Using SEAMLESS

Donald R. Schoolmaster Jr.

June 23, 2016

SEAMLESS simulates biogeogrphs scale barrier creation and disperal resulting from barrier removal in a 1-dimensional landscape.

```
#load the SEAMLESS functions
source('SeamlessMain.R')
#load some tools for quantifing trees
source('TraverseTrees.R')
#load ape package
library(ape)
```

```
##
## Attaching package: 'ape'

## The following objects are masked _by_ '.GlobalEnv':
##
##     node.depth, node.height
```

Start a new simulation by initializing

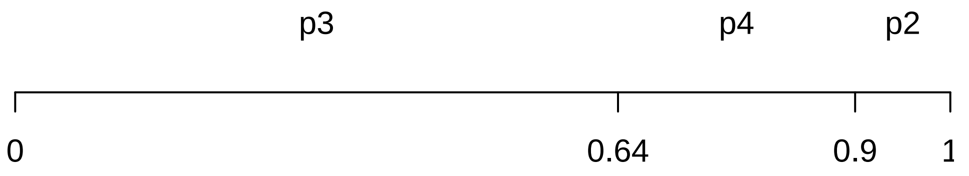
```
#initialize new Landscape
initialize()
#create 2 more 'patches'
for(i in 1:2)add.patch()
#look at lndsp object
lndsp
```

```
## $p2
## An object of class "patch"
## Slot "bnd":
## [1] 0.8981803 1.0000000
##
## Slot "spp":
## [1] "S3"
##
##
## $p3
## An object of class "patch"
## Slot "bnd":
## [1] 0.0000000 0.6446266
##
## Slot "spp":
## [1] "S4"
##
##
## $p4
## An object of class "patch"
## Slot "bnd":
## [1] 0.6446266 0.8981803
##
## Slot "spp":
## [1] "S5"
```

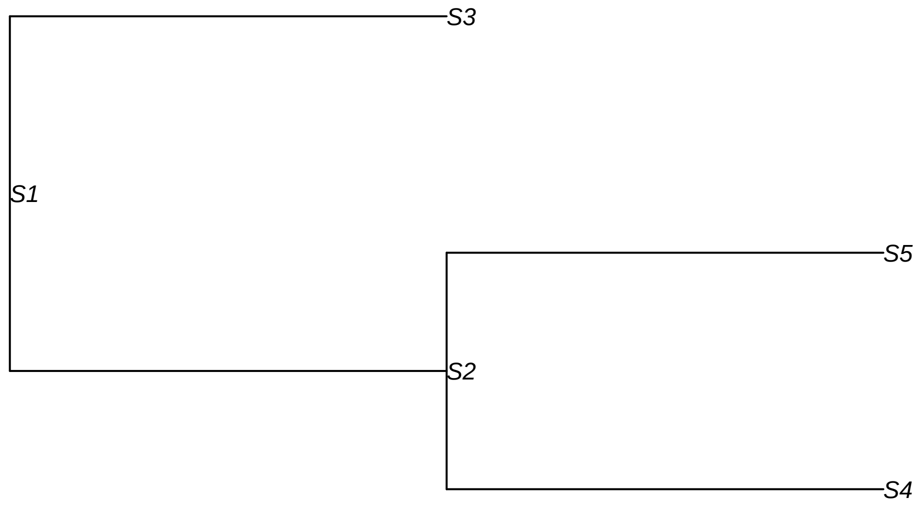
Notice that each patch in the lndsp object has a name i.e. “p3”, a Slot that contains the position of the spatial boundaries called ‘bnd’, and a Slot for the list of species in that patch called “spp”.

We can visualize the landscape and the resulting tree

```
#plot the landscape
plot.lndsp()
```



```
#plot the tree
#first translate the SpList to newick-style notation
newick<-plots.tree(SpList)
#then use functions from ape to plot it
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```

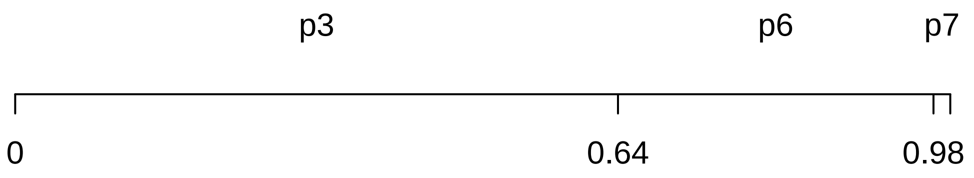


The add.patch() function just pick a patch from the landscape at random and splits it. Next, let’s simulate the movement of boundaries between the patches

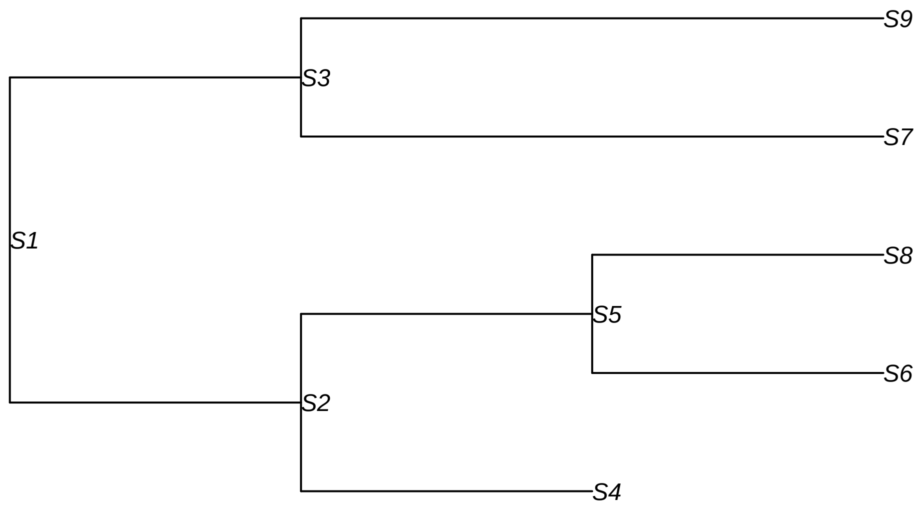
```
remove.patch()
```

```
## [1] 0.982038
```

```
#plot new landscape
plot.lndsp()
```



```
#plot updated tree
newick<-plots.tree(SpList)
#then use functions from ape to plot it
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```



The remove.patch() functions selects a boundary at random, removes it, allowing dispersal between the patches, then places a new boundary randomly in the new larger patch, causing speciation events.

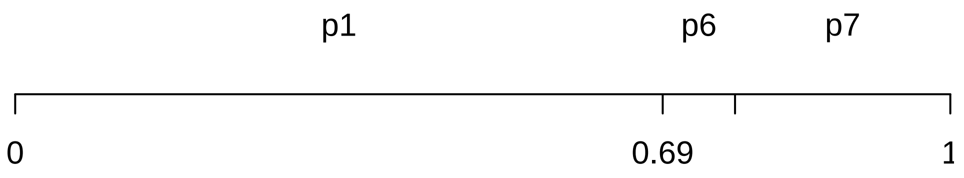
We can look at the distribution of species among the patches in the landscape. the get.spp() function returns a list of species in each patch

```
get.spp()
```

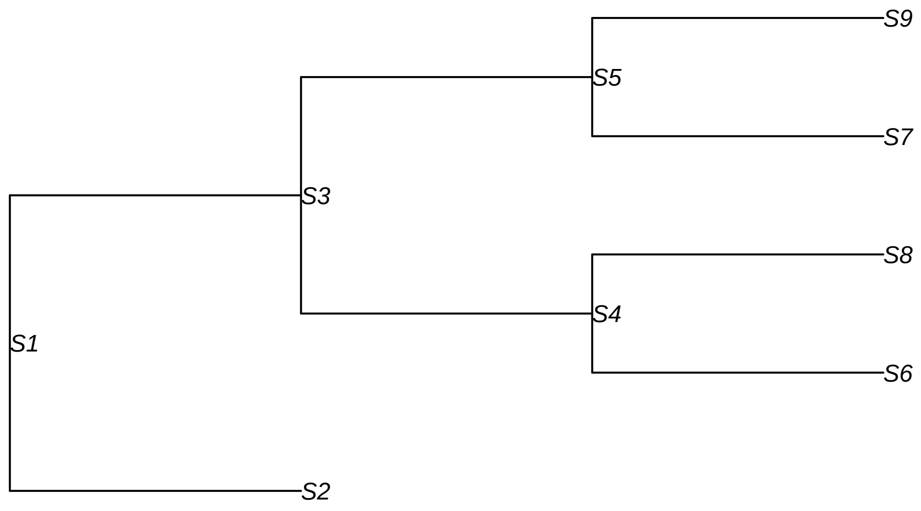
```
## $p3
## [1] "S4"
##
## $p6
## [1] "S6" "S7"
##
## $p7
## [1] "S8" "S9"
```

These simulatoin step are automated with the sim.tree(npatch,ntot), which takes as arguments the number of total patches desired, npatch, and the total number of boundary movement steps you want the simulation to run, ntot. For example, if we want to recreate the simuation we above,

```
initialize()
sim.tree(3,1)
plot.lndsp()
```

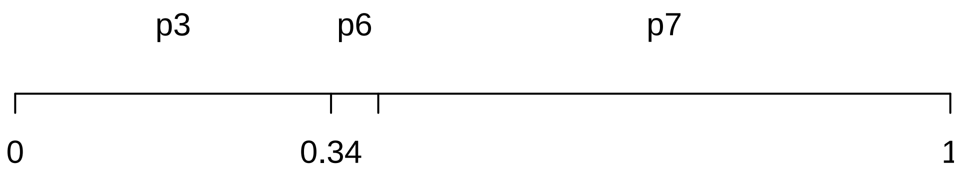


```
newick<-plots.tree(SpList)
#then use functions from ape to plot it
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```

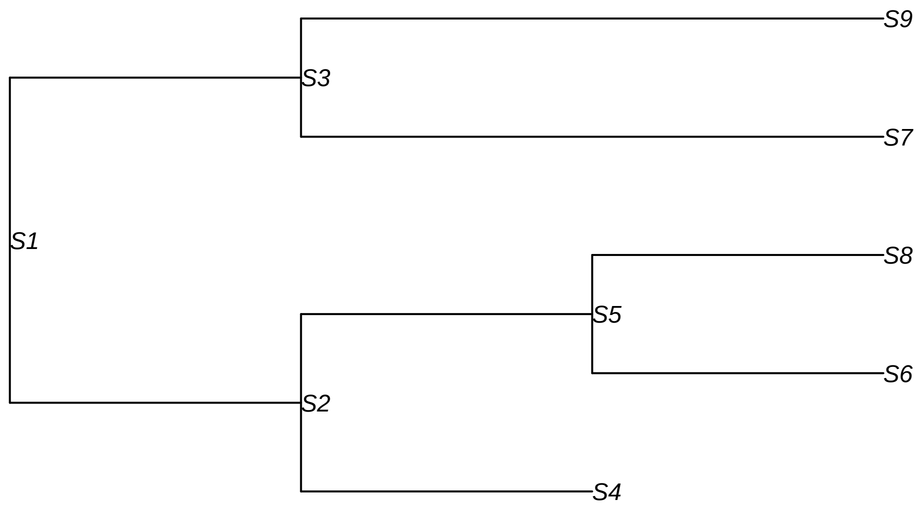


The toosmall.patch(thld) seaches the landscape object for any patches smaller than a given size threshold, thld, and removes all species from it. Simulations with a size threshold can be run with the sim.tree2() function.

```
#run same simulation with extinction thershold of 2.5% of total landscape area
initialize()
sim.tree(3,1.025)
plot.lndsp()
```

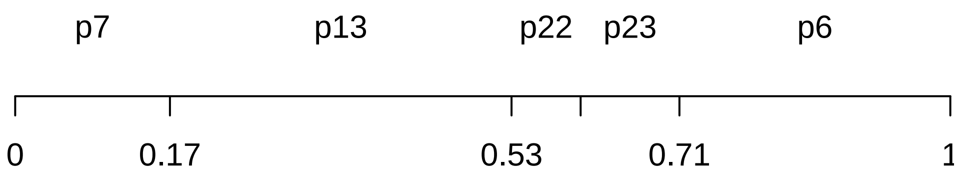


```
newick<-plots.tree(SpList)
#then use functions from ape to plot it
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```

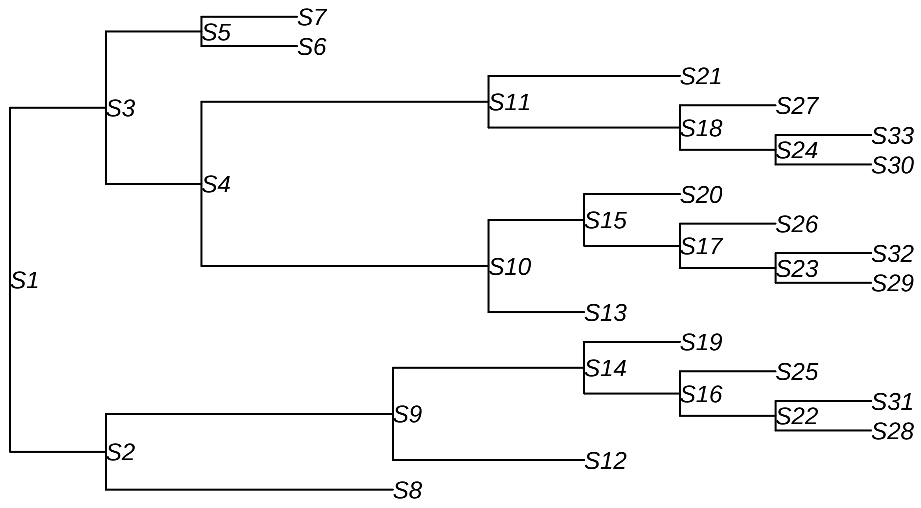


The TraverseTrees.R file contains a number of functions useful for describing the topology of trees.

```
#create a larger tree
initialize()
sim.tree2(5,5,.025)
plot.lndsp()
```



```
newick<-plots.tree(SpList)
#then use functions from ape to plot it
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```



The encode.tree(SpList) function takes a SpList generated by the simulations and creates an object of the tree class. the tree object.

```r
#encode the SpList generated by the simulation into a tree object
my.tree<-encode.tree(SpList)

#List ancestors of given node
my.ancestors(my.tree,"S13")
```

```
## [1] "S10" "S4"  "S3"  "S1"
```

```r
#detmerine if one node is an ancestor of another
is.ancestor(my.tree,"S8","S3")
```

```
## [1] FALSE
```

```r
#note that this function is not symmetrical (of course)
is.ancestor(my.tree, "S3","S8")
```

```
## [1] FALSE
```

```r
#find distance along tree between and "root" node and a "target" node
node.depth(my.tree,"S1","S13")
```

```
## [1] 4
```

```r
##find size of tree from a root node
tree.size(my.tree,"S1")
```

```
## [1] 33
```

```r
##find size of subtree using an interior node
tree.size(my.tree,"S5")
```

```
## [1] 3
```

```r
#node diameter gives the maximum number of steps between to leaves given a tree object and a target nod
node.diameter(my.tree,"S4")
```

```
## [1] 9
```

```r
#node.dia.balance returns the difference in balance between the two subtrees of a given node
node.dia.balance(my.tree,"S1")
```

```
## [1] 3
```

```r
#node.height return the maximum number of steps between a given node and its most distance descendent
node.height(my.tree,"S1")
```

```
## [1] 7
```

```r
#node.height.balance returns the difference in node heights between the two subtrees of a given node
node.height.balance(my.tree,"S1")
```

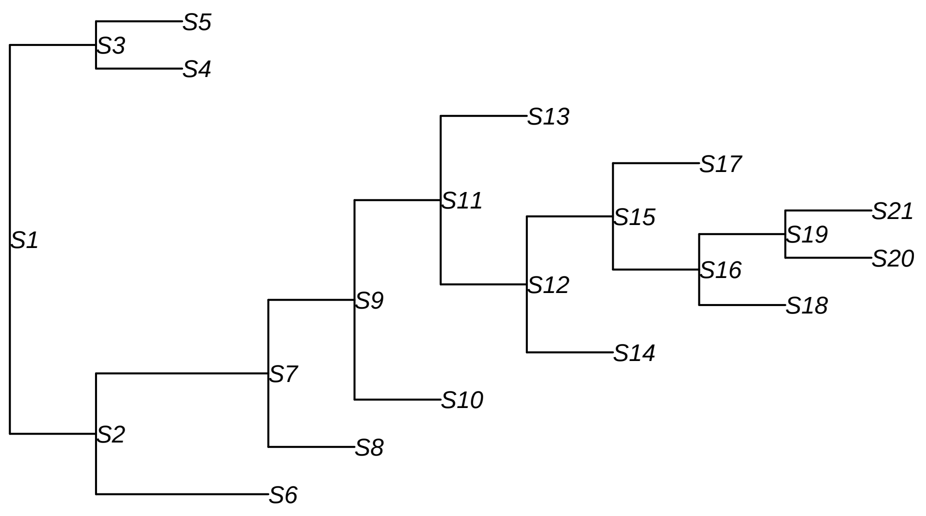
```
## [1] 1
```

```r
#uses the branch length to retun the number of simulation steps between a node and one of its ancestors
node.age(my.tree,"S1","S13")
```

```
## [1] 8
```

There are also functions for creating random trees. This function starts with a single node, then splits it. The for a user-defined number of times it selects one of the terminal nodes at random with equal probability and splits it. Thus, rnd.tree(n), return a SpList and tree object with a tree size of 2*n+1 and n+1 terminal nodes.

```r
rtree<-rnd.tree(10)
newick<-plots.tree(rtree$SpList)
plot(read.tree(text=newick),show.node.label=T,cex=.75)
```



```r
#return total tree size
tree.size(rtree$tree,"S1")
```

```
## [1] 21
```

```r
#return list of terminal nodes
term.nodes<-get.leaves(rtree$SpList)
#find number of terminal nodes
length(term.nodes)
```

```
## [1] 11
```