Agent-Based Modeling and Simulation - Feature Spaces, Macroscopic Statistical Analysis, and Specification Fidelity

Luis Carlos Lara-Lopez
School of Computing, Costa Rica Institute of Technology,
Escazú & Cartago, Costa Rica ,
l3clara@gmail.com

Ignacio Trejos-Zelaya
School of Computing, Costa Rica Institute of Technology &
School of Software Engineering, Universidad CENFOTEC,
Cartago & San José, Costa Rica ,
itrejos@{ucenfotec.ac.cr, tec.ac.cr}

Santiago Nunez-Corrales
NCSA/IQUIST, UIUC
Urbana IL, United States,
nunezco2@illinois.edu

Jose Helo-Guzman
School of Computing, Costa Rica Institute of Technology,
Cartago, Costa Rica ,
jhelo@tec.ac.cr

Abstract

Agent-Based Models (ABMs) are widely used to simulate complex systems through emergent behavior. Agent Based Modeling practice is constrained by the lack of systematic methods to evaluate and select frameworks for a given problem, and by the absence of protocols to study differences across computational implementations. We propose a Feature Space Maturity Model (FSMM) to evaluate how well a framework matches the requirements of a modeling task. We present results of four canonical ABM experiments to demonstrate the FSMM applied to five well-known frameworks. Our results suggest that combining the FSMM with the proposed protocol provides valuable information for both framework builders and modelers facing complex technical and research choices.
Ensuring ABM reliability requires specification fidelity - meaning both the conceptual correctness of the model and its faithful implementation. We study implementation fidelity by analyzing statistical similarity across multiple frameworks. We apply a macroscopic statistical approach to evaluate whether different ABM frameworks yield functionally equivalent results when executing the same specification. We designed a Pareto-based experiment and implemented it in six ABM frameworks. We defined three macroscopic observables and analyzed 10,000 simulation runs per framework using coefficient of variation, Fréchet distance, ANOVA, and F-tests. Five of the six frameworks show statistically indistinguishable behavior, while one exhibits consistent deviations. The results highlight the value of statistical validation to identify implementation inconsistencies and optimize resource allocation in large-scale simulations.

Keywords: agent-based modeling, ABM simulation, statistical indistinguishability, ABM framework, feature space, maturity model

1 Introduction

The effectiveness of agent-based models (ABMs) in complexity science [1], particularly in generative social science [2], lies in their capacity to simulate emergent behavior through *bottom-up* abstraction [3]. In contrast, top-down simulation approaches - such as dynamical systems or discrete-event models -require extensive parametrization and tend to obscure the relationship between micro- and macro-level dynamics [4].

Agent-Based modeling starts with the conceptualization of agents and their interactions; the models are *agentized* to better capture emergent phenomena [5–7]. The model's fidelity to reality [8] depends critically on the selected features. While perfect accuracy is elusive, ABMs aim to replicate system archetypes through agent behaviors and event dynamics. Ideally, simulation outcomes should be independent of the implementation details [9].

To ensure reliability, ABMs must scale to real-world systems and be executed multiple times to capture average behaviors. This paper uses a *Feature Space Maturity Model* (FSMM), a scoring model for ABM framework features.

Section 2 provides a background on Agent Based Modeling. Section 3 introduces our Feature Space Maturity Model. Section 4 presents a set of agent-based modeling problem exemplars. Section 5 briefly describes the ABM frameworks chosen for experimentation. Section 6 shows and discusses the scores obtained by the ABM frameworks. Section 7 explains the case for validating ABM framework's fidelity to reality via statistical similarity. Section 8 sets up a controlled, specification-driven, ABM experiment whose purpose is to test whether different ABM frameworks behave in a statistically equivalent way under identical specifications. Section 9 recalls the frameworks under evaluation. Section 10 presents the experiments' results. Section 11 offers macroscopic statistical analysis and its relation to specification fidelity. Section 12 discusses the experimentation and simulation experience. Section 13 concludes, and future work is covered in Section 14.

The main contributions of this article are: (1) the definition of the FSMM for evaluating ABM frameworks, (2) the application of the maturity model to five widely used frameworks, (3) the definition of a macroscopic statistical protocol for implementation-fidelity analysis, (4) the validation of six independent implementations of the same specification, and (5) empirical evidence that statistical equivalence can be used as an indicator of specification fidelity in ABM simulations.

In addition to the above, the FSMM motivated the design of a reference architecture for ABM frameworks, which was validated via the AB-X Proof-Of-Concept (POC) implementation and subjected to the same experiments as the pre-existing ABM frameworks. The reference architecture and AB-X's design and implementation are not reported in this article. The interested reader may want to consult [10]. All source code related to this research is publicly and freely available [1].

2 ABM background

The use of ABM is heterogeneous. In some cases, designing various types of agents aims to validate the future operation of physical or virtual agents [11]. In other cases, the goal is to develop simulations that improve understanding of global or emergent phenomena in complex adaptive systems [12]. This distinction shows the need to identify key ABM concepts to evaluate current frameworks and propose a new reference architecture as a baseline for future implementations.

2.1 The need for ABM frameworks

Well-established models such as the Game of Life can be coded in almost any programming language [13], but most collective behavior models are too complex to implement from scratch. To address this, many ABMs have evolved to share a common feature space for general-purpose modeling. An early example is the Simula 67 programming language, developed in the mid-1960s and widely used for research by the 1970s. It was the first language to automate step-by-step agent simulations [14] and included features such as objects, classes, and inheritance. However, it was a *programming language*, not a framework, and had important limitations for non-trivial models.

ABM frameworks aim to reduce the mismatch between model specifications and programming language capabilities. Over the years, many ABM frameworks have been developed, ranging from lightweight frameworks with limited simulation capabilities to sophisticated platforms with custom languages for simulation. A broad but shallow comparison of ABM frameworks was presented by Sameera Abar et al. in 2017 [15]. Despite the variety of ABM frameworks available today, there is no common formal specification of the ABM feature space. This lack of standardization makes evaluation difficult. To address this we have developed the FSMM. We expect frameworks ranked higher on our FSMM to support more robust ABM development

[1] gitlab.com/msc_tesis/

and greater fidelity to reality. In addition, the FSMM could serve as a baseline for future framework specifications. The next section lists the most critical elements of the feature space for ABM frameworks identified in our preparatory work.

2.2 Counterfactuals

Counterfactuals provide baselines to evaluate the possibility and impact of alternative explanations, represented as collections -or *ensembles*- of event timelines [16]. Counterfactual support means an ABM framework can generate and compare event timelines under different input parameters (e.g., treatments vs. control) across hypothetical scenarios [17]. It also includes cases where a model branches into different timelines from an initial configuration.

2.3 Framework limitations

We hypothesize that the absence of a rigorous evaluation model for ABM frameworks leads to substantial deficiencies in their implementations. This likely stems from the lack of adequate evaluation parameters (and ideally a reference implementation) to capture fidelity, quality, and utility of the resulting software artifacts. As a result, a perceived lack of capabilities often reappears, driving the specification and implementation of new ABM frameworks from scratch.

Maturity models in computing usually define six maturity levels per focus area [18]. Some of these (e.g., *repeatable*, *managed*) apply to organizations rather than specific software systems. Our FSMM instead follows the facilitator-based strategy proposed by Nuñez-Corrales and Gasser [19], which defines ABM framework maturity as a gradient of technical facilitation with four levels: (0) no facilitators, (1) specific facilitators with high technical involvement, (2) general facilitators with low technical involvement, and (3) facilitators with no technical involvement.

3 Feature Space Maturity Model

The key components of the FSMM are presented in three main subsections: *simulation features*, *interrogation capabilities*, and *research support*. These categories were selected because they capture the core functionalities that ABM frameworks should provide to researchers, encompassing model execution, analysis, and scientific workflow support. To the best of our knowledge, organizing and evaluating ABM frameworks according to these three dimensions constitutes a novel approach that provides a comprehensive yet structured perspective on framework capabilities. Although these features sometimes overlap, redundancy is kept to a minimum.

3.1 Simulation Features

The main purpose of an ABM framework is to allow users to build simulations in a robust and efficient way. The features described below define capabilities that support clear model implementation and high fidelity to reality. We selected five features in the simulation: *usability*, *basic framework functionality*, *locality*, *agent management*, and *force field expressiveness*.

3.1.1 Usability

Usability reflects how easily an average user can install and use the framework in the current development environment (i.e. current Operating System). Although it is not formally part of the *simulation* category, it was included because of its significant impact on overall use. Following Nuñez-Corrales and Gasser's facilitator-based strategy, the four levels of usability technical facilitation are defined thus:

0. No way to install. User must compile from source code. Very limited or no documentation.
1. Difficult to install, major stability issues. Basic documentation.
2. Generally stable with some minor issues. Useful and well-written documentation.
3. Stable and easy to install. Documentation covers all features and includes several working examples.

In this article, *usability* is used in the restricted sense introduced above. More general treatments of usability in software systems can be found in the Software Quality knowledge area in the Software Engineering Body of Knowledge [20]. An international standard on software product's quality characteristics, including usability as one category, is described in ISO/IEC 25010:2023 [21].

3.1.2 Basic framework functionality

This dimension qualifies the completeness of the framework's feature set relative to potential use cases. It includes data processing, communication with external tools, and model validation against a specific problem formulation [22]. It also covers debugging tools such as visualization of control flow across a simulation, breakpoints, and conditional code execution. All these features improve model accuracy [23]. The four levels of functionality technical facilitation are:

0. No standard framework functionalities. No support for random distributions. No ability to save simulation status. No support to read or write structured data files (e.g., CSV, JSON, TOML). No ability to stop simulation.
1. Ability to read/write files and handle structured data.
2. Ability to communicate with other tools for sending and receiving data, e.g., http/https support.
3. Ability to run a simulation conditionally with a truth value or by stepping. Ability to pause simulation to interrogate agents or their environment.

3.1.3 Locality

Many models of interest (e.g., geospatial) involve agents that need awareness of location, direction, and perspective in their environment. Spaces may be continuous or discrete, and described by a geometry and topology (e.g., planar, cylindrical, toroidal). An agent can occupy only one point or cell at a time, usually its center. In most ABM models, agents can modify only their visible (or reachable) surroundings, but non-local effects may be needed to capture information-driven collective behavior. Agents may also face inaccessible regions, useful to model exclusion conditions that are predefined or emergent. The four levels of locality technical facilitation are:

0. All locality features must be implemented from scratch. No directions, no definition of agent neighborhood, and no support for spaces of any kind.
1. Basic locality features (e.g., agent position, simple spaces, simple topologies).
2. Support for discrete (i.e., grid-based) and continuous simulation spaces.
3. Rich locality and spatial features. Discrete and continuous spaces of any geometry and topology. Moore and von Neumann neighborhoods, as well as their extended versions [24, 25].

3.1.4 Agent Management

ABM frameworks must let users manage agents programmatically, including their creation, evolution, destruction, and housekeeping. Housekeeping refers to the set of maintenance operations required throughout an agent's lifecycle, such as releasing virtual and physical resources allocated to agents, cleaning temporary data structures, updating internal references, removing obsolete state information, and ensuring that memory and computational resources are used efficiently. These capabilities are essential for maintaining simulation correctness, scalability, and performance, particularly in large-scale models involving substantial numbers of agents and complex interactions. The four levels of agent management technical facilitation are:

0. No agent construct. Agents must be implemented from scratch.
1. Basic agents available, but extra structures or logic are needed to manage them. Agents must be added to the scheduler and detailed logic is required to move them.
2. Full agent lifecycle management. Selection and filtering by agent types and locality are possible. Move actions proceed without additional structures or information.
3. Agents have internal communication structures (e.g., a mail-box [26]) coordinated with their internal time representation and scheduling, with peer-to-peer and collective primitives.

3.1.5 Force fields

Although uncommon in most ABM frameworks due to their specific uses [27], force fields are a powerful tool to capture complex interactions. They may arise from dynamical properties of the space agents inhabit or from agent actions that propagate across space. The four levels of forced fields technical facilitation are:

0. No force fields. Additional data structures and algorithms required.
1. Basic force field support. Ability to define a point of emission and calculate distance and angle to it from a location in the space.
2. Agents and spaces act upon force fields. Forces have type and intensity and decay with distance. Multiple force types and emission points are possible. Point values and statistical properties of one or more forces are measurable at a location. The force range follows the geometry and topology of the space.
3. Ability to visualize forces at different levels of detail. Ability to show or hide forces by type and layer.

3.2 Interrogation Capabilities

The purpose of building models is to gain scientific insight by examining the consequences of their computation. Interrogation capability is therefore the second most important criterion in our FSMM, with an emphasis on *emergent behavior* [28].

3.2.1 Visualization

Visualization makes ABM results easier to interpret, which is essential given the complexity of the phenomena and the large amounts of data produced. It is also a useful tool for debugging. The four levels of visualization technical facilitation are:

0. No visualization support. Tools must be built using third-party libraries.
1. Limited visualization tools. Usually slow, poorly designed, but functional.
2. Fast, well-documented visualization. Option to disable visualization in favor of batch execution.
3. Comprehensive and well-documented visualization. Ability to visualize both detailed and statistical data, and to define layers by agent type, items, and force field attributes.

3.2.2 Model-level statistics

In a single simulation scenario, domain knowledge questions are usually answered through observables produced after model execution. ABM frameworks support this by tracking, storing, and exposing data interfaces for individual agents, their attributes, and the entire space when needed. The four levels of model-level statistics technical facilitation are:

0. No statistical tools. Implementation entirely bespoke using third-party tools.
1. Basic support for statistical analysis. Main expected metrics: average, moving average, standard deviation, maximum, minimum, 95 and 99 percentiles. Basic ability to produce charts.
2. Well-documented chart interface (e.g., ability to define title, axis data, and min/max for grid).
3. Support for data streams, online analysis, and advanced statistical functions.

3.2.3 Macroscopic support

A key application of ABM models is understanding emergent behavior. We define emergence as the rise of surprising behavior in systems that cannot be analytically deduced or predicted from the properties of their constituents [29]. In our case, *macroscopic support* is the ability to compute metrics for ensembles of agents using statistical methods for analysis and storage, and possibly for comparisons across ensembles. One application is identifying when model implementations deviate from specifications. Detecting and correcting these deviations should improve overall fidelity to reality. Macroscopic statistical analysis also includes standard tests between two or more simulations [30]. The four levels of macroscopic support facilitation are:

0. Single execution only. Statistics computed for a single simulation scenario.

1. Execution and comparison of two scenarios at a time across macroscopic observables in real time over resulting data.

2. Orchestration of multiple simulations on different runners. Comparison of statistical behavior of macroscopic observables using various statistics.

3. Tools to collect, reconcile, and statistically analyze macroscopic observables across different simulations. Results stored in files and created by implementations in possibly different frameworks.

3.2.4 Nearly decomposable system analysis

Partitioning an ABM simulation into dynamical regions based on interaction density is essential for analyzing complex collective behavior [31]. Tools for nearly decomposable system analysis support hypothesis testing at the system level. The four levels of this analysis technical facilitation are:

0. No concept of nearly decomposable systems. Identification of high internal and low external interaction must be implemented manually.

1. Ability to identify regions with many physically close agents in any space type (torus, plane, tube).

2. Ability to identify highly interacting agents through messages or events in their containing space.

3. Ability to visualize decomposable partitions based on interaction density.

3.3 Research support

Research support refers to the ability of a platform to facilitate hypothesis testing. While many ABM studies are exploratory, hypothesis testing advances the field by enabling computational instantiation of the scientific method [32, 33]. We selected three representative features: *graph support*, *counterfactual support*, and *utility features*. To our knowledge, these remain lacking in ABM frameworks; improving them would benefit ABM research by reducing the semantic gap between model specifications, implementations, and the hypotheses they aim to test. Due to the novelty and complexity of experiments requiring these features, they were not used as scoring topics but are briefly described here as a basis for further research.

3.3.1 Graph (network) support

ABM models include relations between agents through events localized in time and space. Graph theory therefore provides valuable tools to analyze simulations as dynamical networks [34]. Useful metrics include centrality, clustering, and path lengths, which give insights into the system's dynamics [35]. The four levels of graph/network support technical facilitation are:

0. No support for graphs, neither algorithmically nor in visualization.

1. Basic graph capabilities (e.g., weighted graphs, path finding); ability to visualize graphs and relations as arrows or lines.

2. Ability to assign agents to graph edges as well as to the grid. Ability to define graph edges as curves and set the percentage of the edge covered by an agent.

3. Environment with multi-grid/multi-graph support. Edges can move agents between grids that can be stacked to simulate building floors, sections of space stations, etc. Agents can belong to multiple graphs simultaneously.

3.3.2 Utility features

Translating concepts and measurements from the scientific problem into experimental data defines commensurability between models and phenomena [36]. In practice, this link requires extra work that creates friction during ABM implementation. Providing a way to implement experiments closer to the specification should improve fidelity to reality and research productivity. The four levels of research utility technical facilitation are:

0. No research support features. Translation from specification values to implementation outcomes is manual.

1. Basic support for units of measurement (e.g., time scales, distances, intensities). Basic unit conversion arithmetic.
2. Ability to define and query resources using set theory and first-order predicate logic. Ability to capture complex behavior compactly.
3. Ability to define environments with several grid layers. Ability to mark cells as inaccessible for movement or information access. Ability to specify arbitrary grid topologies (e.g., Voronoi sets [37]).

3.3.3 Counterfactual support

Establishing causality, a key goal in complex systems research, requires testing hypotheses across primary and alternative scenarios [38, 39]. Counterfactual analysis is relatively new to ABM outside policy research [40], and its support in frameworks is limited. In practice, it requires defining multiple scenarios with the same model or generating new simulations from one simulation by branching through new parameter sets. The risk of combinatorial explosion—when branching logic creates too many possible worlds—must be controlled by a framework setting, such as a hard limit on concurrent branches. The facilitation levels for counterfactual support are:

0. Single model run for the entire simulation. Model branching must be implemented from scratch or is infeasible.
1. Simulation checkpointing available (i.e., stop, reload, restart).
2. Ability to orchestrate concurrent model executions with their initial parameters across runners.
3. Rule-triggered model branching. Ability to manage and orchestrate executions in new runners. Ability to remove macroscopically similar executions.

4 Target ABM simulations

During the reported research, four experiments were performed to help observe each ABM frameworks's features and behavior. Each experiment covers different combinations of features used in Agent Based Modeling. The experiments are: Sugarscape, Pred-Prey, Pareto Principle, and the Covid propagation model. Our selection of ABMs is informed by well-established literature in the field [15], their demonstrated significance across multiple domains, and the relationship between complexity of the systems being modeled, their ABM descriptions, and the informativeness exhibited by the simulations' results.

4.1 Sugarscape

Sugarscape [41] is a family of agent-based models designed to study how individual interactions within a resource-constrained environment can give rise to complex economic, social, and demographic phenomena. Across the different Sugarscape variants, the central element is a population of autonomous agents that compete for resources while interacting through mechanisms such as movement, reproduction, trade, inheritance, disease transmission, and migration. The particular set of behaviors included depends on the specific model variant being studied.

Figure 1 illustrates the implementation used in this work. The model consists of a 150×200 toroidal grid populated by 150 agents of two equally distributed sexes, randomly placed at initialization. Agents possess individual characteristics, including energy reserves, metabolism, vision, and reproductive attributes. During each simulation step, agents consume sugar according to their metabolic requirements, move toward visible resources or potential mates within their line of sight, reproduce when compatible partners occupy the same location, and die when their energy is depleted or when they reach natural life limits. Offspring inherit half of each parental attribute while receiving the same initial energy level as the mother. Resource availability and consumption drive the population dynamics, making this configuration a representative example of the broader Sugarscape framework.

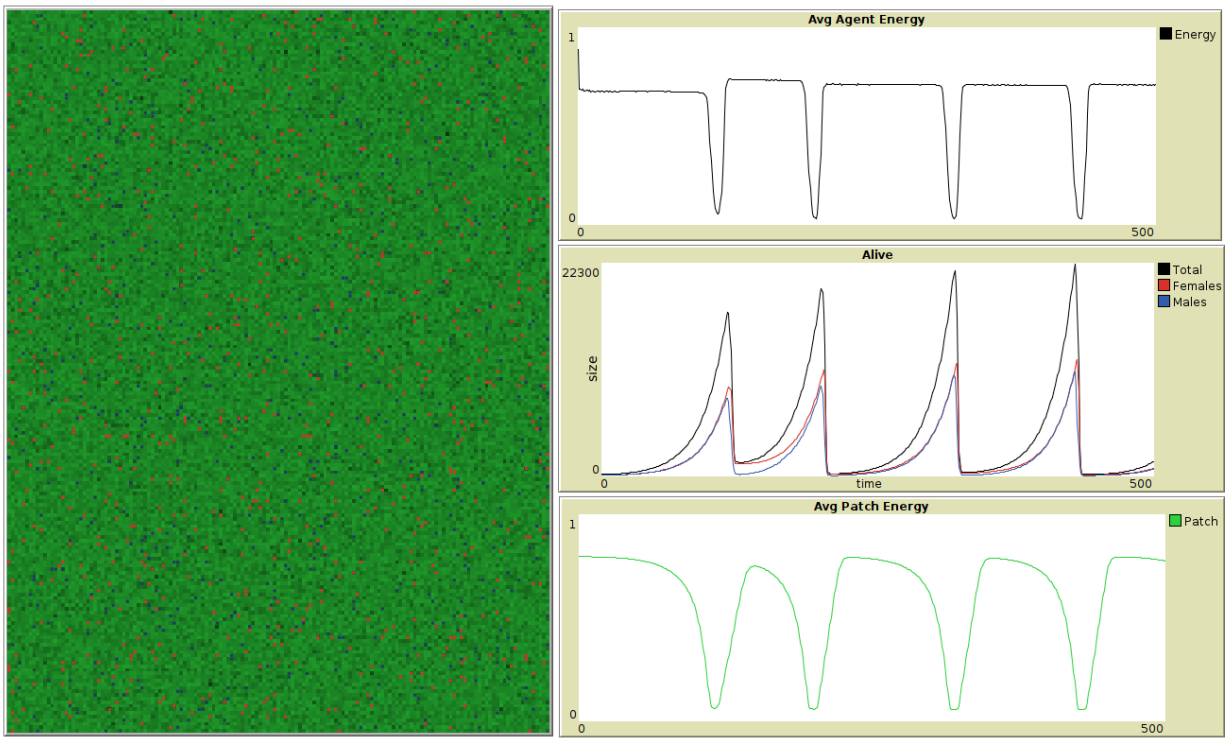

Figure 1: Sugarscape implemented using NetLogo.

4.2 Predator-Prey Dynamics

The Lotka–Volterra -or *predator-prey* model- abstracts population dynamics resulting from the coupling of two interacting species, a prey and a predator [42]. This model and its limitations when applied to real biological systems has been widely studied; we invite the reader to consult the extensive literature around it [43]. In short, the corresponding equations -assuming r stands for rabbits and w for wolves- are

$$\frac{dr}{dt} = \alpha r - \beta rw$$
$$\frac{dw}{dt} = -\gamma w + \delta rw$$

in which α and γ control for intra-species growth in predators and prey, while β and δ account for the effect of their interactions on reproductive rates. When agentized, continuous behavior of the system no longer hold due to the introduction of randomness in the interactions. Simulating these is computationally expensive due to pseudo-random number generation and not necessarily informative about the microscopic behavior of the system or the specifics of its interactions. Hence, this motivates the use of ABM simulation to gain a deeper understanding of both phenomena. Figure 2 illustrates an instance of the predator-prey model implemented in Mason.

Since there is no canonical predator-prey experiment, as the underlying dynamics can be modeled in multiple ways. For this work, we developed a custom version of the model. The model consists of a 500×500 toroidal grid populated by 25 wolves and 250 rabbits. Both wolves and rabbits reproduce asexually. Rabbits reproduce every 15 simulation rounds, whereas wolves reproduce following a successful hunt. Multiple agents may occupy the same grid cell, as no overlap restrictions are enforced. Agent movement is driven by forces generated exclusively by entities located within their respective fields of view.

Rabbits have four forces, some of which are proportional to the respective Euclidean distances. First, a random directional force with intensity varying from 0 to 1, and spatially ranging 40^{o} from 22.5^{o} left to 22.5^{o} right. Second, a flock force that groups rabbits together which grows linearly with the distance to each rabbit with initial intensity of 1. Third, a personal space force that repels a rabbit from its peers, which falls with distance with initial intensity of 40. Finally, rabbits experience a fear force repelling them from predators with intensity is 200 and no decay; it becomes 0 with no wolf in sight.

Complementary, wolves exhibit three different behaviors. With no rabbit in sight, the wolf will walk their maximum distance in random direction in search for rabbits. When one or more rabbits are in sight

but not within walking distance, the wolf will walk in direction to the closest rabbit. When there is a rabbit in walking distance, the wolf will go next to the rabbit and eat it. The rabbit dies and the time to live for the wolf goes back to the maximum of 10 turns.

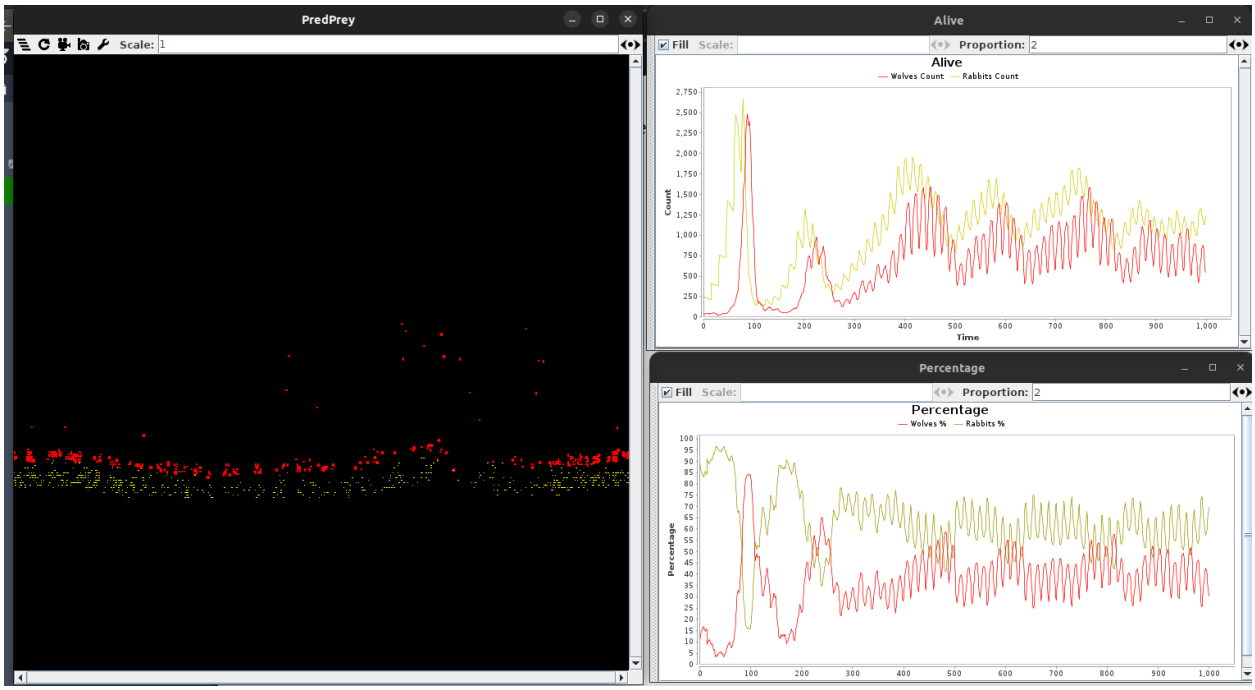

Figure 2: Predator-prey implemented using Mason.

4.3 Economic Wealth Inequality

Modeling the dynamics responsible for wealth distribution is a fundamental question in economics [44] with implications for policy development worldwide [45]. Observations by Pareto on land ownership in Italy led to the formulation of the 80/20 principle [46]. Succinctly, the Pareto principle describes situations in which a small share of the causes explains a dominant portion of the observations. Empirical investigation [47] has resulted in various models with equivalent explanatory power.

For the purposes of our work, we chose a variation of preferential attachment as the underlying generating process of the observed power law distribution. Shortly, for N agents with wealth K_i, we expect their wealth to grow as a function of a power law exponent α and an accumulation bias γ,

$$\frac{dK_i}{dt} = \alpha K_i^{\gamma}.$$

Assuming a constant increase in wealth c during each interaction, we can discretize the model assuming preferential Barabàsi attachment [48] as

$$K_i(t+1) = K_i(t+1) + c \cdot \frac{K_i(t)}{\sum_j K_j(t)}.$$

In our model, $K_i(t)$ is equivalent to exclusive tile ownership mediated by rent paid received across a grid of 100x100 tiles and $N = 100$; the number of agents remains fixed. Tiles have different (random) value, and rent exchange occurs as a result of agents entering a tile these do not own. The simulation starts with $K_i(0) = 0$ with an additional collection $c' = 75$ per time step. Rent is a random value r between 1 and 50 with tile purchase cost of $10r$. If an agent can afford to buy a tile it enters, it will do so. Otherwise, the agent will pay rent if possible, or pay the owner with accumulated wealth and tiles before any debt is forgiven. Successful purchases result in the value of the properties and rent to increase, resulting in migration of wealthy agents to expensive properties while owning as many tiles as possible. The simulation ends when no tiles can be purchased, and results of an experiment correspond to ensemble simulations -e.g., computing statistics for 1000 runs. Figure 3 shows an example of this using Mesa.

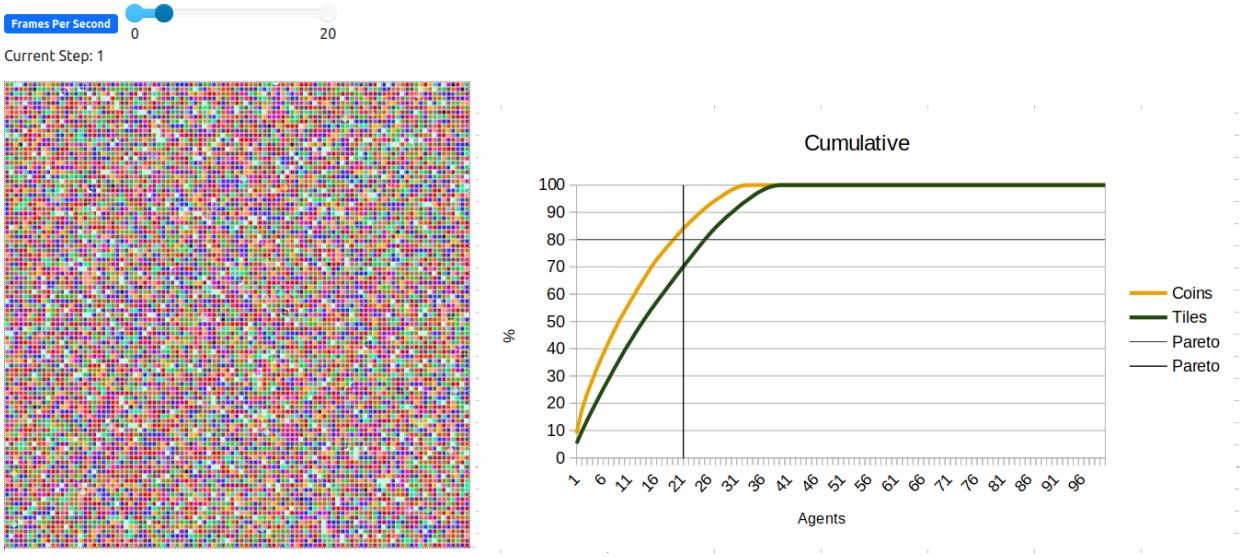

Figure 3: Pareto principle model implemented using Mesa

4.4 Pandemic Disease Spread

Recent events around COVID-19 and associated non-pharmaceutical interventions showcased the power of ABM simulations to plan -and act- at global [49] and local [50] scales. In reality, misinformation and other social factors [51] played a substantial role, which can be captured in ABM simulations of contagion processes with coupled spatial and network components [52]. We invite readers to consult the mathematical formulation of the Susceptible-Infected-Recovered (SIR) compartment model and COVID-19 extensions across extensive literature on the subject.

For our purposes, we produced a simplified model of 100x100 tiles with one agent each. Agents do not move across the grid and only interact via messages as described below. During each time step t, agents may become infected with a probability of $p_t = 0.21$. Agents can reduce their vulnerability by wearing a mask ($p'_t = 0.9p_t$), increase it by assuming immunity to the disease thanks to wearing an amulet ($p'_t = 1.1p_t$), or remain the same by choosing the ineffectual drinking of a glass of orange juice. We ensure $0.01 \leq p_t \leq 0.21$. At the onset, agents lack direct knowledge of the effectiveness of each measure, yet start with an initial trust level corresponding to their state of belief. Choices of measures are made using a roulette wheel algorithm of over these.

Belief update occurs by means of messages sent and forwarded across agents. The network contains up to ten whom the agent forwards messages to, and zero or more from whom it receives messages from. The messages contain trust levels for each measure. A weighted average across all incoming messages comprises the update rule for the belief state of the agent [53]. The update occurs during two time steps: one in which trust in influencers contributes up to 0.02, and another raising it to 0.03 in relation to the weighted average. Agents perform a single Bernoulli trial with the final value p'_t to update their contagion state. If infected, the agent decreases its confidence $c(m^*)$ by $c'(m^*) = 0.8c(m^*)$ and increases confidence in others $c(m)$ by $c'(m) = 1.01c(m)$. Otherwise, $c'(m^*) = 1.1c(m^*)$ and $c(m)$ by $c'(m) = 0.99c(m)$. After this computation, agents forward their belief state to their forwarding contacts. We run the simulation across 1200 time steps and visualized trust level per measure (Figure 4).

Figure 4: Pandemic disease spread implemented using Mesa.

4.5 ABM simulations vs FSMM

Finally, we performed a mapping between each ABM simulation and the simulation features described in Section 3.1. To do so, we extracted individual features required by each ABM simulation and obtained a qualitative picture of common and unique ones. We performed a similar exercise for interrogation capabilities in Section 3.2. Results are summarized in Figures 5-6, which were used to elicit common modeling needs that ABM frameworks should help satisfy. We note that our results are also constrained by the extent in which our choice of models, which in our case disallowed a fair comparison across research support features. Later works will report results using models put in the context of actual hypothesis testing and requiring these features.

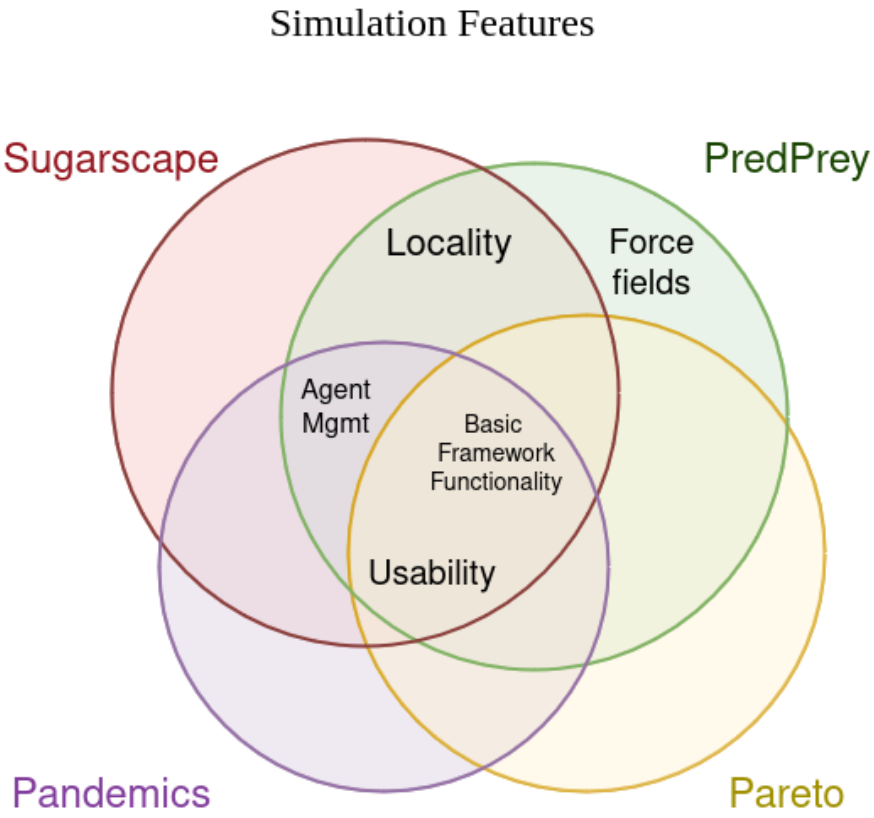

Figure 5: Simulation features exercised by all ABM simulations.

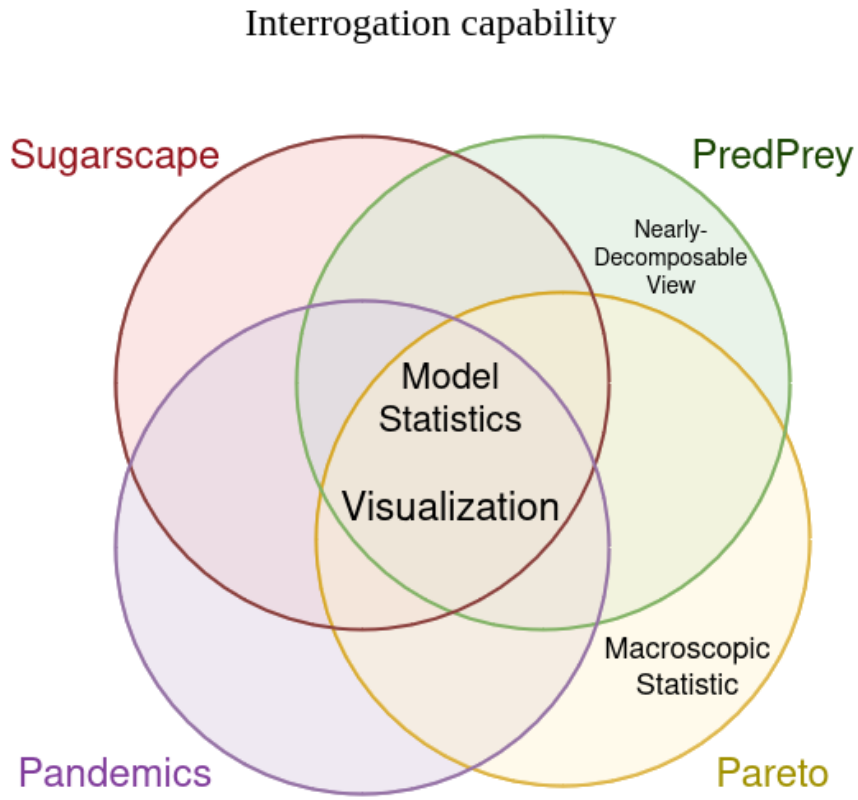

Figure 6: Interrogation capabilities exercised by all ABM simulations.

5 Target Frameworks

Five ABM frameworks were chosen to be evaluated with our Feature Space Maturity Model. Our selection is informed by their prevalence across the ABM research community, quality of documentation, historical relevance, and accessibility. At the onset, we selected Swarm for evaluation [54], released on November 1997 and its last stable release (version 2.4.1) is of April 2009. However, multiple technical obstacles ranging from limiting execution environments to availability of libraries resulted in its elimination from this research process. The following ABM frameworks constitute the basis of our analysis.

NetLogo [55]. First released on 1999, most recent stable version: 6.3.0, December 2021. It provides a multi-agent programmable modeling environment, and targets domain experts with limited or no experience in computer programming. As of today, NetLogo is still widely popular in education and research.

Multi-Agent Simulator Of Neighborhoods (MASON) [56, 57]. First released in 2003, most recent stable version: 22, July 2024. MASON is structured as a Java-based discrete-event multi-agent simulation library core intended to be highly performant. Simulations need to extend from the class sim.engine.SimState, and provides a lightweight basis to which users must add functionality. It assumes computer programming proficiency.

REcursive Porous Agent Simulation Toolkit. (Repast) [58, 59]. First released in 2003 (1.4beta), most recent stable version: Repast Simphony 2.11.0, July 2024. Repast models social actors as porous entities whose definitions benefit from object-oriented features inherited from Java. It contains advanced features, favors interactivity, and can be executed in single systems or clusters (Repast HPC). Repast uses Groovy - a wrapper over the Java programming language [60] usually distributed as a Plug-In for the Eclipse IDE Open Source Platform [61]. While not explicitly mentioned in their documentation, it appears that Repast re-implements all calls present in NetLogo.

Mesa [62, 63]. First released in 2015, most recent stable version: 3.2.0, May 2025. Mesa separates modeling, analysis and visualization through an ABM library intended to facilitate both interactive simulation and batch processing. Its most recent version includes geospatial modeling capabilities, advanced scheduling modes, meta-agents and extensions.

Agents.jl [64]. First released in April 2019, most recent stable version: 6.2.10, April 2025. Agent.jl is implemented in Julia [65], aiming at high performance ABM simulations at scale. It an open source, high quality software package with extensive documentation. Internally, it is structure via modular design and benefits from Julia's functional programming style. Agents.jl provides rich spatial representation features, including access to Open Street Map data.

During this research, a reference architecture for ABM frameworks and its Proof-Of-Concept (POC), AB-X, were developed. Thus, in addition to the five pre-existing ABM frameworks enumerated above, which were evaluated experimentally to be scored according to the FSMM, the AB-X POC was evaluated as well. AB-X was designed to meet several goals: to match the proposed architecture as closely as possible, create a product which followed the architectural goals, keep the framework simple and maintainable, have a simple yet powerful syntax, make it efficient in execution and amenable to develop simulations very quickly [10].

6 FSMM Scoring Results and Discussion

To achieve a comprehensive scoring of each framework described above, we implemented all four models independently and recorded the degree of maturity per feature under evaluation. The scoring system uses the scale from 0 to 3 as described in [19]. Results for simulation features (Table 1) and interrogation capabilities (Table 2) were tabulated and contrasted (Figure 7).

Feature	**NetLogo**	**MASON**	**Repast**	**Mesa**	**Agents.jl**
Usability	3	3	1	2	1
B. fram. funct.	3	3	3	3	3
Locality	3	2	3	2	3
Agent manag.	2	2	2	2	1
Force fields	0	0	0	0	1
Summary	**11**	**10**	**9**	**9**	**9**

Table 1: FSMM score across simulation features.

Feature	NetLogo	MASON	Repast	Mesa	Agents.jl
Visualization	3	3	2	2	1
Mod. lev. stats.	3	3	3	3	3
Macrosc. supp.	2	2	2	2	2
Near. dec. sys.	0	0	0	0	0
Summary	**8**	**8**	**7**	**7**	**6**

Table 2: FSMM score across interrogation capabilities.

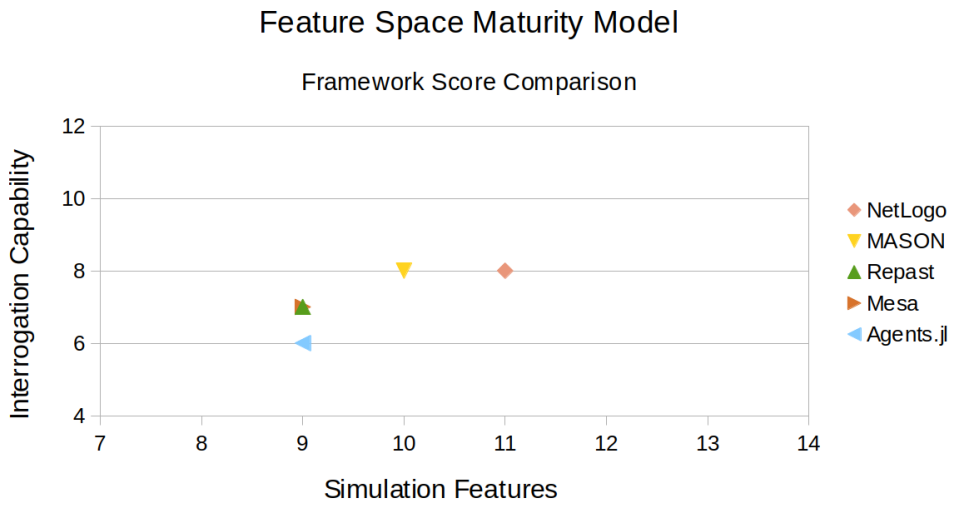

Figure 7: Scatter plot contrasting simulation feature and interrogation capability scores across all ABM frameworks using our FSMM.

At the start of our investigation, we included several additional candidate features in the FSMM. We later simplified and streamlined them to avoid conceptual overlaps. The usefulness of a maturity model depends largely on the practicality of its application and its ability to reduce ambiguity when used as an evaluation tool. The features described here are the result of our efforts in this direction.

Overall, support for force fields and nearly decomposable systems is largely missing from frameworks. Other features also vary in maturity. This may result from negative selection bias linked to domain problems, modeling scale (e.g., number of agents), and community practices. A possible explanation is that frameworks—like other software artifacts—face evolutionary pressures from the incremental identification of needs. Depending on the initial software architecture, some needs are easier to implement in certain frameworks and harder in others. In this sense, software evolution is self-constraining. This fact alone justifies the value of this project: ABM framework builders must evaluate the whole feature space, not just niches.

Our results revealed important observations about the frameworks beyond the features captured by the FSMM. The long use history of both NetLogo and MASON appears to support their overall maturity. Users have had more time to create models and request features, pointing to a software–research co-design cycle of nearly two decades. Surprisingly, despite their recency, Mesa and Agents.jl are close in simulation features and continue to grow. Repast, although mature, requires dealing with the complexity of the Eclipse IDE at a programmatic level and includes some non-intuitive features. For Mesa, the version we tested showed slow interactive simulations in visualization, but the latest release has addressed these performance issues and remains to be tested.

Despite NetLogo being the most capable framework in terms of simulation support, writing certain behaviors remains challenging from a programming perspective. Its LISP-like internal language can be daunting for users more familiar with modern programming paradigms. This suggests that part of its success may be explained by sunk cost and technical debt among its users. In addition, the framework was designed for simulations to be contained in a single file, which is difficult for larger and more complex projects.

Agents.jl has some basic deficiencies in rendering. The simulation uses GlMakie which masked some failures and made debugging harder. Another issue was that starting simulations required at least 3 minutes of prepossessing in a high-end laptop. Starting with a 'precooked' environment decreased this time but did not remove it. Other frameworks started almost immediately.

7 Macroscopic Comparisons and Fidelity to Reality

ABMs serve as *in silico* laboratories where researchers define agent characteristics, set initial conditions, specify agent-agent interaction rules, and implement static or dynamic rules for transitioning between model states.

Non-trivial models often allow multiple implementations of agent behavior. Researchers wishing to maximize their understanding of a complex system, as clearly as they possibly can, must introduce several behaviors for its agents, to identify which ones match better empirical macroscopic results. This strategy is known as a pluralistic methodology, as noted by several authors [66].

Macroscopic comparison metrics are desirable, independent of the models' internal dynamics. For example, there could be two models to measure a pandemic contagion rate, but the first one may be much more resource-consuming than the second because it simulates human behavior in more detail. If the identified contagion rates of both models are similar, regardless of the implementation, it can be said the models are macroscopic comparable, and the second one will be desired for large-scale simulations, as it is less resource-consuming.

Another key concept in ABM is fidelity to reality, which can be viewed as a two-stage chain: fidelity of the specification to reality, and fidelity of the implementation to the specification. The first point is not developed in this work, but there are several papers on this topic, such as [67].

For the second point: there is no such thing as "the test" to validate (increase confidence that an inference about a simulated process is correct for the actual specification) an implementation [68].

One way to tackle correctness and fidelity to the specification is to use macroscopic statistical analysis over multiple implementations of the same specifications in different frameworks. To do that, a list of quantitative macroscopic observables needs to be defined, which should be provided by each implementation. Then, for each implementation, the simulation can be run several times to obtain statistical behaviors of each observable.

Observables can be collected via different approaches. An observable could be an environment value, the aggregate of an agent's values, or some ratio when the simulation completes or reaches a milestone. It could also be the number of rounds that it took to reach some specific status.

When statistical behavior of an observable differs among implementations, it could be seen as an alert for one or more of the implementations that do not follow the specification correctly in some desired area. Likewise, consistent statistical behavior among implementations increases the confidence that all of the implementations follow the specification correctly.

In this work, macroscopic statistical analysis was used to validate that for one of the experiments, all implementations produced the same statistical behavior for three defined macroscopic observables in 10,000 executions of the Pareto experiment.

8 Baseline Model for Statistical Analysis

The Pareto principle known as Economic Wealth Inequity states that for many outcomes, roughly 80% of consequences come from 20% of causes (the "vital few"). Other names for this principle are the 80/20 rule, the law of the vital few, or the principle of factor sparsity. It is based on the works of Italian economist Vilfredo Pareto, who wrote about the 80/20 connection while at the University of Lausanne. In his first work, Pareto showed that approximately 80% of the land in Italy was owned by 20% of the population [46]. An example is in the 1992 United Nations Development Program Report, which showed that the distribution of global income is very uneven, with the richest 20% of the world's population receiving 82.7% of the world's income which follows Pareto's principle [45].

This wealth inequality tends to emerge naturally in interactive environments, even from initially equitable conditions, due to population-level statistical mechanics and the Gini coefficient [69]. The Gini coefficient measures the inequality among the values of a frequency distribution, such as levels of wealth [70].

8.1 What will be modeled?

As mentioned, this model could grow enormously, so the basic model will be similar to a Monopoly game. It will be a 100x100 grid with 100 agents. Each tile has a purchase cost (in coins) and generates rent for its

owner when occupied by another agent. Each agent starts with no coins but will collect a static number of 75 coins per turn.

The rent per tile is calculated from a random number between 1 and 50 coins, and the cost of purchasing will be that value multiplied by ten. Those values do not change during simulation execution.

Tiles can only belong to one agent, and during its turn, the agent first will collect the static 75 coins, then it will move to another tile in a random position. If the tile is not owned and the agent has enough money to buy it, it will buy it. If the tile is owned and the agent can pay the rent, it will pay it to the owner of the tile.

Finally, if the agent cannot pay the rent, it will have to pay an initial amount using its available coins, and then, pay the rest by giving the tile owner some tiles it owns. It could be that agent's tiles cannot match the missing rent, then the agent will have to surrender all its tiles, but the rest of the rent will be forgiven. There is also the case where the list of surrendered tiles will exceed the cost of the missing rent pay, but the owner will not pay the change.

Calculating the list of tiles to pay the rest of the rent, where the excess is minimal, is a optimization variant of the subset sum problem, which is a known NP-hard problem [71], therefore a quick (greedy) approximation algorithm will be used. The agent tiles are sorted according to their cost, from cheaper to more expensive. Iteratively, the tiles will be added to an initially empty bag until the amount is reached or exceeded; thus, the cheaper tiles will be removed while the amount exceeds the rent cost.

After the tile is purchased, or the rent is paid, if the agent has enough coins, it can upgrade its properties by paying the current cost; this duplicates both the cost and the rent. Agents will upgrade from their cheaper to more expensive properties. The agents do not die, and no more agents are added during the simulation. The goal of each agent is to buy as many tiles as possible.

This experiment ends naturally when there are no more available tiles (to buy). The experiment should be run a thousand times with varying values for the tiles.

8.2 What is the goal of the simulation?

The goal of the simulation is twofold, the first is to confirm the Pareto principle at the end of the simulation. The data collected contains the final ratio of both the coins and the land owned by each agent.

Secondly, we want to validate the fidelity to the specification of the five implementations through macroscopic statistical analysis. For each framework, three macroscopic observable values will be collected for 10,000 executions. They are the number of rounds to complete the experiments, the percentage of coins of the top 20% richer agents, and the percentage of tiles owned by the top 20% landowner agents.

These values will be plotted in three histograms. With one series for each framework, we aim to validate that they were implemented identically and exercise the idea of statistical validation of (otherwise) indistinguishable models.

The research approach followed resembles a waterfall. There were three main stages: research around the feature space needed to create the FSMM, then design a reference architecture with robust computational properties, and finally evaluation of the reference architecture through the Proof-Of-Concept (POC) and *in-silico* experiments.

9 Results of the Pareto experiment

9.1 Netlogo

Pareto, as defined in the experiment, is implemented in NetLogo [2].

The implementation required a single experiment file that contains both the user interface and the behavior specifications. The implementation was straightforward with only one type of agent. Visualization was very fast, and for the Model statistics only the cumulative curve in decreasing order (the Pareto curve) for the owned tiles and coins was added.

A Comma Separated Value (CSV) file with data for macroscopic statistical analysis was created using the provided file saving functionality. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times.

[2] here `https://gitlab.com/msc_tesis/NetLogo/-/blob/main/Pareto/`

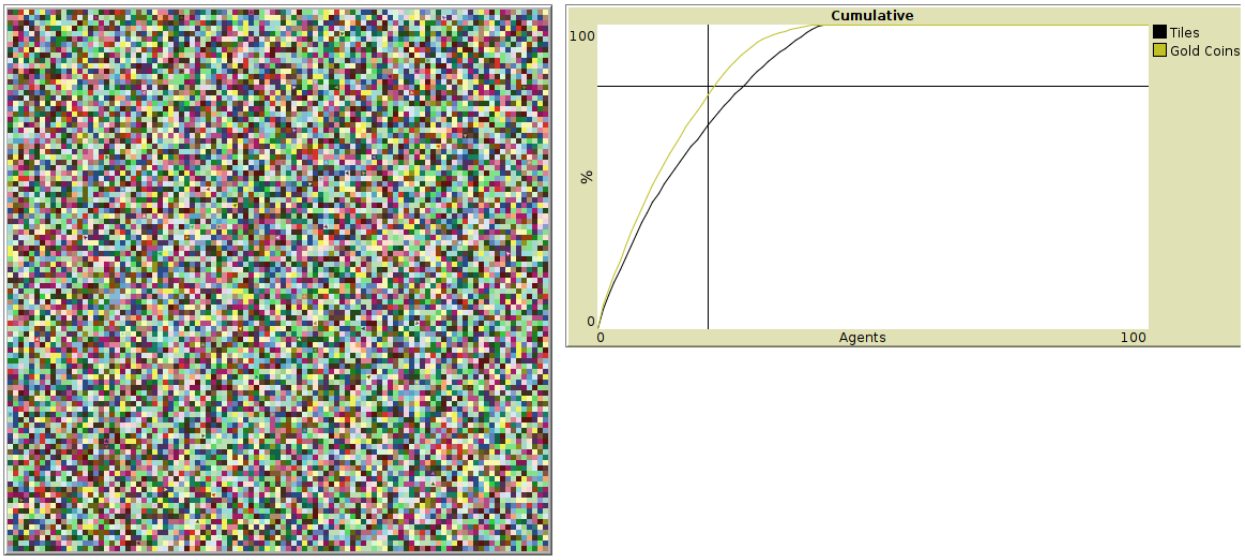

Figure 8: Pareto on NetLogo

9.2 MASON

Pareto, as defined in the experiment, is implemented in MASON [3].

The implementation required five Java files, which contain both the UI and the behavior specifications. The implementation was done with a single type of agent. Visualization was very fast, and for the Model statistics only the cumulative curve in decreasing order (the Pareto curve) for the owned tiles and coins was added. There was a minor limitation to draw the 20/80 lines, so two series were added to draw those lines.

A CSV data file was created for macroscopic statistical analysis using standard Java file system support. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times.

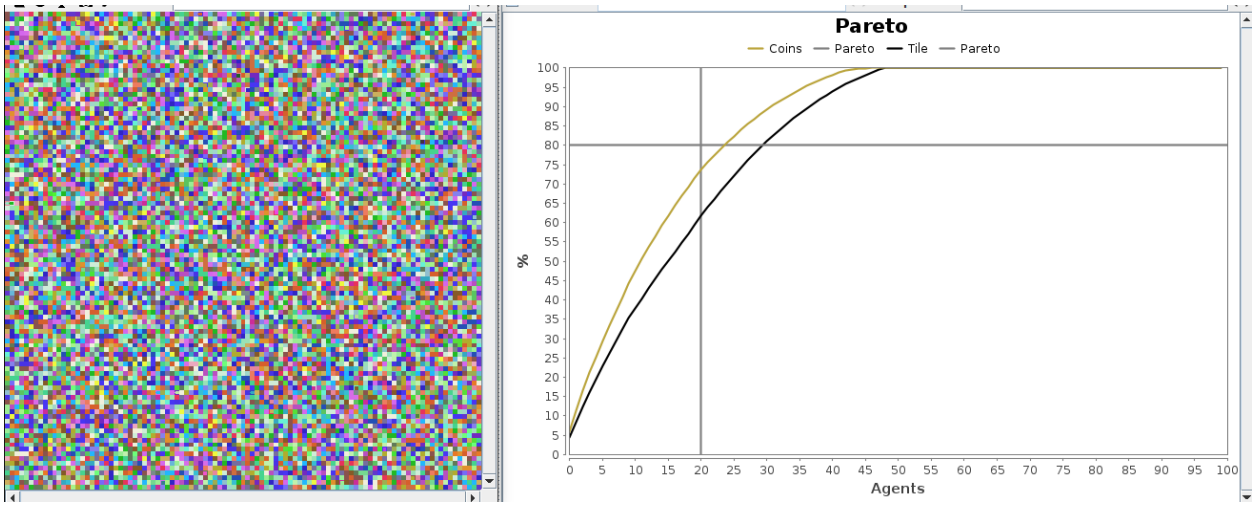

Figure 9: Pareto on MASON

9.3 Repast

Pareto, as defined in the experiment, is implemented in Repast [4].

The implementation required five Groovy files, plus 100+ files created by the Eclipse framework. Basic framework functionality, like debugging capabilities, was provided by Eclipse, whereas visualization of only

[3] here `https://gitlab.com/msc_tesis/MASON/-/blob/main/Pareto/`

[4] here `https://gitlab.com/msc_tesis/Repast/-/blob/main/Pareto/`

time-based graphs is possible in the current version Repast. One Model statistics graph required to visually identify the Pareto behavior was the cumulative of coins and tiles. However, to create it, the data were stored in a CSV file and then post-processed as a graph in a LibreOffice Calc worksheet [72], a third-party tool.

A CSV data file was produced for macroscopic statistical analysis using the standard Java file system support. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times. For this experiment, usability was very poor, as it was required to clone a version of a previous experiment, given that the Eclipse plugin experienced issues when creating new simulation projects from scratch. Although Repast seems to provide the functionality to run multiple iterations at the same time, the documentation was lacking, and the behavior seemed intricate. The simulation was executed in a loop that reset the values, with a user interface checkbox added to allow 10,000 iterations.

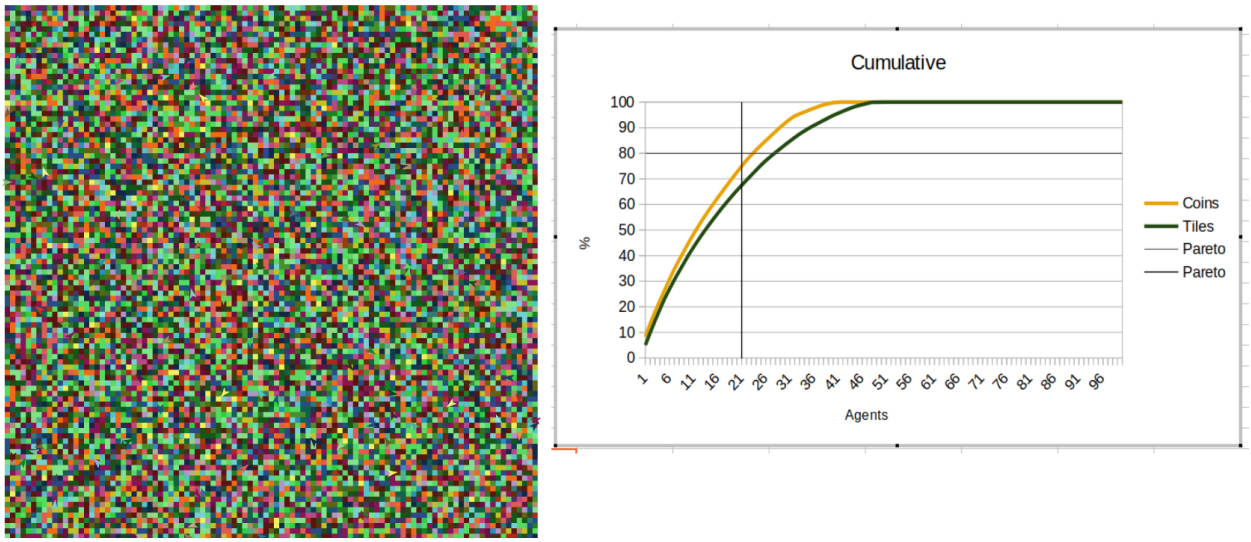

Figure 10: Pareto on Repast

9.4 Mesa

Pareto, as defined in the experiment, is implemented in Mesa [5].

The implementation required four Python files, including the Pareto main file, one for the tile, a visualization helper, and one for the agent. Basic framework functionality, such as debugging capabilities, was provided by IntelliJ. Visualization of graphs other than time-based was not possible in the current version of Mesa. One Model statistics graph required to visually identify the Pareto behavior was the cumulative of coins and tiles. To create it, the data was stored in a CSV file and then post-processed as a graph in a LibreOffice Calc sheet [72], a third-party tool.

A CSV data file was generated for macroscopic statistical analysis using the standard Python file system support. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times. For this experiment, the UI's usability was too slow, but acceptable during the headless 10,000 iterations.

[5] here `https://gitlab.com/msc_tesis/Mesa/-/blob/main/Pareto/`

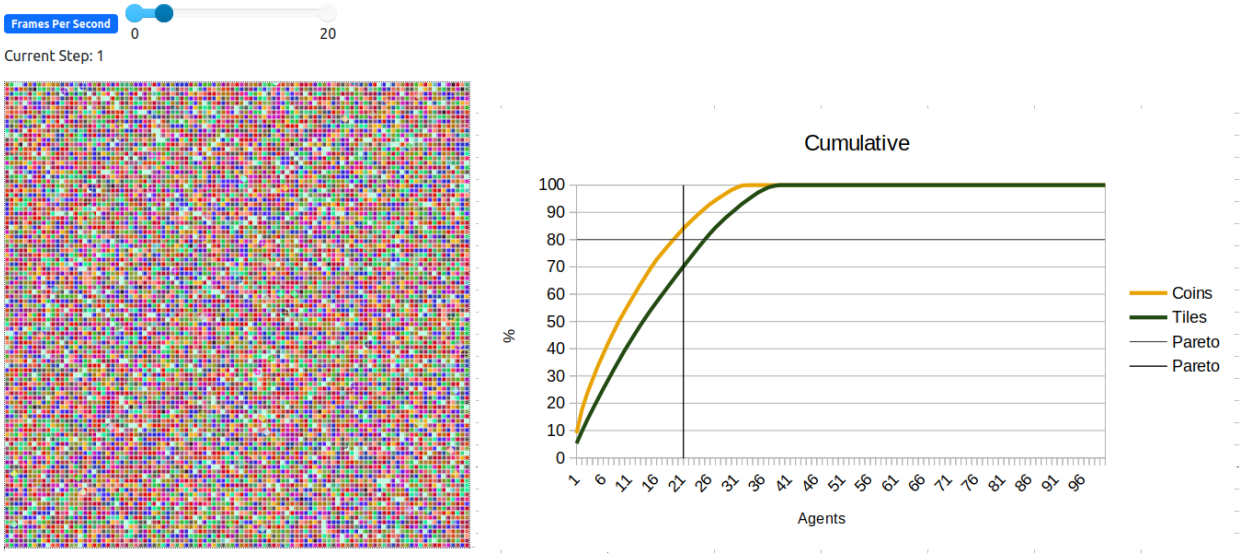

Figure 11: Pareto on Mesa

9.5 Agent.jl

Pareto, as defined in the experiment, is implemented in Agents.jl [6].

The implementation required several Julia files, covering both the user interface and behavioral specifications. No external Julia modules files were required as MS Code provided Basic framework functionality, such as debugging.

The implementation was simple with only one type of agent. Visualization was very fast. As with Repast and Mesa, only time aggregate graphs are allowed in Agents.jl. The expected Model statistics was the cumulative curve in decreasing order (the Pareto curve) for the owned tiles and coins, and had to be saved in a file to be plotted afterward by a third-party tool.

A CSV data file for macroscopic statistical analysis was created using Julia's standard file system support. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times. Due to its functional features, it was natural to create and execute simulations in Julia. If desired, it is easy to modify each simulation execution independently. For this experiment, usability was good, especially the speed of headless executions.

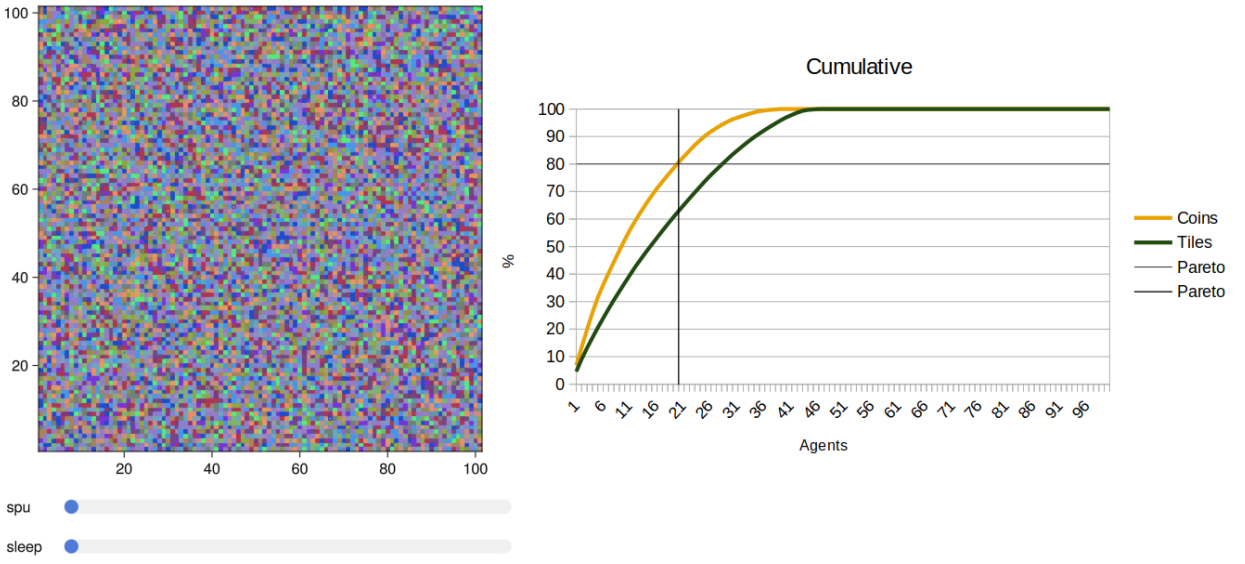

Figure 12: Pareto on Agents.jl

[6] here `https://gitlab.com/msc_tesis/Agents.jl/-/blob/main/Pareto/`

9.6 Pareto on AB-X

Pareto, as defined in the experiment, is implemented in AB-X [7].

The implementation required four JavaScript files that contain both the UI and the behavior specifications. A file was added for testing. No external files were required, as AB-X provided the basic framework functionality.

The implementation was straightforward with only one type of agent, called Player. Visualization was very fast, and for the Model statistics only the cumulative curve in decreasing order (the Pareto curve) for the owned tiles and coins was added. It was simple to add guide lines in AB-X.

A CSV data file was generated for macroscopic statistical analysis using the standard AB-X file system support. Each line contains the three macroscopic observables defined in the specification. The simulation was run 10,000 times.

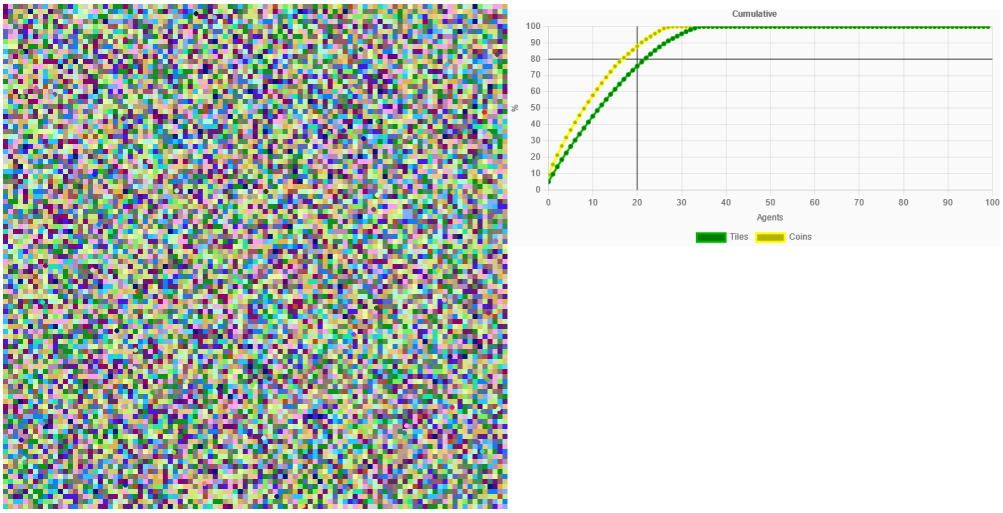

Figure 13: Pareto on AB-X

10 Macroscopic Statistical Analysis & Specification Fidelity

The FSMM evaluates whether a framework provides the facilities required to implement ABM specifications, whereas the macroscopic statistical protocol evaluates whether independent implementations of the same specification produce statistically compatible outcomes. Together, both components address complementary aspects of framework adequacy: capability coverage and implementation fidelity.

10.1 Histogram

The histogram depicted in Figure 14 shows that the macroscopic comparables of AB-X versus all other versions behave statistically identical, which supports the goal of all implementations following the specification.

[7] here `https://gitlab.com/msc_tesis/abx-fsmm/-/tree/main/Pareto`

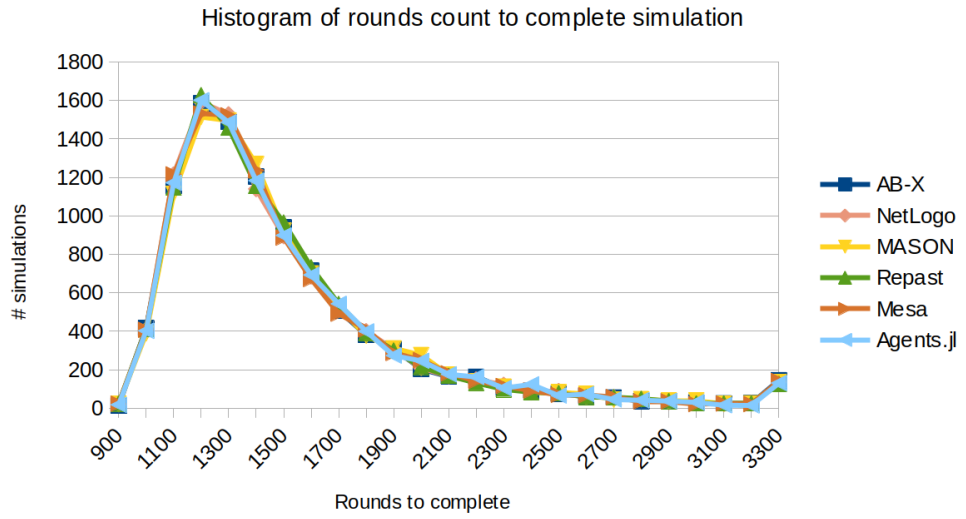

Figure 14: Histogram for rounds including AB-X

As with the other frameworks, AB-X exhibits a tail after 2500 rounds. This is because each tile needs to be purchased to complete the simulation, but agents can only reach a tile by chance, which could create iterations with more than 5000 steps.

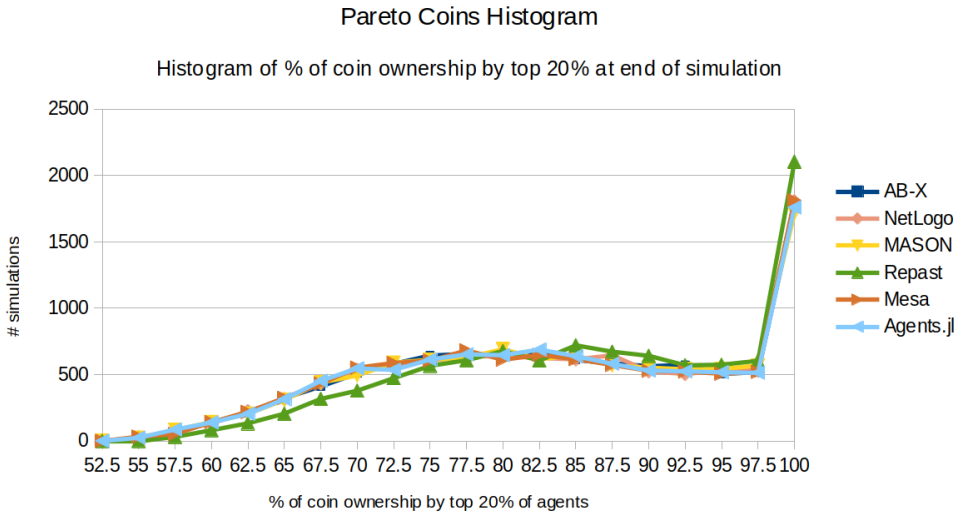

Figure 15: Histogram for coins including AB-X

In the AB-X implementation, coin ownership is clearly around 80% and it also has a hump where the 20% or less of the agents owned the 100% of the coins. This hump was also observed in the other implementations.

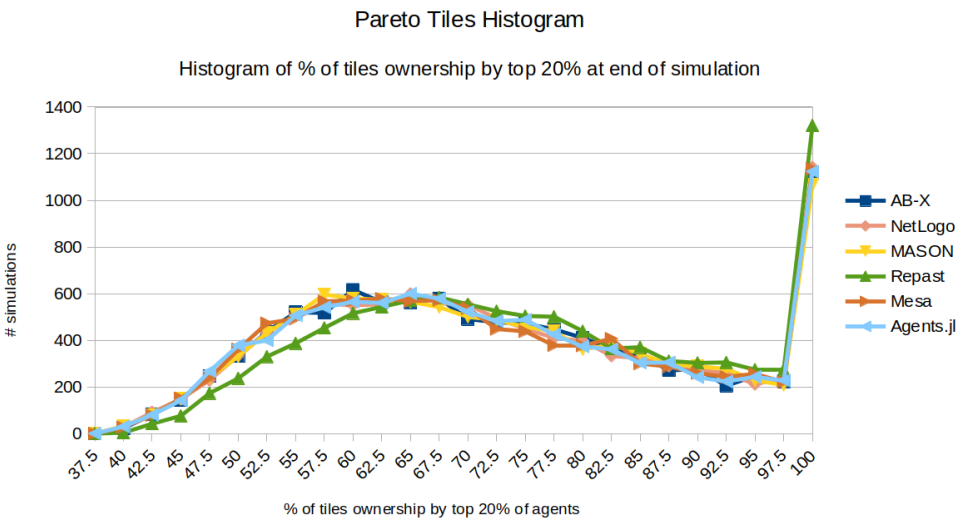

Figure 16: Histogram for tiles including AB-X

As with the other implementations, tile ownership is clearly around 70-80%. Again, there is a hump where 20% or less of the agents owned 100% of the tiles.

10.2 Coefficient of Variation

For the six frameworks, the calculated Coefficient of Variation (CV) for the three histograms (round count to completion, top 20% ownership of coins, top 20% of tile ownership) is shown in the following table.

Table 3: Coefficient of variation, including AB-X results, for three chosen macroscopically observables

Framework	Rounds to finish	Pareto Coins	Pareto Tiles
AB-X	0.010	0.016	0.027
NetLogo	0.011	0.016	0.027
MASON	0.010	0.016	0.027
Repast	0.011	0.015	0.025
Mesa	0.011	0.016	0.028
Agents.jl	0.010	0.016	0.028

As the Coefficient of variation is a unit-less measure, it is particularly useful for comparing variability across datasets with different units or scales. The example shows nearly identical values for almost all implementations, including AB-X - with a small difference in Repast, especially in the CV of the Pareto Tiles.

10.3 Fréchet Distance

The Fréchet distances between axis/frameworks of the three histograms (round count to completion, top 20% ownership of coins, top 20% of tiles ownership) were calculated. Although there are techniques to approach Fréchet distance efficiently [73], the general formula was used because of the small number of points in the curves.

Fréchet distance for rounds follows. The highest values are marked with *.

Table 4: Fréchet distance for histograms of rounds to complete experiment

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	0.0	43.0	49.0	73.0*	65.0	60.0
NetLogo	43.0	0.0	68.0	87.0*	44.0	68.0
MASON	49.0	68.0	0.0	114*	72.0	95.0
Repast	73.0*	87.0*	114.0*	0.0	130*	102*
Mesa	65.0	44.0	72.0	130*	0.0	97.0
Agents.jl	60.0	68.0	95.0	102*	97.0	0.0

Fréchet distance for the histogram of top 20% coin ownership follows. Highest values are marked with *.

Table 5: Fréchet distance for histograms of top 20% coins ownership

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	0.0	39.0	50.0	338*	47.0	42.0
NetLogo	39.0	0.0	89.0	299*	38.3	46.0
MASON	50.0	89.0	0.0	388*	97.0	43.0
Repast	338*	299	388*	0.0	291*	345*
Mesa	47.0	38.32	97.0	291*	0.0	54.0
Agents.jl	42.0	46.0	43.0	345*	54.0	0.0

Fréchet distance for histogram of top 20% tile ownership follows. The highest values are marked with *.

Table 6: Fréchet distance for histograms of top 20% tiles ownership

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	0.0	30.0	54.0	201	38.08	44.2
NetLogo	30.0	0.0	75.0	180*	33.1	41.6
MASON	54.0	75.0	0.0	255*	69.0	57.0
Repast	201*	180*	255*	0.0	186*	198*
Mesa	38.1	33.1	69.0	186*	0.0	31.0
Agents.jl	44.2	41.6	57.0	198*	31.0	0.0

The values show that AB-X results are similar to the previous implementations, except for the unidentified differences in Repast's implementation.

10.4 ANOVA and F-Test

Both ANOVA and F-test provide a p-value [74]. In null-hypothesis significance tests, where the null hypothesis is $H_0 : \mu_1 = \mu_2$, the p-value is the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct [75]. Here, *extreme* means farther away from what would be expected under the null hypothesis.

The ANOVA was applied to all the values in the histograms now also including the AB-X implementation (round count to completion, top 20% ownership of coins, top 20% of tiles ownership) and the F-Test was applied the each pair of histograms.

The usual notation of * is followed:

* $0.01 < p < 0.05$

** $0.001 < p < 0.01$

*** $p < 0.001$

The results are shown in tables below.

10.4.1 Rounds to completion

The p-value for the ANOVA was 0.273, that is more than 0.05, which fails to reject the null hypothesis and means there are no significant differences. However, for F-Test between frameworks, the results were below 0.05.

Table 7: P-value for F-test for histograms of rounds to complete experiment

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	-	.997	.268	.258	.892	.353
NetLogo	.997	-	.291	.275	.893	.376
MASON	.268	.291	-	.027*	.224	.852
Repast	.258	.275	.027*	-	.331	.041*
Mesa	.892	.893	.224	.331	-	.297
Agents.jl	.353	.376	.852	.041*	.297	-

Here, it can be observed that for rounds, the results of Repast are statistically significantly different from Agents.jl and MASON.

10.4.2 Top 20% of coin ownership

The p-value for the ANOVA was $< 10^{-5}$ which rejects the null hypothesis, and indicates the means for the histograms of the top 20% of coin ownership, are significantly different. The F-test results between frameworks are as follows.

Table 8: P-value for F-test for histograms of top 20% of coin ownershipWhere ϵ means $< 10^{-5}$***

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	-	.870	.733	ϵ	.552	.363
NetLogo	.870	-	.860	ϵ	.669	.459
MASON	.733	.860	-	ϵ	.800	.571
Repast	ϵ	ϵ	ϵ	-	ϵ	ϵ
Mesa	.552	.669	.800	ϵ	-	.755
Agents.jl	.363	.459	.571	ϵ	.755	-

It can be seen that for the top 20% of coin ownership, the results of Repast are statistically significantly different from those of all the other frameworks.

10.4.3 Top 20% of tiles ownership

The p-value for the ANOVA was$< 10^{-5}$ which rejects the null hypothesis, and indicates the means of the histograms for the top 20% of tile ownership, are significantly different. The pairwise F-test results are shown below.

Table 9: P-value for F-test for histograms of top 20% of tiles ownershipWhere ϵ means $< 10^{-5}$***

	AB-X	NetLogo	MASON	Repast	Mesa	Agents.jl
AB-X	-	.777	.603	ϵ	.687	.560
NetLogo	.777	-	.423	ϵ	.494	.388
MASON	.603	.423	-	ϵ	.909	.949
Repast	ϵ	ϵ	ϵ	-	ϵ	ϵ
Mesa	.687	.494	.909	ϵ	-	.859
Agents.jl	.560	.388	.949	ϵ	.859	-

Here, it can be seen that for the top 20% of tiles ownership, the results of Repast are statistically significantly different for all the other frameworks.

These results keep indicating a consistency issue in Repast which could be caused by an underlying cause related to pseudo-random number generation or in the experiment setup or a behavior not identified even after several code reviews.

11 Discussion on experimentation and its results

Among the frameworks studied, MASON and NetLogo enabled the creation and execution of the simulation 10,000 times with relatively few technical issues. In contrast, Mesa and Repast presented challenges related to execution speed and stability during development and experimentation. In particular, MASON exhibited shorter execution times and fewer stability-related issues than the other evaluated frameworks.

Agents.jl presented several practical limitations during development. Its rendering capabilities are comparatively limited, and the use of GlMakie complicated debugging because some failures were not directly exposed to the user. In addition, launching the simulation required approximately three minutes of preprocessing on a high-end laptop. Although the use of a precompiled ("precooked") environment reduced startup time, it did not eliminate this overhead. The other evaluated frameworks started with substantially lower initialization times.

For Mesa, the integration between the web-based front end and the back-end server introduced significant overhead during development and execution. Running 10,000 simulations through the graphical interface was not practical because of the associated performance costs. Consequently, the experiments were conducted using a modified version of the model without the user interface. This limitation was not observed in the other frameworks.

In the case of Repast, development was affected by issues associated with its Eclipse IDE plug-in architecture. The framework generated numerous auxiliary files required for simulation execution, which frequently needed to be recreated after source code modifications. During the study, the development environment also experienced failures that prevented the creation of new projects, requiring new work to be based on previously existing project directories. In relation to Repast, despite several rounds of parameter tuning and implementation review, it was not possible to obtain statistical results consistent with those produced by the other five frameworks.

About Repast's statistically differing results, we can only formulate tentative hypotheses. Possible explanations include differences in pseudo-random number generation, variations in agent activation and scheduling order, differences in update semantics, inconsistencies in initialization or state reset procedures between simulation runs, and subtle divergences in internal state management. Any of these factors could influence the emergent macroscopic behavior of the simulation despite an apparently equivalent implementation of the specification.

Determining the actual cause would require a dedicated investigation based on controlled execution traces, simplified benchmark models, and carefully minimized experiments designed to isolate individual framework mechanisms. Such an investigation would involve comparing intermediate states between implementations and systematically examining scheduling, random number streams, and state transitions.

Although this may constitute an interesting avenue for future research, identifying the specific sources of Repast's statistical deviations was outside the scope and objectives of the present study. Our goal was not to diagnose framework-specific implementation issues, but rather to demonstrate that macroscopic statistical analysis can serve as a practical method for detecting potential discrepancies among independent implementations of the same ABM specification.

12 Conclusions

The quality of inferences from ABM simulations depends on the adequacy of the models and the effectiveness of the tools. As models become more sophisticated - with more features needed for fidelity to reality and commensurability of results - they face tensions with the capabilities of ABM frameworks. Not all frameworks are equally expressive for all tasks, reflecting the broader tension between generality and specificity in computing. However, because scientific discovery and exploration share common requirements, finding ways to evaluate frameworks against those needs is both meaningful and productive.

ABM has evolved over the last few decades into a tool with dual use: an educational testing ground for ideas about complexity as well as a virtual laboratory for fundamental and applied research. Certain scientific problems and communities have adopted ABM more readily because of the relational nature of the underlying phenomena, which sometimes creates the impression of restricted use and applicability. The COVID-19 pandemic drastically changed this view, as ABM simulations involved real stakes and became publicly visible. As a result, the ability of current and future users to assess existing ABM frameworks has gained importance in recent literature, even for those running on high-performance computing platforms [76].

Assessing the maturity of complex software presents methodological challenges. These challenges are particularly evident in scientific software, where performance and correctness must be considered alongside reproducibility, accuracy, and long-term sustainability. Consequently, comprehensive evaluation methods are needed to address the characteristics of ABM frameworks and simulations. At the same time, broader evaluation criteria increase the possibility of overlap among categories, as observed during the application of our assessment model. Our results indicate that the practicality and applicability of a maturity model are important considerations in its design.

All ABM frameworks evaluated in this study shared a common set of features. Historically, ABM framework development has followed a path of adopting useful features while introducing alternative solutions to perceived limitations. As in our exercise, assessing the overall fitness of a framework also depends on its intended use. Our effort to replicate classical ABM simulations highlights the need for model implementations as transparent research objects [77] that can be inspected, shared, and criticized. Although we identified strengths and weaknesses across frameworks, additional studies are needed to further validate and extend these findings.

Our results suggest that future work on ABM frameworks may benefit from community efforts to consolidate the elements evaluated here together with other established approaches (e.g., the ODD protocol [78]) into a broader synthesis. The authors are pursuing ongoing work to develop a reference architecture based on the FSMM to address this challenge. The findings also indicate that, despite the flexibility of highly programmable frameworks, their role is to provide structures that support scientific modeling while enabling different implementation strategies.

ABM frameworks are powerful tools for simulating and analyzing complex systems composed of interacting agents. Their main advantage resides in their ability to reproduce emergent behavior observed in or expected from real systems. We proposed a statistical method to validate the functional equivalence of two or more implementations of the same experiment specification. Using it, we verified that, for one of the proposed validation experiments (the Pareto experiment), five of our six implementations are statistically equivalent.

Using statistical tools, we observed that the Repast implementation did not behave as the other implementations. Even after several hours of debugging we could not determine the cause of its slight but statistically confirmed difference in behavior.

Statistical analysis of multiple implementations of the same simulation can provide evidence regarding fidelity to specification. When implementations fail to exhibit equivalent behavior, the results may indicate discrepancies between an implementation and the intended specification.

One of the central challenges of ABM is the fidelity of simulations to the systems they represent, and fidelity to specification constitutes an important intermediate step toward that objective.

Identifying statistically equivalent behaviors can help reduce research costs. If two simulations are indistinguishable in small- or medium-scale experiments, the less computationally expensive implementation may be sufficient for larger-scale simulations.

13 Future work

Some avenues for future work have been identified:

- Expanding the set of models and developing example scientific analysis pipelines around a hypothesis is needed to evaluate research support features.

- Our FSMM outlines a path to build a formal reference architecture and create a proof-of-concept implementation that includes all of its features.
- Using macroscopic analysis of statistical observables helps to identify differences between multiple implementations of a simulation. For this work, four experiments were implemented in five different frameworks. Only the Pareto experiment was subject to the process of collecting observables for statistical analysis. It is desirable to apply similar macroscopic analyses to identify deviation in the remaining experiments and, if found, to correct them.
- It might be interesting to continue research to identify why, out of the five implementations of the Pareto experiment, only Repast produced statistically different results. This was outside of our original research.
- Although there can be statistical analysis of any variable, it would be interesting to design a formal process to identify which variables are important to validate statistical similarity.
- On a more theoretical note, our FSMM is a simple evaluation tool based on numerical facilitator levels. A more powerful alternative would be to redesign it around the concept of focus area maturity models [79]. That approach seems particularly appropriate for ABM frameworks, since they evolve through periods of conceptual and technical expansion across diverse domains. It may also involve creating a rubric that details a minimum set of expected features for automatic assessment and comparison of models and frameworks.

Acknowledgments

We thank the School of Computing Engineering at the Costa Rica Institute of Technology for its support during the development of this work. SNC wishes to acknowledge continued support in ABM research from the National Center of Supercomputing Applications, University of Illinois Urbana-Champaign. ITZ thanks Universidad CENFOTEC for their sponsorship and interest in the research reported herein.

Disclosure on AI use: The authors utilized Large Language Models (LLMs) to help with linguistic condensation and refinement. All technical content, analysis, and conclusions were developed entirely by the authors. The authors assume full responsibility for the final content of the article.

References

[1] F. Li Vigni, "Regimes of evidence in complexity sciences," *Perspectives on Science*, vol. 29, no. 1, pp. 62–103, 2021.

[2] J. M. Epstein, *Generative social science: Studies in agent-based computational modeling*. Princeton University Press, 2012.

[3] C. M. Macal, "Everything you need to know about agent-based modelling and simulation," *Journal of Simulation*, vol. 10, no. 2, pp. 144–156, 2016.

[4] H. Sabzian, M. A. Shafia, A. Bonyadi Naeini, G. Jandaghi, and M. J. Sheikh, "A review of agent-based modeling (ABM) concepts and some of its main applications in management science," *Interdisciplinary Journal of Management Studies (Formerly known as Iranian Journal of Management Studies)*, vol. 11, no. 4, pp. 659–692, 2018.

[5] C. W. Seagren, "Examining social processes with agent-based models," *The Review of Austrian Economics*, vol. 24, pp. 1–17, 2011.

[6] D. A. Robertson, "Agent-based models and behavioral operational research," *Behavioral Operational Research: Theory, Methodology and Practice*, pp. 137–159, 2016.

[7] J. C. Stevenson, "Agentization of two population-driven models of mathematical biology," in *Conference of the Computational Social Science Society of the Americas*. Springer, 2021, pp. 176–189.

[8] S. Núñez-Corrales, M. Friesen, S. Mudigonda, R. Venkatachalapathy, and J. Graham, "In-silico models with greater fidelity to social processes: towards ABM platforms with realistic concurrency," in *Proceedings of the 2020 Conference of The Computational Social Science Society of the Americas*. Springer, 2021, pp. 155–169.

[9] A. Antelmi, G. Cordasco, G. D'Ambrosio, D. De Vinco, and C. Spagnuolo, "Experimenting with agent-based model simulation tools," *Applied Sciences*, vol. 13, no. 1, p. 13, 2022.

[10] L. C. Lara Lopez, "On the feature space and architecture of ABM frameworks," M.Sc. Thesis, Tecnológico de Costa Rica, Cartago, Costa Rica, 2024. [Online]. Available: https://hdl.handle.net/2238/16413

[11] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. John Wiley, 2008.

[12] C. Macal and M. North, "Agent-based modeling and simulation: Desktop ABMS," *Proceedings - Winter Simulation Conference*, pp. 95–106, 12 2007.

[13] M. Gardner, "Mathematical games: the fantastic combinations of john conway's new solitaire game "life"," *Scientific American*, vol. 223, no. October, pp. 120–123, 1970.

[14] K. Nygaard and O.-J. Dahl, *The Development of the SIMULA Languages*. New York, NY, USA: Association for Computing Machinery, 1978, p. 439–480. [Online]. Available: https://doi.org/10.1145/800025.1198392

[15] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. O'Hare, "Agent based modelling and simulation tools: A review of the state-of-art software," *Computer Science Review*, vol. 24, pp. 13–33, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013716301198

[16] N. J. Roese, "Counterfactual thinking." *Psychological Bulletin*, vol. 121, pp. 133–148, 1997. [Online]. Available: https://doi.org/10.1037/0033-2909.121.1.133

[17] B. D. L. Marshall and S. Galea, "Formalizing the role of agent-based modeling in causal inference and epidemiology," *Am J Epidemiol*, vol. 181, no. 2, pp. 92–99, Dec. 2014.

[18] J. Becker, R. Knackstedt, and J. Poeppelbuss, "Developing maturity models for it management," *Business & Information Systems Engineering*, vol. 1, pp. 213–222, 06 2009.

[19] S. Núñez-Corrales and L. Gasser, "Scalable social simulation: Evaluation of current frameworks and a new approach," in *Proceedings of the 2018 International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation (SBP-BRiMS 2018)*, 2018.

[20] IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0*. Los Alamitos, CA, USA: IEEE Computer Society, 2024. [Online]. Available: https://www.computer.org/education/bodies-of-knowledge/software-engineering/v4

[21] *Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Product Quality Model*, International Organization for Standardization and International Electrotechnical Commission Std. ISO/IEC 25 010:2023, 2023.

[22] R. Sargent, "Verification, validation and accreditation of simulation models," in *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, vol. 1, 2000, pp. 50–59 vol.1.

[23] O. Balci, "Verification validation and accreditation of simulation models," in *Proceedings of the 29th Conference on Winter Simulation*, ser. WSC '97. USA: IEEE Computer Society, 1997, p. 135–141. [Online]. Available: https://doi.org/10.1145/268437.268462

[24] E. W. Weisstein, "Moore neighborhood," n.d., from MathWorld–A Wolfram Web Resource. [Online]. Available: https://mathworld.wolfram.com/MooreNeighborhood.html

[25] ——, "von neumann neighborhood," n.d., from MathWorld–A Wolfram Web Resource. [Online]. Available: https://mathworld.wolfram.com/vonNeumannNeighborhood.html

[26] J. Cao, X. Feng, J. Lu, and S. Das, "Mailbox-based scheme for mobile agent communications," *Computer*, vol. 35, no. 9, pp. 54–60, 2002.

[27] G. J. Maclay and M. Ahmad, "An agent based force vector model of social influence that predicts strong polarization in a connected world," *PLoS One*, vol. 16, no. 11, p. e0259625, Nov. 2021.

[28] Z. Li, C. H. Sim, and M. Y. H. Low, "A survey of emergent behavior and its impacts in agent-based systems," in *2006 4th IEEE international conference on industrial informatics*. IEEE, 2006, pp. 1295–1300.

[29] T. Holland, "Foundations for the modeling and simulation of emergent behavior systems," in *Engineering Emergence*. CRC Press, 2018, pp. 217–258.

[30] D. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2008. [Online]. Available: http://books.google.de/books?id=kMMJAm5bD34C

[31] H. A. Simon, "Near decomposability and complexity: How," *The Mind, The Brain And Complex Adaptive Systems*, p. 25, 2018.

[32] M. A. Janssen and E. Ostrom, "Empirically based, agent-based models," *Ecology and society*, vol. 11, no. 2, 2006.

[33] A. Wilhite and E. A. Fong, "Agent-based models and hypothesis testing: an example of innovation and organizational networks," *The Knowledge Engineering Review*, vol. 27, no. 2, pp. 221–238, 2012.

[34] A. Namatame and S.-H. Chen, *Agent-based modeling and network dynamics*. Oxford University Press, 2016.

[35] E. of Mathematics, "Graph theory," 01 2024. [Online]. Available: http://encyclopediaofmath.org/index.php?title=Graph_theory

[36] U. Wilensky and W. Rand, *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT press, 2015.

[37] P. A. Burrough, R. A. McDonnell, and C. D. Lloyd, *Principles of geographical information systems*, 3rd ed. London, England: Oxford University Press, Apr. 2015.

[38] P. Antosz, T. Szczepanska, L. Bouman, J. G. Polhill, and W. Jager, "Sensemaking of causality in agent-based models," *International Journal of Social Research Methodology*, vol. 25, no. 4, pp. 557–567, 2022.

[39] A. Chopra, S. Kumar, N. Giray-Kuru, R. Raskar, and A. Quera-Bofarull, "On the limits of agency in agent-based models," *arXiv preprint arXiv:2409.10568*, 2024.

[40] L. Blume, "Agent-based models for policy analysis," in *Assessing the Use of Agent-Based Models for Tobacco Regulation*. National Academies Press (US), 2015.

[41] J. M. Epstein and R. Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*. Brookings Institution Press, 1996.

[42] P. J. Wangersky, "Lotka-volterra population models," *Annual Review of Ecology and Systematics*, vol. 9, pp. 189–218, 1978.

[43] F. Hoppensteadt, "Predator-prey model," *Scholarpedia*, vol. 1, no. 10, p. 1563, Jan. 2006.

[44] Y. Berman, E. Ben-Jacob, and Y. Shapira, "The dynamics of wealth inequality and the effect of income distribution," *PloS one*, vol. 11, no. 4, p. e0154196, 2016.

[45] M. P. Todaro, "Review of Human Development Report 1992 by United Nations Development Programme," *Population and Development Review*, vol. 18, no. 2, pp. 359–363, 1992. [Online]. Available: http://www.jstor.org/stable/1973685

[46] V. Pareto, *Cours D'économie Politique*. Librairie Droz, 1964.

[47] H. M. Hochman and J. D. Rodgers, "Pareto optimal redistribution," *The American economic review*, vol. 59, no. 4, pp. 542–557, 1969.

[48] G. G. Piva, F. L. Ribeiro, and A. S. Mata, "Networks with growth and preferential attachment: modelling and applications," *Journal of Complex Networks*, vol. 9, no. 1, p. cnab008, 2021.

[49] N. Ferguson, "Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand," 2020. [Online]. Available: DOI:10.25561/77482

[50] S. Núñez-Corrales and E. Jakobsson, "The epidemiology workbench: a tool for communities to strategize in response to covid-19 and other infectious diseases," *MedRxiv*, pp. 2020–07, 2020.

[51] M. M. Ferreira Caceres, J. P. Sosa, J. A. Lawrence, C. Sestacovschi, A. Tidd-Johnson, M. H. U. Rasool, V. K. Gadamidi, S. Ozair, K. Pandav, C. Cuevas-Lou, M. Parrish, I. Rodriguez, and J. P. Fernandez, "The impact of misinformation on the COVID-19 pandemic," *AIMS Public Health*, vol. 9, no. 2, pp. 262–277, Jan. 2022.

[52] S. P. Mudigonda, S. Núñez-Corrales, R. Venkatachalapathy, and J. Graham, "Scheduler dependencies in agent-based models: A case-study using a contagion model," in *Conference of the Computational Social Science Society of the Americas*. Springer, 2021, pp. 56–70.

[53] P. Bevington and D. Robinson, *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill Education, 2003. [Online]. Available: https://books.google.co.cr/books?id=0poQAQAAIAAJ

[54] G. Catalin Balan, "MASON: A Java Multi-Agent Simulation Library," 2009.

[55] U. Wilensky and W. Rand, *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*, ser. The MIT Press. Cambridge: The MIT Press, 2015.

[56] J. Gosling, B. Joy, G. L. Steele, G. Bracha, and A. Buckley, *The Java Language Specification, Java SE 8 Edition*, 1st ed. Addison-Wesley Professional, 2014.

[57] S. Luke, *Multiagent simulation and the MASON library*. George Mason University (CC), August 2019.

[58] N. Collier, "Repast: An extensible framework for agent simulation," *The University of Chicago's social science research*, vol. 36, p. 2003, 2003.

[59] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko, "Complex adaptive systems modeling with Repast Simphony," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, p. 3, Mar 2013. [Online]. Available: https://doi.org/10.1186/2194-3206-1-3

[60] K. A. Barclay and W. J. Savage, "Groovy programming : an introduction for Java developers," Amsterdam, 2007. [Online]. Available: http://www.books24x7.com/marc.asp?bookid=47160

[61] E. Burnette, "Eclipse IDE pocket guide," Sebastopol, CA, 2005.

[62] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[63] J. Kazil, D. Masad, and A. Crooks, "Utilizing Python for agent-based modeling: The Mesa framework," in *Social, Cultural, and Behavioral Modeling*, R. Thomson, H. Bisgin, C. Dancy, A. Hyder, and M. Hussain, Eds. Cham: Springer International Publishing, 2020, pp. 308–317.

[64] G. Datseris, A. R. Vahdati, and T. C. DuBois, "Agents.jl: a performant and feature-full agent-based modeling software of minimal code complexity," *SIMULATION*, vol. 0, no. 0, p. 003754972110688, Jan. 2022. [Online]. Available: https://doi.org/10.1177/00375497211068820

[65] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *arXiv preprint arXiv:1209.5145*, 2012.

[66] P. K. Feyerabend, *Against Method*, 4th ed. London, England: Verso Books, 2010.

[67] M. Roza, J. Voogd, H. Jense, and P. Gool, "Fidelity requirements specification: A process oriented view," 10 2011.

[68] R. L. V. Horn, "Validation of simulation results," *Management Science*, vol. 17, no. 5, pp. 247–258, 1971. [Online]. Available: http://www.jstor.org/stable/2628978

[69] V. M. Yakovenko and J. B. Rosser, "Colloquium: Statistical mechanics of money, wealth, and income," *Reviews of Modern Physics*, vol. 81, no. 4, p. 1703–1725, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1103/RevModPhys.81.1703

[70] C. Gini, "Measurement of inequality of incomes," *The Economic Journal*, vol. 31, no. 121, pp. 124–126, 1921. [Online]. Available: http://www.jstor.org/stable/2223319

[71] J. Kleinberg and E. Tardos, *Algorithm Design*. Upper Saddle River, NJ: Pearson, Jun. 2005.

[72] T. D. Foundation, "Libreoffice calc," 2020. [Online]. Available: https://www.libreoffice.org/discover/calc/

[73] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," 05 1994.

[74] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Wadsworth Publishing, Chapman and Hall/CRC, 2002, 2024.

[75] J. Brenneman, "Experimental design," *J. Qual. Technol.*, vol. 52, no. 4, pp. 423–424, Oct. 2020.

[76] M. Fukuda, K. Wang, S. Panther, N. Wong, I. Dudder, and Q. Shao, "An analysis of three c/c++ cluster-computing libraries for agent-based models," *ACM Transactions on Modeling and Computer Simulation*, 2025.

[77] T. McPhillips, C. Willis, M. R. Gryk, S. Nunez-Corrales, and B. Ludäscher, "Reproducibility by other means: Transparent research objects," in *2019 15th International Conference on EScience (EScience)*. IEEE, 2019, pp. 502–509.

[78] V. Grimm, S. F. Railsback, C. E. Vincenot, U. Berger, C. Gallagher, D. L. DeAngelis, B. Edmonds, J. Ge, J. Giske, J. Groeneveld *et al.*, "The ODD protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism," *Journal of Artificial Societies and Social Simulation*, vol. 23, no. 2, 2020.

[79] M. van Steenbergen, R. Bos, S. Brinkkemper, I. van de Weerd, and W. Bekkers, "The design of focus area maturity models," in *Global Perspectives on Design Science Research*, R. Winter, J. L. Zhao, and S. Aier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 317–332.