Mesa-LLM: Generative Agent-Based Modeling with Large Language Models Empowered Agents

Boyu Wang
University at Buffalo

Colin Frisch
University at CentraleSupélec

Sanika Nair
Independent Researcher

Jackie Kazil
Decision Advantage

Andrew Crooks
University at Buffalo

2026 AAG Annual Meeting, San Francisco, CA

Agent-Based Modeling with Mesa in Python

- **Agent-based modeling (ABM)** simulates system-level outcomes from interactions among individual agents.
- It is useful when behavior, heterogeneity, and local interaction matter.
- **Mesa** is an open-source ABM framework in Python with strong fit to the PyData ecosystem.

3,473
GitHub stars
Mesa repository as of March 2026

41,881
monthly PyPI downloads
as of March 2026 on PyPIStats

Sources: GitHub API for `mesa/mesa` repository stars and PyPIStats recent downloads for package `mesa`.

Mesa 3: Agent-based modeling with Python in 2025

Ewout ter Hoeven [1], **Jan Kwakkel** [1], **Vincent Hess** [2], **Thomas Pike** [3], **Boyu Wang** [4], **rht** [5], **and Jackie Kazil** [3]

1 Delft University of Technology (Faculty of Technology, Policy and Management), the Netherlands **2** Independent Researcher, Germany **3** George Mason University (Department of Computational Social Science), United States **4** University at Buffalo (Department of Geography), United States **5** Independent Researcher

Summary

Mesa is an open-source Python framework for agent-based modeling (ABM) that enables researchers to create, analyze, and visualize agent-based simulations. Mesa provides a comprehensive set of tools and abstractions for modeling complex systems, with capabilities spanning from basic agent management to sophisticated representation of spaces where agents interact. First released in 2014 and published in Masad et al. (2015) (with updates published in Kazil et al. (2020)), this paper highlights advancements and presents Mesa in its current version (3.1.5) as of 2025.

JOSS paper: `10.21105/joss.07668`.

Generative Agent-Based Modeling

- **Generative ABM (GABM)** uses LLMs or related generative models inside agents for reasoning, communication, planning, and action selection.
- It has emerged over the last few years as LLMs became stronger at natural language-based decision support.
- This opens a new paradigm for modeling agent interactions (e.g., negotiation), memory, and adaptive behavior.

Classical ABM
rules, thresholds, probabilities

→

LLM Agents
reasoning, dialogue, memory

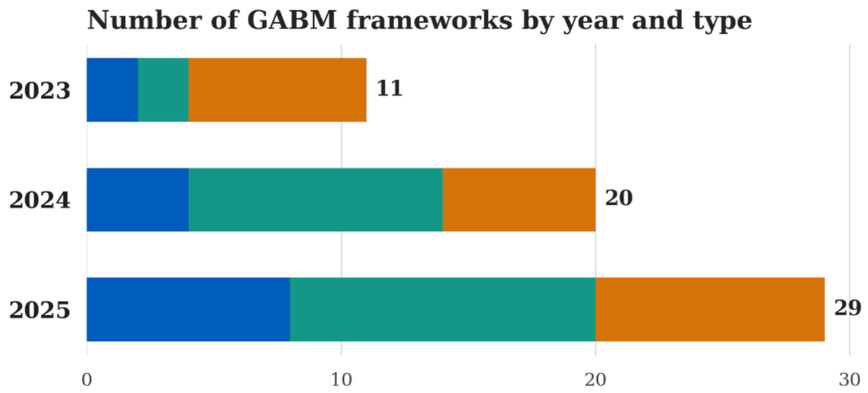
→

GABM
generative cognition inside simulation

Classical ABM Frameworks and LLM Integration

Framework	Language / ecosystem	First release	Latest release	License	Native LLM layer	Typical LLM integration path
NetLogo	NetLogo + JVM	1999	2025	GPL-2.0	No	Python extension (`py:*`) or Controlling API
Repast Simphony	Java / Groovy / ReLogo	2000	2024	New BSD	No	Java SDK or HTTP clients in agent code
AnyLogic	Java-based IDE	2000	2026	Commercial	No	Java customization + Alpyne / Cloud API
MASON	Java library	2003	2019	AFL-3.0	No	Custom Java SDK / HTTP integration
GAMA	GAML + Java	2009	2025	GPL-3.0	No	Extensions, `gama-server` , or external services
Mesa	Python / PyData	2015	2026	Apache-2.0	Yes	**`LLMAgent`** abstraction in Mesa-LLM

Emerging GABM Frameworks



Generic GABM frameworks
Concordia, AgentSociety, GenSim, YuLan-OneSim, SocioVerse, Agent-Kernel, ...

Domain-specific simulators / digital twins
UGI, OpenCity, LMAgent, TravelAgent, EconAgent, TwinMarket, ...

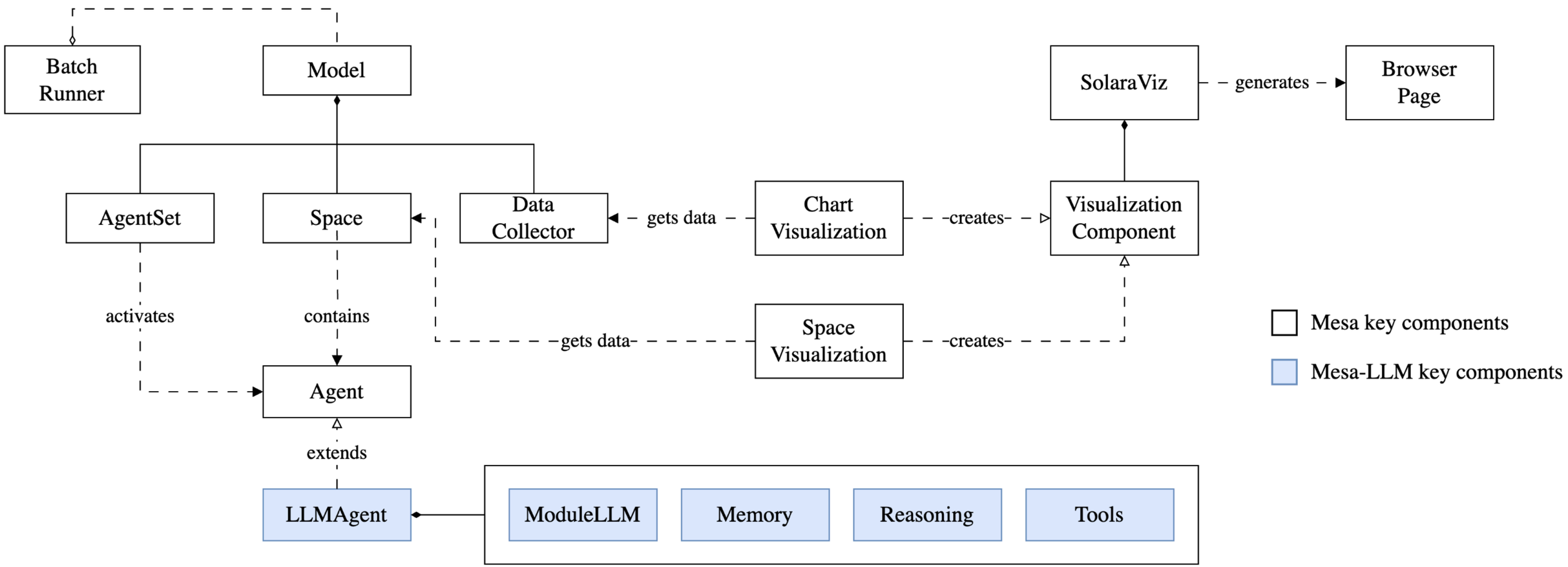
Agent orchestration / evaluation platforms
AgentVerse, AgentScope, SOTOPIA, AgentSims, SocioDojo, SOTOPIA-S4, ...

Framework	Core idea	Typical scale / environment
Concordia	Game-master-mediated generative social simulation library	Small-to-medium grounded scenarios in physical, social, or digital space
AgentSociety	Large-scale social simulator with LLM-driven agents	Explicit city/society-scale simulations with distributed execution
Agent-Kernel	Microkernel framework for adaptive social simulation	Dynamic populations and reusable society components, including large simulations
Mesa-LLM	Mesa-compatible modular cognition layer for ABMs	Existing Mesa models, from toy to moderate scale, with parallel stepping support

Mesa-LLM

- **Mesa-LLM** is a Mesa extension for LLM-powered agents rather than a replacement for Mesa itself.
- It keeps Mesa's core execution model and adds modular cognition:
  - reasoning (e.g., `ReAct`, `CoT`, `ReWOO`)
  - memory (e.g., `STLTMemory`, `EpisodicMemory`)
  - bounded action space (e.g., `move_one_step`)
  - recording and analysis of agent behavior
- The project's aim is to lower the friction for Mesa users to utilize LLMs in ABMs, and take advantage of vast amount of data used to train these models.

Mesa-LLM: Architecture



Mesa-LLM: Key Components

ModuleLLM	LLM agnostic: openai/gpt-4o, ollama/llama3.1, ...
Memory	Retains recent & salient context: STLTMemory, EpisodicMemory, ...
Reasoning	Turns observation & memory into plans: ReAct, CoT, ReWOO, ...
Tools	Bounds actions to executable functions: @tool decorator, ToolManager

Mesa-LLM: Typical Execution Flow

1

Mesa activation

`Model.step()`
`AgentSet.shuffle_do()`

Mesa still controls when each agent is asked to act.

→

2

Observation

`LLMAgent.generate_obs()`
self state + local state

The agent packages nearby context into prompt-ready inputs.

→

3

Planning

`Reasoning` + `ModuleLLM`

Memory, observation, and task prompt are combined into a structured plan.

→

4

Action

bounded action space

Language plans become explicit operations such as move, change state, or arrest.

→

5

Updates

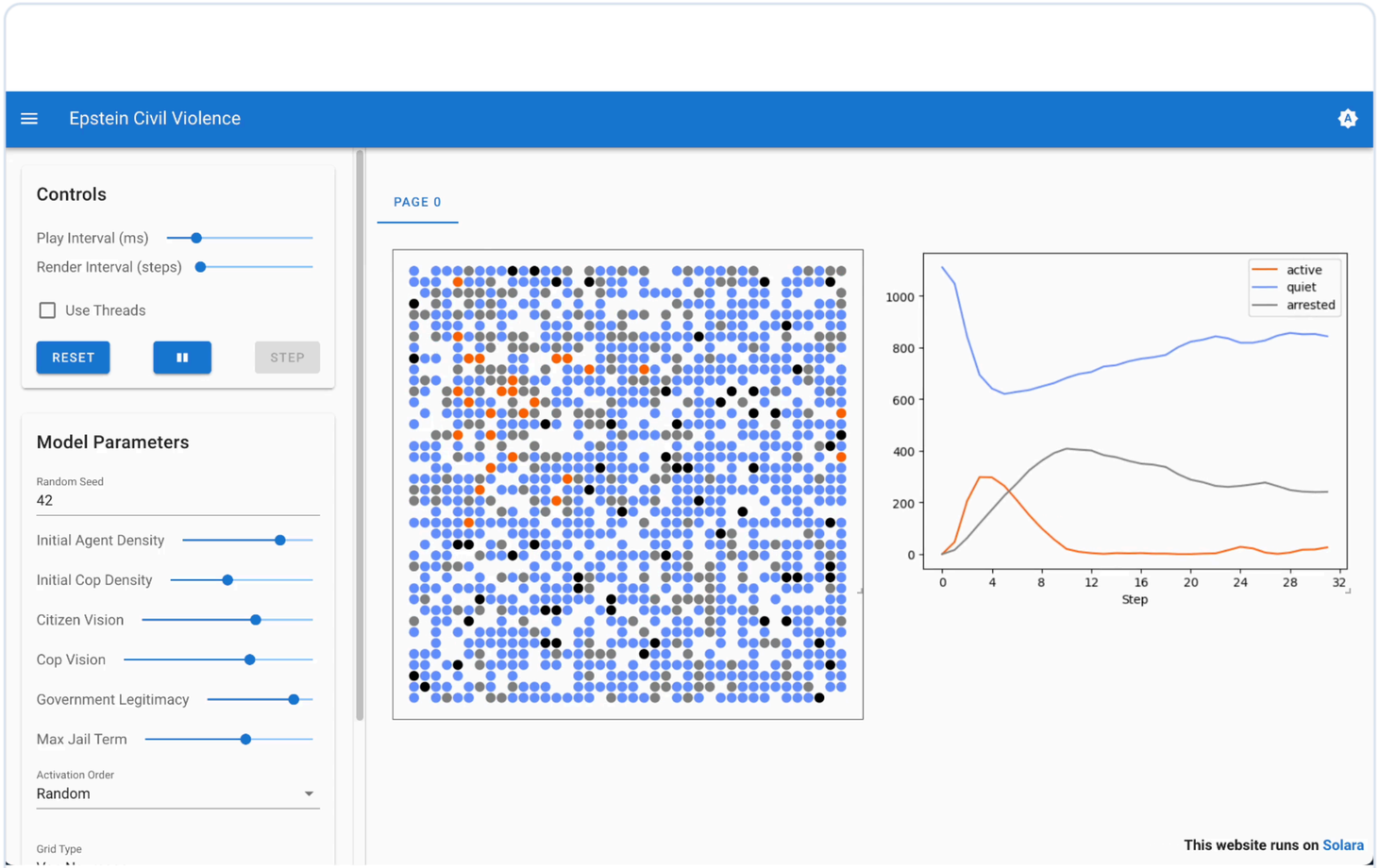
environment + agent state
memory + traces

The simulation records what changed and carries that context into the next step.

Key idea: the LLM decides within a bounded action space, while Mesa still governs scheduling, data collection, and visualization.

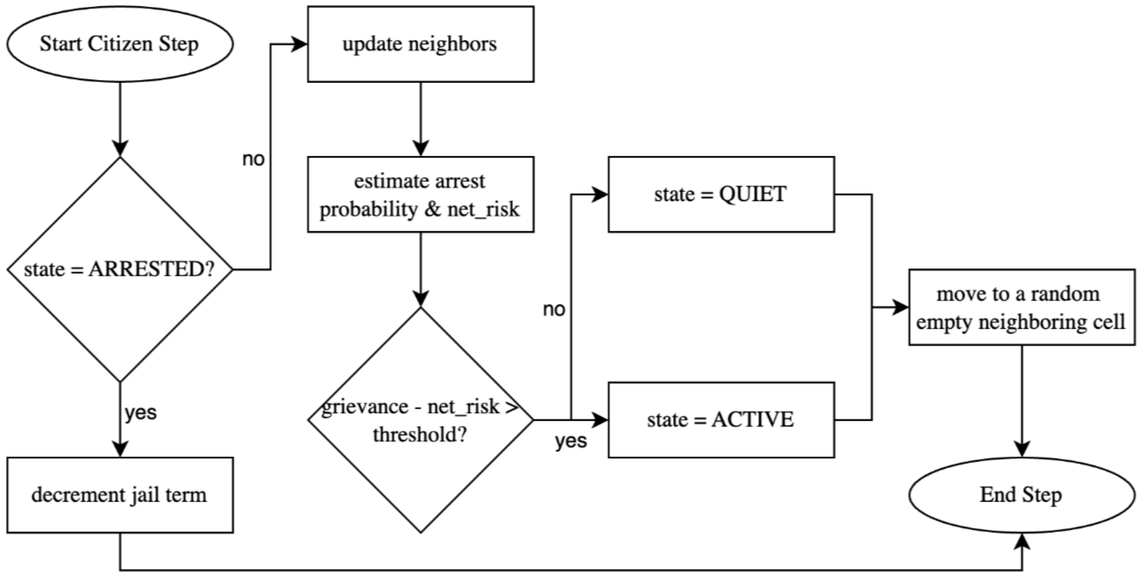
Example: Epstein Civil Violence (Classical)

- Spatial civil-violence ABM: citizens and cops occupy a grid and update behavior from local neighborhood conditions.
- Original paper: Epstein (2002)
- Mesa implementation: mesa/examples/advanced/epstein_civil_violence

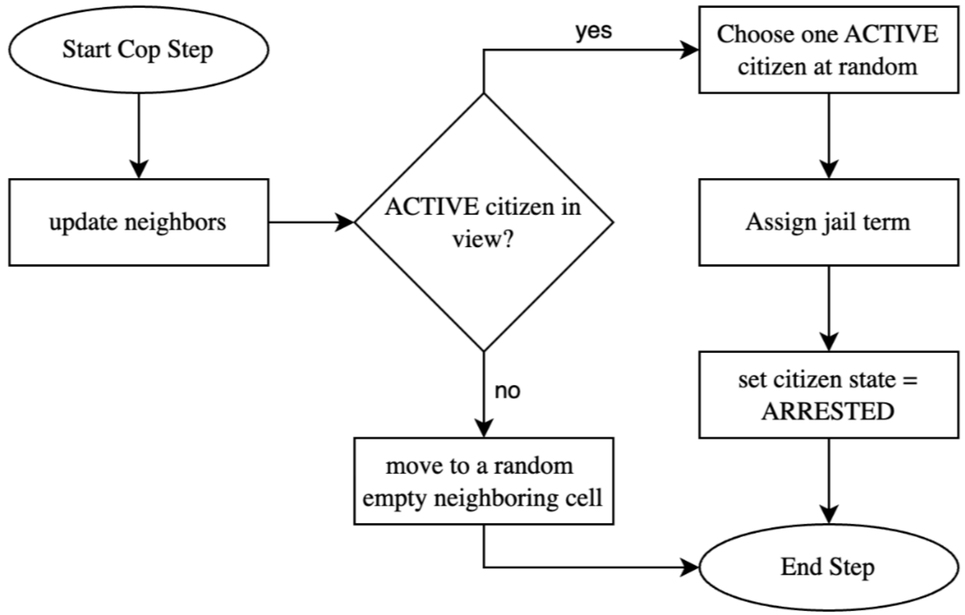


Example: Epstein Civil Violence (Classical)

`Citizen` agent



`Cop` agent



Example: Epstein Civil Violence (Generative)

- With Mesa-LLM, Citizens and cops become `LLMAgent` -based actors with role-specific actions.
- Source code available at github.com/mesa/mesa-llm/examples/epstein_civil_violence

`Citizen` agent prompt stack

Role text set on agent

`You are a citizen in a country that is experiencing civil violence ... You can move one step in a nearby cell or change your state.`

Memory context

`STLTMemory` : short-term memory + long-term summary, including prior observations and plans.

Current observation

`self_state` + `local_state` : grievance, risk aversion, current state, arrest probability, and nearby agents.

Step prompt

`Move around and change your state if the conditions indicate it.`

Possible actions

`change_state` , `move_one_step`

`Cop` agent prompt stack

Role text set on agent

`You are a cop in a country that is experiencing civil violence ... You can move one step in a nearby cell or arrest a citizen.`

Memory context

`STLTMemory` : short-term memory + long-term summary, including prior observations and plans.

Current observation

`self_state` + `local_state` : nearby citizens and whether any active target is in local view.

Step prompt

`Inspect your local vision and arrest a random active agent. Move if applicable.`

Possible actions

`move_one_step` , `arrest_citizen`

Example: Epstein Civil Violence (Generative)

`Citizen` agent decision trace

```
┌─ Step 1 | Citizen 10 ──────────────────────────────────────────────────────┐
[Observation]
   └── self_state :
      └── agent_unique_id : 10
      └── system_prompt : You are a citizen ... You can move
                          one step or change your state.
      └── location : (np.int64(7), np.int64(8))
      └── internal_state :
         └── risk_aversion = 0.0298
         └── state = CitizenState.QUIET
         └── arrest_probability = 0.0000
   └── local_state :
      └── Citizen 18 at (np.int64(2), np.int64(8))
      └── Citizen 19 at (np.int64(9), np.int64(3))
      └── ...

[Plan]
   └── reasoning : low risk aversion means I can
                   move and reassess the situation.
   └── action : move_one_step(direction='NorthEast')

[Action]
   └── name : move_one_step
   └── response : agent 10 moved to (np.int64(8), np.int64(9)).
```

`Cop` agent decision trace

```
┌─ Step 1 | Cop 3 ──────────────────────────────────────────────────────────┐
[Observation]
   └── self_state :
      └── agent_unique_id : 3
      └── system_prompt : You are a cop in a country that is
                          experiencing civil violence. You are
                          a member of the police force and your
                          job is to ... You can move one step
                          or arrest an ACTIVE citizen.
      └── location : (np.int64(6), np.int64(6))
      └── internal_state : []
   └── local_state :
      └── Citizen 13 at (np.int64(1), np.int64(8))
      └── Citizen 18 at (np.int64(2), np.int64(8))
      └── Citizen 19 at (np.int64(9), np.int64(3))
      └── ...

[Plan]
   └── reasoning : I inspect nearby citizens and look
                   for anyone marked active.
   └── action : arrest_citizen(citizen_id=1)

[Action]
   └── name : arrest_citizen
```

Example: Epstein Civil Violence (Generative)

Things to Try

Classical experiments

- **Movement on vs. off**

  Check whether mobility produces more frequent or sharper rebellion episodes, as Epstein reports.

- **Gradual erosion vs. sudden legitimacy drop**

  Compare slow decline in regime legitimacy to an abrupt shock and inspect how the timing of unrest changes.

- **Lower legitimacy vs. fewer cops**

  Separate grievance effects from enforcement effects by weakening the state in two different ways.

Mesa-LLM extensions to classical experiments

- **Nearby citizens can talk / persuade each other**

  Add rumor, encouragement, or fear so rebellion depends on communication as well as local counts.

- **Quiet citizens share memories**

  Let agents pass along stories about arrests or grievances and test whether shared memory shifts later activation.

- **Cops send messages and coordinate**

  Share sightings, warnings, or patrol intent to explore whether coordinated policing suppresses or redirects unrest.

Challenges in GABM

- LLM-based ABMs add new methodological burdens on top of classical ABM rather than removing them.
- Ongoing work spans **memory design**, **reasoning strategies**, **efficiency**, among many others.
- Mesa-LLM aims to provide a reusable, generic framework so researchers can focus on the scientific question rather than rebuilding LLM-agent components from scratch.

Validation

Stochastic trajectories require repeated-run interpretation rather than single-run anecdotes.

Efficiency

Cost, latency, and scalability matter quickly once agent populations or run counts increase.

Fidelity

Model choice, prompt design, and tool definitions can shift behavior through bias, hallucination, or persona drift.

Reproducibility

Guardrails include prompt/model version logging, repeated-run comparisons, and archived traces or recordings.

Opportunities: From artificial landscapes to GIS-backed GABMs

In the Sugarscape model, the environment can be **loaded as data**. It is possible to be swapped from a toy surface to a GIS raster.

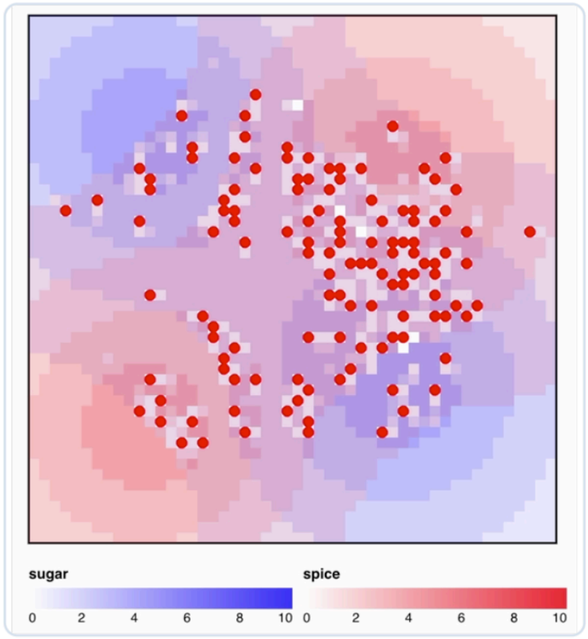
Toy map
`sugar-map.txt`
defines the resource surface.

↓

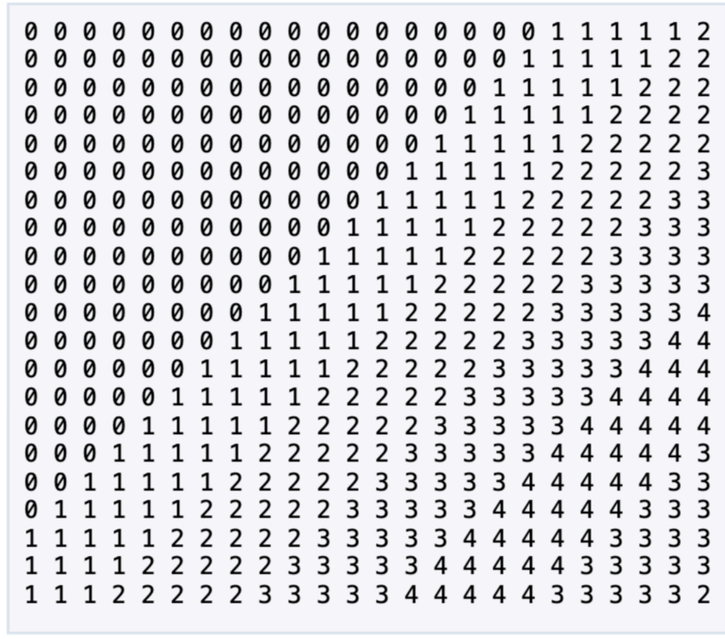
GIS raster
Land cover, accessibility, hazard, suitability, elevation.

+

LLM traders
Agents reason over inventory and nearby resources, then decide whether to move and trade.



Sugarscape with traders model in Mesa



Imported landscape: `sugar-map.txt`

A Mesa-Geo raster layer could fill the same input slot instead of a synthetic text map.

Source code

- classical sugarscape: mesa/examples/advanced/sugarscape_g1mt
- generative sugarscape: mesa-llm/examples/sugarscrap_g1mt

Acknowledgements

- Mesa-LLM is a community effort. Special thanks to the Project Mesa community for their contributions and support.
- Mesa-LLM started as a Google Summer of Code (GSoC) project in 2025. Mesa is participating again in the GSoC 2026 cycle, and contributor applications are welcome!
  - Application opens on `March 16, 2026` and closes on `March 31, 2026`.
  - 👉 Apply via summerofcode.withgoogle.com

Summary

- Mesa-LLM preserves Mesa's simulation backbone rather than replacing the classical ABM workflow.
- It provides reusable building blocks so researchers can focus on the scientific questions rather than re-implementing LLM-agent infrastructure.

Join Us

Matrix Chat

Real-time community chat

`matrix.to/#/#mesa-llm:matrix.org`

Discussions

Questions, ideas, and design feedback

`github.com/mesa/mesa-llm/discussions`

Repository

Code, examples, and documentation

`github.com/mesa/mesa-llm`

Read the Docs

API docs, tutorials, and usage guides

`mesa-llm.readthedocs.io`