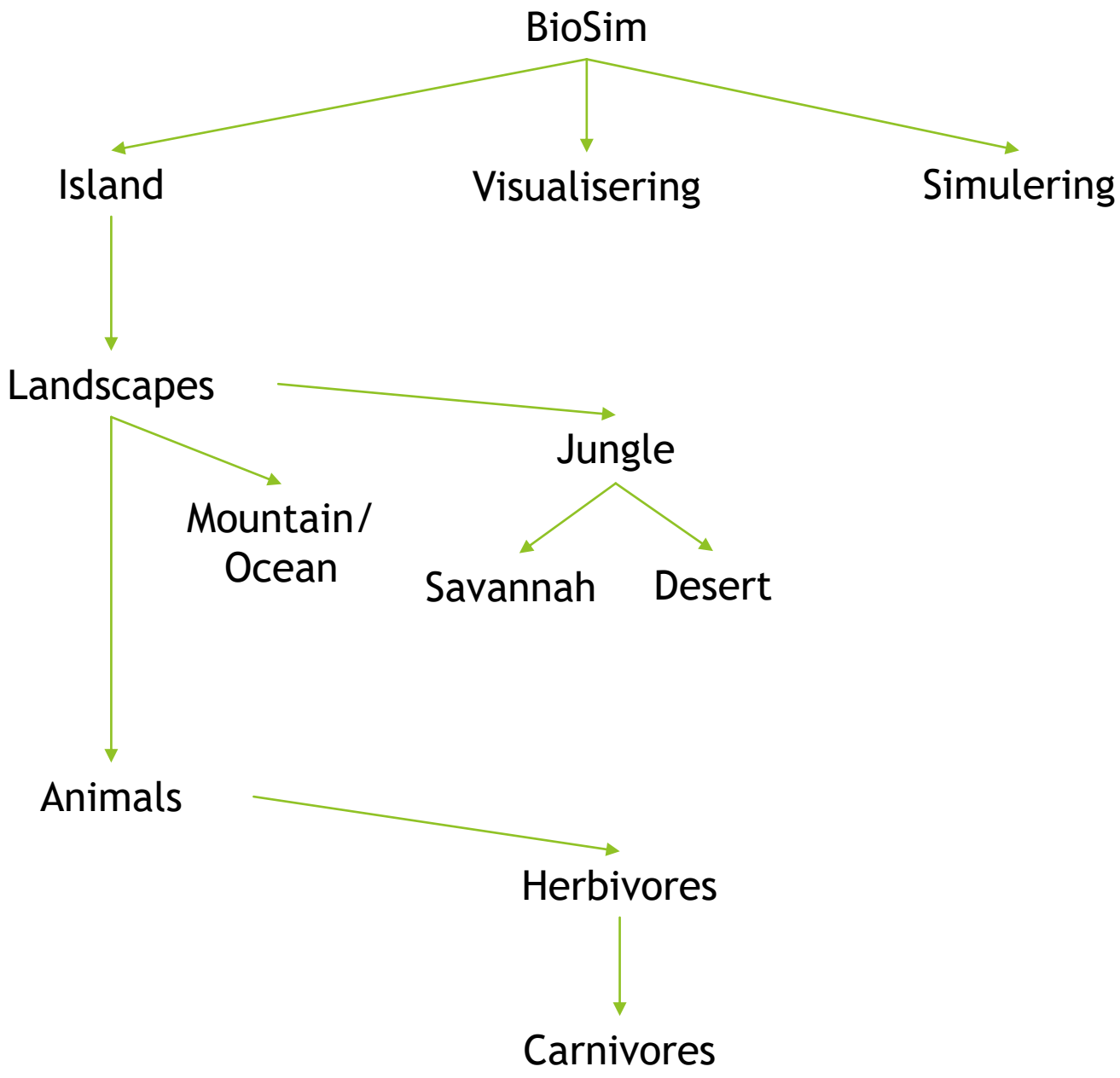
Simulering av Rossumøya

Jon-Fredrik Cappelen og Lars Martin Lied


BioSim
Island
Visualisering
Simulering
Landscapes
Jungle
Mountain/
Ocean
Savannah
Desert
Animals
Herbivores
Carnivores

Name	Call Count	Time (ms) ▼		Own Time (ms)	
check_sim.py	1	72851	100,0%	0	0,0%
simulate	2	71307	97,9%	2	0,0%
cycle	201	71296	97,9%	266	0,4%
migration	201	31469	43,2%	109	0,1%
feeding	29346	19106	26,2%	1141	1,6%
cell_move_herbivores	31557	17449	24,0%	2785	3,8%
cell_move_carnivores	31557	13785	18,9%	885	1,2%
get_direction	382041	11701	16,1%	634	0,9%

- Total: ca 73 sekunder
- migration
- feeding

Name	Call Count	Time (ms)	Own Time (ms)
<method 'choice' of 'mtrand.RandomState' objec	382041	10876 14,9%	10263 14,1%
feeding	368809	11408 15,7%	8785 12,1%
update_fitness	6520777	10030 13,8%	8064 11,1%
get_abundance_carnivore	391866	8010 11,0%	5609 7,7%
get_weight	31782091	3784 5,2%	3784 5,2%
cell_move_herbivores	31557	17449 24,0%	2785 3,8%

Egentid:

- np.random.choice() – treig, annen input?
- feeding: carnivores, mye som skal gjøres
- update_fitness: jit og lazy evaluation

Ser du forskjellen?

```
appetite = copy.deepcopy(self.parameters['F'])
eaten_bool = [True] * len(preys)
for index, prey in enumerate(preys):
    if appetite > prey.get_weight():
        if self.fitness <= prey.fitness:
            continue
        elif 0 < self.fitness - prey.fitness < \
                self.parameters['DeltaPhiMax']:

            probability = (self.fitness - prey.fitness) /\
                          self.parameters['DeltaPhiMax']
            if random.random() < probability:
                eaten_bool[index] = False
                self.weight += self.parameters['beta'] * prey.weight
                appetite -= prey.weight
                self.update_fitness()
        else:
            eaten_bool[index] = False
            self.weight += self.parameters['beta'] * prey.weight
            appetite -= prey.weight
            self.update_fitness()
```

Ser du forskjellen?

```
appetite = copy.deepcopy(self.parameters['F'])
eaten_bool = [True] * len(preys)
for index, prey in enumerate(preys):
    if appetite >= prey.get_weight():
        if self.fitness <= prey.fitness:
            continue
        elif 0 < self.fitness - prey.fitness < \
                self.parameters['DeltaPhiMax']:

            probability = (self.fitness - prey.fitness) /\
                          self.parameters['DeltaPhiMax']
            if random.random() < probability:
                eaten_bool[index] = False
                self.weight += self.parameters['beta'] * prey.weight
                appetite -= prey.weight
                self.update_fitness()
        else:
            eaten_bool[index] = False
            self.weight += self.parameters['beta'] * prey.weight
            appetite -= prey.weight
            self.update_fitness()
```

```python
def test_feeding_carnivores(self):
    """
    Test that all carnivores in the cell feeds: the method feeding

    Does this by manipulating parameter DeltaPhiMax and the food.

    Returns
    -------

    """
    Carnivore.set_parameters({'DeltaPhiMax': 1.000001})
    Jungle.set_parameters({'f_max': 0.0})
    j1 = Jungle()
    j1.herbivores = [Herbivore(1, 40), Herbivore(2, 50), Herbivore(3, 60)]
    j1.carnivores = [Carnivore(1, 20), Carnivore(2, 20), Carnivore(3, 20)]
    for i in range(3):
        j1.herbivores[i].fitness = 0
        j1.carnivores[i].fitness = 1

    j1.feeding()

    Carnivore.set_parameters({'DeltaPhiMax': 10.0})  # default value
    Jungle.set_parameters({'f_max': 800.0})
    assert (j1.carnivores[0].weight, j1.carnivores[1].weight,
            j1.carnivores[2].weight) == (57.5, 57.5, 20)
    assert j1.herbivores == []
```

Coverage Test all

100% files, 78% lines covered in 'biosim'

Element	Statistics, %
.cache	
.idea	
tests	100% files, 100% lines covered
__init__.py	100% lines covered
animals.py	82% lines covered
island.py	79% lines covered
landscape.py	90% lines covered
simulation.py	35% lines covered

- Totalt: 34 tester
- Flere statistiske tester
- simulation.py: mye grafisk

This Page

Show Source

Quick search

Go

Welcome to BioSim G04's documentation!

This file is a simulation

- for herbivores and carnivores
- on an island

Contents:

Indices and tables

- Index
- Module Index
- Search Page

class **biosim.animals.Herbivore**(*age*, *weight*) [source]

Class for herbivores, with all methods and parameters

Constructor-method for making an herbivore of set age and weight.

Parameters:
- **age** (*int*) – Age of herbivore
- **weight** (*float*) – Weight of herbivore

aging() [source]

Method for handling the aging of an animal. Also updates fitness.

death() [source]

Method for for checking if the animal should die from natural causes

Calculates the chance of dying due to low fitness, by using the parameter 'omega'

feeding(*landscape_instance*) [source]

Handles the feeding of the animal

Parameters: **landscape_instance** (*The tile, that the given animal is in*) –

get_weight() [source]

Returns: The weight of the animal

Return type: float

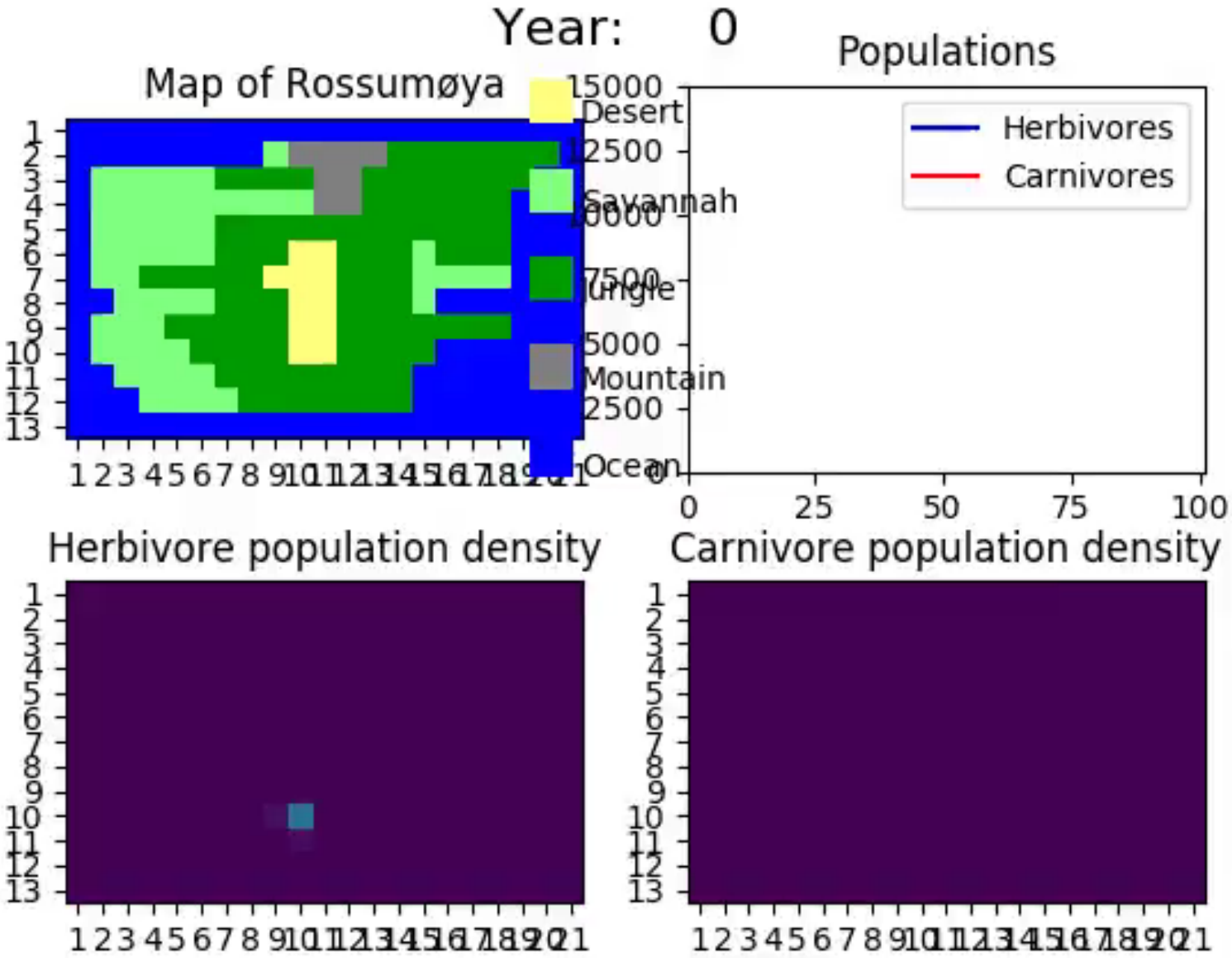
loss_of_weight() [source]

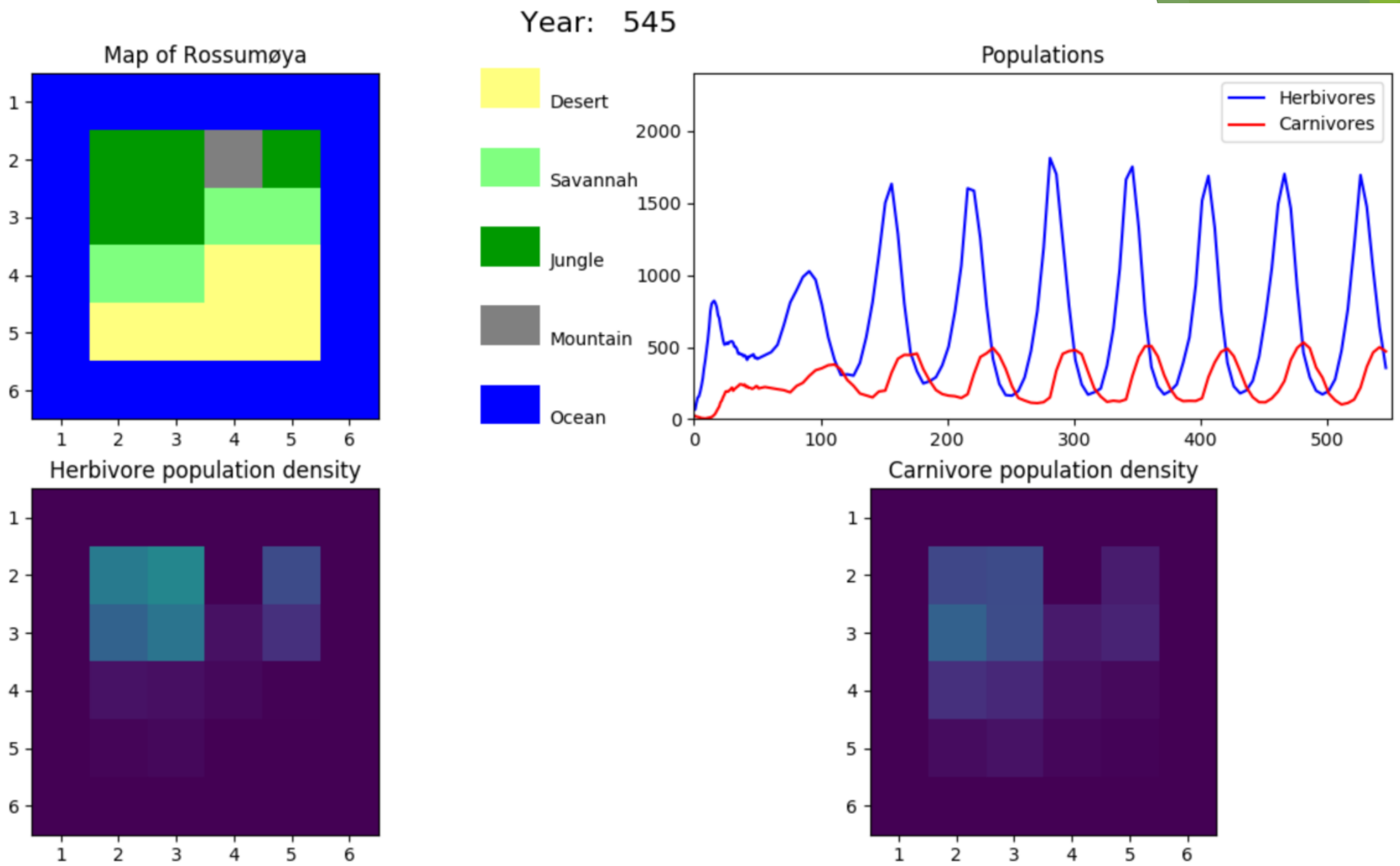
Method for handling the loss of weight, by natural causes

A method for decreasing the weight of animal every year by a parameter 'eta' multiplied by the animals own weight. Also updates fitness.

migration() [source]

Method for checking if the animal will migrate


Year: 0
Map of Rossumøya
Desert
Savannah
Jungle
Mountain
Ocean
Populations
Herbivores
Carnivores
Herbivore population density
Carnivore population density


Year: 545
Map of Rossumøya
Desert
Savannah
Jungle
Mountain
Ocean
Populations
Herbivores
Carnivores
Herbivore population density
Carnivore population density