ESM 232 - Rabbit Population Matrix

Jennifer Truong, Maggie Brickner, and Mauricio Collado

05/14/2021

1. Introduction

A small city with a large urban park has decided to introduce a rare species of rabbits into this park - Rabbits are cute and the kids love them, and giving a rare species a new home sounds like a good idea. The urban park manager is concerned about how this rabbit population might grow over the next few decades. Rabbits have no natural predators in the region where the park is situated. The manager would like to know, approximately, how many rabbits there will be 20 years from now if the rabbits are introduced as planned. The manager reviewed the literature and found the following estimates for survival and fertility rates for the rare rabbit population, for 4 different age classes.

2. Using Leslie Matrices to Evolve Populations

Our first step is building a square matrix that holds the fertility and survivability for four age classes (young, sub-adults, adults and aged) of rabbit population.

```
# Set up number of age classes
nclasses = 4

# create a growth matrix to store fertility and survivability information
gmatrix=matrix(nrow=nclasses, ncol=nclasses)
#gmatrix

# change NAs to zero
gmatrix[]=0.0
#gmatrix

# assign values for fertility for each age class
# fertility numbers are big here because they are RABBITS!!
fert =  c(0,2,6,1)

# enter into our matrix
gmatrix[1,]=fert

 # now add survivability
 # survivability (to the next class) is also per time step
gmatrix[2,1]=0.8
gmatrix[3,2]=0.85
gmatrix[4,3]=0.65

# we also want to to account for the oldest population group - they don't transfer to another group
# but they do die - this will be survivability per time step but they just stay in their class/group
gmatrix[4,4]=0.1

gmatrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  0.0 2.00 6.00  1.0
## [2,]  0.8 0.00 0.00  0.0
## [3,]  0.0 0.85 0.00  0.0
## [4,]  0.0 0.00 0.65  0.1
```

We test our matrix values for the populations in the years 1 and 2. We observe a reduced population for adults and aged groups in years 1 and 2.

```
# start with an initial population of 10 adult rabbits
p0 = rep(10, times=nclasses)

# advance to the next time step
# note the use of matrix multiplication
p1 = gmatrix %*% p0
p1
```

```
##      [,1]
## [1,] 90.0
## [2,]  8.0
## [3,]  8.5
## [4,]  7.5
```

```
# has the total number of individuals changed?
sum(p1)
```

```
## [1] 114
```

```
sum(p0)
```

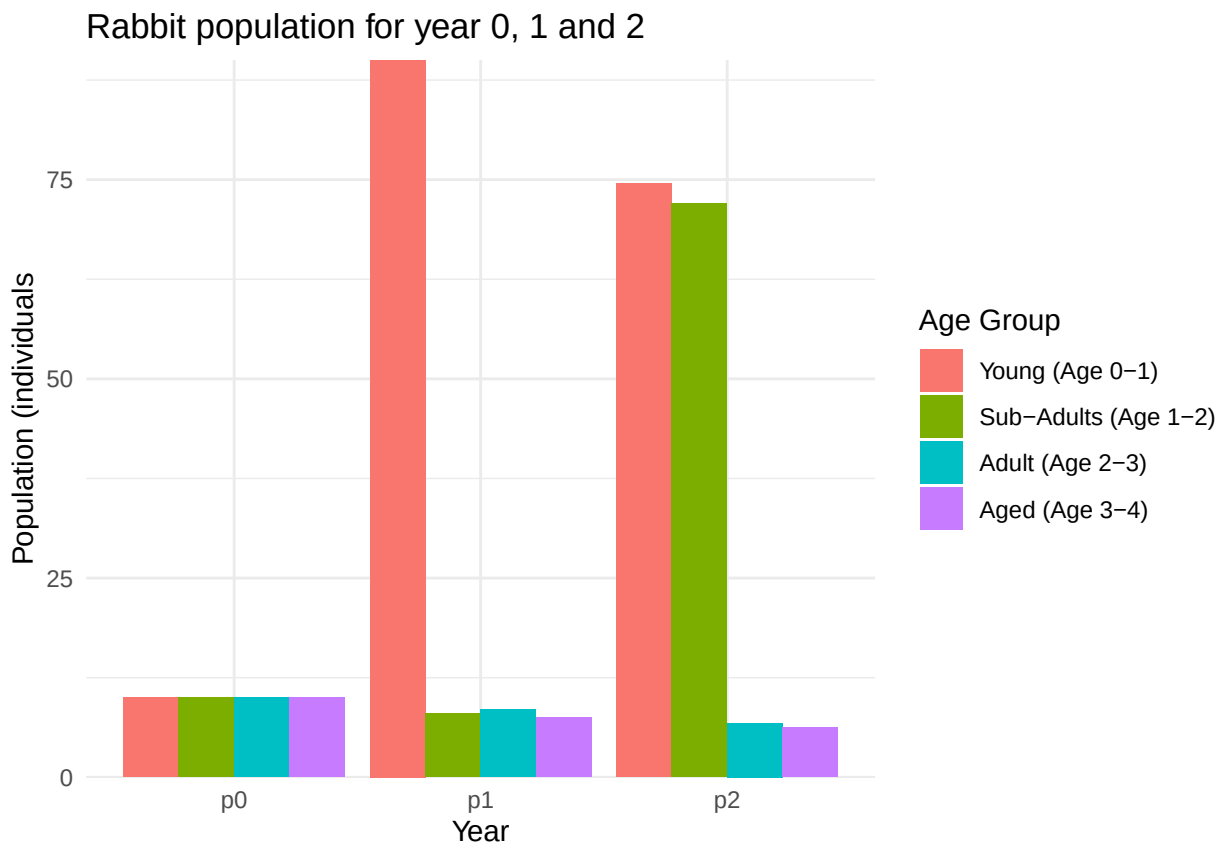
```
## [1] 40
```

```
# growth rate
sum(p1)/sum(p0)
```

```
## [1] 2.85
```

```
#add another year
p2 = gmatrix %*% p1

# combined
pop = cbind.data.frame(p0,p1,p2)
pop$age = c("Young (Age 0-1)",
            "Sub-Adults (Age 1-2)",
            "Adult (Age 2-3)",
            "Aged (Age 3-4)") %>%
  as_factor() # Change to factor class

popl = pop %>%
  gather(key="timestep",value="pop",-age)

ggplot(popl, aes(timestep, pop,fill=as.factor(age)))+
  geom_col(position="dodge")+
  labs(title="Rabbit population for year 0, 1 and 2",
       y="Population (individuals",
       x="Year",
       fill="Age Group") +
```

```
  theme_minimal() +
  scale_y_continuous(expand = c(0,0))
```



3. Rabbit Population in 20 Years

We use function to evolve a population through time considering:

- inputs = survivability, fertility, initial population, time steps
- output = final population matrix
- a dynamic model - difference equations - similar to our diffusion model

```
# call the evolve population function
source(here("R/evolve_pop.R"))

# fertility rates
F1 = 0
F2 = 2
F3 = 6
F4 = 1

# survivability
p12 = 0.8
p23 = 0.85
p34 = 0.65
p44 = 0.1

# initial population parameters
ini = c(0, 0, 10, 0) # start with 10 adult rabbits
```

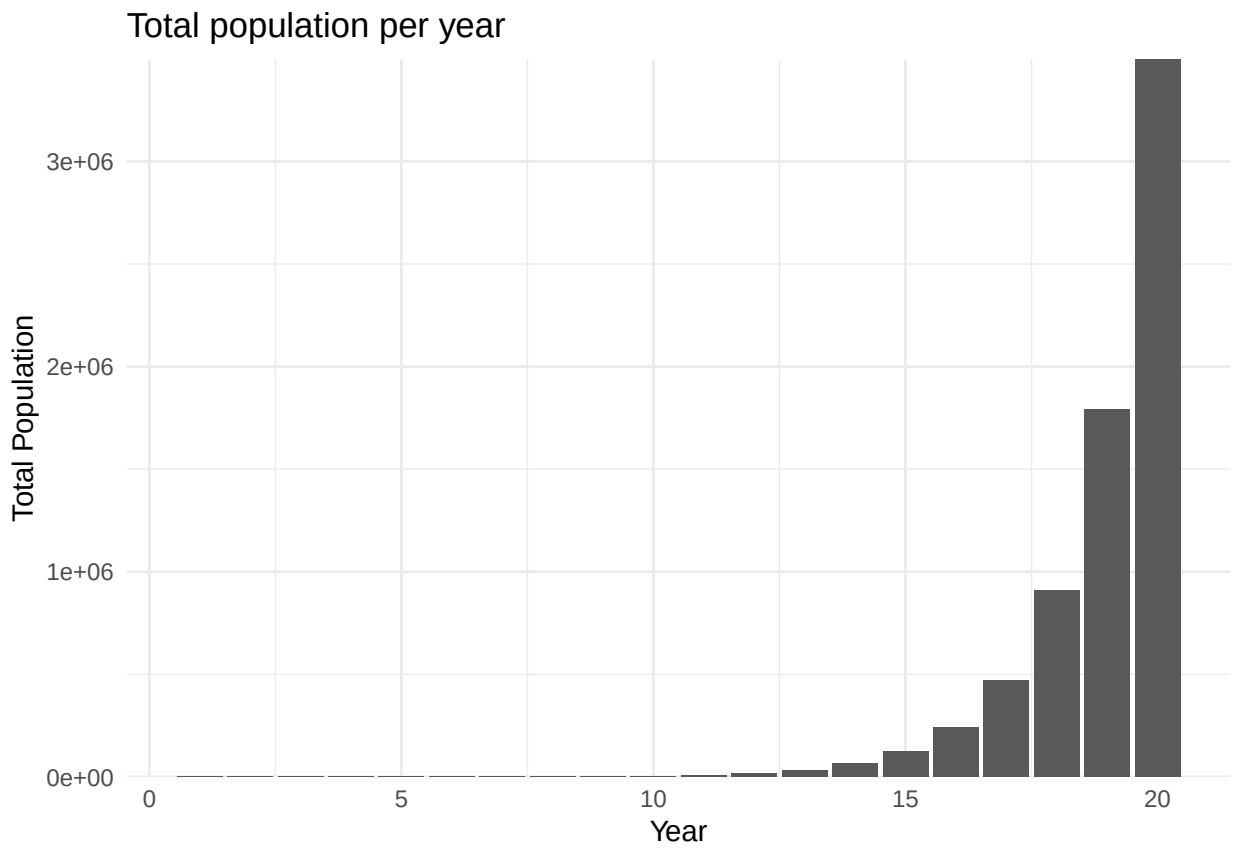
```r
nyears = 20 # number of years (time step) to run
fert_rabbit = c(F1, F2, F3, F4) # fertility for each age class
surv_rabbit = c(p12, p23, p34, p44) # survivability for each age class

#run the equation evolve_pop(fertility, survivability, initial pop, years)
rabbit_pop=evolve_pop(fert_rabbit, surv_rabbit, ini, nyears)

#check the results
head(rabbit_pop)
```

```
## $popbyage
##      [,1] [,2]  [,3]   [,4]     [,5]      [,6]      [,7]      [,8]      [,9]
## [1,]    0 60.0  6.50 96.650 255.2650 207.68650 808.28165 1417.0515 2257.7659
## [2,]    0  0.0 48.00  5.200  77.3200 204.21200 166.14920  646.6253 1133.6412
## [3,]   10  0.0  0.00 40.800   4.4200  65.72200 173.58020  141.2268  549.6315
## [4,]    0  6.5  0.65  0.065  26.5265   5.52565  43.27187  117.1543  103.5129
##          [,10]     [,11]     [,12]     [,13]     [,14]      [,15]     [,16]
## [1,] 5668.5843 9761.6072 18944.518 39810.638 72750.524 145566.477 287659.30
## [2,] 1806.2127 4534.8675  7809.286 15155.614 31848.510  58200.419 116453.18
## [3,]  963.5950 1535.2808  3854.637  6637.893 12882.272  27071.234  49470.36
## [4,]  367.6118  663.0979  1064.242  2611.939  4575.824   8831.059  18479.41
##          [,17]      [,18]     [,19]     [,20]
## [1,] 548207.91 1088169.78 2118523.3 4111679.4
## [2,] 230127.44  438566.33  870535.8 1694818.7
## [3,]  98985.20  195608.32  372781.4  739955.4
## [4,]  34003.67   67740.75  133919.5  255699.8
##
## $poptot
##  [1]       0.0000      10.0000      66.5000      55.1500     142.7150
##  [6]     363.5315     483.1462    1191.2829    2322.0579    4044.5514
## [11]    8806.0038   16494.8533   31672.6830   64216.0834  122057.1304
## [16]  239669.1889  472062.2456  911324.2250 1790085.1785 3495760.0330
```

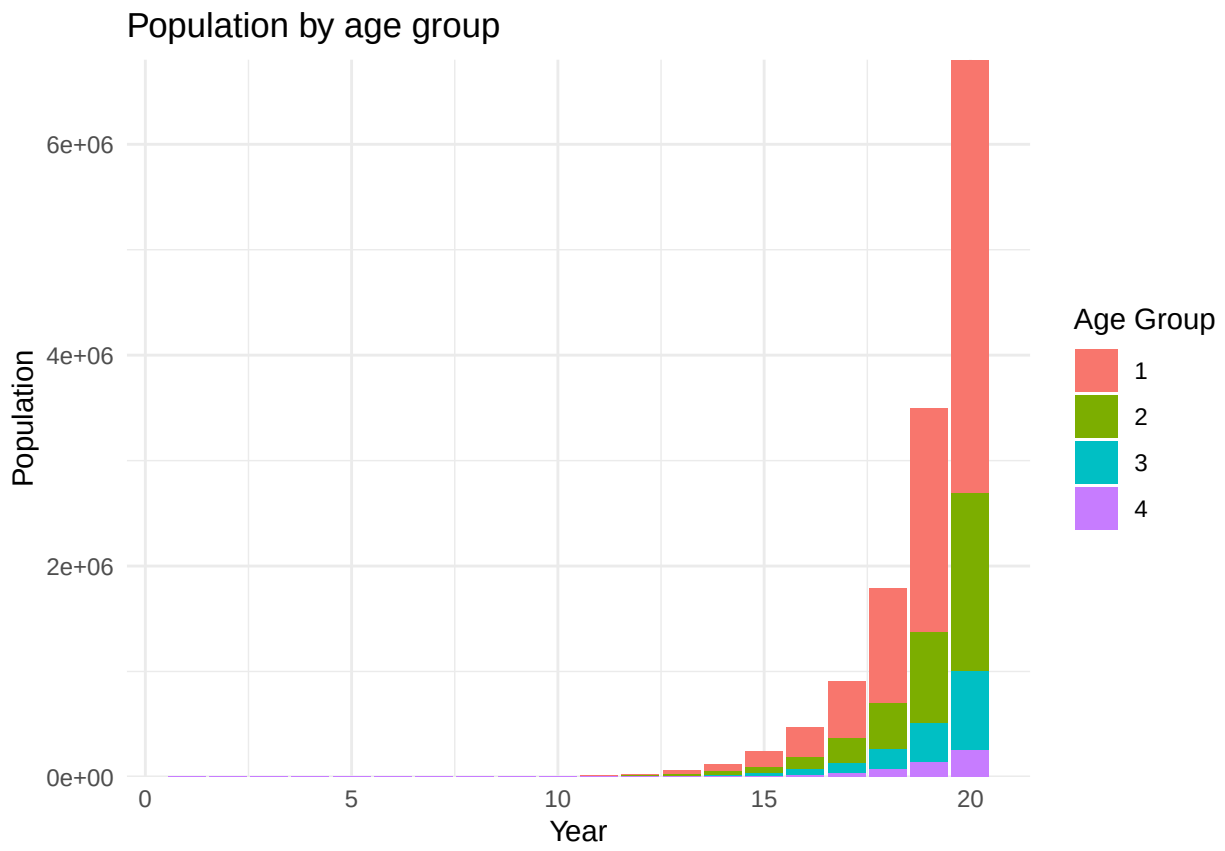
```r
# keep the results for each decade
# graph different components of the output
```

In 20 years, the total rabbit population reach 66.5 individuals. Also in this year, 60 individuals belong to the first age class (young).

```r
# add year
year = seq(from=1, to=nyears)

# total population kept in dataframe
rabbit_tot = cbind.data.frame(year=year, poptot=rabbit_pop$poptot)

# plot total population per decade
ggplot(rabbit_tot, aes(year, poptot))+
  geom_col()+
  labs(title="Total population per year",
       y="Total Population",
       x="Year") +
  theme_minimal() +
  scale_y_continuous(expand = c(0,0))
```




```
# population by age group kept in dataframe
rabbit_ages = cbind.data.frame(year=year, t(rabbit_pop$popbyage))

rabbit_agesl = rabbit_ages %>%
  gather(key="agecat", value="pop",-year)

# plot information about ages
ggplot(rabbit_agesl, aes(year, pop, fill=agecat))+
  geom_col()+
  labs(title="Population by age group",
       y="Population",
       x="Year",
       fill="Age Group") +
  theme_minimal() +
  scale_y_continuous(expand = c(0,0))
```



4. Rabbit Population with Hawks

Hawks generally only eat younger rabbits- thus they reduce the survivability of the young and sub-adults age classes (the first two classes). The estimates are that survivability reduced to between 0.65 and 0.75 for Ages 0-1 and between 0.75 and 0.8 for Ages 1-2. You can assume that distributions are uniform

Our first step is to generate the samples for the Sobel analysis.

```
library(sensitivity)

# survivability - based on mortality rates per thousand per decade
nsample=200

# fertility rates
F1 = 0
F2 = 2
F3 = 6
F4 = 1

# survivability
p12 = 0.8 #original value
p23 = 0.85 # original value
p34 = 0.65
p44 = 0.1


# we do not vary our ferility parameters
```

```
fs = cbind.data.frame(F1=F1,
                      F2=F2,
                      F3=F3,
                      F4=F4)

# create our two samples for Sobel
# first do our survivability
ps1 = cbind.data.frame(p12 = runif(min=0.65, max=0.75, n=nsample),
                       p23 = runif(min=0.75, max=0.8, n=nsample),
                       p34 = p34,
                       p44 = p44)

ps2 = cbind.data.frame(p12 = runif(min=0.65, max=0.75, n=nsample),
                       p23 = runif(min=0.75, max=0.8, n=nsample),
                       p34 = p34,
                       p44 = p44)

# put servivability and fertility together
allp1 = cbind.data.frame(ps1,fs)
allp2 = cbind.data.frame(ps2,fs)

# get sobel samples
sens_rabbit=soboljansen(model = NULL, allp1, allp2, nboot = 100)

head(sens_rabbit$X)
```

```
##         p12       p23  p34 p44 F1 F2 F3 F4
## 1 0.7410316 0.7992523 0.65 0.1  0  2  6  1
## 2 0.6843269 0.7649563 0.65 0.1  0  2  6  1
## 3 0.7138217 0.7845572 0.65 0.1  0  2  6  1
## 4 0.7173975 0.7707550 0.65 0.1  0  2  6  1
## 5 0.6737917 0.7806086 0.65 0.1  0  2  6  1
## 6 0.7019273 0.7667977 0.65 0.1  0  2  6  1
```

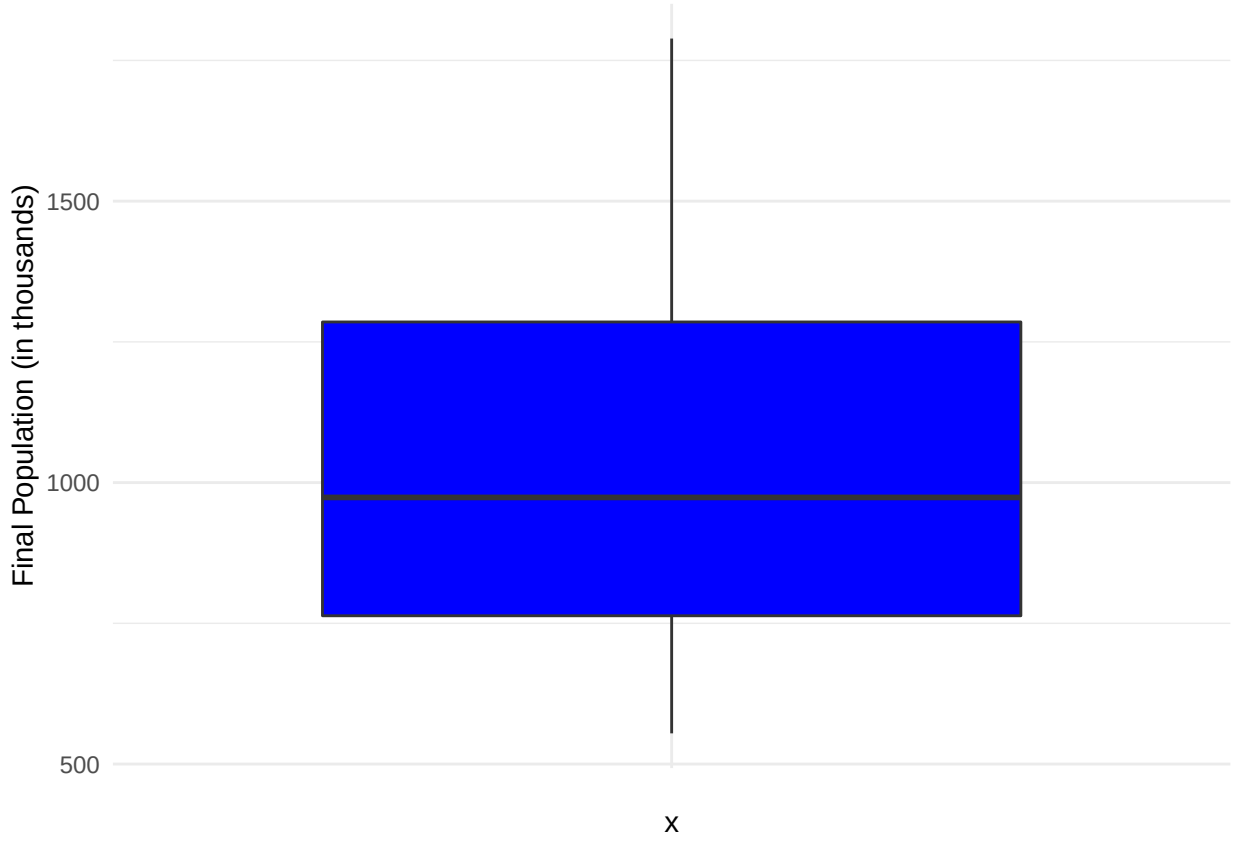
```
nsim=nrow(sens_rabbit$X)
```

Our second step is to create our wrapper function that contains our evol population function, and the parameter set selected by Sobel.

```
# run model and save what we care about: final population after 2 decades
# this is already output by evolve_pop so we don't need a compute_metric function

ini = c(0, 0, 10,0) # 10 adult rabbits
nyears = 20 # number of years


# parameter set, with code to extract our metric of interest (final population)
p_wrapper = function(p12, p23, p34, p44, F1, F2, F3, F4, use_func, initialpop, nstep ) {

fertility=c(F1,F2, F3, F4) #fertility data
survivability= c(p12, p23, p34, p44) #survivability data

res = use_func(survivability =survivability, fertility = fertility, initialpop=initialpop, nstep=nstep)
# now return the final population total
```

```
return(finalpop=res$poptot[nstep])
}

# use pmap here so we can specify rows of our sensitivity analysis parameter object
res = as.data.frame(sens_rabbit$X) %>% pmap_dbl(p_wrapper,
                                                initialpop=ini,
                                                nstep=nyears,
                                                use_func=evolve_pop)
```

```
# transform our result into a data frame

ggplot(data.frame(finalpop=res), aes(x="", y=finalpop/1000) )+
  geom_boxplot(fill="blue")+
  theme(axis.title.x = element_blank())+
  labs(y="Final Population (in thousands)") +
  theme_minimal()
```




```
# give our results to sensitivity structure

sens_rabbit=tell(sens_rabbit, res)

# loot at results
sens_rabbit$S
```

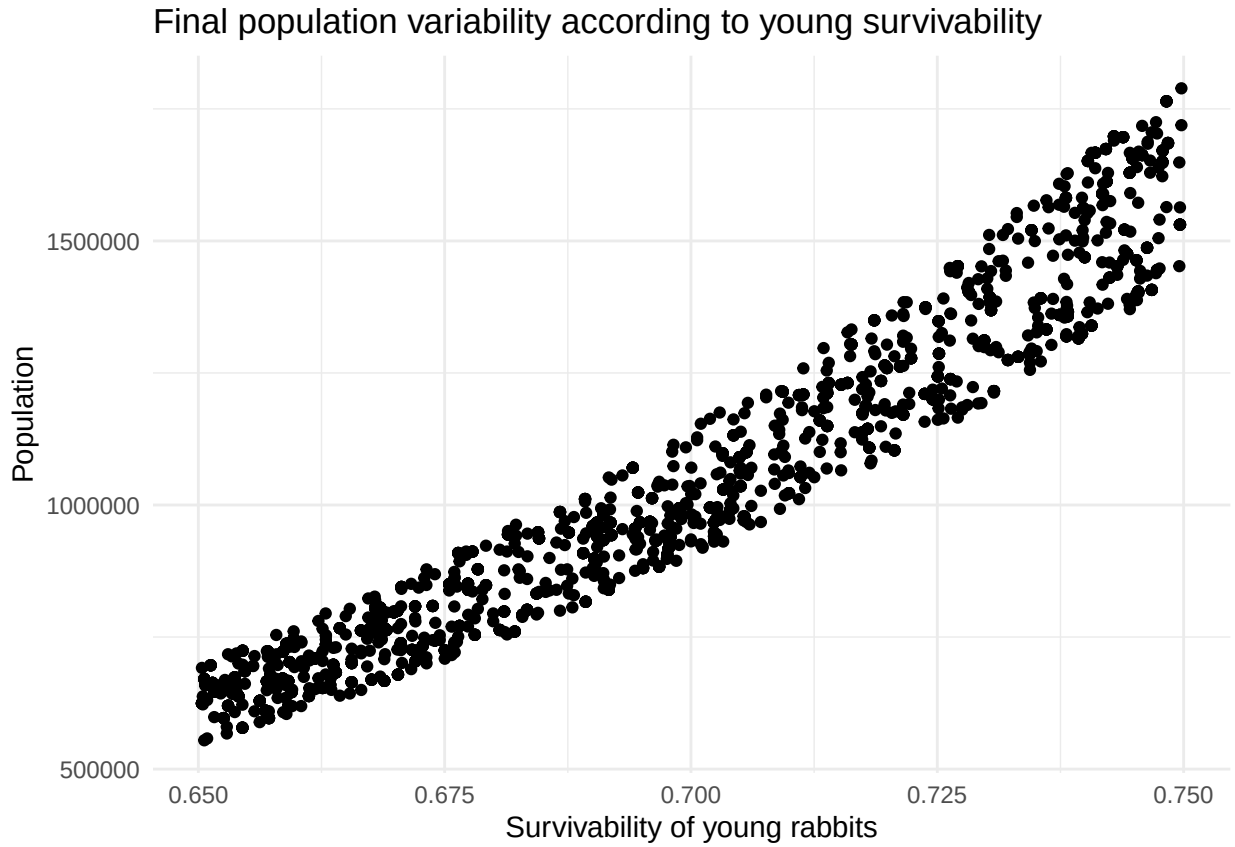
```
##      original          bias  std. error   min. c.i. max. c.i.
## p12 0.9427316  0.0002056641 0.007390261  0.92787253 0.9563968
## p23 0.1684852 -0.0250498869 0.072471996  0.01545462 0.3166113
```

```
## p34 0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
## p44 0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
## F1  0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
## F2  0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
## F3  0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
## F4  0.1260998 -0.0251829876 0.076221601 -0.02804478 0.2849519
```

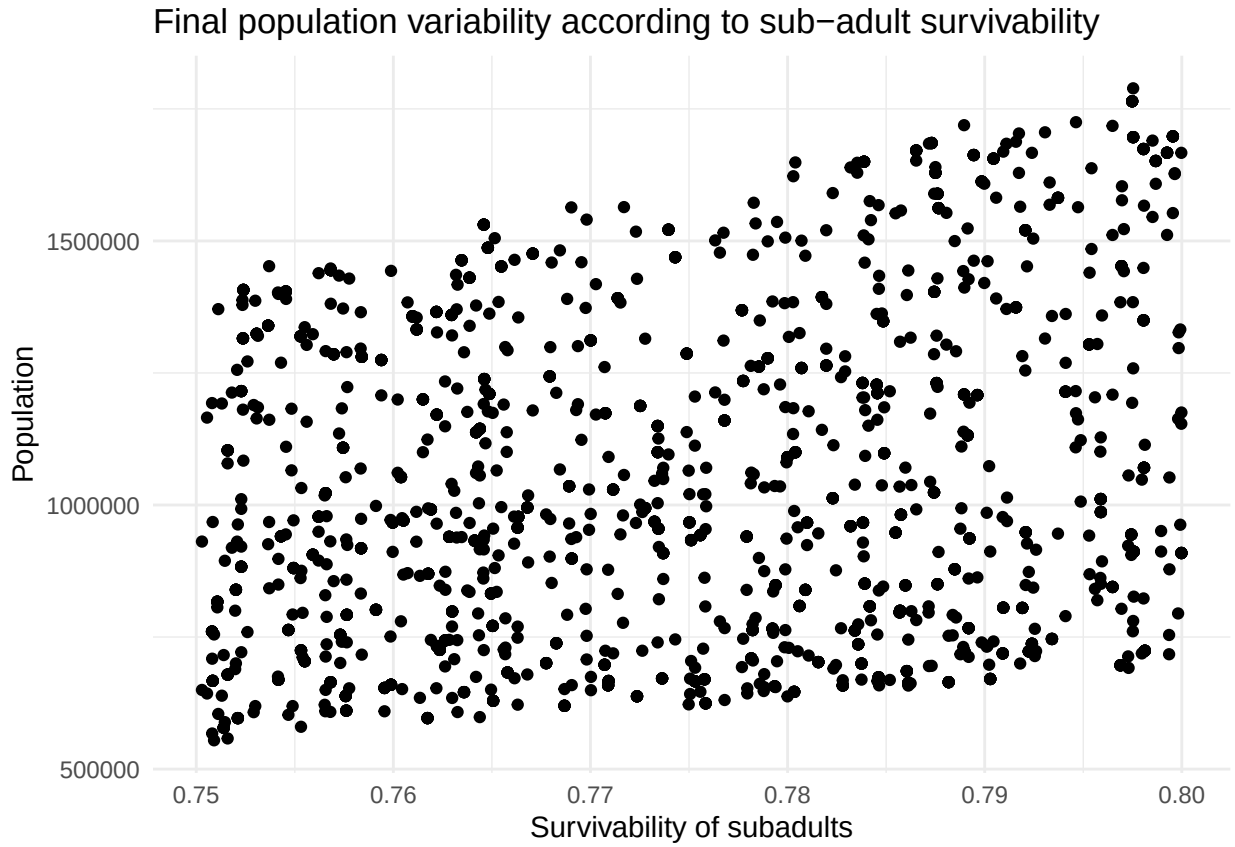
```
sens_rabbit$T
```

```
##       original         bias std. error  min. c.i.  max. c.i.
## p12 0.86324631 0.0257046278 0.07547294 0.69623503 1.01335211
## p23 0.05388797 0.0003025051 0.00738614 0.03827026 0.06876591
## p34 0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
## p44 0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
## F1  0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
## F2  0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
## F3  0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
## F4  0.00000000 0.0000000000 0.00000000 0.00000000 0.00000000
```

```
# graph the most sensitive parameter
tmp = cbind.data.frame(sens_rabbit$X, pop12=sens_rabbit$y)

ggplot(tmp, aes(p12, pop12))+
  geom_point()+
  labs(title="Final population variability according to young survivability",
       x="Survivability of young rabbits",
       y="Population") +
  theme_minimal()
```




```
ggplot(tmp, aes(p23, pop12))+
  geom_point()+
  labs(title="Final population variability according to sub-adult survivability",
       x="Survivability of subadults",
       y="Population") +
  theme_minimal()
```



Appendix: population evolution function

```
#' Population Evolution using Leslie Matrix
#' Evolve a population
#' @param fertility fertility rates
#' @param survivability survivability rates
#' @param initialpop initial population
#' @param nstep number of time steps
#' @return population structure for each time step (OR error message if population cannot be defined)


evolve_pop = function(fertility, survivability, initialpop, nstep) {


nclasses = length(fertility)

# make sure inputs are in the right format
if ((nclasses!=length(survivability) ))
{ return(sprintf("fertility %d doesn't match survivability %d",
                 nclasses, length(survivability))) }

if ((nclasses!=length(initialpop) ))
{ return(sprintf("population initialization %d  doesn't match fertility %d ", length(initialpop),
         length(fertility))) }
```

```
#initialize the Leslie matrix
leslie_matrix = matrix(nrow=nclasses, ncol=nclasses)
leslie_matrix[,] = 0.0
leslie_matrix[1,] = fertility

for (i in 1:(nclasses-1)) {
leslie_matrix[i+1,i] = survivability[i]
}
leslie_matrix[nclasses,nclasses] = survivability[nclasses]

# create an matrix to store population structure
pop_structure = matrix(nrow=nclasses, ncol=nstep)
total_pop = rep(0, times=nstep)
pop_structure[,1] = initialpop


for (i in 2:nstep) {

total_pop[i]=sum(pop_structure[,i-1])
pop_structure[,i] = leslie_matrix %*% pop_structure[,i-1]

}

return(list(popbyage=pop_structure, poptot=total_pop))
}
```