BioSim package

A package for simulating the ecosystem of Rossumøya

Krishna Mohan Shah

Osama Abidi

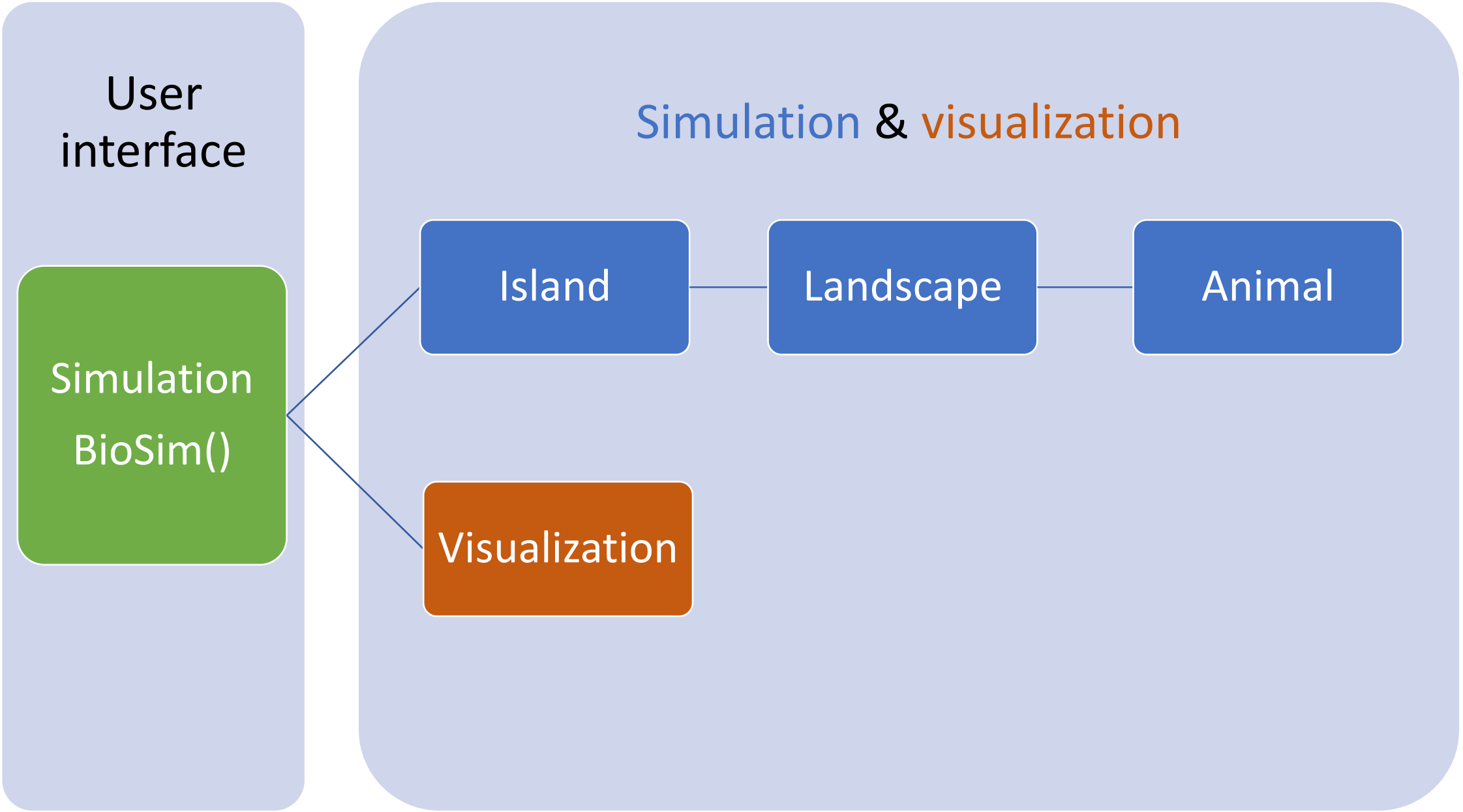
Highlights

- What is the BioSim package?
  - Package overview
- Why should you choose this package?
  - Simulation
  - Visualization

What is the BioSim package?

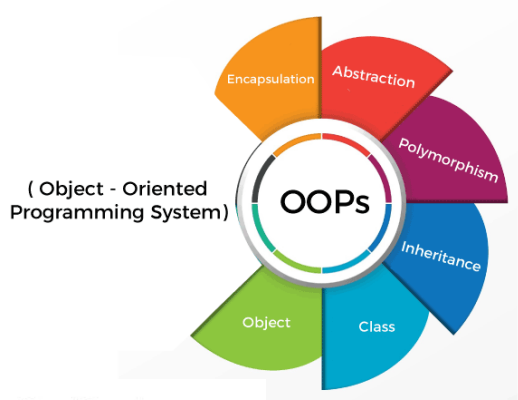
BioSim allows the user to …

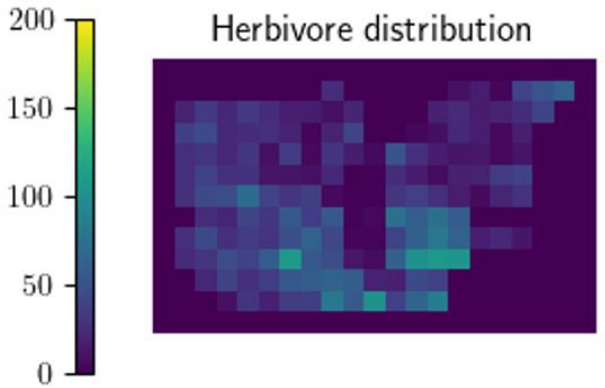
- Simulate island ecosystems of…
  - Various landscapes
  - Herbivores and carnivores
- Simulate
- Define animal and landscape parameters
- Define graph axes
- Create log file of animal counts
- Make movie
- Saving and loading of island states


User interface
Simulation & visualization
Simulation
BioSim()
Island
Landscape
Animal
Visualization

Why should you choose this package?

Simulation

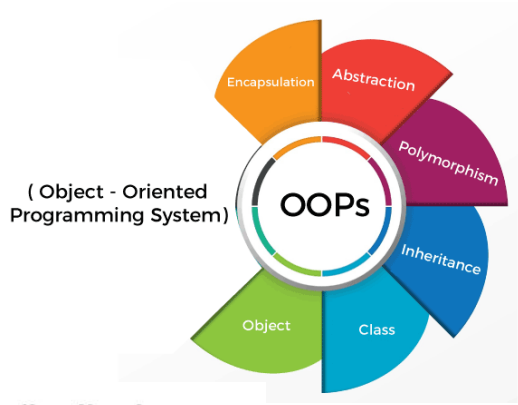

(Object - Oriented
Programming System)
OOPs
Encapsulation
Abstraction
Polymorphism
Inheritance
Class
Object

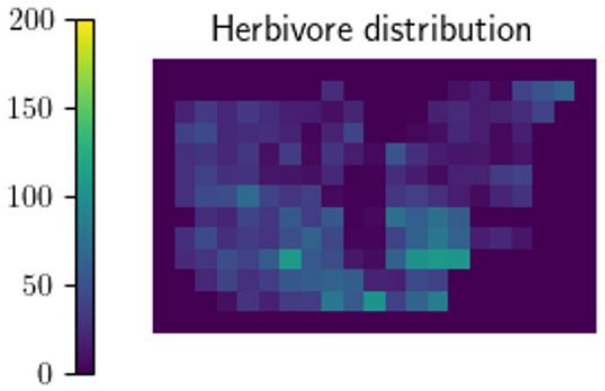

Herbivore distribution
200
150
100
50
0


BioSim
Search docs
CONTENTS:
Introduction
Usage
Simulation and Laboratory
Docs » Welcome to BioSim's documentation
View page source
Welcome to BioSim's documentation
BioSim is a Python package developed for creating and visualizing agent-based simulations of island ecosystems consisting of various landscape and fauna.
The package was originally made to simulate the ecosystem of the fictional island of Rossumoya, yet it allows for the simulation of island ecosystems with different configurations of landscape and fauna through the user interface in the BioSim class.

Simulation

- Utilizes principles of Object-Oriented Programming

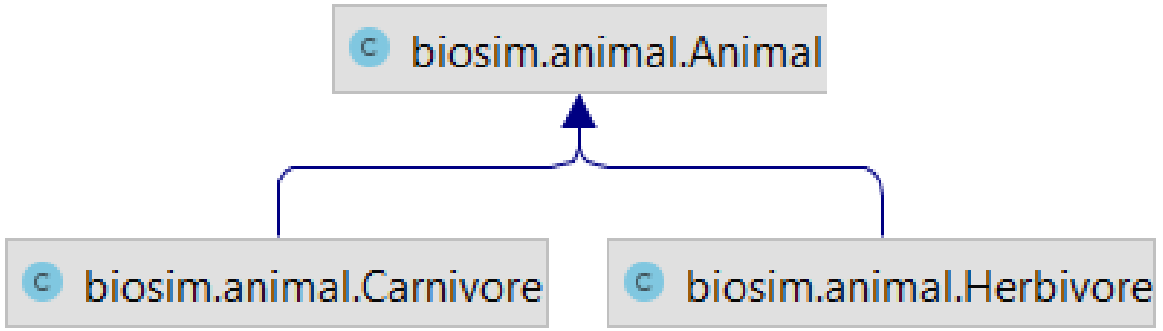




OOP principles

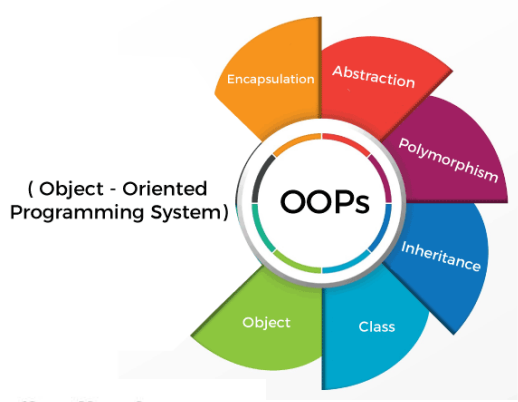
Allows for well-structured and efficient code

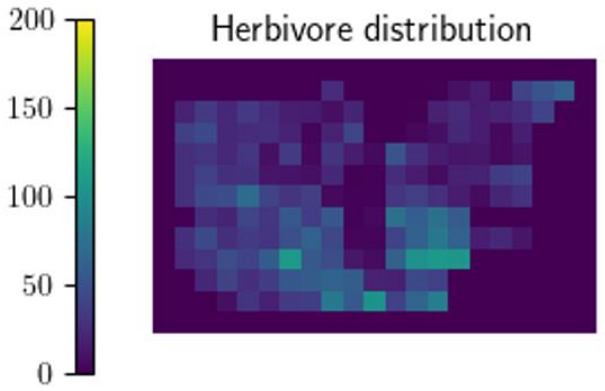
```python
class Animal():
    """
    Animal class with object properties.
    """
    def __init__(self, weight=None, age=0):

        self.initiate_weight(weight)
        self.initiate_age(age)
        self.count_new()

    @classmethod
    def reset_counter(cls):
        cls.count = 0

    @classmethod
    def count_new(cls):
        cls.count += 1

    @classmethod
    def remove_one_count(cls):
        cls.count -= 1
```



Simulation

- Utilizes principles of Object-Oriented Programming
- Thoroughly documented code



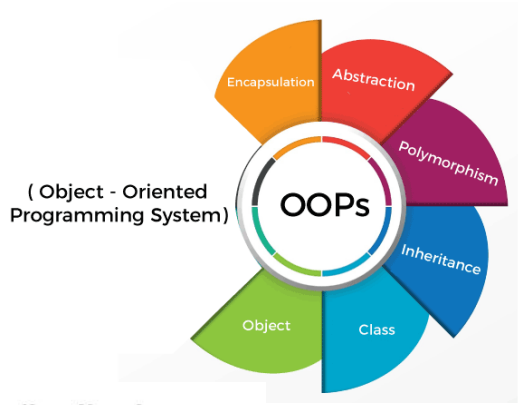


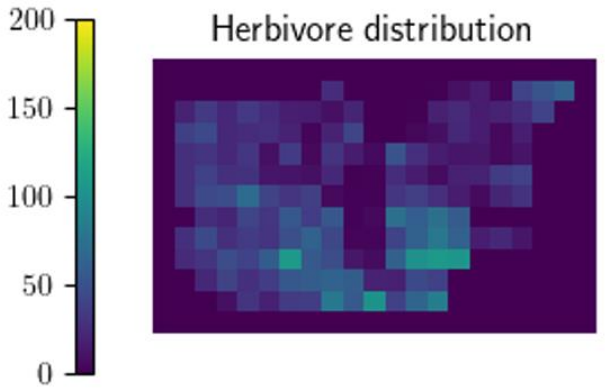
Documentation

Detailed documentation of the package from setup to usage.

Simulation

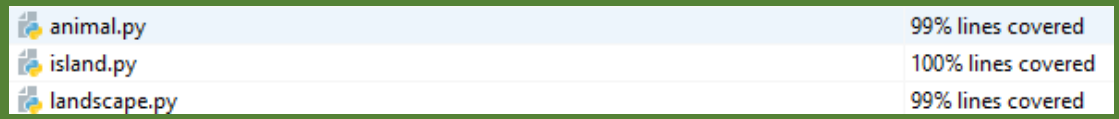
- Utilizes principles of Object-Oriented Programming
- Thoroughly documented code
- Solid testing





Testing

High coverage



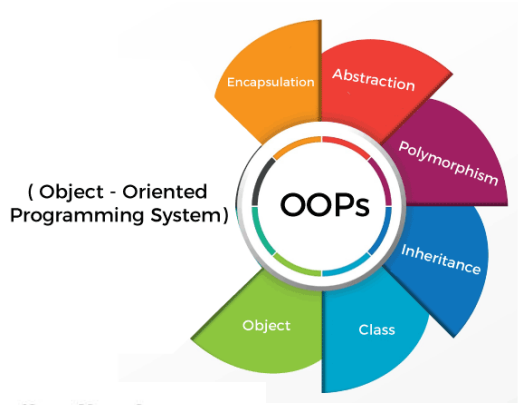
Utilizes both black box and white box testing

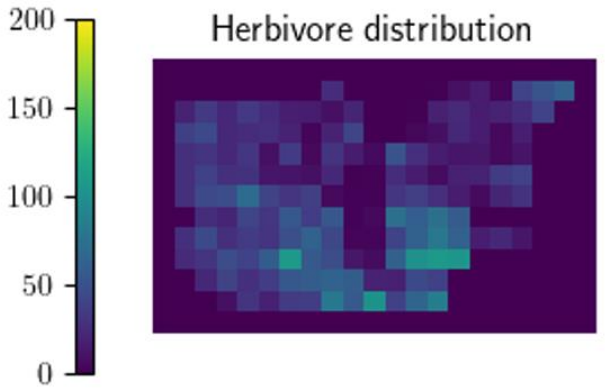
Use of statistical tests

- Chi-square tests
- Z-tests
  - Proportions
  - Truncated normal distributions

Simulation

- Utilizes principles of Object-Oriented Programming
- Thoroughly documented code
- Solid testing
- User friendly

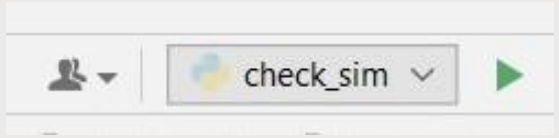




User friendly

- Simple user interface
- Displays error with human readable messages

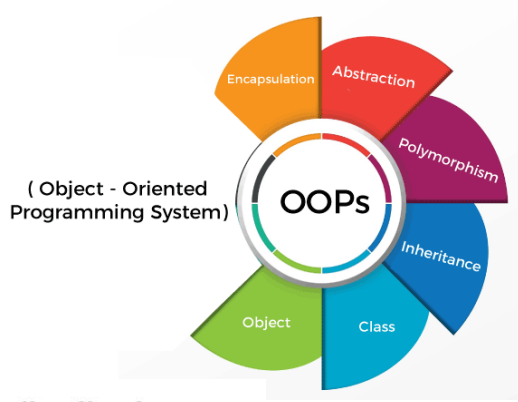
```
class BioSim:
    def __init__(self, island_map, ini_pop, seed,
                 vis_years=1, ymax_animals=None, cmax_animals=None, hist_specs=None,
                 img_dir=None, img_base=None, img_fmt=None, img_years=None,
                 log_file=None):
```

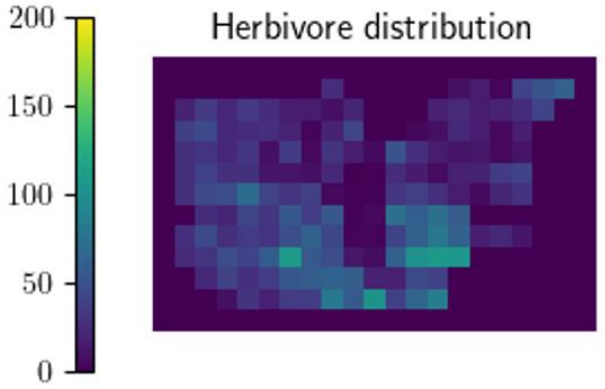



```
    raise ValueError("Island input must have consistent row length")
ValueError: Island input must have consistent row length
```

Simulation

- Utilizes principles of Object-Oriented Programming
- Thoroughly documented code
- Solid testing
- User friendly
- Clever algorithms



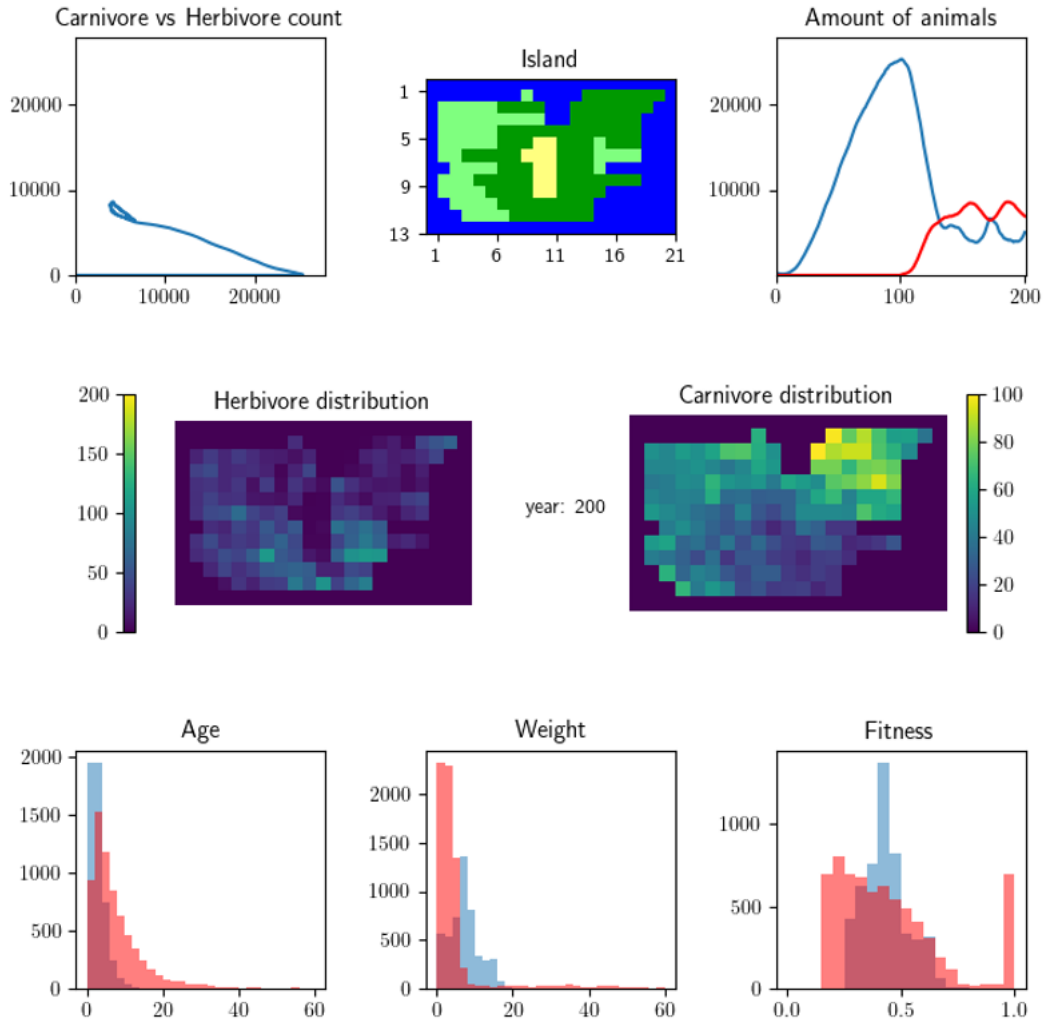


Clever algorithms

```
def migrate_herbivores(self):
    """
    Gives every herbivore in the cell the option of migrating or not.
    """
    if "," in self.herb_list:
        index = self.herb_list.index(",")
    else:
        index = len(self.herb_list)

    for herbivore in self.herb_list[0:index]:
        if herbivore.migrate_or_not():
            direction = herbivore.migration_direction()
            destination = self.adjacent_cells[direction]
            if destination.accessability:
                if 0 < direction < 3 and "," not in destination.herb_list
                    destination.herb_list.append(",")
                destination.herb_list.append(herbivore)
                self.herb_list.remove(herbivore)

    while "," in self.herb_list:
        self.herb_list.remove(",")
```

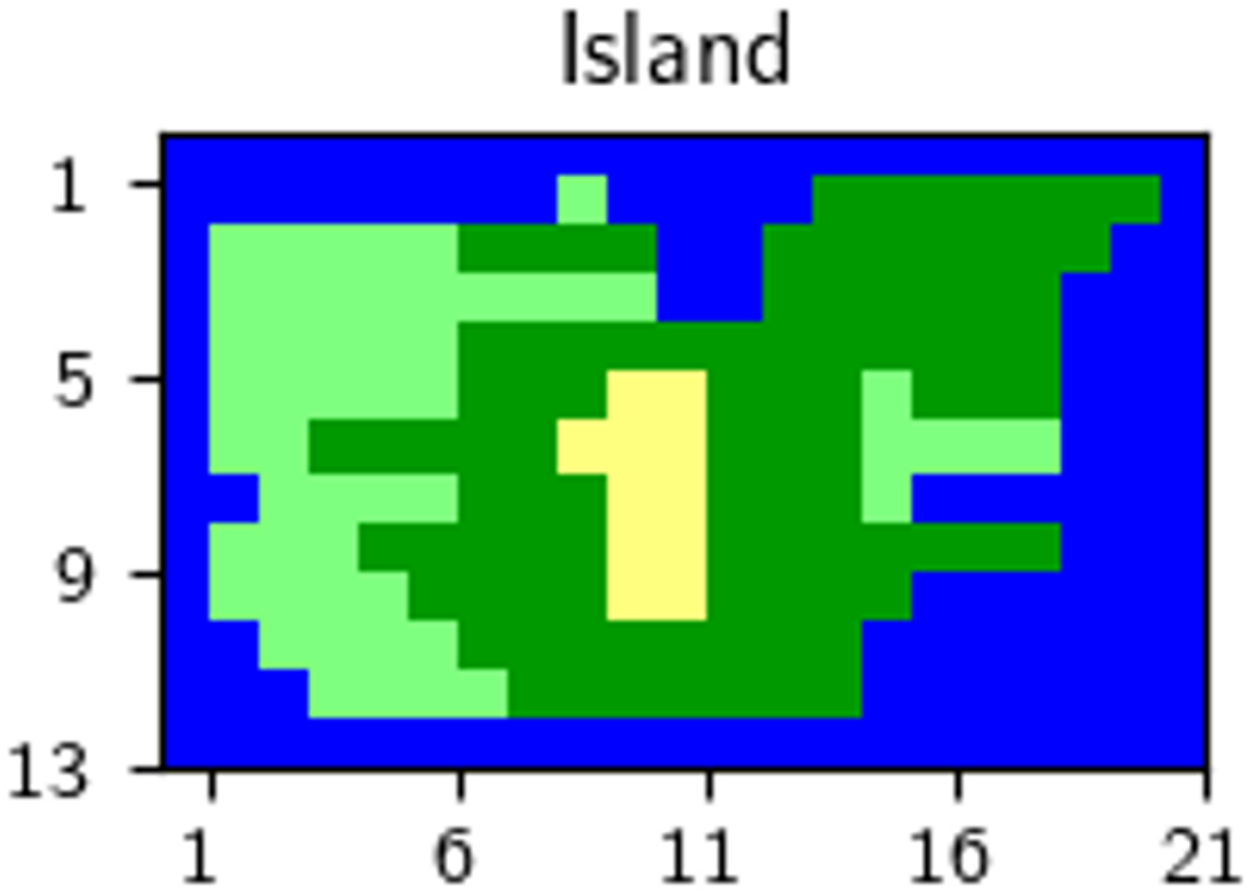
- Runtime complexity: O(N)

Visualizations



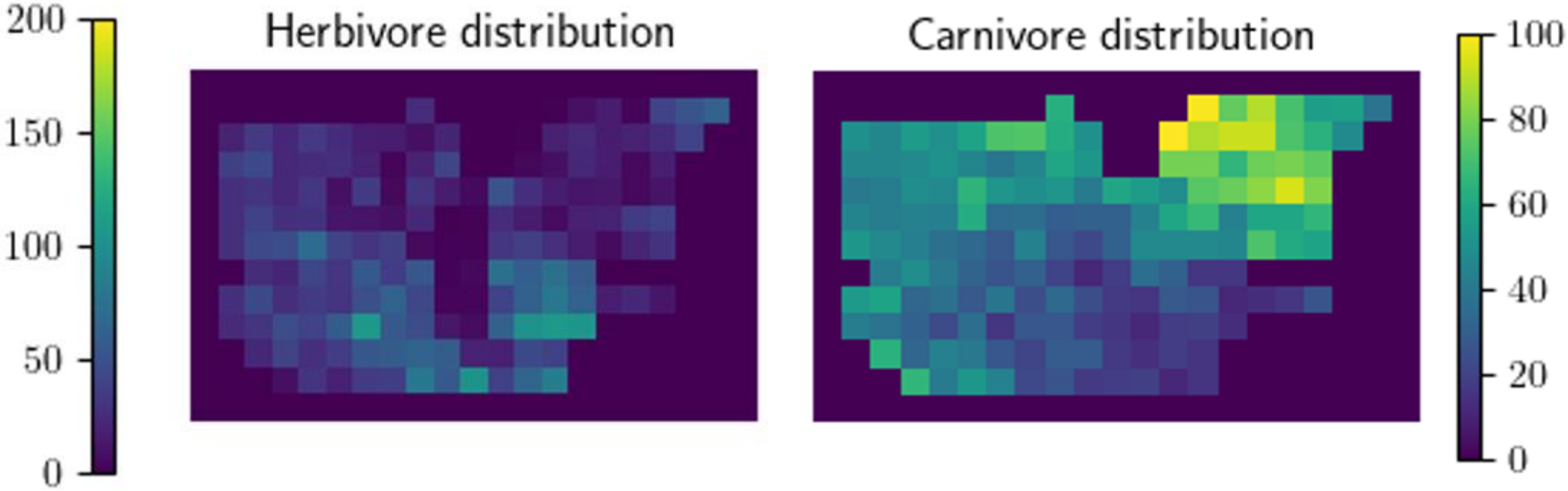
Visualizations

Island



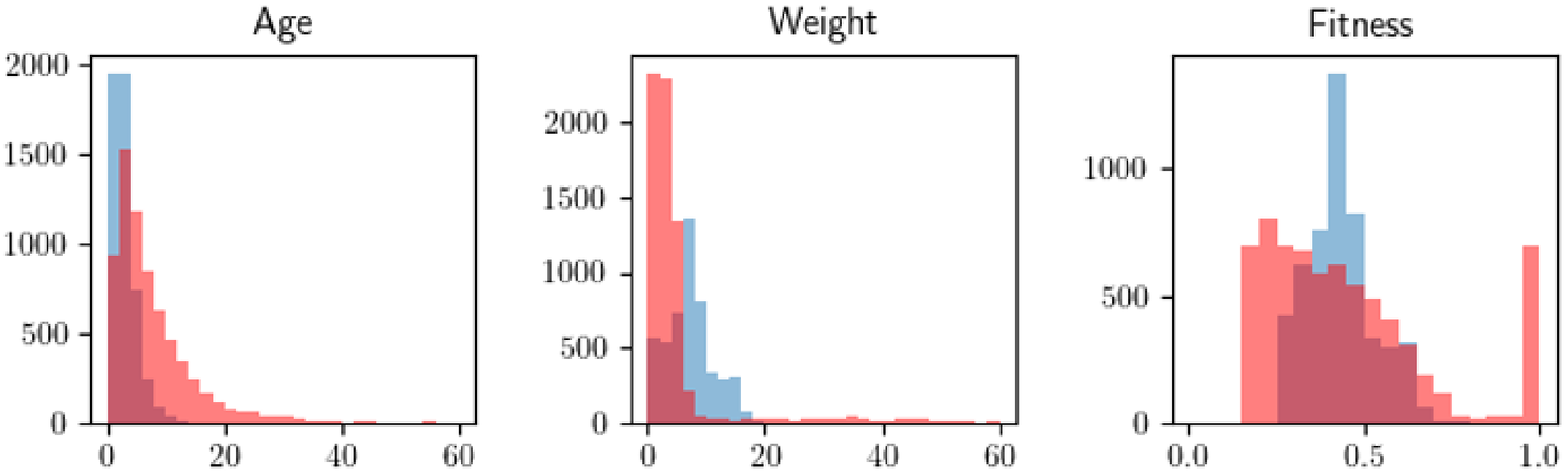
Visualizations

Herbivore & Carnivore distributions



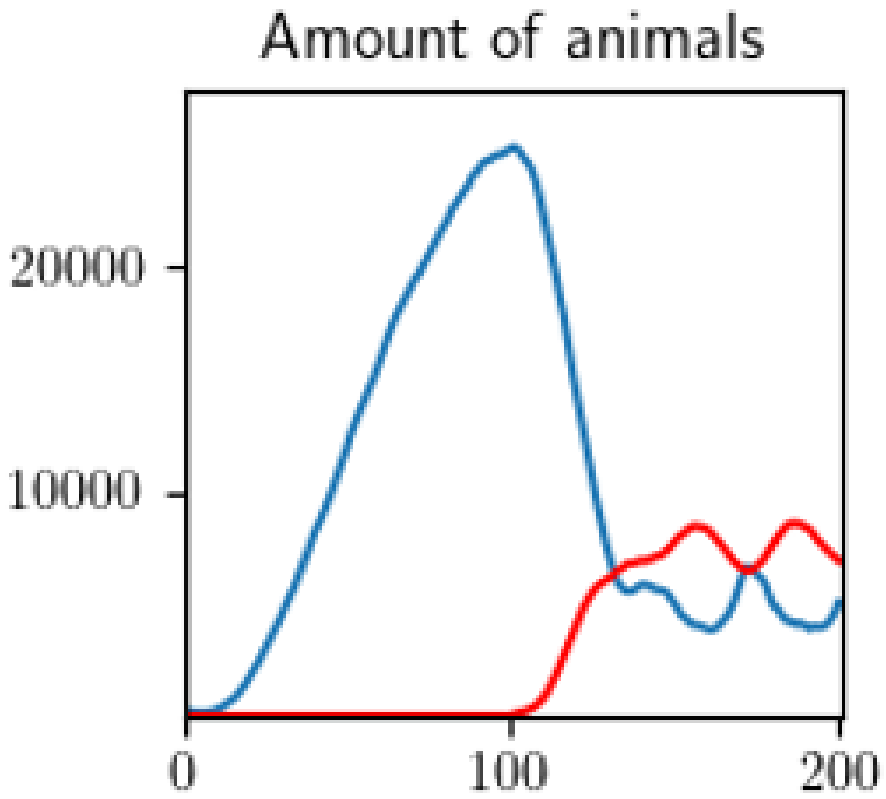
Visualizations

Vitals histograms – Herbivores & Carnivores



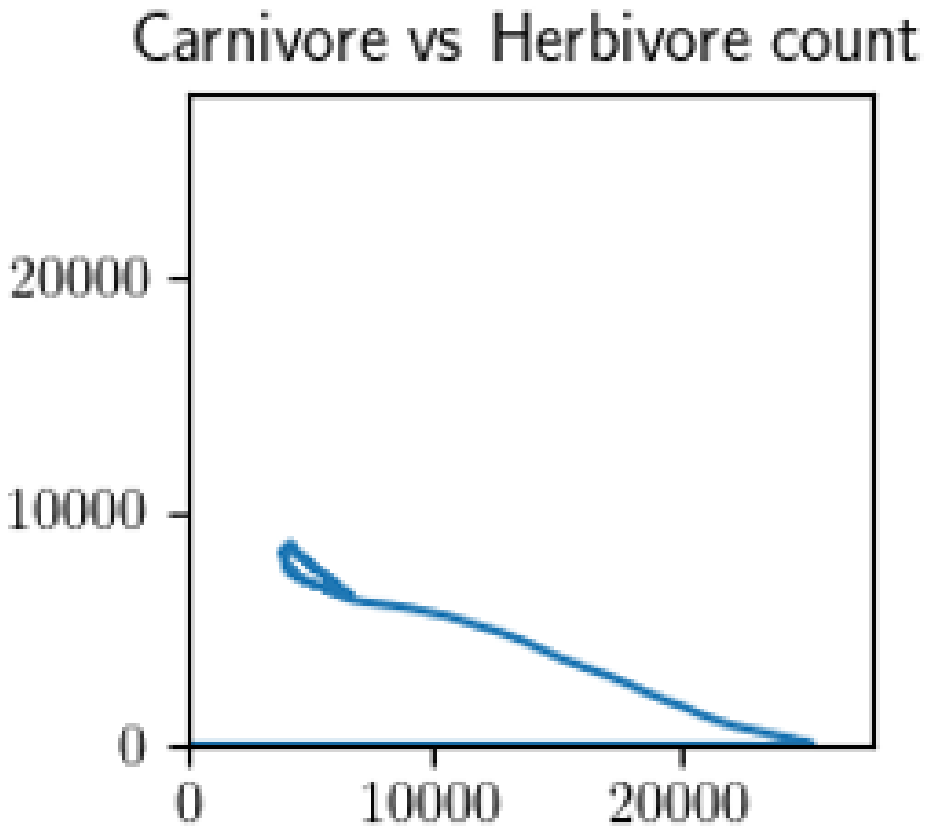
Visualizations

Counts – Herbivores & Carnivores



Visualizations

Counts – Herbivores vs Carnivores



Future improvements

- GUI to control the simulation
- Optimization of the code for faster simulation
- Making save files compatible with dynamic visualization.
- Additional graphs showing other information in the simulation
- Notes as the simulation happens
  - User can write their notes in parallel to the simulation

References

- Code and overview by Hans Ekkehard Plesser, https://gitlab.com/nmbu.no/emner/inf200/h2021/inf200-course-materials/-/tree/main/january_block
- Pytest, https://docs.pytest.org/en/6.2.x/
- Tox, https://tox.wiki/en/latest/
- Matplotlib, https://matplotlib.org/
- Sphinx, https://www.sphinx-doc.org/en/master/
- FFMPEG, https://ffmpeg.org , https://imagemagick.org