Exponential population growth

Matthew Malishev[1]*
[1] *Department of Biology, Emory University, 1510 Clifton Road NE, Atlanta, GA, USA, 30322*

Contents

Date: 2018-10-30
R version: 3.5.0
*Corresponding author: matthew.malishev@gmail.com
This document can be found at https://github.com/darwinanddavis

R session info

```
params$session
```

```
R version 3.5.0 (2018-04-23)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: OS X El Capitan 10.11.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_3.5.0  backports_1.1.2 magrittr_1.5    rprojroot_1.3-2 tools_3.5.0     htmltools_0.3.6
 [7] pillar_1.2.3    tibble_1.4.2    yaml_2.2.0      Rcpp_0.12.18    stringi_1.2.3   rmarkdown_1.10
[13] knitr_1.20      stringr_1.3.1   digest_0.6.15   rlang_0.2.1     evaluate_0.10.1
```

Overview

Examples of exponential population growth in R.

Install dependencies

```r
packages <- c("dplyr","deSolve","pdftools")
if (require(packages)) {
    install.packages(packages,dependencies = T)
    require(packages)
}
lapply(packages,library,character.only=T)
```

Section 1

Exponential growth equation

$$N_t = N_0 \cdot e^{rt}$$

Nt = the number of individuals in the population after t units of time
No = the initial population size (t = 0)
r = the exponential growth rate
t = time unit (usually in years)
e = the base of the natural logarithms (2.72)

Exponential rate of growth is commonly named the parameter lambda λ

$$\lambda = e^r$$

e^r = lambda. Exponential growth rate parameter.
The natural log (ln) of e = 1

$$ln(e) = 1$$

because $e^1 = e$.

The natural log of 1 = 0

$$ln(1) = 0$$

because $e^0 = 1$.

Parameters

```
# parameters
N_t <- 0 # expected pop size
N_0 <- 500 # initial pop size
e <- exp
r <- 0.012 # exponetial rate of growth
lambda <- e(1^r)
t <- 10 # time (in years)
```

```
# putting the above all together in R
N_t <- N_0 * e(r*t)
N_t
```

Example

A moose population has a growth rate of 0.02. In 2000, the population was 500. What will the population be in 2020?

```
# input your R code here
```

Instantaneous rate of growth

Equation showing the rate of population increase

$$\frac{dN}{dt} = rN$$

dN = change in number
dt = change in time
r = the per head maximum potential growth rate
N = number of individuals in a population

```
# in R
N <- 1000
dNdt <- r*N
dNdt
```

Example

A population of 100 individuals. Each individual can on average contribute 1/4 of an individual (new individual) to the population in a given unit of time. Find the rate of population increase.

```
# your r code
```

Simulating population growth

Set your parameters for the population

```
N_0 <- 20; # initial population size
```

Over time

```
N_1 <-  N_0 * r; # population size at t = 1
```

What does this look like at each time point?

```r
N_2 <- # ??
N_3 <- # ??
# etc
```

Population size

```r
popsize = c(N_0, N_1, N_2, N_3, N_4, N_5)
popsize
popsize[2]
```

Section 2

Simulation loop

```r
r <- 2
N_0 <-  20
t <- 10
popsize <- list() # create empty list to populate in loop
tt <- 2:t # now make t a vector to make a time period

for(i in tt){ # start  loop
  N_0[i] <- N_0[i-1]*r # middle of loop
  popsize <- N_0 # put the pop size into the list we created
} # end loop
popsize
```

Plot

```r
# plot it
plot(popsize)

# extend plot params
tta <- c(1,tt)# add the first year onto the time vector
ttm <- max(tt) # get the maximum value of the time vector
xlim <- c(1,ttm) # put this in a vector
ylim <- c(0,max(N_0)) # do the same for growth
par(las=1,bty="n") # set plotting params

plot(tta,popsize, # vectors to plot
     xlim=xlim, # set the x limits
     ylim=ylim, # y limits
     xlab="Time",
     ylab="Population growth",
     pch=20, # set point type
     col="pink"
     )

?plot # help page for plot function
```

Event-based conditions

Changing conditions halfway through a simulation. Using `if` and `else` statements.

```r
popsize <- list() # reset the list

for (i in tt){
  if((i%%5) == 0){ #modify growth rate depending on the year
    r <- 4
    }else{ # or just an ok year
      r <- 2
      }
  N_0[i] <- N_0[i-1]*r # the population growth equation
  popsize <- N_0
}

popsize

par(las=1,bty="n") # set plotting params
plot(popsize,pch=20,col="pink")
```

Another `for` loop example

```r
popsize <- list() # reset the list

for (i in tt){
  if (i <= 6){ # for the first five years, use r = 2
    r <- 2
    }else if(i >= 10){
      # and also at 10 years
      r <- 2
      }else{
    # but between 5th and 9th year,
    # we grow slower
        r <- 0.5
        } # end of if elseif else statement
  N_0[i] <- N_0[i-1]*r # the population growth equation
  popsize <- N_0
}

popsize

par(las=1,bty="n") # set plotting params
plot(popsize,pch=20,col="pink")
```

How do we figure out at what time point a population has reached a certain growth stage? Using `while` loops.

```r
r <- 2 # ?? # set r
N_0 <-  20 # initial size
popsize <- list() # reset the list
popsize_d <- 5000 # desired pop size

t <- 0 # set time var
while (N_0[t + 1] <= popsize_d){ # while we're under the desired pop size
   t = t + 1 ; # advance one time step
  N_0[t + 1] <- N_0[t]*r; # population growth eq
  time_to_popsize_d <- NULL # time to reach popsize_d?
  popsize <- N_0
} # close loop
```

How can we use R to get the time to reach popsize_d?

```
# get the length of the populaiton size vector and replace the "NULL" in the above loop with this funct
time_to_popsize_d <- NULL # ??

popsize[time_to_popsize_d] # this is the size of the pop at the desired time step
```

Making reusable R code: functions.

Putting all the above into reusable code that replaces the population growth parameters with input variables. Using function.

Functions pass user-defined input variables into an R procedure so that you can package code into a reusable and general program, the function.

Take your simulation you want to put into the function.

```
r <- 2 # set r
popsize <-  20 # initial size
popsize_d <- 5000 # desired size
time_to_popsize_d <- NULL # ?? # use your answer from above

t <- 0 # set time var
while (popsize[t + 1] <= popsize_d){ # while we're under the desired pop size
   t = t + 1 ; # advance one time step
  popsize[t + 1] <- popsize[t]*r; # population growth eq
  time_to_popsize_d <- NULL # time to reach popsize_d??
} # close loop

# writing the function. we'll call it 'discrete_growth'.
discrete_growth <- function(r){ # make popsize an input variable
  d <- popsize*r
  return(d) # we're just returning the popsize multipled by growth rate
}

# now we replace the 'r' parameter with a value for 'r'
discrete_growth(5)
```

What would the function look like if we wanted to figure out the following parameters?

```
r <- NULL #?? pop growth
N_0 <- NULL #?? initial pop size
t <- NULL
popsize <- list() # create empty list to populate in loop

# write the below function
discrete_growth <- function(r, ?? , ?? ){ # <--- there will be three input variables
  popsize <- ??
  tt <- 2:?? # make t a vector to make a time period
  for (i in tt){
    N_0[i] <- N_0[i-1]*r # middle of loop
    popsize <- N_0 # put the pop size into the list we created
    } # end loop
```

```r
  d <- popsize
  plot(d) # plot popsize with the appropriate time scale
  return(d) # get d
}

# now just replace the input variable with the values you want to simulate
discrete_growth()
```

Save the function in the `R` memory. Call it 'popsize_func'.

```r
popsize_func <- ??
```

Now save a separate function with different input parameters, then run both functons. This should give you an idea of how `R` saves things to memory.

```r
# use these values for popsize_func_2
0.1
500
10000

popsize_func_2 <- ?? #
popsize_func # is the output for popsize_func look different to popsize_func_2?
```