Modelling microbial ecosystem stability through metabolite leakage

Léo Besançon & Larissa Geiser

with Snorre Sulheim

Paradox of the plankton

- **Competitive exclusion** :
  if two species compete for the same resource, one of them will be driven to extinction
- How can **many species** thrive in environments with **limited resources** ?

→ Can we model it ?

→ When is it stable ?

Consumer-resource modelling

Energy intake → $$J_{i\alpha}^{in} = C_{i\alpha} \frac{R_\alpha}{R_\alpha + K_{i\alpha}}$$

Population variation → $$\frac{dN_i}{dt} = g_i N_i \left[\sum_\alpha w_\alpha (1 - l_\alpha) J_{i\alpha}^{in} - m_i\right] - N_i d$$

Resource variation → $$\frac{dR_\alpha}{dt} = (R_\alpha^0 - R_\alpha)d - \sum_i N_i J_{i\alpha}^{in} + \sum_{i,\beta} D_{i\alpha\beta} \frac{w_\beta}{w_\alpha} l_\beta N_i J_{i\beta}^{in}$$

- $J_{i\alpha}^{in}$: *energy intake*
- $C_{i\alpha}$: *maximum uptake of resource*
- R_α : *resource concentration*
- $K_{i\alpha}$: *affinity for resource*
- N_i : *population of species i*
- g_i : *conversion factor (energy uptake to growth rate)*
- w_α : *energy content of* α
- l_α : *leakage fraction*
- m_i : *minimal energy uptake for maintenance*
- d : *dilution rate*
- $D_{i\alpha\beta}$: *fraction of byproduct*

Consumer-resource modelling but simpler

Energy intake → $$J_{i\alpha}^{in} = C_{i\alpha} \frac{R_\alpha}{R_\alpha + K_{i\alpha}}$$

Population variation → $$\frac{dN_i}{dt} = g_i N_i \left[\sum_\alpha w_\alpha (1 - l_\alpha) J_{i\alpha}^{in} - m_i \right] - N_i d$$

Resource variation → $$\frac{dR_\alpha}{dt} = (R_\alpha^0 - R_\alpha) d - \sum_i N_i J_{i\alpha}^{in} + \sum_{i,\beta} D_{i\alpha\beta} \frac{w_\beta}{w_\alpha} l_\beta N_i J_{i\beta}^{in}$$

- $J_{i\alpha}^{in}$: *energy intake*
- $C_{i\alpha}$: *maximum uptake of resource*
- R_α : *resource concentration*
- $K_{i\alpha}$: *affinity for resource*
- N_i : *population of species i*
- g_i : *conversion factor (energy uptake to growth rate)*
- w_α : *energy content of* α
- l_α : *leakage fraction*
- m_i : *minimal energy uptake for maintenance*
- d : *dilution rate*
- $D_{i\alpha\beta}$: *fraction of byproduct*

Full model

```python
# complete model
def CR_model(t, y, c, K, w, l, m, g, d, D):

  N = y[:n_species]
  R = y[n_species:]
  R[R<0] = 0 #no negative values
  N[N<1e-6] = 0 #no negative values
  J = np.zeros((n_species, n_cs)) # uptake of resource i by species a
  for i in range(n_species): # for every resource
    for a in range(n_cs): # for every species
      J[i,a] = c[i,a]*R[a]/(R[a]+K[a]) # uptake ~ max uptake, resource concentration, affinity

  dNdt = np.zeros(n_species) # variation in population ~ resource uptake
  for i in range(n_species): # for every species
    growth_intake = np.sum([(J[i, a]*w[a]*(1-l[i,a]))-m[i]for a in range(n_cs)]) # growth ~ uptake, energy content of resource, leakage, maintenance
                                                                                 # -> sum of every resource
    dNdt[i] = g[i]*N[i]*growth_intake-(N[i]*d) # population evolution ~ conversion rate, population, growth intake, dilution rate

  dRdt = np.zeros(n_cs) # variation in resource ~ consumption and byproduct production
  for a in range(n_cs): # for every resource
    consumption = np.sum([N[i]*J[i,a] for i in range(n_species)]) # consumption ~ population, resource uptake
                                                                  # -> sum for every species
    byproduction_tot = np.zeros((n_species, n_cs)) # byproduct production
    for i in range(n_species): # for every species
      for b in range(n_cs): # for every resource
        byproduction_tot[i,b] = D[i,a,b]*w[b]*l[i,b]*N[i]*J[i,b]/w[a] # production ~ fraction converted, energy content, leakage, population, uptake
    byproduction = np.sum(byproduction_tot) # add all byproduct production FOR ONE RESOURCE (see [24])
    dRdt[a] = (R0[a]-R[a])*d - consumption + byproduction # resource evolution ~ dilution, consumption, byproduct production

  return np.concatenate((dNdt,dRdt))
```

Population and resources (can’t go below 0)

Resource uptake for every species and resource

Population variation for every species

Resource variation for every resource

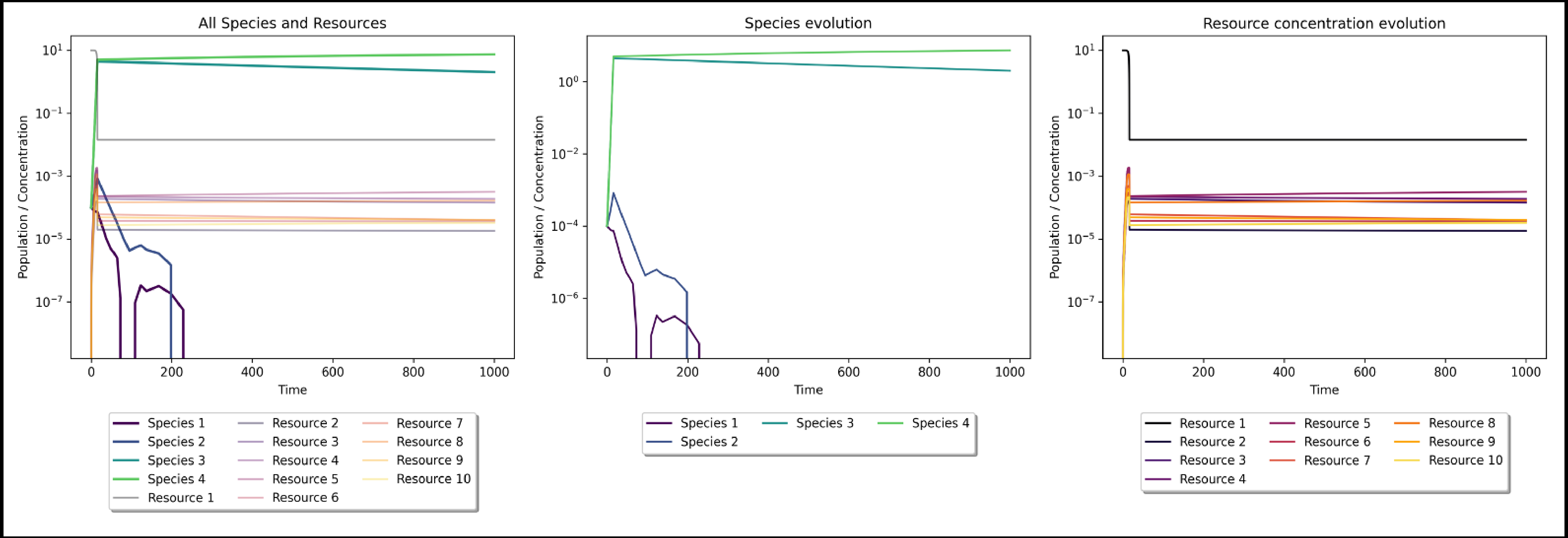
```python
t_span = (0,100)
solutions = solve_ivp(CR_model, t_span, y0, args=(c, K, w, l, m, g, d, D), method = "BDF")
```

Time span of the simulation

SciPy function for solving differential equations

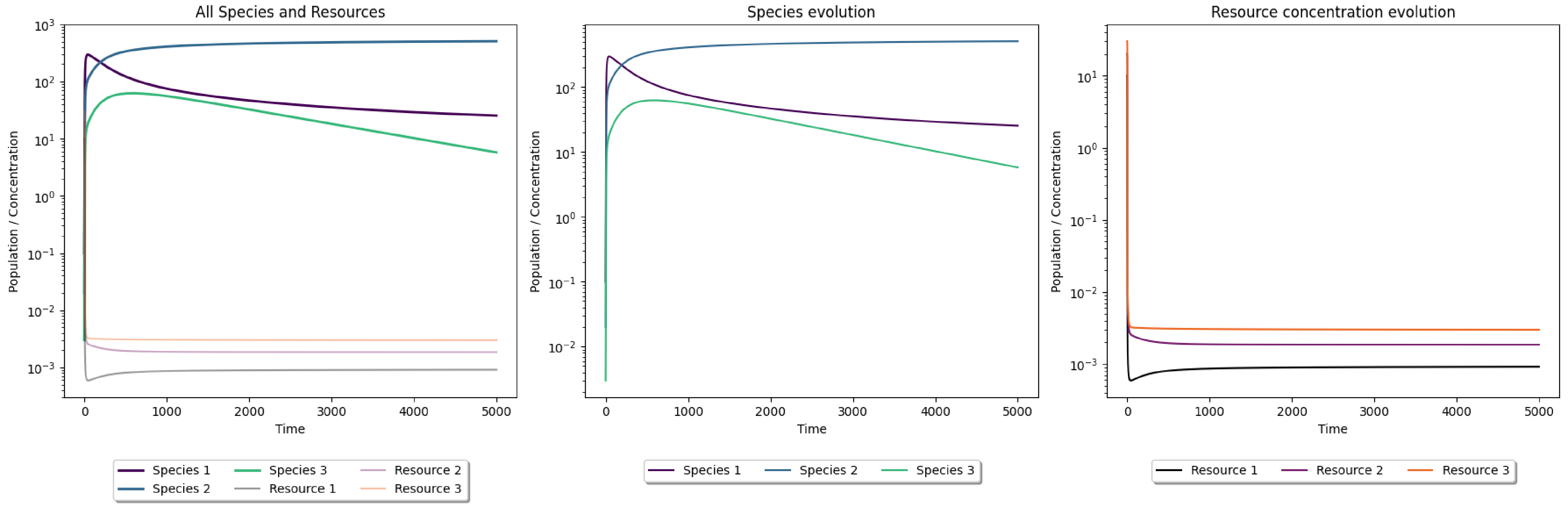
First approach – shots in the dark

Random parameters



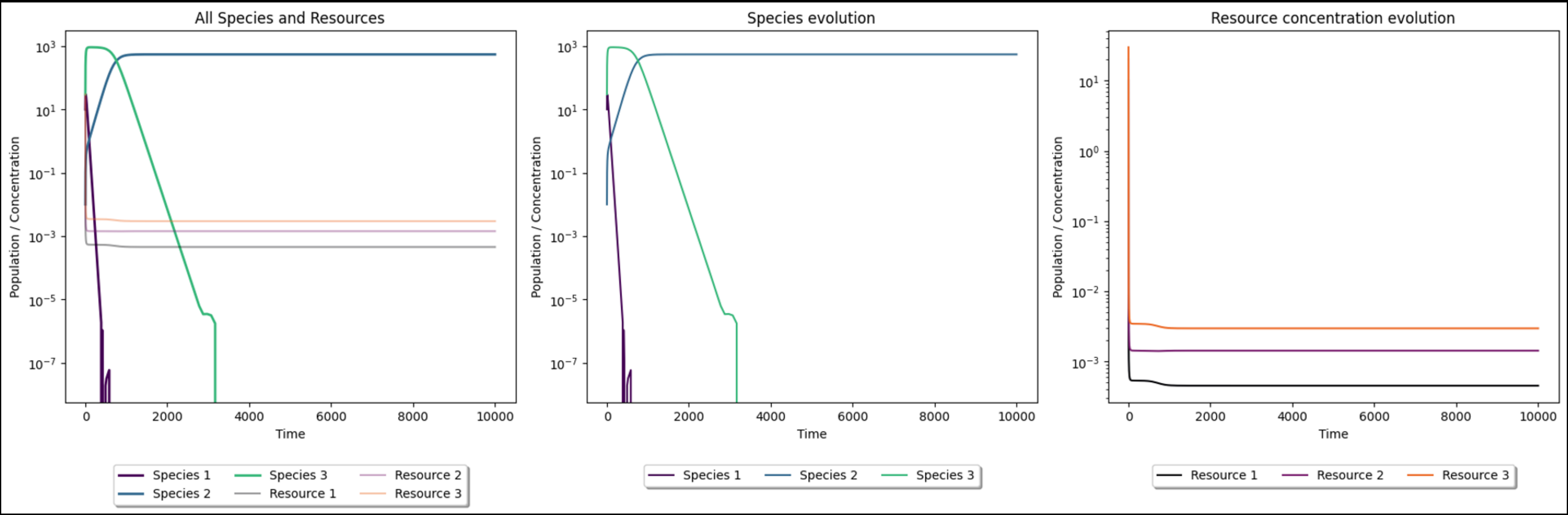
First approach – shots in the dark

Random parameters



First approach – shots in the dark

Random parameters



2^{nd} approach : brute force

Randomization and statistical tests

```
dNdt[i] = g[i]*N[i]*growth_intake-(N[i]*d)
```

g[x]*growth_intake[x] = g[y]*growth_intake[y]

Equilibrium !

2^{nd} approach : brute force

Randomization and statistical tests

Simulations x1000

Cohabitation

Domination

	sim	g	c	D	alive	dead	J	ggrowth	J_dead	ggrowth_dead
0	simulation 0	[0.6545044920636776, 0.4366606661442731, 0.651...	[[0.31208447811936935, 0.33889299315413646, 0....	[[[0.1938064799496923, 0.0003500693929802592, ...	[0, 1]	[2]	[[0.03180972923805029, 0.07473907597080215, 0....	[0.09966516358026334, 0.10000022396959549]	[0.00035211634991528585, 0.03946129692256482, ...	[0.09926965922294585]
1	simulation 2	[0.9773072571801051, 0.8654501833204424, 0.640...	[[0.7646501915503346, 0.027114127731092452, 0....	[[[0.03341973548732526, 0.24473090638746683, 0...	[0, 1]	[2]	[[0.06284507929175859, 0.0024045583268984326, ...	[0.1, 0.10000000000000003]	[0.020738528855554297, 0.021707063450876033, 0...	[0.06555820871884734]
2	simulation 5	[0.047899308333432034, 0.39708651231459957, 0....	[[0.17800962618540928, 0.21174201011498506, 0....	[[[0.042013684688096256, 0.1985929280007215, 0...	[1, 2]	[0]	[[0.11375614920339742, 0.04582038033840712, 0....	[0.1000000006732733, 0.09999999965080827]	[0.0399309497285032, 0.012547277401372953, 0.0...	[0.012062688399241354]
3	simulation 6	[0.7023985656841081, 0.8979012646752885, 0.521...	[[0.1319557685523638, 0.04651175151429354, 0.3...	[[[0.04340905280672976, 0.34723507286198046, 0...	[1, 2]	[0]	[[0.011929000617884246, 0.03861392121360965, 0...	[0.09999999999999974, 0.1000000000000004]	[0.019978545265274077, 0.00305691973458804, 0....	[0.07822670412855665]
4	simulation 8	[0.4066353592076434, 0.4963184863499276, 0.778...	[[0.7421806460450815, 0.9805162186926151, 0.45...	[[[0.00014079539144785022, 0.19301513422582164...	[0, 2]	[1]	[[0.16450514443232306, 0.03815268758944623, 0....	[0.0999987029054839, 0.10000001658761903]	[0.04592468419692121, 0.017286870271099247, 0....	[0.12205334272876725]

2^{nd} approach : brute force

Randomization and statistical tests

g*growth_intake	Cohabitation	Domination
Alive vs Alive Or Dead vs Dead	Insignificant pvalue	Insignificant pvalue
Alive vs Dead	Significant pvalue	Significant pvalue

2^{nd} approach : brute force

Randomization and statistical tests

ggrowth
[0.1000000000000492, 0.0999999999985125]
[0.1, 0.1000000000000002]
[0.1000093042796803, 0.09988607853201728, 0.09...
[0.1000000000001025, 0.1000000000000239]

```
dNdt[i] = g[i]*N[i]*growth_intake-(N[i]*d)
```

g[x]*growth_intake[x] = g[y]*growth_intake[y] = d

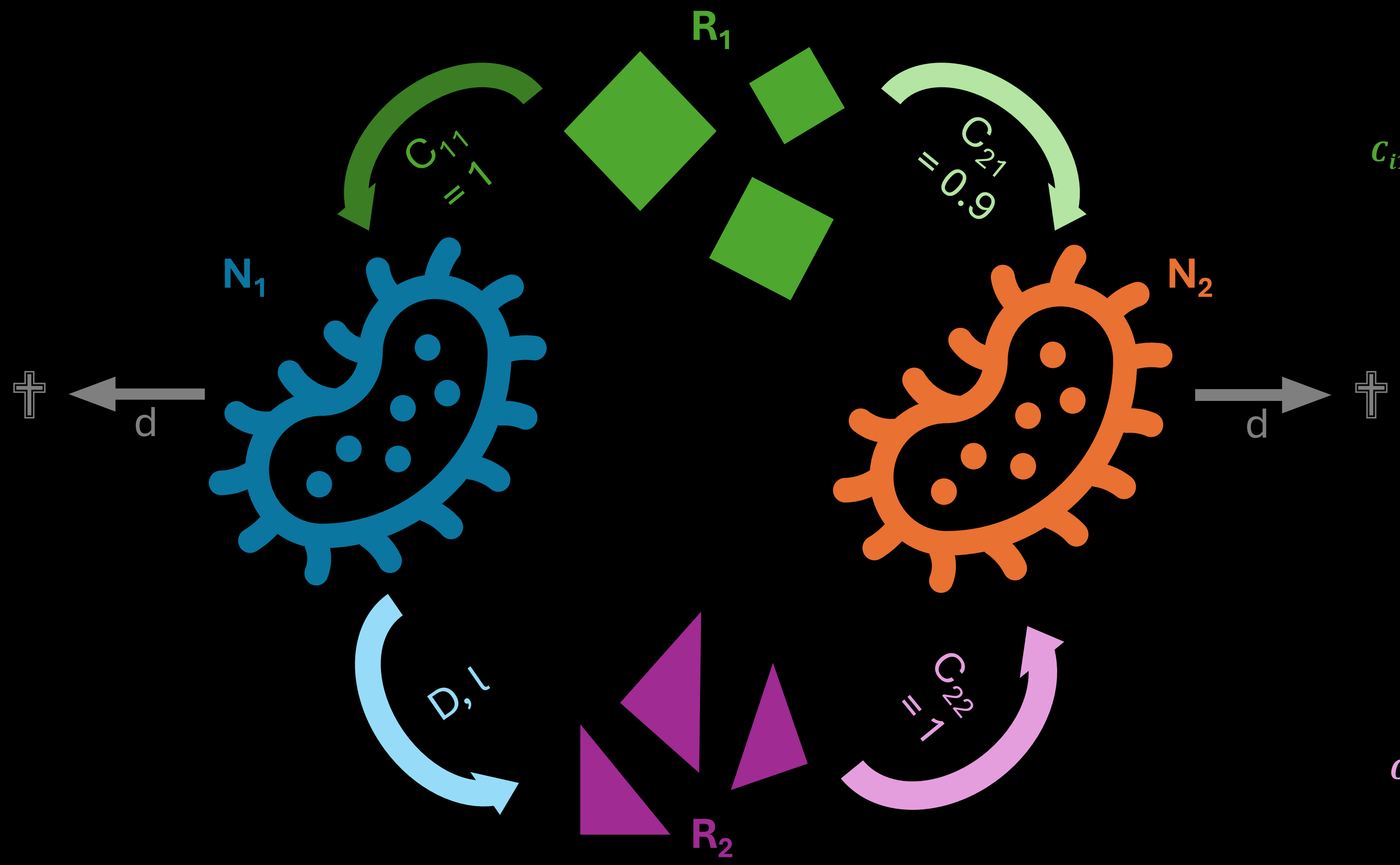
DN/dt = 0 = Equilibrium !

d = 0.1

2nd approach : brute force

TOO LONG
AND
NOT EFFECTIVE

3rd approach : the answers are in the numbers



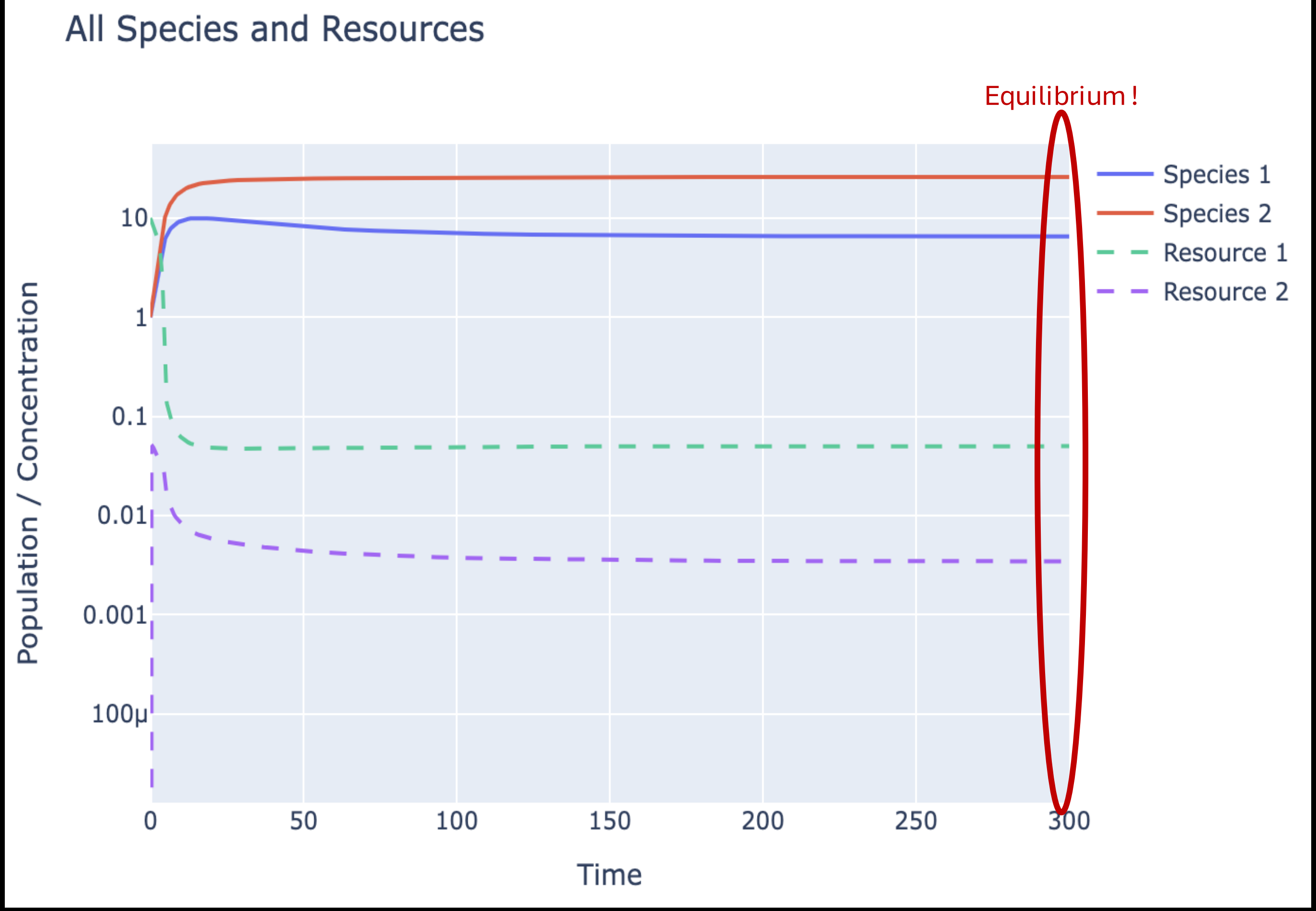
C_{i1} = *maximum uptake of resource 1*

d = *dilution (decay)*

D : *fraction of byproduct*
l : *leakage*
C_{i2} = *maximum uptake of resource 2*

3rd approach : the answers are in the numbers

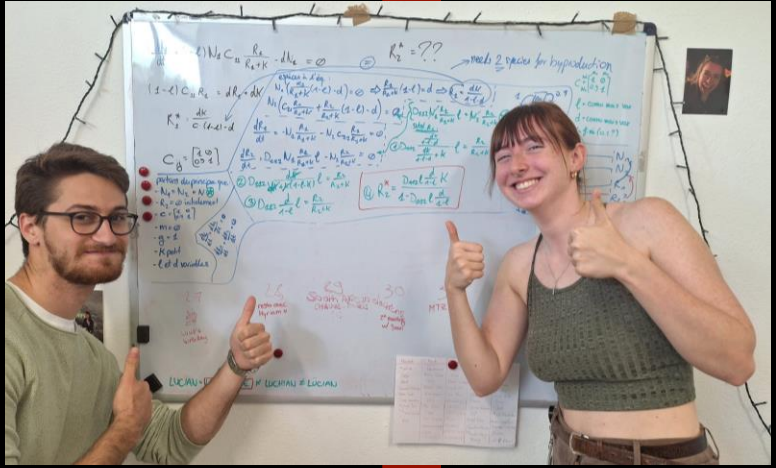
Simulation of the previous slide



$$J^{in}_{i\alpha} = C_{i\alpha}\frac{R_\alpha}{R_\alpha+K_{i\alpha}}$$

$$\frac{dN_i}{dt} = g_i N_i\left[\sum_\alpha w_\alpha (1-l_\alpha)J^{in}_{i\alpha} - m_i\right] - N_i d$$

$$\frac{dR_\alpha}{dt} = (R^0_\alpha - R_\alpha)d - \sum_i N_i J^{in}_{i\alpha} + \sum_{i,\beta} D_{i\alpha\beta}\frac{w_\beta}{w_\alpha} l_\beta N_i J^{in}_{i\beta}$$



At equilibrium :

$$N_1^* = \frac{d(R_1^0 - R_1^*) - N_2^* J_{21}^*}{l * J_{21}^* J_{22}^*}$$

$$R_1^* = \frac{dK}{(1-l) - d}$$

$$N_2^* = \frac{d(l\,(R_1^0 - R_1^*) - R_2^*}{J_{11}^*}$$

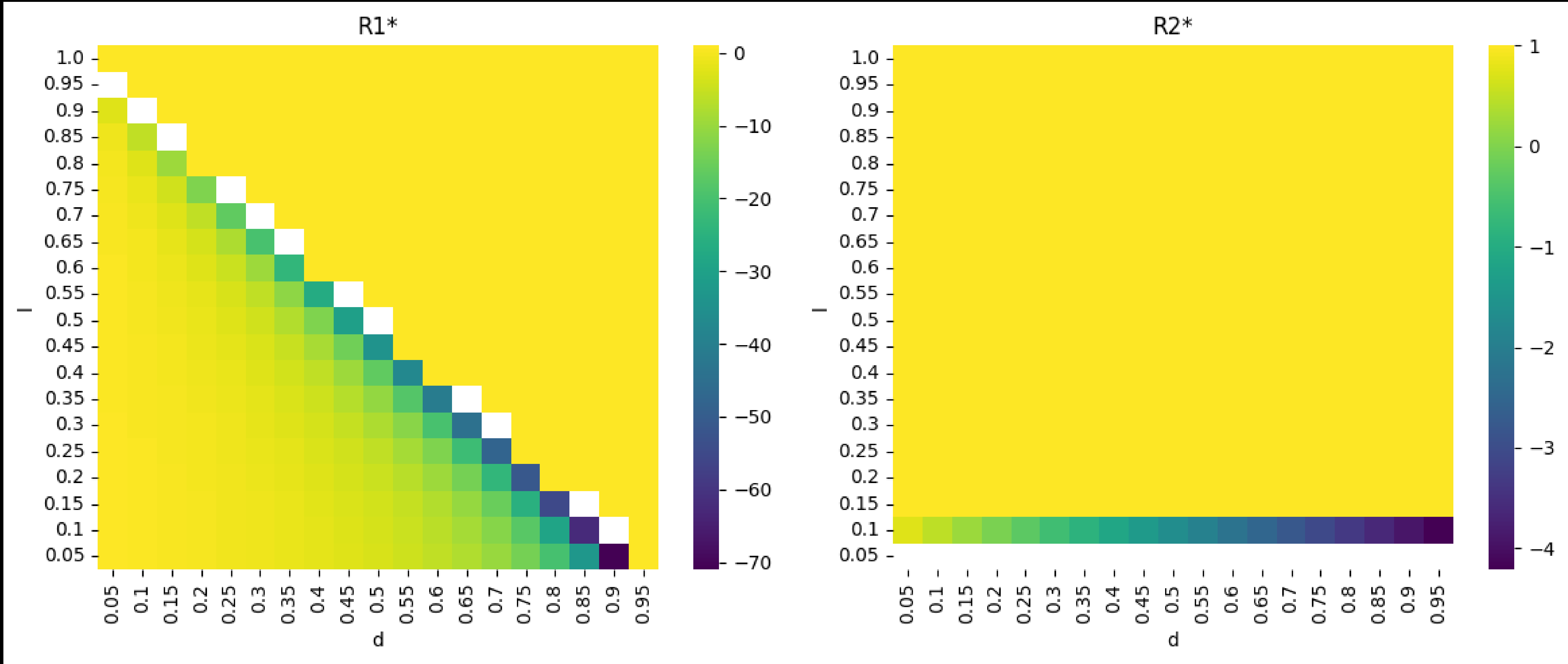
$$R_2^* = \frac{K(d - J_{21})}{J_{21} - d}$$

(Thank you Snorre !!!)

d : dilution rate (decay)
l : leakage

Does the math match the simulations ?

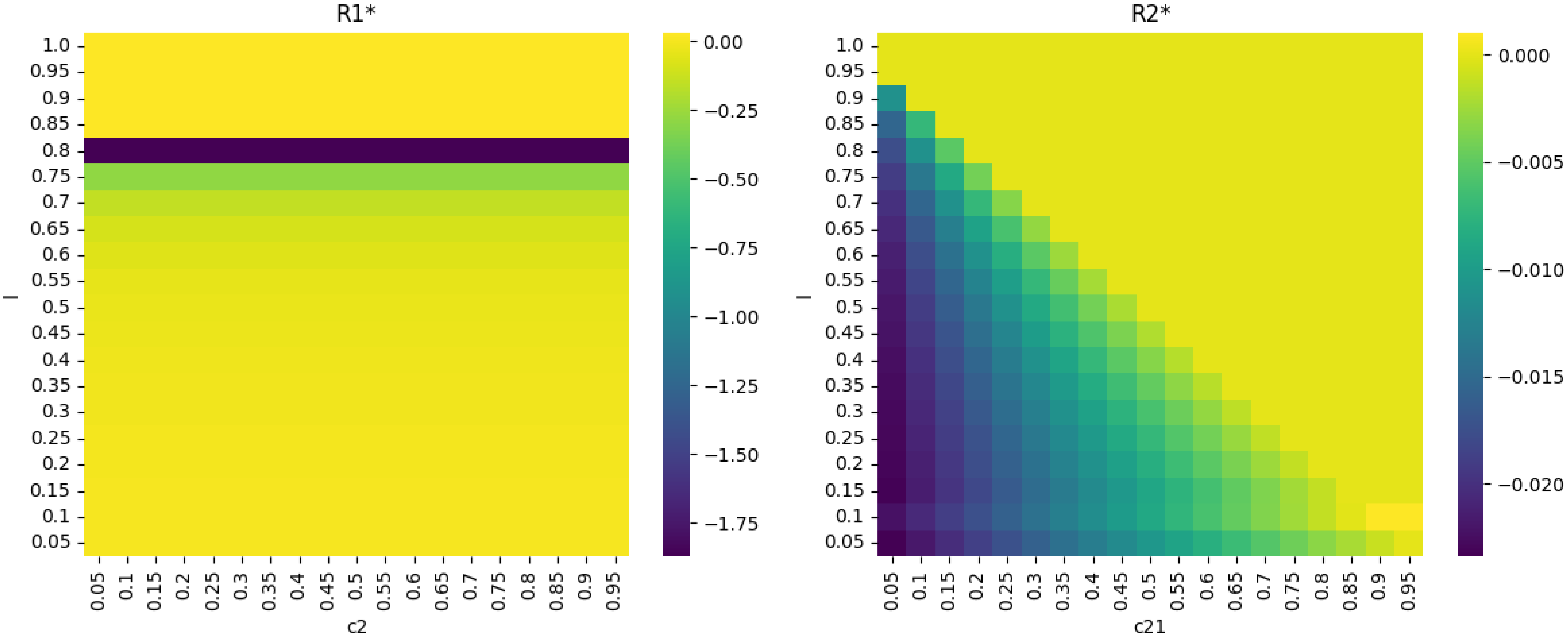
Normalized difference between the theory and the simulations final resources, when varying d and l :



$c:$ *maximum uptake of resource*
$l:$ *leakage*

Does the math match the simulations ? part 2

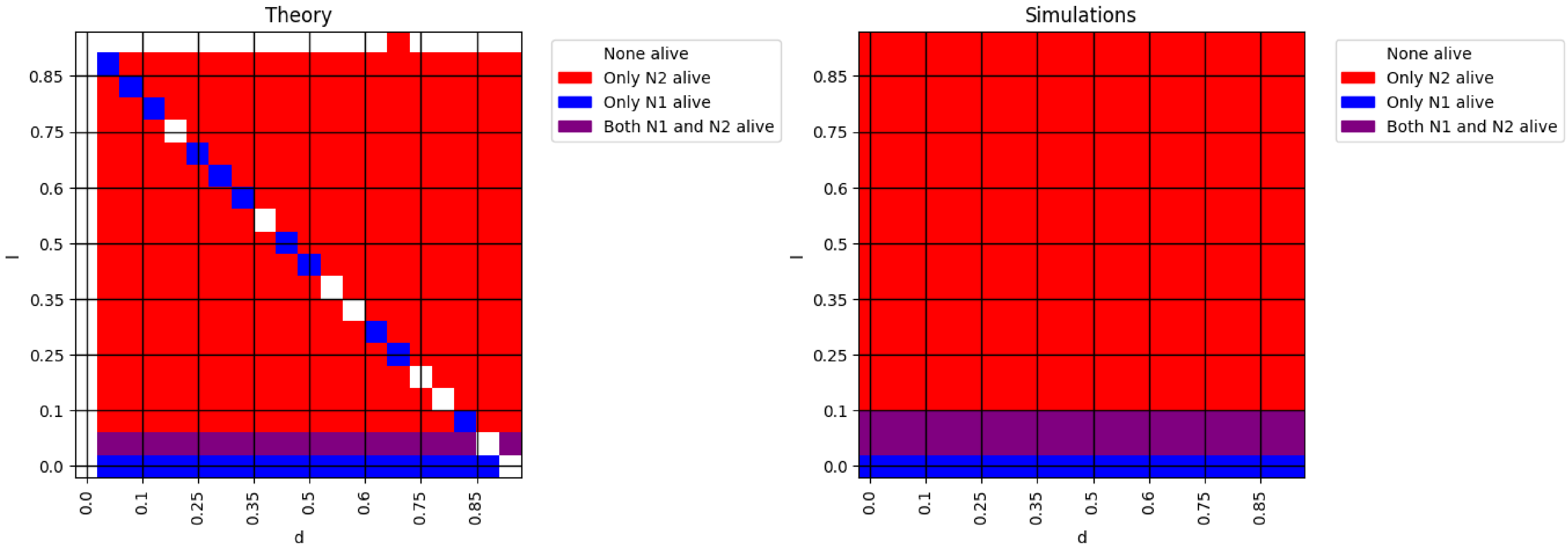
Normalized difference between the theory and the simulations final resources, when varying c_{21} and l :



d : *dilution rate (decay)*
l : *leakage*

Influence of d and l on population stability

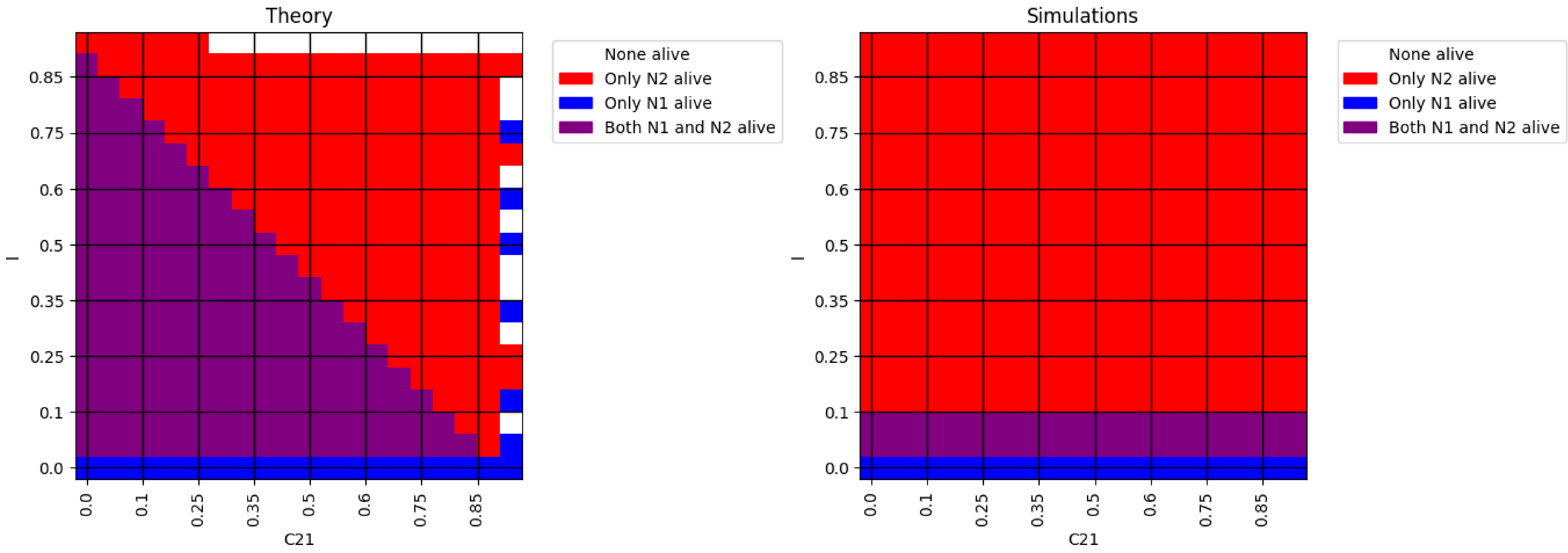
Population outcome by varying d and l



c_{21} : *maximum uptake of resource 1 by species 2*
l : *leakage*

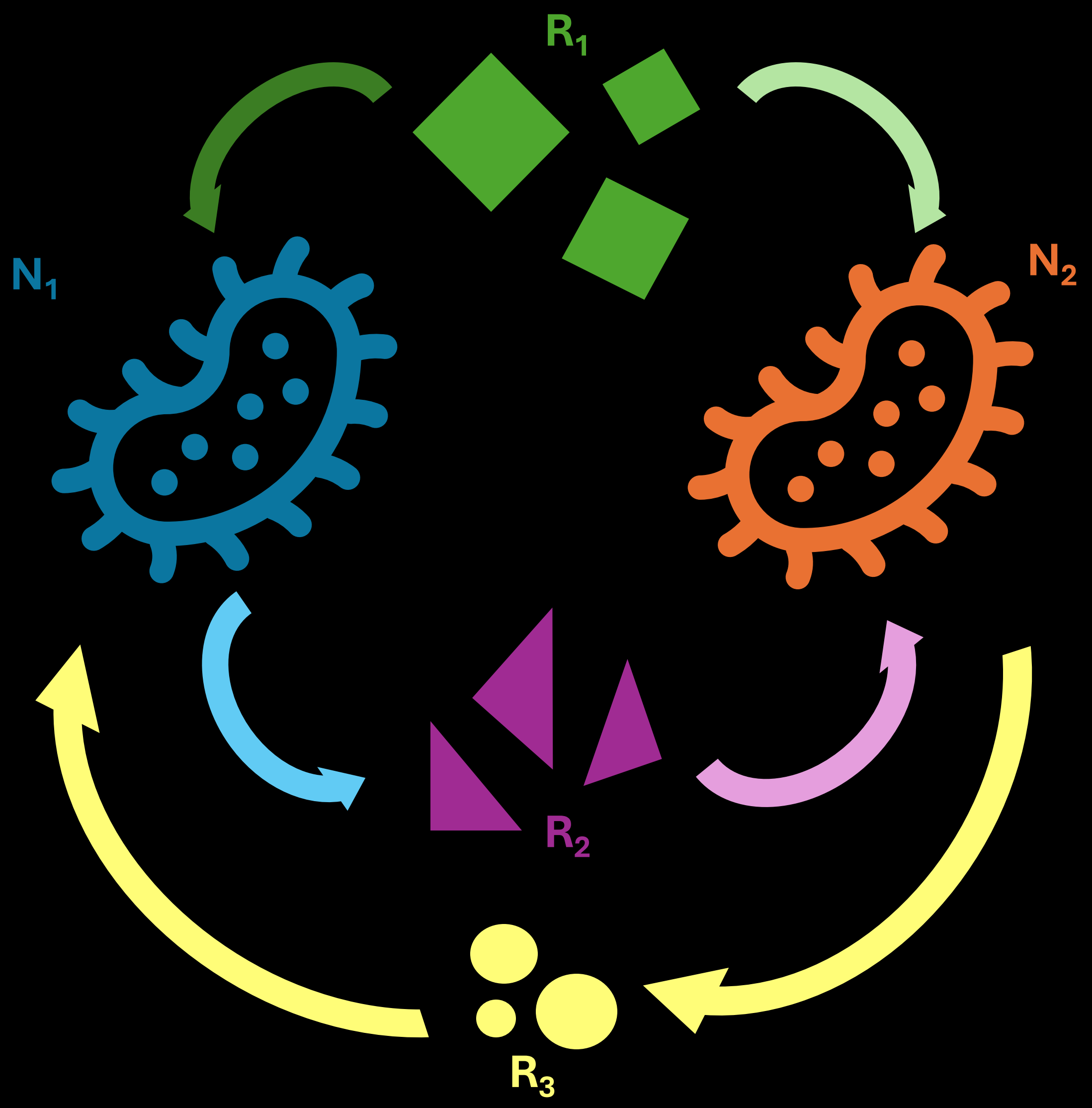
Influence of c_{21} and l on population stability

Population outcome by varying c_{21} and l



Future investigations

- Control the simulation code
- "Double crossfeeding" : species 2 leaks a 3rd resource that can be used by species 1
  - Requires a lot of math !!
- Different energy contents for R_a



Feedback on the course

- Lots of maths !!!
- Coding as a group is challenging but interesting
- “Learn by doing” : overwhelming but rewarding
- We did a lot of work that was not useful in the end, but it made us progress every time
- We liked the freedom that we had for the project

Thank you for listening !

And thank you Snorre <3