Group: Amalia Bogri, Christian Berrig & Jonas Bolduan

Trophic control

Trophic chain: model parameters

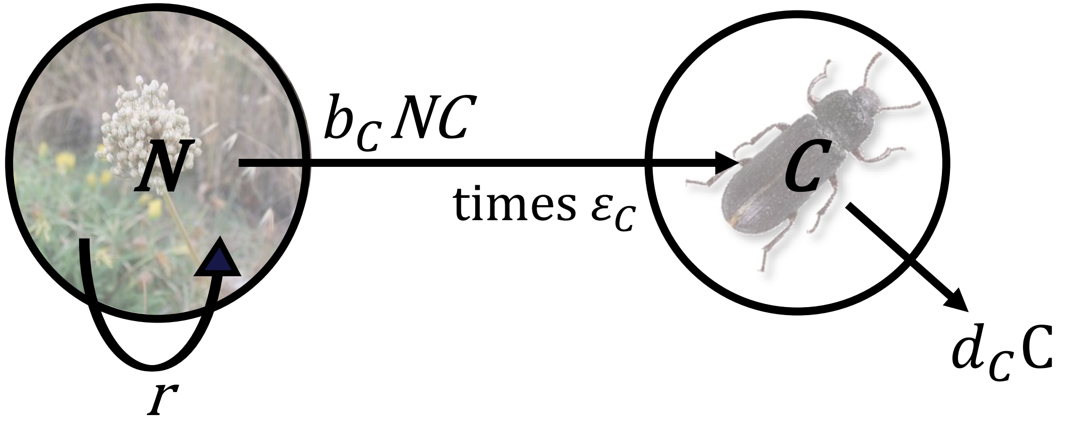
PRODUCERS -Plants

CONSUMER -Beetle

PREDATOR -Lizard

Parameter	Unit	Value
Producer density, N	kg/m^2	
Carrying capacity, K	kg/m^2	15
Growth rate, r	$1/\mathrm{year}$	1
Consumer density, C	$\#_C/m^2$	
Clearance rate, b_C	$m^2/\#_C/year$	0.07
Death rate, d_C	$1/\mathrm{year}$	1/2
Efficiency, ε_C	$\#_C/kg$	0.8
Predator density, C	$\#_P/m^2$	
Clearance rate, b_P	$m^2/\#_P/year$	0.10
Death rate, d_P	$1/\mathrm{year}$	1/6
Efficiency, ε_P	$\#_P/\#_C$	0.7

Lotka-Volterra equations: Producers-Consumers

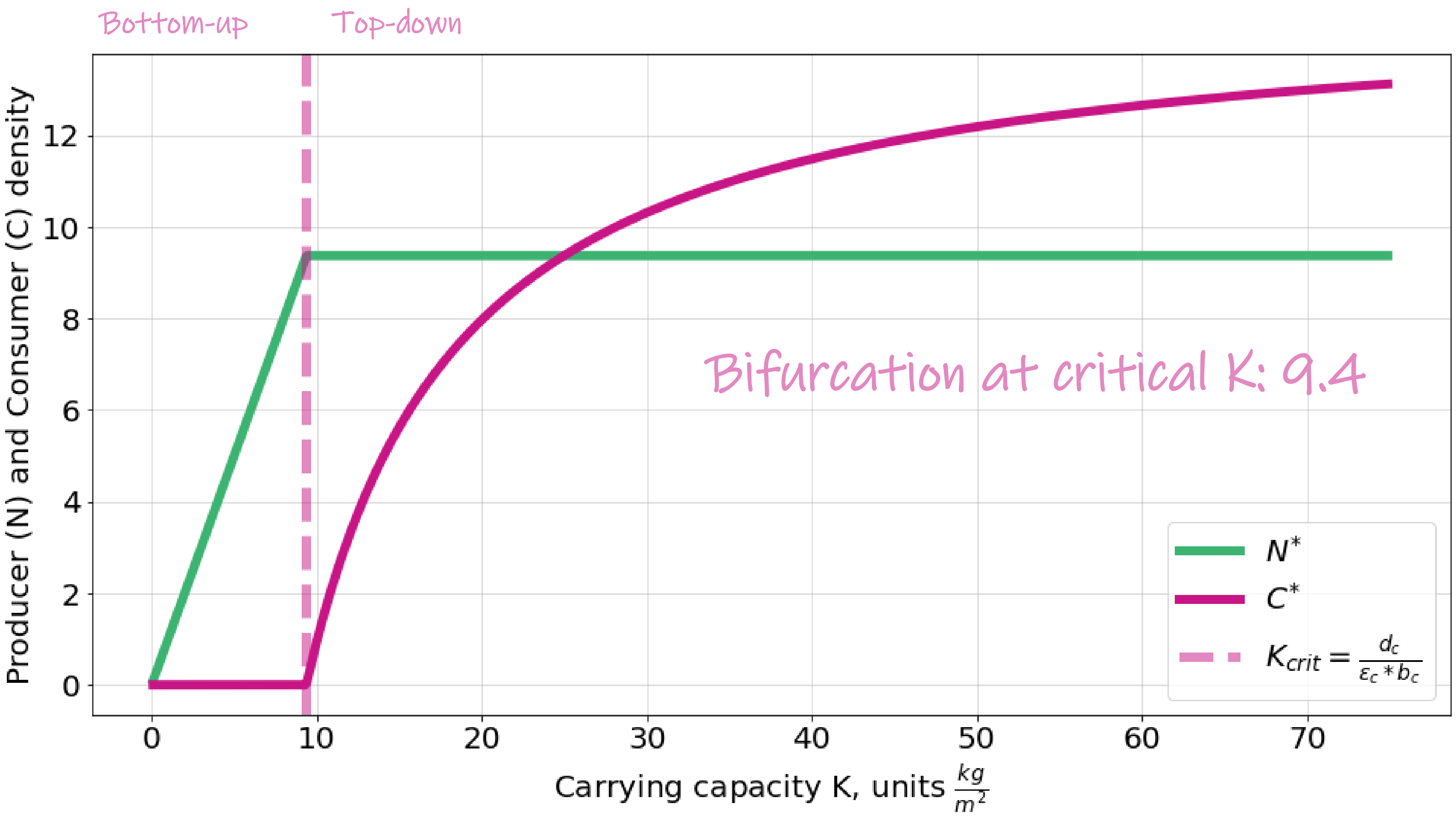


$$\frac{dN}{dt} = r\left(1 - \frac{N}{K}\right)N - b_c CN$$

$$\frac{dC}{dt} = \mathcal{E}_c b_c CN - d_c C$$

Region	$\bar{N}$	$\bar{C}$
Trivial	0	0
$K < \frac{d_c}{\mathcal{E}_c b_c}$	K	0
$K > \frac{d_c}{\mathcal{E}_c b_c}$	$\frac{d_c}{\mathcal{E}_c b_c}$	$\frac{r}{b_c}(1 - \frac{d_c}{\mathcal{E}_c b_c K})$

Bifurcation diagram (K): Producers-Consumers



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

$$\frac{dC}{dt} = \mathcal{E}_c b_c C N - d_c C$$

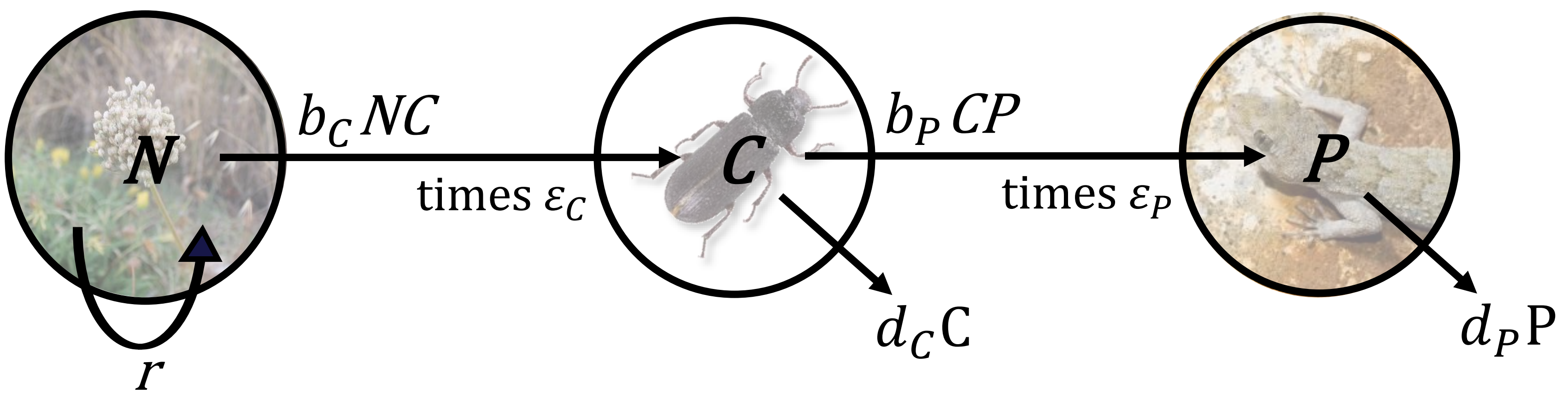
Region	$\bar{N}$	$\bar{C}$
Trivial	0	0
$K < \frac{d_c}{\mathcal{E}_c b_c}$	K	0
$K > \frac{d_c}{\mathcal{E}_c b_c}$	$\frac{d_c}{\mathcal{E}_c b_c}$	$\frac{r}{b_c}(1 - \frac{d_c}{\mathcal{E}_c b_c K})$

Python code

```
# Import stuff:
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np

from sympy import Symbol
from sympy.solvers import solve

fs_label = 20
parameters = {
                'figure.titlesize': fs_label+6,
                'axes.labelsize': fs_label,
                'axes.titlesize': fs_label+4,
                'xtick.labelsize': fs_label,
                'ytick.labelsize': fs_label,
                'legend.fontsize': fs_label,
                'lines.linewidth': 6,
            }
plt.rcParams.update(parameters)



# Question 1.

t = np.linspace(0, 1000, 10000)
# we run it for so long bc we want to be sure the system reaches equilibrium

r = 1           # Growth rate of producers. Dimension: 1/t, Unit: 1/year
K = 15          # Carrying capacity of producers. Dimension: weight/area, Unit: kg/sqm
bc = 1/15       # Clearance rate of consumers. Dimension: area/time/consumer, Unit:
sqm/year/#consumers
dc = 1/2        # Death rate of consumers. Dimension: 1/t, Unit: 1/year
epsc = 0.8      # Efficiency of consumption. Dimension: consumers/producers, Unit: #consumers/kg

def deriv(state, t, *params):
    N, C = state
    r, K, bc, epsc, dc = params
    dNdt = r*(1-N/K)*N - bc*C*N
    dCdt = epsc*bc*C*N - dc*C
    return np.array([dNdt, dCdt])
```

```
# Question 2: Making bifurcation diagram

N_star, C_star = K, 0.02 # initial N and C are very close to equilibrium, so that the computations are faster

k_list = np.arange(0.1, 75, 0.1) # make array of K values
eq_list = [] # make list to place the values of N and C when they reach equilibrium
for k in k_list:
    params = (r, k, bc, epsc, dc)
    state_init = np.array([N_star, C_star])

    ns = odeint(deriv, state_init, t, args=params)
    N, C = ns.T
    N_star, C_star = N[-1], C[-1]
    # assuming that all runs reached equilibrium, we save the last value of N and C as the equilibria
    eq_list.append([N_star, C_star])
    N_star += 0.1
    # Then we pertrub a bit, because if we start from the exact equilibrium, the simulation gets stuck
    C_star += 0.1

N_eq, C_eq = np.array(eq_list).T # Put the list of equilibria in an array, and transpose

fig, ax = plt.subplots(figsize=(16,8))
ax.plot(k_list, N_eq, label="$N^{*}$", color = 'mediumseagreen')
ax.plot(k_list, C_eq, label="$C^{*}$", color = 'mediumvioletred')
ax.set_xlabel("Carrying capacity K, units $\\frac{kg}{m^{2}}$")
ax.set_ylabel("Producer (N) and Consumer (C) density")
ax.axvline(K_crit, alpha=0.5, linestyle="--", color='mediumvioletred', label="$K_{crit} =
\\frac{d_{c}}{\\varepsilon_{c}*b_{c}}$")
ax.legend()
ax.grid(alpha = 0.5)
```

Lotka-Volterra equations: Producers-Consumers-Predators



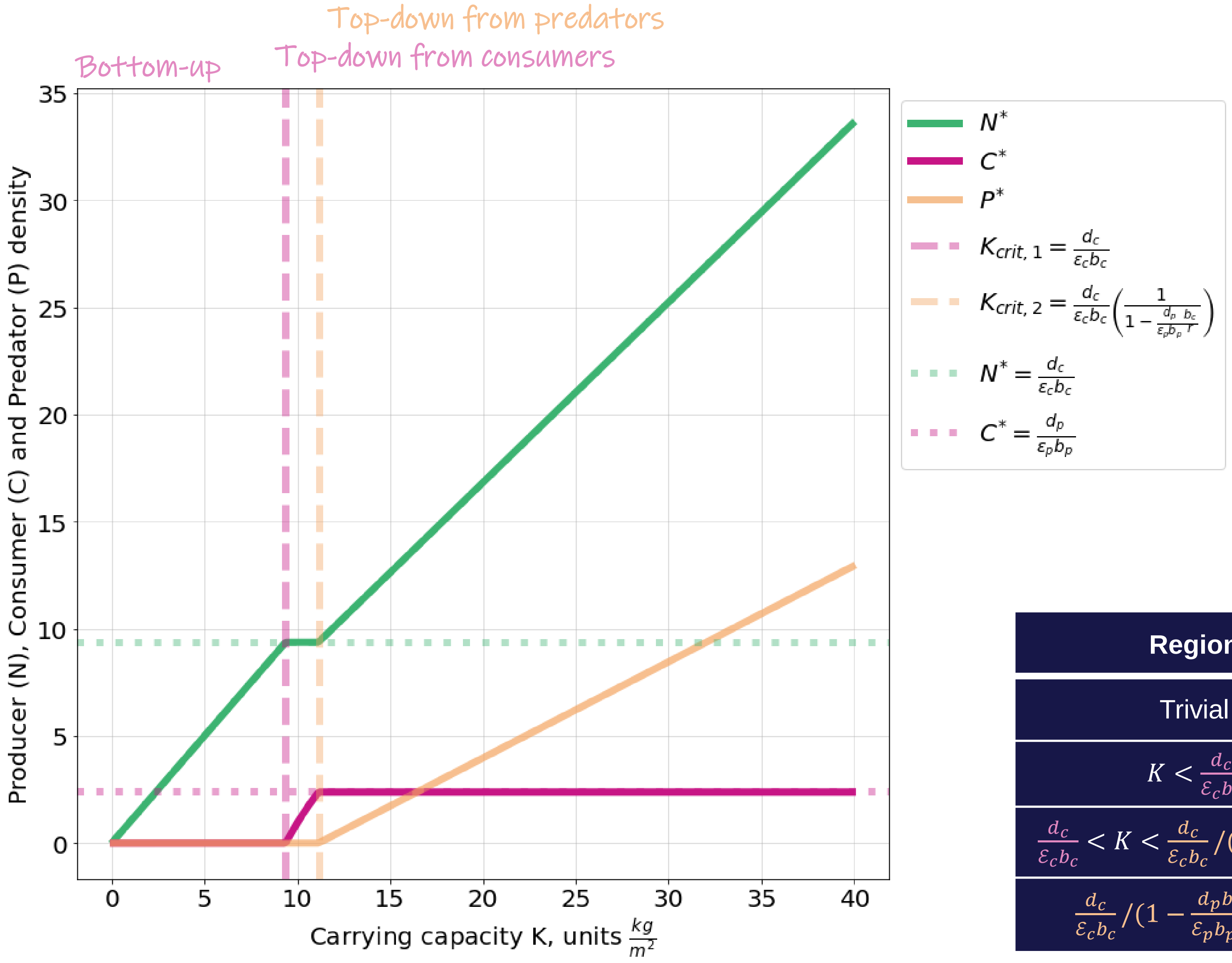
$$\frac{dN}{dt} = r\left(1 - \frac{N}{K}\right)N - b_c CN$$

$$\frac{dC}{dt} = \mathcal{E}_c b_c CN - d_c C - b_p CP$$

$$\frac{dP}{dt} = \mathcal{E}_p b_p CP - d_p P$$

Region	$\bar{N}$	$\bar{C}$	$\bar{P}$
Trivial	0	0	0
$K < \frac{d_c}{\mathcal{E}_c b_c}$	K	0	0
$\frac{d_c}{\mathcal{E}_c b_c} < K < \frac{d_c}{\mathcal{E}_c b_c}/(1 - \frac{d_p b_c}{\mathcal{E}_p b_p r})$	$\frac{d_c}{\mathcal{E}_c b_c}$	$\frac{r}{b_c}(1 - \frac{d_c}{\mathcal{E}_c b_c K})$	0
$\frac{d_c}{\mathcal{E}_c b_c}/(1 - \frac{d_p b_c}{\mathcal{E}_p b_p r}) < K$	$\mathrm{K}(1 - \frac{d_p b_c}{\mathcal{E}_p b_p r})$	$\frac{d_p}{\mathcal{E}_p b_p}$	$\frac{\mathcal{E}_c b_c}{b_p}\mathrm{K}\left(1 - \frac{d_p b_c}{\mathcal{E}_p b_p r}\right) - \frac{d_c}{b_p}$

Bifurcation diagram (K): Producers-Consumers-Predators



1st bifurcation at critical K: 9.4

2nd bifurcation at critical K: 11.2

Region	$\bar{N}$	$\bar{C}$	$\bar{P}$
Trivial	0	0	0
$K < \frac{d_c}{\varepsilon_c b_c}$	K	0	0
$\frac{d_c}{\varepsilon_c b_c} < K < \frac{d_c}{\varepsilon_c b_c}/(1-\frac{d_p b_c}{\varepsilon_p b_p r})$	$\frac{d_c}{\varepsilon_c b_c}$	$\frac{r}{b_c}(1-\frac{d_c}{\varepsilon_c b_c K})$	0
$\frac{d_c}{\varepsilon_c b_c}/(1-\frac{d_p b_c}{\varepsilon_p b_p r}) < K$	$\mathrm{K}(1-\frac{d_p b_c}{\varepsilon_p b_p r})$	$\frac{d_p}{\varepsilon_p b_p}$	$\frac{\varepsilon_c b_c}{b_p}\mathrm{K}\left(1-\frac{d_p b_c}{\varepsilon_p b_p r}\right)-\frac{d_c}{b_p}$

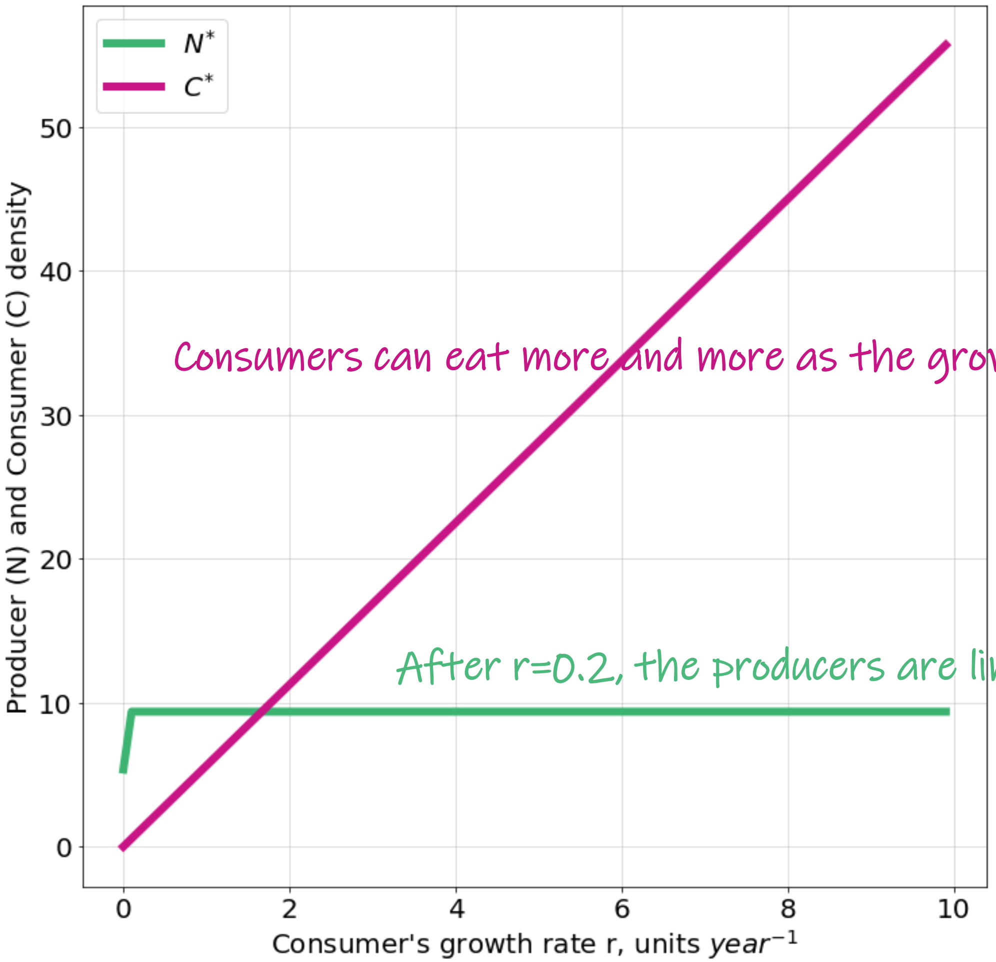
Python code

```
# Question 3: Add a predator
t = np.linspace(0, 1000, 10000)

r = 1           # Growth rate of producers. Dimension: 1/t, Unit: 1/year
K = 15          # Carrying capacity of producers. Dimension: weight/area, Unit: kg/sqm
bc = 1/15       # Clearance rate of consumers. Dimension: area/time/consumer, Unit:
sqm/year/#consumers
dc = 1/2        # Death rate of consumers. Dimension: 1/t, Unit: 1/year
epsc = 0.8      # Efficiency of consumption. Dimension: consumers/producers, Unit: #consumers/kg

bp = 1/10       # Clearance rate of predators. Dimension: area/time/predator, Unit:
sqm/year/#predators
dp = 1/6        # Death rate of consumers. Dimension: 1/t, Unit: 1/year
epsp = 0.7      # Efficiency of consumption. Dimension: predators/consumers, Unit:
#predators/#consumers


def deriv_predator(state, t, *params):
    N, C, P = state
    r, K, bc, epsc, dc, bp, epsp, dp = params
    dNdt = r*(1-N/K)*N - bc*C*N
    dCdt = epsc*bc*C*N - dc*C - bp*C*P
    dPdt = epsp*bp*C*P - dp*P
    return np.array([dNdt, dCdt, dPdt])


N_star, C_star, P_star = K, 0.01, 0.01

k_list = np.arange(0.1, 40, 0.1)
eq_list = []
for k in k_list:
    params = (r, k, bc, epsc, dc, bp, epsp, dp)
    state_init = np.array([N_star, C_star, P_star])

    ns = odeint(deriv_predator, state_init, t, args=params)
    N, C, P = ns.T
    N_star, C_star, P_star = N[-1], C[-1], P[-1]
    eq_list.append([N_star, C_star, P_star])
    N_star += 0.1
    C_star += 0.1
    P_star += 0.1

N_eq, C_eq, P_eq = np.array(eq_list).T
```

```
r_krit = (bc*dp)/(epsp*bp)

K_crit1 = dc/(epsc*bc)
K_crit2 = (dc/(epsc*bc))/(1-C_star*bc/r)

C_crit = dp/(epsp*bp)
N_crit = K_crit1

fig, ax = plt.subplots(figsize=(12,12))
ax.plot(k_list, N_eq, label="$N^{*}$", color = 'mediumseagreen')
ax.plot(k_list, C_eq, label="$C^{*}$", color = 'mediumvioletred')
ax.plot(k_list, P_eq, label="$P^{*}$", color = 'sandybrown', alpha = 0.7)

crit_line_style = {"alpha":0.4, "linestyle":"--"}
ax.axvline(K_crit1, **crit_line_style, color='mediumvioletred',
    label="$K_{crit, 1} = \\frac{d_{c}}{\\varepsilon_{c} b_{c}}$")
ax.axvline(K_crit2, **crit_line_style, color='sandybrown',
    label="$K_{crit, 2} = \\frac{d_{c}}{\\varepsilon_{c} b_{c}} \\left( \\frac{1}{1-
\\frac{d_{p}}{\\varepsilon_{p} b_{p}} \\frac{b_{c}}{r}} \\right)$")

crit_line_style = {"alpha":0.4, "linestyle":":"}
ax.axhline(N_crit, **crit_line_style, color = 'mediumseagreen',
    label="$N^{*} = \\frac{d_{c}}{\\varepsilon_{c} b_{c}}$")
ax.axhline(C_crit, **crit_line_style, color = 'mediumvioletred',
    label="$C^{*} = \\frac{d_{p}}{\\varepsilon_{p} b_{p}}$")


ax.legend(bbox_to_anchor=(1,1))
ax.grid(alpha = 0.5)

ax.set_xlabel("Carrying capacity K, units $\\frac{kg}{m^{2}}$")
ax.set_ylabel("Producer (N), Consumer (C) and Predator (P) density")
```

Bifurcation diagram (r): Producers-Consumers

K = 15



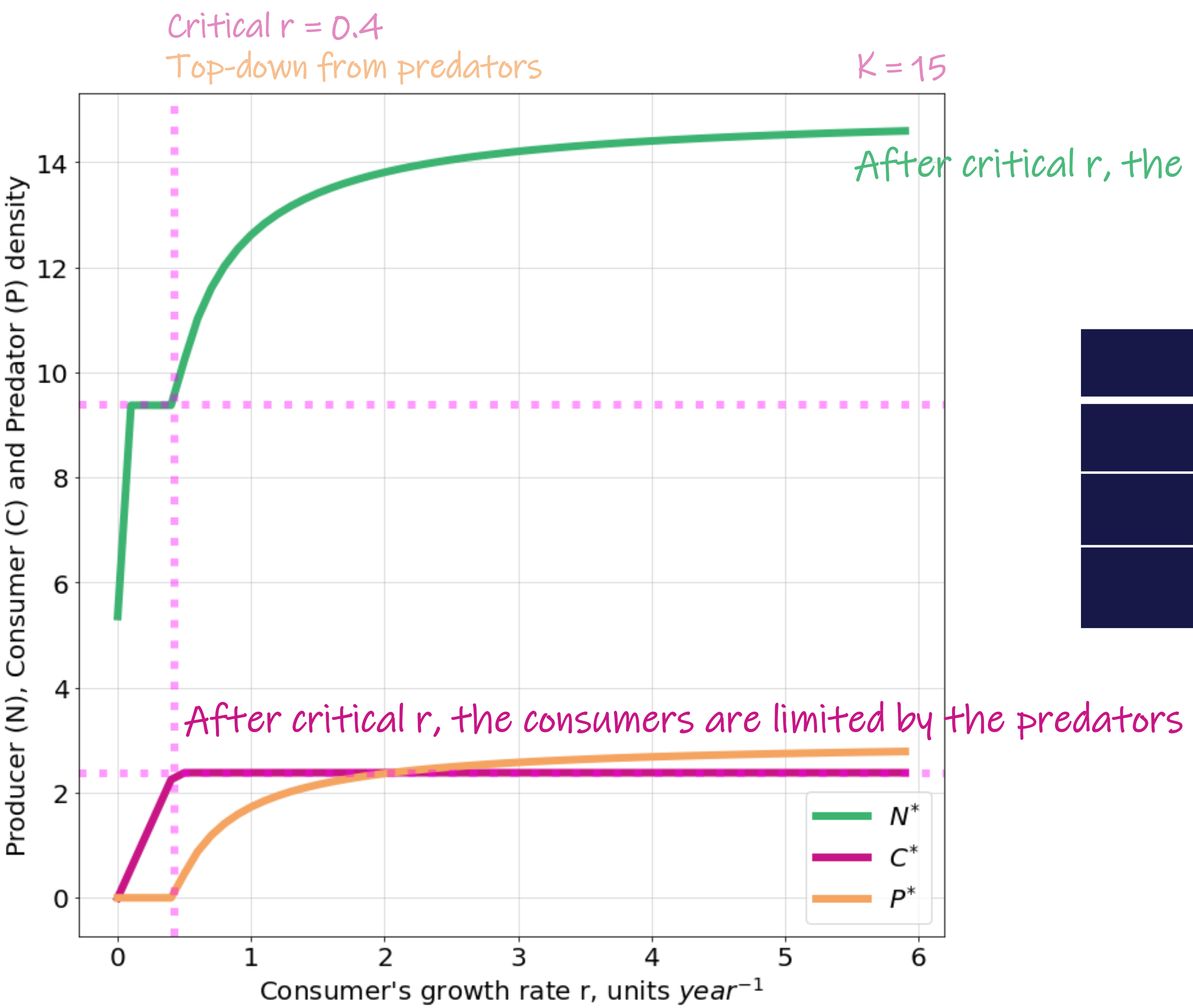
Consumers can eat more and more as the growth rate of producers increases

After r=0.2, the producers are limited by their carrying capacity

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

$$\frac{dC}{dt} = \mathcal{E}_c b_c C N - d_c C$$

Bifurcation diagram (r): Producers-Consumers-Predators



After critical r, the producers grow until limited by their carrying capacity

$$N_{crit} = \frac{d_c}{\varepsilon_c b_c}, \; C_{crit} = \frac{d_p}{\varepsilon_p b_p}, \; r_{crit} = C_{crit} b_c / (1 - \frac{N_{crit}}{K})$$

Region	$\bar{N}$	$\bar{C}$	$\bar{P}$
Trivial	0	0	0
$r < r_{crit}$	$\frac{d_c}{\varepsilon_c b_c}$	$\frac{r}{b_c}(1 - \frac{d_c}{\varepsilon_c b_c K})$	0
$r > r_{crit}$	$K(1 - \frac{d_p b_c}{\varepsilon_p b_p r})$	$\frac{d_p}{\varepsilon_p b_p}$	$\frac{\varepsilon_c b_c}{b_p} K\left(1 - \frac{d_p b_c}{\varepsilon_p b_p r}\right) - \frac{d_c}{b_p}$

Python code

```
# Question 4: Bifurcation along another parameter

# Producer-consumer
N_star, C_star = K, 0.01

r_list = np.arange(0.0, 10, 0.1)
r_eq_list = []
for r_tmp in r_list:
    params = (r_tmp, K, bc, epsc, dc)
    state_init = np.array([N_star, C_star])

    ns = odeint(deriv, state_init, t, args=params)
    N, C = ns.T
    N_star, C_star = N[-1], C[-1]
    r_eq_list.append([N_star, C_star])
    N_star += 0.1
    C_star += 0.1


N_eq, C_eq = np.array(r_eq_list).T



fig, ax = plt.subplots(figsize=(12,12))
ax.plot(r_list, N_eq, label="$N^{*}$", color = 'mediumseagreen')
ax.plot(r_list, C_eq, label="$C^{*}$", color='mediumvioletred')
ax.legend()
ax.set_xlabel("Consumer's growth rate r, units ${year^{-1}}$")
ax.set_ylabel("Producer (N) and Consumer (C) density")
ax.grid(alpha = 0.5)
```

```
#Producer-Consumer-Predator
N_star, C_star, P_star = K, 0.01, 0.01

r_list_predator = np.arange(0.0, 6, 0.1)
r_eq_list_predator = []
for r_tmp in r_list_predator:
    params = (r_tmp, K, bc, epsc, dc, bp, epsp, dp)
    state_init = np.array([N_star, C_star, P_star])

    ns = odeint(deriv_predator, state_init, t, args=params)
    N, C, P = ns.T
    N_star, C_star, P_star = N[-1], C[-1], P[-1]
    r_eq_list_predator.append([N_star, C_star, P_star])
    N_star += 0.1
    C_star += 0.1
    P_star += 0.1

N_eq, C_eq, P_eq = np.array(r_eq_list_predator).T



r_crit1 = bc/(1-N_crit/K)*C_crit
C_crit = dp/(epsp*bp)
N_crit = dc/(epsc*bc)



fig, ax = plt.subplots(figsize=(12,12))
ax.plot(r_list_predator, N_eq, label="$N^{*}$", color = 'mediumseagreen')
ax.plot(r_list_predator, C_eq, label="$C^{*}$", color='mediumvioletred')
ax.plot(r_list_predator, P_eq, label="$P^{*}$", color = 'sandybrown')
#ax.plot(r_list_predator[:50], r_list_predator[:50]*C_crit2, label="$P^{*}$", alpha=0.5,
linestyle=":")

ax.axvline(r_crit1, **crit_line_style, color="magenta")
ax.axhline(C_crit, **crit_line_style, color="magenta")
ax.axhline(N_crit, **crit_line_style, color="magenta")
ax.legend()
ax.grid(alpha = 0.5)
ax.set_xlabel("Consumer's growth rate r, units ${year^{-1}}$")
ax.set_ylabel("Producer (N), Consumer (C) and Predator (P) density")
```