

BIOS13 - Question 4

Pham Xuan Huy Nguyen

March 22, 2023

a R function of simulation

The *simulation* function takes two arguments *pe* as the extinction probability and *ps* as speciation probability, and one optional argument *n* is the number of species in the first period, the default *n* is set as 1.

A possible outcome for the function *simulation(0.1,0.2)* is shown below the code part.

```
rm(list=ls())
simulation <- function(pe,ps,n) {
  if(missing(n)) {
    x = c(1,rep(0,99)) #If n is not given, then it is set as 1
  } else {
    x = c(n,rep(0,99))
  }
  for (iter in 1:99) {
    temp=0
    if (x[iter]==0) {
      break #At any period, when population reaches 0, stop the loop.
    } else {
      for (i in 1:x[iter]) {
        e = runif(1) # Extinct prob and Speciate prob
                        # are generated differently randomly for every species
        s = runif(1)
        if (e<=pe){
          temp=temp-1
        } else if (e>pe & s<=ps) {
          temp=temp+1
        } else if (e>pe & s>ps) {
          temp=temp+0
        }
      }
      x[iter+1]=x[iter]+temp
    }
  }
}
```

```

plot(NA, type='n',xlim=c(0,10),ylim=c(0,max(x)),
     ylab='Number of species',xlab='time (years) x 10^6')
lines(seq(0.1,10,by=0.1),x)
return(x)
}

```

images/4a.png

b Histogram

A possible outcome when running the *4b.R* file is shown below the code part.

```

rm(list=ls())
simulation <- function(pe,ps,n) {
  if(missing(n)) {
    x = c(1,rep(0,99)) #If n is not given, then it is set as 1
  } else {
    x = c(n,rep(0,99))
  }
}

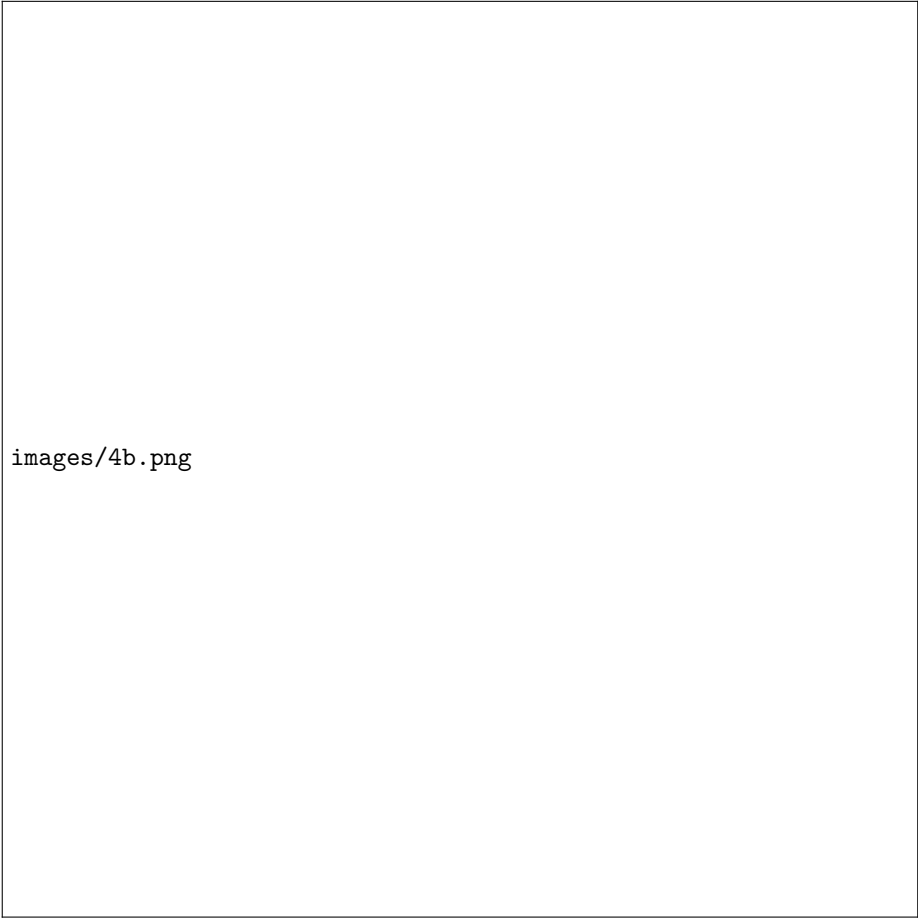
```

```

}
for (iter in 1:99) {
  temp=0
  if (x[iter]==0) {
    break
  } else {
    for (i in 1:x[iter]) {
      e = runif(1)
      s = runif(1)
      if (e<=pe){
        temp=temp-1
      } else if (e>pe & s<=ps) {
        temp=temp+1
      } else if (e>pe & s>ps) {
        temp=temp+0
      }
    }
    x[iter+1]=x[iter]+temp
  }
}
return(x)
}

# histogram
sum_of_simulation=0
for (i in 1:1000) {
  sum_of_simulation=sum_of_simulation+simulation(0.1,0.2)
}
hist(sum_of_simulation)

```



images/4b.png

c Modification of function for density dependent

The addition to this function is the *nmax* argument, the maximum number of species. When the number of species is larger or equal to the *nmax*, the *ps* argument is multiple with by the ratio of *nmax* over the number of current population that was multiplied by 1.8. I found that using the number from 1.6 to 1.8 is a good range of number to keep the population approximately equal to *nmax*.

The code *simulation(0.1,0.2,1500)* is used to illustrated to plot.

```
rm(list=ls())
simulation <- function(pe,ps,nmax,n) {
  if(missing(n)) {
    x = c(1,rep(0,99))
```

```

} else {
  x = c(n,rep(0,99))
}
for (iter in 1:99) {
  temp=0
  if (x[iter]==0) {
    break
  } else {
    ps_temp=ps*((nmax/1.8)/x[iter])
    pe_temp=pe
    for (i in 1:x[iter]) {
      e = runif(1)
      s = runif(1)
      if (e<=pe_temp){
        temp=temp-1
      } else if (e>pe_temp & s<=ps_temp) {
        temp=temp+1
      } else if (e>pe_temp & s>ps_temp) {
        temp=temp+0
      }
    }
    x[iter+1]=x[iter]+temp
  }
}
plot(NA, type='n',xlim=c(0,10),ylim=c(0,max(x)),
      ylab='Number of species',xlab='time (years) x 10^6')
lines(seq(0.1,10,by=0.1),x)
return(x)
}
simulation(0.1,0.2,1500)

```

images/4c.png