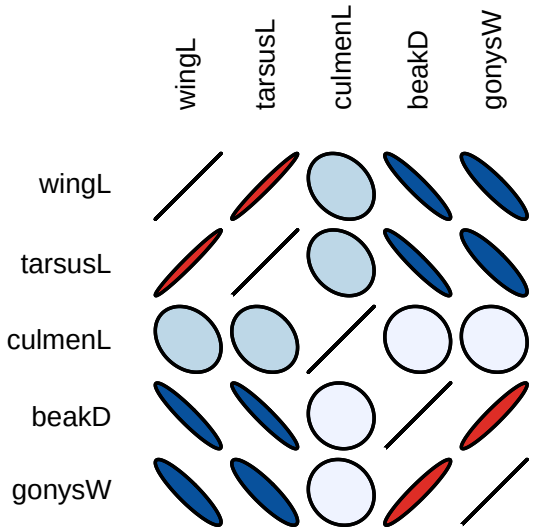


Figure 13: The evolutionary correlation matrix for the size corrected *Geospiza* data.