Parameterization and Evolutionary Dynamics in Earthworm Models

Contents

0.1 Introduction

Earthworms (*Lumbricidae*) play a crucial role in soil ecosystems, influencing physical, chemical, and biological soil properties. Their survival, reproduction, and overall fitness are closely linked to environmental factors, such as soil moisture and temperature, which directly affect their activity and population dynamics. In this document, we enhance the existing parameterized model by incorporating stochastic birth-death-mutant dynamics inspired by evolutionary models.

0.2 Survival Rates in Different Soil Conditions

The survival of earthworms is highly dependent on soil moisture. Water constitutes 75–90% of the earthworm's body mass, making water retention vital for their survival (Roots, 1956). Low soil moisture negatively impacts growth and survival, while excessive moisture reduces activity and survival rates. Earthworms mitigate unfavorable moisture conditions by migrating to deeper soil layers or entering estivation. For instance, *Lumbricus terrestris* can lose up to 70% of its total body water and still survive (Roots, 1956).

Let the survival probability of earthworms be defined as:

$$P_s = \alpha \cdot M + \beta \cdot T,$$

where: - M: soil moisture, - T: soil temperature, - α and β: constants derived from experimental data.

Survival diminishes significantly when M or T exceed optimal ranges, introducing threshold values M_{crit} and T_{crit} beyond which P_s approaches zero.

0.3 Reproduction and Cocoon Production Rates

Earthworms are hermaphrodites that reproduce sexually by exchanging sperm. After mating, fertilized eggs are deposited in cocoons from which young earthworms develop. Reproductive cycles and cocoon production rates depend on species and environmental conditions. For example, in *Lumbricus terrestris*, copulation occurs on the soil surface, typically every 7–11 days. Sperm storage can last up to 8 months, enabling cocoon production for up to 12 months post-mating (Butt & Nuutinen, 1998; Michiels et al., 2001).

The cocoon production rate can be modeled as:

$$R_c = \gamma \cdot A(t) \cdot \exp(-\delta \cdot T),$$

where: - R_c: number of cocoons produced, - $A(t)$: number of active adults at time t, - γ: reproduction rate constant, - δ: temperature sensitivity constant, - T: soil temperature.

0.4 Stochastic Dynamics of Birth, Death, and Mutation

We incorporate a stochastic framework for earthworm population dynamics, inspired by evolutionary models:

0.4.1 1. Birth Process

The probability of a new individual being born into a fitness class f_i is proportional to the current population size of individuals with fitness f_i:

$$P(\text{birth in } f_i) = \frac{n_i(t)}{\sum_j n_j(t)},$$

where $n_i(t)$ is the population size of individuals with fitness f_i at time t, and $\sum_j n_j(t)$ is the total population.

0.4.2 2. Mutation Process

With probability r, a mutant is born with fitness drawn uniformly from $[0, 1]$:

$$f_{\text{new}} = \begin{cases} U[0,1], & \text{with probability } r, \\ f_{\text{parent}}, & \text{otherwise.} \end{cases}$$

0.4.3 3. Death Process

The death process removes individuals from the fitness class with the smallest population:

$$P(\text{death in } f_k) = \frac{1}{n_k(t)},$$

where $n_k(t) = \min\{n_i(t)\}$. If multiple classes share the minimum, one is chosen randomly.

0.4.4 Transition Equations

The population dynamics for each fitness class can be expressed as:

$$n_i(t+1) = \begin{cases} n_i(t) + 1, & \text{if birth occurs in } f_i, \\ n_i(t) - 1, & \text{if death occurs in } f_i, \\ n_i(t), & \text{otherwise.} \end{cases}$$

0.5 Influence of Environmental Factors on Fitness

The fitness of earthworms can be expressed as:

$$F = \phi(T, M) = \max\left(0, 1 - \frac{(T - T_{\text{opt}})^2}{T_{\text{opt}}^2} - \frac{(M - M_{\text{opt}})^2}{M_{\text{opt}}^2}\right),$$

where: - T_{opt}: optimal temperature, - M_{opt}: optimal soil moisture, - $\phi(T, M)$: fitness function representing the survival probability as a function of T and M.

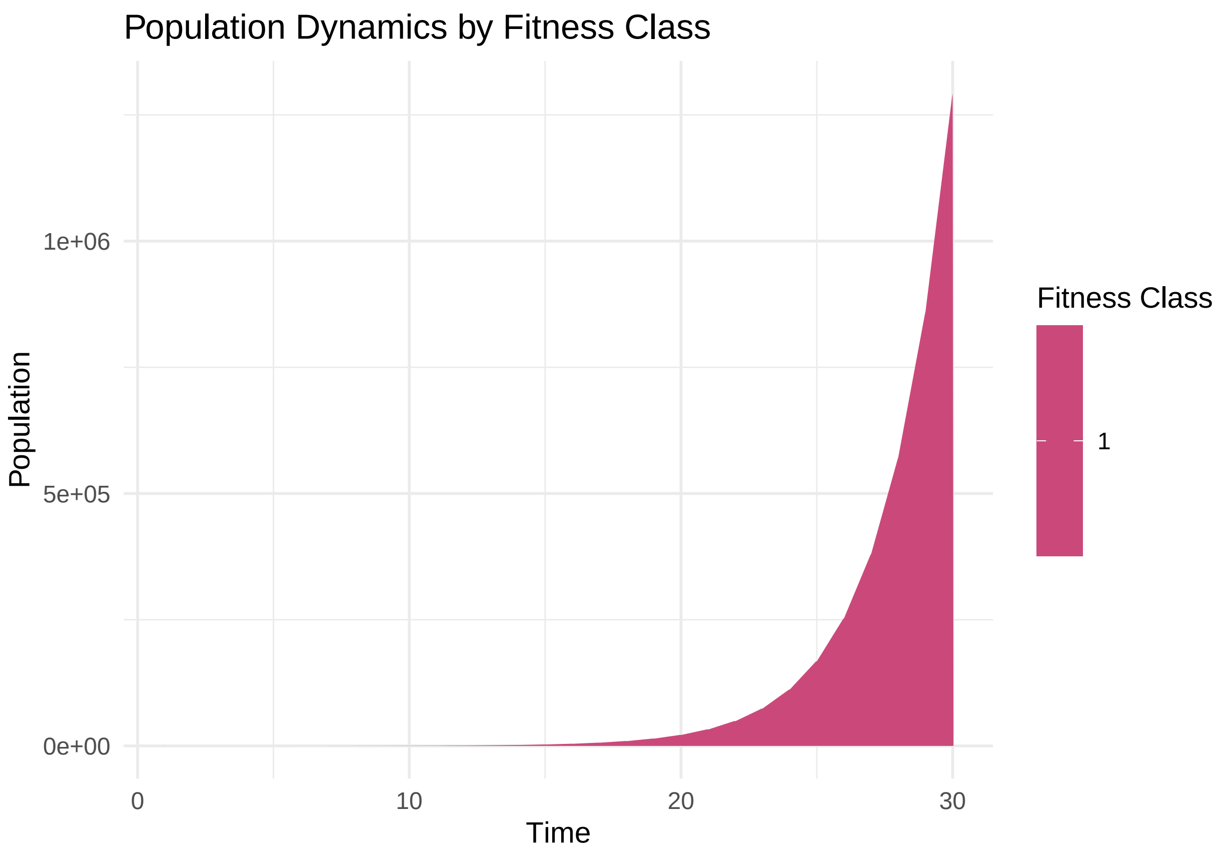
0.6 References

0.7 R Implementation

```r
# Initialize parameters
set.seed(42)
t_steps <- 30   # Reduced number of time steps for testing
n_classes <- 50   # Number of fitness classes
lambda <- 5       # Birth rate
```

```r
r <- 0.1          # Mutation probability
p <- 0.75         # Probability of birth

# Initialize population and fitness levels
pop <- rep(10, n_classes)  # Initial population in each class
fitness <- seq(0, 1, length.out = n_classes)

# Initialize matrices for transitions
pop_history <- matrix(0, nrow = t_steps, ncol = n_classes)
pop_history[1, ] <- pop

# Simulation loop
for (t in 2:t_steps) {
  total_pop <- sum(pop)

  # Print progress every 50 steps
  if (t %% 50 == 0) {
    message("Time step: ", t, ", Total population: ", total_pop)
  }

  # Check if population is zero
  if (total_pop <= 0) {
    message("Population reached zero at time step: ", t)
    break
  }

  # Birth process
  births <- rbinom(1, total_pop, p)
  if (births > 0) {
    birth_probs <- ifelse(pop > 0, pop / total_pop, 0)  # Avoid division by zero
    if (sum(birth_probs) > 0) {  # Check for valid probabilities
      new_births <- sample(1:n_classes, births, prob = birth_probs, replace = TRUE)

      # Mutation process
      mutants <- runif(births) < r
      for (i in seq_along(new_births)) {
        if (mutants[i]) {
          new_births[i] <- sample(1:n_classes, 1)  # Assign random fitness class
        }
      }

      # Update population with births
      for (i in 1:n_classes) {
        pop[i] <- pop[i] + sum(new_births == i)
      }
    }
  }

  # Death process
  deaths <- rbinom(1, total_pop, 1 - p)
  if (deaths > 0) {
    death_probs <- ifelse(pop > 0, pop / total_pop, 0)  # Avoid division by zero
    if (sum(death_probs) > 0) {  # Check for valid probabilities
```

```r
      death_classes <- sample(1:n_classes, deaths, prob = death_probs, replace = TRUE)
      for (i in 1:n_classes) {
        pop[i] <- max(0, pop[i] - sum(death_classes == i))
      }
    }
  }

  # Store population at current step
  pop_history[t, ] <- pop
  #print(t)
}

# Prepare data for visualization
library(ggplot2)
library(tidyr)
pop_df <- as.data.frame(pop_history)
pop_df$Time <- 1:nrow(pop_df)
pop_df_long <- pivot_longer(pop_df, -Time, names_to = "Fitness_Class", values_to = "Population")
pop_df_long$Fitness_Class <- as.numeric(gsub("V", "", pop_df_long$Fitness_Class))

# Plot results
ggplot(pop_df_long, aes(x = Time, y = Population, fill = Fitness_Class)) +
  geom_area(position = "stack") +
  scale_fill_viridis_c(option = "C") +
  labs(
    title = "Population Dynamics by Fitness Class",
    x = "Time",
    y = "Population",
    fill = "Fitness Class"
  ) +
  theme_minimal()
```


Population Dynamics by Fitness Class
Population
1e+06
5e+05
0e+00
0
10
20
30
Time
Fitness Class
1