Introduction to Agent-Based Modeling

—

Simone Callegari, 2016

Theory, experiment, simulation.

Simulations: equation-based (*top-down*, find numerical solution to established equations), agent-based (*bottom-up*, modelization of system components), hybrid/multiscale (sub-grid parametrization), Monte Carlo (also used within other simulation types).

1 Game of Life

Conway's *Game of Life* (1970) is a cellular automaton: a dynamical model/system in which space is discretized into a regular grid of *cells*; for each cell, different variables can represent its *state*. In the case of the GoL, there is only one variable attached to each cell, which can take the values DEAD (0) or ALIVE (1). Thus, the GoL is a *discrete space* model, as are by definition all cellular automata; moreover, in this model each cell variable can only have a discrete set of states (two).

Any dynamical system has rules for its *time evolution*, i.e. a way to specify the state of the system at a time $t' > t$ given the state of the system at time t (and possibly even previous times, if the system has *memory*) – the word "evolution" here simply means "change as a function of time". In the case of a cellular automaton, cells cannot change relative position; they can only change state. In cellular automata, as well as in agent-based models, such rules specify which **actions** can be performed.

Take a cellular automaton made of infinite cells distributed on a rectangular grid in 2 dimensions. Let's define a cell's *neighbors* as all the cells that are horizontally, vertically, or diagonally adjacent to that cell (this is called *Moore neighborhood*). The rules for GoL also assume *discrete time steps*: the system "jumps" from the initial time t to $(t+\tau)$, to $(t+\tau+\tau)$, and so on; or, alternatively, from timestep $T = 1$ to $T = 2$, then $T = 3$ etc.

Here are the rules:

- any DEAD cell with 3 ALIVE neighbors becomes ALIVE
- any ALIVE cell with < 2 or > 3 ALIVE neighbors becomes DEAD
- all other cells do not change state

At each time, the updates of all cells are **independent**. Once you specify an initial set of ALIVE cells, the time evolution of the system is uniquely determined by these rules; as such, GoL is a *deterministic* (i.e., not *stochastic*) cellular automaton. Implement one using the concept of a `class` in Python.

One can already clearly see the difference between *behavioral rules* (that are the same for every cell) and actual *behavior* (which depends on **internal state** and on the **environment**).

Q: what could be the meaning of these rules, if you think in (very loose) analogy with biological populations?

Q: how fast is the "speed of light" in the GoL universe? It is determined by the minimum time interval after which the state of one cell can be affected by the state of another cell, depending on their distance.

Q: how does the behavior of GoL change if one switches from a Moore to a von Neumann neighborhood (only vertical and horizontal adjacency)?

$\Longrightarrow$ See `game_of_life_DT.py` for an introductory implementation.

2 Random walkers

The main difference between a cell and an *agent*, or *individual*, is that the agent can move in space. Particularly in sociology and economics (and sometimes in ecology), agents are also described as *purposeful*, in the sense that they try to maximize some function of their own state variables, usually representing fitness or gain.

Let us consider an agent living in discrete space and time. In particular, it lives on a rectangular grid with von Neumann neighborhoods. The agent is characterized by only two variables: its coordinates on the x and on the y axis. Suppose that the agent starts at $x = 0$, $y = 0$, and call a the distance between two neighboring cells (the *step* of the lattice).

Simulate an agent that increases its own x position at every odd timestep, and its y position at every even one.

Q: what is the distance of the agent from its starting position, after 1, 2, 3 steps? what is its speed when $T \to \infty$?

2.1 Discrete time

Modify the agent behavior so that it performs a *random walk*: at each time, it chooses a neighboring cell with uniform distribution (implement a `goToRandomNeighbor()` function for this), and moves to it by updating its own x or y position. We are switching from a **deterministic** to a **stochastic** model (MONTE CARLO method).

Assuming that all agents start at coordinates $(0,0)$, the agent's expected r.m.s. distance from the origin after T steps is:

$$\begin{aligned}
\langle d_T^2 \rangle^{1/2} &= \sqrt{\langle x_T^2 + y_T^2 \rangle} \\
&= \sqrt{\sum_{i=1}^{T} 4 \cdot \frac{1}{4} a^2} \\
&= a\sqrt{T}
\end{aligned}$$

Remember in fact that $\langle x_T \rangle = 0$, and so:

$$\langle x_T^2 \rangle = \langle \left(\sum_{i=1}^{T} x_i \right)^2 \rangle = \sum_{i=1}^{T} \langle x_i^2 \rangle = T \, (P_+ + P_-) \, a^2$$

where $P_+ = P_- = \frac{1}{4}$. An analogous result can be derived from the variance of a binomial process in the case $P_+ \neq P_-$.

Q: how can you check for the scaling of the effective speed $v_{\text{eff}} \equiv \sqrt{\langle d_T^2 \rangle}/T$ with T, for example, using a linear regression method such as `numpy.linalg.lstsq`?

Q: what does it mean if you don't find the correct exponent of 0.5?

Q: what is the expected r.m.s. speed of the walker? what is its instantaneous speed? what is the "speed of light" in this model, i.e. the fastest speed at which a walker can reach some distance? what does this imply, if you give some real (physical) meaning to the space and time units of the simulation?

$\Longrightarrow$ See `random_walkers_DT.py`.

A stochastic process is a collection of random variables, indexed by some discrete or continuous parameter (time), $\mathbf{X}(t)$. A **realization** of the process is the series of actual observed values of these variables as a function of time. The **ensemble** is the set of (some or) all possible realizations of the process.

Run simulations of 10, 100, 1000 random walkers together and check again the scaling of the variance in their positions, which corresponds to the r.m.s. distance from the origin calculated above.

Q: what can you learn if you build a distribution of scaling exponents from many realizations of the random walk?

Q: it is not easy to visualize the walkers, especially close to the start of the simulation, when it is likely that many agents will occupy the same cells; how can you aggregate information about their state (position) in a way that is easier to visualize?

Q: is there any difference between running a simulation with 100 random walkers, or running 100 simulations with one random walker each and combining the results appropriately? (independence, RNG seeding)

The last ingredient we need to add to the random walkers is the *movement probability*. Rather than having all agents move at all times, we can specify a new parameter of the model, P_m, which is the same for all agents, representing the probability that each agent will move in any of the possible directions in a timestep; $1 - P_m$ is the probability that it will stay where it is. This introduces the notions of **macroscopic** vs. **microscopic** description of a system.

Q: how does this new parameter affect the dispersal speed of the agents?

In this case, our model now yields

$$\langle d^2 \rangle^{1/2} = a\sqrt{P_m T} \tag{1}$$

2.2 Continuous time

Here we introduce the concept of **waiting times** in the *Gillespie algorithm* (Gillespie 1976, originally by Doob), and their relation to the Poisson distribution.

Let's take the random walkers; suppose that we want to simulate the same population of agents, but using different numerical timesteps. If we want their macroscopic properties to be the same, independently of how we choose the timestep of the simulation, P_m will have to depend on the duration of the timestep. Consider as an example two models: model A with timestep τ and model B with timestep $\tau' = 2\tau$. Looking at eq. 1 it may seem clear that, if we want to have the same $\langle d^2 \rangle$ at the same real time (remember that the real – macroscopic – time elapsed is proportional to T in eq. 1), in case B we will have had half the number of timesteps of case A, so we should need *twice* the move probability. In other words, the meaningful quantity for the model is the **rate** of movement

$m = P_m/\tau$, which is a *probability per unit time*; once you choose a time unit, you can derive the correct probability to put into your ABM.

Is this an exact correspondence between models A and B? Look at the probability that an agent moves *exactly once* within a time 2τ in model A:

$$P(1\,\mathrm{move\,in\,2\,steps}) = P_m \cdot (1-P_m) + (1-P_m)\cdot P_m = 2\,P_m - 2\,P_m^2$$

This means that the equivalence between models A and B is exact *up to a term of the order of* $P_m^2 \propto \tau^2$. The smaller the timestep (better time resolution), the better the correspondence between models.

Q: to what kind of event does a probability $\propto P_m^2$ correspond?

Q: how do you reconcile this with the scaling argument based on eq. 1?

Q: is there any limitation due to the fact that probabilities have to be comprised between 0 and 1? any relation with what we called the "speed of light" of the model?

What is the probability that a particle will make a single movement within T timesteps? The answer is given by the binomial distribution for 1 success among T trials (a.k.a. Bernoulli distribution):

$$P(1\,\mathrm{move\,in}\,T\,\mathrm{steps}) = \frac{T!}{1!\,(T-1)!} P_m\,(1-P_m)^{T-1}$$

What happens if we now go to the *continuum limit*, i.e. we consider models with smaller and smaller timesteps, $\tau \to 0$? This is the same as saying $P_m \to 0$ and $T \to \infty$; in this case, the binomial becomes a Poisson distribution for 1 success in a time $t \equiv T\tau$ (which is a sequence of Bernoulli trials):

$$P(1\,\mathrm{move\,in\,time}\,\tau) = m\,t\,\mathrm{e}^{-mt}$$

where, again, m is the rate of movement. For this reason, this type of events in continuous time is also called *Poisson point process*. One key feature of such process is the complete independence of the events in different time intervals (the agents have **no memory**, and every time interval is statistically equivalent). The expected number of moves performed by an agent within a time interval t is proportional to the length of the time interval itself, and it is equal to $\langle N_{\mathrm{moves}}\rangle = mt$; it does not depend on *when* we pick such interval.

What is the expected time before an agent decides to move (the **waiting time** of this process)? In general the Poisson distribution for k successes in time t reads

$$P(k,t) = \frac{1}{k!}(mt)^k\,\mathrm{e}^{-mt}$$

so the probability that the *first* move will be at t_1, *after* some time t, is equal to the probability that the agent does not move before then:

$$P(t_1 > t) = P(k(t) = 0) = \frac{1}{0!}(mt)^0\mathrm{e}^{-mt} = \mathrm{e}^{-mt}$$

Implement a continuous-time ABM of random walkers in which each agent waits for an exponentially distributed time before its next move. How would you do that?

A few techniques for sampling variables from a continuous probability density:

- rejection sampling: given uniformly distributed x and u, reject x if $u > p(x)$

- inverse transform: if you want x distributed according to $f(x)$ and $F(x) \equiv \int^x f(y)\mathrm{d}y$, then take $x = F^{-1}(u)$ with u uniformly distributed between 0 and 1, since

  $$\Pr(F^{-1}(u) \leq x) = \Pr(u \leq F(x)) = F(x)$$

  if needed, implemented via a look-up table

- slice sampling and others (will not be used here)

$\Longrightarrow$ See `random_walkers_CT.py`.

Q: how do the results from the continuous-time and the discrete-time ABM compare to each other?

3 Master equation

The Master equation describes the time evolution of a system that can be in one of many *states*; this description is usually employed for stochastic systems and involves *transition probabilities* from one state to another. For example, in a system consisting of a single random walker on a grid, the transition probabilities are $P_m/4$ for the walker to move to each of the neighboring cells, and $1 - P_m$ for the walker to remain in the same state (not change its position). Probability is always *conserved.* A stochastic process in which the transition probabilities between the states at time t and time $t + \tau$ depend only on the state of the system at time t (i.e., without any effect of the previous history) is said to be a *Markov process*, or, in the specific case of discrete-time, a *Markov chain*.

In general, the Master equation can be written in many forms; we are interested for the moment in the equation for a system with a discrete set of states (for example, the positions of particles on a spatial grid), that is

$$\frac{\mathrm{d}p_i(t)}{\mathrm{d}t} = \sum_j \left[W_{ij}p_j(t) - W_{ji}p_i(t)\right]$$

where $p_j(t)$ is the probability that the system is in state j at time t, and W_{ij} is the transition rate from state j to state i. In words, this equation says: *"the rate of change in the probability that the system will be in a particular state is equal to the sum of all possible transitions <u>to</u> this state minus all possible transitions <u>away from</u> this state"*.

In the case of the random walkers, the rate of change in the probability that you will find agents in a particular cell is equal to the probability that agents will walk **to** such cell from its neighbors, minus the probability that agents already present in that cell will walk **away** from it.

Once you specify some initial conditions in terms of a probability distribution (e.g., a delta function centered on a particular state), the stochastic evolution of the system can be followed with the Master equation. For example, the *Leslie matrix* used in structured population dynamics is actually an application of the Master equation.

The Master equation is extremely useful, for example in relating the microscopic and the macroscopic properties of a system, as shown in the next section.

4 Diffusion as the continuum limit of random walks

Let's apply the master equation concretely to our discrete-time random walkers. The probability of moving from a cell to one of its neighbors is always $\frac{1}{4}P_m$ (we assume each cell to have 4 neighbors).

So, the change in probability that there are particles in cell i (whose neighbors are indicated by the index j) is

$$\frac{p_i(t+\tau)-p_i(t)}{\tau} = \frac{1}{\tau}\left[\sum_j \frac{1}{4} P_m p_j(t) - P_m p_i(t)\right]$$

In the continuum limit ($\tau \to 0$ and also the distance between cells $a \to 0$), calling $n(x,t)$ the expected density of agents at point x and time t,

$$\frac{\mathrm{d}n(x,t)}{\mathrm{d}t} = D\nabla^2 n(x,t) \tag{2}$$

where we have introduced the 2-dimensional **macroscopic diffusion coefficient**

$$D = \lim_{\tau,a\to 0} P_m \frac{1}{4}\frac{a^2}{\tau}$$

connecting the large-scale behavior of the ABM to the microscopic movement probability and geometry of the cell grid. Eq. 2 is the **diffusion equation**.

In order to derive it, remember that the finite-difference approximation of the second derivative is (to lowest order):

$$\left(\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right)_{x=x_i} \approx \frac{f(x_{i+1})+f(x_{i-1})-2f(x_i)}{(\Delta x)^2}$$

where $\Delta x = (x_{i+1}-x_i) = (x_i - x_{i-1})$.

The solution to the diffusion equation applied to the density of a group of agents (all starting their movement from the same point in space) is a (multi-variate) Gaussian whose variance σ^2 increases linearly with time, according to eq. 1.

5 Population growth

Let's introduce birth and death into a population of agents. The number of agents as a function of time, $N(t)$, will be

$$N(t+\tau)-N(t) = B(N,t,\tau,...) - D(N,t,\tau,...)$$

where B and D are, respectively, the number of births and deaths within time τ. In analogy with the movement rate, we can define the **growth rate** of a population as

$$r(N,t,\tau,...) \equiv \frac{1}{\tau}\frac{N(t+\tau)-N(t)}{N(t)} = \frac{1}{\tau}\frac{B(N,t,\tau,...)-D(N,t,\tau,...)}{N(t)}$$

Consider a stochastic, continuous-time ABM, in which the *per capita* rates for giving birth and/or dying are $b(N,t,...)$ and $d(N,t,...)$:

$$\begin{aligned}\langle B(N,t,\tau,...)\rangle &= \langle b(N,t,...)N(t)\,\tau\rangle \\ \langle D(N,t,\tau,...)\rangle &= \langle d(N,t,...)N(t)\,\tau\rangle\end{aligned}$$

assuming that birth and death events are uncorrelated. The expected growth rate is then

$$\langle r\rangle = \langle b-d\rangle\ .$$

In continuous time, we can assume that the probability of having more than one birth or death at a single instant is negligible. Then, according to the master equation:

$$\begin{aligned}\frac{\mathrm{d}P(N(t))}{\mathrm{d}t} &= b(N-1)\,(N-1)\,P(N-1,t)\\ &\quad + d(N+1)\,(N+1)\,P(N+1,t)\\ &\quad - [b(N)+d(N)]\,N\,P(N,t)\end{aligned}$$

from which we get

$$\left\langle \frac{\mathrm{d}N(t)}{\mathrm{d}t} \right\rangle = \langle bN \rangle - \langle dN \rangle$$

We could write a (more complex) Master equation also for a discrete-time ABM of the same type, remembering that we can have many, binomially distributed births and deaths within a timestep τ.

5.1 Exponential

If a population has a constant growth rate,

$$N(t+\tau) = N(t) \cdot (1 + r\tau)$$

- Q: implement two discrete-time ABMs with constant growth rate, one deterministic, the other stochastic

It is easy to integrate analytically the continuum limit ($\tau \to 0$)

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t) \tag{3}$$

(add expectation values if the model is stochastic!) to obtain

$$N(t) = N(0) \cdot \mathrm{e}^{rt}$$

that is, exponential growth, stationarity, or decline: $r \leq 0$ is also allowed.

- Q: implement a continuous-time ABM of exponential growth and compare it with the analytical solution; for this model, you can use the random walkers that you have already implemented: why? (because they are all independent, and therefore space and movement have no effect on growth)

- Q: how would you integrate eq. 3 numerically? how does it compare to the discrete- and continuous-time ABMs?

5.2 Logistic

No population can grow indefinitely; some factor will intervene to slow down and eventually halt growth. The population may reach a "saturation" point and stay there, in equilibrium. This stable point is often quantified with a parameter called **carrying capacity** K, which represent a sustainable population size or density.

In classical population models, this is usually described by logistic population growth:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN\left(1 - \frac{N}{K}\right) \tag{4}$$

This equation has a sigmoid or exponential analytical solution (depending if $N(0) < K$ or $> K$), has stable points at $N = 0$ and $N = K$, and looks like exponential growth for small populations (this is why r is sometimes called "zero-population growth rate"). The sigmoid solution is:

$$N(t) = \frac{KN(0)}{(K - N(0))e^{-rt} + N(0)} .$$

If you consider eq. 4 in discrete time (as a *recurrence relation*), it displays notriously a "deterministically chaotic" behavior; this is the case for real populations with non-overlapping generations ("Simple mathematical models with very complicated dynamics", R. M. May, Nature, 1976).

Note that there are two r.h.s. terms: one term $\propto N$ and the other $\propto N^2$. The first can be interpreted as an "intrinsic" characteristic of each agent, the second some kind of "interaction" between the agents, in this case with a negative effect on population growth.

There are many ways to implement eq. 4 as an ABM. For example, we cast the logistic growth equation into a large-scale expectation value of some agent-level behavior, by considerig the expectation value of the *per capita* growth rate (i.e., the contribution of each individual to population growth):

$$\left\langle \frac{1}{N}\frac{\mathrm{d}N(t)}{\mathrm{d}t} \right\rangle = \langle\alpha\rangle - \langle\beta N(t)\rangle$$

Assuming $\alpha = r$ and $\beta = r/K$ constant, the rates at which an agent gives birth and dies, respectively, can be given by the Verhulst model:

$$\begin{aligned} \tilde{b} &= r \\ \tilde{d}(N) &= r\frac{N(t)}{K} \end{aligned}$$

This may be interpreted as a constant fertility of the agents, combined with a density-dependent death rate (for example, due to competition for limited resources). But this is just one of many possible interpretations.

These processes are sometimes expressed as follows in analogy with chemical reactions, with their respective probabilities:

$$A \xrightarrow{\beta} A + A$$
$$A + A \xrightarrow{\delta} A$$

and in the same way, "intrinsic" death would be

$$A \xrightarrow{\delta'} \emptyset$$

Q: try an ABM in which the self-interaction term corresponds to agents killing others with a certain probability; plus, the agents have intrinsic birth and death probabilities; where does the "natural death" probability enter in the large-scale logistic equation?

Q: code ABMs of logistic growth with "linear death"

What happens to our large-scale growth rate in the Verhulst formulation? Remember that $\langle N^2\rangle = \langle N\rangle^2 + \sigma_N^2$ where σ_N is the variance in population size. Therefore, the expectation of the ABM growth rate is actually

$$\left\langle \frac{\mathrm{d}N(t)}{\mathrm{d}t} \right\rangle = \alpha\langle N(t)\rangle - \beta\langle N(t)\rangle^2 - \beta\,\sigma_N^2$$

i.e., with this stochastic ABM we are *underestimating* the growth rate compared to the classical model. The equilibrium population size is also lower than the classical K. This is in addition to any discrepancy due to your choice of timestep.

6 Fisher-Kolmogorov-Skellam populations

When random walk and logistic growth are combined into a single ABM, we get a discrete, stochastic version of the popular Fisher-Kolmogorov-Skellam population model.

$\Longrightarrow$ See `verhulst_DT.py`.

Q: what **emergent property** do you see in the population?

Q: how would you estimate the most important quantity related to such emergent phenomenon, using the parameters of the model? (concepts of **dimensional analysis** and **scaling relations**)

Q: test an ABM with constant birth and death probability, in which only one agent at a time can occupy a cell; offspring is created only if space is available in a neighboring cell

$\Longrightarrow$ See `exclusion_CT.py` and `killers_CT.py` for examples of emergent carrying capacity.¶

7 Environment resources

Suppose that large-scale carrying capacity arises from the balance between resource production and use by a single species; how would you estimate an expected K from the parameters of an ABM designed to test this hypothesis? Implement the ABM "cows and grass" and compare to your expectations.

Q: random walkers with BMR vs. resource growth evolve their population-averaged movement rate (at birth, each cow inherits the mother agent's rate plus/minus some random variation); do they settle around an "optimum" move probability? what happens if you add an energetic cost to each movement action?

Q: what happens if you add the restriction that an agent *needs another agent at the same spatial location* in order to reproduce (a version of the **Allee effect**)?

$\Longrightarrow$ See `cows_grass_DT.py` and `cows_grass_CT.py`. For a more complex model, check out `cows_grass_evo_CT.py` in which the cow population evolves its movement probability parameter with time.

8 Predators and prey

In the classical Lotka-Volterra model, the"self-interaction" term of logistic growth becomes a pairwise *inter-specific interaction* term. In a two-species system, we assume:

- in the absence of species B (predator), species A (prey) should thrive
- in the absence of A, B should die out

In ABM terms, we can give to the species the following actions: a *sheep* (A) moves and gives birth (e.g., with constant rates), while a *wolf* (B) moves and feeds (if feeding is successful, wolf reproduces and target sheep dies; if it is not successful, wolf dies). A model with more long-term oscillations would be to give wolves an energy content, which increases at every successful feeding and decreases with time and actions; a zero or negative energy leads to the wolf's death. Anisotropic movement can be introduced for sheep, wolves, or both; in the case of sheep, how would you introduce a preference for *low* wolf densities? (see 10.1)

$\Longrightarrow$ Try running `predprey_DT.py` a few times and check for different emerging behaviors!

9 Epidemics: the S-I-R model

To study the spread of disease in a population, we use agents characterized by a 3-state variable representing their health state, which can take the values SUSCEPTIBLE, INFECTED, or RECOVERED (and immune to further contagion). Let's call β the rate at which each of the N total individuals go from susceptible to infected (in dependence of contact with other infected individuals within some *infection radius*, giving rise to a pairwise interaction term), and γ the rate at which infected individuals recover and become immune to the disease.

In classical dynamics of non-spatial epidemics, calling the number of individuals in each of these states as S, I, and R, respectively, this standard model corresponds to the following time evolution:

$$\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{d}t} &= -\beta\frac{S\cdot I}{N} \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= \beta\frac{S\cdot I}{N} - \gamma I \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I
\end{aligned}$$

In the classical model, the parameter $R_0 \equiv \beta/\gamma$ determines if there will be an epidemic ($R_0 > N/S(0)$). A typical application of this model is to the Hong Kong Flu that hit New Yorkers in 1968-1969, with a contageous period of 3 days and a 12-week duration of the epidemic.

Q: the classical model assumes that the population is *well-mixed*: you can check the classical solution by using the `odeint()` function in `scipy.integrate`; how does the ABM differ from the equation-based expectations, as a function agent density and movement rate?

Q: experiment with some small level of *inoculation* against the disease, which adds the possibility that an agent switches directly from SUSCEPTIBLE to RECOVERED directly

Q: modify this system into a "zombie apocalypse" scenario: which behavioral rules would you add / change / remove?

Q: the S-I-R model can be modified into a S-I-R-S, or a S-I-S model, among others; what changes do you expect in the population's behavior?

$\Longrightarrow$ Compare the outcomes of `contagion_DT.py` and `classic_SIR.py`.

10 Movement and Environment

10.1 Environment perception

By introducing a non-homogeneous enviroment, we have a chance to explore non-isotropic movement strategies. In order to do so, we use the *"Roulette wheel"* algorithm. We use a function $W(...)$ that converts some property of each spatial location i (e.g., density of grass in a cell) into a statistical weight W_i. Then, the probability of moving to cell i is $P_i \propto W_i$. By using the standard random variable u uniformly generated in the $[0,1)$ interval, the agent will decide to move to neighboring cell i (with $i = 1,2,3,4$ and $W_0 = 0$) if

$$\sum_{k=0}^{i-1} W_k \leq u < \sum_{k=0}^{i} W_k \ .$$

Of course, the sum of the weights for all possibilities presented to an agent has to be normalized to 1. It is important to note that this is just another version of the *inverse transform sampling*: we are using the sum, i.e. the (discrete) integral of the statistical weights (probabilities), to sample the agent's probability distribution of moving to each neighboring cell.

The weight function can be of any form, but keep in mind that constant multiplicative factors will always be normalized away. A good choice would be a weight of the form $w_i = a + b \cdot G_i^\alpha$, where G_i is the grass content (or some other property) of cell i and α is a *non-linearity parameter* ($\alpha > 1$ biases cows more strongly towards places with more grass, while $\alpha < 1$ tends towards the random movement $\alpha = 0$ limit); a high a/b makes the movement less biased and more isotropic.

Q: what happens when you introduce anisotropic movement into the cows–grass system? do you expect that it will increase or decrease the carrying capacity? how does it change the spatial distribution and the time evolution of the population?

$\Longrightarrow$ Check out `cows_grass_aniso_DT.py`, where cows tend to move towards higher grass densities, and compare the large-scale system behavior with that of `cows_grass_DT.py`, where cows move "blind".

When spatial or temporal patterns arise, two of the most common analyses you can perform are the *2-point correlation function* and the *Fourier decomposition*. The latter is a standard analysis whose details can be found in many places, and is implemented in the `numpy.fft` library. The former measures the excess probability of finding two agents at some mutual distance, relative to the expectation due to a random distribution of the agents. A simple formulation is the "Landy-Szalay" equation

$$\xi(r) = \frac{DD - 2DR + RR}{RR}$$

where DD indicates the probability of finding two agents ("D" for "data") at distance r in your simulation (computed among all possible pairs of agents), RR the probability of finding two random points at the same distance ("R" for "random", and DR the cross-correlation of agents and random sample. The "R" sample correspond to generating a number (at least N_{agents} of points randomly and uniformly distributed on your grid (as you may do, e.g., for your initial conditions).

10.2 Self-organization and collective motion

Self-organization can arise by including some form of *memory* in the agents or their environment, which is at the basis of an autocatalytic reaction. For the grass-cow system, this memory is "recorded" in the grass density. In the case of ant colonies, pheronome trails mediate a positive feedback loop between the trajectories of ants at different times. A classical ABM of forager ant self-organization (Goss et al. 1989, Deneubourg et al. 1990), models ants as agents characterized by their position on a 2-d "floor" and by an internal state that can be either SEARCHING or NOT SEARCHING (for food). Ants start their journey at the colony the ($x = 0$, $y = 0$), and food is situated on the opposite corner and at random locations. Ants that are SEARCHING can only increase their x or y position (moving away from the colony); in the original papers, the probability of increasing x is a slightly modified version of the weights discussed in section 10.1:

$$P_{+x} = \frac{(k + N_{+x})^n}{(k + N_{+x})^n + (k + N_{+y})^n}$$

where N_{+x} is the number of ants that have already gone through that cell; and analogously for the $+y$ choice. The information about previous ants is "saved" on the floor in the form of a pheromone trail.

The parameter k guarantees that relatively unexplored areas are approached with a roughly random choice, when little information has been left by previous ants; the parameter n determines the degree of non-linearity of the choice when information is available. When an ant reaches food, it switches to the NOT SEARCHING state; ants that are NOT SEARCHING can only go back in x or y until they get to the colony.

- Q: a best fit to the data in the original paper gives $k = 20$ and $n = 2$; compare with a random walk ($n = 0$) and with the weights model in 10.1
- Q: how does the system behavior change by making it more realistic: pheromones fade with time and food is exhausted after repeated visits?
- Q: code an evolutionary ABM (e.g., in which ants die if they do not reach food within a certain number of moves) and see if the system approaches an optimum configuration
- Q: is there any difference between an ABM in which all ants start at the same time, and one in which ants are gradually "injected" into the system from the origin?

$\Longrightarrow$ See `ants.py`.

Flocks of birds and schools of fish exhibit collective motions, in which all individuals seem to agree on the direction followed by the group. This same level of self-organization can be achieved without memory, by using a short-range interaction ABM and some kind of **positive feedback loop**. Consider a population in which each agent moves at the same speed v and reacts to its neighbors' behavior by adjusting its own direction of motion θ_i:

$$\theta_i(t+\tau) = \langle \theta(t) \rangle_{\text{neighbors}} + \eta$$

where η is some random noise term (uniform, gaussian...). Note that you may have to use a function such as `math.atan2(s,c)` to compute $\langle \theta \rangle$. As you can see, one of the collective effects of flocking behavior is that a **consensus** is reached, and spatial clusters (i.e., flocks) may arise.

- Q: study the emergent behavior of a 2-dimensional group of agents as a function of how the neighborhood is defined, and how large η is (concept of **phase transition**); do you expect flocks to arise when the "neighborhood" is large, or small?
- Q: refine the ABM by including the fact that agents can only see the neighbors that they have in front (cutoff angle)

$\Longrightarrow$ See `flock_DT.py` for a basic model of bird-oids.

We can modify the agents' behavioral rule slightly, by asking that they try to avoid close contact with others. This can be done by using a very short-range *repulsive force* between the agent. If you add a preferred direction for each agent, this type of model is used to reproduce movement patterns of crowds of pedestrians.

10.3 Lévy walks and foraging

A Lévy walk (or Lévy flight) is a superdiffusive motion, in which the probability distribution of spatial steps is "heavy tailed". Here we focus on a particular case of Levy walk, for which the cumulative probability of step length a is:

$$P(a < l) = \begin{cases} 0 & l < L \\ 1 - \left(\frac{l}{L}\right)^{-d} & l \geq L \end{cases}$$

where L is the minimum length of a spatial step and $0 < d < 2$. This implies that the probability density for step lengths above L is $P(a) \propto a^{-\mu}$ with $1 < \mu < 3$ (higher μ converges to Brownian motion, lower μ cannot be normalized). The Lévy walk also seems to reproduce some statistical properties of the movement *patterns* of many foraging species (bees, albatross, sharks, humans, amoebas), when positions are recorded at fixed time intervals (i.e., they imply changes in movement speed). However, it is highly debated whether they arise from a Lévy-like *process*, or they are an artefact due to time sampling and data binning. An alternative explanation for such patterns can be that the forager switches between two random walks (or more generally Brownian Motion) with different step sizes, a large one for SEARCHING mode and a small one for FEEDING mode (when in a patch).

We can use this distribution to generate spatial steps for our agents with the usual inverse transform sampling: $a = L \cdot u^{-1/d}$ where u is our uniform random variable. The direction of the step will have to be chosen independently.

Q: compare a Levy walk with a **random walk with persistence** (also called **correlated random walk**), i.e. a random walk in which the agent is more likely to go in the same direction as the previous step(s); what are your expectations before you run the ABM?

Consider agents with a Lévy-flight movement strategy, who are foraging in an environment where resources have a sparse, "patchy" distribution. You can define a **search efficiency** as some measure of the ratio between the number of visited patches — or the total collected food — and the distance travelled, either in total or between each patch visit.

Q: how would you set up such an ABM, assuming you are using a spatial grid for the environent?

Q: with an adaptive population of agents for whom movement costs energy, find out if there is an optimal value of the Lévy scaling exponent d (maximum efficiency), depending on the degree of patchiness in the environment

$\Longrightarrow$ A basic Lévy flight model in continuous time is implemented in `levy.py`.

The **marginal value theorem** (Charnov 1976) applies to the case of foraging with *diminishing returns* (i.e., the longer you stay in the same patch, the lower your food collection rate becomes): the predator should leave the patch when the marginal capture rate in the patch (i.e., the rate of energy assimilation minuts the rate of energy expenditure) drops to the average rate of patch encounters in the habitat.

Q: how would you test the theorem with a stochastic ABM?

11 Schelling's segregation model

Schelling's 1971 segregation model was a pioneering work in showing how collective behavior can emerge from simple agent-level rules. In it, agents belonging to one of two groups live on a grid with Moore neighborhood (see section 1). An agent is "happy" about its current position if the fraction of neighbors belonging to its same group is $f \geq R$, where R is an "intolerance" parameter.

Different update rules are possible. For example, at each iteration of the model, a random agent is considered, and, if unhappy, it is moved to a random empty location on the grid. The emergent result is that strong spatial segregation of group members can happen even with what could be considered a low intolerance level of each individual.

$\Longrightarrow$ See `schelling.py`.

12 Evolutionary games

We briefly discuss only a few key concepts: normal form and payoff matrix, Hawk/Dove, Nash equilibrium.

13 Projects

- zombie apocalypse in different environments
- evolution of cooperation in a hawk-dove population
- optimal foraging strategies in patchy environments, without or with agent interaction
- flock/herd/school reaction to predator: group breakup, chasers and escapers
- grass+herbivore+carnivore system, evolving towards equilibrium