Agent-Based Modelling Tooling Review for AMR-HUB

Hospital Movement & Infection Dynamics Simulation

Project Context

What We're Building

- Simulate patient & staff movement within hospital floors
- Model infection spread (AMR pathogens)
- Room-based spatial navigation with doors connecting spaces
- Support both topological (graph) and spatial (coordinate) modes

What Exists until this point

- Backbone of the model prototype in Python
- Custom spatial abstractions (Building → Floor → Room → Door)
- YAML-based space definitions
- Precursor SEIR infection status in the model

Main ABM Platform Options

- **GAMA**

Production-ready, spatial-explicit modeling with native GIS, A* pathfinding

- **Mesa**

Python-native ABM framework, seamless ML/inference integration

- **NetLogo**

Educational platform, large community, limited complex spatial modeling

- **Repast HPC**

High-performance computing focus, cluster-scale simulations

GAMA

Strengths

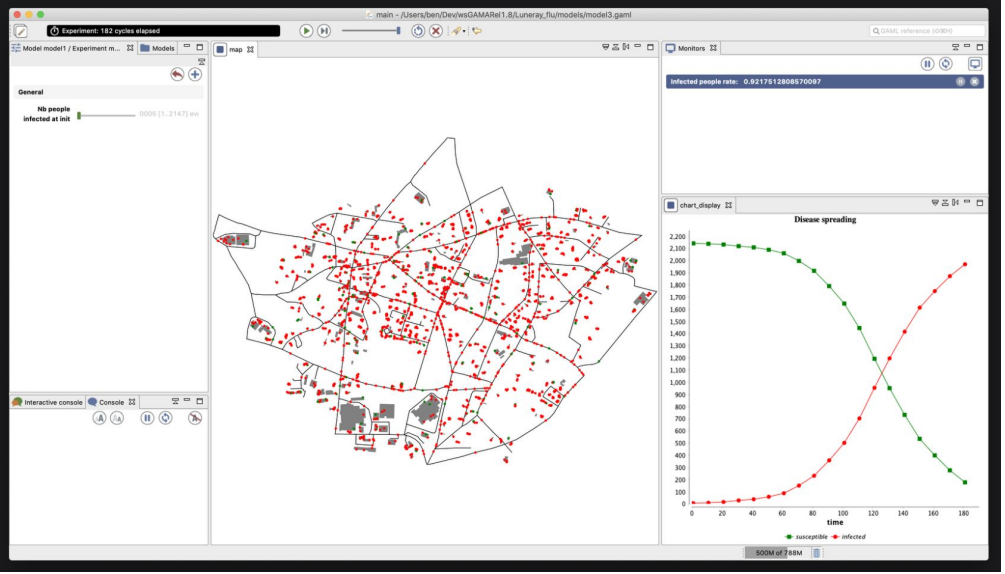
- Native GIS: shapefiles, CAD, GeoTIFF
- Built-in A* pathfinding & collision avoidance
- Multi-floor building support
- GPU acceleration available
- Direct precedent: HAI Simulator
- OpenMole Integration

Limitations

- Learning curve with GAML Language
- Python integration via export/import
- Inference tools require external plug-ins
- Would require significant rewriting of existing code

HAI Simulator (Healthcare Associated Infection Simulator) - direct example containing several pathogens, transmission modes, 3D hospital model, agent behaviours, disease models… → cannot find an open version of this

https://healthdatainsight.org.uk/project/healthcare-associated-infection-simulator/



From the flu tutorial

https://gama-platform.org/wiki/LuneraysFlu_step1

```
experiment main type: gui {
	parameter "Nb people infected at init" var: nb_infected_init min: 1 max: 2147;

	output {
		monitor "Infected people rate" value: infected_rate;

		display map {
			species road aspect:geom;
			species building aspect:geom;
			species people aspect:circle;
		}

		display chart_display refresh: every(10 #cycles)  type: 2d {
			chart "Disease spreading" type: series {
				data "susceptible" value: nb_people_not_infected color: #green;
				data "infected" value: nb_people_infected color: #red;
			}
		}
	}
```

```
global {
	int nb_people <- 2147;
	int nb_infected_init <- 5;
	float step <- 5 #mn;
	file roads_shapefile <- file("../includes/roads.shp");
	file buildings_shapefile <- file("../includes/buildings.shp");
	geometry shape <- envelope(roads_shapefile);
	graph road_network;


	int nb_people_infected <- nb_infected_init update: people count (each.is_infected);
	int nb_people_not_infected <- nb_people - nb_infected_init update: nb_people - nb_peop
	float infected_rate update: nb_people_infected/nb_people;


	init{
		create road from: roads_shapefile;
		road_network <- as_edge_graph(road);
		create building from: buildings_shapefile;
		create people number:nb_people {
			location <- any_location_in(one_of(building));
		}
		ask nb_infected_init among people {
			is_infected <- true;
		}
	}
}

species people skills:[moving]{
	float speed <- (2 + rnd(3)) #km/#h;
	bool is_infected <- false;
	point target;

	reflex stay when: target = nil {
```

Example project - **Healthcare Associated Infection Simulator**

https://healthdatainsight.org.uk/project/healthcare-associated-infection-simulator/

Developing a hospital acquired infection simulator

The first step was to model or simulate the spread of an infection within a hospital. Fred decided to use GAMA Platform, an open-source development environment for GAML, an agent-based modelling language.

Agent based modelling simulates the actions and interactions of agents (individual and/or collective entities) to understand the behaviour of a system. Due to the complexity of the hospital setting, agent-based modelling is an effective method of simulating the spread of infection within a healthcare system.

Using GAMA, Fred created a simulator that models the movement of people (agents) within a hospital and the spread of an infection amongst them, with an output of the number of infected people and the level of infectivity of each room within the hospital. The simulator can test different interventions, such as masking policies, to evaluate their effectiveness in reducing the spread of HAIs and to evaluate their cost-benefit.

The hospital acquired infection simulator was developed in 5 stages:

1. Creating a 3D model of a hospital
2. Modelling the behaviour of people in a hospital
3. Modelling the spread of an infection in a hospital
4. Bringing everything together to create a HAI simulator
5. Using the HAI simulator to evaluate strategies to reduce HAIs

Mesa

Strengths

- Writing a Multi-Agent simulation loop can be messy – scheduler
- Direct inference library integration (pyABC, sbi, ELFI)
- Modular, extensible architecture
- Built-in DataCollector and BatchRunner for parameter sweeps
- Browser-based visualization (SolaraViz)
- Existing prototype compatibility
- Multi-Agent RL compatibility via Gym wrapper from OpenAI, Evolutionary Algorithms and Inverse RL
- We can insert custom monitoring for disease “hotspots”

Limitations

- Manual pathfinding implementation required – we already made progress in this
- Collision avoidance not built-in
- mesa-geo exists but less mature than GAMA GIS
- Complex continuous space requires more work

Fit with Existing Codebase

- Our SpaceInputReader already handles Building→Floor→Room→Door hierarchy
- Agent class has SEIR states
- Natural extension path without rewrite

Topological Model Support

Our model uses both:

- Topological: Room connectivity graph (Room A → Door → Room B)
- Spatial: Coordinates within rooms (for fine-grained movement)

Mesa supports both natively:

- Network - graph-based topology
- ContinuousSpace - coordinate-based movement
- OrthogonalMooreGrid - grid-based

Mesa-geo

- Agents themselves with Shapely geometry (points, polygons)
- Space that understands coordinates, CRS, projections
- Import formats of Shapefiles, GeoJSON, GeoPackage, GeoPandas DataFrames

```python
from mesa.discrete_space import OrthogonalMooreGrid
from mesa.examples.basic.conways_game_of_life.agents import Cell


class ConwaysGameOfLife(Model):
    """Represents the 2-dimensional array of cells in Conway's Game of Life."""

    def __init__(self, width=50, height=50, initial_fraction_alive=0.2, rng=None)
        """Create a new playing area of (width, height) cells."""
        super().__init__(rng=rng)
        # Use a simple grid, where edges wrap around.
        self.grid = OrthogonalMooreGrid(
            (width, height), capacity=1, random=self.random, torus=True
        )

        # Place a cell at each location, with some initialized to
        # ALIVE and some to DEAD.
        for cell in self.grid.all_cells:
            Cell(
                self,
                cell,
                init_state=Cell.ALIVE
                if self.random.random() < initial_fraction_alive
                else Cell.DEAD,
            )

        self.running = True

    def step(self):
        """Perform the model step in two stages:
        - First, all cells assume their next state (whether they will be dead or
        - Then, all cells change state to their next state.
        """
        self.agents.do("determine_state")
        self.agents.do("assume_state")
```

GAMA vs Mesa Trade-offs

Consideration	GAMA	Mesa
Spatial modeling	Built-in A*, collision, GIS	Manual implementation
Disease models	Validated implementations	Build from Mesa-SIR or scratch
Python ML/inference	Bridge required	Native integration
Existing codebase	Rewrite needed	Direct extension
Team learning curve	GAML syntax	Already familiar
Long-term flexibility	Platform-dependent	Full Python ecosystem

Basically GAMA excels at the spatial simulation and visualisation aspect, Mesa excels at tool integration

NetLogo https://docs.netlogo.org/

Optimized for simpler educational models. Huge community, low barrier to entry.

- GUI components (sliders, buttons) for parameter exploration
- Agent inspectors for interactive debugging
- HubNet extension for networked/distributed simulations
- GIS extension available, RNetLogo for R integration

Limitations: Not designed for complex descriptive models, limited GIS compared to GAMA, simplified environment representation.

Repast

Designed for massive-scale simulations and brute-force parameter sweeps.

Strengths: Scalability across HPC clusters, multi-threaded scheduler, scikit-optimize integration.

Consideration: May be overkill for single-floor hospital simulations

Some notable crowd Simulation tools

Menge — Modular Crowd Simulation

Open-source framework implementing multiple pedestrian dynamics models:

- Continuum Crowds Model
- Social Force Model
- Model switching based on density/zone

Useful for high-traffic areas (ED waiting rooms, cafeterias).

CrowdNav — RL for Navigation

Reinforcement learning approach to crowd navigation. Aligns with inverse RL objectives:

- Learn staff movement patterns from trajectory data
- Policy inference for patient navigation behaviors

Disease Modelling - COMOKIT for COVID-19 and FRED

COMOKIT (COVID-19 Modeling Kit) https://comokit.org/docs/gettingStarted

Built on GAMA platform, similar architecture to HAI Simulator but for community-scale COVID-19 modeling.

- Individual agents with clinical status, agendas, activities
- Building-level environmental transmission
- Policy intervention testing (masking, lockdowns, vaccination)
- Multi-scale versions: micro (buildings), meso (neighborhoods), macro (cities)

Open source ++ but not best fit to our floor plan modelling

FRED (Framework for Reconstructing Epidemiological Dynamics)

University of Pittsburgh agent-based epidemic simulator.

- Synthetic populations from census data
- Geographic mobility patterns
- Used by CDC for pandemic planning
- Large-scale (millions of agents)

https://comokit.org/, https://github.com/COMOKIT/COMOKIT-Model

Parameter Inference

Forward: Parameters -> Simulation -> Outputs

Backward: Observed Outputs -> ??? -> Parameters

- Stochastic nature
- High-dimensional impact of parameters
- Expensive simulation runs

This problem is likely to equally impact on our chosen ecosystem

ABC: Trial and Error with Quality Filter

- Start with random selection of parameters
- Simulation run
- Comparison to observed data ex MSD
- Keep or discard, and we can use what fits to narrow down our selection of parameters: Sequential Monte Carlo ABC

Available in pyABC, ELFI, OpenMole (IslandABC())

sbi — Neural Simulation-Based Inference

sbi uses neural networks to learn the mapping from simulation outputs back to parameters

Training phase:

1. Sample random parameters
2. Run simulation for each outputs
3. Train neural network on (x, θ) pairs

Inference phase (use the trained model):

1. Feed observed data to trained network
2. Get posterior distribution over parameters

+Powerful to learn a distribution of plausible starting parameters from simulated outbreaks, learn what parameters are high-risk

https://sbi-dev.github.io/sbi/

Going back to Mesa - possible integration with sbi + Mesa

```python
from sbi.inference import SNPE
from sbi.utils import BoxUniform

# Define prior over parameters
prior = BoxUniform()

# Wrap Mesa model as simulator
def simulator(parameters):
    model = AMRHUBSimulation(parameters)
    model.run(100)
    return [model.stats]

# Train sbi network - posterior
inference = SNPE(prior)
theta = prior.sample((1000,))
x = [simulator(t) for t in theta]
posterior = inference.append_simulations(theta, x).train()

# Infer from the network
samples = posterior.sample((10000,), x=observed_data)
```

Integrating RL tools to Mesa ecosystem

mesa-gym (github.com/gsileno/mesa-gym)

Direct bridge between Mesa and Gymnasium:

- Mesa handles environment state and agent mechanics
- Gymnasium wraps it for RL algorithm compatibility

Mesa model ←→ mesa-gym wrapper ←→ Gymnasium env ←→ RL algorithm

For multi-agent scenarios: PettingZoo

- Multi-agent version of Gymnasium
- Handles turn-based and simultaneous agent interactions
- mesa-gym has PettingZoo integration on roadmap

Inverse RL

	sbi	Inverse RL
observation	Aggregate simulation outcome	Real-world action trajectory
infer	Model parameters	Reward - what agents are optimising for

We would use real-world trajectory data to basically train a reward function ex.

reward = -0.5 * distance_walked
-2.0 * time_in_crowded_corridor
+3.0 * checked_patient
-1.5 * entered_isolation_room_without_mask -> then we integrate this to the Agent logic in simulation

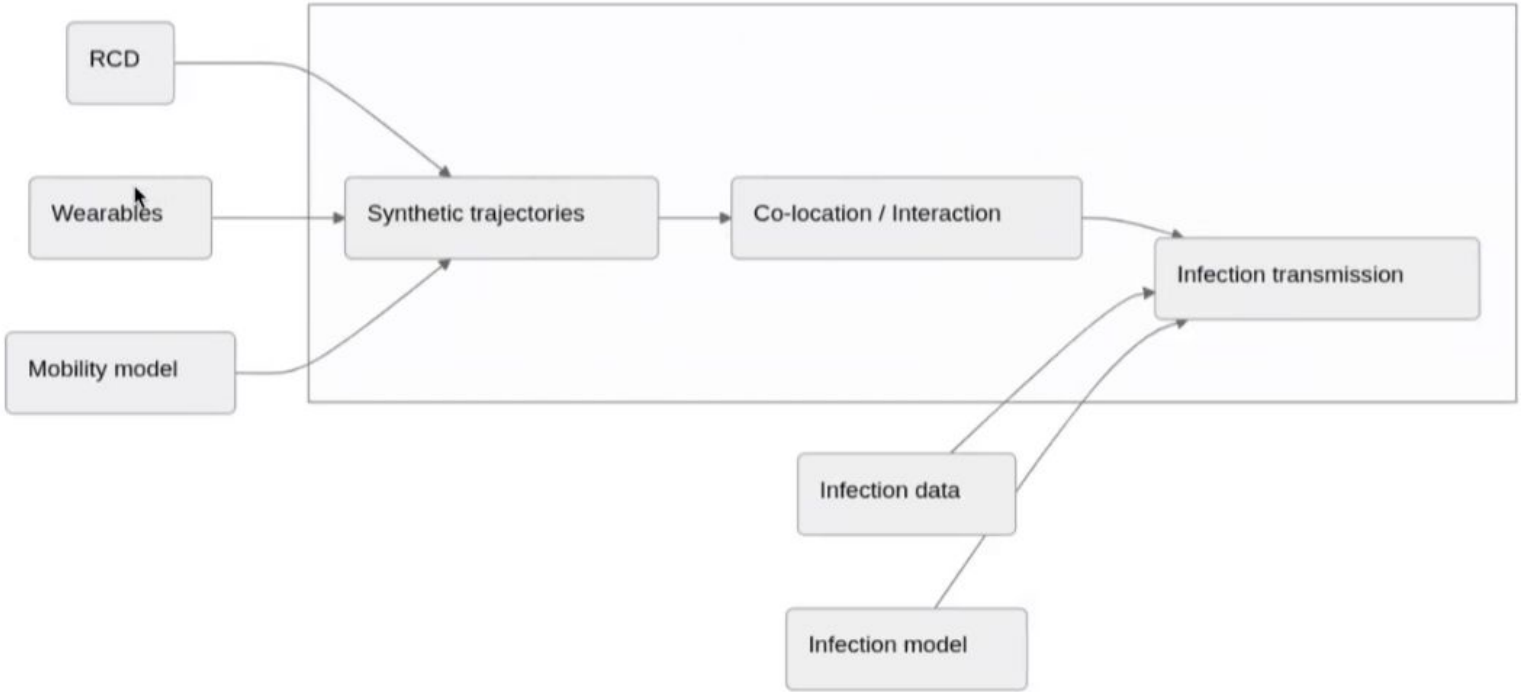
- Maximum Entropy IRL — find reward that makes observed trajectories most probable
- GAIL (Generative Adversarial Imitation Learning)
- Behavioral Cloning — simpler supervised learning (state → action), no reward inference
- https://github.com/HumanCompatibleAI/imitation -> Gym environment compatible, so Mesa integration possible via mesa-gym

Our data:

- Person ID, role, location, timestamp
- We are extracting trajectories of who went where, when, for how long

```
from imitation.algorithms import bc
import gymnasium as gym

# 1. Convert data to trajectories
trajectories = []
for Agent_type in data:
    extract_trajectory_for_Agent
    add_to_trajectories()

# 2. Wrap AMR model in Mesa as OpenAI Gym model
env = MesaGymWrapper(AMRModel())

# 3. Ex. Use behavioural cloning to learn a policy from movement
bc_trainer = bc.BC(
    action_space, observation_space, trajectories
)
bc_trainer.train()
learned_policy = bc_trainer.policy

# 4. Use learnt policy in Mesa simulation
AMRModel.agent.policy = learned_policy
```

Simulation under uncertainty - problem framing



Using sbi + Inverse RL together for prediction

Step 1 - Inverse RL

Using real-world trajectory data to create a realistic reward function for agents, we simulate realistic agent movement

Step 2 - sbi for disease modelling

We run simulations with realistic movement + varying disease parameters, and train a network that maps disease outcomes to disease parameters

Sensitivity Analysis

"Which parameters matter most?"

Vary parameters systematically, measure impact on outcomes.

Python tools:

- SALib — main library for sensitivity analysis
- OpenMOLE — distributed sensitivity analysis (works with GAMA, NetLogo, Python)

OpenMOLE

Platform that wraps the simulation and runs exploration/calibration workflows on it. It integrates directly with GAMA, NetLogo, R, and Python models.

Key capabilities:

- Parameter space exploration
- Sensitivity analysis (Morris, Sobol indices)
- ABC via IslandABC()
- Multi-objective optimization (genetic algorithms)

Possible Inverse RL + Inference roadmap

1. Step 1: Learn realistic movement (Inverse RL)

Input: RTLS trajectories from UCLH
Method: Inverse RL (imitation library)
Output: Movement policy that mimics real staff behavior

2. Step 2: Identify which parameters matter (Sensitivity Analysis)

Input: Mesa simulation batch parameter sweep with learned RL movement + varying disease parameters
Method: Sobol / Morris (SALib)
Output: "Transmission rate explains 60% of variance,
contact duration 25%, recovery rate 10%"

3. Step 3: ex. Test hypothesis ex. Identify where risks are

Input: Same simulation, track infection locations
Method: Aggregate across runs
Output: Heatmap of hotspots

4. Step 4: Test interventions

input: Simulation + proposed changes (ex room reassignments)
Method: Compare baseline vs intervention scenarios
Output: "Moving handwashing station reduces transmission 18%"

Temporal sequence modelling - hypotheses

- **H1: Behaviour** - Specific short-term behavioural sequences increase the instantaneous hazard of entering an outbreak.
- **H2: Hierarchy** - Seniority / role modulates the strength of pre-cursors.
- **H3: Connection** - Bursts of networked interactions precede an outbreak