

# Intro to OUCH

Marguerite A. Butler<sup>1,2</sup>

<sup>1</sup>Department of Zoology, University of Hawaii, Honolulu, HI 96822

<sup>2</sup>[mbutler@hawaii.edu](mailto:mbutler@hawaii.edu)

August 12, 2011



# Contents

<b>1</b>	<b>Stochastic Models in Cartoons</b>	<b>5</b>
1.1	Brownian motion model . . . . .	5
1.2	Exercises . . . . .	9
1.3	R tricks – feel free to skip . . . . .	9
1.4	The OU Process . . . . .	10
1.5	Exercises . . . . .	12
1.6	Using the OU to model adaptive evolution . . . . .	13
1.7	Exercises . . . . .	14
1.8	Making movies . . . . .	14
1.9	RGL graphics . . . . .	15
<b>2</b>	<b>Introduction to OU Models</b>	<b>17</b>
2.1	The OU Model for Comparative Analysis . . . . .	17
2.2	ouch . . . . .	18
2.2.1	The Data . . . . .	18
2.2.2	Plotting ouchtrees . . . . .	21
2.2.3	Fitting models . . . . .	22
2.2.4	hansentree and ouchtree methods . . . . .	24
2.2.5	painting regimes on trees . . . . .	27
<b>3</b>	<b>Bivariate ouch</b>	<b>33</b>
3.0.6	The Bivariate model . . . . .	33
3.0.7	No Correlations . . . . .	33

3.1	Correlated Evolution . . . . .	34
3.2	Implementation in <code>ouch</code> . . . . .	34
3.3	Exercises . . . . .	42
3.4	Variations of the OU Model . . . . .	42
	<b>Bibliography</b>	<b>45</b>

# Chapter 1

## Stochastic Models in Cartoons

Let's make some graphical animations to illustrate the BM and OU model. We have all the tools to do this in R, to help us understand the differences between the models and how we can exploit their features to explore evolutionary scenarios.

### 1.1 Brownian motion model

We can describe a Brownian motion process using the following equation:

$$dX(t) = \sigma dB(t). \quad (1.1)$$

This equation says that the change in value of  $X$  in some small interval in time is determined by a random amount of change  $dB(t)$  times a scaling factor  $\sigma$ . That is, over a small increment of time,  $dX(t)$  is the infinitesimal change in the character  $X$  over the infinitesimal interval from time  $t$  to time  $t + dt$ . The term  $dB(t)$  is “white noise”; that is, the random variables  $dB(t)$  are independent and identically-distributed normal random variables, each with mean zero and variance  $dt$ .

We can mimic this behavior by a simulation in discrete time. If we take the above equation and approximate it as:

$$\Delta X = X_{i+1} - X_i = \sigma dB(t) \quad (1.2)$$

$$X_{i+1} = X_i + \sigma dB(t) \quad (1.3)$$

We can see that a simulation is simply taking the old value of  $X_i$  and adding a random amount of change to it to get the next value ( $X_{i+1}$ ), scaled (multiplied) by a constant value  $\sigma$ . If  $\sigma = 1$ , then it is in its most simplest form and we can code this simply as:

```
> x[i+1] <- x[i] + devs[i]
```

Where `devs` is a random deviate. So let's do it! Start with 100 generations, and make 100 random draws.

```
> ngens = 100          # number of generations in our simulation
> devs =rnorm(ngens)    # 100 draws from a normal distribution or ``deviates"
```

Create a vector for our phenotype `x`, and fill it by starting from 0, and adding a random deviate to `x` at each generation.

```
> x <- c(0:100)        # initialize x (create a vector of the appropriate length), our phe
> for (i in 1:ngens)    # a loop which will run ngens times, each time increasing the
+ {
+   x[i+1] <- x[i] + devs[i]
+ }
```

```
> x          # take a look at the values of x, this is how x changes each generation
```

```
[1] 0.000000000 0.744462138 -0.216999387 0.277856120 -0.850111862
[6] -1.045048247 0.124522715 1.994246140 2.205044952 1.145236347
[11] 0.722000790 0.111762633 -0.070591768 0.379538270 -0.540557529
[16] 0.320167994 -0.628951227 0.897313139 -0.625029234 1.454085336
[21] 1.485054936 2.494038803 2.557030189 3.119973868 4.153774817
[26] 2.939491983 2.661497197 0.645652361 0.916461859 1.712422724
[31] 1.567829823 -0.008616502 -1.055501233 -2.042286346 -1.921369177
[36] -2.541656751 -1.527155194 -0.660870854 -2.345543414 -1.984317043
[41] -1.210537259 -2.553489274 -2.864482182 -1.374771512 0.017544791
[46] 0.400916346 -0.512301506 -0.396377305 -0.466476620 -1.387386283
[51] -2.002001457 -1.803606005 -1.266239782 -2.124252317 -2.296738038
[56] -0.528885592 -0.815100953 0.160312384 -1.330286139 -2.549568164
[61] -2.313574420 -2.776887146 -2.851452203 -2.859195351 -2.681308278
[66] -2.151420135 -2.513683598 -3.034923120 -2.524896260 -2.300414295
[71] -3.657710236 -2.936822999 -2.198246288 -2.343157252 -5.014247378
[76] -3.435313675 -4.315546247 -4.068479751 -6.193193538 -7.094018953
[81] -7.500748544 -6.812502078 -6.460119097 -5.612358045 -7.337165206
[86] -6.848191225 -6.885888016 -5.932932776 -5.642627458 -5.281215971
[91] -6.643569466 -7.102560988 -8.076057099 -9.457997064 -8.242059483
[96] -7.639423576 -7.206378679 -8.146783918 -8.004129281 -8.192078248
[101] -8.051097325
```

We can plot this single random walk. The function `plot()` creates a new plot window (and plots data, except that we told it to plot nothing by the parameter `type="n"` – we did this in order to plot the simulation slowly. The actual simulations are added one generation at a time by the command `lines()`).

```

> plot(1:length(devs), devs, type = "n", col="red",
+ ylim = c(-max(devs)*30, max(devs)*30),
+ xlab="Time", ylab="Value", main="BM Simulation")
> x <- c(0:100)
> for (i in 1:ngens)
+ {
+   x[i+1] <- x[i] + devs[i]
+   lines(i:(i+1), x[i:(i+1)], col="red")      # plotting the simulation
+ }

```

We can add  $\sigma$  in now simply by modifying the line of code specifying the BM:

```

>   x[i+1] <- x[i] + sigma * devs[i]

```

In order to do 30 random walks (30 lineages), we need to place an outer loop, once for each random walk:

```

> bm.plot <- function( sigma=1, ngens=100, nwalks=30, ylim=c(-50, 50) )
+ {
+   plot(1:length(devs), devs, type = "n", col="red", ylim = ylim,
+     xlab="Time", ylab="Value", main="BM Simulation")
+
+   for (i in 1:nwalks)      # number of lineages to simulate
+   {
+     x <- c(0:ngens)
+     devs = rnorm(ngens)    # draws from a normal distribution, one per generation
+     for (i in 1:ngens)
+     {
+       x[i+1] <- x[i] + sigma*devs[i]      # BM equation
+       # step through time, increasing x a little bit each time
+       lines(i:(i+1), x[i:(i+1)], col="red") # plot line segment
+     }
+   }
+ }

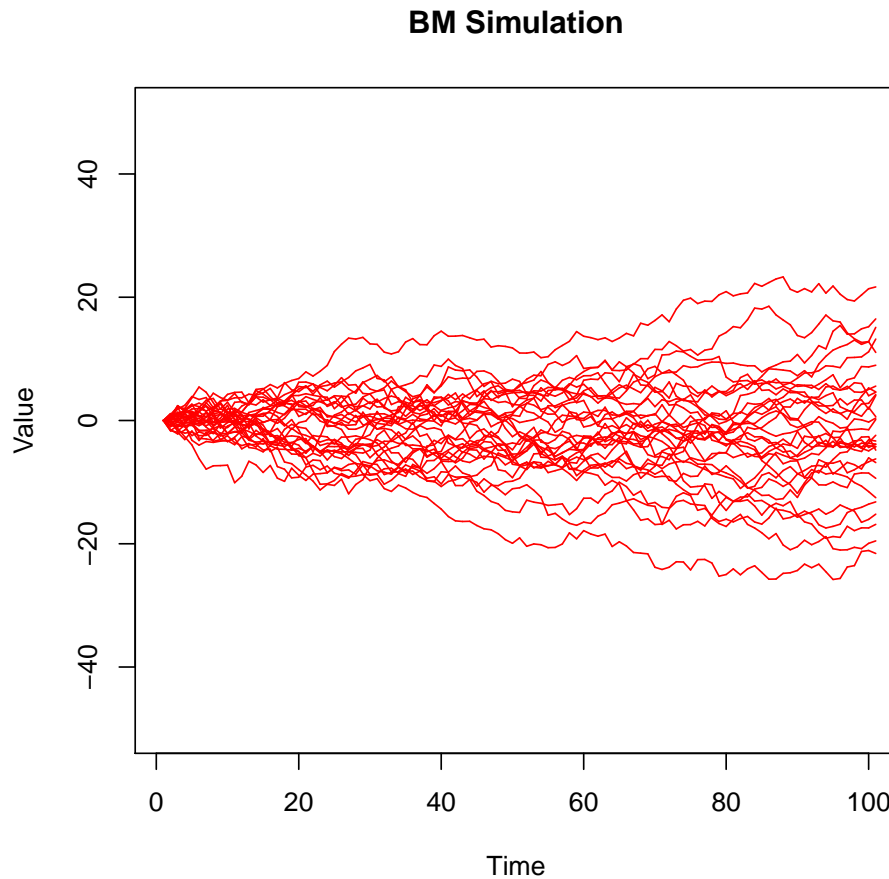
```

Now run the simulation:

```

> bm.plot()

```



Now that we have created a function, we can play with it by varying the parameters to see what they mean. For example: `bm.plot( sigma=4 )` see the difference? `bm.plot( sigma=1, ngens=300 )` @

For the following exercises, it will be helpful to collect the final values of each lineage. To do this, we need to (1) collect the final values, and (2) return them from the function. We can do this by adding in the lines for `xfinal` below:

```
> bm.plot <- function( sigma=1, ngens=100, nwalks=30, ylim=c(-30, 30) )
+ {
+   plot(1:length(devs), devs, type = "n", col="red", ylim = ylim,
+   xlab="Time", ylab="Value", main="BM Simulation")
+   xfinal <- c(1:nwalks)      ### initialize xfinal
+
+   for (j in 1:nwalks)
+   {
+     x <- c(0:ngens)
+     devs =rnorm(ngens)
```



```

+       for (i in 1:ngens)
+       {
+           x[i+1] <- x[i] + sigma*devs[i]      # BM equation
+           # step through time, increasing x a little bit each time
+           lines(i:(i+1), x[i:(i+1)], col="red") # plot line segment
+       }
+       xfinal[j] <- x[ngens+1]      ### collect the last value at each lineage
+   }
+   return(xfinal)      ### return the final values
+ }

```

## 1.2 Exercises

The values at the end of the simulation represent the phenotypic value after evolving for *codengens* generations. Prove to yourself the following properties of Brownian Motion processes:

1. The mean value at the end of the process is centered on the value at the start of the process (although it may be very noisy from simulation to simulation, meaning you may have to do a lot of simulations to get a good estimate of the mean).
2. The variance grows in proportion to time. Remember that in the BM, the variance is given by  $\sigma^2 t$ .

## 1.3 R tricks – feel free to skip

The loop is easier to understand in terms of a stochastic process, but actually we can write this code much more compactly:

Adding up a series of BM steps using the cumulative sum function:

```

> sigma=1
> cumsum(rnorm(ngens, sd=sigma))

```

Plotting all the line segments at once:

```

> y <- c(0, cumsum(rnorm(ngens, sd=sigma)))
> lines(0:ngens, y)

```

Or even more compactly:

```
> sigma=1
> lines(0:ngens, c(0, cumsum(rnorm(ngens, sd=sigma))))
```

And doing all of the lineages using `lapply`:

```
> bm.plot <- function( sigma=1, ngens=100, nwalks=30, ylim=c(-50, 50))
+ {
+ # Set up plotting environment
+   plot(0, 0, type = "n", xlab = "Time", ylab = "Trait",
+        xlim=c(0, ngens), ylim=ylim)
+
+ # Draw random deviates and plot
+   lapply( 1:nwalks, function(x)
+     lines(0:ngens, c(0, cumsum(rnorm(ngens, sd=sigma))))))
+ }
```

If you want to show the simulations to screen, then you may actually prefer to do the slower for-loops, as the `lapply` is too fast.

## 1.4 The OU Process

Recall that we described a Brownian motion process using the following equation:

$$dX(t) = \sigma dB(t). \quad (1.4)$$

The simplest stochastic model that incorporates selection is the Ornstein-Uhlenbeck process [Uhlenbeck and Ornstein \(1930\)](#):

$$dX(t) = \alpha (\theta - X(t)) dt + \sigma dB(t). \quad (1.5)$$

Eq. [2.2](#) adds two new parameters to our changing phenotype. The parameter  $\alpha$  measures the strength of selection. When  $\alpha = 0$ , the deterministic part of the OU model drops out and [\(2.2\)](#) collapses to the familiar BM model of pure drift.

The OU process was first described in 1930. Russ Lande showed that the evolution of a species' mean phenotype could be described by an OU process, representing stabilizing selection [Lande \(1976\)](#), and was discussed by [Felsenstein \(1988\)](#). Thomas Hansen first recognized its usefulness in phylogenetic comparative analysis [Hansen \(1997\)](#) to describe adaptive evolution along a phylogeny. Butler and King developed this application further [Butler and King \(2004\)](#) and expanded upon the use of the OU model to represent alternative models of adaptive and non-adaptive evolution, as well as a rigorous model-comparison framework.

We can again write a simple simulation by copying the form of the equation, with parameters for  $\alpha$  and  $\theta$ :

```
> x[i+1] = alpha*(theta-x[i])+x[i] + sigma*devs[i]
```

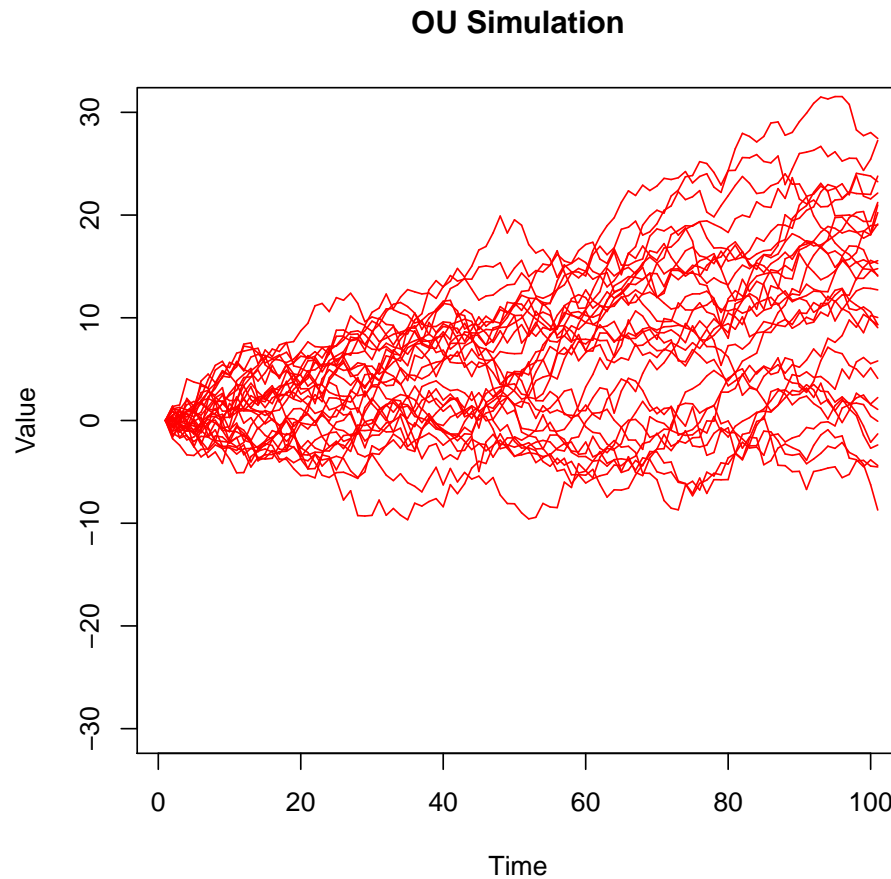
OK! Let's simulate the OU process:

```
> ou.plot <- function( alpha=0.005, theta=0, sigma=1, ngens=100, nwalks=30,
+ ylim=c(-30, 30) )
+ {
+   plot(1:length(devs), devs, type = "n", col="red", ylim = ylim,
+   xlab="Time", ylab="Value", main="OU Simulation")
+   xfinal <- c(1:nwalks)
+   for (j in 1:nwalks)      # number of lineages to simulate
+   {
+     x <- c(0:ngens)
+     devs =rnorm(ngens)
+     for (i in 1:ngens)
+     {
+       x[(i+1)] = alpha*(theta-x[i])+x[i] + sigma*devs[i]    # OU eq
+       lines(i:(i+1), x[i:(i+1)], col="red")    # plot line segment
+     }
+     xfinal[j] <- x[ngens+1]    ## collect last value at each lineage
+   }
+   return(xfinal)            ### return the final values
+ }
```

You can run the function by typing:

```
> ou.plot( theta=30 )
```

```
[1]  6.08921384  4.72895357 14.29856946  7.30704195 31.01381330  8.94375197
[7] 19.67031117 14.26942261  8.42887230  5.01125743 29.38009569  3.43223404
[13]  6.12642411 21.23142466 25.83394865 15.02515763 18.18470002  2.36389169
[19]  0.29109179 15.57007748 10.65020548 10.72746658 11.46775078  8.33916492
[25] 10.38746828 18.72944767 15.54081632  0.06960877 12.65612399 13.64733399
```



Do you see any difference? The  $\sigma$  value is 1 as in the BM simulation, but here the default value of  $\alpha = -.005$ , and we set the optimum value as  $\theta = 30$ .

## 1.5 Exercises

1. What is the effect of bigger or smaller  $\alpha$ ? Does the simulation always reach the optimum value you set? What else could you change?
2. What is the effect of changes in  $\theta$ ?
3. What is the difference between BM simulations and OU simulations with similar  $\sigma$ ? Try your own or do a BM simulation with  $\sigma = 1$ , and OU with  $\sigma = 1$ ,  $\alpha = 0.01$ , and  $\theta = 0$ .

## 1.6 Using the OU to model adaptive evolution

[Simpson \(1953\)](#) painted a vivid picture of adaptive evolution in macroevolutionary time. He imagined lineages evolving through time within an adaptive zone. On rare occasions, the lineage may enter a new adaptive zone, which would result in morphological adaptations that reflect their new functional demands. In some cases, entering a new adaptive zone may even result in special bursts of evolution, perhaps adaptive radiation. However, we are getting ahead of ourselves.

We can use the OU to model adaptive evolution. Not only does it have a term for selection, but also for adaptive optima. These adaptive optima can be used to represent selective regimes, or lifestyles, or different suites of ecological pressures. While we limit each branch of the phylogeny to having only one optima, we can assign different optima to different branches, according to our biological hypothesis. For example, we may have a lizard lineage that is ancestrally insectivorous, but they may have evolved herbivory along some of its branches (Fig. 1.1).

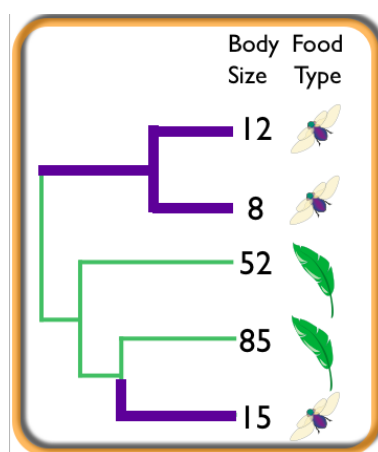


Figure 1.1: Hypothetical adaptive regimes. We group branches leading to herbivory in a separate adaptive regime relative to those leading to insectivory.

Note that we don't assign a unique optimum to every branch, that would result in too many parameters and not enough data to estimate them unambiguously. Rather, we are postulating a limited number of adaptive regimes – the simplest model that can explain most of the data.

Download the function `OU.sim.branch` and load it into your R workspace by the following command (make sure you download it to your working directory):

```
> source("OU.sim.branch.R")
```

If you'd like to see the function, just type it's name:

```
> OU.sim.branch
```

and the function will dump to screen. In particular look at the default parameters in the function call and do the exercises below.

## 1.7 Exercises

Is the OU process, as simple as it is, flexible enough to suit our needs? Let's try some simulations, this time with a branch along the phylogeny, and see if we can simulate changes in the phenotype.

1. Can you make the phenotype diverge in different branches of the phylogeny?
2. Food for thought. Does the simulation always reach the optimum? What are the possible explanations?

## 1.8 Making movies

In order to make a movie of the plot, you will need to save a series of plots as separate graphics files, similar to the "flip-books" you played with as a child. You need to make a plot with the first lineage, then the first two lineages, then the first three lineages, and so on.

So it would make sense to make a matrix to store the lineages, then plot through cumulatively:

```
> ngens=100
> nwalks=30
> sigma=1
> sims <- sapply(1:nwalks, function(x) c(0, cumsum(rnorm(ngens, sd=sigma))))
> ylim <- c(-30, 30)
> png(filename="movies/Rplot%03d.png")
> # turn on png graphical device (write to file)
> for (i in 2:nwalks)
+ {
+   plot(0, 0, type = "n", xlab = "Time", ylab = "Trait",
+   xlim=c(0, ngens), ylim=ylim)
+   apply( sims[,1:i], 2 , function(x) lines(0:ngens, x, col="red"))
+ }
> dev.off() # turn off png
```

Then in a terminal, move into the `movies` directory. If you have `imagemagik` installed:

```
convert -delay 10 Rplot.png Rplot.gif
```

To make a `.mov` file, you can use Quicktime Pro (but you have to pay for the Pro upgrade). In R version 2.8 there is a new package named `animation` which calls ImageMagick from R.

## 1.9 RGL graphics

The 3D animations that I showed were produced using the package `rgl`.

You can see the graph gallery at <http://rgl.neoscientists.org/gallery.shtml>. I have also included my source code in the wiki. Under `ou2drgl.R`.





# Chapter 2

## Introduction to OU Models

Goals:

- Approaches for adaptive evolution (ouch, slouch, others)
- Model-based vs statistical approaches

Concepts:

- Model comparison tools in R
- Process-based models

### 2.1 The OU Model for Comparative Analysis

Recall that we described a Brownian motion process using the following equation:

$$dX(t) = \sigma dB(t). \quad (2.1)$$

If we imagine the phenotype  $X$  as changing through time  $t$ , this equation says that in a small increment of time, the change will be proportional to the parameter  $\sigma$ . Here,  $dB(t)$  is a sample from a Brownian (white noise) process.

A small step towards reality is the OU Process:

$$dX(t) = \alpha (\theta - X(t)) dt + \sigma dB(t). \quad (2.2)$$

Eq. 2.2 expresses the amount of change in character  $X$  over the course of a small increment of time: specifically,  $dX(t)$  is the infinitesimal change in the character  $X$  over the

infinitesimal interval from time  $t$  to time  $t + dt$ . The term  $dB(t)$  is “white noise”; that is, the random variables  $dB(t)$  are independent and identically-distributed normal random variables, each with mean zero and variance  $dt$ . The parameter  $\alpha$  measures the strength of selection. When  $\alpha = 0$ , the deterministic part of the OU model drops out and (2.2) collapses to the familiar BM model of pure drift,

## 2.2 ouch

See ouch lecture.

Good starting points:

?bimac help page for *Bimaculatus* character displacement dataset

example(bimac) example of bimac analysis

?anolis.ssd help page for *Anolis* sexual size dimorphism dataset

ouch is a package designed to test adaptive hypotheses using variations of the OU process, including BM. OUCH implements a model that fits an alpha and sigma parameters to the entire phylogeny, but allows the user to specify which branches belong to different selective regimes. The location of the optima are also fit.

### 2.2.1 The Data

The data in OUCH are most easily assembled as a data frame. Load the built in example from ouch and then print it to the screen (I only printed the head of the dataset here):

```
> require(ouch)
> data(bimac)
> bimac
```

	node	species	size	ancestor	time	OU.1	OU.3	OU.4	OU.LP
1	1	<NA>	NA	NA	0	ns	medium	anc	medium
2	2	<NA>	NA	1	12	ns	medium	anc	medium
3	3	<NA>	NA	2	32	ns	medium	anc	small
4	4	<NA>	NA	3	34	ns	medium	anc	small
5	5	<NA>	NA	4	36	ns	medium	anc	small
6	6	<NA>	NA	3	36	ns	medium	anc	small

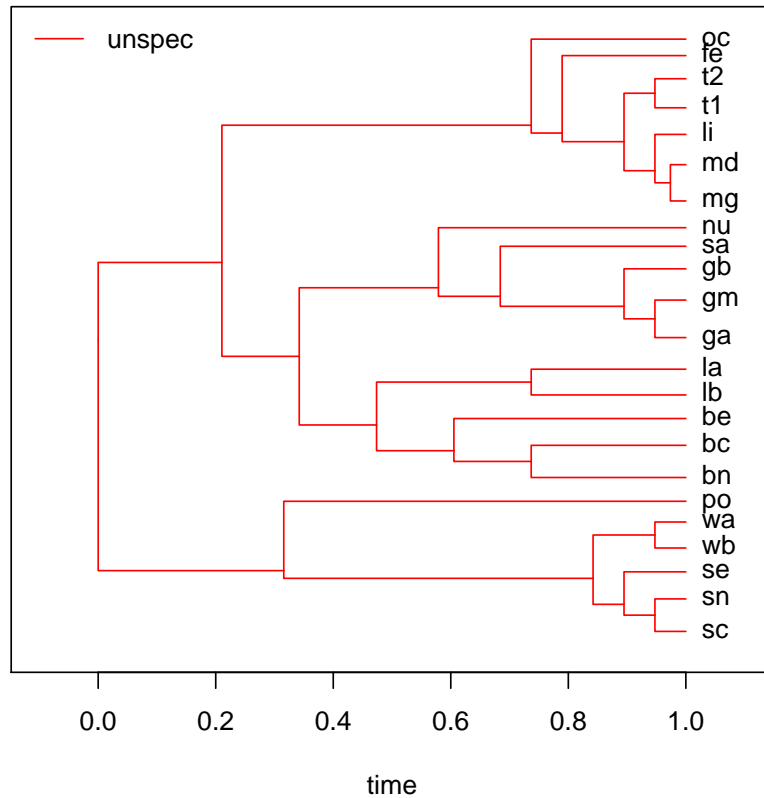
NOTE: a very important detail about **ouch** is that it matches trees with data and regimes using the node labels stored in the rownames of the objects you pass to the **ouch** functions. So it is important to make sure that your dataframes and vectors are appropriately named. The dataframe **bimac** already has the correct row names, but we do so here just to illustrate.

```
> rownames(bimac) <- bimac$node
```

**ouch** was designed around a rectangular data model, so although the tree object is not a dataframe internally, it still helps us to build the data as a dataframe before making the **ouchtree** objects. The central organizing element is the **node**: it has a node number (usually an integer but it is actually a unique character string), an **ancestor** to which it is joined by a branch, a **time** since the root of the tree, and optional **label** such as a species name. The hypotheses which we use are assigned by "painting" particular regimes on branches. It is convenient to represent each model or hypothesis as a column on the dataframe, with the regime assigned to the node (that is, it is assigned to the branch connecting the node to its ancestor).

Make an **ouchtree** object using the **ouchtree** constructor. **with** is a very nice function to create a small local environment so that you can use a dataframe's elements directly without using the **bimac\$** prefix. It is similar to an **attach** but it is temporary – only lasting as long as the call itself. I like it much better than **attach** because I sometimes forget what I've attached and run into problems later. Also, with **attach**, you are actually working with a copy of the original dataframe object, so updating values is tricky. With **with**, it is more clear what's going on, and I don't tend to make those mistakes.

```
> tree <- with(bimac, ouchtree(node,ancestor,time/max(time),species))
> plot(tree)
```



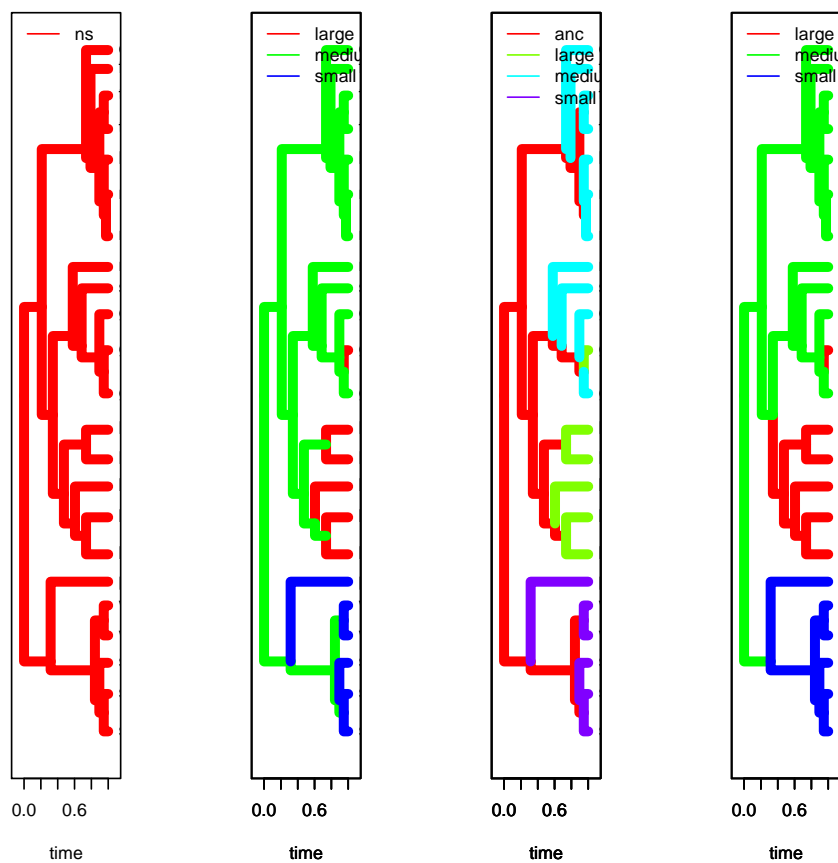
ouch fits the OU model Eq. 2.2 along each branch of the phylogeny. While  $\alpha$  and  $\sigma$  are held constant across the entire tree, the optima along each branch  $\theta$  are allowed to vary. Users can then “paint” various combinations of optima on the tree to reflect various biological scenarios.

For example, the dataset **bimac** was used to test the hypothesis of character displacement using an interspecific dataset of body sizes and sympatry/allopatry [Butler and King \(2004\)](#). The analysis tested several different models, which are included with **bimac**. They are: “OU.1” or global optimum, “OU.3” or small, medium, and large regimes depending on the body size of the observed species (terminal branches only, internal branches painted “medium”, “OU.4” or the same as “OU.3” but with internal branches given their own unique regime called “ancestral”, and “OU.LP” based on a linear parsimony reconstruction of the colonization events (i.e., that as species came into sympatry, they diverged in body size).

### 2.2.2 Plotting ouchtrees

You can plot the regime paintings on the tree, and set options such as line widths for prettier plots. `ouch` has a very nice feature which allows plotting of the alternative models on one plot.

```
> plot(tree, regimes=bimac[c("OU.1", "OU.3", "OU.4", "OU.LP")], lwd=6)
```



Remember that you can pass a single vector or a data frame to the regimes parameter, but it must have the appropriate row names or names in the case of a vector. The regimes are not part of the ouchtree object, because they represent our hypothesis of evolution along the tree, rather than the tree itself. It is part of the original dataframe from which we derived the tree, so remember to refer to `bimac` when passing the regimes to the `plot` function.

### 2.2.3 Fitting models

There are two main model fitting functions in **ouch**, **brown**, which fits Brownian motion models, and **hansen**, which fits OU models to comparative data. The call to **brown** is particularly simple, as it takes only the data and the tree:

```
> brown(log(bimac['size']), tree)
```

call:

```
brown(data = log(bimac["size"]), tree = tree)
```

	nodes	ancestors	times	labels	size
1	1	<NA>	0.0000000	<NA>	NA
2	2	1	0.3157895	<NA>	NA
3	3	2	0.8421053	<NA>	NA
4	4	3	0.8947368	<NA>	NA
5	5	4	0.9473684	<NA>	NA
6	6	3	0.9473684	<NA>	NA
7	7	1	0.2105263	<NA>	NA
8	8	7	0.3421053	<NA>	NA
9	9	8	0.4736842	<NA>	NA
10	10	9	0.6052632	<NA>	NA
11	11	10	0.7368421	<NA>	NA
12	12	9	0.7368421	<NA>	NA
13	13	8	0.5789474	<NA>	NA
14	14	13	0.6842105	<NA>	NA
15	15	14	0.8947368	<NA>	NA
16	16	15	0.9473684	<NA>	NA
17	17	7	0.7368421	<NA>	NA
18	18	17	0.7894737	<NA>	NA
19	19	18	0.8947368	<NA>	NA
20	20	19	0.9473684	<NA>	NA
21	21	20	0.9736842	<NA>	NA
22	22	19	0.9473684	<NA>	NA
23	23	2	1.0000000	po	2.602690
24	24	4	1.0000000	se	2.660260
25	25	5	1.0000000	sc	2.660260
26	26	5	1.0000000	sn	2.653242
27	27	6	1.0000000	wb	2.674149
28	28	6	1.0000000	wa	2.701361
29	29	10	1.0000000	be	3.161247
30	30	11	1.0000000	bn	3.299534
31	31	11	1.0000000	bc	3.328627
32	32	12	1.0000000	lb	3.353407

```

33    33      12 1.0000000    la 3.360375
34    34      13 1.0000000    nu 3.049273
35    35      14 1.0000000    sa 2.906901
36    36      15 1.0000000    gb 2.980619
37    37      16 1.0000000    ga 2.933857
38    38      16 1.0000000    gm 2.975530
39    39      17 1.0000000    oc 3.104587
40    40      18 1.0000000    fe 3.346389
41    41      20 1.0000000    li 2.928524
42    42      21 1.0000000    mg 2.939162
43    43      21 1.0000000    md 2.990720
44    44      22 1.0000000    t1 3.058707
45    45      22 1.0000000    t2 3.068053

```

```

sigma squared:
      [,1]
[1,] 0.04311003

```

```
theta:
```

```
NULL
```

```

      loglik deviance      aic      aic.c      sic      dof
17.33129 -34.66257 -30.66257 -30.06257 -28.39158  2.00000

```

What is returned is an object of class **browntree**. It contains all input including the function call, the tree and data), as well as the parameter estimate for  $\sigma$  and the model fit statistics including: the log-likelihood, the deviance ( $-2 * \log(L)$ ), the information criteria  $AIC$ ,  $AIC_c$  (corrected for small sample size), and  $SIC$ , and the model degrees of freedom.

It is a good practice to save this, as it encapsulates the analysis. From this, we can rerun the model fit.

```
> h1 <- brown(log(bimac['size']),tree)
```

**hansen** models are slightly more complex. In addition to  $\sigma$ , we are now fitting  $\alpha$ , the strength of selection, and all of the optima  $\theta$  specified by our model. This maximum-likelihood search now requires an initial guesses. If you have no idea, a good starting guess is 1. If you want to be sure, you can initiate searches with different starting guesses. You can also specify alternative optimization algorithms and increase or decrease the relative tolerance, which is the stringency by which convergence is assessed. Typically, the default is roughly `reltol=1e-8`, and the limit of machine precision is in the neighborhood of `reltol=1e-15`.

```

> h2 <- hansen(log(bimac['size']),tree,bimac['OU.1'],sqrt.alpha=1,sigma=1)
> h3 <- hansen(log(bimac['size']),tree,bimac['OU.3'],sqrt.alpha=1,sigma=1)

```

```
> h4 <- hansen(log(bimac['size']),tree,bimac['OU.4'],sqrt.alpha=1,sigma=1)
> h5 <- hansen(log(bimac['size']),tree,bimac['OU.LP'],sqrt.alpha=1,sigma=1,reltol=1e-5)
```

## 2.2.4 hansentree and ouchtree methods

We can see the model results by typing `h5`, which will execute the `print` method for this class. You could also use the `attributes` function, but this will dump too much information. `ouchtree` objects and the classes derived from them contain information that is used in internal calculations of the algorithms, not of general interest to users.

Additional accessor functions include:

```
> coef(h5)      # the coefficients of the fitted model
```

```
$sqrt.alpha
[1] 1.61658
```

```
$sigma
[1] 0.2249274
```

```
$theta
$theta$size
      large   medium   small
3.355087 3.040729 2.565249
```

```
$alpha.matrix
      [,1]
[1,] 2.61333
```

```
$sigma.sq.matrix
      [,1]
[1,] 0.05059232
```

```
> logLik(h5)    # the log-likelihood value
```

```
[1] 24.81823
```

```
> summary(h5)   # everything in the print method except the tree+data
```

We can now generate a table of our model fits:



```
> unlist(summary(h5)[c('aic', 'aic.c', 'sic', 'dof')]) # just the model fit statistics
```

```
      aic      aic.c      sic      dof
-39.63645 -36.10704 -33.95898  5.00000
```

```
> # on a single line
> h <- list(h1, h2, h3, h4, h5) # store all our fitted models in a list
> names(h) <- c("BM", "OU.1", "OU.3", "OU.4", "OU.LP")
> sapply( h, function(x) unlist(summary(x)[c('aic', 'aic.c', 'sic', 'dof')]) )
```

```
      BM      OU.1      OU.3      OU.4      OU.LP
aic   -30.66257 -25.39364 -29.15573 -35.22319 -39.63645
aic.c -30.06257 -24.13048 -25.62631 -29.97319 -36.10704
sic   -28.39158 -21.98715 -23.47826 -28.41022 -33.95898
dof     2.00000  3.00000  5.00000  6.00000  5.00000
```

```
> # table with all models
```

By storing the model fits in a list, we can use apply methods to get the statistics from all the models at once. `sapply` returns a matrix if possible.

```
> h.ic <- sapply( h, function(x) unlist(summary(x)[c('aic', 'aic.c', 'sic', 'dof')]) )
> print( h.ic, digits = 3)
```

```
      BM  OU.1  OU.3  OU.4  OU.LP
aic   -30.7 -25.4 -29.2 -35.2 -39.6
aic.c -30.1 -24.1 -25.6 -30.0 -36.1
sic   -28.4 -22.0 -23.5 -28.4 -34.0
dof     2.0   3.0   5.0   6.0   5.0
```

Simulation and bootstrap methods: `simulate` generates random deviates or sets of simulated tip data based on the fitted model. The input is a fitted model `hansentree` or `browntree`, and the output is a list of dataframes, each comparable to the original data. These can then be used to refit the model.

```
> h5.sim <- simulate(object = h5, nsim=10) # saves 10 sets of simulated data
> # based on OU.LP
```

`update` refits the model, with one or more parameters changed.

```
> summary( update( object = h5, data = h5.sim[[1]] ) ) # fit the first dataset
```

```
$call
hansen(data = data, tree = object, regimes = regimes, sqrt.alpha = sqrt.alpha,
        sigma = sigma)

$conver.code
[1] 0

$optimizer.message
NULL

$alpha
      [,1]
[1,] 5.646629

$sigma.squared
      [,1]
[1,] 0.04180986

$optima
$optima$size
      large   medium   small
3.277436 3.018628 2.681822

$loglik
[1] 33.35916

$deviance
[1] -66.71831

$aic
[1] -56.71831

$aic.c
[1] -53.1889

$sic
[1] -51.04084

$dof
[1] 5

> h5.sim.fit <- lapply( h5.sim, function(x) update(h5, x)) # fit all 10 simulations
```

`bootstrap` is a convenience function for generating parametric bootstraps of the parameter estimates. It takes the fitted model, performs the simulations, refits, and outputs a dataframe of parameter estimates.

```
> bootstrap( object = h5, nboot=10)
```

	alpha	sigma.squared	optima.size.large	optima.size.medium
1	4.147800	0.05325362	3.330190	3.036104
2	6.419653	0.07680162	3.238138	2.970032
3	5.303491	0.05297610	3.312611	3.047818
4	2.237698	0.04437131	3.397779	3.054254
5	7.222022	0.03545880	3.321527	3.092630
6	14.613127	0.12219208	3.285785	3.049920
7	2.419415	0.03871822	3.384419	3.082739
8	7.796940	0.13684568	3.221195	2.995464
9	1.771920	0.02622271	3.477032	2.996087
10	6.212593	0.09162454	3.374101	3.050028

	optima.size.small	loglik	aic	aic.c	sic	dof
1	2.664868	27.85601	-45.71203	-42.18262	-40.03456	5
2	2.619641	27.55767	-45.11534	-41.58592	-39.43787	5
3	2.687239	30.06571	-50.13143	-46.60202	-44.45396	5
4	2.484811	25.24241	-40.48482	-36.95540	-34.80734	5
5	2.674400	37.57124	-65.14248	-61.61307	-59.46501	5
6	2.660018	30.60587	-51.21174	-47.68233	-45.53427	5
7	2.687687	27.34489	-44.68979	-41.16037	-39.01232	5
8	2.621692	22.78675	-35.57350	-32.04409	-29.89603	5
9	2.445950	29.79428	-49.58856	-46.05915	-43.91109	5
10	2.550013	25.22055	-40.44109	-36.91168	-34.76362	5

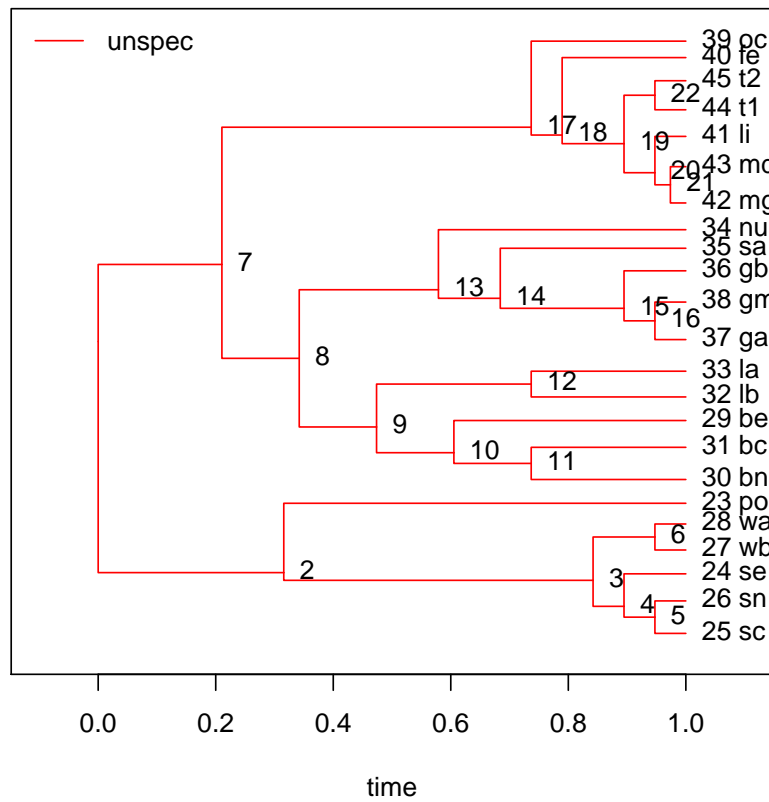
### 2.2.5 painting regimes on trees

A new function in `ouch` is `paint`. Previously, it was up to users to set up regimes manually by editing spreadsheets. `paint` helps with this task by specifying the regimes on particular species, subtrees, or particular branches.

There are two parameters to `paint`, `subtrees`, which paints the entire subtree which descends from the node, and `branch`, which paints the branch connecting the node to its ancestor. For either, you specify the node label (remember it's a character and needs to be quoted), and set it equal to the name of the regime you want to specify.

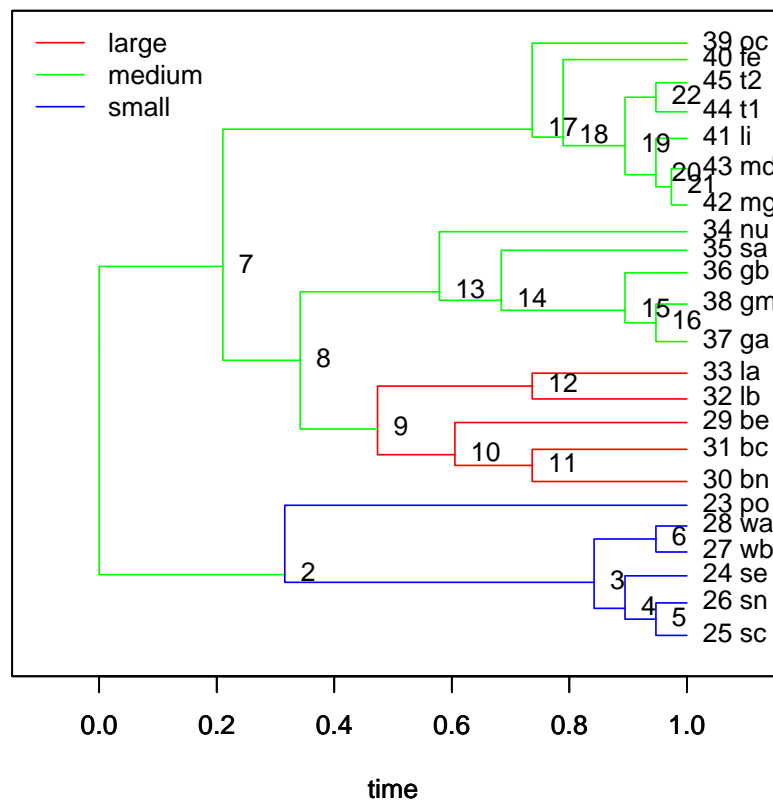
Let's try it on the `bimac` tree and try to recreate the OU.LP regime:

```
> plot(tree, node.names=T)
```



Paint the subtrees first, take a look:

```
> ou.lp <- paint( tree, subtree=c("1"="medium", "9"="large", "2"="small") )
> plot(tree, regimes=ou.lp, node.names=T)
```

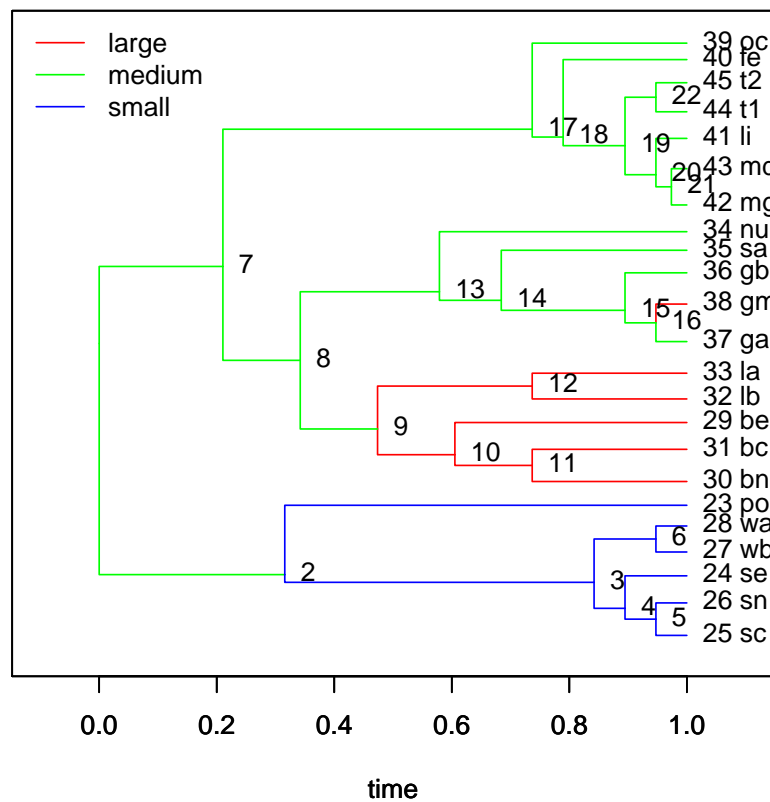


But there was an independent switch from medium to large at species "gm", or node "38", and the node connecting "9" to its ancestor:

```
> ou.lp <- paint( tree, subtree=c("1"="medium", "9"="large", "2"="small"),
+ branch=c("38"="large", "2"="medium"))
```

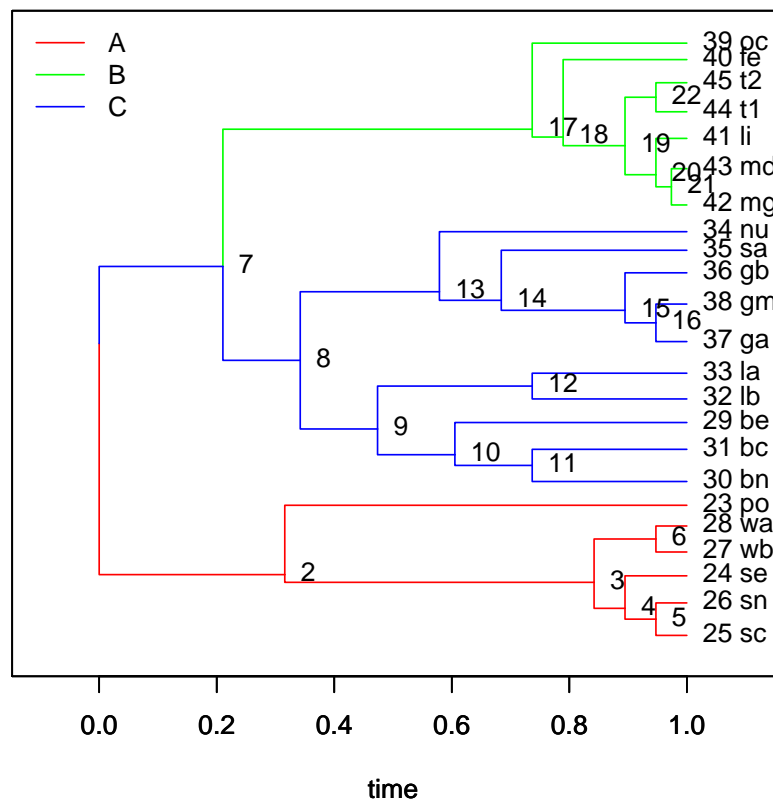
Compare it to the original "OU.LP" from above.

```
> plot(tree, regimes=ou.lp, node.names=T)
```



We can create alternative paintings of the regimes to test against the data. Suppose we wanted to add a "clade specific" hypothesis that diverged in a similar time period (this is a completely made-up hypothesis, just for example):

```
> ou.clades <- paint( tree, subtree=c("1"="A", "7"="B", "8"="C"),
+ branch=c("8"="C", "7"="C", "1"="A"))
> plot(tree, regimes=ou.clades, node.names=T)
```



Run the model:

```
> h6 <- hansen(log(bimac['size']), tree, regimes=ou.clades, sqrt.alpha=1, sigma=1)
```

Rebuild our table and compare models:

```
> h <- append(h, h6)           # append (add on) new model results to our list h
> names(h)[length(h)] <- "OU.clades" # add the name of the new model
> names(h)
```

```
[1] "BM"           "OU.1"         "OU.3"         "OU.4"         "OU.LP"        "OU.clades"
```

```
> h.ic <- sapply(h, function(x) unlist(summary(x)[c('aic', 'aic.c', 'sic', 'dof')]))
> print(h.ic, digits = 3)
```

```
      BM  OU.1  OU.3  OU.4  OU.LP  OU.clades
aic -30.7 -25.4 -29.2 -35.2 -39.6    -30.7
```

aic.c	-30.1	-24.1	-25.6	-30.0	-36.1	-27.1
sic	-28.4	-22.0	-23.5	-28.4	-34.0	-25.0
dof	2.0	3.0	5.0	6.0	5.0	5.0



# Chapter 3

## Bivariate ouch

The `ouch` package has been completely rewritten by Aaron King to implement a bivariate model, as well as the new S4 class system described previously.

Correlated evolution is a major feature of evolutionary theory, and of great interest among comparative biologists. However, there have been few attempts to develop a bivariate OU model for comparative analysis. NOTE: We are about to submit a paper on this, please cite us (and wait for us to publish first!).

### 3.0.6 The Bivariate model

The Hansen model describes the evolutionary processes operative on a single quantitative character Hansen (1997); Butler and King (2004). In the case of two characters, we will accordingly have two equations.

$$dX_1(t) = \alpha_1 (\theta_1 - X_1(t)) dt + \sigma_1 dB_1(t). \quad (3.1)$$

$$dX_2(t) = \alpha_2 (\theta_2 - X_2(t)) dt + \sigma_2 dB_2(t). \quad (3.2)$$

The above system of eqs. 3.3 can be written in matrix form with vectors in the place of  $dX$ ,  $X$ , and  $dB(t)$ , and square matrices in place of  $\alpha$  and  $\sigma$ . The  $\theta$  are already vector valued in the univariate case with a single value per adaptive regime. Here we simply have separate  $\theta$  vectors for each character.

### 3.0.7 No Correlations

When evolution is uncorrelated, the  $\alpha$  and  $\sigma$  matrices are diagonal:

$$\alpha = \begin{pmatrix} \alpha_{11} & 0 \\ 0 & \alpha_{22} \end{pmatrix} \quad \sigma = \begin{pmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{22} \end{pmatrix}.$$

This form implies that neither character influences the evolution of the other (i.e., they are evolving independently of one another, and we have a simple duplication of the univariate case).

### 3.1 Correlated Evolution

We can readily see, however, that the matrices may have off-diagonal elements. These evolutionary correlations can enter in as off-diagonal terms in either the  $\alpha$  or the  $\sigma$  terms. In particular, they will have the following form:

$$\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{12} & \alpha_{22} \end{pmatrix} \quad \sigma = \begin{pmatrix} \sigma_{11} & 0 \\ \sigma_{12} & \sigma_{22} \end{pmatrix}.$$

Where  $\alpha_{12} = \alpha_{21}$ , and without loss of generality,  $\sigma$  is lower-triangular.

Written out as separate equations, the model has the following form:

$$\begin{aligned} dX_1(t) &= \alpha_{11} (\theta_1 - X_1(t)) dt + \alpha_{12} (\theta_2 - X_2(t)) dt + \sigma_{11} dB_1(t). \\ dX_2(t) &= \alpha_{22} (\theta_2 - X_2(t)) dt + \alpha_{12} (\theta_1 - X_1(t)) dt + \sigma_{22} dB_2(t) + \sigma_{12} dB_1(t). \end{aligned}$$

### 3.2 Implementation in ouch

To illustrate, we reanalyzed the evolution of sexual size dimorphism in association with habitat specialization in *Anolis* lizards ([Butler and King, 2004](#)), reformulated as evolution in male and female body size.

Load the data (tree+quantitative data) and regimes:

```
> require(ouch)
> regimes <- read.csv("Rdata/regimes.csv", row.names = 1)
> ssd <- read.csv("Rdata/ssd.data.csv", row.names = 1)
> otree <- with(ssd, ouchtree(nodes, ancestors, times, labels))
> xdata <- log(ssd[c("fSVL", "mSVL")])
> names(xdata) <- paste("log", names(xdata), sep = ".")
> nreg <- length(regimes)
```

ouch now requires you to specify an initial guess for the alpha and sigma matrices. Univariate models are specified by providing a single value. **The bivariate model is specified by providing multiple values for these guesses. The three element vector will be transformed into a symmetric 2x2 matrix for alpha and a lower-triangular matrix for sigma.**

```
> alpha.guess <- c(1, 0, 1)
> sigma.guess <- c(1, 1, 1)
```

Fit the model for the first regime:

```
> tic <- Sys.time()
> hansen(data = xdata, tree = otree, regimes = regimes[5], sqrt.alpha = alpha.guess,
+   sigma = sigma.guess, method = "Nelder-Mead", maxit = 3000,
+   reltol = 1e-12)
```

call:

```
hansen(data = xdata, tree = otree, regimes = regimes[5], sqrt.alpha = alpha.guess,
  sigma = sigma.guess, method = "Nelder-Mead", maxit = 3000,
  reltol = 1e-12)
```

	nodes	ancestors	times	labels	TW.6	TW.6.1	log.fSVL
1	1	<NA>	0.00		twig	twig	NA
2	2	3	0.62	trunk-crown	trunk-crown		NA
3	3	4	0.37	trunk-crown	trunk-crown		NA
4	4	5	0.14	crown-giant	crown-giant		NA
5	5	1	0.08		twig	twig	NA
6	6	8	0.65	trunk-ground	trunk-ground		NA
7	7	8	0.61	trunk-ground	trunk-ground		NA
8	8	10	0.54	trunk-ground	trunk-ground		NA
9	9	10	0.65	grass-bush	grass-bush		NA
10	10	12	0.50	trunk-ground	trunk-ground		NA
11	11	12	0.79	trunk-crown	trunk-crown		NA
12	12	14	0.43	trunk-ground	trunk-ground		NA
13	13	14	0.63	trunk	trunk		NA
14	14	23	0.29	trunk-ground	trunk-ground		NA
15	15	16	0.78	trunk-ground	trunk-ground		NA
16	16	17	0.57	trunk-ground	trunk-ground		NA
17	17	22	0.36	trunk-ground	trunk-ground		NA
18	18	19	0.77	trunk-crown	trunk-crown		NA
19	19	20	0.72	trunk-crown	trunk-crown		NA
20	20	21	0.51	trunk-crown	trunk-crown		NA
21	21	22	0.46	trunk-ground	trunk-ground		NA
22	22	23	0.27	trunk-ground	trunk-ground		NA

23	23	28	0.14	trunk-ground	trunk-ground	NA
24	24	124	0.60	trunk-crown	trunk-crown	NA
124	124	25	0.47	trunk-crown	trunk-crown	NA
25	25	26	0.41	trunk-crown	trunk-crown	NA
26	26	27	0.33	twig	twig	NA
27	27	28	0.23	twig	twig	NA
28	28	34	0.11	twig	twig	NA
29	29	145	0.40	crown-giant	crown-giant	NA
145	145	34	0.28	crown-giant	crown-giant	NA
30	30	31	0.38	grass-bush	grass-bush	NA
31	31	34	0.22	twig	twig	NA
32	32	33	0.77	trunk-ground	trunk-ground	NA
33	33	161	0.72	trunk-ground	trunk-ground	NA
161	161	34	0.44	trunk-ground	trunk-ground	NA
34	34	1	0.08	twig	twig	NA
36	36	2	1.00	A_aliniger	trunk-crown	trunk-crown 3.815512
62	62	24	1.00	A_allisoni	trunk-crown	trunk-crown 4.100989
54	54	17	1.00	A_allogus	trunk-ground	trunk-ground 3.749504
64	64	134	1.00	A_alutaceu	grass-bush	grass-bush 3.487375
134	134	27	0.54		grass-bush	grass-bush NA
60	60	110	1.00	A_angustic	twig	twig 3.653252
110	110	26	0.40		twig	twig NA
49	49	13	1.00	A_breviros	trunk	trunk 3.693867
37	37	2	1.00	A_chlorocy	trunk-crown	trunk-crown 3.992681
38	38	3	1.00	A_coelesti	trunk-crown	trunk-crown 4.000034
44	44	7	1.00	A_cooki	trunk-ground	trunk-ground 3.728100
43	43	7	1.00	A_cristate	trunk-ground	trunk-ground 3.797734
66	66	29	1.00	A_cuvieri	crown-giant	crown-giant 4.782479
70	70	32	1.00	A_cybotes	trunk-ground	trunk-ground 3.927896
50	50	13	1.00	A_distichu	trunk	trunk 3.797734
39	39	413	1.00	A_equestri	crown-giant	crown-giant 5.045359
413	413	4	0.90		crown-giant	crown-giant NA
48	48	11	1.00	A_evermann	trunk-crown	trunk-crown 3.958907
56	56	18	1.00	A_garmani	crown-giant	crown-giant 4.412798
57	57	19	1.00	A_grahami	trunk-crown	trunk-crown 3.784190
42	42	6	1.00	A_gundlach	trunk-ground	trunk-ground 3.811097
35	35	403	1.00	A_henderso	grass-bush	grass-bush 3.693867
403	403	4	0.58		grass-bush	grass-bush NA
53	53	16	1.00	A_homolech	trunk-ground	trunk-ground 3.706228
69	69	150	1.00	A_insolitu	twig	twig 3.676301
150	150	31	0.65		twig	twig NA
46	46	9	1.00	A_krugi	grass-bush	grass-bush 3.671225
59	59	101	1.00	A_lineatop	trunk-ground	trunk-ground 3.788725

101	101	21	0.62		trunk-ground	trunk-ground	NA
61	61	113	1.00	A_loysiana	trunk	trunk	3.575151
113	113	25	0.59		trunk	trunk	NA
40	40	5	1.00	A_occultus	twig	twig	3.668677
68	68	30	1.00	A_olssoni	grass-bush	grass-bush	3.703768
55	55	18	1.00	A_opalinus	trunk-crown	trunk-crown	3.701302
52	52	15	1.00	A_ophiolep	grass-bush	grass-bush	3.440418
41	41	6	1.00	A_poncensi	grass-bush	grass-bush	3.678829
63	63	24	1.00	A_porcatus	trunk-crown	trunk-crown	3.998201
45	45	9	1.00	A_pulchell	grass-bush	grass-bush	3.610918
65	65	29	1.00	A_ricordii	crown-giant	crown-giant	4.933754
51	51	15	1.00	A_sagrei	trunk-ground	trunk-ground	3.688879
67	67	30	1.00	A_semiline	grass-bush	grass-bush	3.616309
71	71	32	1.00	A_shrevei	trunk-ground	trunk-ground	3.832980
47	47	11	1.00	A_stratulu	trunk-crown	trunk-crown	3.686376
58	58	20	1.00	A_valencie	twig	twig	4.226834
72	72	33	1.00	A_whiteman	trunk-ground	trunk-ground	3.873282

log.mSVL

1	NA
2	NA
3	NA
4	NA
5	NA
6	NA
7	NA
8	NA
9	NA
10	NA
11	NA
12	NA
13	NA
14	NA
15	NA
16	NA
17	NA
18	NA
19	NA
20	NA
21	NA
22	NA
23	NA
24	NA
124	NA

25	NA
26	NA
27	NA
28	NA
29	NA
145	NA
30	NA
31	NA
32	NA
33	NA
161	NA
34	NA
36	4.063885
62	4.382027
54	4.030695
64	3.569533
134	NA
60	3.788725
110	NA
49	3.864931
37	4.276666
38	4.247066
44	4.085976
43	4.195697
66	4.877485
70	4.188138
50	3.935740
39	5.129899
413	NA
48	4.258446
56	4.700480
57	4.182050
42	4.171306
35	3.869116
403	NA
53	3.958907
69	3.737670
150	NA
46	3.906005
59	4.177459
101	NA
61	3.706228
113	NA

```

40  3.663562
68  3.802208
55  3.901973
52  3.600048
41  3.819908
63  4.261270
45  3.848018
65  5.023222
51  3.977811
67  3.728100
71  4.001864
47  3.843744
58  4.374498
72  4.082609

```

alpha:

```

      [,1]      [,2]
[1,] 3.2885637 0.2651125
[2,] 0.2651125 5.0304925

```

sigma squared:

```

      [,1]      [,2]
[1,] 0.1164814 0.1606944
[2,] 0.1606944 0.2412312

```

theta:

\$log.fSVL

crown-giant	grass-bush	trunk	trunk-crown	trunk-ground	twig
4.966605	3.605634	3.678655	3.912812	3.772731	3.798625

\$log.mSVL

crown-giant	grass-bush	trunk	trunk-crown	trunk-ground	twig
5.003842	3.763422	3.832975	4.171052	4.076491	3.874264

loglik	deviance	aic	aic.c	sic	dof
82.05319	-164.10638	-128.10638	-116.10638	-86.15318	18.00000

```

> toc <- Sys.time()
> print(toc - tic)

```

Time difference of 9.833533 secs

Fitting the Hansen model is much more complex than the Brownian motion, because the  $\alpha$  enters non-linearly into the likelihood function. With more variables, the complexity

increases, with a less well-behaved likelihood surface than the univariate case. There are frequently convergence issues.

Using the subplex method from package `subplex` helps. Other things to try include increasing the tolerance, increasing the number of maximum iterations allowed to reach convergence, and drawing initial guesses for alpha and sigma at random (and discarding the bad guesses). This of course increases computer time.

```
> tic <- Sys.time()
> h.subplex <- hansen(data = xdata, tree = otree, regimes = regimes[1],
+   sqrt.alpha = alpha.guess, sigma = sigma.guess, method = "subplex",
+   maxit = 20000, reltol = 1e-12)
> toc <- Sys.time()
> print(toc - tic)
```

Time difference of 1.243227 mins

```
> summary(h.subplex)
```

\$call

```
hansen(data = xdata, tree = otree, regimes = regimes[1], sqrt.alpha = alpha.guess,
  sigma = sigma.guess, method = "subplex", maxit = 20000, reltol = 1e-12)
```

\$conv.code

```
[1] 0
```

\$optimizer.message

```
NULL
```

\$alpha

```
      [,1]      [,2]
[1,] 3.0345742 0.7042747
[2,] 0.7042747 4.3493339
```

\$sigma.squared

```
      [,1]      [,2]
[1,] 0.1232642 0.1585031
[2,] 0.1585031 0.2201614
```

\$optima

\$optima\$log.fSVL

ancestral	crown-giant	grass-bush	trunk	trunk-crown	trunk-ground
4.688810	4.957811	3.588247	3.657690	3.888971	3.758807



```

      twig
      3.777041

$optima$log.mSVL
      ancestral  crown-giant  grass-bush      trunk  trunk-crown trunk-ground
      6.001253    4.959877    3.710175    3.775685    4.111575    4.050079
      twig
      3.807817

$loglik
[1] 82.4903

$deviance
[1] -164.9806

$aic
[1] -124.9806

$aic.c
[1] -109.7079

$sic
[1] -78.36594

$dof
[1] 20

The Brownian motion model is fit:

> brown.fit <- brown(data = xdata, tree = otree)
> summary(brown.fit)

$call
brown(data = xdata, tree = otree)

$sigma.squared
      [,1]      [,2]
[1,] 0.1523957 0.1564986
[2,] 0.1564986 0.1746020

$theta
$theta$log.fSVL

```

```
[1] 3.954663
```

```
$theta$log.mSVL
```

```
[1] 4.118798
```

```
$loglik
```

```
[1] 22.56325
```

```
$deviance
```

```
[1] -45.12651
```

```
$aic
```

```
[1] -35.12651
```

```
$aic.c
```

```
[1] -34.26937
```

```
$sic
```

```
[1] -23.47284
```

```
$dof
```

```
[1] 5
```

### 3.3 Exercises

1. Run the Hansen model on the remaining regimes. Can you use an apply method to run them all at once?
2. Plot the multiple regime hypotheses.
3. Compare results.

### 3.4 Variations of the OU Model

Instead of assuming a constant  $\sigma$  across the entire tree, [O'Meara et al. \(2006\)](#) developed a Brownian motion model that allows two or more  $\sigma$  values. This can be interpreted as having different rates of evolution in different regions of the tree.

Other possibilities exist. The difficulty for the future will be twofold: (1) Ensuring that the complexity of the model is reasonable given the information content of the data (i.e.,

are the parameter estimates and likelihoods well-behaved?). (2) Thinking hard about the best evolutionary and biological interpretations of the models.



# Bibliography

- Butler, M. A. and A. A. King. 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. *American Naturalist*, **164**:683–695. URL <http://www.journals.uchicago.edu/AN/journal/issues/v164n6/40201/40201.html>.
- Felsenstein, J. 1988. Phylogenies and quantitative characters. *Annual Review of Ecology and Systematics*, **19**:445–471. URL <http://links.jstor.org/sici?sici=0066-4162%281988%2919%3C445%3APAQC%3E2.0.CO%3B2-I>.
- Hansen, T. F. 1997. Stabilizing selection and the comparative analysis of adaptation. *Evolution*, **51**:1341–1351.
- Lande, R. 1976. Natural selection and random genetic drift in phenotypic evolution. *Evolution*, **30**:314–334.
- O’Meara, B. C., C. Ane, M. J. Sanderson, and P. C. Wainwright. 2006. Testing for different rates of continuous trait evolution using likelihood. *Evolution*, **60**:922–933.
- Simpson, G. G. 1953. *The Major Features of Evolution*. Columbia University Press.
- Uhlenbeck, G. E. and L. S. Ornstein. 1930. On the theory of brownian motion. *Physical Review*, **36**:823–841.