Building the individual based model

Sebastian Schreiber

April 8, 2015

First, clear all the variables and load the build of the IPM which I saved in the IPMs.Rdata file. Then pull out the key functions from the IPM build which are $s(x)$ the size-dependent survival function, probability density $g(y,x)$ of growing from size x to y, the mean number of offspring $f(x)$ produced by an individual of size x given the individual is having offspring, the probability $p_f(x)$ of having offspring at size x, and $os(y)$ the probability density of an offspring being size y at the next census.

```
rm (list = ls ())
load("IPMs.Rdata")
g=function(y,x)Growth(y,x)
s=function(x)Survival(x)
f=function(x)f.fecundity(x)*establishment.prob
pf=function(x)1
os=function(y)dnorm(y,mean=recruit.size.mean,sd=recruit.size.sd)
```

Now comes the important new piece. Need to define the probability generating function associated with the fecundity function. Since we used the Poisson family for getting the fecundity function, I assume that the number of offspring is Poisson distributed with mean $f(x)$ for an individual of size x. Hence, the probability generating function (with size x treated like a parameter) is given by

```
phi=function(x,s)exp(f(x)*(s-1))
```

Using all of these pieces, we can define the probability generating functional for individual based IPM by

$$\Psi(h)(x) = \left(1 - p_f(x) + p_f(x)\phi\left(x, \int_a^b os(y)h(y)dy\right)\right)\left(1 - s(x) + s(x)\int_a^b g(y,x)h(y)dy\right)$$

where h is a continuous function from $[a,b]$ to $[0,1]$.

To create this probability generating functional numerically, I use the grid on x values xs which go from the minimum size α to the maximum size β with n bins.

```
n=100
xs=seq(alpha,beta,length=n)
dx=xs[2]-xs[1]
```

Now create the survival, probability of having offspring, and offspring size vectors.

```
s.mat=s(xs)
pf.mat=pf(xs)
os.mat=os(xs)*dx
```

We need to make sure that the o.mat integrates to one i.e. there is no eviction. Eviction creates a major problem for the probability generating functionals.

```
sum(os.mat)
```

```
## [1] 0.9961
```

As this sum is pretty close to one, I just renormalize it.

```
os.mat=os.mat/sum(os.mat)
sum(os.mat)
```

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

Next need to define the growth matrix and deal with the eviction issue. In this case, I am treating eviction as mortality as this is consistent with what was being done in the Nicols et al. paper. The new growth model is s.evict*g.evict which agrees with the orginal growth kernel g but has a column stochastic matrix g.evict. Note: Most of this eviction is occuring at negative sizes.

```
g.mat=outer(xs,xs,g)*dx
q.evict=1-colSums(g.mat)
g.evict=g.mat%*%diag(1/(1-q.evict))
s.evict=(1-q.evict)*s.mat
```

Now, we are ready to create the discretized probability generation function which takes vectors of length n (i.e. the discretized function h) and returns vectors of length n. Note: I need to take the transpose to take integrals down columns.

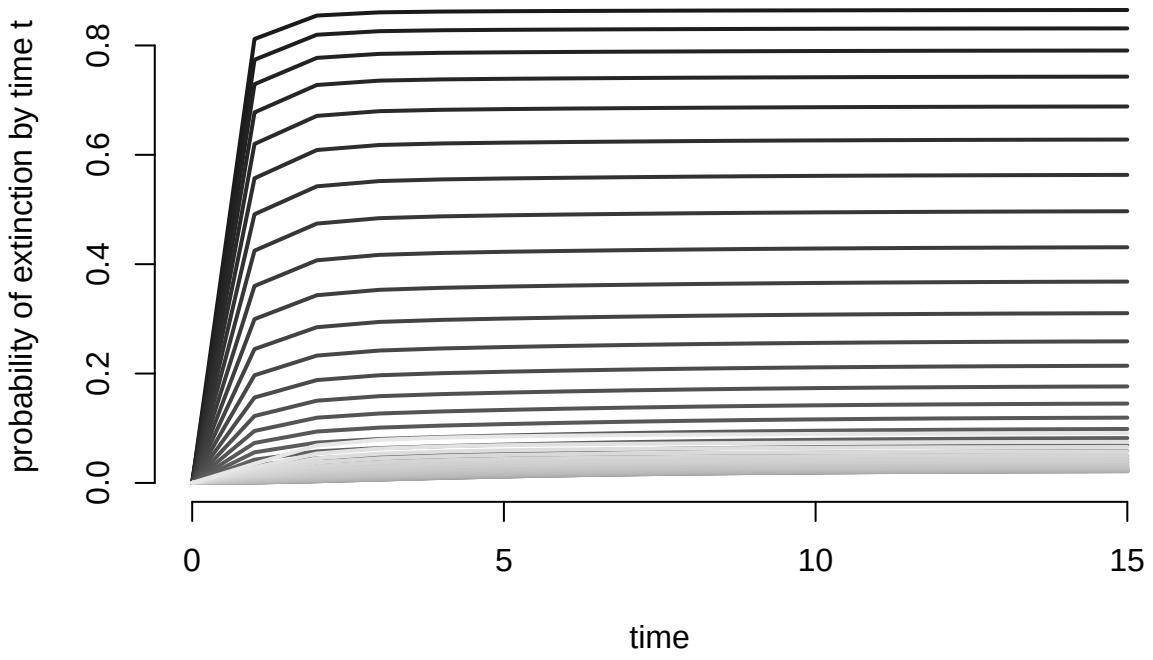
```
Psi=function(h){
    h.new=(s.evict*(t(g.evict)%*%h)+1-s.evict)*(1-pf(xs)+pf(xs)*phi(xs,os.mat%*%h))
    return(h.new)
}
```

We can use this functional to determine how the probability of extinction varies in time. In particular, iterating the $h = 0$ function for t time steps yields $\Psi^t(h)$ where $\Psi^t(h)(x)$ is the probability of the population going extinct by year t given initially there was one individual of size x in the population. The following function iterates the discretized $h = 0$ function until time T and returns a matrix of dimension $n \times (T + 1)$ whose $t - 1$-th column correspondds to $\Psi^t(h)$.

```
Psi.iterate=function(T){
    hs=matrix(0,T+1,n)
    for(t in 1:T){
        hs[t+1,]=Psi(hs[t,])
        }
    return(hs)
}
```

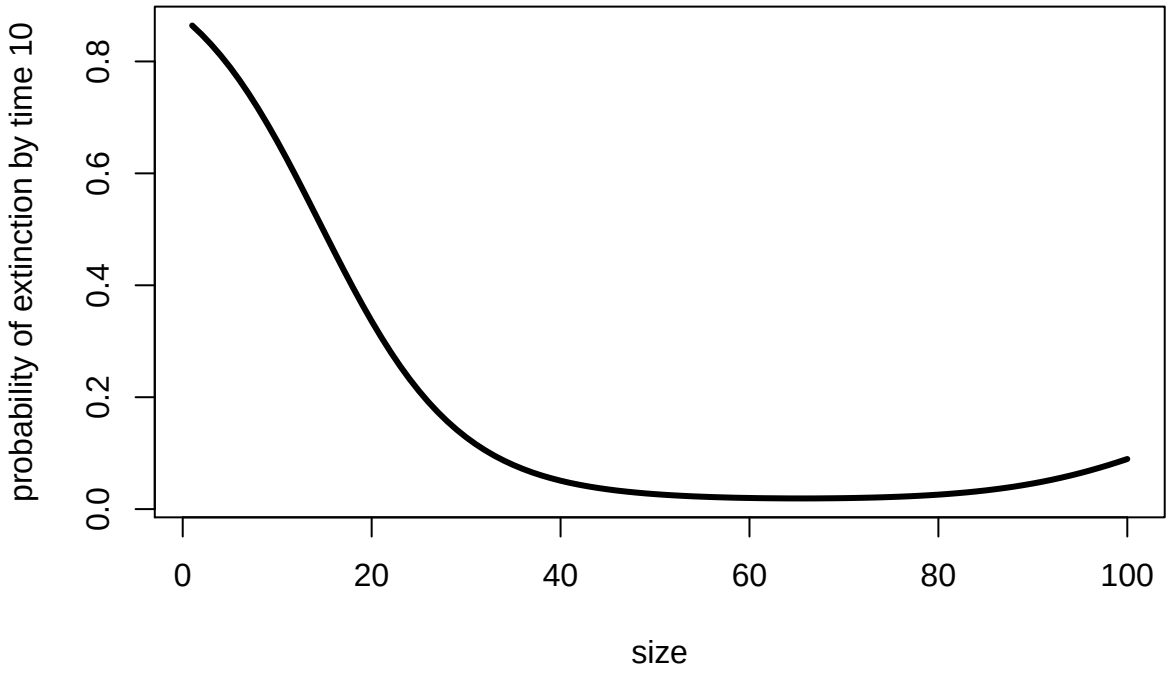
OK. Lets try this function out and plot the probabilities of extinction as functions of time for several sizes.

```
T=15
out=Psi.iterate(T)
sizes=seq(1,n,length=50)
matplot(0:T,out[,sizes],type='l',lwd=2,lty=1,bty="n",col=grey(seq(0.1,0.9,length=length(sizes))),xlab="
```



Can also plot exinction by time 10 as a function of the initial size of the "founding individaul"

```
plot(out[11,],type="l",lwd=3,xlab="size",ylab="probability of extinction by time 10")
```



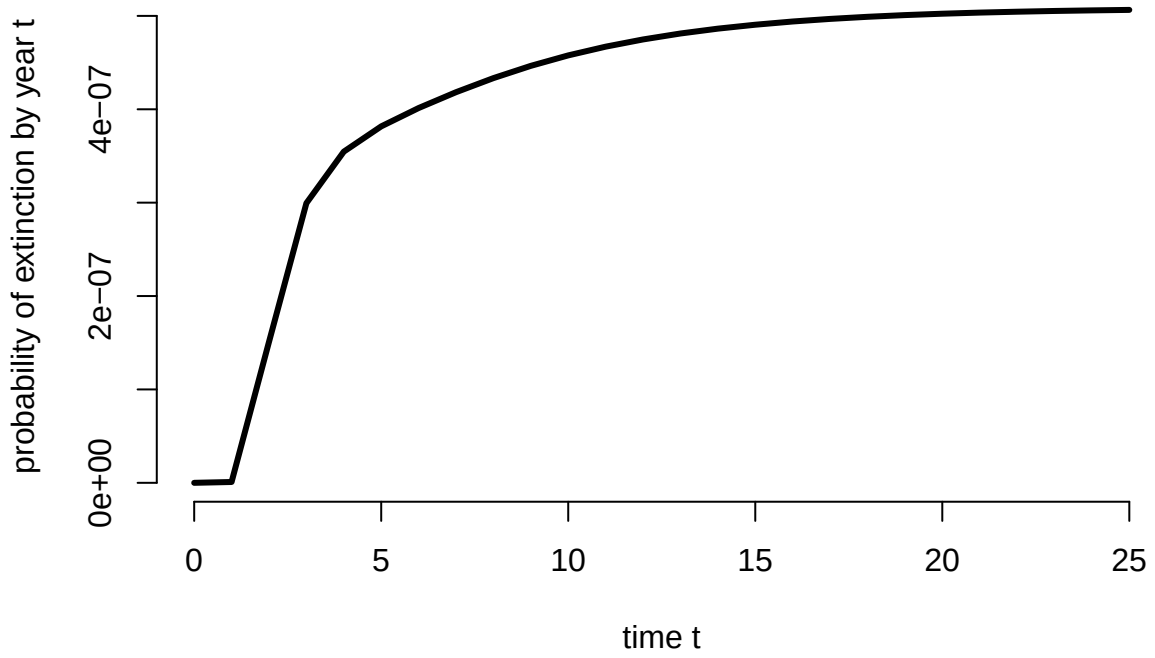
To scale things up to an entire population, we can introduce the following function with two inputs: N vector of length n which specifies the initial number of individuals in each size bin, and T the length of the time to look things over. The function returns a vector of length $T+1$ corresponding to the probabilities of extinction by time t or sooner.

```
PVA=function(N,T){
    hs=Psi.iterate(T)
```

```
    output=numeric(T+1)
    for(t in 1:(T+1)){
        output[t]=prod(hs[t,]^N)
    }
    return(output)
}
```

Lets try this out by considering a population of 100 individuals of the smallest size.

```
T=25
N=rep(0,n)
N[1]=100
out=PVA(N,T)
plot(0:T,out,typ="l",lwd=3,bty="n",xlab="time t",ylab="probability of extinction by year t")
abline(h=0.01)
```

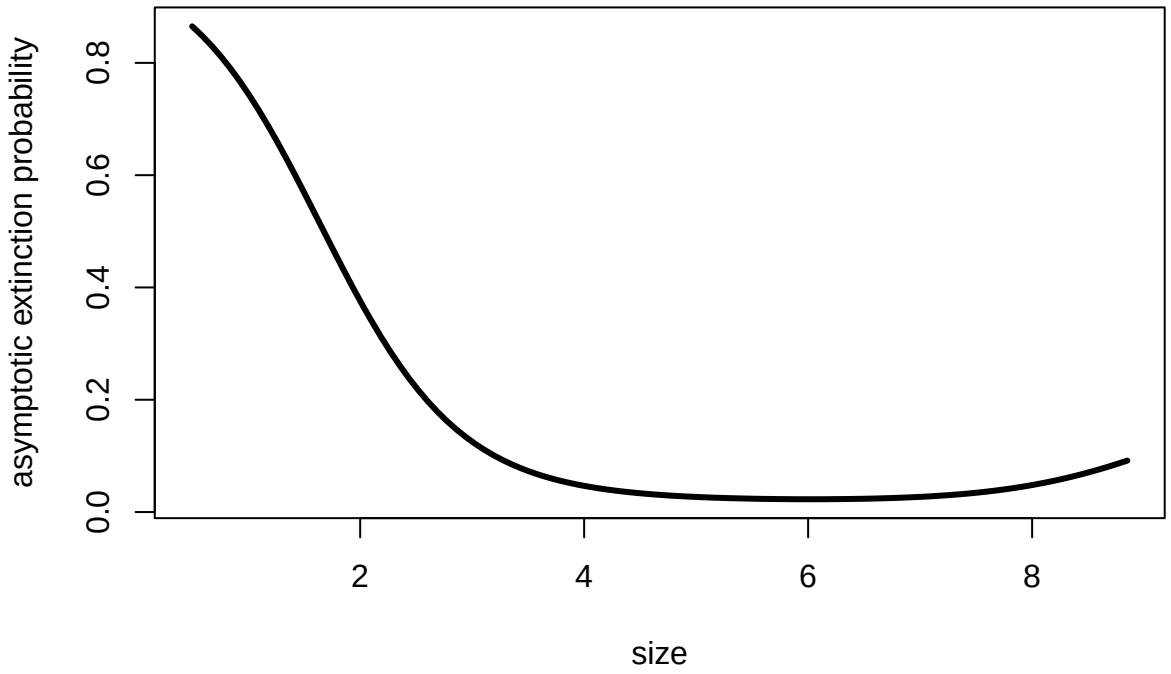


Clearly these probabilities asymptote. The following function computes this asymptotic probability of extinction by iterating toward the steady state.

```
AE=function(tol=0.0001){
  h=rep(0,n)
  diff=1
  while(diff>tol){
    h.old=h
    h=Psi(h)
    diff=sum(abs(h.old-h))
  }
  return(h)
}
```

Lets try this out.

```
h=AE()
plot(xs,h,type="l",xlab="size",ylab="asymptotic extinction probability",lwd=3)
```



Can see that the asymptotic exinction probability tends to decrease with the size of the founding individual but at large sizes begins increasing again. This latter trend is due to the eviction in the model at large sizes.