Appendix B - Note about the incorrectness of discrete path calculations

James D. Boyko

09/07/2021

Introduction

This appendix will demonstrate that the simple calculation of the probability of a path presented in May and Moore (2020) fails to account for probability of a change overall. For an equal rates model with the MLE of zero changes, any path of 0, 2, or 100 changes has exactly the same likelihood. We suspect this is an issue of treating the pdfs independently when their convolution is actually required (the time of the second transition depends on the timing of the first and the total branch length available). Proving this is left as an exercise to the reader. Instead, we will demonstrate: (1) problematic behavior of May and Moore (2020)'s calculation and (2) our approach of integration over all paths between nodes on the tree, after first adding additional degree 2 nodes behaves in a way that is appropriate.

Incorrect calculation of joint probabilities

First it is important to understand that when calculating the probability of a particular mapping we are finding the joint probability of the data and a particular mapping $P(M_a, D)$. This is in contrast to usual implementations of Markov models (such as those in corHMM), which calculate the marginal probability of a dataset $P(D)$. This distinction is important because the marginal probability of a dataset can be found only by summing over every possible reconstruction $P(D) = \sum_{a=1}^{\infty} P(M_a, D)$ which is efficiently calculated using the pruning algorithm and the transition rate matrix. Therefore, we know with certainty that $P(M_a, D) \leq P(D)$. In this section we demonstrate that calculating the probability in the same way as May and Moore 2020:

$$P(M_a, D|Q, \psi) \;=\; P(x_0|Q, \psi) \prod_{\ell=1}^{2n-2} P(z_\ell|Q,\; T_\ell)$$

does not satisfy this property.

First, we introduce our functions for calculating the probability of a particular stochastic mapping. We leave it to the reader to check the validity of the implementation below. Briefly, for a given path, we calculate the probability of a transition occurring after a particular waiting time as an exponentially distributed random variable and at the end of the branch the final waiting time is the probability of no transition occurring.

```
# the probability of a given simmap is the product of all it's individual branches.
getMapProbability <- function(simmap, Q){
  BranchProbs <- lapply(simmap$maps, function(x) probPath(x, Q))
  LnLik_map <- sum(unlist(BranchProbs))
  return(LnLik_map)
}

# the probability of a particular path
probPath <- function(path, Q){
```

```
  nTrans <- length(path)
  P <- vector("numeric", length(path))
  for(i in sequence(nTrans-1)){
    state_i <- as.numeric(names(path)[1])
    state_j <- as.numeric(names(path)[2])
    time_i <- as.numeric(path[1])
    rate_i <- abs(Q[state_i,state_j])
    P[i] <- dexp(time_i, rate_i)
    path <- path[-1]
  }
  state_j <- as.numeric(names(path))
  time_j <- as.numeric(path)
  rate_j <- abs(Q[state_j,state_j])
  P[nTrans] <- 1 - pexp(rate_j, time_j)
  P[P == 0] <- 1e-300
  P <- sum(log(P))
  return(P)
}
```

Create a dataset to evaluate.

```
require(geiger)
require(corHMM)
phy <- sim.bdtree(b = 1, d = 0, stop = "taxa", n = 100)
phy <- drop.extinct(phy)
phy$edge.length <- phy$edge.length/max(branching.times(phy))
rate <- 0.5
Q <- matrix(c(-rate, rate,rate,-rate),2,2)
data <- sim.char(phy, Q, 1, "discrete")[,,1]
dat <- data.frame(sp = names(data), d = data)
print(head(dat))
```

```
##    sp d
## s1 s1 1
## s2 s2 1
## s3 s3 1
## s4 s4 1
## s5 s5 2
## s6 s6 2
```

We can calculate the probability of this dataset under and equal rates model with rate=0.5 using the function corHMM.

```
corhmm_res <- corHMM(phy, dat, model = "ER", p = 0.5, rate.cat=1)
```

```
## State distribution in data:
## States:  1   2
## Counts:  84  16
## Calculating likelihood from a set of fixed parameters
## Finished. Inferring ancestral states using marginal reconstruction.
```

```
print(c(corHMM_likelihood = corhmm_res$loglik))
```

```
## corHMM_likelihood
##         -32.73508
```

We can now compare that likelihood to a particular path probability which we will generate by simulating

a stochastic map and evaluating its probability at a rate of 0.5 under the same equal rates model. Let's evaluate 5 stochastic maps to see whether any of them are less likely than what we found with corHMM.

```
simmap <- makeSimmap(tree=phy, data=dat, model=Q, rate.cat=1, nSim=5, nCores=1)
simmap_llik <- unlist(lapply(simmap, function(x) getMapProbability(x, Q)))
names(simmap_llik) <- paste0("Map", 1:5, "_llik")
print(c(corHMM_likelihood = corhmm_res$loglik, simmap_llik))
```

```
## corHMM_likelihood         Map1_llik         Map2_llik         Map3_llik
##         -32.73508         -21.39870         -22.78500         -22.78500
##         Map4_llik         Map5_llik
##         -21.39870         -22.09185
```

All of the map probabalities are greater than the probability of the dataset. This would mean to marginilize over every possible map would result in value much greater than 1 for the dataset. In the next section we will take a more detailed approach and examine these probabilities on a path by path basis.

The number of transitions should effect the probability of a path

When calculating the probability according to the equation presented in May and Moore (2020), we find that the number of transitions that occur along a branch has no effect on the probability of a path (we believe this can help explain some of the odd behavior we discussed in Appendix A).

First we calculate the probability of several pathways according to May and Moore (2020). We find that no matter how many transitions occur the probabilities of the branch are unchanging.

```
# The probability of no change along a branch of length 1 with rate of 1
## 00
print(1 - pexp(1, 1))
```

```
## [1] 0.3678794
```

```
# The probability of 1 change along a branch of length 1 with rate of 1
## 010
print(dexp(1/3, 1) * dexp(1/3, 1)* (1 - pexp(1/3, 1)))
```

```
## [1] 0.3678794
```

```
# The probability of 2 changes along a branch of length 1 with rate of 1
# 01010
print(dexp(1/5, 1) * dexp(1/5, 1) * dexp(1/5, 1) * dexp(1/5, 1) * (1 - pexp(1/5, 1)))
```

```
## [1] 0.3678794
```

```
# etc...
```

This is counter-intuitive but that does not necessarily mean it is incorrect. To demonstrate that it is, let us first examine what the probability of a branch starting in state 0 and ending in state 0.

```
library(expm)
Prob <- function(t, start=0, end=0, Q=matrix(c(-1,1,1,-1), nrow=2)) {
    return(expm(Q*t)[start+1, end+1])
}
print(c("P00_all_possible_paths"=Prob(1, start=0, end=0)))
```

```
## P00_all_possible_paths
##              0.5676676
```

This is actually somewhat positive for the pathway calculations. This is the marginal probability of the path and it is greater than the joint probabilities above. Although we can see that if we were to sum over every

path probability above (00, 010, 01010, ...) the probability would quickly exceed 1 instead of approaching the correct probability of 0.568. The actual probability that approaches this is a summation of a poisson distributed random variable of the number of transitions that are required to start in end in particular states.

```
pois <- c("00"=dpois(0, 1), "010"= dpois(2, 1), "01010"= dpois(4, 1), "0101010"= dpois(6, 1))
print(pois)
```

```
##           00          010        01010      0101010
## 0.3678794412 0.1839397206 0.0153283100 0.0005109437
```

```
print(c("P00_sum_pois_paths"=sum(pois), "P00_all_possible_paths"=Prob(1, start=0, end=0)))
```

```
##     P00_sum_pois_paths P00_all_possible_paths
##              0.5676584              0.5676676
```

We can see that the more transitions occur, the less likely a particular path is. Additionally, we find that this number converges to the correct probability of integrating over all possible paths and therefore represents the correct joint probability.

Now we examine the behavior of our inter node approach to pathway probability. Does it add up the sum of all possible paths?

```
# 00
print(Prob(1, start=0, end=0))
```

```
## [1] 0.5676676
```

```
# 010
print(Prob(0.5, start=0, end=1) * Prob(0.5, start=1, end=0))
```

```
## [1] 0.0998941
```

```
# 000
print(Prob(0.5, start=0, end=0) * Prob(0.5, start=0, end=0))
```

```
## [1] 0.4677735
```

```
# 010 + 000
print(Prob(0.5, start=0, end=0) * Prob(0.5, start=0, end=0) + Prob(0.5, start=0, end=1) * Prob(0.5, sta
```

```
## [1] 0.5676676
```

```
# c(0000, 0010, 0100, 0110)
df <- data.frame(a0000=Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=0
    a0010=Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=0) ,
    a0100=Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=0) * Prob(1/3, start=0, end=0) ,
    a0110=Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=1) * Prob(1/3, start=1, end=0) )
print(round(df,3))
```

```
##   a0000 a0010 a0100 a0110
## 1 0.433 0.045 0.045 0.045
```

```
# 0000 + 0010 + 0100 + 0110
print(
    Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=0) +
    Prob(1/3, start=0, end=0) * Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=0) +
    Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=0) * Prob(1/3, start=0, end=0) +
    Prob(1/3, start=0, end=1) * Prob(1/3, start=1, end=1) * Prob(1/3, start=1, end=0)
)
```

```
## [1] 0.5676676
```

Indeed, it has the desired behaviour.