Appendix A - discrete rate upper bound set to 1

As we discuss in the paper we calculate the probability of a discrete map by calculating the probability of a series of independent waiting times along each branch. However, we have discovered that there is what is best described as a phase transition at $q = 1$. Before digging deeper into the explanation let's demonstrate the problem in a practical situation.

First we'll define a pair of functions: ProbPath (the probability of a given path and rate matrix)

```
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 <- sum(log(P))
  return(P)
}
```

and getMapProbability (the probabity of a particular stochastic mapping).

```
getMapProbability <- function(maps, Q){
  BranchProbs <- lapply(maps, function(x) probPath(x, Q))
  LnLik_map <- sum(unlist(BranchProbs))
  return(LnLik_map)
}
```

Let's evaluate these functions on a primates dataset included with corHMM.

```
require(corHMM)
```

```
## Loading required package: corHMM
```

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

```
## Loading required package: nloptr
```

```
## Loading required package: GenSA
```

```
require(nloptr)
data(primates)
phy <- primates[[1]]
phy <- multi2di(phy)
data <- primates[[2]]
```

```
dat <- data.frame(sp = data[,1], d = rowSums(data[,c(2,3)]))
##run corhmm
MK <- corHMM(phy, dat, 1, model = "ER")
```

```
## State distribution in data:
## States:  1   2   3
## Counts:  29  10  21
## Beginning thorough optimization search -- performing 0 random restarts
## Finished. Inferring ancestral states using marginal reconstruction.
```

```
##get simmap from corhmm solution
model <- MK$solution
simmap <- makeSimmap(tree=phy, data=data, model=model, rate.cat=1, nSim=1, nCores=1)
Q <- MK$solution
Q[is.na(MK$solution)] <- 0
diag(Q) <- -rowSums(Q)
MapProb <- getMapProbability(simmap[[1]]$maps, Q)
```

```
print(c(LnLikCorHMM = MK$loglik, LnLikMapping = MapProb))
```

```
##  LnLikCorHMM LnLikMapping
##    -45.60640    -89.70003
```

We can see that a particular mapping is less likely than the corHMM likelihood which makes intuitivee sense since a corHMM likelihood integrates over many such mappings.

However, in hOUwie, we are not given any particular mapping. Rather, given a particular rate, we generate stochastic maps and then evaluate the likleihood of those stochastic maps given our rate parameter and continuous model. So, to examine that behaviour we will have to optimize a discrete dataset where stochastic maps are generated and fit.

The function we'll be using as the optimization is optMap. Given a set of parameters it will generate a set of stochastic maps, evaluate their likelihood under that model, then return the maximum of those values.

```
optMap <- function(p, phy, dat, nMap){
  model <- getRateCatMat(3)
  model[model > 0] <- p
  diag(model) <- -rowSums(model)
  simmap <- makeSimmap(tree=phy, data=dat, model=model, rate.cat=1, nSim=nMap, nCores=1)
  maps <- lapply(simmap, function(x) x$maps)
  llik <- max(unlist(lapply(maps, function(x) getMapProbability(x, model))))
  return(-llik)
}
```

Set x0 (the intial parameter) to be 0.1.

```
opts <- list("algorithm"="NLOPT_LN_SBPLX", "maxeval"="1000000", "ftol_rel"=.Machine$double.eps^0.5)
# with 0.1 we're good
set.seed(1)
out <- nloptr(x0 = 0.1, eval_f = optMap, lb = 1e-5, ub = 100, opts = opts, phy = phy, nMap = 10, dat = d
```

```
c(OptimMapLnLik = -out$objective, RateMLEMap = out$solution, RateMLEcorHMM = Q[2,1])
```

```
## OptimMapLnLik    RateMLEMap RateMLEcorHMM
## -70.656767937   0.007234314   0.009199781
```

If we start at a rate of 0.1, we find an optimization that was not unlike when our rates were estimated by corHMM.

However, this has occurred because the optimizer was stuck on the left side of a phase transition which begins at a rate of 1. If we set x0 to a higher value, we should find a very different result. Rather than running the optimizer though, let's look at 3 specific likelihoods rate at the phase transition: p = 1, 1.1, and 2

```
p1 <- -optMap(1, phy, dat, 10)
p1.1 <- -optMap(1.1, phy, dat, 10)
p2 <- -optMap(2, phy, dat, 10)
```

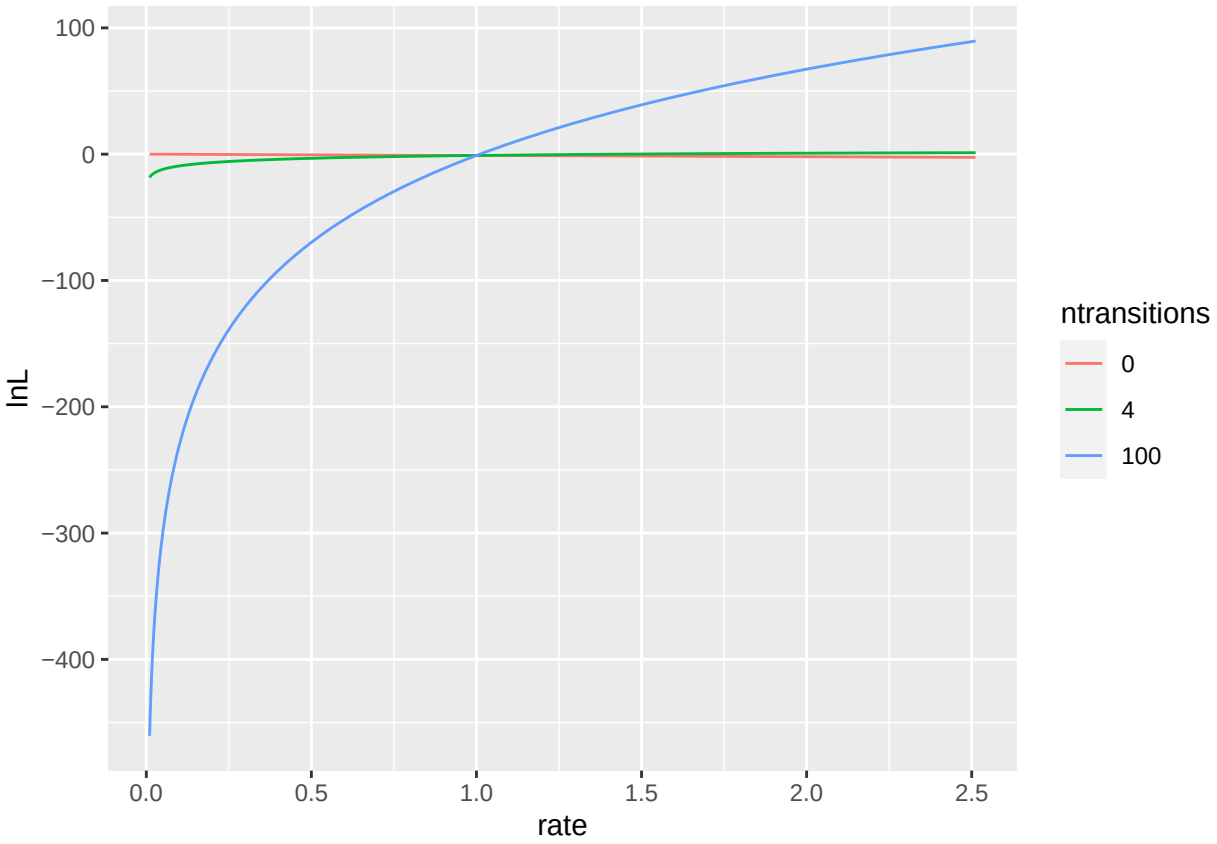
```
c(Rate_1 = p1, Rate_1.1 = p1.1, Rate_2 = p2)
```

```
##    Rate_1  Rate_1.1    Rate_2
## -889.6221 -789.9766  651.6008
```

Here we see that likelihood estimates directly around 1 are actually quite poor (much lower than when we optimized). But, as we get higher rate values we begin to see likelihoods that are exceptionally large. This is the essence of the phase transition problem. As rates increase past one the number of transitions and most likely transition rate both go to 1.

To get a better understanding of this let's examine what this looks like on a single branch. This code modified from Brian O'Meara.

```
likelihood_given_number_transitions <- function(ntransitions=1, rate=0.1, time=1) {
  timeinterval <- time/(ntransitions+1)
  return( ((rate*exp(-rate*timeinterval))^ntransitions) * (exp(-rate*timeinterval)))
}
all_values <- expand.grid(ntransitions=c(0,4,100), rate=c(10^seq(from=-2, to=.4, length.out=100)))
all_values <- all_values[order(all_values$ntransitions),]
all_values$likelihood <- NA
for (i in sequence(nrow(all_values))) {
  all_values$likelihood[i] <- likelihood_given_number_transitions(all_values$ntransitions[i], all_value
}
# print(all_values[which.max(all_values$likelihood),])
all_values$lnL <- log(all_values$likelihood)
all_values$ntransitions <- as.factor(all_values$ntransitions)
library(ggplot2)
ggplot(all_values, aes(x=rate, y=lnL, group=ntransitions)) + geom_line(aes(color=ntransitions))
```



This plot has the likelihood (y axis) of a rate (x axis) for a given number of transitions along a branch (colour). The branch starts in state 0 and ends in state 0 so intuitively we would expect that a rate of 0 is the maximum probability (no change is necessary). And this is true so long as rates are below 1 - the maximum likelihood value is a rate of 0 with 0 transitions. And in general, the 0 transitions likelihood line its above multiple transition competitors until rate = 1. At rate = 1, the probability suddenly shifts.

One interesting thing to note is that at a rate of 1, any number of transitions will occur at the same probability (intersection point on the graph above).

This property comes from the fact that the MLE for the number of regimes grows exponentially if rate > 1. The PDF of the exponential distribution is:

$$\lambda e^{-\lambda x}$$

If lambda = 1, then no matter the number of transitions the probability will be:

$$e^{-x}$$

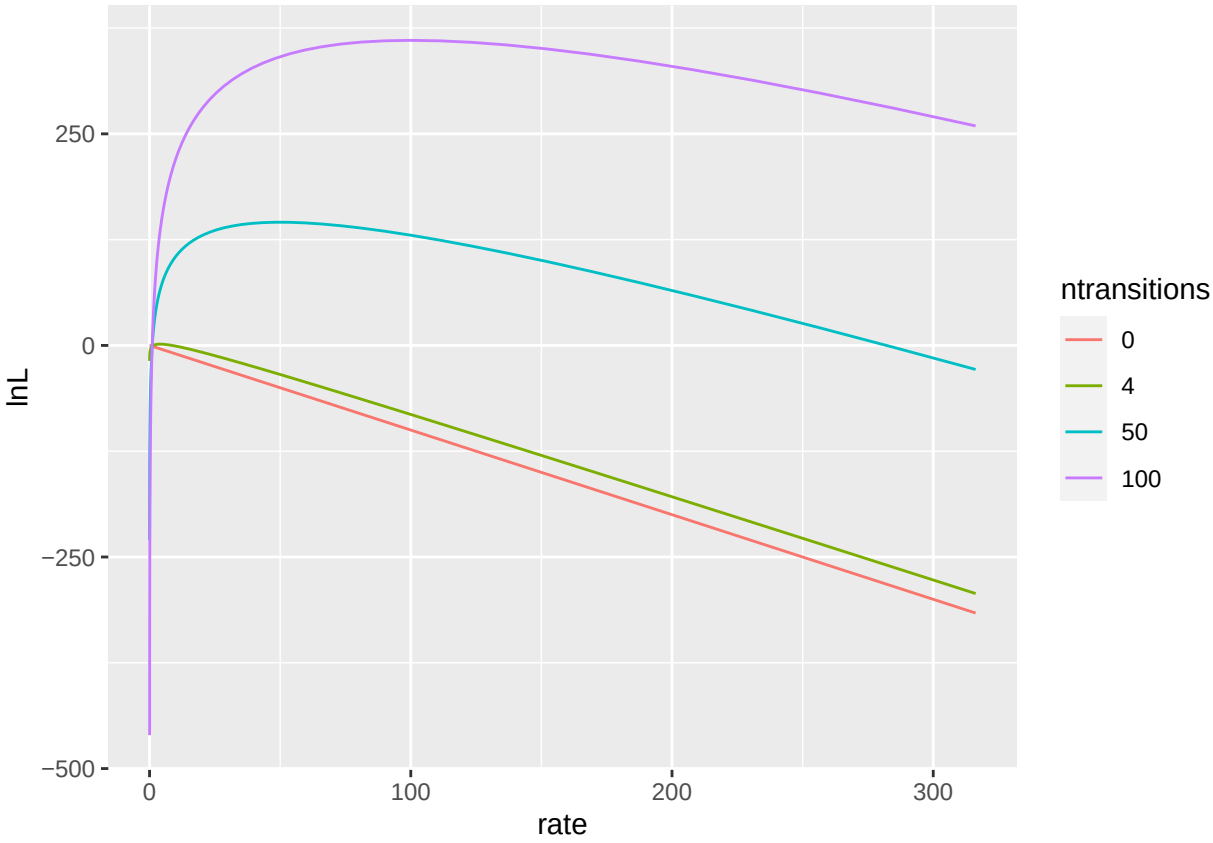
Which for any number of n transitions will be:

$$e^{-n(x/n)} = e^{-x}$$

x/n because the the branch length x will be divided into n subunits and the probability of the final branch segment (probability of no-transition) for lambda of one is $1 - (1 - e^{-\lambda x}) = e^{-x}$

```
likelihood_given_number_transitions <- function(ntransitions=1, rate=0.1, time=1) {
  timeinterval <- time/(ntransitions+1)
  return( ((rate*exp(-rate*timeinterval))^ntransitions) * (exp(-rate*timeinterval)))
}
```

```r
all_values <- expand.grid(ntransitions=c(0,4,50, 100), rate=c(10^seq(from=-2, to=2.5, length.out=100)))
all_values <- all_values[order(all_values$ntransitions),]
all_values$likelihood <- NA
for (i in sequence(nrow(all_values))) {
  all_values$likelihood[i] <- likelihood_given_number_transitions(all_values$ntransitions[i], all_value
}
# print(all_values[which.max(all_values$likelihood),])
all_values$lnL <- log(all_values$likelihood)
all_values$ntransitions <- as.factor(all_values$ntransitions)
library(ggplot2)
ggplot(all_values, aes(x=rate, y=lnL, group=ntransitions)) + geom_line(aes(color=ntransitions))
```



If we extend this graph to see higher rate values we find the true MLE is much further to the right once the phase transition has occurred and rates are greater than 1. So if we were to allow rates > 1, we would find an infinite number of transitions per branch and an infinite rate. This makes little biological sense (why would a state of a species change every millisecond), so we limit the rate to be < 1.