Modelling the Ecosystem of Rossumøya

Hans Ekkehard Plesser, NMBU

January 5, 2023

1 The Nature of Rossumøya

1.1 Geography

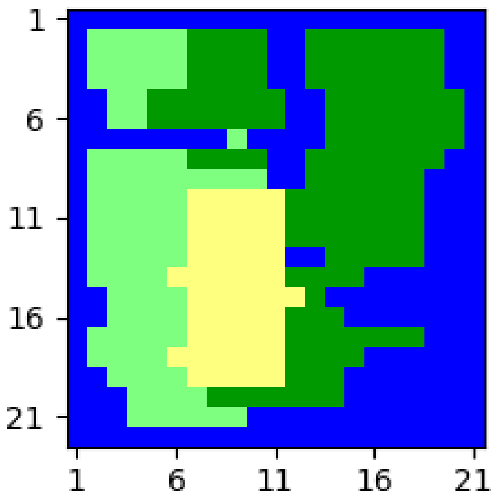

Figure 1: Landscape types on Rossumøya according to the last survey. Blue: Water, dark green: lowland, light green: highland, yellow: desert.

Rossumøya is divided into squares (or cells), where each square is of one of the following five types:

- water,
- desert,
- highland,
- lowland.

1.1.1 Water

Water cannot be entered by animals. Cells of type will be completely passive in the simulation. Water may be sea surrounding the island or lakes within the island.

1.1.2 Desert

Animals may stay in the desert, but there is no fodder available to herbivores there. Carnivores can prey on herbivores in the desert.

1.1.3 Highland

Herbivores will find fodder in the highland. Each year, a fixed amount of fodder is available (see also Sec. 1.3):

$$f_{ij} \leftarrow f_{\text{max}}^{\text{Highland}} . \tag{1}$$

Carnivores can prey on herbivores in the highland.

1.1.4 Lowland

The same rules apply as to the highland, but usually more fodder is available in the lowland than the highland, i.e.,

$$f_{\text{max}}^{\text{Lowland}} > f_{\text{max}}^{\text{Highland}} . \tag{2}$$

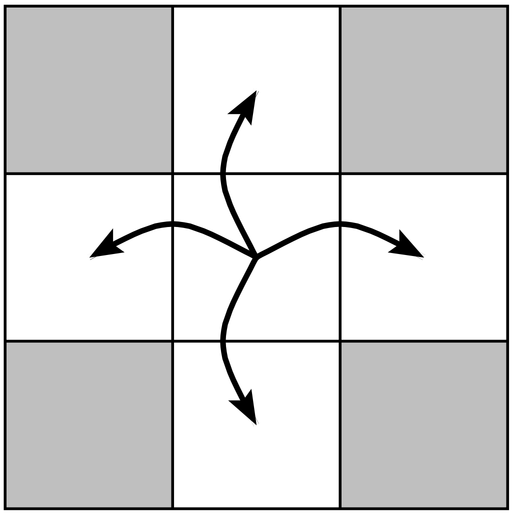

Figure 2: An animal that is in the middle cell can move to one of the four neighboring cells but not to cells diagonally displaced (gray).

1.2 Fauna

Herbivores and carnivores have certain characteristics in common, but feed in different ways, see section 1.2.1 and 1.2.2. The similarities are as follows[1]:

1. Age At birth, each animal has age $a = 0$. Age increases by one year for each year that passes.

2. Weight The weights of newborn animals is distributed according to a log-normal distribution[2] with mean w_{birth} and standard deviation σ_{birth}. When an animal eats an amount F of fodder, its weight increases by βF. Every year, the weight of the animal decreases by ηw.

3. Fitness The overall condition of the animal is described by its fitness, which is calculated based on age and weight using the following formula

$$\Phi = \begin{cases} 0 & w \leq 0 \\ q^{+}(a, a_{\frac{1}{2}}, \phi_{\text{age}}) \times q^{-}(w, w_{\frac{1}{2}}, \phi_{\text{weight}}) & \text{else} \end{cases} \tag{3}$$

where

$$q^{\pm}(x, x_{\frac{1}{2}}, \phi) = \frac{1}{1 + e^{\pm\phi(x - x_{\frac{1}{2}})}} . \tag{4}$$

Note that $0 \leq \Phi \leq 1$.

4. Migration Animals migrate depending on their own fitness. Animals can only move to the four immediately adjacent cells. Animals cannot move to water cells.

An animal moves with probability $\mu\Phi$.

If an animal moves, the destination cell is chosen at random between the four nearest neighbor cells, with equal probability for each direction, see Fig. 2. If the selected destination cell is Water, the animal does not move.

An animal can move only once per year.

[1]See section 1.3 for details on how the various processes are distributed throughout the year.

[2]`https://en.wikipedia.org/wiki/Log-normal_distribution`

5. Birth Animals can produce offspring according to the following rules:

(a) An animal can only have offspring if its weight is $w \geq \zeta(w_{\text{birth}} + \sigma_{\text{birth}})$.

(b) For each animal in a cell, the probability to produce offspring in a year is

$$\min\left(1, \gamma \times \Phi \times N\right) , \tag{5}$$

where N is the number of animals of the same species in the cell at the start of the breeding season.

(c) Each animal can give birth to at most one offspring per year.

(d) At birth, the parent loses ξ times the actual birthweight of the baby.

(e) If the parent would lose more than their own weight, then no baby is born and the weight of the parent remains unchanged.

6. Death An animal dies

- with certainty if its weight is $w = 0$;
- with probability

$$\omega(1 - \Phi) \tag{6}$$

otherwise.

1.2.1 Herbivores

Herbivores find fodder exclusively in the low- and highland. Animals residing in a cell eat in random order. Each animal tries every year to eat an amount F of fodder, but how much feed the animal obtain depends on fodder available in the cell, see section 1.1. Given that the animal eats an amount $\tilde{F}$ of fodder, its weight increases by $\beta\tilde{F}$.

1.2.2 Carnivores

Carnivores can prey on herbivores everywhere, but do not prey on each other. One carnivore hunts at a time, in order order of descending fitness. When hunting, each carnivore tries to kill one herbivore at a time, beginning with the herbivore with the lowest fitness. A carnivore continues to kill herbivores until

- the carnivore has eaten an amount F, i.e., eaten herbivores with a total weight $\geq F$
- or has tried to kill each herbivore in the cell.

Carnivores will kill a herbivore with probability

$$p = \begin{cases} 0 & \text{if } \Phi_{\text{carn}} \leq \Phi_{\text{herb}} \\ \frac{\Phi_{\text{carn}} - \Phi_{\text{herb}}}{\Delta\Phi_{\text{max}}} & \text{if } 0 < \Phi_{\text{carn}} - \Phi_{\text{herb}} < \Delta\Phi_{\text{max}} \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

The carnivore's weight increases by βw_{herb}, where w_{herb} is the weight of the herbivore killed[3]. The fitness of the carnivore is re-evaluated each time it kills a herbivore.

Param.	Herb.	Carn.	Name
w_{birth}	8.0	6.0	`w_birth`
σ_{birth}	1.5	1.0	`sigma_birth`
β	0.9	0.75	`beta`
η	0.05	0.125	`eta`
$a_{\frac{1}{2}}$	40.0	40.0	`a_half`
ϕ_{age}	0.6	0.3	`phi_age`
$w_{\frac{1}{2}}$	10.0	4.0	`w_half`
ϕ_{weight}	0.1	0.4	`phi_weight`
μ	0.25	0.4	`mu`
γ	0.2	0.8	`gamma`
ζ	3.5	3.5	`zeta`
ξ	1.2	1.1	`xi`
ω	0.4	0.8	`omega`
F	10.0	50.0	`F`
$\Delta\Phi_{\mathrm{max}}$	—	10.0	`DeltaPhiMax`

Table 1: Example values for parameters of herbivores and carnivores. These values are used in the examples shown in the text, but have no special meaning. All parameter values shall be positive (≥ 0), except for $\Delta\Phi_{\mathrm{max}}$, which shall be strictly positive (> 0). Furthemore, $\eta \leq 1$ is required.

Param.	Lowland	Highland	Name
f_{max}	800.0	300.0	`f_max`

Table 2: Example values for the parameter of Lowland and Highland cells. These values are used in the examples shown in the text, but have no special meaning.

1.2.3 Parameters

The parameters w_{birth}, σ_{birth}, β, η, $a_{\frac{1}{2}}$, $w_{\frac{1}{2}}$, ϕ_{age}, ϕ_{weight}, μ, γ, ζ, ξ, ω, F, and $\Delta\Phi_{\mathrm{max}}$ are identical for all animals of the same species, but may be different between herbivores and carnivores. Example values are given in Table 1 and 2.

[3]If the weight of the herbivore exceeds the amount of food desired by the carnivore, the carnivore eats only the amount it wants. The remainder of the herbivore goes to waste.

1.3 The Annual Cycle on Rossumøya

Nature on Rossumøya follows a fixed annual cycle. The components of annual cycle are:

1. Procreation Animals give birth, see section 1.2, No. 5. When calculating the probability of birth according to equation (5), the number of animals N at the start of the breeding season is used, i.e., newborn animals do not count.

2. Feeding Animals eat: first herbivores, then carnivores, see section 1.2.1 and 1.2.2. Growth of fodder in lowland and highland occurs at the very beginning of the year, i.e., immediately before herbivores eat.

3. Migration Animals migrate to neighboring cells subject to the condition that each animal can migrate at most once per year.

4. Aging Each animal becomes one year older.

5. Loss of weight All animals lose weight, see section 1.2, No. 2.

6. Death For each animal, it is determined whether the animal dies or not, see section 1.2, No. 6.

Steps 1–6 can be seen as the seasons on Rossumøya, i.e., all animals undergo steps simultaneously.

1.4 Deliverables

Software A Python package

1. compatible with Python 3.10 or later;
2. installable using standard Python distribution tools;
3. structured according to the supplied project template;
4. organized so that a user can carry out simulations using the methods of a class `BioSim`, which forms the interface for the package, see also Appendix **??**;
5. written in well-structured, documented, and efficient code following PEP8 guidelines (maximum line length 100 characters);
6. including unit and integration tests covering the code (PyTest framework);
7. including user-level documentation of all modules, classes and public methods generated with Sphinx from docstrings, allowing domain experts to use the software;
8. including working examples;
9. passing a standard test of interface tests provided by EPAP and working with the compatibility check script given in Appendix A.

Presentation Software and exemplary results will be presented in virtual oral presentations. The presentation shall discuss the main aspects of the chosen implementation, its advantages and disadvantages.

`W`	water
`L`	lowland
`H`	highland
`D`	desert

Table 3: Codes for landscape types.

1.5 Parameters and Initialization

1.5.1 Geography

The software shall be able to read a Python multi-line string specification of the island's geography. All lines in the string must have the same length. Each character in the string represents a cell with character code shown in Table 3.

The geography string must consist solely of "W" around the edges and no characters other than the letters shown in Table 3 must occur in the string. The software will raise a `ValueError` if any requirements on the geography specification is violated.

The coordinate system for the island is as illustrated in Figure 1:

- The upper left corner has coordinates $(1, 1)$.
- Coordinates increase downward and to the right.
- The first coordinate enumerates the rows, the second the columns.

1.5.2 Parameters

It shall be possible to specify the parameters of the animal species and landscape types, respectively, by providing a dictionary of proper contents to a suitable method in the package. Four dictionaries are required to provide a complete parameterization. Furthermore, one method setting the parameters is required for each class (herbivores, carnivores, lowland and highland). Additionally, the following guidelines apply to parameters:

1. The parameter names given in Table 1 and 2 shall be used.
2. The software shall have built-in default values for all parameters, so that simulations can be carried out without setting parameters.
3. It shall be possible to change a subset of parameters by providing a dictionary with only those parameters that are to be changed to the method carrying out the parameterization.
4. The parameterisation methods shall report an error if a dictionary contains unknown parameters.
5. The parameterization method shall guard against illegal parameter values, such as negative amounts of fodder.
6. Any errors detected in parameter dictionaries shall raise a `ValueError`.

1.5.3 Populations

It shall be possible to place animals on the island before the simulation starts and during breaks in the simulation. Placement will take place by passing a list[4] to a suitable method in the interface class. The list shall have the following format:

1. Each item in the list is a *dictionary* with two elements, `'loc'` (Location) and `'pop'` (Population).
2. `'loc'` is a tuple with two elements and provides a coordinate on the island, see section 1.5.1. It is an error to specify nonexistent coordinates.
3. `'pop'` is a list with one element per animal.
4. Each item in `'pop'` is a dictionary with elements `'species'`, `'age'` and `'weight'`.
5. The `'species'` element has either the value `'Herbivore'` or `'Carnivore'`.
6. `'age'` shall be a non-negative integer.
7. `'weight'` shall be a positive number.

The list could, for example, look like this:

```
[{'loc': (3,4),
  'pop': [{'species': 'Herbivore',
           'age': 10, 'weight': 12.5},
          {'species': 'Herbivore',
           'age':  9, 'weight': 10.3},
          {'species': 'Carnivore',
           'age':  5, 'weight':  8.1}]},
 {'loc': (4,4),
  'pop': [{'species': 'Herbivore',
           'age': 10, 'weight': 12.5},
          {'species': 'Carnivore',
           'age':  3, 'weight':  7.3},
          {'species': 'Carnivore',
           'age':  5, 'weight':  8.1}]}]
```

For each list item, animals given in `'pop'` will be placed in the cell specified by `'loc'`. The program shall ensure that

- all animals have positive weight and non-negative age, and
- animals are only placed in cells where animals can stay;

If a placement violates these conditions, the program shall raise a `ValueError`.

If there are animals in a cell from before, then these will remain in the cell.

1.5.4 Random Numbers

Seed values for the random number generator must be set before the simulation starts. Running a simulation twice with the same random seed shall yield identical results.

Use the same random number generator everywhere random numbers are used in the simulation. You shall use the plain Python `random` module.

[4] More precisely, an *iterable*.

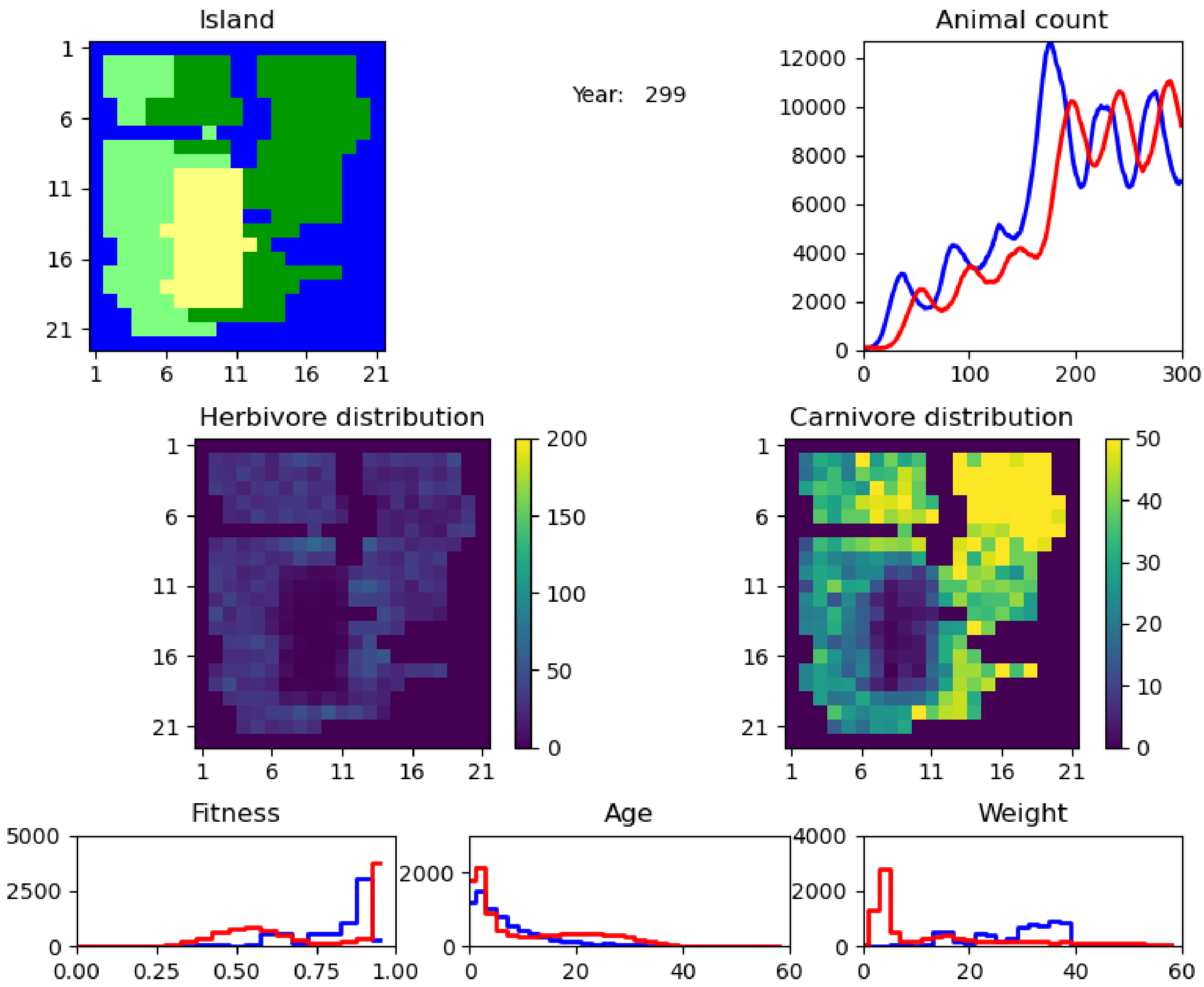

Figure 3: Example of a simulation visualisation. The figure is only meant as an illustration. Teams are encouraged to improve on the graphics.

.

1.6 Simulation and Recording

1.6.1 Simulation

The simulation will run for a given number of years. After that, it shall be possible to investigate the island's status, change parameters, set out more animals, and resume the simulation for further years. One should, e.g., be able to simulate the first 100 years with only herbivores on the island, before a small group of carnivores is added and the simulation continued.

1.6.2 Status Information

When the simulation is stopped, it shall be possible to obtain the following information through Python properties of the `BioSim` class with prescribed names:

Year Number of years that have been simulated (property: `year`).

Total number of animals The total number of animals on the island (property: `num_animals`).

Total number of animals by species The total number of herbivores and carnivores, respectively, in a dictionary with keys `'Herbivore'` and `'Carnivore'` (property: `num_animals_per_speci`

1.7 Visualization

The user shall be able to visualize the simulation results while the simulation is in progress, as exemplified in Fig. 3. The visualization shall be in one graphics window with the following elements:

Geography The island's geography is shown, with a color code for the landscape types.

Total number of animals by species is shown as line graph, with one line for each species.

Population map For each species a map shall be provided showing the number of animals per cell using a color code, including color bars.

Histograms shall show the distribution of animal ages, weights and fitnesses for herbivores and carnivores.

Year The year shall appear in the graphics window.

The user shall be able to specify

- after how many years the graphics are to be updated (the default is each year; setting this to zero shall disable graphics);
- the limits for the vertical axis in the line graph of animal numbers;
- the limits for the color code in the population maps, separate for both species;
- the upper limit and bin width of the histogram for each parameter (age, weight, fitness);
- that the graphic is saved as file, and in this case also
  - directory in which images are to be stored; if not given, no images will be stored;
  - beginning of file name under which images are to be stored;
  - the image file name suffix, which will determine the image format (e.g., `pdf`, `png`);
  - after how many years an image is to be saved (default: save on every visualisation update; value must be multiple of interval for updating graphics).

The files that are stored must be numbered consecutively. To allow the user to convert a series of graphics files into a movie using the encoding program `ffmpeg`, files must be numbered consecutively, eg. `bs_00000.png`, `bs_00001.png`, `bs_00002.png`, ...

More information about creating movies will be provided later.

A Compatibility check

Two compatibility checks are provided by EPAP. The `biosim` package shall pass both:

- `test_biosim_interface.py` contains a set of tests for the interface of the `BioSim` class and is to be used with PyTest.
- `check_sim.py` (see below) is a sample simulation script that shall work with the package you develop. Running the script shall not require any user input.

Both files are available from the project template. The test file shall be included in the `tests` directory in your package, the sample script in the `examples` directory.

No changes shall be made to either file except with express permission of EPAP.