Lab - Week 2

In this week's lab you will:
- familiarise yourself with the Schelling model as one of the simplest (but most famous!) examples of a multi-agent system,
- start working on the software structure that you will need for the semester long project in this context.

Along the way, we will ensure that you have all the software available and running for the rest of the Semester.

Enjoy!

1. The Schelling Model

We implement the Schelling model that we have seen in the lecture. This is a very useful exercise because it will prepare you for working with grid worlds later and these are the staple of agent-based reinforcement learning. You may also later want to the visualisation method provided here for your project, so this is a good point to get familiar with it.

Tasks

Implement a full grid-based agent-based simulation of the Schelling model in Python.

Since visualising the state interactively can be a bit tricky in Python, particularly from within a Jupyter notebook, we provide a *Jupyter* notebook skeleton in a separate file that contains the code for the interactive visualisation. In this way you can focus on the core functionality. Note that this work in Anaconda, our reference platform, but may not work in some other environments. Specifically, it will *not* work in Colab.

Jupyter notebook implementations can be very fickle with updating graphics dynamically, so it is

possible that you experience sluggish updates etc. depending on the Jupyter platform that you are using. If this is the case, it may be best to use the plain Python version instead. To obtain a plain Python *.py file from the notebook use the "*Save and Export Notebook as...*" menu item, saving as "*Executable script*"

1.1 Find the bits in the skeleton code that are left undefined (initialisation, stepping and data collection) and complete these.

For the purpose of this simulation, we defined that an agent as "unhappy" and consequently moves if less than a fraction *c* of the other agents in their immediate 8-neighborhood are of the same color. For the demo in the lecture we used $c = 0.7$

1.2 Define sensible measures for overall (population-wide) happiness and segregation and use your ABM to show how these develop and that our assertions about the behaviour of the Schelling model are correct. In which regions of *c* do you find complete segregation?

2. A basic software structure for grid-worlds

You may not be able to complete this second task in the lab. This is expected and not a problem at all. Just continue to work on it outside the lab - the main purpose of this is to get started on your project (the assignment).

The Schelling model is based on a structure that is widely used to experiment with or to conceptualise multi-agent systems: A grid-world, in which multiple agents interact. Each agent occupies a grid cell, can move between cells, and agents can sense their environment (in limited ways) to make decision about their movement. Agents may be of multiple types that can have different behaviours.

Your task is to recast the above implementation of the Schelling model, into such a "generalised" software structure.

The main purpose of this exercise is for you to start creating a software structure that you can use as the basis for the semester-long project. You can start to work in teams already if you choose to do so (but you don't have to!)

It is useful to use object-oriented structures for this implementation. Note, however, that this is not strictly required. This is not a programming class! While it is very sensible to program this in an object-oriented way, our problems are so simple (from a coding perspective) that it is equally possible to create a good implementation using parallel matrices.

Tasks

2.1 Design the interface structure for your grid-world. Your software should, in principle, be able to cope with grid worlds of arbitrary size and with arbitrary numbers of agents (we will normally use

relatively small sizes, such as 10x10 and small numbers of agents). It should also support different agent types (we will not sue more that two types but there is no good reason why you should limit this in the implementation).

Note that agents, beyond their position may need to be able to store other attributes (in principle, arbitrary contents). Furthermore, each agent type needs to define a) function/method that performs sensing (in the Schelling model this would be to determine the proportion of agent types in the immediate neighborhood) and one that executes an action (e.g. moving) based on the sensed values.

Grid cells may also have additional attributes that can be modified by the agents.

You software should support functions/methods to perform the following

- initialise a new grid world of size *n* x *m*.
- initialise a new agent at a given position
- initialise a population of agents at random positions without allocating two agents to the same cell
- a main loop in which each agent is updated by letting it sense the environment and execute its associated action.
- move an agent by a step (dx, dy)

Agents should support functions/methods to:

- retrieve information from the environment (sensing)
- decide and execute an action based on the sensed information (e.g. to move). Actions may include moving and/or modifying the attributes of the grid cell in which the agent is present); note that the latter is not used for the Schelling model.

2.2 Re-implement the Schelling model using this software structure.