Popbio tutorial

Matrix population modeling workshop

3/4/2019

R package 'Popbio' tutorial

This is a short tutorial of R package Popbio using the example dataset from the package. Popbio package allows for construction and analysis of of matrix population models.

```
#install popbio
#install.packages("popbio")
library(popbio)

###let's see all the data in popbio
data(package="popbio")

####load whale data matrix
data("whale")
head(whale)
```

```
##            yearling juvenile mature postreprod
## yearling     0.0000   0.0043 0.1132     0.0000
## juvenile     0.9775   0.9111 0.0000     0.0000
## mature       0.0000   0.0736 0.9534     0.0000
## postreprod   0.0000   0.0000 0.0452     0.9804
```

Create matrix

```
# Create a matrix
# 1. Using values
tiger.vector<-c(0,0,2.34,0.368,0, 0, 0, 0.452, 0)
ages<-c("cubs", "subadult", "adult") #age categories
A.tiger=matrix2(tiger.vector, ages)
A.tiger
```

```
##           cubs subadult adult
## cubs     0.000    0.000  2.34
## subadult 0.368    0.000  0.00
## adult    0.000    0.452  0.00
```

```
#2. using lower level prameters

goose.vr<-list( Ss0=0.1357, Ss1=0.8926, Sf2=0.6388, Sf3= 0.8943) #vital ratevalues
goose.el<-expression( 0, 0, Sf2*Ss1,Sf3*Ss1,
                      Ss0,0, 0, 0,
                        0, Ss1,0, 0,
                        0, 0, Ss1, Ss1) #matrix elements
Avec=sapply(goose.el, eval, goose.vr) # evaluate the expression (calculate values)
A=matrix(Avec, nrow=sqrt(length(goose.el)), byrow=TRUE) # stick values in a matrix
A
```

```
##        [,1]   [,2]      [,3]      [,4]
## [1,] 0.0000 0.0000 0.5701929 0.7982522
## [2,] 0.1357 0.0000 0.0000000 0.0000000
## [3,] 0.0000 0.8926 0.0000000 0.0000000
## [4,] 0.0000 0.0000 0.8926000 0.8926000
```

Let's project the population

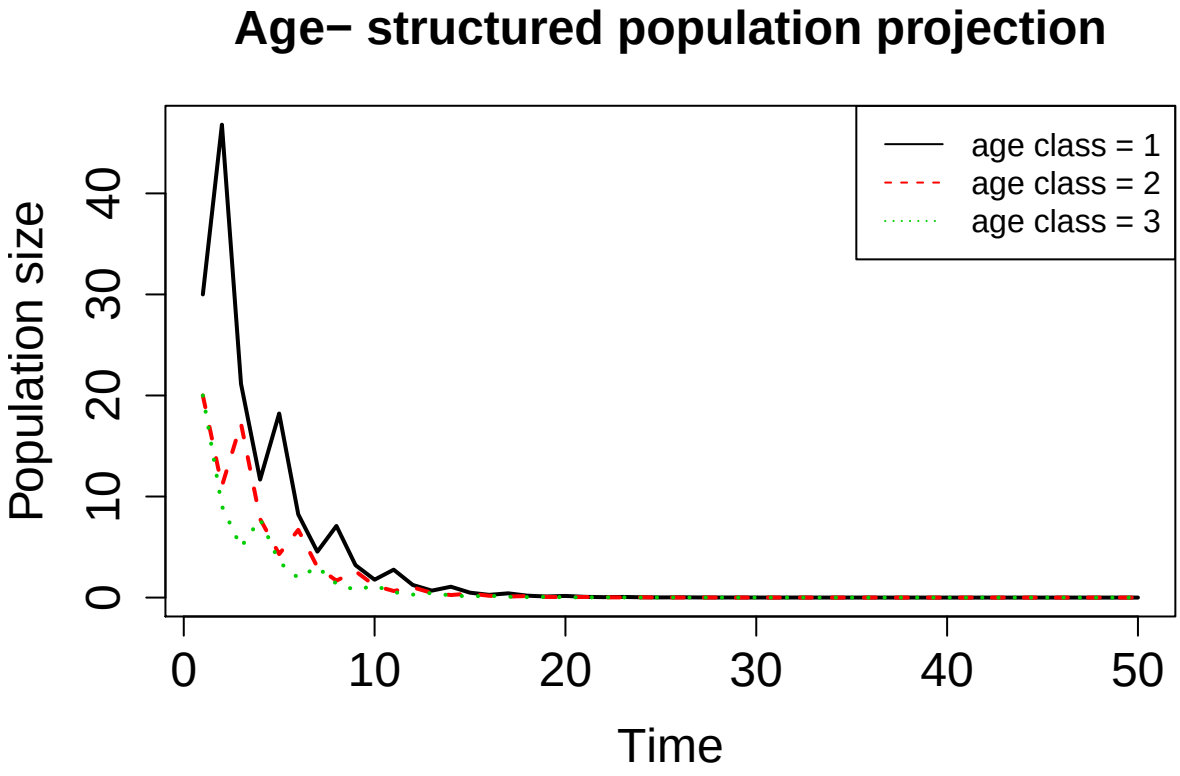
```
A.tiger
```

```
##           cubs subadult adult
## cubs     0.000    0.000  2.34
## subadult 0.368    0.000  0.00
## adult    0.000    0.452  0.00
```

```
n0 =c(30,20,20)
results = pop.projection(A.tiger, n0, iterations = 50)
names(results)
```

```
## [1] "lambda"        "stable.stage"  "stage.vectors" "pop.sizes"
## [5] "pop.changes"
```

```
####lets project this pop for 10 years
N_vec=t(results$stage.vectors) #
#t transposes the matrix so we can
#use matplot which plots columns of a matrix
N =results$pop.sizes
t_max=50
time=1:t_max
age_class = length(N_vec[1,])
matplot(time,N_vec,type='l',xlab = "Time",
ylab = "Population size", cex = 1.5,
cex.main = 1.5, cex.lab = 1.5, lwd=2,
cex.axis = 1.5,lty=c(1:age_class),
col = c(1:age_class), main = "Age- structured population projection")
legend("topright", legend = paste
  ("age class = ",1:age_class, sep = "" ),
lty = c(1:age_class), col = c(1:age_class))
```



Some basic matrix properties

```
lambda(whale) #population growth rate
```

```
## [1] 1.025441
```

```
#finite population growth rate is the dominant
#(i.e., largest) eigenvalue of the population projection matrix A

stable.stage(whale) #stable stage distribution
```

```
##   yearling   juvenile     mature postreprod
## 0.03697187 0.31607121 0.32290968 0.32404724
```

```
#proportional stable stage distribution
#the stable age/stage distribution is the
#right eigenvector corresponding to the dominant eigenvalue (usually scaled to sum to 1)

reproductive.value(whale) #reproductive value
```

```
##     yearling     juvenile       mature   postreprod
## 1.000000e+00 1.049045e+00 1.571320e+00 6.002137e-16
```

```
#reproductive value vector is the left eigenvector corresponding to the dominant eigenvalue
#(usually scaled such that reproductive value of the youngest age class or stage is 1)

net.reproductive.rate(whale) #net reproductive rate
```

```
## [1] 2.013144
```

```
damping.ratio(whale) #damping ratio
```

```
## [1] 1.04594
```

```
#The damping ratio is calculated by dividing
#the dominant eigenvalue by the eigenvalue
#with the second largest magnitude.

generation.time(whale) #generation time
```

```
## [1] 27.85079
```

```
##########All of these at the same time
eigen.analysis(whale)
```

```
## $lambda1
## [1] 1.025441
##
## $stable.stage
##   yearling   juvenile     mature postreprod
## 0.03697187 0.31607121 0.32290968 0.32404724
##
## $sensitivities
##                yearling     juvenile       mature   postreprod
## yearling   4.220825e-02 3.608369e-01 3.686439e-01 3.699426e-01
## juvenile   4.427835e-02 3.785341e-01 3.867240e-01 3.880864e-01
## mature     6.632269e-02 5.669904e-01 5.792577e-01 5.812983e-01
## postreprod 2.533397e-17 2.165792e-16 2.212651e-16 2.220446e-16
##
## $elasticities
##              yearling    juvenile       mature   postreprod
## yearling   0.00000000 0.001513103 4.069515e-02 0.000000e+00
## juvenile   0.04220825 0.336325827 0.000000e+00 0.000000e+00
## mature     0.00000000 0.040695151 5.385625e-01 0.000000e+00
## postreprod 0.00000000 0.000000000 9.753053e-18 2.122916e-16
##
## $repro.value
##     yearling     juvenile       mature   postreprod
## 1.000000e+00 1.049045e+00 1.571320e+00 6.002137e-16
##
## $damping.ratio
## [1] 1.04594
```

Stochasticity

```
data(hudsonia)
lambdas<-sapply(hudsonia, lambda)
lambdas
```

```
##       A85       A86       A87       A88
## 0.9593438 1.0098094 0.8453119 1.0183199
```

```
#with equal probabilities:
#Calculates the log stochastic growth rate by Tuljapukar's approximation and by simulation
```

```
sgr<-stoch.growth.rate(hudsonia, verbose=F)
sgr
```

```
## $approx
## [1] -0.03712598
##
## $sim
## [1] -0.03658441
##
## $sim.CI
## [1] -0.03716281 -0.03600602
```

```
exp(sgr$approx) #transformation
```

```
## [1] 0.9635547
```

```
#with unequal probabilities:
sgr_unequal=stoch.growth.rate(hudsonia, prob=c(0.2,0.2,0.4,0.2),verbose=F)
sgr_unequal
```

```
## $approx
## [1] -0.05693125
##
## $sim
## [1] -0.05619419
##
## $sim.CI
## [1] -0.05676914 -0.05561924
```

```
#stochastic population projection:
n<- c(4264, 3,30,16,25,5) #count data
t_max=50
?stoch.projection
### compare equal and unequal probabilities for matrix selection:
x.eq<-stoch.projection(hudsonia, n, nreps=1000)
head(x.eq) ##environmental conditions same across all years
```

```
##          [,1]      [,2]      [,3]      [,4]      [,5]     [,6]
## [1,] 772.8831 0.4616260 3.0057583 2.0502570 3.6821609 5.717136
## [2,] 322.6155 0.1973371 1.2542062 1.1778261 1.6524910 1.355533
## [3,] 794.0860 0.6510262 2.5084212 3.3072262 4.0077767 3.346403
## [4,] 520.0023 0.4276197 1.2531436 1.6969464 2.0552108 2.777347
## [5,] 229.4614 0.1888255 0.7508023 0.7529485 0.5956937 2.034822
## [6,] 886.6126 0.5291913 2.2974227 1.1329297 3.7001991 7.785487
```

```
N_eq= rowSums(x.eq)
N_eq=sum(N_eq)
N_eq
```

```
## [1] 621852.6
```

```
###Unequal environment probability
x.uneq<-stoch.projection(hudsonia, n, nreps=1000, prob=c(0.2,0.2,0.4,0.2))
head(x.uneq)
```

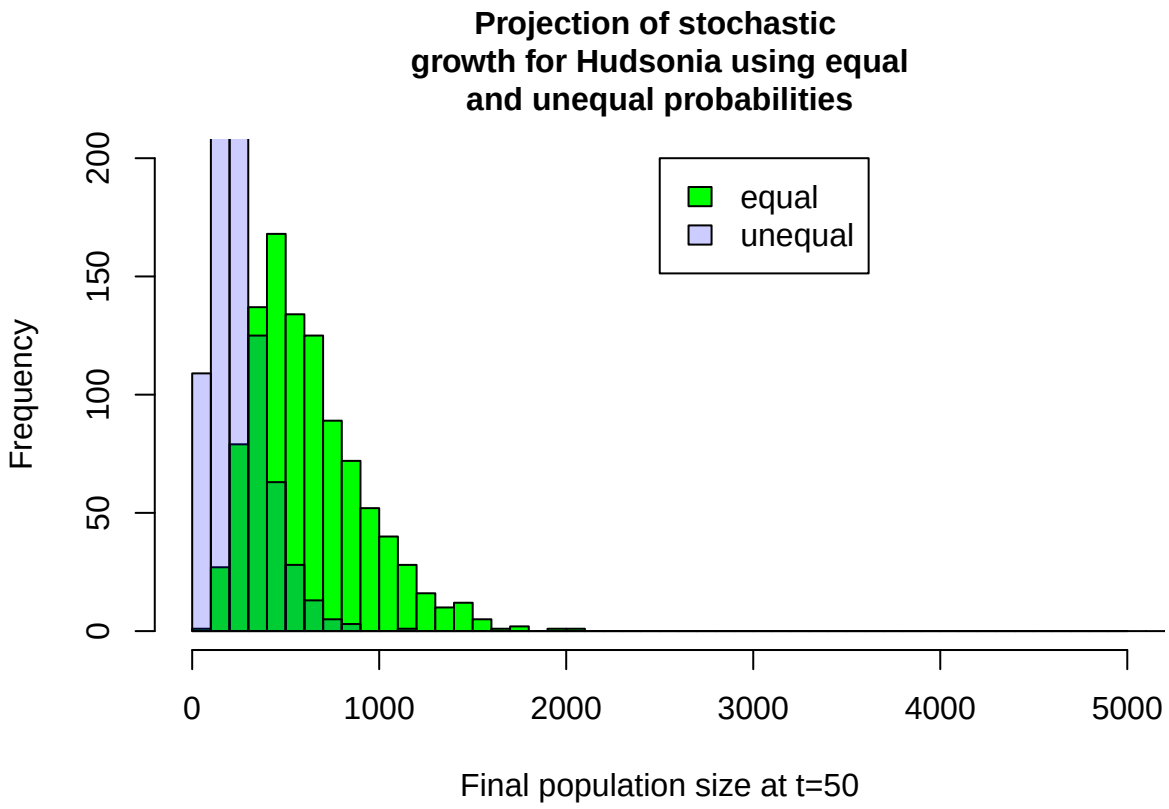
```
##          [,1]       [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 180.9129 0.14865059 0.6607416 0.6740252 0.5315768 1.4581284
## [2,] 113.1819 0.06919554 0.6674977 0.5847809 0.6022559 0.4082728
```

```
## [3,] 300.7475 0.24638931 1.6583241 1.6158100 2.0225663 0.9911860
## [4,] 287.2206 0.23546905 1.1304863 1.2614480 1.5980577 1.1333290
## [5,] 129.2757 0.10604214 0.4401584 0.4770480 0.4670704 0.9520202
## [6,] 135.9997 0.11190322 0.3103136 0.4602251 0.5002611 0.7643536
```

```
N_uneq= rowSums(x.uneq)
N_uneq=sum(N_uneq)
N_uneq
```

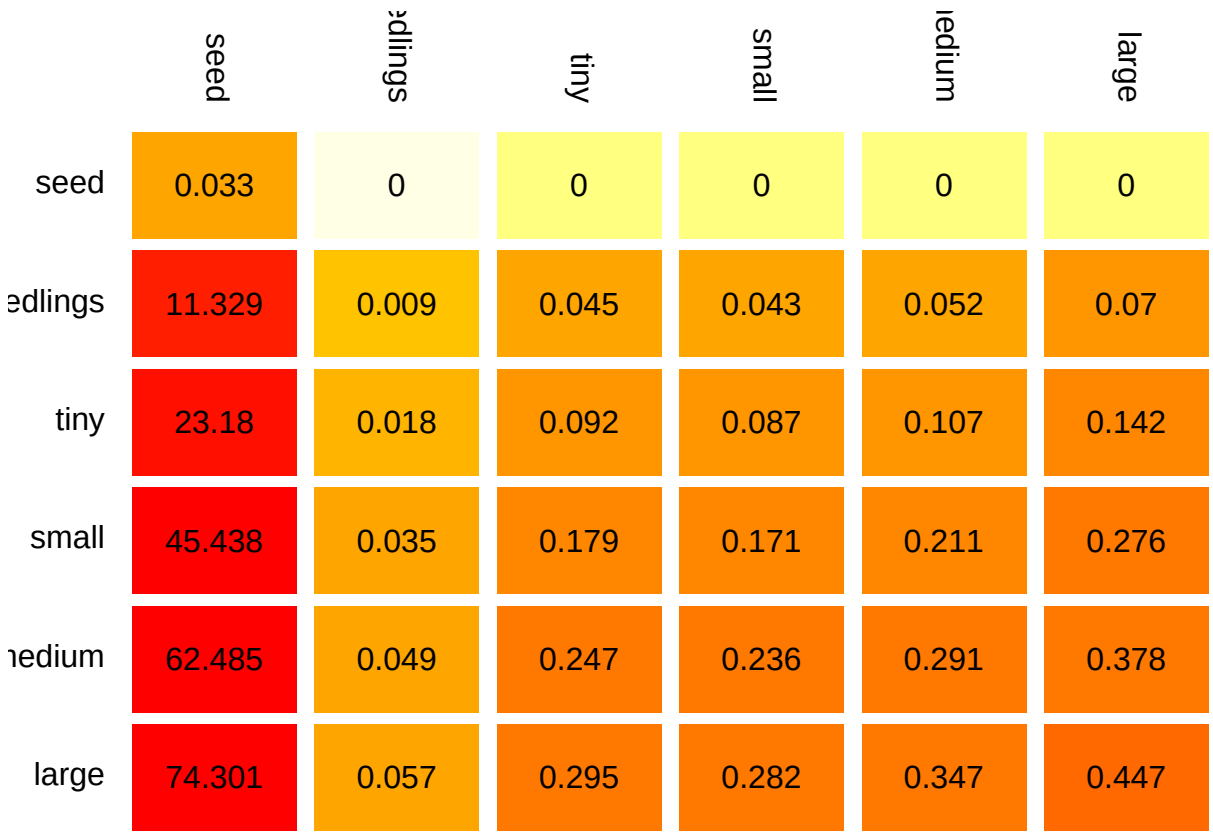
```
## [1] 234299.4
```

```
#environmental conditions of last year may be more frequent
##example- hot years may be more frequent in the future
hist(apply(x.eq, 1, sum),
xlim=c(0,5000), ylim=c(0,200), col="green", breaks=seq(0,5000, 100),
xlab="Final population size at t=50", main="")
par(new=TRUE)
hist(apply(x.uneq, 1, sum),
xlim=c(0,5000), ylim=c(0,200),
col = rgb(0, 0, 1, 0.2),
xaxt='n', yaxt='n', ylab='', xlab='',
breaks=seq(0,10000, 100), main='')
legend(2500,200, c("equal", "unequal"),
fill=c("green", rgb(0, 0, 1, 0.2)))
title(paste("Projection of stochastic
growth for Hudsonia using equal
and unequal probabilities"), cex.main=1)
```



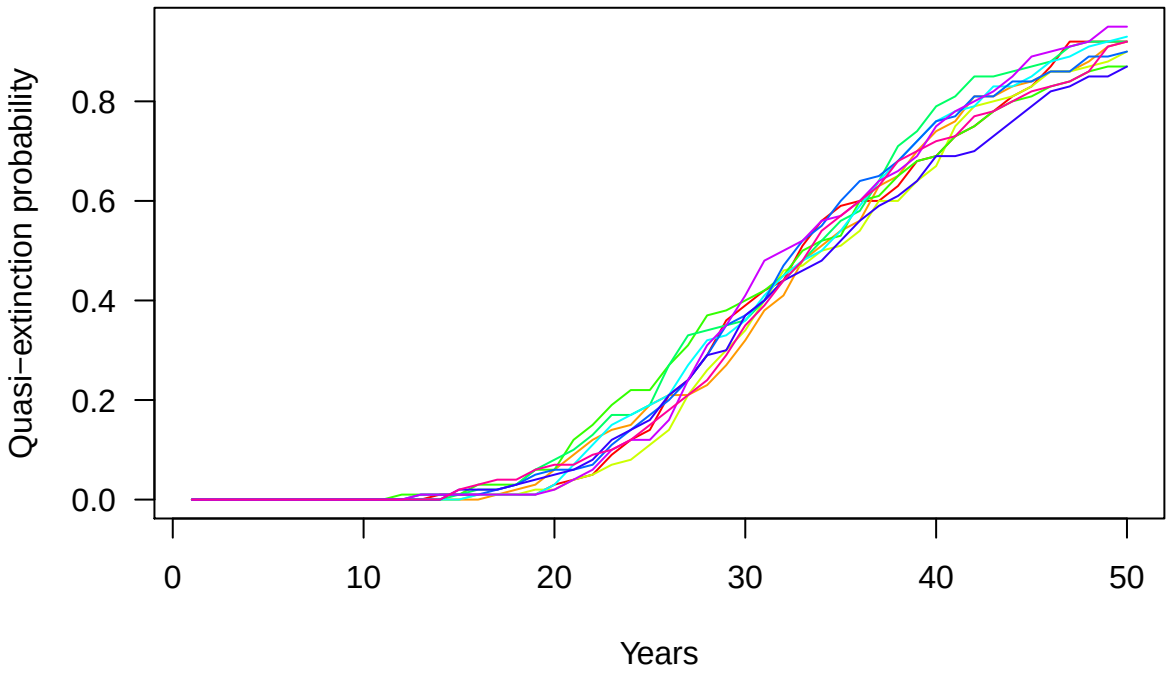

```
#stochastic sensitivities:
stsens=stoch.sens(hudsonia)
```

```
#Calculates the sensitivity of the stochastic
#growth rate to perturbations
#in the mean demographic projection matrix.
stsens= stoch.sens(hudsonia)
row.names(stsens$sensitivities)=
  colnames(stsens$sensitivities)
image2(stsens$sensitivities)
```




```
# (quasi)-extinction risk:
n <- c(4264, 3, 30, 16, 25, 5)
Nx=20 #critical threshold
# exclude seeds using sumweight:

x <- stoch.quasi.ext(hudsonia, n,
Nx=Nx, nreps=100, sumweight=c(0,1,1,1,1,1),
verbose=F)
matplot(x, xlab="Years", ylab="Quasi-extinction probability",
type='l', lty=1, col=rainbow(10), las=1)
```


Quasi−extinction probability
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50
Years