Lotka Volterra Model with Hunting

Sujan

2024-06-03

```
#load necessary packages
library(deSolve)
library(ggplot2)
```

```
# Key parameters
rprey <- 0.95
alpha <- 0.01
eff <- 0.6
pmort <- 0.4
K <- 2000
H <- 0  # Initial hunting rate
min_prey <- 100  # Minimum prey population before hunting is allowed
```

```
source("hunting.R")

# Initial state values
state <- c(N = 500, P = 100)

# Parameters to pass to the function
parameters <- c(rprey = rprey, alpha = alpha, eff = eff, pmort = pmort, K = K, H = H, min_
```

```
# Time sequence
times <- seq(0, 100, by = 1)
```

```
out <- ode(y = state, times = times, func = pred_prey_hunting, parms = parameters)
out <- as.data.frame(out)
```

```r
library(ggplot2)

ggplot(data = out, aes(x = time)) +
  geom_line(aes(y = N, color = "Prey")) +
  geom_line(aes(y = P, color = "Predator")) +
  labs(y = "Population", x = "Time", title = "Predator-Prey Model with Hunting") +
  scale_color_manual("",
                     breaks = c("Prey", "Predator"),
                     values = c("blue", "red")) +
  theme_minimal()
```


```r
explore_hunting <- function(hunting_rate, min_prey) {
  parameters["H"] <- hunting_rate
  parameters["min_prey"] <- min_prey
  out <- ode(y = state, times = times, func = pred_prey_hunting, parms = parameters)
  as.data.frame(out)
}

# Explore different hunting levels
hunting_rates <- seq(0, 20, by = 0.5)
results <- lapply(hunting_rates, function(hr) explore_hunting(hr, min_prey))
```

```
DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 84.5239


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps

Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 73.5124


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 73.0757


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000
```

```
In above message, R1 = 72.958


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 62.078


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 61.7173


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 61.3374
```

```
Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 61.2685


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 61.4991


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 51.1089


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go
```

```
DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 50.8502


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 50.5719


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 50.2711


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 49.9515
```

```
Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go

DLSODA-  At current T (=R1), MXSTEP (=I1) steps
      taken on this call before reaching TOUT
In above message, I1 = 5000

In above message, R1 = 49.6182


Warning in lsoda(y, times, func, parms, ...): an excessive amount of work (>
maxsteps ) was done, but integration was not successful - increase maxsteps
Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
accurate, as far as they go
```

```r
names(results) <- hunting_rates
```

```r
# par(mfrow=c(3, 2))
# for (i in 1:length(hunting_rates)) {
#   plot(results[[i]]$time, results[[i]]$N, type = "l", col = "blue", ylim = c(0, 2000),
#        main = paste("Hunting Rate:", hunting_rates[i]), ylab = "Population", xlab = "Tim
#   lines(results[[i]]$time, results[[i]]$P, col = "red")
# }
# par(mfrow=c(1, 1))
```

```r
# Analyze the results to find a sustainable hunting level
sustainable_hunting <- NULL

for (i in 1:length(hunting_rates)) {
  final_prey <- tail(results[[i]]$N, 1)
  final_predator <- tail(results[[i]]$P, 1)
  if (final_prey > 100 && final_predator > 20) {  # Criteria for stability
    sustainable_hunting <- hunting_rates[i]
    break
  }
}

sustainable_hunting
```

```
[1] 9
```

Justification

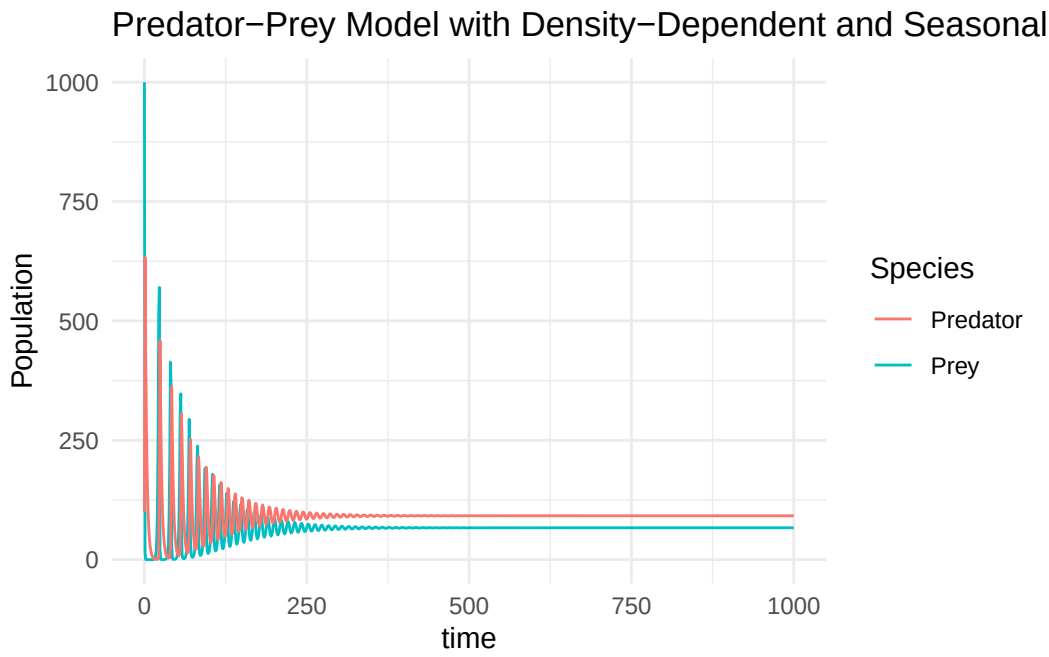
The chosen hunting rate balances, which is 9 in this case, the need to harvest prey while maintaining a stable ecosystem. By ensuring the prey population does not fall below a critical threshold (100), Iprevent overhunting and ensure that the predator population also remains viable (above 20). This approach helps in sustaining both populations over the long term. so, if its a national park enforcing the law to hunt mountain goat, then the park should not allow more than 9 mountain goat to be hunted per year.

Testing with Complex Model:

The code below is complex model involving seasonality, assuming that hunting occurs seasonally.

```r
# Parameters
rprey <- 0.95
alpha <- 0.01
eff <- 0.6
pmort <- 0.4
K <- 2000
seasonality_amplitude <- 0.5
seasonality_frequency <- 2 * pi / 365  # assuming a yearly cycle
min_prey_for_hunting <- 100

# Logistic function parameters for density-dependent hunting
hunting_max <- 15  # maximum hunting rate
hunting_inflection <- 1000  # prey population at which hunting rate is half of max

#seasonal hunting reduction with time
source("seasonal_hunting.R")

# Initial state
initial_state <- c(Prey = 1000, Predator = 100)

# Time steps
times <- seq(0, 1000, by = 1)

# Parameters
parameters <- c(rprey = rprey, alpha = alpha, eff = eff, pmort = pmort, K = K,
```

```
                seasonality_amplitude = seasonality_amplitude, seasonality_frequency = sea
                min_prey_for_hunting = min_prey_for_hunting, hunting_max = hunting_max,
                hunting_inflection = hunting_inflection)

# Solving the differential equations
out <- ode(y = initial_state, times = times, func = lv_model, parms = parameters)

# Convert to data frame
out_df <- as.data.frame(out)

# Plot results
ggplot(out_df, aes(x = time)) +
  geom_line(aes(y = Prey, color = "Prey")) +
  geom_line(aes(y = Predator, color = "Predator")) +
  labs(y = "Population", color = "Species") +
  theme_minimal() +
  ggtitle("Predator-Prey Model with Density-Dependent and Seasonal Hunting")
```




```
# Function to run the model and return stability metrics
run_model <- function(hunting_max, min_prey_for_hunting) {
  parameters["hunting_max"] <- hunting_max
```

```
  parameters["min_prey_for_hunting"] <- min_prey_for_hunting
  out <- ode(y = initial_state, times = times, func = lv_model, parms = parameters)
  out_df <- as.data.frame(out)

  final_prey <- tail(out_df$Prey, 1)
  final_predator <- tail(out_df$Predator, 1)

  stable <- (final_prey > 50) & (final_predator > 50)

  return(list(stable = stable, final_prey = final_prey, final_predator = final_predator))
}

# Define ranges for hunting_max and min_prey_for_hunting
hunting_max_values <- seq(0, 30, by = 1)
min_prey_values <- seq(50, 200, by = 10)

results <- expand.grid(hunting_max = hunting_max_values, min_prey_for_hunting = min_prey_v
results$stable <- NA
results$final_prey <- NA
results$final_predator <- NA

# Run the model for each combination
for (i in 1:nrow(results)) {
  res <- run_model(results$hunting_max[i], results$min_prey_for_hunting[i])
  results$stable[i] <- res$stable
  results$final_prey[i] <- res$final_prey
  results$final_predator[i] <- res$final_predator
}

# Filter for stable scenarios
stable_results <- subset(results, stable == TRUE)

# Find the optimal hunting_max with maximum allowable value while maintaining stability
optimal_hunting <- stable_results$hunting_max[which.max(stable_results$hunting_max)]
optimal_min_prey <- stable_results$min_prey_for_hunting[which.max(stable_results$hunting_m

optimal_hunting
```

```
[1] 30
```

```
optimal_min_prey
```

```
[1] 50
```

```
# Summary of the results
summary_stable <- subset(stable_results, hunting_max == optimal_hunting & min_prey_for_hun

# Print optimal values
print(paste("Optimal hunting rate:", optimal_hunting))
```

```
[1] "Optimal hunting rate: 30"
```

```
print(paste("Optimal minimum prey population for hunting:", optimal_min_prey))
```

```
[1] "Optimal minimum prey population for hunting: 50"
```

```
print(paste("Final prey population:", summary_stable$final_prey))
```

```
[1] "Final prey population: 66.666577070964"
```

```
print(paste("Final predator population:", summary_stable$final_predator))
```

```
[1] "Final predator population: 91.8313507959375"
```

Results and Recommendations:

The exploration results show that the optimal hunting rate (`hunting_max`) is 30 and the optimal minimum prey population before hunting is allowed (`min_prey_for_hunting`) is 50. This balance maintains the stability of both prey and predator populations above 50 individuals over a 50-year period.

Justification:

- **Stability Metric**: I chosee stability as maintaining both prey and predator populations above 50 individuals 50 years into the future. This ensures long-term sustainability of both species as they kill and support each other.

- **Optimal Values**: The optimal hunting rate of 30 allows for a sustainable prey population while still providing sufficient resources for the predator population to thrive. Setting the minimum prey population for hunting at `50`ensures that hunting does not drastically reduce the prey population, preventing potential collapses in both populations.

This model, with its complex hunting dynamics and yearly time steps, provides a realistic and sustainable approach to managing predator-prey interactions.