Introduction to character evolution

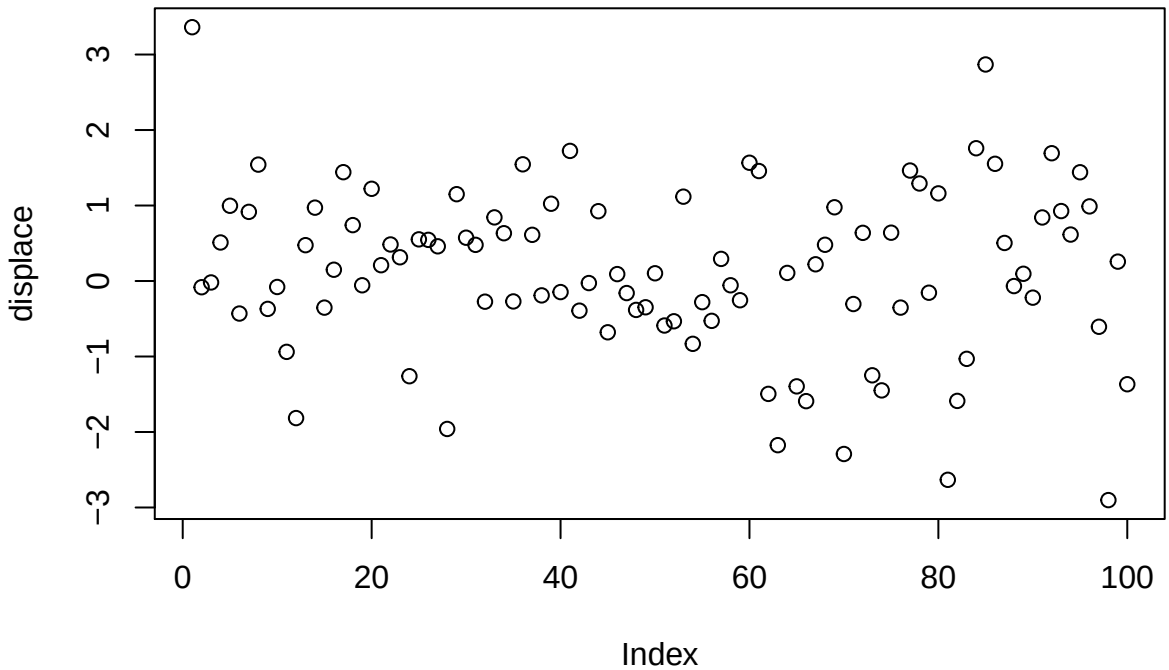
In this exercise you will learn the most common models of morphological evolution and gain experience with their dynamics.These exercises are based on the 2014 Paleobiology and Phylogeny for Macroevolution NESCent summer course.

Brownian motion simulations

Brownian motion describes a process where changes in a trait are governed by a "random walk" meaning that the value of a trait can increase or decrease at any point in time. Trait changes are not directed so a change from a previous instant in time does not predict the value of new change. We can model the expected dispalcements of a trait under Brownian motion as a series of draws from a normal distribution with a mean of 0 and a variance proprtional to time. To get a sense for this draw 100 "evolutionary steps" for a continuous trait using rnorm.

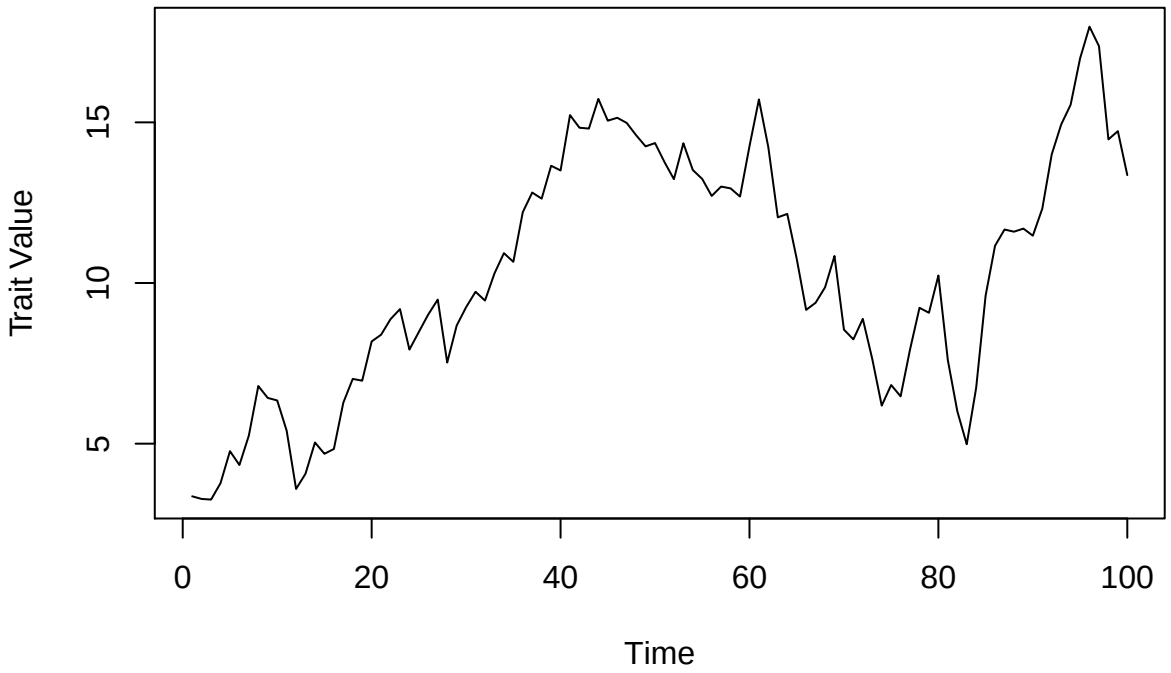
```
par(mfrow = c(1,1))
displace<-rnorm(100)
plot(displace)
library(geiger)
```

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



If we sum these evolutionary steps we can generate a trajectory of the trait through time

```
x<-cumsum(displace)
plot(x, type="l", xlab="Time", ylab="Trait Value")
```

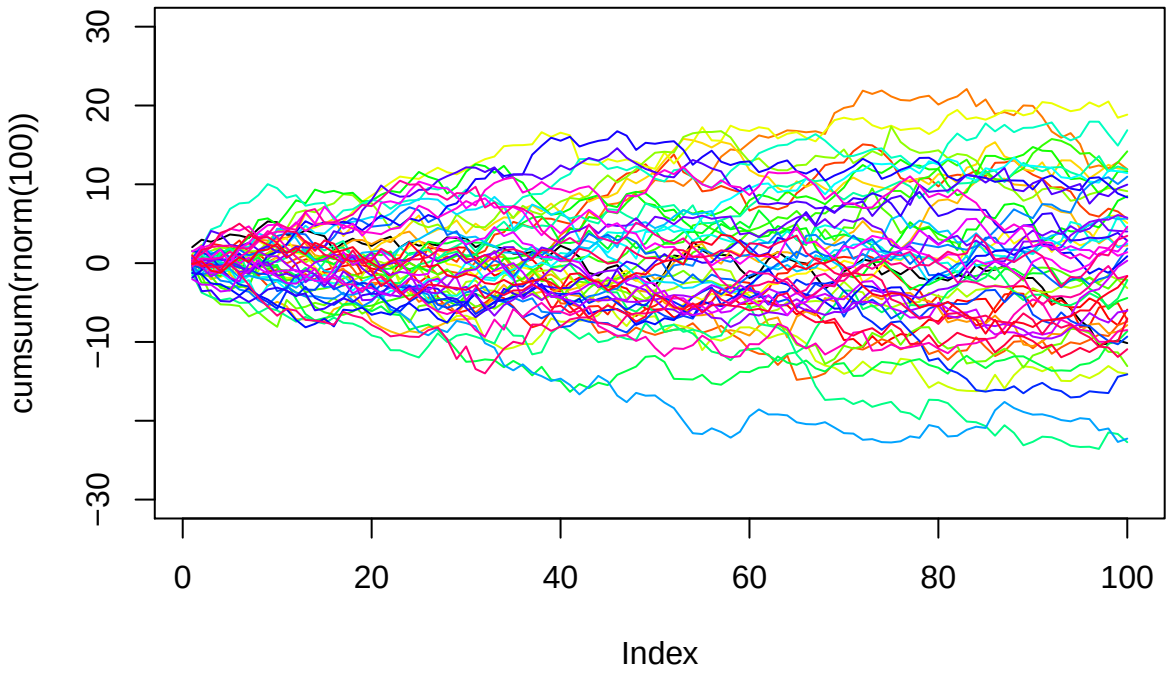


Notice that the value of the trait at time 100 has wandered away from the starting value under this process.

Exercise 1: What value do you expect this trait to have after 100 time units? 1000 time units? Why?

Now let's imagine we have 50 independent lineages that all start out with the same value of a continuous trait. If we let those 50 lineages evolve under Brownian motion (BM) what is the expected pattern of their trait values after 100 time units?

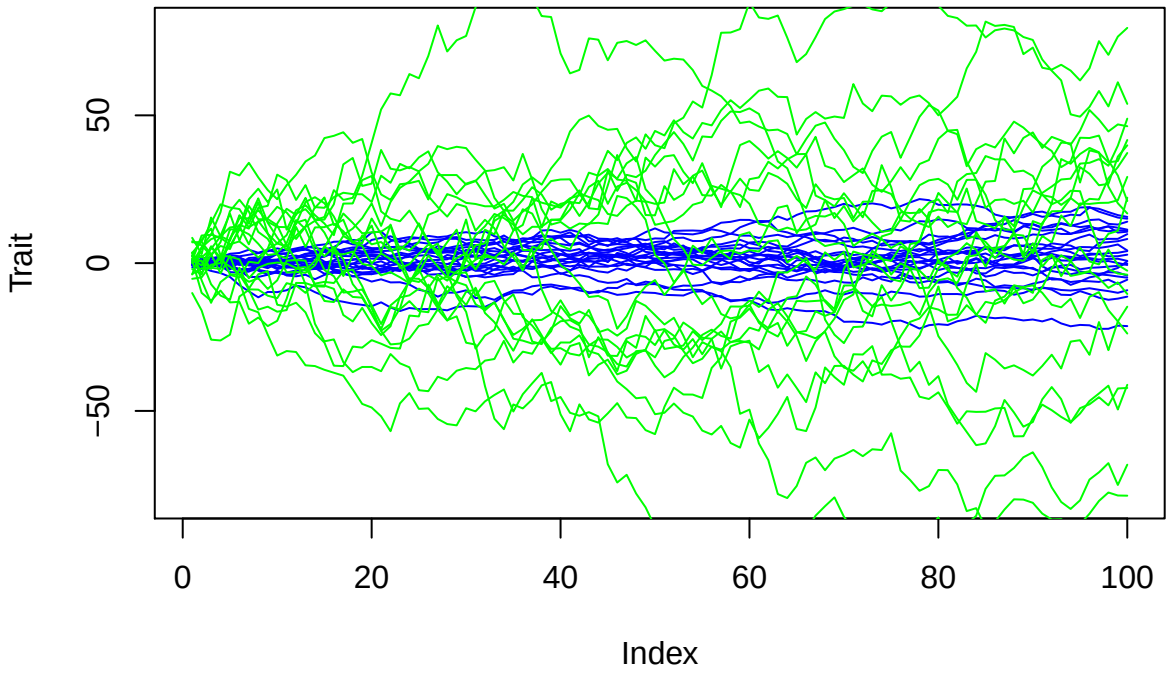
```
cols<-rainbow(50) # put 50 samples from the rainbow pallette into a new vector called cols
plot(cumsum(rnorm(100)), type="l", ylim=c(-30,30))
for(i in 1:length(cols)) lines(cumsum(rnorm(100)), col=cols[i]) # simple for loop for each of the 50 co
```



Exercise 2: Write an R block that plots Brownian evolution for 50 independent lineages for 50 time units side by side with BM for 100 time units. What is the effect of time on the diversity in trait values you see in those simulations? How does time affect diversity under BM?

You have just explored the effect of time on BM. The second controlling factor of Brownian evolution is the rate. Brownian rates describe the size of "steps" taken at unit of time. We can describe step size as the standard deviation of the draws from a normal distribtion. In the simulations above using rnorm the default sd is 1. let's try evolving 20 lineages for 100 time units with differing rates by changing the sd parameter of rnorm.

```
plot(cumsum(rnorm(100)), type="l", ylim=c(-80,80), ylab="Trait", col="blue")
for(i in 1:20) lines(cumsum(rnorm(100)), col="blue")
for(i in 1:20) lines(cumsum(rnorm(100, sd=5)), col="green")
```

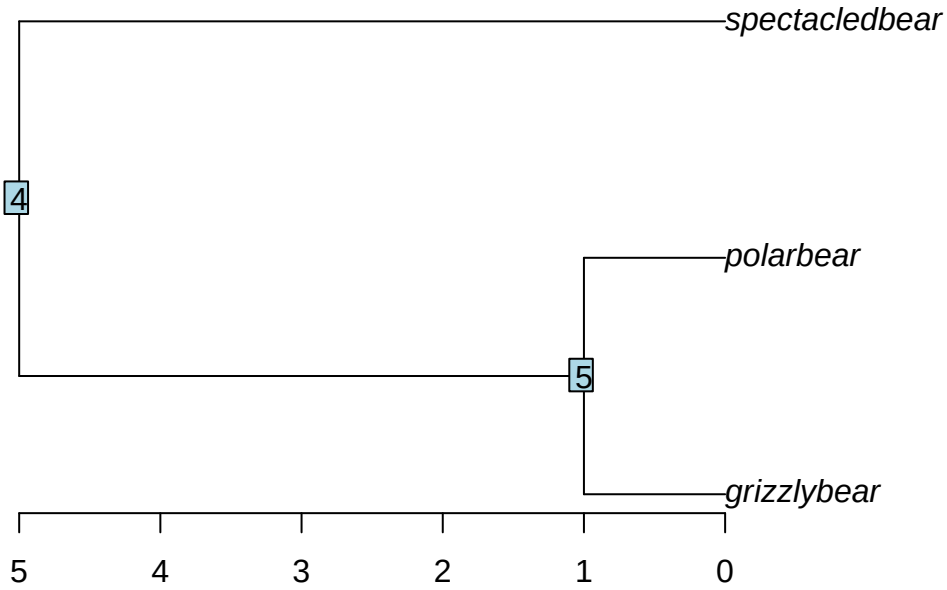


note that the Brownian rate parameter is typically considered as sigma^2 (the variance or square of the standard deviation) but that most simulation functions take the standard deviation (the square root of the variance) as an argument.

Exercise 3: What is the rate parameter of Brownian motion? How does trait variance in a clade evolving under BM scale with rate?

So far we have simulated evolution of traits as if lineages are independent. However we know that species share a common evolutionary history. How might this shared history affect the evolution of continuous traits? let's look at a simple tree.

```
tree<-read.tree(text="((grizzlybear:1, polarbear:1):4, spectacledbear:5);")
plot(tree)
nodelabels()
axisPhylo()
```



You can see that polar bears and grizzlies diverged from on another 1 MY ago while spectacled bears have evolved independently from that clade for 5 million years. Think about what this tree implies about the evolution of a continuous trait. Node 4, the root of the tree, represents the common ancestor of all three bears. Let's imagine we are interested in simulating the evolution of body size from this common ancestor. If the ancestral value was 250 kg we could simulate one outcome of BM from node 4 to spectacled bear by drawing one "walk" away with a step size that is determined by both the sigma^2 (the Brownian rate) and time. To get the correct standard deviation for multiple time steps at once we simple take the square root of the product of time X sigma^2. See the example below.

```
rootMass <- 250 # size of ancestor
sigmasq = 2.5 # Brownian rate
time = 5 # 5 million years of independent evolution from the root
sd <- sqrt(time * sigmasq) # Brownian evolution is proportional to rate X time)
specbearDeltaMass <- rnorm(1, mean = 0, sd = sd)
specbearMass <- rootMass + specbearDeltaMass
specbearMass
```

```
## [1] 248.6755
```

```
##another way to do this using the tree structure itself to supply the time argument
specbearDeltaMass<-rnorm(1, mean=0, sd=sqrt(sigmasq*tree$edge.length[1])) # in R, tree tip labels are n
specbearMass <- rootMass + specbearDeltaMass
specbearMass
```

```
## [1] 250.2975
```

Now think about how you would approach size estimates for polar bear and grizzly bear. The phylogeny tells you that 1) the ancestral mass for ALL bears was 250 kg 2) the lineage leading to the clade polar bear + grizzly evolved independently for 4 MY and 3) that lineage splits (at node 5) and then polar bears and grizzlies evolved independently from each other for 1 MY.

Exercise 4. Write a block of R code that simulate BM evolution for this three taxon tree. You can do this as a series of steps or write a general function.

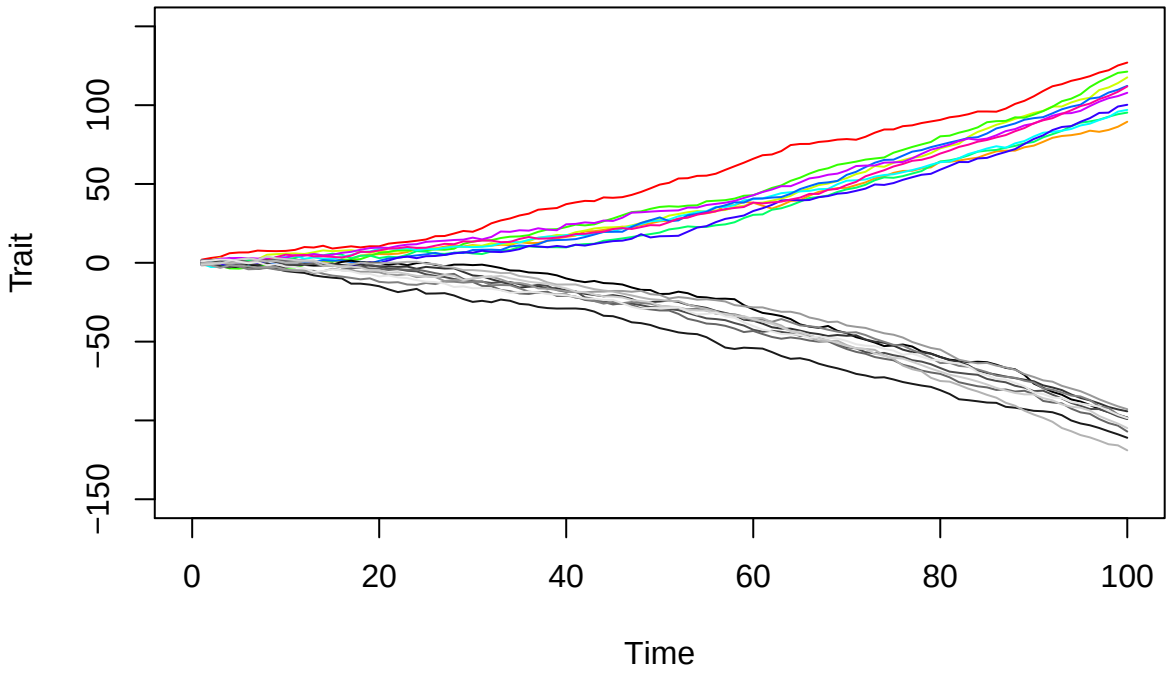
Trends

What if we think that there has been a trend in the evolution of a trait? One way we could simulate this process is by saying that the mean of the normal distrbution is changing with time. We will do this by adding a new parameter to our model, mu. mu is going to shift the mean of the random normal distribution as a function of time (see the function below)

```
bmtrend<-function(time, mu, stdev){ # a function that requires mu which is the trend parameter showing
  displace<-vector(,length=time) # set up an empty vector of the same length as time
    for(i in 1:time){
            displace[i]<-rnorm(1, mean=mu*i, sd=stdev) # sample from rnorm with a mean with a mean of m
            }
    traj<-cumsum(displace) # calculates the trajectory over time
    return(traj)
}
```

Now we are going to run this bmtrend function 10 times using the replicate function, once with a mu of 0.02 and the other with mu=-0.02

```
par(mfrow = c(1,1))
bmtrend0.02<-replicate(10, bmtrend(100, 0.02, 1))
bmtrendneg0.02<-replicate(10, bmtrend(100, -0.02, 1))

cols1<-rainbow(10) # sample 10 colours from the rainbow palette
cols2<-grey(0:10/10) # sample 10 colours from the grey/gray level specification - it has a different sy

plot(bmtrend0.02[,1], xlim=c(0,100), ylim=c(-150, 150), type="l", col=cols1[1], xlab="Time", ylab="Trai
for(i in 2:10) lines(bmtrend0.02[,i], col=cols1[i])
for(i in 1:10) lines(bmtrendneg0.02[,i], col=cols2[i])
```



Exercise 5. Simulate 15 runs of a trend process for two groups of species: one with a mu of 0.5 and and the other with a mu of -0.5 for 100 MY. If you were to measure trait diversity in these groups at 100 only (in other words, if you did not knowm the true trait history) is there any information in those trait values that could distinguish evolution under a trend from regular Brownian evolution? Can you think of any additional information that would be helpful in distinguishing these models?

How to simulate Ornstein-Uhlenbeck

The final model we will consider is the Ornstein-Uhlenbeck or OU model. This model contains BM but adds a constraint parameter that "pulls" the trait towards a peak value with increasing strength as that trait wanders away. There are four parameters to consider for OU: θ (the optimal trait value), α (the pull towards the optima), σ (sigmaˆ2 the stochastic (Brownian) motion parameter) and the starting value for the trait(x_0). OU models a change in trait X over an increment in time t as ##

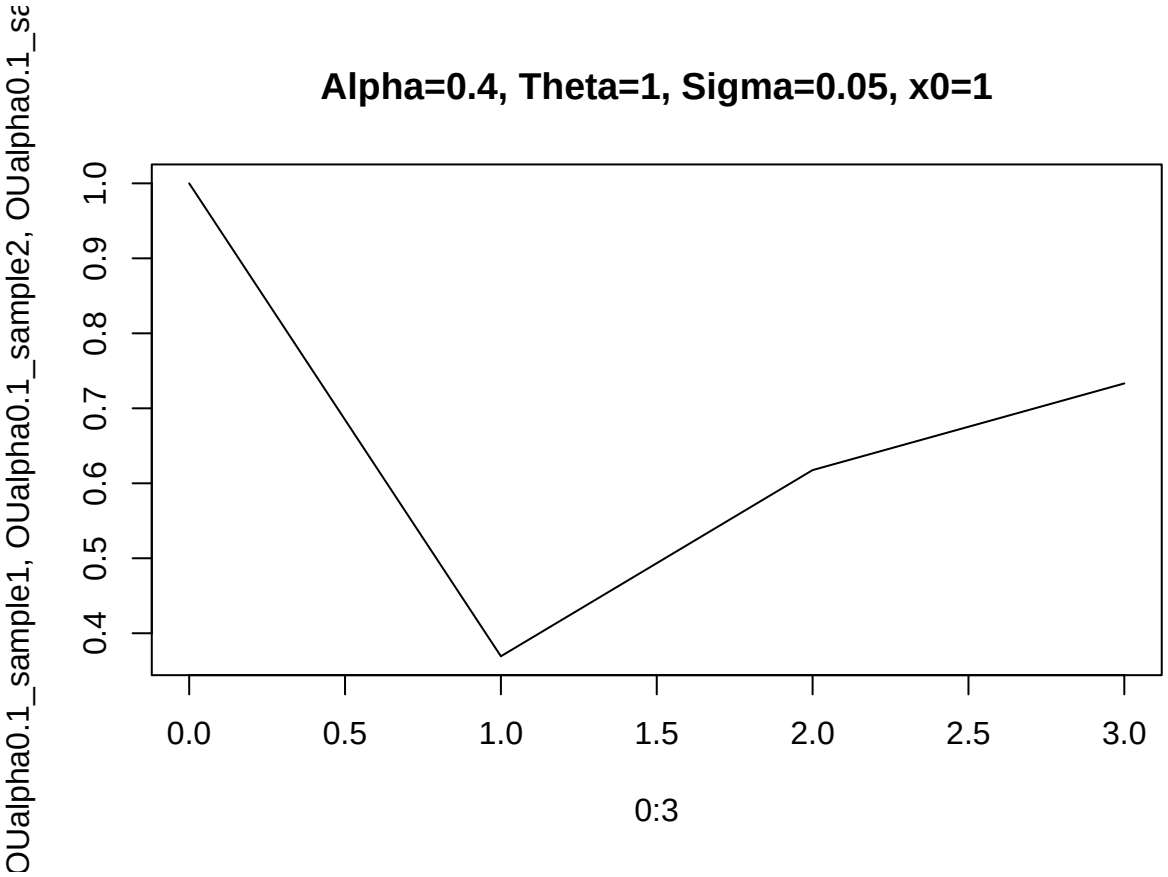
$$dX_t = \alpha[\theta - X_t]dt + \sigma dB_t$$

We will begin our invesitgation of OU by considering all lineages to be independent. Let's assume that the ancestral value of the trait is 0, the optimum is 1, the pull towards the optima is 0.4 and sigmaˆ2=0.05

```
alpha<-0.4
theta<-1
sigma<-0.05
x0<-0
```

We can draw OU samples in a (somewhat painful) way like this:

```
OUalpha0.1_sample1<- x0 + alpha*(theta-x0)+sigma*(rnorm(1,mean=0))

# the next sample would be

OUalpha0.1_sample2<- OUalpha0.1_sample1 + alpha*(theta-OUalpha0.1_sample1)+sigma*(rnorm(1,mean=0))

OUalpha0.1_sample3<- OUalpha0.1_sample2 + alpha*(theta-OUalpha0.1_sample2)+sigma*(rnorm(1,mean=0))

plot(x=0:3, y=c(1,OUalpha0.1_sample1, OUalpha0.1_sample2, OUalpha0.1_sample3), type="l", main="Alpha=0.
```



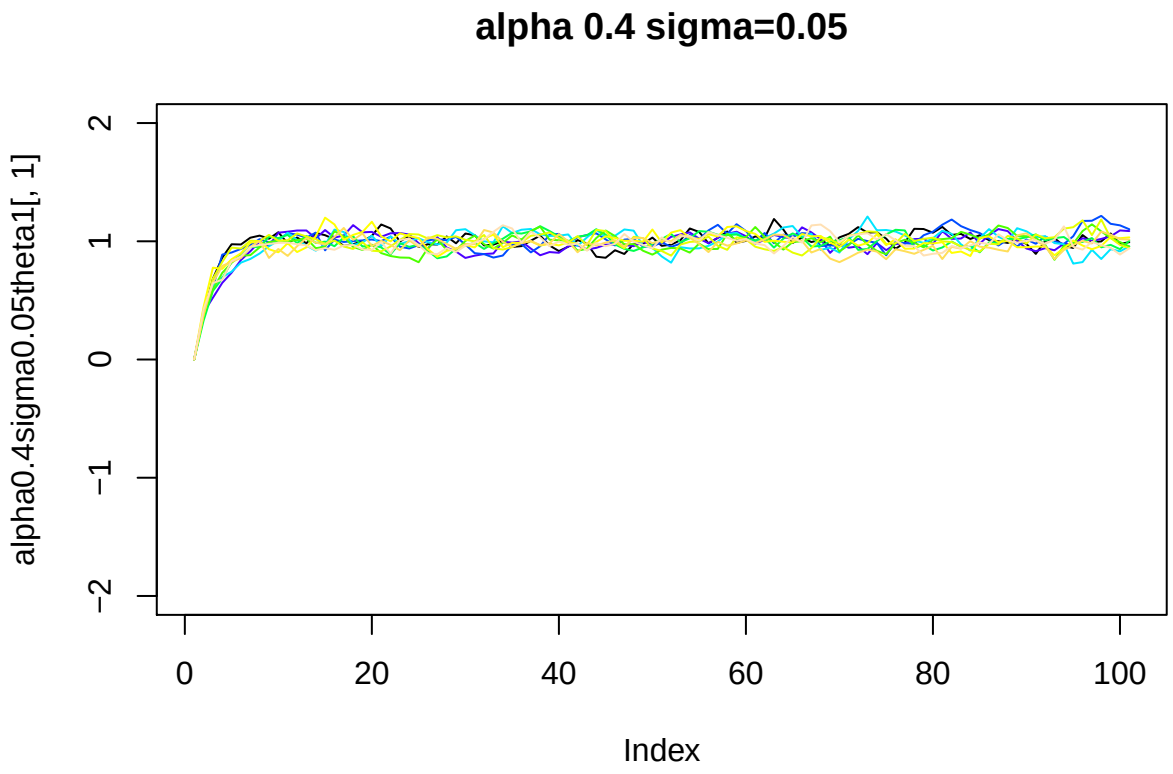
Here is a function to make OU evolution simpler.

```
ornstein_uhlenbecksim <- function(n,theta,alpha,sigma2,x0){# n is the number of time units to simulate
  dw  <- rnorm(n, 0)
    x <- c(x0)
    for (i in 2:(n+1)) {
        x[i]  <-  x[i-1] + alpha*(theta-x[i-1]) + sigma2*dw[i-1]
    }
    return(x);
 }
```

OK, let's look at 10 realizations of the OU process for 100 MY years

```
par(mfrow = c(1,1))
alpha<-0.4
theta<-1
sigma<-0.05
x0<-0
time = 100

alpha0.4sigma0.05theta1<-replicate(10, ornstein_uhlenbecksim(time, theta, alpha, sigma, x0)) # replicat


colr<-topo.colors(9)
```

```
plot(alpha0.4sigma0.05theta1[,1], type="l", main="alpha 0.4 sigma=0.05", ylim=c(-2, 2))
for(i in 2:ncol(alpha0.4sigma0.05theta1)) lines(alpha0.4sigma0.05theta1[,i], type="l",col=colr[i-1] )
```

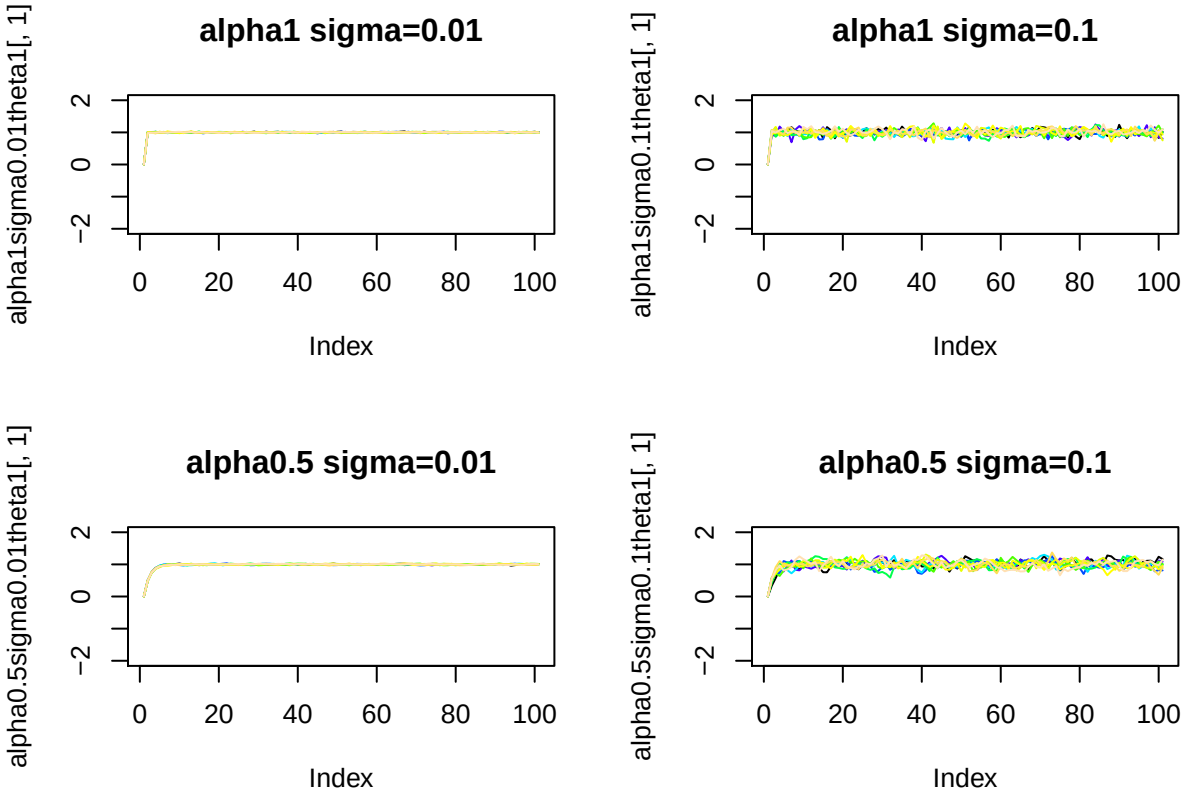


Exercise 6. Look at the plot you have just created. Describe the evolutionary trajectory of the trait under OU. Make a second plot of BM with the same starting value, sigma, and time and show it side by side with your OU simulation. Compare the evolutionary trajectories and ending diversities of BM and OU. In what ways is OU different from BM?

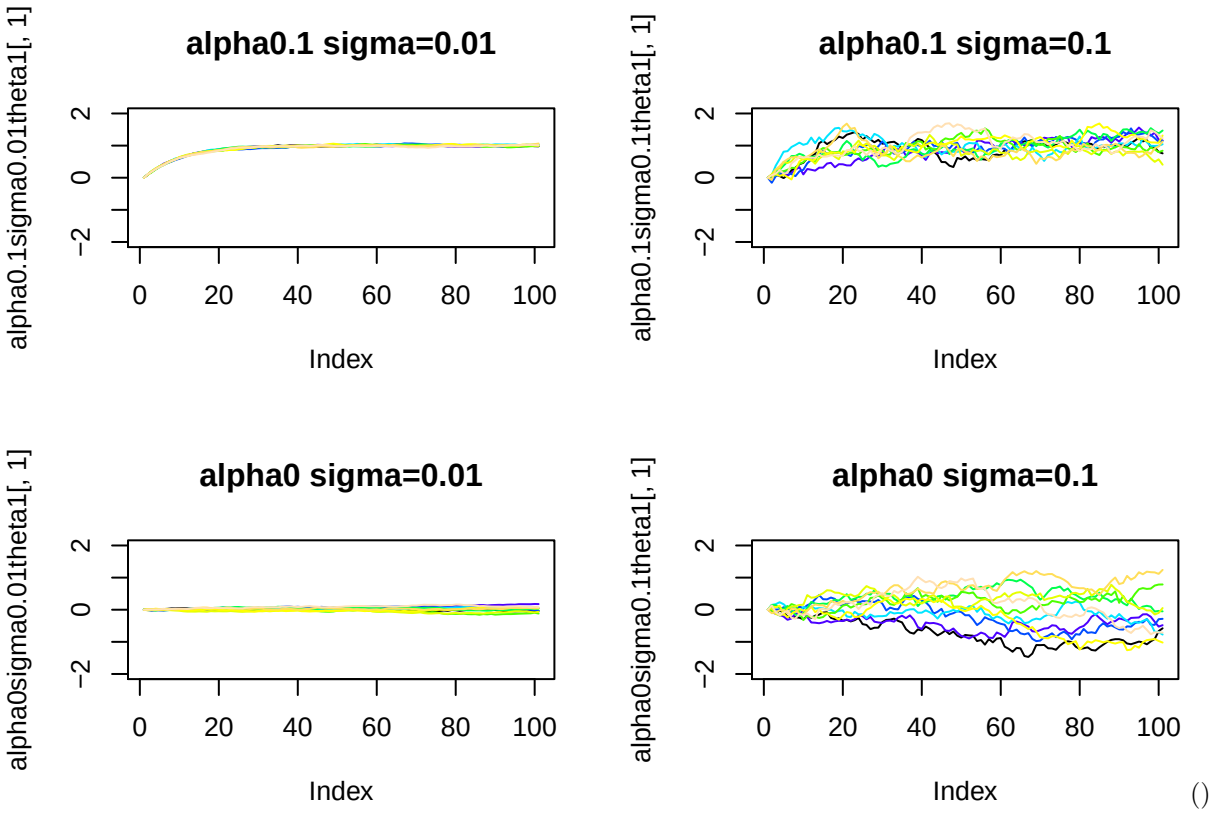
Here are some additional OU simulations to help you gain insight into the model.The following figure is too large for an HTML page but if you paste the code into yout console you should get something that looks decent.

```
alpha1sigma0.01theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 1, 0.01, 0)) # replicate just runs i


colr<-topo.colors(9)

plot(alpha1sigma0.01theta1[,1], type="l", main="alpha1 sigma=0.01", ylim=c(-2, 2))
for(i in 2:ncol(alpha1sigma0.01theta1)) lines(alpha1sigma0.01theta1[,i], type="l",col=colr[i-1] )

# play with different parameters.

alpha0.5sigma0.01theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0.5, 0.01, 0))
alpha0.1sigma0.01theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0.1, 0.01, 0))
alpha0sigma0.01theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0, 0.01, 0))
alpha1sigma0.1theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 1, 0.1, 0))
alpha0.5sigma0.1theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0.5, 0.1, 0))
alpha0.1sigma0.1theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0.1, 0.1, 0))
```

```
alpha0sigma0.1theta1<-replicate(10, ornstein_uhlenbecksim(100, 1, 0, 0.1, 0))

par(mfrow=c(2,2))

plot(alpha1sigma0.01theta1[,1], type="l", main="alpha1 sigma=0.01", ylim=c(-2, 2))
for(i in 2:ncol(alpha1sigma0.01theta1)) lines(alpha1sigma0.01theta1[,i], type="l",col=colr[i-1] )
plot(alpha1sigma0.1theta1[,1], type="l", main="alpha1 sigma=0.1", ylim=c(-2, 2))
for(i in 2:ncol(alpha1sigma0.1theta1)) lines(alpha1sigma0.1theta1[,i], type="l",col=colr[i-1] )
plot(alpha0.5sigma0.01theta1[,1], type="l", main="alpha0.5 sigma=0.01", ylim=c(-2, 2))
for(i in 2:ncol(alpha0.5sigma0.01theta1)) lines(alpha0.5sigma0.01theta1[,i], type="l",col=colr[i-1] )
plot(alpha0.5sigma0.1theta1[,1], type="l", main="alpha0.5 sigma=0.1", ylim=c(-2, 2))
for(i in 2:ncol(alpha0.5sigma0.1theta1)) lines(alpha0.5sigma0.1theta1[,i], type="l", col=colr[i-1] )
```




```
plot(alpha0.1sigma0.01theta1[,1], type="l", main="alpha0.1 sigma=0.01", ylim=c(-2, 2))
for(i in 2:ncol(alpha0.1sigma0.01theta1)) lines(alpha0.1sigma0.01theta1[,i], type="l", col=colr[i-1] )
plot(alpha0.1sigma0.1theta1[,1], type="l", main="alpha0.1 sigma=0.1", ylim=c(-2, 2))
for(i in 2:ncol(alpha0.1sigma0.1theta1)) lines(alpha0.1sigma0.1theta1[,i], type="l", col=colr[i-1] )
plot(alpha0sigma0.01theta1[,1], type="l", main="alpha0 sigma=0.01", ylim=c(-2, 2))
for(i in 2:ncol(alpha0sigma0.01theta1)) lines(alpha0sigma0.01theta1[,i], type="l", col=colr[i-1] )
plot(alpha0sigma0.1theta1[,1], type="l", main="alpha0 sigma=0.1", ylim=c(-2, 2))
for(i in 2:ncol(alpha0sigma0.1theta1)) lines(alpha0sigma0.1theta1[,i], type="l", col=colr[i-1] )
```



()

Part II: Model Fitting

Now let's look at some real data to test whether some morphological traits have evolved under different models of character evolution.

```
setwd("~/Dropbox/teaching/200a 2014/lab2 files")
library(ape)
library(geiger)
library(parallel)
tre <- read.tree("Haemulidae4models.tre")
dat<-read.csv("Haemulidae4models.csv", stringsAsFactors=FALSE) # Contains data on standard length,

trait <- dat[,2] # Selecting a trait
names(trait) <- tre$tip.label # Assigning the appropriate tip labels to the trait object
```

Let's start by fitting a Brownian motion model. We can do this with the fitContinuous() function in 'geiger'. We have to specify the phylogeny (tre), the trait(trait), and the type of model (BM = Brownian Motion).

```
BM.model <- fitContinuous(tre, trait, model="BM")
```

Geiger provides us with lots of information on the fitted model.The ouput includes details on estimated model parameters, including sigma squared and root state (z0) for BM. Model summaries (likelihood etc.) and convergence diagnostics are listed as well.

```
BM.model
```

```
## GEIGER-fitted comparative model of continuous data
##  fitted 'BM' model parameters:
##  sigsq = 0.002143
##  z0 = 2.233988
##
##  model summary:
##  log-likelihood = 12.308798
##  AIC = -20.617596
##  AICc = -20.362277
##  free parameters = 2
##
## Convergence diagnostics:
##  optimization iterations = 100
##  failed iterations = 0
##  frequency of best fit = 1.00
##
##  object summary:
##  'lik' -- likelihood function
##  'bnd' -- bounds for likelihood search
##  'res' -- optimization iteration summary
##  'opt' -- maximum likelihood parameter estimates
```

The output also tells you how to retrieve additional information, for examples the bounds for the likelihood search. You can extract the bounds with the following line:

```
    BM.model$bnd
```

```
##                  mn           mx
## sigsq 7.124576e-218 2.688117e+43
```

Primary results of the model fit are stored in the "opt" list, which can be accessed like this:

```
 BM.model$opt # The ouput includes info on sigma squared and the root state (z0)
```

```
## $sigsq
## [1] 0.002143211
##
## $z0
## [1] 2.233988
##
## $lnL
## [1] 12.3088
##
## $method
## [1] "Brent"
##
## $k
## [1] 2
##
## $aic
## [1] -20.6176
##
## $aicc
## [1] -20.36228
```

If you need to have the maximum likelihood estimate, you can type this command:

```
    BM.model$opt$lnL
```

```
## [1] 12.3088
```

You can get the AIC score (corrected for sample size) by

```
    BM.model$opt$aicc # You get the drift :)
```

```
## [1] -20.36228
```

Let's compare the fit of a BM and an OU model to the haemulid tree.

```
   OU.model <- fitContinuous(tre, trait, model="OU")
```

```
## Warning in fitContinuous(tre, trait, model = "OU"): Parameter estimates appear at bounds:
##  alpha
```

You may run into a problem here. Sometimes the bounds of the parameter estimates are too narrow. You can change the bounds with the bounds argument. See ?fitContinuous for more info.

```
   OU.model$bnd
```

```
##                  mn           mx
## alpha 7.124576e-218 2.718282e+00
## sigsq 7.124576e-218 2.688117e+43
```

```
    OU.model <- fitContinuous(tre, trait, model="OU", bounds=list(alpha=c(0, 150)))

  # How does the likelihood compare to the BM model?

    BM.model$opt$lnL
```

```
## [1] 12.3088
```

```
    OU.model$opt$lnL
```

```
## [1] 47.05235
```

```
  # It looks like the OU model has the higher likelihood...

  # One way to assess the difference between these two models more formally is the likelihood ratio tes
  # BM is a special case of OU (for alpha=0), so the models are nested and the LR test is fine.
  # Note that for thorough tests you should account for phylogenetic uncertainty and perform parameteri

    delta.BM.OU <- 1-pchisq(2*(OU.model$opt$lnL - BM.model$opt$lnL),1)
    delta.BM.OU
```

```
## [1] 1.110223e-16
```

```
  # Which model do you prefer?
```

We can also use AICc scores and Akaike weights to compare models. Brian O'Meara has very nice summary on his webpage: http://www.brianomeara.info/tutorials/aic. We don't have to calculate AIC and samplesize-corrected AIC scores (AICc) by hand, because 'geiger' does this for us:

```
    BM.model$opt$aicc # Higher AICc score is worse!
```

```
## [1] -20.36228
```

```
    OU.model$opt$aicc # Lower AICc score is better!
```

```
## [1] -87.58297
```

AIC and AICc scores are usually presented as differences to the best model (lowest score).

```
   all.aicc <- c(BM.model$opt$aicc, OU.model$opt$aicc)
    delta.aicc <- all.aicc - min(all.aicc) # This may seem a bit circumstantial for just two models, bu
    delta.aicc # delata AIC or AICC scores >2 are ususally considered to provide positive support
```

```
## [1] 67.22069  0.00000
```

AIC and AICc scores can also be expressed as Akaike weights, representing relative likelihood of the model (=exp(-0.5 * deltaAIC score for that model). Akaike weight for a given model are the rel. likelihood of the model divided by sum of all relative likelihhods across all models:

```
    rel.L <- exp(-delta.aicc*0.5)
    AK.weights <- rel.L/sum(rel.L)
    AK.weights
```

```
## [1] 2.530532e-15 1.000000e+00
```

Exercise 7

Have a look at all of the models that fitContinuous can evaluate.

```
 ?fitContinuous
```

Your assignment is to 1) choose buccal width or eye width as your trait 2) compare the fit of BM, OU, trend, and EB models using likelihood and AIC. The likelihood ratio test won't be very useful here as you don't have a nested hierachy of models. Report the likelihoods, AICcs, and delta AICs in a table (see Lab 1 for an example of how to extract and display these in an orderly way). What model or models fit your data best? Can you explain what the pattern of variation in your data set must be to support those fits? In other words, do you have any intuition for how clade trait diversity should scale with clade diversity for BM versus OU? How do these expectations differ for the trend model? We did not discuss the early burst model yet but you can read a brief description of the model using the help function. How should trait diveristy change through time under EB?

When fitting EB, you may again run into a little bit of a problem with the bounds of the model parameters. Remember that you can see the bounds used for the model fit by ,your.model.$bnd You can overwrite the

default bounds by adding the following argument to fitContinuous() Let's assume the parameter that must be changed is called 'a', and the new bounds are X and Y:

bounds=list(a=c(X, Y))

so... EB.model <- fitContinuous(tre, trait, model="EB", bounds=list(a=c(-10, 10)))