Calculating the conditional probability of a node under the hOUwie model

Our goal is to calculate the conditional probability of a particular discrete ancestral state ($P(D_{internal} = d_s)$) given the tips below that node have particular discrete states (d_s) and continuous values (x). For this example, we will say that there are two states for the discrete character $D = d_{s=1}$ *or* $d_{s=2}$, but this calculation can apply to any number of discrete state and character. X can take on any continuous value.

We use an approximation to calculate this probability. Specifically, for a particular node, n, we will separate all descendants into *leftward* (l) and *rightward* (r). The choice of leftward or rightward is arbitrary and can be switched without consequence. Next, we will calculate the conditional probability of the node given only a single pair of tips, one chosen from the leftward set and the other chosen from the rightward tip. This process will be repeated for all pairs.

The probability that the internal value takes on a particular discrete state is the probability of observing the discrete and continuous values for the given leftward and rightward pair:

$$P(d_{internal}) = P(d_l|d_{internal}) * P(x_l|d_{internal}, d_l, x_{internal}) * P(d_r|d_{internal}) * P(x_r|d_{internal}, d_r, x_{internal})$$

We use the standard Chapman-Kolmgorov equation to calculate $P(d_l|d_{internal})$ and $P(d_r|d_{internal}$ where we start in state $d_{internal}$ and end in state d_l or d_r for a given length of time defined by the branch length from the internal node to the tip.

```
Q <- matrix(c(-1,1,1,-1), 2, 2)
branch.length <- 10
internal_d <- 1
external_d <- 2
probability <- expm:::expm(Q * branch.length)[internal_d, external_d]
print(probability)
```

```
## [1] 0.5
```

Calculating $P(x_l|d_{internal}, d_l)$ and $P(x_r|d_{internal}, d_r)$ is more difficult because the internal value of the continuous character can take on any continuous value. Rather than integrating over all possible values, the ancestral continuous value is set to be equal to the expected value for a given discrete state. I.e., $x_{internal} = \theta_{d_{internal}}$ where θ is defined by the model parameters. Next we will calculate the expected value and variance for the tip given the model parameters θ, α, and σ^2 and branch length from node to tip. We say that x_l is normally distributed with a mean equal to the expected value at the tip and variance at the tip. The expected values and variances are calculated following Beaulieu et al. (2012).

```
# given parameters
branch_length <- 1
times <- c(0, branch_length/2, branch_length) # see Beaulieu et al. (2012)
theta_1 <- 5
theta_2 <- 10
alpha_1 <- alpha_2 <- 2
sigma2_1 <- sigma2_2 <- 1
# tip data
intneral_d <- 1
```

```
internal_x <- theta_1 # theta 1 because the current internal discrete state is 1
external_d <- 2
external_x <- 7.7
tip_weight <- exp(-((alpha_1 * (times[2] - times[1])) + (alpha_2 * (times[3] - times[2]))))
theta_1_weight <- tip_weight * (exp(alpha_1 * times[2]) - exp(alpha_1 * times[1]))
theta_2_weight <- tip_weight * (exp(alpha_2 * times[3]) - exp(alpha_2 * times[2]))
weights <- c(tip_weight, theta_1_weight, theta_2_weight)/sum(c(tip_weight, theta_1_weight, theta_2_weigh
expected_value <- sum(c(internal_x, theta_1, theta_2) * weights)
var_weight <- exp(-((2 * alpha_1 * (times[2] - times[1])) + (2 * alpha_2 * (times[2] - times[1]))))
var_1 <- sigma2_1/(2*alpha_1) * (exp(2 * alpha_1 * times[2]) - exp(2 * alpha_1 * times[1]))
var_2 <- sigma2_2/(2*alpha_2) * (exp(2 * alpha_2 * times[3]) - exp(2 * alpha_2 * times[2]))
variance <- (sum(c(var_1, var_2)) * var_weight)
# expected value and variance
print(c(expected_value=expected_value))
```

```
## expected_value
##       8.160603
```

```
print(c(variance = variance))
```

```
##  variance
## 0.2454211
```

```
# calculate the probability
probability <- dnorm(external_x, expected_value, sqrt(variance))
print(c(probability = probability))
```

```
## probability
##   0.5226857
```

This calculation is repeated for the leftward and rightward branches and multiply them together to find $P(d_{internal})$. This is repeated for all possible $d_{internal}$ and for all possible leftward and rightward pairs (See Figure 3 of the main text). Finally, our adaptive sampling algorithm works by using a map generated based on the above calculation to generate a new set of expectations at the internal node, rather than assuming that $x_{internal} = \theta_{d_{internal}}$.