Modeling Innovation Diffusion in sub-Saharan Africa – An Agent-Based Approach

Joël Inglin[a], Ann-Kathrin Kübler[a], Hannah Rohe[a]

[a]*University of Zurich, Switzerland*

Abstract

This master project presents an agent-based model (ABM) designed to simulate the diffusion of agricultural innovations among sub-Saharan smallholder farmers through specific interventions. It builds and expands upon the foundational work of Marc Zwimpfer's NetLogo model from 2022, whose capabilities were extended and generalized for broader application. Our enhancements inter alia include the incorporation of neighborhood dynamics and the concept of "independent farmers". Moreover, we have further refined the integration of variables for the BehaviorSearch optimizer to fine-tune intervention strategies effectively. Also, we propose specific default values based on unpublished data from research on sustainable food systems in Africa led by the University of Zurich at the Zurich Knowledge Center for Sustainable Development (ZKSD), ensuring relevance and feasibility of the optimization. Furthermore, we demonstrate how NetLogo models equipped with an optimization tool can be made easily accessible to researchers with varying levels of technical expertise using our suggested architecture - shipped in an isolated environment using Docker. The result is a versatile package and tool that makes ABMs and optimization accessible to researchers, supporting the evaluation and comparison of intervention impacts on innovation diffusion.

1. Motivation

Despite advancements, poverty and food insecurity persist, especially in regions like sub-Saharan Africa. Addressing this issue requires innovative solutions to improve the storage of farm products to minimize post-harvest losses. Such storage solutions should not only be affordable for the farmers but especially feasible for distribution and introduction by local entities. A pivotal question we address with our model is which strategy enables local research groups to most efficiently introduce these innovations, reaching key opinion leaders effectively and providing the best incentives to farmers. Therefore, our project endeavors to refine an existing model of innovation diffusion, enhancing its capabilities to support researchers in deploying effective strategies for delivering crucial innovations to agricultural sectors. We aim to equip semi-technically skilled researchers with a simplified, user-friendly tool, streamlining the application of Agent-Based Models (ABMs) and their optimization. Our goal is to facilitate these research groups in evaluating the impact of varied intervention strategies on disseminating information about agricultural advancements. Our report begins with an introduction of Agent-Based Models and their role in optimizing the diffusion of innovation. Following this, we describe the architectural framework that underpins our enhanced model. Subsequently, we take a closer look at our specific setup and application of this model to Tanzanian data. Our conclusion projects into future work, suggesting pathways for further research.

2. Agent Based Models and their Optimization

In Agent-Based Modeling, the modeler has to define decisions and interactions at an agent-level in contrast to other modeling approaches where macro-level phenomena must be directly modeled. This crucial step, happening before runtime involves creating rules for probabilistic

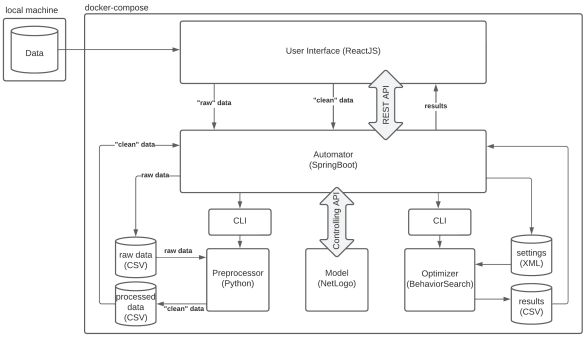


Figure 1: The Application architecture

behavior or establishing conditional rules for each agent.

During runtime, the earlier defined agents operate autonomously, guided by these predefined rules and probabilities. They engage in decision-making, interact with each other, and may adapt their behavior in response to the dynamics of the simulation. To mirror the unpredictability and diversity inherent in real-world scenarios, elements of randomness and variation are integrated into the model. This leads to the emergence of macro-level scenarios from the micro-level interactions of individual agents, thereby allowing for the exploration and optimization of defined outcomes [1].

Agent-Based Modeling also uniquely incorporates abstract time intervals, where each ”tick” in the model’s temporal framework can trigger specific predefined actions or procedures. This is particularly beneficial for implementing and assessing different intervention strategies over time, allowing for a dynamic modeling environment. We use NetLogo [2], an open-source language and environment specifically designed for agent-based simulations. Complementing this is BehaviorSearch [3], a powerful tool for the automated exploration and optimization of agent-based models, which is conveniently bundled with NetLogo. This creates a robust platform and ideal basis for our application architecture.

3. Architecture

The MIDisSa Application is built up of five major parts: The Automator (Backend), the User Interface (Frontend), the underlying Agent-based Model, the Optimizer, and the Preprocessor. Figure 1 shows how the architecture of the final application looks like. In this chapter each part will be briefly elaborated on.

3.1. Automator

The Automator is the backend of our application. It is is written in Java and utilizes the Spring Boot framework. Its main function is to connect the User Interface (section 3.2) to all the other components of the application. It does so by providing a REST API that can be accessed over HTTP.

The Automator provides an endpoint to access the Preprocessor (section 3.5). It expects a CSV file which it stores at a place where the Preprocessor can access it. The Automator then checks whether the file contains all of the necessary columns and values. It then starts the Preprocessor which stores its results in another CSV file. The Automator reads out the CSV file containing the results and returns them in a JSON format.

The parameters set by the user in the User Interface are represented as classes in the Automator. The classes contain a default value for each parameter. There are multiple endpoints that allow the readjustment of these parameters.

We integrated the ABM (section 3.3) into the Automator using the Controlling API [4]. The Controlling API is provided by NetLogo and allows models written in NetLogo to be invoked and controlled by a script or an application. It allows the adjustment of input parameters and the output of variables or parameters during and after running the model. The Automator uses the Controlling API to adjust the parameters of the ABM before running it and to read the relevant parameters after running it.

The Optimizer (section 3.4) can be controlled via the command line interface (CLI) and using XML files. The Automator stores necessary settings in an XML file that can be read by the Optimizer, then starts the Optimizer via the CLI, and reads out the results from the CSV file that the optimizer created, before returning them.

Whenever the Automator runs the ABM or the Optimizer, it stores the results in a continuous CSV file. The Automator provides endpoints which allow the download or reset of those files.

An overview of all endpoints provided by the Automator can be found in Appendix B.

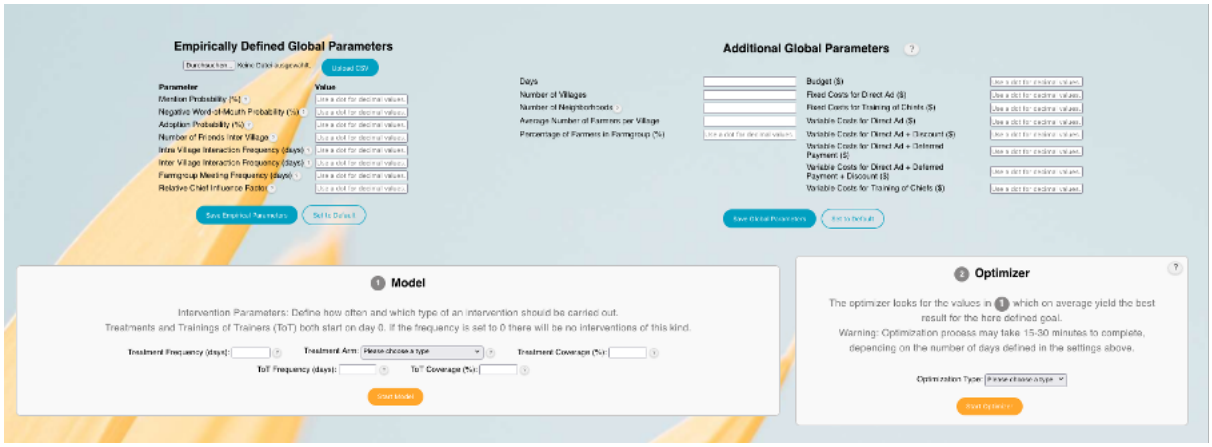


Figure 2: The upper part of the User Interface

3.2. User Interface

Our application's user interface, written in JavaScript and using react JS is utilizing REST API for backend communication. It is designed to be clean and straightforward and catering specifically to our key users – economic researchers with intermediate technical skills but a strong foundation in statistics. The User Interface allows for parameter input, offering users the flexibility to modify values as needed. However, to guide and orientate users, especially those less familiar with the model, we provide a set of default values.

Visualization features in the interface are oriented towards clarity and comprehension. There are no real-time visualizations, our app generates graphical representations of the model's output after each run, effectively illustrating the development over time measured in days. The User Interface provides real-time status updates to users, though. This indicates whether the model is actively running or idle. For the longer optimization process, which can take a considerable amount of time depending on the chosen parameters, the interface includes a progress indicator that displays a percentage update. This ensures that users are always aware of the model's current state and can approximate how much longer a process might take to complete.

For data analysis and further research, the interface enables the export of simulation results in a CSV file format, making it convenient for users to perform additional statistical analysis, data visualization or data manipulation using external tools.

The User Interface is further enhanced with help resources. Tool-tips are strategically placed to offer descriptions and insights into various parameters, making the navigation and understanding of the interface more intuitive. For detailed guidance and advanced usage, the interface references this report, particularly table B.4, providing users with a deeper understanding of the parameters and their capabilities. Additionally, we included an extensive example showcasing how to interpret optimization results. This ensures that users can effectively leverage the model for their decision making and research, regardless of their initial familiarity with the system.

3.3. ABM (NetLogo)

This part of the application contains the ABM which does the actual modeling. It is written in NetLogo [2] and connected to the Automator (section 3.1) via the Controlling API [4]. The Controlling API is provided by NetLogo and allows models written in NetLogo to be invoked and controlled by another application [4]. To make the ABM as compatible as possible with the Controlling API, we configured most of the parameters as sliders or input on the NetLogo graphical user interface. Only parameters that are defined that way can be adjusted using the Controlling API.

3.4. Optimizer

The Optimizer as its name says is responsible for the optimization process. It employs NetLogo's very own optimization tool called "BehaviorSearch" which comes with NetLogo and was specifically built for the optimization of ABMs written in NetLogo. BehaviorSearch relies on a specific structure of XML file to store detailed information about the planned optimization, hence the parameters set by the user in the User Interface are stored there as well. Each time an optimization process is started the XML file is updated with the current settings coming from the User Interface. After that, BehaviorSearch is launched in the background via a Shell-command referencing the updated file.

3.5. Preprocessor

The application architecture also includes a Preprocessor, a Python script integrated into the backend, designed to enhance the model's applicability to real-world scenarios. Specifically, this Preprocessor is a key component in incorporating unpublished data from research on sustainable food systems in Africa led by the University of Zurich at the Zurich Knowledge Center for Sustainable Development (ZKSD). It is an optional step and working with the provided application is also possible without the aforementioned research data. The preprocessing script transforms and analyzes the survey data to output the empirically defined global parameters, later used within the model.

3.6. Docker

Docker [5] is a platform that enables the creation and execution of images and containers. An image is a standalone, executable package that contains everything needed to run a piece of software, including the code and necessary libraries. These images can be shared, allowing others to run a container using the image. A container is a running instance of an image. Containers are isolated from the host system, allowing users to run the containerized application on any computer regardless of its operating system and without locally installing any software except for Docker itself.
To run multi-container Docker applications, Docker provides a tool called Compose. Compose is configured with a YAML file where the necessary containers are specified.

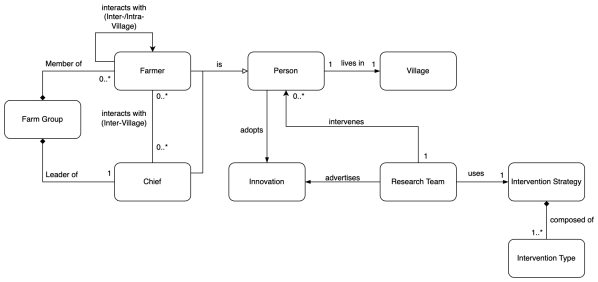


Figure 3: The model's simplified social system (Zwimpfer, 2022)

This allows a user to start all configured containers using a single command.
In our application we created one image for the automator, ABM, preprocessor, and optimizer and another image for the user interface. Both images are published on Docker Hub. In a YAML file, we reference these two images thus allowing users to start both of them using a single command and easily access our application on their local machine. With respect to our semi-technically skilled user group, we also wrote a short step-by-step instruction on how to download and install all necessary files, including a short troubleshooting section. It is a separate document, but its content can also be found in Appendix F .

4. Modeling Innovation Diffusion

4.1. NetLogo Model

The MIDisSa app employs an extended and generalized version of Marc Zwimpfer's NetLogo Model from 2022 [6] which was specifically developed for the purpose of simulating innovation diffusion in rural Tanzania. Figure 3 visualizes the basic structure of the model's social system: Each agent is assigned to a village and can have the role of either a farmer or a chief. Every village has one single chief which is the leader of the village's farmgroup. All other agents in the village are "ordinary" farmers which may or may not be members of the farmgroup. A research team then introduces the innovation via a modifiable intervention strategy (section 4.1.2) to a certain number of villages. Subsequently the innovation diffuses through agents interacting with each other. Depending on these interactions an agent regularly decides whether they want to adopt the innovation or not.

To help with understanding how the information diffusion process in Zwimpfer's model works we visualized it in a flowchart diagram. (Appendix D or in [7])

The following sections highlight the most important changes and adjustments made to Zwimpfer's original model. A more technical description of the additions and changes can be found in the GitHub wiki of the model repository [7]. An interested reader can learn more about the model's intricacies in Marc Zwimpfer's work "Diffusion of Innovation among Smallholder Farmer Households in Tanzania: An Agent-based Modelling Approach", 2022. [6]

4.1.1. Neighborhoods

The original model by Marc Zwimpfer already distinguishes between inter-village interactions (i.e. an agent communicates with another agent from the same village) and intra-village interactions (i.e. an agent communicates with another agent from a different village). Not everyone in sub-Saharan Africa has a phone, access to the internet, or even knows how to read and write. Therefore most of the communication is face to face and the assumption that people from adjacent villages are more likely to communicate is reasonable. To map this in the ABM, villages are divided up into so-called *neighborhoods*. When setting up the links between agents to model the intra-village interaction between them, it is more likely for an agent to be linked to another agent from the same neighborhood than to an agent from a different neighborhood.

4.1.2. Interventions

Interventions are a researcher's efforts to try to introduce an innovation to the agents. In the original model, researchers have two kinds of interventions at their disposal:

- Direct Ad
- Training of Trainers (ToT)

To build on that, the extended version of the model distinguishes between four different types of Direct Ads:

- Direct Ad: Standard Treatment with opportunity to buy innovation.
- Direct Ad + Discount: Direct Ad with discount on innovation prices.
- Direct Ad + Deferred Payment: Direct Ad but payments for innovation can be deferred until after harvest.
- Direct Ad + Deferred Payment + Discount: Direct Ad with both, discount and deferred payments.

Direct Ad	32%
Direct Ad + Discount	35%
Direct Ad + Deferred Payment	76%
Direct Ad + Deferred Payment + Discount	72%
Training of Trainers	88%

Table 1: Influence of interventions on likelihood of adaption.

An intervention strategy always consists of one type of Direct Ads and ToTs. The frequency as well as the coverage of both, Direct Ads and ToTs can be adjusted by the user. The frequency defines how many days elapse between interventions. The coverage sets the percentage of villages that will be part of the respective intervention. All these parameters can be adjusted by the user in the application's interface.
The different interventions vary in attractiveness to the agents because of different costs and effectiveness. We model this in the ABM by introducing a different influence level to each intervention. Table 1 shows the influences of the different interventions. The percentages that each intervention has on the likelihood of adaption are taken from observations of previous UZH-related studies in sub-Saharan Africa.

4.1.3. Budget and Costs

The third major change to the original model is the introduction of intervention costs and a budget. Intervention costs always consist of fixed costs and variable costs. Naturally, different types of interventions generate different costs. Reasonable default values are provided in the User Interface. The budget serves as an upper limit to how much money can be spent on interventions. Both, intervention costs and the budget can be adjusted by the user.
The addition of budget and costs was necessary because they introduce restrictions to how many interventions can

Risk seeking	2.5%
Risk inclined	13.5%
Normal	34%
Risk conscious	34%
Risk averse	16%

Table 2: Distribution of risk aversion among agents.

be carried out. This is integral to the optimization process since - without any restrictions - the Optimizer would carry out as many interventions as possible, regardless of costs. This way, the budget restriction forces the Optimizer to choose the most efficient and not the most expensive intervention strategy.

4.1.4. Risk Aversion

The original model attributed each agent with an adopter type. This attribution is based on Rogers [8] who defined five adopter types individuals can be categorized as. The main difference between those categories is, how quickly they adopt an innovation.
The original ABM used the adopter type of each agent to influence the mention probability, attitude change, and adoption decision of an agent. As there is no indication that Rogers categorization of adopter types can be carried over to rural smallholder farmers, we decided to replace the attribute with risk aversion wherever it made sense. As the mention probability is presumably not determined by the risk aversion of the agent, we decided to remove the influence of the adopter type and not replace it with anything else. A corresponding study about the risk aversion of smallholder farmers in sub-Saharan Africa is in planning. The result of this study is meant to replace the current distribution of risk aversion among agents, as it is currently still based on Rogers probabilities of adopter types. Table 2 shows the current distribution of risk aversion among agents.

4.1.5. "Independent" Farmers

The agents in the model are part of a farmgroup that is led by a chief and holds meetings regularly. The model assumes that agents are influenced in interactions with the chief of their farmgroup to a higher level than they are in interactions with regular agents. If the interaction with the chief contains the topic of the intervention, the agent's attitude toward the intervention is influenced more heavily. In the initial model by Marc Zwimpfer, all agents are part of a farmgroup. As this is not always the case in reality, we introduced a parameter to select the percentage of agents that should be part of a farmgroup. Agents that are not part of a farmgroup cannot be influenced by a chief and are therefore not influenced by the intervention strategy ToT (section 4.1.2).

4.1.6. Parameter

To model the complex relationships between agents and to account for a variety of different circumstances, the ABM defines several global parameters. Those can be categorized into parameters that can be set by the user, static parameters, and variables whose value is calculated within the model. To gain an overview and a better understanding of the ABM, we created a diagram of the composition of all parameters and interaction among them. The diagram covers only parameters, that were present in the original ABM by Marc Zwimpfer. Any parameters that were added in the extensions and adjustments mentioned in section 4.1 are not part of the diagram. In general, we enhanced the model to reduce the static parameters to a minimum, providing the most possible flexibility. All default values within the model are based and extracted from ongoing UZH research as described in the several paragraphs above. Many can be adjusted on the User Interface (section 3.2). The diagram can be found in Appendix E or in the GitHub repository of the ABM [7], which offers a digital copy for better zooming.
The table B.4 in Appendix A lists all parameters that can be adjusted by users on the user interface and gives a short explanation on what the parameter does in the model.

4.1.7. Bug Fixes

Additionally, when making ourselves familiar with the model we noticed a strange behavior when increasing the Negative Word-of-Mouth Percentage. This parameter increases the probability of having an interaction that negatively impacts an agent's attitude towards the innovation. However, the number of adopters did not drop, on the opposite, it sometimes even increased compared to a lower percentage. This was caused by a previously undetected bug in the original model that caused the agents' negative attitudes toward the innovation to be reset to neutral if the agent did not interact with anyone for at least one day

which is very common. We managed to fix the bug and the model is now behaving as should be expected: More negative interactions cause fewer adopters.

4.2. Optimizer

The Optimizer offers 3 different types of optimization:

- Max Adopters: Maximizes the number of adopters.
- Max Knowledge: Maximizes the sum of adopters and aware farmers.
- Min Costs: Minimizes the cost per adopter.

As mentioned in section 3.4, the Optimizer employs NetLogo's optimization tool BehaviorSearch and runs a Random Mutation Hill Climb algorithm. This type of algorithm starts off with a random solution to the problem. It then generates a slightly mutated, neighboring version of the solution and checks if it yields a better result. If so, the new solution is adopted and the process repeated. [3] Because of the model's natural randomness a possible new best solution is only accepted if the average of 10 runs still provides a better result.

For performance reasons, a maximum of 400 model runs per optimization are executed. For the same reason some of the parameters the user can set are limited: The number of days per run cannot exceed 720 days, the number of villages cannot exceed 100 villages and the number of farmers per village cannot exceed 20 farmers. Lower numbers and all other parameters are adopted from the user interface. These limitations still allow for most use cases while ensuring functionality and reasonable performance.

Even so, the user needs to be aware that the run-limit combined with the model's natural randomness might cause the optimizer to yield a result that is not the global but a local optimum. Though in practice, 400 runs have proven to be a good trade-off between performance and accuracy.

The Optimizer's output in the User Interface looks as portrayed in figure 4 and this specific example can be interpreted as follows:

To maximize adopters the Optimizer suggests to carry out two Direct Ad + Deferred Payment interventions in 95% of the villages combined with two ToT interventions in 94% of the villages. Direct Ad + Deferred Payment should be scheduled for day 0 and day 60, ToT for day 0 and day 90. This intervention strategy yielded the best results, averaging 652.7 adopters on day 360. The total cost of the executed interventions adds up to 187'600.

This example was run for 360 days with a budget of 200'000 and default cost values.

Optimization Results

Optimization Type	Treatment Arm	Treatment Frequency (days)	ToT Frequency (days)	Treatment Coverage (%)	ToT Coverage (%)	Best Fitness	Number of Treatments	Number of ToTs	Total Cost ($)
Max Adopters	Direct Ad + Deferred Payment	60	90	95	94	652.7	2	2	187,600

Figure 4: Example Optimizer output

4.3. Preprocessor

In this step we extracted important parameters from real data to assist in building our final ABM. As the model is set up to mimic the complex social interactions found in rural sub-Saharan farming communities, certain key parameters play a crucial role in showing how interaction behaviors affect the spread of new innovative farming methods, like using hermetic storage bags.

"ToT Adoption Probability" was quantified using logistic regression and expressed as a probability via a sigmoid function to reflect the potential impact on training of Chiefs (ToTs). We also looked at how often people in the village talk to each other and with people outside the village, known as "Interaction Frequency". Here, we excluded outliers and calculated the mean. Further, we assessed the prevalence of hermetic bags in those discussions, both within and beyond the village. Those values are stored in the "Average Mention Percentage" parameters.

Lastly, the Number of Friends outside the village was calculated as the average scale of a farmer's external network, which is critical for understanding inter-village communication patterns as well as key-information spreader.

We've made sure to also calculate the standard deviations for certain parameters to understand how much these behaviors can vary from one person to another. However, we're aware that NetLogo has some limits, like not being able to depict more complex statistics in our model, e.g. skewed distributions. Even so, the parameters we've determined provide solid starting points that let the model offer initial insights into how farmers communicate and might take up new agriculture techniques.

In addition to the previously mentioned parameters, our model integrates several other key metrics derived from ongoing but yet unpublished research affiliated with the University of Zurich. These include "Average Chief-Farmer Meeting Frequency", "Percentage Negative Word-of-Mouth", and "Base Adoption Probability".

All the parameters mentioned above offer dual utility in our modeling efforts. They can be directly implemented in the ABM to simulate the specific socio-economic conditions of the studied region. Alternatively, these parameters serve as a mental benchmark, providing a reference for adjustments when adapting the model to other, similar regions. This flexibility allows for the extrapolation of our model's insights to comparable settings, while also accommodating the unique characteristics of different rural sub-Saharan communities.

The Preprocessor only works, if the provided CSV file contains certain columns. Table C.5 shows which columns must be present and what types of values it must contain. In the GitHub repository of the Preprocessor [9] a template of the CSV file, including all necessary columns, is provided.

5. Future Work

We aimed to provide semi-technically skilled researchers with a simplified, intuitive tool, enhancing their use and optimization of ABMs. Our goal was to assist these groups in assessing the effects of diverse intervention strategies for spreading information on agricultural advancements. In this context, we believe that the application we have presented offers significant value. However, to further enhance its utility, we are considering several potential additions.

The graphical user interface of the NetLogo model itself provides a map, that visually represents the agents and villages. This dynamic map evolves while the model runs to graphically depict which agents have adopted the innovation. While the map does not introduce new information beyond what the application already presents, it enhances the user experience, makes the model more tangible and helps visually tracking the progress. It would be a beneficial addition to add such a map to the User Interface of the presented application.

To ensure the accuracy and relevance of our model, we have carefully selected parameter values based on data from relevant research. We tried to base the variables used in the model on numbers taken from relevant studies. The distribution of risk aversion among agents however is still based on Rogers probabilities of adopter types as explained above in section 4.1.4. There is a planned study on the risk aversion of smallholder farmers in sub-Saharan Africa and it would be extremely valuable to use the results of that study to adjust the distribution of risk aversion.

Furthermore, additional research could shed light into the extent to which conversations with a community chief might influence farmers' innovation adoption decision as compared to interactions with their peers. For instance, such studies could quantify the persuasiveness of community leaders and assess how their advice or opinions weigh against those of fellow farmers in decision-making processes. This could provide critical insights into social dynamics and leadership roles in rural agricultural communities.

Another area to explore would be enhancing the NetLogo model to adjust not only the mean but also the skewness of certain functions. Currently, this level of detailed manipulation is beyond the scope of what NetLogo's language can provide. Implementing such a feature would need substantial updates to the underlying Scala code. This enhancement could not only lead to a more accurate representation of data distributions, but would also offer researchers a more precise depiction of different scenarios, making it a more comprehensive tool.

The potential for future enhancements in our project also includes the Optimizer. It currently runs with specific settings that have empirically proven to be a good trade-off between performance and accuracy of results. Nonetheless, these settings could still be improved by applying meta-optimization to them. Meta-optimization means running a separate optimization algorithm that is looking for the optimal settings for another optimization process

that is dealing with the actual optimization problem. In short: Optimizing the optimization settings. In our case this might help fine-tune mutation rate, evaluation limit, encoding representation or other settings to further improve accuracy.
In conclusion, while our tool already stands as a robust and effective solution, these proposed future developments are geared towards further refining its capabilities, ensuring its continued relevance in research applications.

Acknowledgements

We would like thank Dr. Matthias Huss and Prof. Dr. Lorenz Hilty for their support throughout our master's project. Dr. Huss not only provided essential background information on research on sustainable food systems in Africa but also served as a critical test user, offering valuable feedback on the interface – an important perspective given his role within our key user group. Additionally, we are grateful to Prof. Dr. Hilty, head of the Informatics and Sustainability Research Group, for introducing us to the principles of simulations and agent-based modeling. His guidance on the technical architecture of our project was instrumental in ensuring its ease of use and accessibility for researchers.

Appendix A. Application Parameters

Table A.3: Application Parameters

Parameter name	Description	Default
ToT Adoption Probability	Defines the average probability of a chief adopting the innovation after receiving a ToT.	88%
Number of Friends Inter-Village	Defines the number of friends an agent has outside the village they live in.	5
Intra-Village Interaction Frequency	Defines the average number of days between intra-village interactions initiated by an agent. Does not prevent interaction if initiated by another agent.	4%
Inter-Village Interaction Frequency	Defines the average number of days between inter-village interactions initiated by an agent. Does not prevent interaction if initiated by another agent.	5%
Farmgroup Meeting Frequency	Defines the average number of days between farmgroup meetings.	30
Intra-village Mention Probability	Defines the average probability for the innovation to come up as a topic during an intra-village interaction.	2%
Inter-village Mention Probability	Defines the average probability for the innovation to come up as a topic during an inter-village interaction.	1%
Negative Word-of-Mouth Probability	Defines the probability of an interaction being unfavorable regarding the innovation.	10%
Adoption Probability	Defines the base probability of an agent adopting the innovation.	1%
Days	Defines for how many days the model should run. (Max. 720)	360
Number of Villages	Defines how many villages will be generated (Max. 100)	100
Number of Neighborhoods	Defines how many neighborhoods will be generated. If set to zero, there will not be any neighborhoods generated.	20
Average Number of Farmers per Village	Defines how many agents are assigned to a village on average.	10
Percentage of Farmers in Farmgroup	Defines percentage of farmers that are members of a farmgroup.	50%
Budget	Defines the maximum amount of money available to be spent on interventions.	100'000
Fixed Costs for Direct Ad	Defines the fixed costs for one Direct Advertisement Treatment.	6'000
Fixed Costs for Training of Chiefs	Defines the fixed costs for one Training of Trainers.	5'000
Variable Costs for Direct Ad	Defines variable costs for Direct Advertisement Treatment. Incurs once per village.	400

Variable Costs for Direct Ad + Discount	Defines variable costs for Direct Advertisement Treatment with Discount. Incurs once per village.	500
Variable Costs for Direct Ad + Deferred Payment	Defines variable costs for Direct Advertisement Treatment with Deferred Payment. Incurs once per village.	700
Variable Costs for Direct Ad + Deferred Payment + Discount	Defines variable costs for Direct Advertisement Treatment with Deferred Payment & Discount.	800
Variable Costs for Training of Chiefs	Defines variable costs for Training of Trainers. Incurs once per village.	400
Treatment Frequency	Defines the number of days between Direct Advertisement Treatments.	360
Treatment Arm	Defines the type of Direct Advertisement Treatment that will be carried out.	360
Treatment Coverage	Defines the percentage of villages that will be part of the Direct Advertisement Treatment.	50%
ToT Frequency	Defines the number of days between Training of Trainers.	360
ToT Coverage	Defines the percentage of villages in which Training of Trainers will be carried out.	50%
Optimization Type	Defines the objective which the Optimizer maximizes/minimizes for.	-

Appendix B. Automator Endpoints

Table B.4: Application Parameters

REST Mapping	Endpoint	Request Body	Return Body	Status
GET	/downloadOptimizationResultsCSV		CSV	OK - 200
GET	/downloadModelResultsCSV		CSV	OK - 200
POST	/clearResultCSVs			OK - 200
POST	/clearResultCSVs			Bad - 400
POST	/updateInput	DataInput (json)		OK - 200
POST	/updateInput	DataInput (json)		Bad Request - 400
PUT	/resetInput		workingDataInput (json)	OK - 200
POST	/updateGlobalInput	UserInput (json)		OK - 200
POST	/updateGlobalInput	UserInput (json)		Not Acceptable - 406
POST	/resetGlobalInput		workingGlobalInput (json)	OK - 200
POST	/results	UserInput (json)	ModelResults (json)	OK - 200
POST	/results	UserInput (json) (faulty)	ModelResults (json)	Not Acceptable - 406
POST	/results	UserInput (json)	ModelResults (json) (faulty)	Conflict - 409
POST	/optimization	UserInput (json)	OptimizationOutput (json)	Ok - 200
POST	/optimization	UserInput (json) (faulty)	OptimizationOutput (json)	Not Acceptable - 406
POST	/optimization	UserInput (json)	OptimizationOutput (json) (faulty)	Conflict - 409
POST	/uploadRawCSV	CSV File		Ok - 200
POST	/uploadRawCSV			Bad Request - 400
POST	/uploadRawCSV	non-CSV File		Unsupported Media Type - 415
GET	/optimizerBuffer		lastOutput (String)	OK - 200
PUT	/abortOptimization			OK - 200

Appendix C. Preprocessor CSV Template

Table C.5: Necessary columns and value types in Preprocessor CSV

Column name	**Value type**
Enumerator: Select Farmer Group type	[”Treatment group”, ”Control group”]
PART 2: STORAGE AND POST-HARVEST/What kind of storage methods do you use?/Hermetic bag	[0, 1]
PART 2: STORAGE AND POST-HARVEST/Have you received training on hermetic bag?	[”Yes”, ”No”]
PART 7: SOCIAL CAPITAL AND NETWORKING/What is your/partner’s role in the group/institution?	[”Member”, ”Leader”, ”Other”]
PART 7: SOCIAL CAPITAL AND NETWORKING/Think about the people you know in your village. In a typical week, how many times do you communicate with each person you know? The way of communication (in-person or phone) does not matter.	integer
PART 7: SOCIAL CAPITAL AND NETWORKING/Think about the last 10 discussions you had with contacts in your village. In how many discussions were hermetic storage bags mentioned?	integer
PART 7: SOCIAL CAPITAL AND NETWORKING/Now think about people you know that don’t live in your village. In a typical week, how many people outside your village do you communicate with? Include all relatives, friends, traders, extension officers and other people. The way of communication (in-person or phone) does not matter.	integer
PART 7: SOCIAL CAPITAL AND NETWORKING/Think about the last 10 discussions you had with contacts outside of your village. In how many discussions were hermetic storage bags mentioned?	integer

Appendix D. Information Diffusion Process

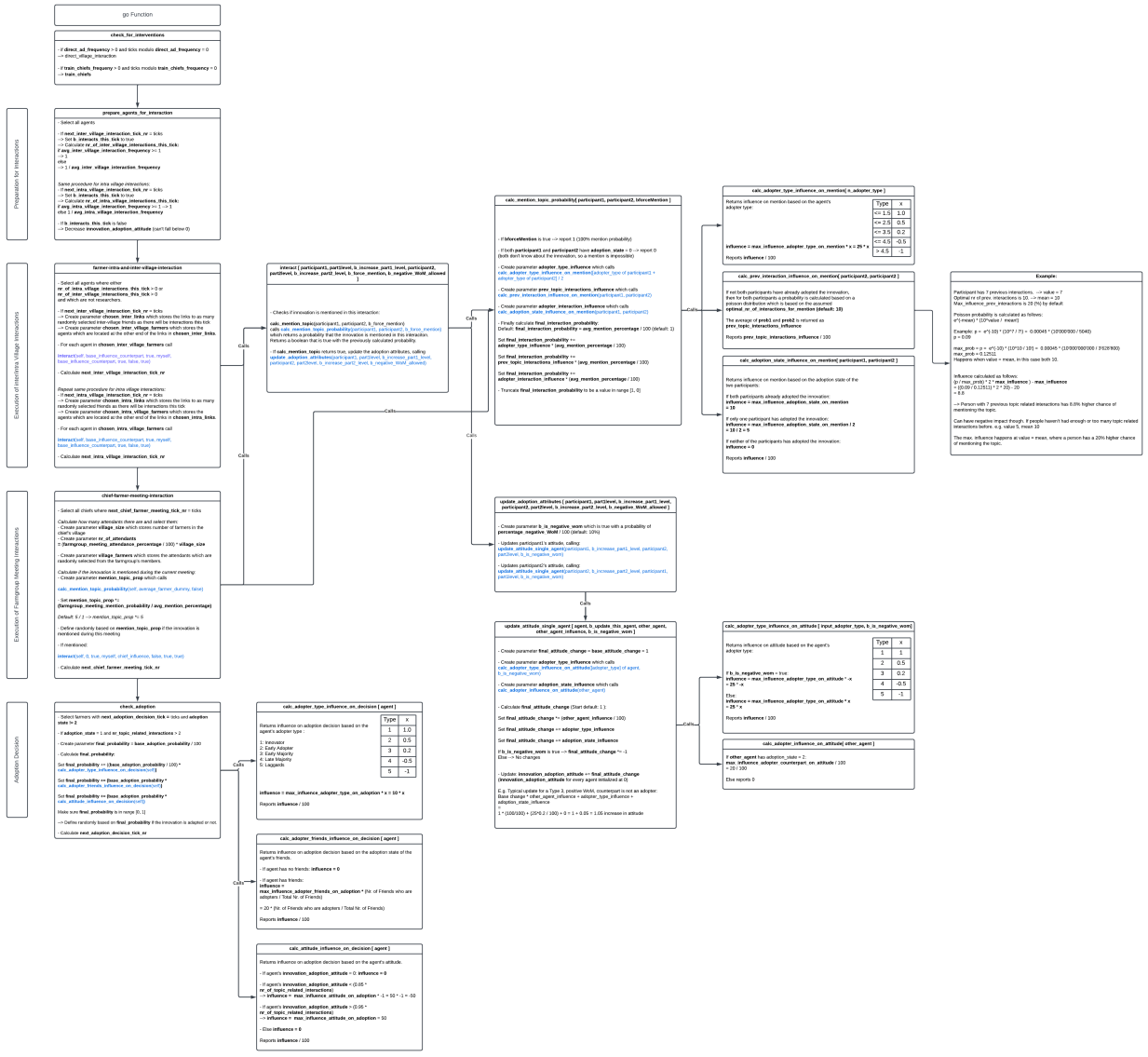


Figure D.5: Information Diffusion Process in the original ABM. A digital copy for further zooming can be found here [9]

Appendix E. Model Parameters

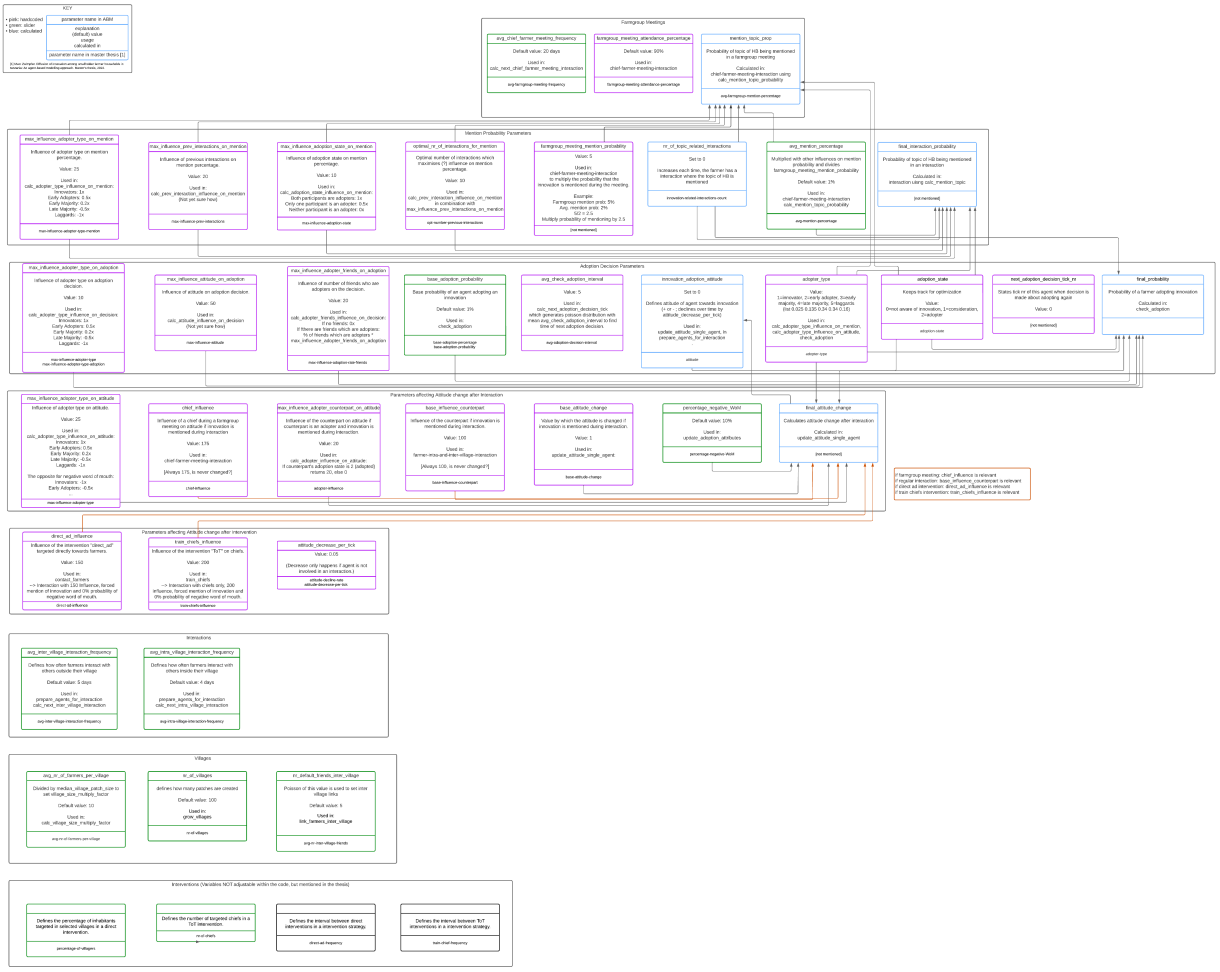


Figure E.6: Parameters in the original ABM. A digital copy for further zooming can be found here [9]

Appendix F. Instructions on how to install the Application via Docker

MIDisSA - Instructions

Joël Inglin, Ann-Kathrin Kübler, Hannah Rohe

General Description

The Application (here called "App") as part of the master project "Modeling Innovation Diffusion in sub-Saharan Africa - An Agent-Based Approach" (MIDisSa) mainly consists of two parts, the backend - also called the "automator" - and the frontend or "user interface". Backend as well as frontend are both stored in GitHub repositories and deployed in separate Docker containers. To facilitate running the App we rely on Docker Compose to start up both required Docker containers at once.

Where to download the App and how to install it is explained below.

Initial installation of the App via Docker

Docker

Docker is a platform that enables developers to deploy an application in so-called containers. The containers are isolated and independent from their environment which allows an application to be run regardless of the computer's operating system. The App's containers are stored at `https://hub.docker.com/u/abminnovationdiffusion` but there is no need to download them manually.

GitHub

GitHub is a platform for developers to store, manage and share code online in repositories. In our case we only need the "application" repository containing the Docker-Compose file which automatically downloads and starts up the correct Docker containers. However, all of our code can be found at `https://github.com/orgs/MIDisSa/repositories`

Run ABM

Step 1) To run the required Docker containers you first need to install the Docker Desktop app from `https://www.docker.com/products/docker-desktop/`). If clicking in the link does not work, copy and paste it in your browser. You might need to restart your device after installing the Docker Desktop app. It is then optional to sign up for Docker.

Step 2) Download the "application" repository from GitHub (`https://github.com/MIDisSa/application/archive/refs/heads/main.zip`). By clicking on the link, the folder should be downloaded automatically. If you face issues, click on the rear part of the link or copy and paste it in your browser. Afterwards extract the repository from the .zip-folder.

Step 3) Now that you've downloaded all the necessary tools, start up Docker Desktop.
If you have problems starting up Docker head to the **Troubleshooting** section below.

Step 4) On your device, open the command-line interface and navigate to the extracted "application-main" folder using the "cd" command. Make sure the folder you are in contains the "docker-compose.yml" file. You can check the contents of a folder using the "dir" command on Windows or the "ls" command on Linux/MacOS.

Step 5) In the command-line, execute the following command:

```
docker-compose -f docker-compose.yml up --build
```

The command tells Docker to download the necessary containers and start them up. This may take a few minutes.

Step 6) Open your browser and enter "localhost:3000" into the address bar. We recommend using either Firefox, Google Chrome or Safari.

After initial installation

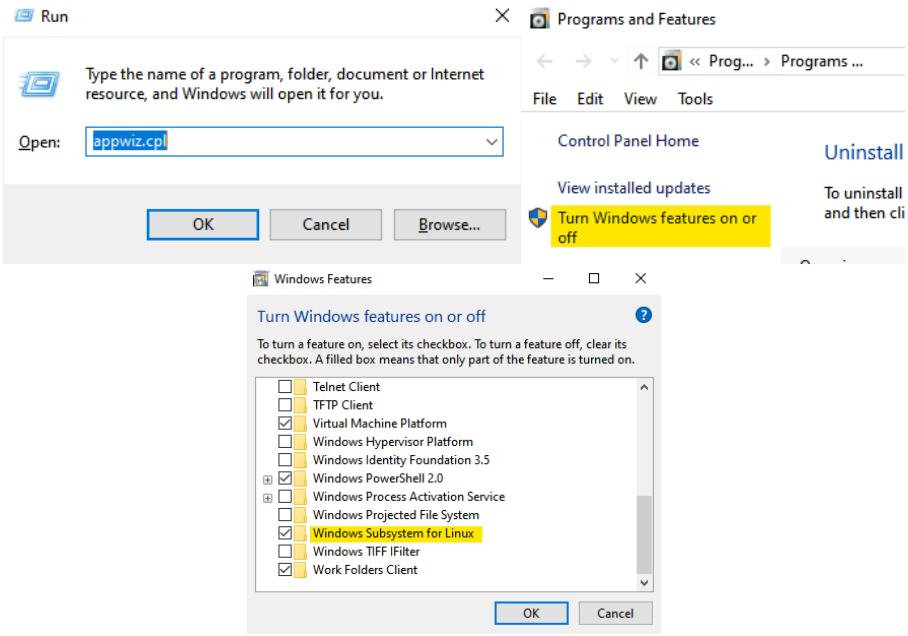
After you have successfully installed the App for the first time, starting it up again, becomes very simple. Start up Docker Desktop, navigate to the "Containers" and start the "application-main" container.
Finally, open your browser and enter "localhost:3000" again into the address bar.

Troubleshooting

If you have encountered an error when starting up Docker Desktop this might be because of one of Windows' default settings. Here are some steps you can take that hopefully help you to get Docker up and running:

WSL

Make sure Windows Subsystem for Linux (WSL) is enabled on your system. To do this, press Windows key + R, type "appwiz.cpl" and hit enter. Then navigate to "Turn Windows features on or off". Verify that "Windows Subsystem for Linux" is enabled. Then restart your PC.



Check if WSL is working by opening the commandline (Windows key + R, "cmd") and execute the command "wsl -l -v". The output should show NAME, STATE, and VERSION. If VERSION is not 2, run the command

```
wsl --set-default-version 2
```

Now try again to run Docker.

Virtualization

Make sure you have Virtualization enabled on your system. To do this, again press Windows key + R, type "appwiz.cpl", hit enter and navigate to "Turn Windows features on or off".
Make sure the following features are enabled:

- Virtual Machine Platform or VM-Platform
- Hyper-V (if existent)

Restart your PC and try running Docker again.

If there is still an error, virtualization might be disabled on your motherboard. To change that, access your system's BIOS/UEFI. How to access your BIOS is dependent on your system. Usually, this is done by restarting the PC and pressing F2 during start-up. Find the CPU-configuration in the BIOS within the advanced settings and turn on Virtualization. Now restart your PC and try again.

References

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

[2] NetLogo Home Page — ccl.northwestern.edu. `https://ccl.northwestern.edu/netlogo/`. [Accessed 10-01-2024].

[3] F. Stonedahl and U. Wilensky. *BehaviorSearch*. Center for Connected Learning and Computer Based Modeling, Northwestern University, Evanston, IL, http://www.behaviorsearch.org, 2010 [Accessed 11-01-2024].

[4] Flynn049. Controlling API — github.com. `https://github.com/NetLogo/NetLogo/wiki/Controlling-API`. [Accessed 10-01-2024].

[5] Home — docs.docker.com. `https://docs.docker.com/`. [Accessed 16-01-2024].

[6] Marc Zwimpfer. Diffusion of innovation among smallholder farmer households in tanzania: An agent-based modelling approach. Master's thesis, 2022.

[7] J. Inglin, A.-K. Kübler, and H. Rohe. model. `https://github.com/MIDisSa/model`, 2024.

[8] Greg Orr. Diffusion of innovations, by everett rogers (1995). *Retrieved January*, 21:2005, 2003.

[9] J. Inglin, A.-K. Kübler, and H. Rohe. preprocessor. `https://github.com/MIDisSa/preprocessor`, 2024.