Two-site_matrix_models_for_black-headed_gulls.R

akane

Fri Aug 11 12:19:49 2017

```
# Matrix Models for Population Management & Conservation
# 2014
# CNRS CEFE workshop, Montpellier, France
# Jean-Dominique Lebreton, Olivier Gimenez, Dave Koons

# EXERCISE 3: Two-site matrix models for black-headed gulls

rm(list=ls(all=TRUE))   # clears the R memory, which is sometimes useful

################################################################################
# Piece 1 of code                                                              #
################################################################################
library(MASS) # an R package needed for some matrix functions
library(quadprog)
```

```
## Warning: package 'quadprog' was built under R version 3.3.2
```

```
library(popbio)

# First, define the demographic parameters
# Good site
sg0 <- 0.4
sg1 <- 0.6
s <- 0.82  # adult survival common to both sites
fg <- 0.8  # half the mean clutch size (modeling females only)
pg2 <- 0.3 # proportion of adults that attempt to breed each year at age 2, etc.
pg3 <- 0.5
pg4 <- 0.7
pg5 <- 1
# Bad sites
sb0 <- 0.4
sb1 <- 0.5
fb <- 0.5  # half the mean clutch size at Bad sites
pb2 <- 0.5
pb3 <- 0.8
pb4 <- 1
pb5 <- 1
# Effective migration rates (juvenile dispersal * their survival)
s1gb <- 0.2 # Good to Bad
s1bg <- 0.3 # Bad to Good

# Next, create the two-site pre birth-pulse matrix model for black-headed gulls
A <- matrix(c(
  0, sg0*fg*pg2, sg0*fg*pg3, sg0*fg*pg4, sg0*fg*pg5, 0, 0, 0, 0, 0,
  sg1, 0, 0, 0, 0, s1bg, 0, 0, 0, 0,
  0, s, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, s, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, s, s, 0, 0, 0, 0, 0,
```

```
  0, 0, 0, 0, 0, 0, sb0*fb*pb2, sb0*fb*pb3, sb0*fb*pb4, sb0*fb*pb5,
  s1gb, 0, 0, 0, 0, sb1, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, s, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, s, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, s, s), nrow = 10, byrow = TRUE)
A
```

```
##       [,1]  [,2] [,3]  [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  0.0 0.096 0.16 0.224 0.32  0.0 0.00 0.00 0.00  0.00
##  [2,]  0.6 0.000 0.00 0.000 0.00  0.3 0.00 0.00 0.00  0.00
##  [3,]  0.0 0.820 0.00 0.000 0.00  0.0 0.00 0.00 0.00  0.00
##  [4,]  0.0 0.000 0.82 0.000 0.00  0.0 0.00 0.00 0.00  0.00
##  [5,]  0.0 0.000 0.00 0.820 0.82  0.0 0.00 0.00 0.00  0.00
##  [6,]  0.0 0.000 0.00 0.000 0.00  0.0 0.10 0.16 0.20  0.20
##  [7,]  0.2 0.000 0.00 0.000 0.00  0.5 0.00 0.00 0.00  0.00
##  [8,]  0.0 0.000 0.00 0.000 0.00  0.0 0.82 0.00 0.00  0.00
##  [9,]  0.0 0.000 0.00 0.000 0.00  0.0 0.00 0.82 0.00  0.00
## [10,]  0.0 0.000 0.00 0.000 0.00  0.0 0.00 0.00 0.82  0.82
```

```
# Then use the following popbio function; quite easy!
lambda(A)
```

```
## [1] 0.9970859
```

```
# The following code demonstrates the matrix algebra that
# is used 'behind the scences' of the lambda function in the popbio package.
rows <- dim(A)[1]
cols <- dim(A)[2]
eig <- eigen(A)           # eigenvalues of A
EigVecs <- eig$vectors    # eigenvectors of A
Lambdas <- Re(eig$values) # real number components of eigenvalues
Lambda <- max(Lambdas)    # long-term geometric rate of population growth
Lambda
```

```
## [1] 0.9970859
```

```
pos <- which.max(Lambdas)  # finding the position of the dominant eigenvalue
w <- Re(eig$vectors[1:rows,pos]) # its associated right eigenvector
w
```

```
##  [1] -0.32790653 -0.23670566 -0.19466592 -0.16009259 -0.74131229
##  [6] -0.13090649 -0.13141752 -0.10807732 -0.08888242 -0.41157200
```

```
sad <- w/(sum(w))
sad <- round(sad,3) # scaled dominant right eigenvector: Stable Age Distribution
# In the following, the ginv function inverts a matrix by calculating the
# Moore-Penrose generalized inverse of a matrix.
V <- Conj(ginv(EigVecs))   # left eigenvector; NOTE this notation from H Caswell
v <- Re(t(t(V[pos,])))     # dominant left eigenvector
v
```

```
##             [,1]
##  [1,] -0.2975160
##  [2,] -0.4121467
##  [3,] -0.4663221
##  [4,] -0.5089763
##  [5,] -0.5376213
##  [6,] -0.2477686
```

```
##  [7,] -0.2468051
##  [8,] -0.2698891
##  [9,] -0.2798288
## [10,] -0.2798288
```

```
rv <- v/(sum(v))
rv <- round(rv,3)          # scaled to provide proportional Reproductive Values
sad
```

```
##  [1] 0.130 0.094 0.077 0.063 0.293 0.052 0.052 0.043 0.035 0.163
```

```
rv
```

```
##        [,1]
##  [1,] 0.084
##  [2,] 0.116
##  [3,] 0.131
##  [4,] 0.144
##  [5,] 0.152
##  [6,] 0.070
##  [7,] 0.070
##  [8,] 0.076
##  [9,] 0.079
## [10,] 0.079
```

```
# Conduct a sensitivity and elasticity analysis for the lower-level vital rates
# (i.e, those that make up the matrix elements) using the popbio package.
# Just put the vital rates in a list, and write the matrix as an expression
gull.vr <- list(sg0=0.4,sg1=0.6,s=0.82,fg=0.8,pg2=0.3,pg3=0.5,pg4=0.7,pg5=1,
                sb0=0.4,sb1=0.5,fb=0.5,pb2=0.5,pb3=0.8,pb4=1,pb5=1,s1gb=0.2,s1bg=0.3)

gull.A <- expression(
  0, sg0*fg*pg2, sg0*fg*pg3, sg0*fg*pg4, sg0*fg*pg5, 0, 0, 0, 0, 0,
  sg1, 0, 0, 0, 0, s1bg, 0, 0, 0, 0,
  0, s, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, s, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, s, s, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, sb0*fb*pb2, sb0*fb*pb3, sb0*fb*pb4, sb0*fb*pb5,
  s1gb, 0, 0, 0, 0, sb1, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, s, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, s, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, s, s)

# then apply the following popbio function
llsenselas <- vitalsens(gull.A,gull.vr)
llsenselas
```

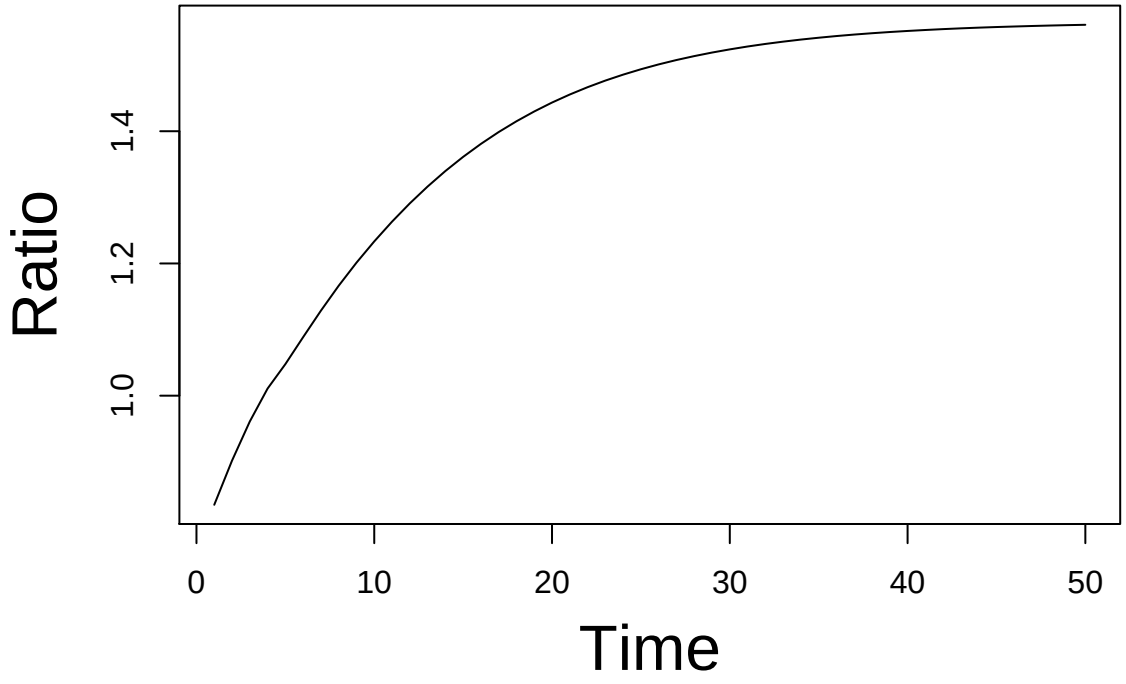
```
##      estimate sensitivity  elasticity
## sg0       0.40 0.243182886 0.097557451
## sg1       0.60 0.135145587 0.081324343
## s         0.82 0.899828725 0.740016066
## fg        0.80 0.121591443 0.097557451
## pg2       0.30 0.022535593 0.006780437
## pg3       0.50 0.018533195 0.009293680
## pg4       0.70 0.015241636 0.010700327
## pg5       1.00 0.070576734 0.070783006
## sb0       0.40 0.080849994 0.032434516
```

```
## sb1      0.50 0.032308391 0.016201409
## fb       0.50 0.064679995 0.032434516
## pb2      0.50 0.006512227 0.003265630
## pb3      0.80 0.005355633 0.004297029
## pb4      1.00 0.004404454 0.004417327
## pb5      1.00 0.020394924 0.020454531
## s1gb     0.20 0.080929011 0.016233108
## s1bg     0.30 0.053952674 0.016233108
```

```
##############################################################################
# Piece 2 of code                                                            #
##############################################################################
ratio <- numeric()  # a storage bin for holding calculations of a ratio
# to be interpreted for question 5
n <- matrix(1,10,1) # a vector with an initial abundance of 1 individual per
# age and location
tspan <- 50
for (t in 1:tspan){
  n <- A%*%n        # %*% = matrix multiplication in R
  # note that we simply overwrite the abundance vector at each time step
  # since we have no need to store it for these questions. It is updating.
  Nbreedg <- pg2*n[2]+pg3*n[3]+pg4*n[4]+pg5*n[5]
  Nbreedb <- pb2*n[7]+pb3*n[8]+pb4*n[9]+pb5*n[10]
  ratio[t] <- Nbreedg/Nbreedb  # this we store
}
ratio
```

```
##  [1] 0.8349683 0.9017267 0.9609764 1.0107846 1.0476793 1.0885523 1.1285981
##  [8] 1.1662590 1.2011606 1.2333482 1.2632080 1.2908452 1.3163311 1.3397523
## [15] 1.3612152 1.3808409 1.3987505 1.4150629 1.4298941 1.4433568 1.4555596
## [22] 1.4666058 1.4765932 1.4856134 1.4937523 1.5010894 1.5076986 1.5136477
## [29] 1.5189994 1.5238107 1.5281340 1.5320170 1.5355031 1.5386317 1.5414384
## [36] 1.5439556 1.5462126 1.5482358 1.5500490 1.5516737 1.5531292 1.5544329
## [43] 1.5556005 1.5566461 1.5575824 1.5584206 1.5591709 1.5598426 1.5604439
## [50] 1.5609820
```

```
par(mar = c(5, 6, 4, 2))
plot(1:tspan,ratio,type="l",xlab=list("Time",cex=2),ylab=list("Ratio",cex=2))
```




```
################################################################################
# Piece 3 of code                                                              #
################################################################################
# Define adult dispersal probabilities
gb <- 0 # Good to Bad
bg <- 0 # Bad to Good

A <- matrix(c(
  0, sg0*fg*pg2, sg0*fg*pg3, sg0*fg*pg4, sg0*fg*pg5, 0, 0, 0, 0, 0,
  sg1, 0, 0, 0, 0, s1bg, 0, 0, 0, 0,
  0, s*(1-gb), 0, 0, 0, 0, s*bg, 0, 0, 0,
  0, 0, s*(1-gb), 0, 0, 0, 0, s*bg, 0, 0,
  0, 0, 0, s*(1-gb), s*(1-gb), 0, 0, 0, s*bg, s*bg,
  0, 0, 0, 0, 0, 0, sb0*fb*pb2, sb0*fb*pb3, sb0*fb*pb4, sb0*fb*pb5,
  s1gb, 0, 0, 0, 0, sb1, 0, 0, 0, 0,
  0, s*gb, 0, 0, 0, 0, s*(1-bg), 0, 0, 0,
  0, 0, s*gb, 0, 0, 0, 0, s*(1-bg), 0, 0,
  0, 0, 0, s*gb, s*gb, 0, 0, 0, s*(1-bg), s*(1-bg)), nrow = 10, byrow = TRUE)

ratio <- numeric()  # a storage bin for holding calculations of a ratio
# to be interpreted for question 5
n <- matrix(1,10,1) # a vector with an initial abundance of 1 individual per
# age and location
tspan <- 50
for (t in 1:tspan){
  n <- A%*%n        # %*% = matrix multiplication in R
```

```
  # note that we simply overwrite the abundance vector at each time step
  # since we have no need to store it for these questions. It is updating.
  Nbreedg <- pg2*n[2]+pg3*n[3]+pg4*n[4]+pg5*n[5]
  Nbreedb <- pb2*n[7]+pb3*n[8]+pb4*n[9]+pb5*n[10]
  ratio[t] <- Nbreedg/Nbreedb  # this we store
}

ratio
```

```
##  [1] 0.8349683 0.9017267 0.9609764 1.0107846 1.0476793 1.0885523 1.1285981
##  [8] 1.1662590 1.2011606 1.2333482 1.2632080 1.2908452 1.3163311 1.3397523
## [15] 1.3612152 1.3808409 1.3987505 1.4150629 1.4298941 1.4433568 1.4555596
## [22] 1.4666058 1.4765932 1.4856134 1.4937523 1.5010894 1.5076986 1.5136477
## [29] 1.5189994 1.5238107 1.5281340 1.5320170 1.5355031 1.5386317 1.5414384
## [36] 1.5439556 1.5462126 1.5482358 1.5500490 1.5516737 1.5531292 1.5544329
## [43] 1.5556005 1.5566461 1.5575824 1.5584206 1.5591709 1.5598426 1.5604439
## [50] 1.5609820
```