

Introduction to R

Lecture 4

EEB C119/C219 (Winter 2012)

Christopher C. Strelhoff

Overview

- Packages
 - What are they?
 - Installing packages, **deSolve** in particular
 - What packages are loaded? Where are they located? What packages are installed on your computer?
 - Using packages in scripts
- Ordinary differential equations (ODEs)
 - Define function for use with deSolve
 - How to (numerically) integrate ODE
- Examples
 - Density-dependent growth
 - Lotka-Volterra predator-prey

Packages

What are they?

- Packages
 - Allow expansion of R's available tools – set of functions coded by someone else and 'packaged' for you to use
 - Developed by many different people (universities, industry, etc.)
 - We will use **deSolve**, a nice ODE tool
- Other packages
 - Browse: <http://cran.r-project.org/>
 - Link to 'Packages' on left

Installing packages

- Can install package using interface
 - **rstudio**
 - Win, Mac – interface with R install
 - ** Might need to run program as **administrator** **
- R command line
 - Use command:

```
> install.packages('packagename', dep = TRUE)
```
 - ** Might need to start R as **administrator** **
- Repository
 - Choose repository (where packages are downloaded from) – one is at UCLA
- Only need to install package **once**

Package information

Helpful commands

- What packages are installed on **your computer**?
> *library()*
 - Note: these are not 'loaded' and ready to use
- What packages are 'loaded' in session?
> *search()*
- Where does R look for packages (on your computer)?
> *.libPaths()*

Using packages

Command line and scripts

- To use package at command line use **library** command, ex:

```
> library(deSolve) # load deSolve
```

- Again, to see what packages are loaded, type **search**:

```
> search()
```

```
[1] ".GlobalEnv"          "package:deSolve"    "package:stats"
[4] "package:graphics"    "package:grDevices"  "package:utils"
[7] "package:datasets"    "package:methods"    "Autoloads"
[10] "package:base"
```

- In script, use (usually at top of file):

```
> library(packageName)
```

or

```
> require(packageName)
```

deSolve

Numerical integration of ODEs

- deSolve uses well established algorithms for numerical integration of ODEs
- You don't have to worry about details (ask if interested)
- Using deSolve requires specific syntax
 - For a given ODE, have to define **function** for right-hand side of ODE
 - We'll do both 1d and 2d today
- Once you know the syntax, it's fairly simple to implement most ODEs

deSolve

Format for lsoda

- lsoda syntax:
 - > *lsoda(IC, times, ODEfunction, pars)*
 - IC: initial condition for state variables
 - times: vector of times where we want to know value of state variables
 - ODEfunction: function **we** define, describing ODE
 - pars: parameters for the ODE
- We'll do examples, so hang on.

deSolve

Format for function describing ODE

- ODE function syntax:

```
ODEfunction <- function(t, y, pars) {  
  
  derivs <- [insert model equations]  
  
  return(list(derivs))  
}
```

- `t`: is a variable used by R to keep track of the time
- `y`: is the state variable (or vector of state variables)
- `pars`: is a vector of model parameters
- `derivs`: describes right hand side (RHS) of ODE

Density-dependent growth

Remember the basics

- Equation is:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$

- State variable: N
- Parameters: r, K

Density-dependent growth

Define a function for the ODE

```
> desDepGrowthODE <- function(t, N, pars) {  
+   #  $dN/dt = r * N * (1 - N/K)$   
+   dNdt <- pars['r'] * N * (1 - N/pars['K'])  
+  
+   # return as list  
+   return(list(dNdt))  
+ }
```

- This defines RHS of ODE, still need a script to call `lsoda` and get output
- This is function, just like last lecture!

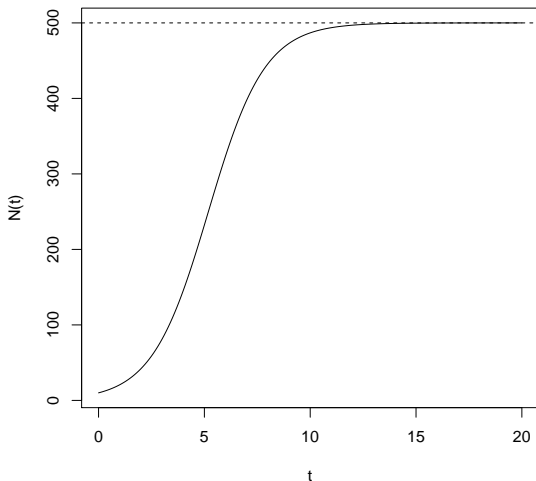
Density-dependent growth

Script to plot output from ODE – use **source**

```
> library(deSolve) # deSolve loaded
> # define ODE
> desDepGrowthODE <- function(t, N, pars) {
+   #  $dN/dt = r * N * (1 - N/K)$ 
+   dNdt <- pars['r'] * N * (1 - N/pars['K'])
+   # return as list
+   return(list(dNdt))
+ }
> # setup Init condition, times, parameters
> IC <- c(N=10)
> times <- seq(0, 20, by=0.1)
> pars <- c(r = 0.75, K=500)
> # call lsoda to get output
> output <- lsoda(IC, times, desDepGrowthODE, pars)
> # plot
> plot(output[,1],output[,2],type='l',xlab='t',ylab='N(t)')
> abline(h=pars['K'],lty=2) # dashed line at  $y=K$ 
```

Density-dependent growth

Plot resulting output from script



Density-dependent growth

What does the output look like?

```
> # setup Init condition, times, parameters
> IC <- c(N=10)
> times <- seq(0, 20, by=0.1)
> pars <- c(r = 0.75, K=500)
> # call lsoda to get output
> output <- lsoda(IC, times, desDepGrowthODE, pars)
> # get start of output
> head(output)
```

	time	N
[1,]	0.0	10.00000
[2,]	0.1	10.76208
[3,]	0.2	11.58086
[4,]	0.3	12.46035
[5,]	0.4	13.40480
[6,]	0.5	14.41871

- Using `desDepGrowthODE`, as defined on previous slide.

Demo: Lecture04_Ex01.R

Lotka-Volterra predator-prey

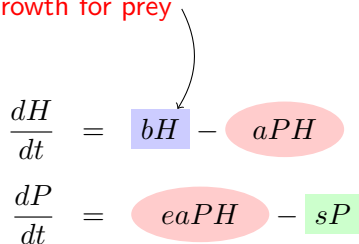
PER Section 6.1,pg 162

- Example of predator-prey (system of ODEs)
- Demonstrate deSolve implementation (not math)
- Details:
 - State variables:
 - H : prey (herbivore?)
 - P : predator
 - Parameters:
 - b : per-capita growth rate of prey
 - a : rate of encounter/eating
 - e : efficiency of prey to predator conversion
 - s : per-capita death rate of predator

Lotka-Volterra predator-prey

Equations and meaning (PER Section 6.1,pg 162)

- Exponential growth for prey

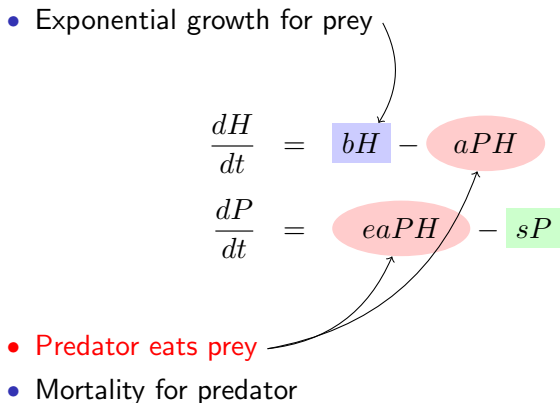
$$\begin{aligned}\frac{dH}{dt} &= bH - aPH \\ \frac{dP}{dt} &= eaPH - sP\end{aligned}$$


- Predator eats prey
- Mortality for predator

Lotka-Volterra predator-prey

Equations and meaning (PER Section 6.1,pg 162)

- Exponential growth for prey

$$\begin{aligned}\frac{dH}{dt} &= bH - aPH \\ \frac{dP}{dt} &= eaPH - sP\end{aligned}$$


- Predator eats prey
- Mortality for predator

Lotka-Volterra predator-prey

Equations and meaning (PER Section 6.1,pg 162)

- Exponential growth for prey

$$\begin{aligned}\frac{dH}{dt} &= bH - aPH \\ \frac{dP}{dt} &= eaPH - sP\end{aligned}$$

The diagram illustrates the Lotka-Volterra predator-prey model with two differential equations. The first equation, $\frac{dH}{dt} = bH - aPH$, describes the change in prey population (H). The term bH (in a blue box) represents exponential growth of prey, and the term aPH (in a red oval) represents the loss of prey due to predation. The second equation, $\frac{dP}{dt} = eaPH - sP$, describes the change in predator population (P). The term $eaPH$ (in a red oval) represents the gain in predators due to eating prey, and the term sP (in a green box) represents the mortality of predators. Arrows connect the descriptive text to the corresponding terms in the equations.

- Predator eats prey
- Mortality for predator

Lotka-Volterra predator-prey

Define function for ODE

```
> predPreyLVODE <- function(t, N, pars) {  
+   # get state variables from N  
+   H <- N[1]  
+   P <- N[2]  
+  
+   #  $dH/dt = b*H - a*P*H$   
+   dHdt <- pars['b'] * H - pars['a']*P*H  
+   #  $dP/dt = e*a*P*H - s*P$   
+   dPdt <- pars['e']*pars['a']*P*H - pars['s']*P  
+  
+   # return as list  
+   return(list(c(dHdt,dPdt)))  
+ }
```

- Note: return of function should have same ordering as N vector – H , then P in this case

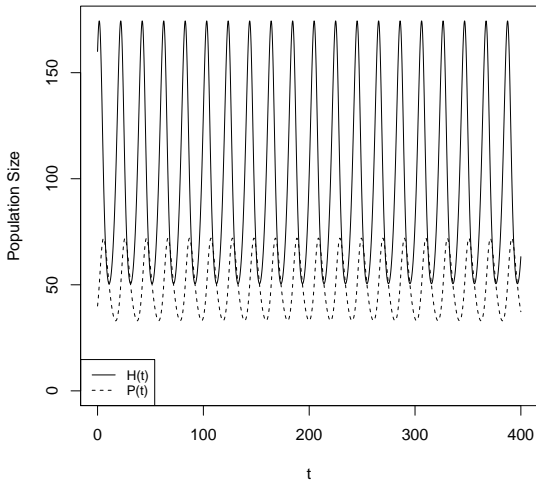
Lotka-Volterra predator-prey

Script to plot output

```
> library(deSolve) # load deSolve
> predPreyLVODE <- function(t, N, pars) {
+   # get state variables from N
+   H <- N[1]; P <- N[2]
+   #  $dH/dt = b \cdot H - a \cdot P \cdot H$ 
+   dHdt <- pars['b'] * H - pars['a'] * P * H
+   #  $dP/dt = e \cdot a \cdot P \cdot H - s \cdot P$ 
+   dPdt <- pars['e'] * pars['a'] * P * H - pars['s'] * P
+   # return as list
+   return(list(c(dHdt, dPdt)))
+ }
> # IC, time, parameters
> IC <- c(H=160, P=40); pars <- c(b=0.5, a=0.01, e=0.2, s=0.2)
> times <- seq(0, 400, by=0.1)
> # run lsoda
> output <- lsoda(IC, times, predPreyLVODE, pars)
> # plot
> plot(output[,1], output[,2], type='l', lty=1, ylim=c(0, max(output[,2])),
+       xlab='t', ylab='Population Size')
> lines(output[,1], output[,3], lty=2)
> # make a legend
> legend("bottomleft", c("H(t)", "P(t)"), cex=0.8, lty=c(1, 2))
```

Lotka-Volterra predator-prey

Plot resulting from script



Demo: Lecture04_Ex02.R