Projeto 4 - Predador de Ápice

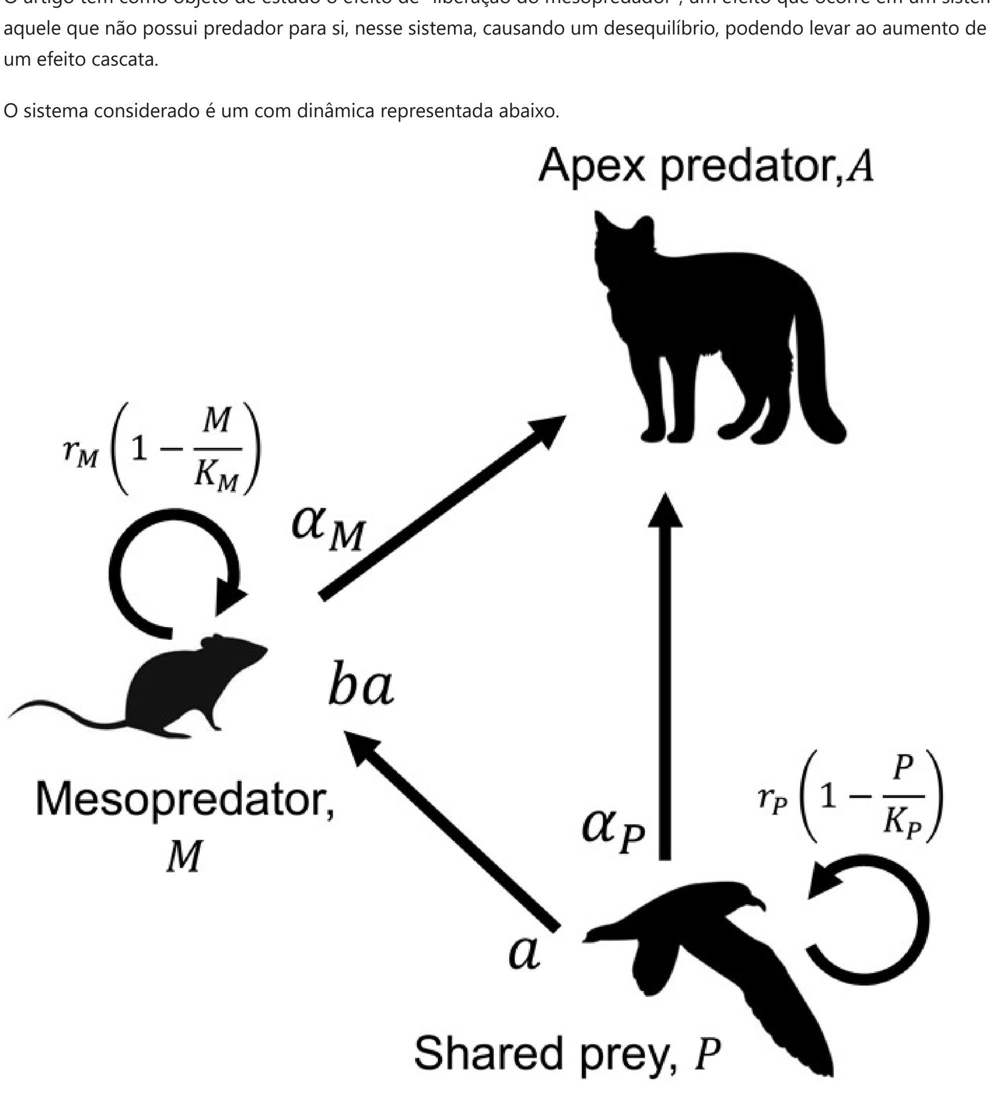
João Victor Dell Agli Floriano, 10799783

Objetivos

Esse projeto tem como objetivo estudar e reproduzir a imagem 3 do artigo "A simple theory for the mesopredator release effect: when does an apex predator protect their shared prey from a mesopredator?" de Gaku Takimoto and Shota Nishijima, publicado na Oikos 2022: e09021.

O artigo tem como objeto de estudo o efeito de "liberação do mesopredador", um efeito que ocorre em um sistema predador-mesopredador-presa. Esse efeito ocorre quando há extinção do "predador de ápice", aquele que não possui predador para si, nesse sistema, causando um desequilíbrio, podendo levar ao aumento de mesopredadores e a extinção de suas presas compartilhadas, levando enfim, à extinção do sistema em um efeito cascata.

O sistema considerado é um com dinâmica representada abaixo.



Modelagem

De acordo com o artigo, o sistema pode ser modelado da seguinte maneira, abseando-se no sistema logístico:

Dinâmica da presa compartilhada: $\frac{dP}{dt} = r_P(1-\frac{P}{K_P})P - \alpha_P AP - aMP$

Dinâmica do mesopredador : $\frac{dM}{dt} = r_M(1-\frac{M}{K_M})M - \alpha_M AM - baPM$

Aqui, P é a densidade de população da presa compartilhada, M a densidade do mesopredador, e A a densidade do predador de ápice.Os K são a capacidade de suporte de cada população, α_P é a porcentagem de ataque per capita do predador de ápice na presa, α_M é a porcentagem de ataque per capita do predador de ápice ao mesopredador, r_P e r_M são as taxas de crescimento intrínsecas de cada população, a é a taxa de ataque do mesopredador à presa, b é a taxa de conversão de presas capturadas em população do mesopredador.

Para uma melhor abordagem desse modelo, façamos uma conversão de unidades para "enxugar" esse modelo. Primeiro, façamos a seguinte transformação:

$p = \frac{P}{K_P}$

$m = \frac{P}{K_M}$

Que transforma as equações em:

$\frac{dp}{dt} = r_P(1-p)p - \alpha_P Ap - aK_m pm$

$\frac{dm}{dt} = r_M(1-m)m - \alpha_M Am - bapmK_P$

Segundo, consideremos agora uma escala de tempo diferente, aonde $\tau = r_P t$:

$\frac{dp}{d\tau} = (1-p)p - \frac{\alpha_P}{r_P}Ap - \frac{aK_m}{r_P}pm$

$\frac{dm}{d\tau} = \frac{r_M}{r_P}(1-m)m - \frac{\alpha_M}{r_M}Am - \frac{baK_P}{r_M}pm$

A partir dessa transformação, podemos ainda renomear algumas constantes:

- $\rho = \frac{r_M}{r_P}$
- $c_P = \frac{\alpha_P}{r_P}$
- $d_P = \frac{aK_m}{r_P}$
- $c_M = \frac{\alpha_M}{r_M}$
- $d_M = \frac{baK_P}{r_M}$

Que enfim, resulta nas seguintes equações:

$\frac{dp}{d\tau} = p[(1-p) - c_P A - d_P m]$

$\frac{dm}{d\tau} = m[\rho(1-m) - c_M A - d_M p]$

Com as equações da dinâmica enfim estabelecidas, podemos analisar o comportamento do mesopredador e da presa à procura do efeito descrito anteriormente.

Equilíbrio

De acordo com o artigo, o comportamento do sistema à procura desse efeito é analisado no equilíbrio, mas especificamente no ponto de coexistência equlibrada entre os dois atores. Analisando as equações acima, isso acontece quando:

$\frac{dp}{d\tau} = 0$

$\frac{dm}{d\tau} = 0$

Ou seja, quando:

$0 = p[(1-p) - c_P A - d_P m]$

$0 = m[\rho(1-m) - c_M A - d_M p]$

Nessa situação, há quatro possibilidades:

1. $p = 0$, $m = 0$
2. $p = 0$, $m \neq 0$
3. $p \neq 0$, $m = 0$
4. $p \neq 0$, $m \neq 0$

A primeira é uma situação óbvia de solução do problema, visto que quando a densidade populacional dos dois atores é zero, não há dinâmica no problema. Sendo assim, analisemos as outras situações.

2. $p = 0$, $m \neq 0$

Com $p = 0$ e $m \neq 0$, temos:

$0 = 0[(1-0) - c_P A - d_P m_1^*]$

$0 = m_1^*[\rho(1-m_1^*) - c_M A - d_M 0]$

Que se desenvolve em:

$0 = \rho(1-m_1^*) - c_M A$

$\rho m_1^* = \rho 1 - c_M A$

$m_1^* = 1 - \frac{c_M A}{\rho}$

3. $p \neq 0$, $m = 0$

Com $p \neq 0$ e $m = 0$, temos:

$0 = p_2^*[(1-p_2^*) - c_P A - d_P 0]$

$0 = 0[\rho(1-0) - c_M A - d_M p_2^*]$

Que se desenvolve em:

$0 = (1-p_2^*) - c_P A$

$p_2^* = 1 - c_P A$

4. $p \neq 0$, $m \neq 0$

Como dito anteriormente, a situação de interesse é o equilíbrio de coexistência entre os dois atores, ou seja, a situação 4.

$0 = p_3^*[(1-p_3^*) - c_P A - d_P m_3^*]$

$0 = 1 - p_3^* - c_P A - d_P m_3^*$

$p_3^* = 1 - c_P A - d_P m_3^*$

$0 = m_3^*[\rho(1-m_3^*) - c_M A - d_M p_3^*]$

$0 = \rho - \rho m_3^* - c_M A - d_M p_3^*$

$0 = \rho - \rho m_3^* - c_M A - d_M(1 - c_P A - d_P m_3^*)$

$0 = \rho - \rho m_3^* - c_M A + d_M c_P A + d_M d_P m_3^* - d_M$

$0 = \rho + m_3^*(d_M d_P - \rho) + (d_M c_P - c_M)A - d_M$

$m_3^*(d_M d_P - \rho) = -\rho - (d_M c_P - c_M)A + d_M$

Portanto:

$m_3^* = \frac{-\rho-(d_M c_P - c_M)A + d_M}{(d_M d_P - \rho)}$

$p_3^* = 1 - c_P A - d_P\frac{-\rho-(d_M c_P - c_M)A + d_M}{(d_M d_P - \rho)}$

Existem, então, quatro situações de equilíbrio:

1. $p_0^* = 0$, $m_0^* = 0$
2. $p_1^* = 0$, $m_1^* = 1 - \frac{c_M A}{\rho}$
3. $p_2^* = 1 - c_P A$, $m_2^* = 0$
4. $p_3^* = 1 - c_P A - d_P\frac{-\rho-(d_M c_P - c_M)A + d_M}{(d_M d_P - \rho)}$, $m_3^* = \frac{-\rho-(d_M c_P - c_M)A + d_M}{(d_M d_P - \rho)}$

Implementação

Implementemos então, essas situações, para quatro casos:

1. $\alpha_p = 0.5$, $K_m = 0.75$
2. $\alpha_p = 0.5$, $K_m = 0.25$
3. $\alpha_p = 2.0$, $K_m = 0.5$
4. $\alpha_p = 0.5$, $K_m = 2.0$

Os demais parâmetros são:

- $r_m = 1.0$
- $r_p = 1.0$
- $a = 1.0$
- $b = 0.1$
- $\alpha_m = 1.0$
- $K_p = 2.0$

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
def m_func(rho, cm, A, dm, dp, cp):
    """Função de cálculo do valor de m no equilíbrio de coexistência."""
    return (-rho - (dm*cp - cm)*A + dm)/(dm*dp - rho)

def p_func(rho, cm, A, dm, dp, cp):
    """Função de cálculo do valor de p no equilíbrio de coexistência."""
    return 1 - cp*A - dp*(-rho - (dm*cp - cm)*A + dm)/(dm*dp - rho)
```

In [ ]:
```
# Parâmetros Crus
rm = 1.0
rp = 1.0
alfa_p = np.array([0.5, 0.5, 2.0, 0.5])
a = 1.0
b = 0.1
Km = np.array([0.75, 0.25, 0.5, 2])
alfa_m = 1.0
Kp = 2.0

# Parâmetros Derivados
rho = rm/rp
cp = alfa_p/rp
dp = a*Km/rp
cm = alfa_m/rm
dm = b*a*Kp/rm
```

In [ ]:
```
def iterator(rho, cm, A, dm, dp, cp):
    """Itera a simulação."""
    p = np.zeros(A.shape)
    m = np.zeros(A.shape)

    # Inicialização dos valores
    p[0] = p_func(rho, cm, A[0], dm, dp, cp)
    m[0] = m_func(rho, cm, A[0], dm, dp, cp)

    # Cálculo para todo A
    for i in range(1, A.shape[0]):

        p_ = 0
        m_ = 0

        if m[i - 1] <= 0.0:
            # Caso o mesopredador se extingua,
            # o equilíbrio agora será com apenas p,
            # obedecendo p_2*
            p_ = (1 - cp*A[i])
        else:
            # Se não, obedece p_3* e m_3*
            p_ = p_func(rho, cm, A[i], dm, dp, cp)


        if p[i - 1] <= 0.0:
            # Caso a presa se extingua,
            # o equilíbrio agora será com apenas m,
            # obedecendo m_1*
            m_ = (1 - (cm/rho)*A[i])
        else:
            # Se não, obedece p_3* e m_3*
            m_ = m_func(rho, cm, A[i], dm, dp, cp)

        # Endereçando população negativa e
        # atualizando os valores de p e m
        p[i] = p_ if p_ >= 0.0 else 0.0
        m[i] = m_ if m_ >= 0.0 else 0.0

    return p, m
```

In [ ]:
```
A = np.arange(0, 3, 0.01)

p0, m0 = iterator(rho, cm, A, dm, dp[0], cp[0])
p1, m1 = iterator(rho, cm, A, dm, dp[1], cp[1])
p2, m2 = iterator(rho, cm, A, dm, dp[2], cp[2])
p3, m3 = iterator(rho, cm, A, dm, dp[3], cp[3])

plt.figure(figsize = (10, 5))
plt.subplot(241)
plt.ylabel("p")
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, p0)

plt.subplot(245)
plt.xlabel("A")
plt.ylabel("m")
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, m0)

plt.subplot(242)
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, p1)

plt.subplot(246)
plt.xlabel("A")
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, m1)

plt.subplot(243)
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, p2)

plt.subplot(247)
plt.xlabel("A")
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, m2)

plt.subplot(244)
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, p3)

plt.subplot(248)
plt.xlabel("A")
plt.xlim(0.0, 3.0)
plt.ylim(0.0, 1.0)
plt.plot(A, m3)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1df5ef75fc0>]
```