Multi-Dimensional Dynamic Systems

Andre Bourbonnais

2023-11-19

Exercise 2, Dynamic systems

Multidimensional ablalalalala

```r
library(tidyverse)
library(deSolve)
library(ggpubr)
```

1. Lotka-Volterra predator-prey equations

Lotka-Volterra (LV) predator-prey equations are given by the following system of differential equations:

$$\frac{dn}{dt} = rn - anp$$
$$\frac{dp}{dt} = anp - \mu p$$

Where $n(t)$ is the prey population, $p(t)$ is the predator population, r is the intrisic prey growth rate, a is the predation rate (*attack rate*), and μ is the predator death rate (*mortality*).

a) Write the ***isoclines*** of the system.

"What will the ***predator*** *population be when the* ***prey*** *population growth equals zero?"* Meaning, there is a sweet spot in predator population where the prey population is stable. This is called the *prey isocline* and *vice versa*. This value is called the *isocline* because it is the line where the population is stable.

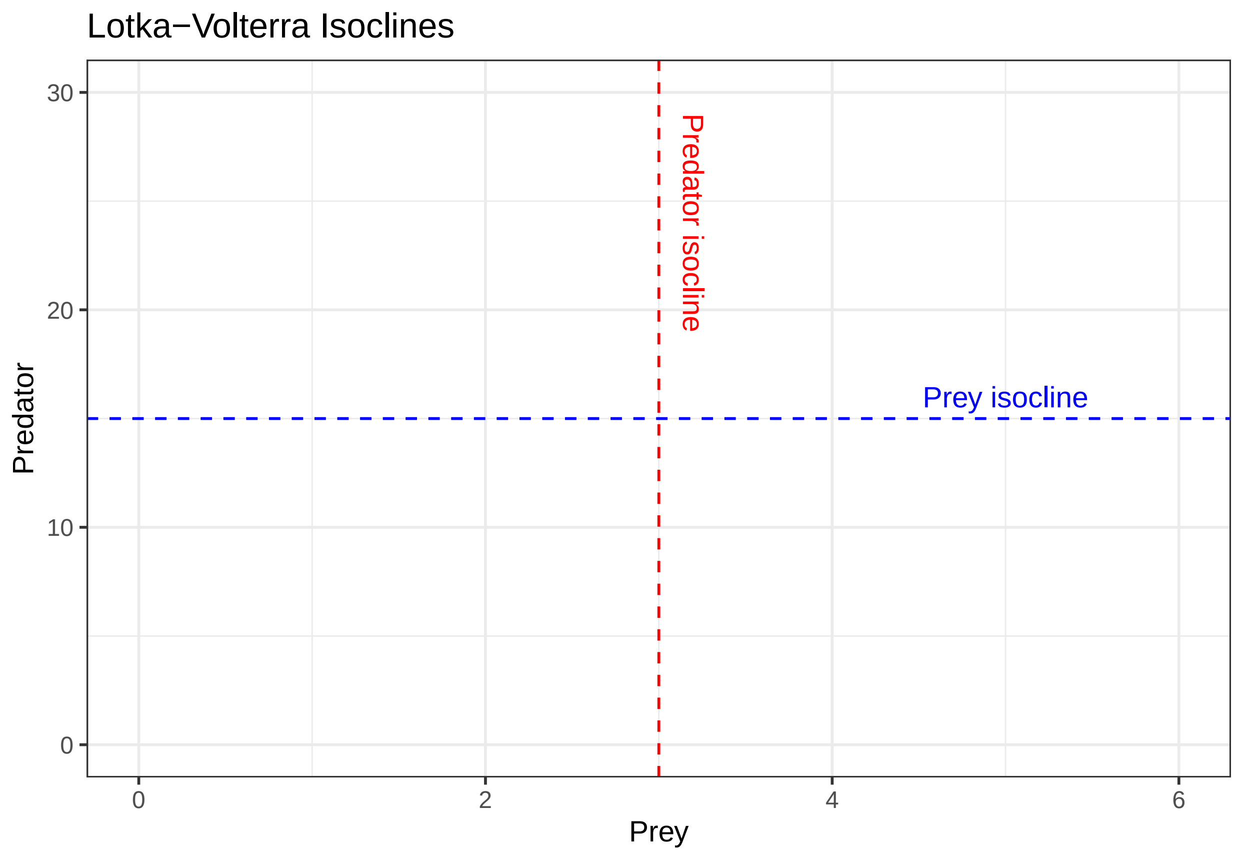
Thus, solve each *differential equation* at a time (equal to zero) and we will get the corresponding population value to keep the other value at 0 growth.

$$\frac{dn}{dt} = 0 \leftrightarrow rn - anp = 0 \leftrightarrow n(r - ap) = 0$$
$$\frac{dp}{dt} = 0 \leftrightarrow anp - \mu p = 0 \leftrightarrow p(an - \mu) = 0$$

We get two trivial equilibrium points, $n = 0, p = 0$ and two equilibrium points of interest:

$$\text{Prey: } p = \frac{r}{a}, \quad \text{Predator: } n = \frac{\mu}{a}$$

```r
# Define the function for LV isoclines
LV_iso <- function(r, a, mu){
  ne <- r/a
  pe <- mu/a
  return(list(prey_iso = ne, pred_iso = pe))
}

# Define the parameters
P <- list(r = 3,
          a = 0.2,
          mu = 0.6
          )
isoc <- LV_iso(P$r, P$a, P$mu)

# Plot the isoclines

# Create a blank ggplot object with the axes limits where i add the isoclines
# using horizontal and vertical lines.
ggplot() +
  geom_hline(yintercept = isoc$prey_iso, color = "blue", linetype = "dashed") +
  geom_vline(xintercept = isoc$pred_iso, color = "red", linetype = "dashed") +
  labs(x = "Prey", y = "Predator") +
  # Multiply by 2 as the isoclines are the equilibrium points.
  xlim(0, isoc$pred_iso*2) +
  ylim(0, isoc$prey_iso*2) +
  theme_bw() +
  annotate("text", x = 5, y = 16, label = "Prey isocline", color = "blue") +
  annotate("text", x = 3.2, y = 24, label = "Predator isocline", color = "red",
           angle = -90) +
  ggtitle("Lotka-Volterra Isoclines")
```



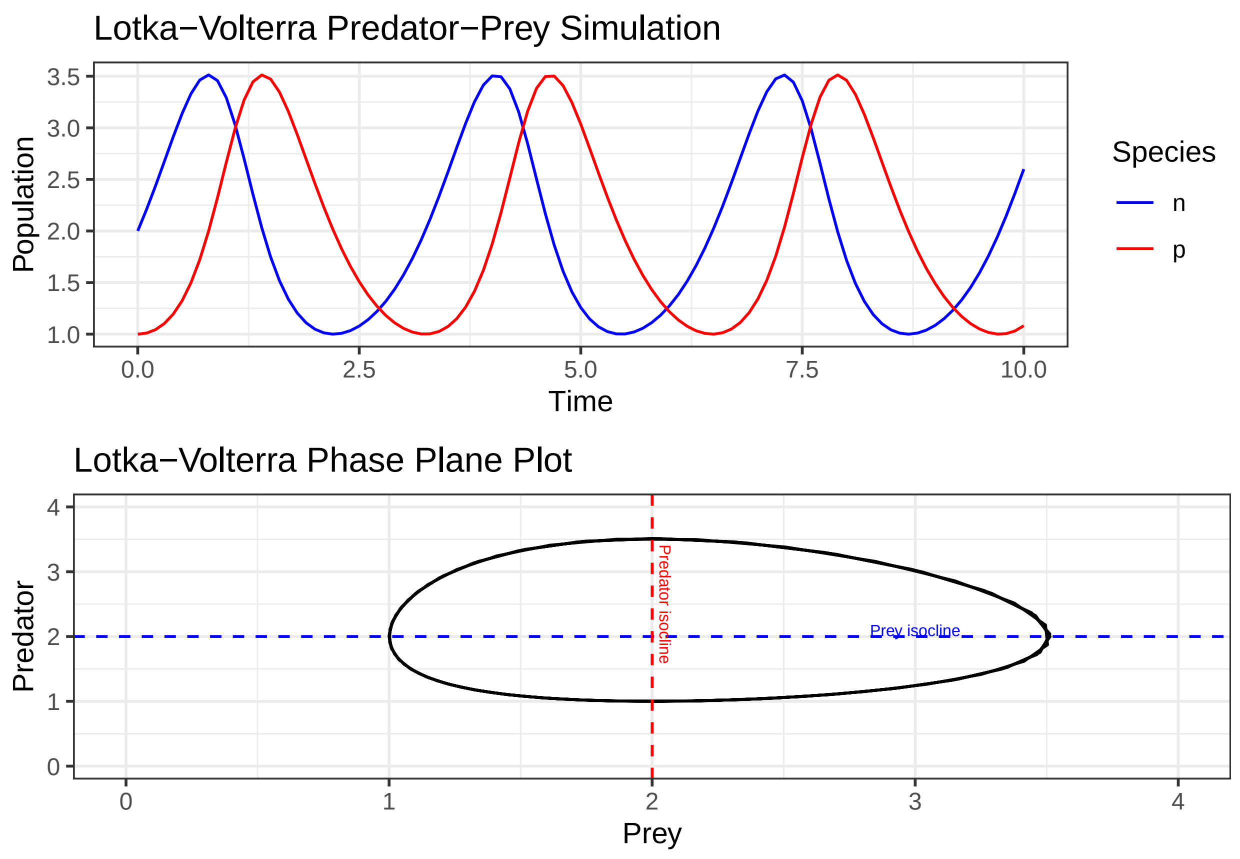
b) Simulate using `ode()` and plot a time-plot and phase-plane plot.

Run simulations with ode() from deSolve in R. Use two state variables: prey and predator densities. Return their time derivatives as a vector in a list. Simulate the system, outputting a matrix with time, prey density, and predator density. Plot both populations over time and create a phase-plane plot (predator vs. prey). Incorporate isoclines using LV_iso().

1. **Simulation Setup**: Utilize the `ode()` function from the `deSolve` package.
2. **State Variables**: Focus on two variables - prey and predator densities.
3. **Output Format**: System function should return a list with a vector of these time derivatives.
4. **Run Simulation**: Generate a matrix output with columns for time, prey density, and predator density.
5. **Plotting**: - Create a time plot showing both populations. - Construct a phase-plane plot with predator vs. prey densities.
6. **Isoclines**: Use `LV_iso()` function to add isoclines to the phase-plane plot.

```r
# Define the function for LV isoclines
LV_iso <- function(r, a, mu){
  ne <- r/a
  pe <- mu/a
  return(list(prey_iso = ne, pred_iso = pe))
}
```

```r
# LV system function
LV_sys <- function(t, np, P){
  # State variables
  n <- np[1]
  p <- np[2]

  # Differential equations
  dndt <- P$r*n - P$a*n*p
  dpdt <- P$a*n*p - P$mu*p

  return(list(c(dndt, dpdt)))
}

# Define the parameters
P2 <- list(
  r = 2,
  a = 1,
  mu = 2
  )

# Define the initial population size
np0 <- c(n = 2, p = 1)

# Time range
time_vec <- seq(0, 10, 0.1)

# Run the simulation
res_sim <- ode(y = np0, times = time_vec, func = LV_sys, parms = P2)

# Pivot the data to a long formated dataframe
df_sim <- res_sim %>%
  as.data.frame() %>%
  pivot_longer(cols = c("n", "p"), names_to = "Species",
               values_to = "Population")

# Calculate the isoclines
isoc <- LV_iso(P2$r, P2$a, P2$mu)

# Plotting the results
timeplot <- df_sim %>%
  ggplot(aes(x = time, y = Population, color = Species)) +
  geom_line() +
  scale_color_manual(values = c("blue", "red")) +
  theme_bw() +
  labs(x = "Time", y = "Population") +
  ggtitle("Lotka-Volterra Predator-Prey Simulation")

# Phase plane plot
phase_plane <- df_sim %>%
  pivot_wider(names_from = Species, values_from = Population) %>%
  ggplot(aes(x = n, y = p)) +
  geom_path() +
  geom_hline(yintercept = isoc$prey_iso, color = "blue", linetype = "dashed") +
  geom_vline(xintercept = isoc$pred_iso, color = "red", linetype = "dashed") +
  labs(x = "Prey", y = "Predator") +
  # Multiply by 2 as the isoclines are the equilibrium points.
  xlim(0, isoc$pred_iso*2) +
  ylim(0, isoc$prey_iso*2) +
  theme_bw() +
  annotate("text", x = isoc$pred_iso+1, y = isoc$prey_iso+0.1,
           label = "Prey isocline", color = "blue", size = 2) +
  annotate("text", x = isoc$pred_iso+0.05, y = isoc$prey_iso+0.5,
           label = "Predator isocline", color = "red", angle = -90, size = 2) +
  ggtitle("Lotka-Volterra Phase Plane Plot")

ggarrange(timeplot, phase_plane, ncol = 1, nrow = 2)
```



c) Calculate the Jacobian matrix

To determine the stability of an equilibrium, one should calculate the *Jacobian* matrix of the system at the equilibrium point. If all *eigenvalues* of the Jacobian matrix have a **negative real part**, the equilibrium is stable.

Calculate the Jacobian matrix for the system at the equilibrium point. Write a function that takes three parameters (r, a, μ) as input and returns the Jacobian matrix as output. We have extended the LV equation with c as the conversion efficiency of prey to predator biomass:

$$\frac{dn}{dt} = f_n(n,p) = rn - anp$$
$$\frac{dp}{dt} = f_p(n,p) = canp - \mu p$$

i) Calculating the Jacobian matrix: The partial derivative of f_n with respect to n is written $\frac{\partial f_n}{\partial n}$. It can be interpreted as the slope of the surface parallel to the *n-axis*. It is calculated by *'pretending'* that all other variables (p) are constant, and vice versa for $\frac{\partial f_n}{\partial p}$.

Thus the partial derivatives of f_n and f_p are:

$$\frac{\partial f_n}{\partial n} = r - ap, \quad \frac{\partial f_n}{\partial p} = -an$$
$$\frac{\partial f_p}{\partial n} = cap, \quad \frac{\partial f_p}{\partial p} = can - \mu$$

Giving us the *Jacobian matrix*:

$$J = \begin{bmatrix} r - ap & -an \\ cap & can - \mu \end{bmatrix}$$

Now, with the added conversion efficiency c, the equilibrium points are:

$$\text{Prey: } p = \frac{r}{a}, \quad \text{Predator: } n = \frac{\mu}{ca}$$

Lets test the function with these as well.

```r
# LV Jacobian matrix
LV_jac <- function(np, P){
  r <- P$r
  a <- P$a
  mu <- P$mu
  c <- P$c
  n <- np[1]
  p <- np[2]

  J <- matrix(c(r-a*p, -a*n, c*a*p, c*a*n-mu), nrow = 2, ncol = 2)

  return(J)
}

# Define the parameters
P3 <- list(
  r = 2,
  a = 1,
  mu = 2,
  c = 1
  )

# Define initial population
np <- c(n = 2, p = 1)
```

```
# The non-trivial equilibrium points
np_eq <- c(n = P3$mu/(P3$c*P3$a), p = P3$r/P3$a)

# The Jacobian matrix
LV_jac(np, P3)
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]   -2    0
```

```
# The Jacobian matrix at the equilibrium point
LV_jac(np_eq, P3)
```

```
##      [,1] [,2]
## [1,]    0    2
## [2,]   -2    0
```

d) Calculate the eigenvalues of the Jacobian matrix

Use the `LV_jac()` function to calculate the Jacobian for a set of parameter values. Also calculate the cycle period associated with the complex eigenvalues. Is it a good approximation of the length of the actual cycles?

```
# The Jacobian matrix
J <- LV_jac(np_eq, P3)

eigenval <- eigen(J)

# Compare the eigenvalues to the cycle period
2*pi/Im(eigenval$values[1])
```

```
## [1] 3.141593
```