Working Draft

Patrick McKenzie & Rachel Swenie

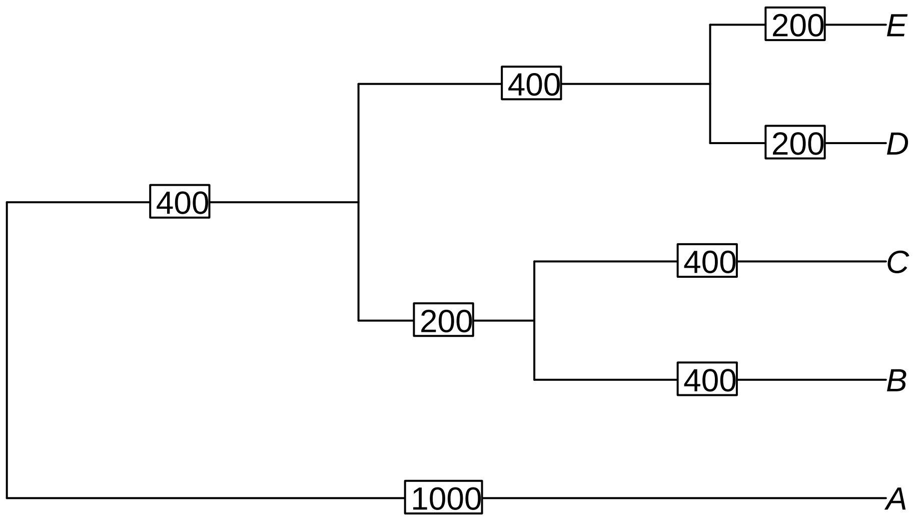
April 24, 2017

Biogeography Model

Using data matrices, movement function, and speciation function to reconstruct ancestral ranges.

Sample tree

(A:1000,((B:400,C:400):200,(D:200,E:200):400):400);



Generate a species range in a matrix with this cool function

Input the number of matrix rows and columns as the first two arguments, and the number of cells you want to be occupied by the species as the second argument. If you want to species' range to be centered around a particular point, designate that using a two-element vector as the argument for "startingcell" – otherwise, the range will be centered around a random point.

```
get_random_species_range <- function(nrow, ncol, numbercells, startingcell = NULL) {
  rows <- nrow
  cols <- ncol
  randommatrix <- matrix(rep(0,rows*cols), nrow = rows, ncol = cols)
  if (is.null(startingcell)) {
    index <- sample(rows*cols,1)
    randommatrix[index] <- 1
    pointindices <- arrayInd(index,c(rows,cols))
  } else {
    pointindices <- startingcell
    dim(pointindices) <- c(1,2)
  }

  while(sum(randommatrix) < numbercells) {
  workingcell <- pointindices[sample(nrow(pointindices),1),]
  random_adjacent <- sample(c(-1:1),2,replace = TRUE) #new cell row and col
  check_occupancy <- workingcell + random_adjacent
  if (sum(check_occupancy > 0) == 2 && sum(check_occupancy <= c(rows,cols)) == 2) {
    if (randommatrix[check_occupancy[1],check_occupancy[2]] == 0) {
      randommatrix[check_occupancy[1],check_occupancy[2]] <- 1
      pointindices <- rbind(pointindices,check_occupancy)
    }
  }
  }
  randommatrix
}
```

Movement and speciation

For each step in time, the following eqation is applied to each cell to calculate its new "probability" value. * These are normalized at each node and no longer represent true probabilities when moving up the tree.

$$P_{s,x,y}(t-1) = E_{x,y}(t-1)(((1-P_{s,x,y}(t)) * \alpha * \frac{\bar{N}}{8}) + (P_{s,x,y}(t) * \beta * \frac{\bar{N}}{8}))$$

The same equation is written out in code below. Note that `get_Nbar()` is a separate function written to calculate $\bar{N}$.

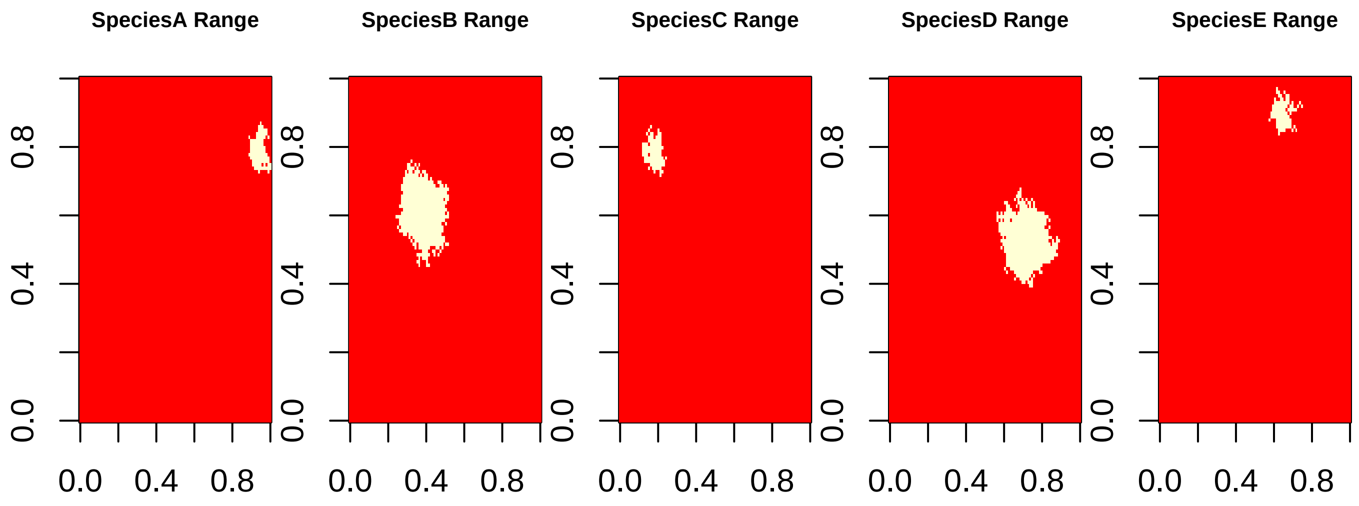
```
get_prob_matrix <- function(Ps,Es, alpha, beta) {
  probs_older <- Ps
  for (i in 1:nrow(Ps)) { # rows
    for (q in 1:ncol(Ps)) { # columns
      Pcell <- Ps[i,q]
      Ecell <- Es[i,q]
      Nbar <- get_Nbar(Ps,i,q)
      Polder <- Ecell * ( ( (1-Pcell) * (Nbar/8) * alpha) + (Pcell * (Nbar/8) * beta) )
      probs_older[i,q] <- Polder
    }
  }
  probs_older
}
```

When a node is reached (moving from tips to root), the probability matrix for each lineage is normalized, and the two matrices are multiplied together.

Species Range Data

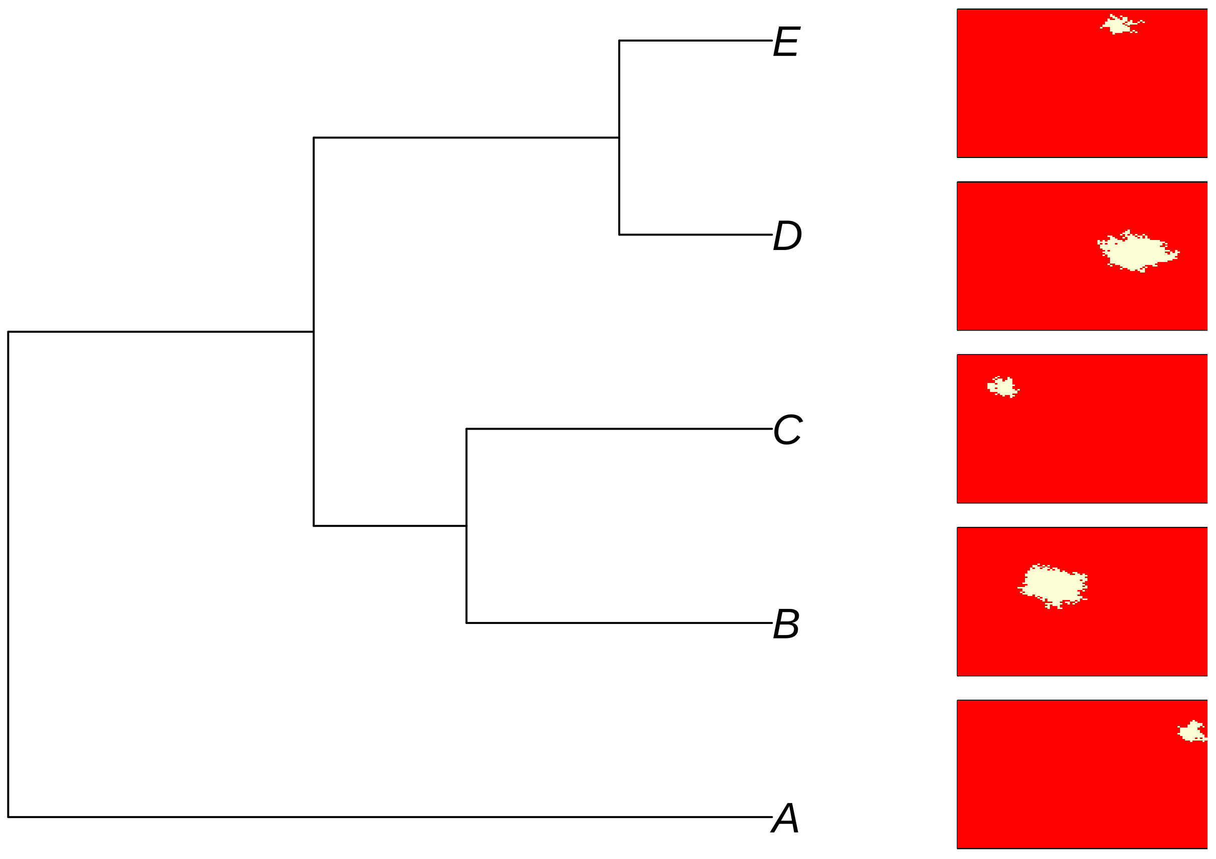
We first need to set ranges for the extant lineages on the phylogeny above. We'll use random ranges generated by the `get_random_species_range()` function above, having arbitrarily chosen occupancy of 100 cells for species A, C, and E and 500 cells for species B and D.

```r
speciesA <- get_random_species_range(100,100, 100)
image(speciesA,main="SpeciesA Range")
speciesB <- get_random_species_range(100,100, 500)
image(speciesB,main = "SpeciesB Range")
speciesC <- get_random_species_range(100,100, 100)
image(speciesC,main = "SpeciesC Range")
speciesD <- get_random_species_range(100,100, 500)
image(speciesD,main="SpeciesD Range")
speciesE <- get_random_species_range(100,100, 100)
image(speciesE,main="SpeciesE Range")
```



Here we can see the random ranges generated for each species. Now we want to plot them beside our phylogeny – easily accomplished with the `layout()` function.

```r
Ntip <- length(tree$tip.label)
layoutmatrix <- matrix(c(rep(1,Ntip),2:(Ntip+1)),ncol=2)
layout(layoutmatrix,widths = c(.8,.2))
par(mar = c(.5,.5,.5,0))
plot.phylo(tree,cex = 2)
#par(mar = c(2,1,2,3))
par(mar = c(0.5,0,0.5,0))
image(speciesE,xaxt='n', yaxt='n', ann=FALSE)
image(speciesD,xaxt='n', yaxt='n', ann=FALSE)
image(speciesC,xaxt='n', yaxt='n', ann=FALSE)
image(speciesB,xaxt='n', yaxt='n', ann=FALSE)
image(speciesA,xaxt='n', yaxt='n', ann=FALSE)
```



Scenario 1: Homogeneous environment

First, we want to make an environmental matrix that is completely homogeneous, where every cell is equally good for colonization by any species.

```r
Es <- matrix(nrow = 100,ncol = 100)
Es[1:10000] <- 1
```

Now we will run our `get_prob_matrix()` function up each branch of the phylogeny to reconstruct ancestral ranges at each node of the phylogeny.

```r
ancE <- speciesE
ancD <- speciesD
for (w in 1:200) {
  probs <- get_prob_matrix(ancD,Es,alpha = .5,beta = 0.8)
  ancD <- probs
  probs <- get_prob_matrix(ancE,Es,alpha = .5,beta = 0.8)
  ancE <- probs
  print(w)
}
ancD <- ancD/max(ancD)
ancE <- ancE/max(ancE)
```

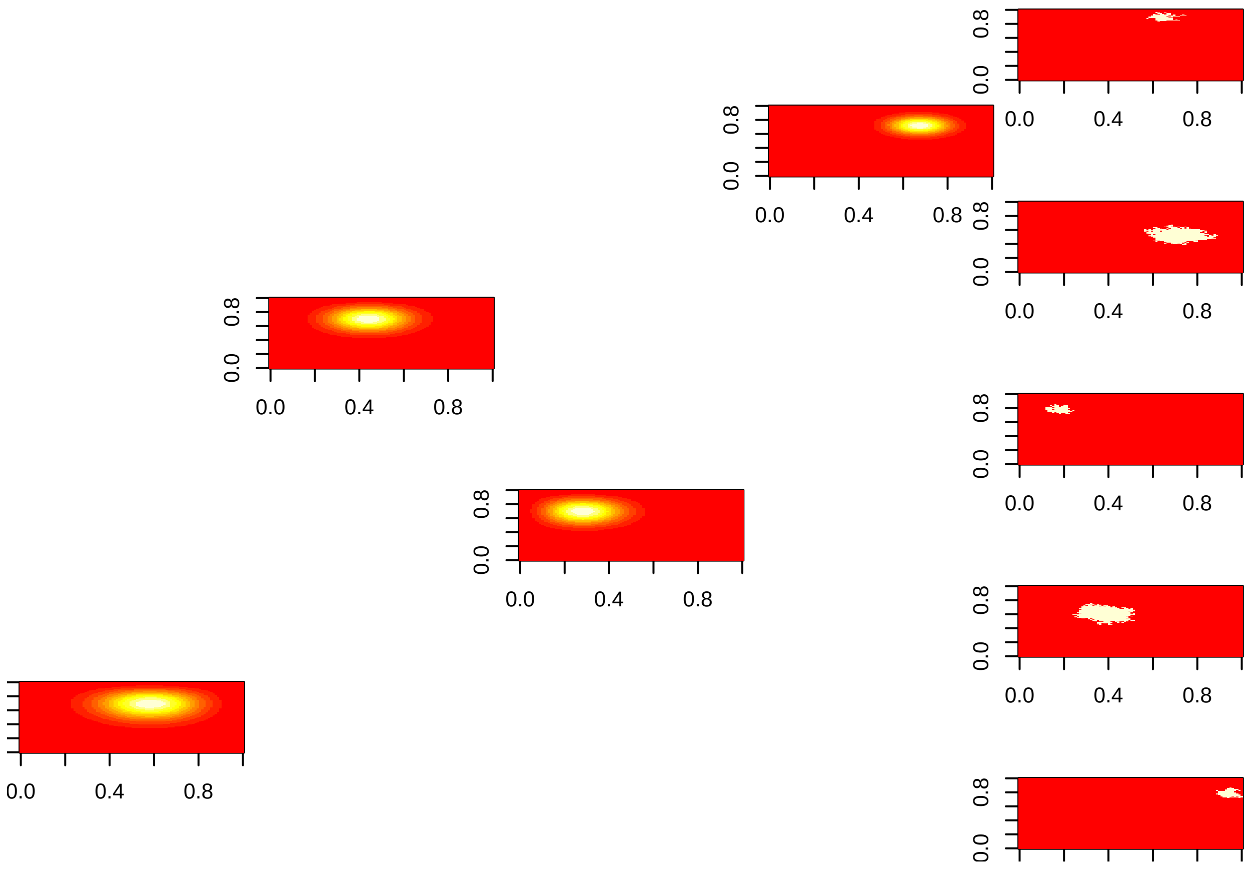
```r
ancDE <- ancD*ancE

ancC <- speciesC
ancB <- speciesB
for (w in 1:400) {
  probs <- get_prob_matrix(ancB,Es,alpha = .5,beta = 0.8)
  ancB <- probs
  probs <- get_prob_matrix(ancC,Es,alpha = .5,beta = 0.8)
  ancC <- probs
  print(w)
}
ancB <- ancB/max(ancB)
ancC <- ancC/max(ancC)
ancBC <- ancB*ancC

for (w in 1:400) {
  probs <- get_prob_matrix(ancDE,Es,alpha = .5,beta = 0.8)
  ancDE <- probs
  print(w)
}
for (w in 1:200) {
  probs <- get_prob_matrix(ancBC,Es,alpha = .5,beta = 0.8)
  ancBC <- probs
  print(w)
}
ancDE <- ancDE/max(ancDE)
ancBC <- ancBC/max(ancBC)
ancBCDE <- ancBC*ancDE

for (w in 1:400) {
  probs <- get_prob_matrix(ancBCDE,Es,alpha = .5,beta = 0.8)
  ancBCDE <- probs
  print(w)
}
ancA <- speciesA
for (w in 1:1000) {
  probs <- get_prob_matrix(ancA,Es,alpha = .5,beta = 0.8)
  ancA <- probs
  print(w)
}
ancBCDE <- ancBCDE/max(ancBCDE)
ancA <- ancA/max(ancA)
ancABCDE <- ancA*ancBCDE
```

Let’s look at the ancestral states plotted as nodes on the phylogenetic tree:

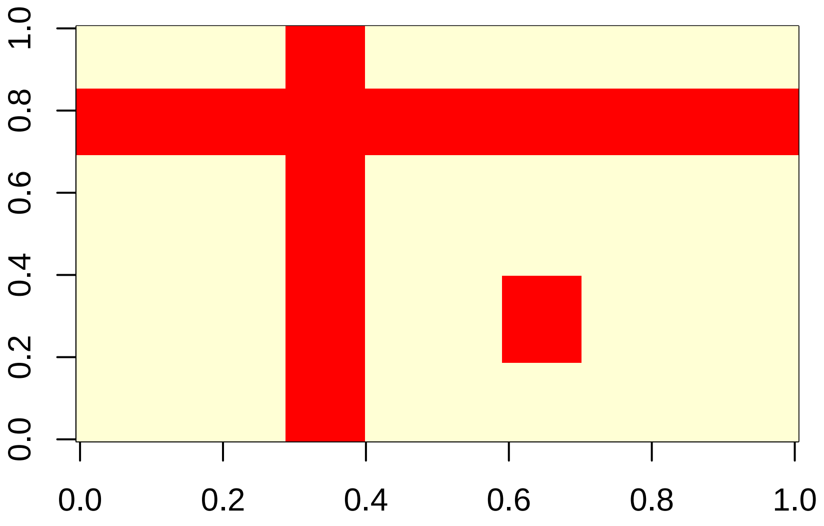


Scenario: Adding bands of poorer-quality environment

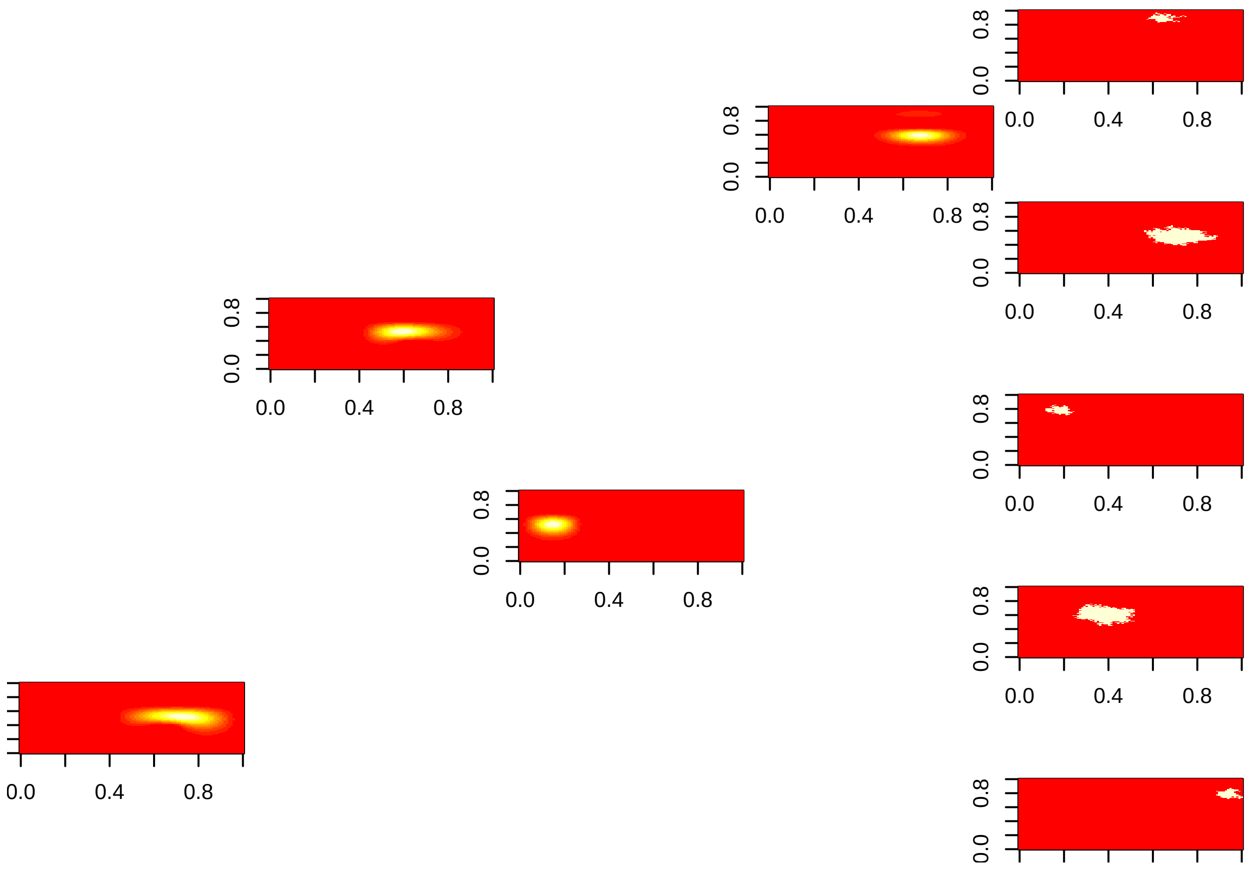
Now we can explore how heterogeneity in the environmental matrix can affect decisions for ancestral states. We'll add bands of less-hospitable environment below.

```
Es <- matrix(nrow = 100,ncol = 100)
Es[1:10000] <- 1

Es[,70:85] <- .8
Es[30:40,] <- .8
Es[60:70,20:40] <- .8
image(Es)
```



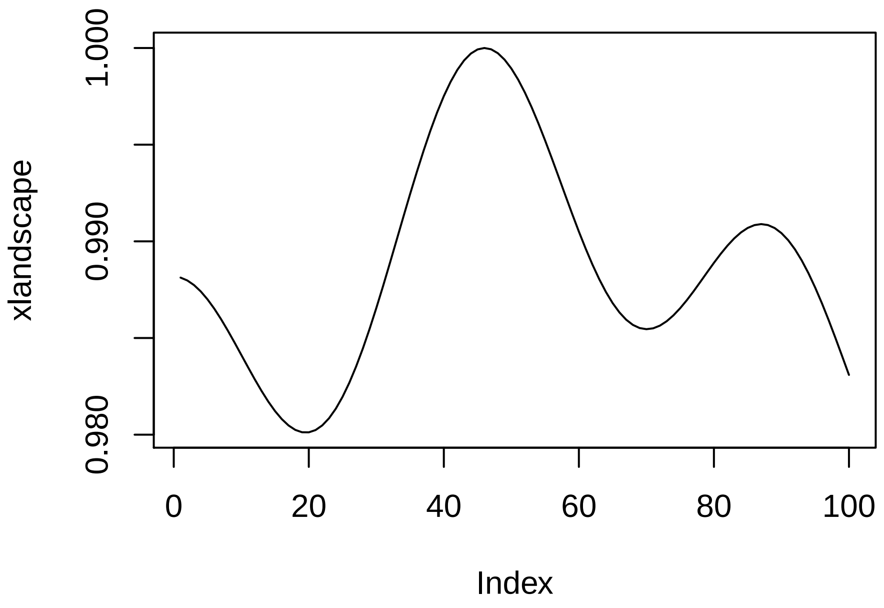
Now, again, we can see how ancestral states are reconstructed on the tree.



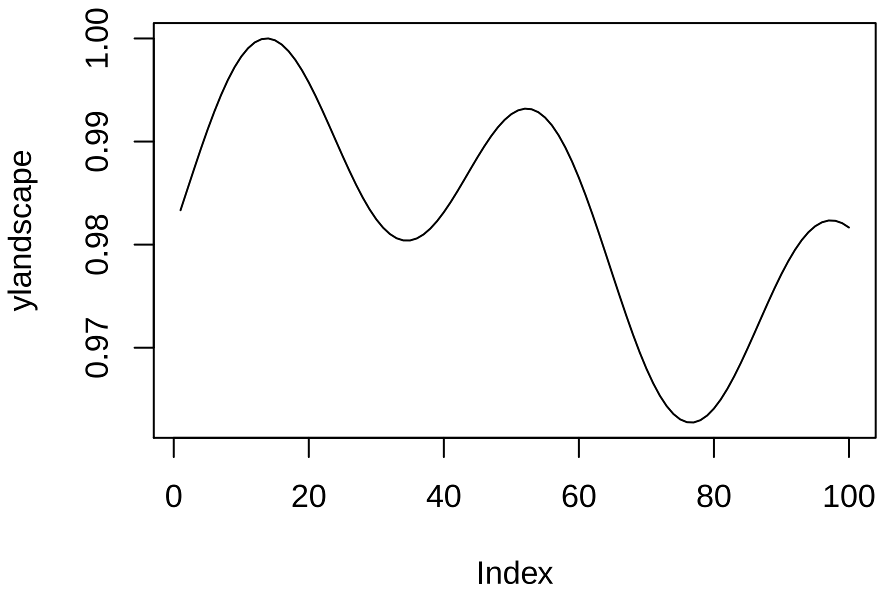
Scenario: Hotspot Environment

Let's make a new environmental matrix with "hotspots" of hospitable regions generated from intervering trig functions. The code producing the environmental matrix, along with a heatmap, is pictured below.

```
x <- 1:100
xlandscape <- (sin(.04*x))/9 * (sin(.1*x))/9
xlandscape <- xlandscape-min(xlandscape)
xlandscape <- 1 - xlandscape
plot(xlandscape,type="l")
```

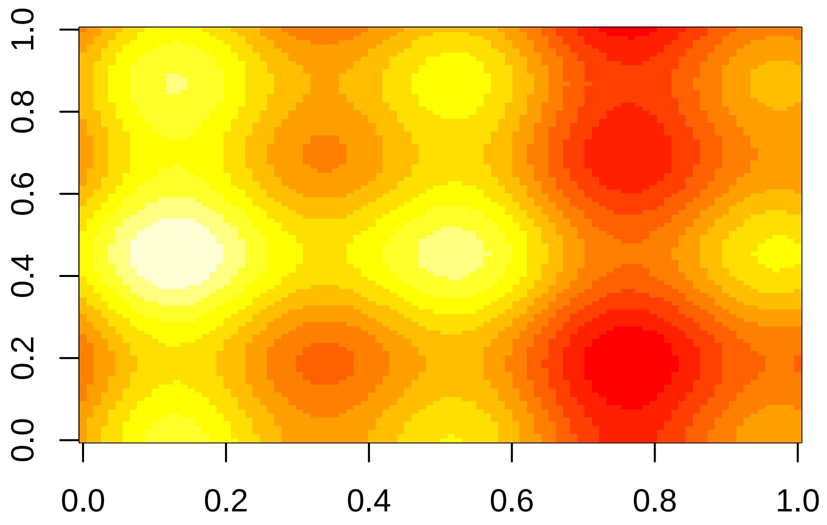


```
y <- 1:100
ylandscape <- sin(.05*y +.4*pi)/7 * sin(.1*y +pi)/7
ylandscape <- ylandscape-min(ylandscape)
ylandscape <- 1 - ylandscape
plot(ylandscape,type="l")
```



```
Es <- matrix(nrow = 100,ncol = 100)
Es[1:10000] <- 1

for(i in 1:100) {
  Es[,i] <- Es[,i]*xlandscape[i]
}
for(i in 1:100) {
  Es[i,] <- Es[i,]*ylandscape[i]
}
image(Es)
```



Now we can see how the this influences the ancestral state predictions.