Accessible Data Mining for Agent-Based Simulation Models

Robin Faber

Technische Universiteit Delft

TU Delft

Accessible Data Mining for Agent-Based Simulation Models

by

Robin Faber

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on the 12th of July 2021.

Student number:	4560175	
Project duration:	November 1, 2020 – July 12, 2021	
Thesis committee:	Dr. N. Yorke-Smith,	TU Delft, supervisor
	Dr. ir. I. Nikolic,	TU Delft, supervisor
	Dr. ir. U. Gadiraju,	TU Delft

Abstract

Agent-based simulation models are rising in popularity recently due to their ability to model real-world problems in a wide range of domains. Inherent to these types of simulations is the fact that an enormous amount of data can be generated, which needs to be analysed in order to make the simulation useful. At the moment, the available tools to perform this analysis are limited for most platforms in terms of functionality or usability. Model designers are often not experienced programmers, so processing and analysing the data from these experiments in Python or R can be challenging.

The goal of this master thesis is to determine the best way to make data mining methods accessible to model designers so they can easily execute and analyse model experiments for NetLogo. In order to achieve this goal, the limitations of the currently available tools will be established by communicating with the target group. When there is a clear overview of what exists and what still needs to be done, we will attempt to develop an accessible tool that addresses the current limitations. The tool that was created consists of a GUI that allows the user to design experiments, run simulations and visualise results. With this GUI, the coding aspect of data analysis is taken completely out of the users hands, so anyone with NetLogo experience should be able to conduct extensive analysis of their model and its results.

To determine whether the tool achieved its goal of being easy to use while still providing advanced analysis options, a usability study will be conducted with the target group. In this study, the users will execute a certain number of tasks with the tool and the results are recorded. After the tasks are completed, the user will fill in a questionnaire to get more opinion based data to give a complete overview of the usability. This usability study will allow us to determine whether our tool is indeed an improvement for the field and something that people could see themselves actually using during model development.

Every measurement goal that was set before the usability study was achieved, most of them with wide margins. The ASQ questionnaires obtained an average result of 4.69 out of 5, where the target was set at 4. Furthermore, the SUS questionnaire obtained an average result of 85 out of 100, with the target being 75. These results show that developing an accessible data mining tool is possible and thus that the goal of this thesis was achieved. This research can be used as a basis for future development in order to provide the agent-based simulation community with more tools for the analysis of models.

Preface

This document contains the Master Thesis report about accessible data mining methods for agent-based simulation models. This thesis has been written to fulfill the graduation requirements for the Computer Science Master program at the TU Delft. The project was executed in cooperation with the TU Delft, in order to provide relevant and interesting research that can be built on in the future.

I would like to thank my two supervisors Neil Yorke-Smith and Igor Nikolic for the guidance, advice and support I received during this thesis. I would also like to thank everyone in the modelling community that helped me by filling out the survey, participating in the usability study and giving me important feedback that helped shape this thesis. Lastly, I would like to thank my friends and family for always supporting me and providing me with some much needed distractions whenever I needed them.

I hope you enjoy the read.

Robin Faber
Delft, July 2021

List of Figures

Contents

1 Introduction

An agent-based model simulation (ABMS) [97] is an approach to run a simulation in a certain environment with the use of agents, which behave according to a user specified model [59]. ABMS offers a straightforward method to model the behaviour of individual agents and analyse their interaction with each other and the environment they are in. Its ability to model many real-world scenarios in society and nature has lead to widespread use of the technology [45]. Some examples of applications include modelling agent behaviour in biology [24], the stock market [4], the spread of epidemics [33] and manufacturing [12].

At the core of ABMS lies the environment and the agents it contains. During a simulation, the agents and environment behave in a way as specified by the model [16]. This behaviour includes interactions between agents, interactions between an agent and the environment, or changes in the environment itself. The model runs according to a specified time unit known as a tick. Every tick, the agents perform their actions and the environment and variables update accordingly. The values of the environment, agents and variables can be obtained with reporters, which are commands that access and return these values. The reporter values are the main source of the data used for model analysis.

The modelling language that will be considered during this thesis is NetLogo, which is a free open-source programming language and modelling environment [95]. NetLogo can be used to simulate natural and social situations to gain a meaningful insight without analysing these situations in the real world. NetLogo is commonly used by modellers due to the easy to use interface, large collection of sample models and support of concurrency. NetLogo also includes a tool for experimental design called BehaviorSpace. This tool allows the user to run a model with a wide range of parameter settings to study the models behaviour. The BehaviorSpace is currently the easiest way to design and run NetLogo experiments.

1.1. Problem Statement

Due to the nature of simulation studies, specifically ABMS, an enormous amount of data is generated and stored during a simulation [9]. The simulation has to keep track of the global variables, the properties and location of all agents and their interactions for every tick in the simulation. Depending on the model, there can be hundreds of agents active at one time and the simulation could last for hundreds of ticks. In recent years, the quantity of generated data has been increasing due to the increase in computational capability and the development of more complex models [50]. These data sets contain useful information that can be used to gain a meaningful insight into the behaviour of the model and its parameters [70].

The large data sets generated by simulation studies are only useful if they can be effectively analysed, for example with the use of data mining (DM). However, data mining the output of ABMS is an underexplored field at the moment. In the past, some efforts have been made to combine standard DM technologies with ABMS by Remondino and Correndo in [77, 78]. However, this research was proposed more than a decade ago and the problems discussed in those papers are still present today. Furthermore, these papers only discuss the theoretical possibilities of DM in ABMS rather than usable platforms or implementations, so the actual contributions of these papers are limited. Being able to successfully combine these two fields of research could contribute towards a solution for model validation, which is the process of determining whether a simulation model represents the real-world application with sufficient accuracy [7]. Model validation is an open problem in the modelling community and being able to effectively use DM to analyse models could provide the first steps towards solving this problem [11].

Many agent-based modelling platforms include basic analysis tools, but these are typically not sufficient to meet the requirements of a comprehensive analysis and documentation process [41]. In order to access the DM tools that other languages such as Python or R provide, the data needs to be shared between the simulation and the analysis platform, for example by writing and reading data files. This process is known as batch simulations. For larger simulations with complex models however, this process can quickly become infeasible, both in terms of computational power and the efficiency of the development process itself. Since these batch simulations are cumbersome to execute and analyse, model designers often only perform them at the end of the development process, which can limit the model development, testing and understanding [94].

Along with the computational limitations, there is also a large problem with the use of external data analysis tools themselves. The most common programming languages for data analysis are MATLAB, Python and R [69]. Being able to use these languages effectively requires programming experience, which is often lacking for model designers with a background in social science, political science or biology. NetLogo is built to be extremely easy to learn and use, especially when compared to other ABMS platforms such as RePast or Swarm [81]. It only takes a few hours to learn the basics in NetLogo, while it can take multiple days or even weeks for the other platforms. This is an obvious advantage for NetLogo itself, but it can also result in the user being reluctant to learn other technologies or programming languages. Programming can be extremely difficult to learn, certainly if it is not the main focus of someone's research [42]. Attempting to use advanced data analytics tools with little to no programming experience is often overwhelming for the user, resulting in them giving up and staying with the NetLogo analytics they are familiar with [80].

1.2. Research Questions

In order to solve the problem stated above, a research question needs to be defined such that the answer to this question provides a solution to the problem. In this case, the main problem is that there is a severe lack in DM tools for ABMS that are accessible to inexperienced programmers, so our research question will be defined as: *How can data mining tools be made accessible to inexperienced programmers in order to analyse NetLogo simulations?*

Since this question is too broad to solve directly, a set of sub-questions need to be defined that will serve as building blocks towards solving the main research question. For this thesis, a total of four sub-questions are defined as follows:

1. What type of data can be collected from the simulations?
2. Which type of visualisations are the most effective to present the data?
3. What is lacking with the currently available data analysis tools?
4. How can the tool be implemented such that it can be used by people with little programming experience, e.g. model designers?

1.3. Contributions

The four main contributions made in this thesis can be summarized as follows:

1. A survey of the state-of-the-art regarding DM tools and techniques for ABMS.
2. An accessible and easy to use interface that allows users to design, execute and visualise NetLogo experiments. This tool is publicly available at: `https://github.com/robinfaber97/NetLogoDOE`.
3. A reproducible usability study to formally determine the usability and ease of use of the tool for the ABMS target group.

4. A set of guidelines for the development of accessible DM tools for ABMS.

These contributions are relevant because each of them provides a step towards solving the problem described in Section 1.1. The survey of state-of-the-art methods will allow us to research the limitations of the previous research done in this field. The interface and usability study provide new research that hopefully addresses these limitations and provides a useful alternative to them. Finally, the set of guidelines will answer the main research question of this thesis and provide a basis for future research and development.

1.4. Chapter Overview

The rest of this thesis is structured as follows. Chapter 2 will give an overview of the related work, as found in the literature study done at the start of the thesis. In Chapter 3, the approach that was used to answer the research questions will be discussed in detail. The implementation of the tool and its features, along with motivation for certain design choices will be explained in Chapter 4. Chapter 5 will describe the usability study that was performed with the target group. The results of the usability study will be presented in Chapter 6. In Section 7, the results will be discussed with respect to the reliability, impact and relevance. Finally, the conclusion of the thesis and interesting directions for future work will be given in Chapter 8.

2 Related Work

This chapter will give an overview of relevant research found in the literature study at the start of the thesis. This literature study attempts to identify what has been done in the field of DM in ABMS, what techniques can be built on and what still needs to be done. Section 2.1 will discuss DM techniques that are relevant within the context of ABMS. In Section 2.2, ABMS platforms and their functionalities will be described. Finally, an overview of the currently available tools for NetLogo will be given in Section 2.3.

2.1. Data Mining Techniques

This section will give an overview of the relevant DM techniques found in literature. Only techniques that are relevant within the context of ABMS will be discussed, since the field of DM is too broad to be considered in its entirety. The three techniques that will be described are parameter optimisation, sensitivity analysis and data visualisation.

2.1.1. Parameter Optimisation

When developing a model, one of the most impactful aspects is the values of the parameters of the model. The values of the parameters can have an enormous effect on how the model behaves and thus the results it obtains [23]. In order to find the most suitable parameters for a model, parameter optimisation is used.

Burrows et al. [20] present a method that uses clustering techniques to determine equivalence classes of parameter settings of models. Because a parameter space increases exponentially in the number of parameters, exploring the entire space can quickly become infeasible. By clustering similar parameter settings together and choosing a single representative, the search space can be decreased drastically and thus results in a more efficient search during parameter optimisation.

In recent years, many research studies have tried to integrate optimisation technology with simulations in order to find the optimal parameters in a certain setting, which was considered the biggest challenge in simulation studies for a long time. In order to use the optimisation technology however, optimisation variables first need to be defined. Brady and Yellig [17] present an approach for parameter optimisation for simulation models (not necessarily agent based). It provides information concerning the dependencies between input variables used in the model, which can then be used as the basis for selecting optimization variables.

Better et al. [13] give an approach that is effective in guiding the search for optimal values of input parameters to a simulation model. The approach uses DM along with state-of-the-art optimisation technologies. Most of the approach revolves around dynamic DM however, which might be out of the scope of this thesis. Nevertheless, the approach given in this paper could still be a useful methodology to give insights into the optimal parameter values for the model designer.

2.1.2. Sensitivity Analysis

Sensitivity analysis (SA) is an important method during the analysis of ABMS. The main goal of SA is to determine which input parameters are the most influential in determining the output values, along with studying the interaction between different parameters [40]. SA can be applied for many different reasons, including model verification, validation and simplification. Because the effect of parameters on the output and the interaction between parameters is often non-trivial, being able to efficiently perform SA can provide enormous insight into the inner workings of a model. Patel et al. [73] describe an approach which uses a combination of exploratory, SA and DM techniques to understand data generated from an agent-based model. The combination of these technologies allows them to get a good understanding of the model's behaviour and the sensitivity of different parameters.

2.1.3. Data Visualisation

One of the most important aspects of DM is the way that the output data is shown to the user, known as data visualisation. Effective data visualisation is extremely important because it allows users to interpret the data and use it to improve their model or draw conclusions [55]. If the visualisation is not done properly, the data, and thus also the simulation itself, will be meaningless. In order to find useful data visualisation methods for ABMS, we looked at the results chapters of papers in which NetLogo models were presented, as well as a literature survey of data visualisation methods. This gives us an indication of the visualisation types that are used in the field and which will be the most useful for the users. The most common and useful visualisations obtained from these papers are: timeseries plots [3, 8, 68, 82, 86], boxplots [86], histograms [61, 68, 86], heatmaps [64], spatial plots [8, 61, 82], scatter plots [3, 8, 61] and parallel coordinates [3]. The visualisations that will actually be used in the tool are largely dependent on which graphing library will be used, but this does give us a good overview of the possibilities of data visualisation within ABMS.

2.2. Agent-Based Modelling Platforms

There are many ABMS platforms that modellers can choose from other than NetLogo [65]. Exploring these platforms adds context to the rest of this thesis by giving an overview of the analysis tools available for other platforms. This allows us to determine the general approach to model analysis in the ABMS community as a whole, as opposed to only considering NetLogo itself. The following sections will discuss three of these platforms that are commonly mentioned in literature [1].

2.2.1. MESA

MESA is not a separate ABMS platform, but rather a framework implemented in Python [46]. MESA is most suited for 2D visualisations, similarly to NetLogo [61]. It provides the user with an interface in the browser that shows the simulation and its visualisations, which makes it easy to monitor the simulation. Due to the framework being implemented in Python, the user is required to have a basic understanding of the language, which makes the framework less accessible than NetLogo. However, this is a large advantage of MESA for the people that manage to learn Python, since both the simulation and analysis can be done within the same environment. Furthermore, MESA provides better performance on complex models and a more detailed overview of the interactions in the simulation [91]. In conclusion, MESA is not as easy to use for people without a programming background, but is certainly an excellent alternative to NetLogo for people that are willing to take the time to learn Python.

2.2.2. GAMA

GAMA is a stand-alone simulation platform that is mostly targeted towards complex models that require spatial visualisation with a high level of detail [89]. This detail is achieved by supporting Geographic Information Systems (GIS), which are systems that store, retrieve, analyse and display geographical data [58]. With these GIS, GAMA is able to represent large-scale complex systems in a realistic way that makes the model easier to understand and analyse. GAMA provides an IDE for model development, a GUI for model simulations and built-in visualisations for model analysis [30]. The models are built using the GAML programming language, which is easy to use even for people with low programming skills [35]. GAMA attempts to provide the ease of use of NetLogo, while still providing more functionality and performance capabilities than other modelling platforms.

2.2.3. Repast

Repast is an ABMS toolkit implemented in Java. Repast provides many functionalities for simulation, but model development is very difficult with the tool. The user needs to be experienced with Java programming, along with Java paradigms such as object-orientated programming and constructors [35]. Once the model is developed, a GUI is provided from which the simulation can be controlled, which does increase the ease of use [92]. During the simulation, all the data is collected, but the visualisation options for this data are limited [67]. Similar to NetLogo, there are some standard 2D visualisations such as histograms and timeseries, but there are no advanced analytics available to do an in depth analysis of the model [66]. In conclusion, Repast provides very advanced model development, but the toolkit is extremely hard to use for beginners and the analysis options are limited.

2.3. NetLogo Experiments & Analysis

This section will give an overview of experimental analysis tools for NetLogo that are currently available. These tools provide the functionality that is needed for advanced model analysis, but are often not accessible to inexperienced programmers. Some of these technologies do not have a research paper dedicated to them, so a footnote is used instead.

2.3.1. PyNetLogo

NetLogo lacks the tools needed to run and analyse experiments on a large scale. For this reason, Jaxa-Rozen and Kwakkel [41] developed PyNetLogo, which is a library that allows NetLogo models and commands to be run from Python. All the agent data is stored by the library and can be accessed in Python for analysis. With PyNetLogo, the modelling capabilities of NetLogo can be combined with the wide variety of data analytics tools available in Python without compromising on the advantages of either language. The main disadvantage of PyNetLogo is that it only provides the functionality to connect to NetLogo. The experimental design, data processing and data visualisation all have to be implemented manually, which can be difficult and time-consuming.

2.3.2. EMA Workbench

EMA Workbench[1] stands for Exploratory Modelling and Analysis and provides a research methodology that uses experiments to evaluate complex systems. EMA Workbench provides support for modelling in various languages, including a connector from Python to NetLogo. The library is built on top of the functionality of PyNetLogo and addresses the main limitations of PyNetLogo given in the previous section. EMA Workbench provides functionality for experimental design, multiprocessing and a large set of built-in visualisations. In terms of the functionality, EMA Workbench provides everything that is needed to do a comprehensive analysis of NetLogo models. However, the library requires considerable experience with Python programming and data processing, which limits the accessibility.

2.3.3. NLexperiment

NLexperiment[2] is a library written in R that provides extensive tools to design, execute and visualise NetLogo experiments from R. The usage of it is similar to EMA workbench and provides implemented methods for parameter optimisation and sensitivity analysis. Because NLexperiment is implemented in R and we are using Python for this thesis, we will not be able to use this library directly. However, it does give a good idea of what experiment types and visualisations can be useful.

2.3.4. OpenMOLE

OpenMOLE is a platform that provides a convenient way to explore models from various modelling languages by designing experiments [79]. The biggest advantage of OpenMOLE is the parallelisation that it supports by running the experiments across distributed machines. OpenMOLE also provides a GUI that gives a clear overview of all the files, experiments and executions of a project, which is good for the ease of use and accessibility [72].

The experiments are defined by using the OpenMOLE programming language, which is an extension of the Scala programming language [79]. This is a disadvantage of OpenMOLE, because having to learn a new programming language just to run experiments is a huge barrier for many NetLogo users. Another disadvantage is the fact that OpenMOLE does not provide any tools for the analysis of the

[1]https://emaworkbench.readthedocs.io/en/latest/index.html
[2]http://bergant.github.io/nlexperiment/index.html

experiments, such as visualisations. This means that once the output is obtained, the user would still have to use another programming language in order to actually analyse it, which is not optimal.

2.3.5. SALab

SALib[3] is an open source Python library that provides methods to perform SA. The library includes implementations for many common SA methods such as Sobol [85], Morris [22] and Fourier Amplitude Sensitivity Test (FAST) [84], among others. This library can not be used to design or execute experiments with NetLogo, but it can be used to analyse the data obtained from a simulation. Being able to determine the sensitivity of the system in terms of its parameters gives a meaningful insight into the model. This information can then be used to draw conclusions from the simulation or to optimise the parameters to improve the performance.

2.4. Discussion

The previous sections have given an overview of the current state-of-the-art regarding DM in ABMS, including both the theoretical methods and the implementations of these methods. In Section 2.3, five options are given that provide tools for the analysis of NetLogo experiments. These packages provide a good overview of the current possibilities that users have regarding the analysis of NetLogo models. Two of the most important techniques used during model development and analysis are parameter optimisation [63] and sensitivity analysis [93]. This means that they should certainly be included in any DM tool for ABMS.

EMA Workbench and NLexperiment provide these functionalities by giving the user access to customisable experimental design and extensive visualisation options. The accessibility of these two tools is very low however. Both of them require the user to be quite experienced in their respective programming language, Python and R. Experiments are designed, ran and visualised by programming with the various classes, parameters and data analysis tools that the libraries provide. This is different from the NetLogo BehaviorSpace environment, where experiments can be designed by simply inputting values in a GUI. I suspect most NetLogo users would not be able to use EMA Workbench and NLexperiment effectively without extensive practice, guidance and documentation.

OpenMOLE provides the accessibility that EMA and NLexperiment are lacking by providing a GUI that the user can use to design and run experiments. The user still needs to program a little in the OpenMOLE programming language, extended from Scala, but there is extensive documentation and examples on the website and only the basics in Scala are needed. This is much more accessible than the other tools which require extensive programming experience, both in the design and analysis of experiments, and have limited documentation. The main disadvantage of OpenMOLE is the fact that it only provides functionality to design and run the experiments, not to analyse or visualise them, similar to the NetLogo BehaviorSpace. This means that the user would still have to defer to external tools for advanced data analytics, which results in the same problems as mentioned before.

In conclusion, these tools provide good alternatives to the NetLogo BehaviorSpace for the analysis of experiments in their own context. However, each of them still lack in critical aspects, either functionality or accessibility. These problems are not exclusive to NetLogo, since the platforms mentioned in Section 2.2 exhibit the same limitations. MESA and Repast are difficult to learn and use, while GAMA has limited capabilities to perform an advanced analysis of a given model. Being able to develop a DM tool that provides both functionality and accessibility to a large user base could contribute to a concrete implementation of the theoretical DM methods mentioned in the literature [11, 77, 78]. This implementation could be used to improve model development, testing and understanding and potentially help solve the open problem of model validation for ABMS.

[3]https://salib.readthedocs.io/en/latest/index.html

3 Approach

This chapter will describe the general methodology that was used to answer the research questions. After the literature study, there was a good indication of the current gap in the field of ABMS, but the way to fill this gap was not apparent yet. For this reason, a survey with questions about the current state-of-the-art was made, which will be discussed in detail in Section 3.1. From this survey, along with the literature study, the set of requirements for the tool was created, shown in Section 3.2. Finally, in order to determine whether the tool is successful in answering the research question, a usability study will be performed with the target group. In Section 3.3, a high level description of the usability study will be given, while the detailed explanation and results of it will be discussed in Chapter 5.

From this overview, it is clear that a large part of the approach revolves around interaction with the target group, both in the design and evaluation of the product. This practice is known as User-centered design (UCD), which is a term used for processes where end-users have a large influence on the design and development process [2]. This design strategy was used because this thesis attempts to solve an open problem in the modelling community. The goal is to develop something that this community deems useful and considers an improvement on the currently available tools. In order to achieve this goal, the input of the target group is extremely useful because it gives a good idea for the direction of the thesis and ensures that the usability goals are met.

3.1. Target Group Survey

UCD ensures that the goals and needs of the end-users are the main focus of the development process [18]. In order to ensure this, the goals and needs of the target group first need to be determined. There are many strategies to establish these needs, including focus groups, interviews, user surveys, personas and field studies [60]. Most of these options were infeasible for this thesis due to time restrictions and limited access to end-users, so a user survey was the best option to provide us with easy and useful communication with the target group.

3.1.1. Survey Questions

In order to make the survey as useful as possible, the questions were defined in such a way that they provided information about the four research sub-questions. By answering these research questions as early as possible, a solid foundation is built from which we can work towards the main research question. The survey consisted following questions (demographic questions such as name/email are omitted here):

1. What is currently preventing you from using external data analysis tools?

   Motivation: To get an idea of what is currently limiting the target group from using existing tools, so we can address these limitations during the development of the tool. Related to research question 3.

2. Which requirements would you define to make the tool easy to use for non programmers?

   Motivation: To get an idea of what exactly would be considered easy to use by the target group, since accessibility is one of the main goals of this tool. Related to research question 4.

3. What type of information would you like my tool to present? (e.g variables, locations, agent counts)

   Motivation: To get an idea of what type of information from the simulations people are interested in analysing. Related to research question 1.

4. How should this information be presented? (e.g, types of graphs/tables/visualisation)

   Motivation: To get suggestions for visualisations of the simulation output that the target group would find useful. Related to research question 2.

3.1.2. Survey Results

The survey was distributed in a few different ways, including the NetLogo and CoMSES forums and various mailing lists from journals related to simulation, such as JASSS, ESSA, RofASS and SIMSOC. In the end, a total of 6 people filled in the survey. The complete set of responses to all questions is shown in Appendix A. Six responses does not seem like a lot, but the information obtained from these answers did provide a good insight into the needs of the target group. For example, from the answers to the first two questions, it could be concluded that most people did not want to learn a new programming language in order to perform experiments. Furthermore, many people mentioned that a graphical user interface (GUI), something similar to the NetLogo BehaviorSpace, would be a good option to make the tool easy to use. This information was useful because it showed that making a tool where the user has to program in Python, which was the original plan, would most likely not be accessible to a large part of the target group. The answers to the last two questions were mostly used as inspiration for the data visualisation aspect of the tool. These answers included graph types like scatter plots, timeseries, histograms and heatmaps, which were largely similar to the visualisation types identified in the literature study.

3.2. Requirements

Requirement engineering is one of the most important aspects of the entire software development process [71]. For the tool developed during this thesis, the requirement engineering framework for UCD developed by Zimmermann & Grötzbach is used [99]. This approach was chosen because it focuses heavily on the usability of the product and is specialised towards the development of GUI's. The framework classifies three types of requirements: usability, workflow and user interface (UI). The usability requirements focus on the end-user and attempt to provide a reliable baseline for the effectiveness, efficiency and satisfaction of the tool. The workflow requirements focus on the actual tasks the user performs, which mostly includes the functional requirements of the product. Finally, the UI requirements describe the physical properties of the system, including the way the widgets, screens and interaction elements look, behave and interact. Using this framework, the set of requirements is defined as follows:

Usability Requirements:

- The tool is easy to use.
- The tool is intuitive to use.
- Users learn to use the tool quickly.
- Users are able to execute tasks in a reasonable amount of time.

Workflow Requirements:

- The tool is able to run Netlogo model simulations.
- The tool is able to visualise the output of NetLogo simulations.
- The tool is able to work with user specified input.
- The tool is generally applicable to any NetLogo model.
- The tool provides various parameter sampling options.

- The tool is be able to execute multiple simulations in parallel.
- The tool is able to run on every operating system (Windows/Linux/Mac).

User Interface Requirements

- The usage of the UI is similar to the NetLogo BehaviorSpace.
- The user navigates the UI with buttons.
- The UI provides help buttons that provide explanations.
- The UI shows a clear error message when something goes wrong.

These requirements are defined in such a way that the fulfillment of all of them results in an accessible DM tool for ABMS. The most important part of this accessibility is the GUI, because it has been shown that the accessibility of many applications increases drastically when the functionality is provided inside of a GUI [36]. The workflow requirements are largely similar to the existing DM tools such as EMA Workbench, because EMA provides extensive functionality and options to execute and analyse ABMS experiments. By emulating this functionality, but packaging it in a tool that is accessible and easy to use, we provide the users with exactly what is currently missing for ABMS.

3.3. Usability Study

At the end of the UCD process, the product is often evaluated by means of usability testing. Usability testing is defined as ”a systematic way of observing actual users trying out a product and collecting information about the specific ways in which the product is easy or difficult for them” [31]. Usability testing naturally fits well within the UCD process due to the large emphasize on communication with end-users. During the usability study, both quantitative data related to measurable usability criteria, as well as qualitative data related to the user satisfaction are collected [2]. The results from the usability study will give a concrete measurement of the usability of the tool, which is connected directly to the accessibility [32]. This means that we will be able to answer the research question with the results from the usability study and thus complete the UCD process. Furthermore, the recommendations from participants during and after the usability study can be used as input for future work towards the accessibility of tools for ABMS. Since these recommendations are direct feedback from end-users, they should be considered by anyone attempting to make a successful tool for this same target group.

4 Final Product

This chapter will discuss the implementation of the final product and the motivation for the design choices that were made. First, the technologies and libraries that were used to implement the tool will be discussed in Section 4.1. After this, Section 4.2 will give an in-depth overview of the features of the tool, along with the motivation for these features.

4.1. Technologies

This section will give an overview of the most important libraries that were used to implement the tool. At a very high level, the tool can be split up into three parts. These parts are the GUI, the NetLogo back-end and the visualisations. The following sections will describe the libraries used for all three of these parts in detail and explain why these libraries were chosen for this thesis. Other libraries were used alongside these such as pandas or numpy during data-processing for example. However, since these libraries are relatively standard, they will not be discussed here.

4.1.1. PySimpleGUI

PySimpleGUI is a Python library that allows the user to easily implement GUI's. It is mainly known for its simplicity and ease of use. Compared to other GUI frameworks, it requires half as much code to implement the same functionality [75]. Despite the ease of use, it still provides a good balance between usability and power [36]. Since the library launched in 2018, it has been actively supported and expanded by the developers, including extensive documentation with example programs and explanations. This contributes towards the maintainability of the tool if it becomes an open-source project, since developers will be able to quickly understand and improve the code of the GUI.

4.1.2. PyNetLogo

When the user has designed an experiment in the GUI, a connection between NetLogo and Python has to be made in order to execute the simulation. The most well-known Python library that implements this functionality is pyNetLogo [41]. When the user clicks the run button, the inputs are used to create a link to NetLogo with pyNetLogo. When this link is successfully created, the NetLogo simulation can easily be controlled by executing commands and reporting the values of specified reporters.

At the start of this thesis, it was decided that the implementation would be done in Python, because of the amount of useful libraries it gives access to and my personal experience with the language. As far as I know, pyNetLogo is the only library that provides a link to NetLogo from Python, so it was the obvious choice for this thesis.

4.1.3. Plotly

In order to extract useful information from the experiments, the data needs to be analysed by the user. This analysis is performed by using various visualisations that the user has access to in the GUI. These visualisations are implemented with Plotly[1], which is an open-source graphing library that

[1]https://plotly.com/python/

provides interactive, high-quality graphs. The usage of Plotly is similar to matplotlib, which means it is easy to use, style and customize.

The main advantages of Plotly are its ease of use and interactivity. The interactivity of the graphs allows users to dynamically change the graphs by zooming in, dragging axis, selecting which lines to plot and showing tooltips with extra information of certain data points. This interactivity is incredibly useful when exploring the parameter configurations, because it allows the user to focus on a specific subset of the data that they are interested in and thus makes the graph easier to interpret.

A potential issue with Plotly is the fact that the graphs open in the browser, which can be confusing if users are using the GUI and suddenly a new window opens. However, this is an advantage while implementing because it removes the need to embed figures into the GUI itself, which makes it much easier to generate and show the figures. However, the advantages of Plotly outweigh this potential negative in my opinion, which is why it was chosen for this thesis.

4.2. Features

This section will discuss the features of the final product in detail. As opposed to the previous section, this section will go more into what was done with the technologies instead of what the technologies are themselves. Every subsection will explain a different functionality of the tool that the users have access to, along with the motivation for the feature.

4.2.1. Experimental Design

For the implementation of the experimental design, inspiration was taken from the NetLogo BehaviorSpace. This was done because the interface is clear and easy to use, and people that are experienced with NetLogo will be familiar with it. For this reason, making the overall workflow of the experimental design similar to it is a good way to improve the ease of use of the tool. However, one major difference is that the tool provides two different types of experimental designs. The first, called parameter space search (PSS), allows for exploration of the parameter search space by performing runs with different parameter settings. The other, called reporter value analysis (RVA), allows the user to analyse the output of a single parameter configuration by running multiple repetitions of the simulation. An explanation of how the experimental design is used, along with a screenshot of the GUI is given in Appendix B.

Parameter space search The PSS run type allows the user to enter a custom search space for the model parameters by giving a lower and upper bound for the values of each parameter. The user also inputs the number of scenarios (N) they would like to run, where a scenario here is defined as a single parameter configuration. By using the parameter bounds, N scenarios are generated by using one of five sampling methods that the user has access to. These sampling options were chosen by looking at which ones were the easiest to understand while still giving a good mixture of methods that the user can choose from. Full factorial was included specifically to give the user something they are familiar with, since it is the only sampling option available in the NetLogo BehaviorSpace. The five sampling options are:

- Monte Carlo: The parameter values are sampled completely random and independently within the parameter bounds [87].
- Latin Hypercube: The parameter values are sampled near-random, but depended on each other to obtain a good spread [57].
- Full factorial: Cartesian product of the integers between all parameter bounds will be used as parameter configurations [5]. This method ignores the input for N because the number of scenarios is defined by the parameter bounds input itself.
- Saltelli: The parameter values are sampled using Saltelli’s sampling scheme [25].
- Sobol: The parameter values are sampled using Sobol’s sampling scheme [88].

Other than parameter bounds, the user can also input a parameter name with a single value to have that value set the same for every scenario. If neither bounds or a specific value are provided, the simulation uses the default value for that parameter as specified in the model.

The motivation for this run type was to provide the user with a way to analyse the effect of the parameters on the model simulation. Tuning the model is an important step in the design process of an agent-based model, also known as parameter optimisation. Agent-based models are often characterized by many parameters that determine the dynamics of the simulation [21]. The large number of parameters, along with possible dependencies between parameters and high sensitivity of parameters can make this tuning process challenging. By allowing the user to systematically explore the search space and visualise the results of these simulations, the optimal parameters of the model parameters can be estimated.

Another application of this run type is sensitivity analysis, which is the process of determining how the uncertainty in the output of a model can be contributed to different sources of uncertainty in the model parameters [83]. Sensitivity analysis is a commonly used method to explore the behaviour of models [19, 56, 93]. By providing the user with a method to explore the search space, sensitivity analysis can be performed by analysing the output of the simulations with respect to the various parameter configurations.

Reporter value analysis The RVA runs type allows the user to run a single parameter configuration for a certain number of repetitions and plot its results. The main usage for this run type is to analyse the output of a parameter configuration, specifically the variance and distribution of this output. Since model runs are usually stochastic, it is important to run simulations more than once in order to ensure that the conclusions are not drawn from results that happened to be an outlier [62]. By running multiple repetitions and averaging the results, it gives a better view of what can be expected from the output.

4.2.2. Import/Export

The tool allows for the import and export of experimental designs and results from and to text files (.txt) that the user can save on their computer. It takes time and effort to fill in all fields for an experimental design and a user may want to reuse (parts of) this design in the future. For this reason, being able to save a configuration and import it later on is a useful feature to make the design process less tedious. When the user imports a valid configuration file, all the fields in the GUI are automatically filled out with its values, but the configuration is not run yet. This means that the user can import a configuration and still change the values before running it, thus increasing the usability of a single configuration since it can be reused partly.

Besides run configurations, results of simulations can also be saved and imported. Depending on the model and its configuration, a single experimental design may take a long time to run and the user is not always ready to do a full analysis right away. When the user imports a valid results file, the data is read into the GUI and the user is taken to the screen where they would normally end up after running a configuration. This screen allows access to all the visualisations that belong to the run type of the result file. Being able to save the results and instantly importing them again later on without running the model itself improves the usability of the tool by allowing the users more freedom.

4.2.3. Visualisations

The most important feature of the tool is the built-in visualisations that it provides. Being able to run models is important, but if the data from these runs can not be analysed, it is still not useful for the model designers [14]. The tool offers different visualisations for the two different run types and allows the user to customize them. These visualisations provide the users with a clear overview of the data, but the interpretation of this data still has to be done by the users themselves and is out of the scope of this thesis. In total, there are eight different visualisation types, three being for PSS and five for RVA. These visualisations were selected based on the responses from the target group survey, the literature study and looking at the available graphs in Plotly documentation.

Parameter space search The goal of the PSS run type is to gain a good understanding of the effect of certain parameters on the output of a simulation. Every visualisation available for this run type provides a different type of insight between these correlations. There are three visualisations available for PSS runs: parallel coordinates, scatter plots and heatmaps. All three of these will be described in detail below, including what they are and what the best use case for them is.

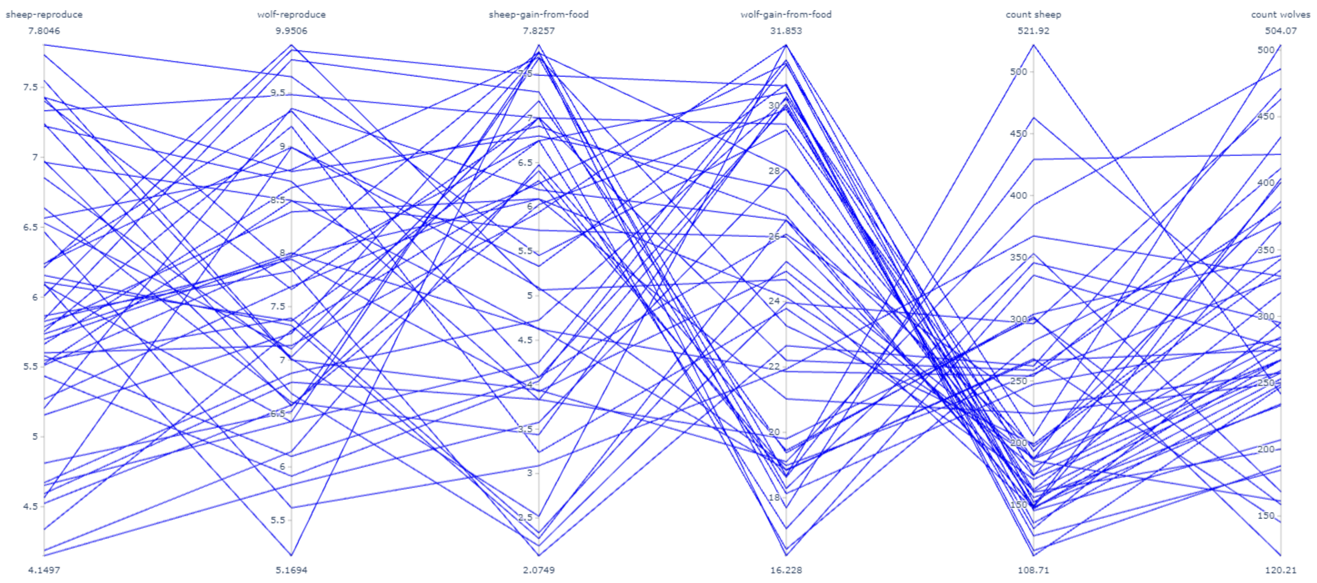


Figure 4.1: Parallel coordinates graph showing 50 scenarios with 4 parameters and 2 output reporters.

Parallel coordinates Parallel coordinates is a graph type that is commonly used to visualise and analyse high-dimensional datasets. In order to visualise an N-dimensional dataset, N vertical axes are placed in parallel [37]. Then, for every row in the dataset, a line is drawn through the axes corresponding to the values of the N dimensions of that row. An example of a parallel coordinates plot is given in Figure 4.1. Parallel coordinates naturally pair very well with ABMS experiments, because each scenario can be expressed in the same N-dimensional space. The N dimensions here are made up of both the parameter configuration and its corresponding output values, because both of them are needed to extract useful information from the graph.

Parallel coordinates are useful because it shows the actual parameter values for all scenarios instead of compressing them and simply shows the correlation that was found, like a heatmap does. This allows the user to get a deeper insight into what the result of a certain configuration is. This is further helped by the fact that the parallel coordinates are made with Plotly, which is fully interactive. The most useful part of the interactivity is that the user can select a range on one or more axes where the values have to be between. Any configurations that do not fall within these specified ranges are greyed out, which makes the figure much clearer and allows the user to only focus on the areas they are interested in.

The main disadvantage of parallel coordinates is the fact that it does not show a correlation between input and output directly [38]. The user has to use the interactivity to obtain a clear insight, because it is too chaotic to draw any conclusions otherwise. Even when a possible correlation is found, there is no numerical value that defines how big this correlation is exactly. For this reason, the best use case of the parallel coordinates is to run large ranges for the parameter values to explore a large part of the search space. When interesting areas of the search space are then identified, these can be investigated in more detail with the other visualisations.

Scatter plot A scatter plot is a simple graph type that allows one to determine the correlation between two variables [26]. It is best suited for 2-dimensional data, which can be seen in this context as one input variable and one output variable. Figure 4.2 shows an example of scatter plots that can be generated with the tool.

The main advantage of scatter plots is that they are very easy to interpret, especially when a trend-line for the data is shown [52]. It allows the user to focus on the effect of a single parameter, rather than the parameter configuration as a whole. It also shows the actual data points in the graph which give a more in-depth overview on how a certain correlation is obtained. Both parallel coordinates and heatmaps do not show the data points themselves, so this is an aspect where the scatter plots are useful.

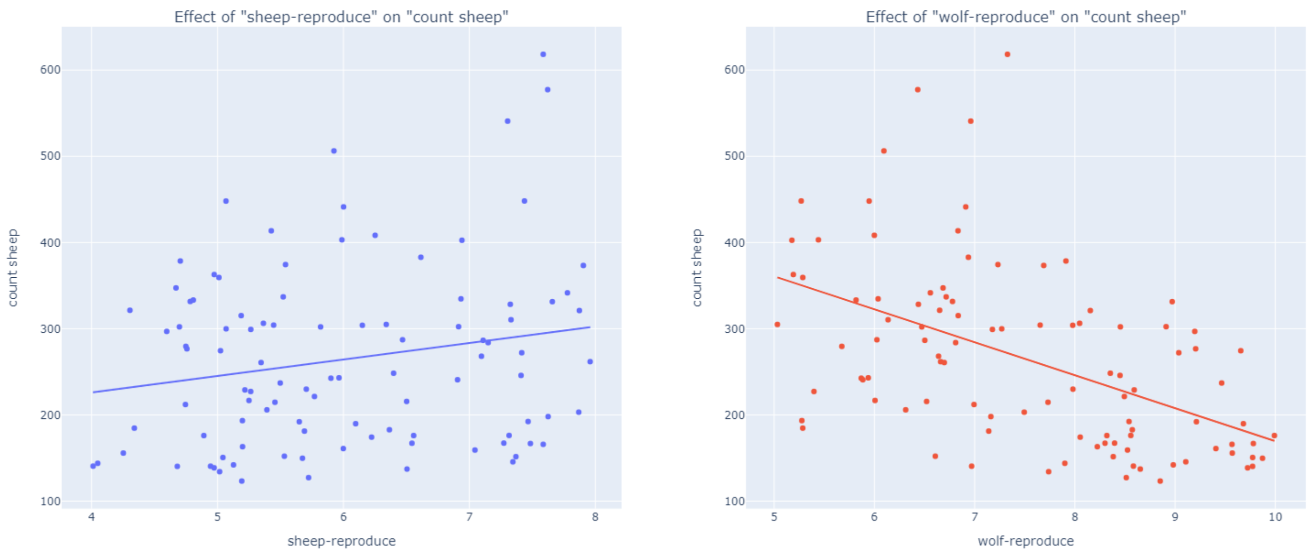


Figure 4.2: Scatter plots showing the correlation of the ”sheep-reproduce” and ”wolf-reproduce” parameters with the output of ”count sheep”.

Heatmap A heatmap is a visualisation type where the correlation between two variables is shown with numerical values represented as colours [76]. The heatmap consists of a symmetrical grid where the correlation is shown for each pair of variables. By looking at the squares that indicate the correlation between input and output variables, a meaningful understanding can be obtained about which parameters have the largest effect on the output. A heatmap usually uses a colour scale where the darker the colour, the higher the correlation. An example of a heatmap is shown in Figure 4.3.

Similarly to the scatter plot, the heatmap is used to determine the correlation between input and output. However, unlike the scatter plot, the entire dataset is represented in a single figure. This condensed form of visualisation contains much more information than the other types, but can also be harder to interpret sometimes [49]. Heatmaps can be quite confusing at first, certainly with a large number of dimensions. However, they are actually relatively simple and extremely useful when it is clear how to interpret them. Another advantage of the heatmap is the fact that it shows a numerical value that indicates the correlation. This is different from the parallel coordinates and scatter plots where a correlation can be seen, but how large it is exactly can be hard to determine.

Reporter value analysis The goal of the RVA run type is to analyse the output of a certain parameter configuration, specifically the variance and distribution of this output. Some of the visualisations available for this run type provide a way to gain insight into the variance, but some are also available to plot basic run information, such as timeseries and histograms of reporter values. There are five visualisations available for PSS runs: timeseries plot, histograms, distribution plots, box plots and violin plots. All five of these will be described in detail below, including what they are and what the best use case for them is. However, since histograms and distribution plots, and box plots and violin plots are similar to each other, they will be discussed together.

Timeseries plot A timeseries plot is a graph type where the value of a variable is plotted over some unit of time, in this case ticks [28]. Because all NetLogo simulations progress by increasing the tick value, timeseries plots naturally pair very well with these simulations. NetLogo already has some built-in functionality for timeseries plots of model values, but these are quite limited. These are shown in the interface of a simulation and thus will only show the data from a single run. With our tool, it is possible to plot the data from multiple repetitions into a single graph, including an indication for the variance between the repetitions. An example of such a graph is shown in Figure 4.4. The shaded areas of the lines indicates the standard deviation of the results.

The main use of the timeseries plot is to provide extra functionality on top of the graph already available in NetLogo itself. This can be used to analyse the progression of a model value throughout the simulation, and thus to gain a deeper understanding of the behaviour of the model for a certain

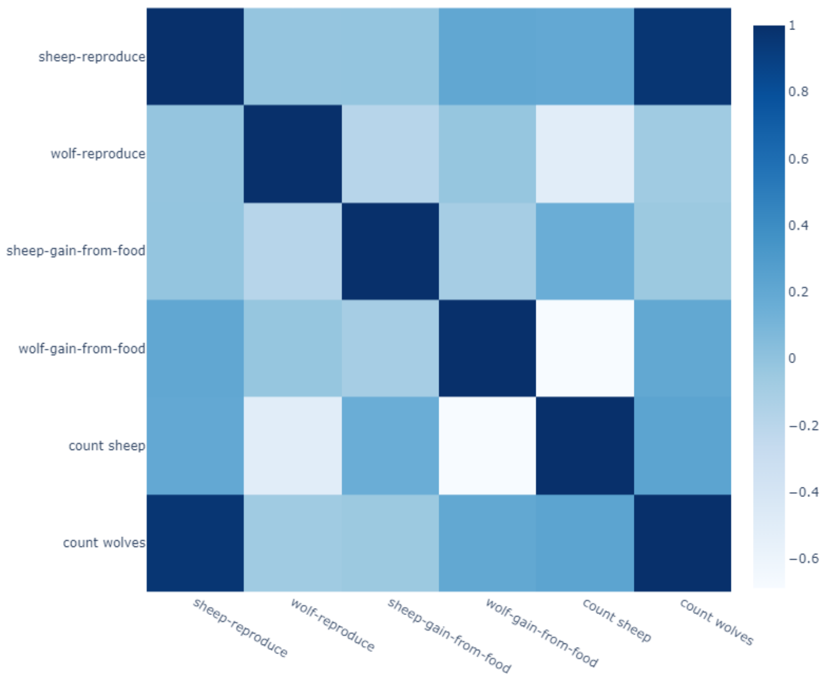


Figure 4.3: Heatmap showing the correlation between the 5 input parameters and two output parameters of the model.

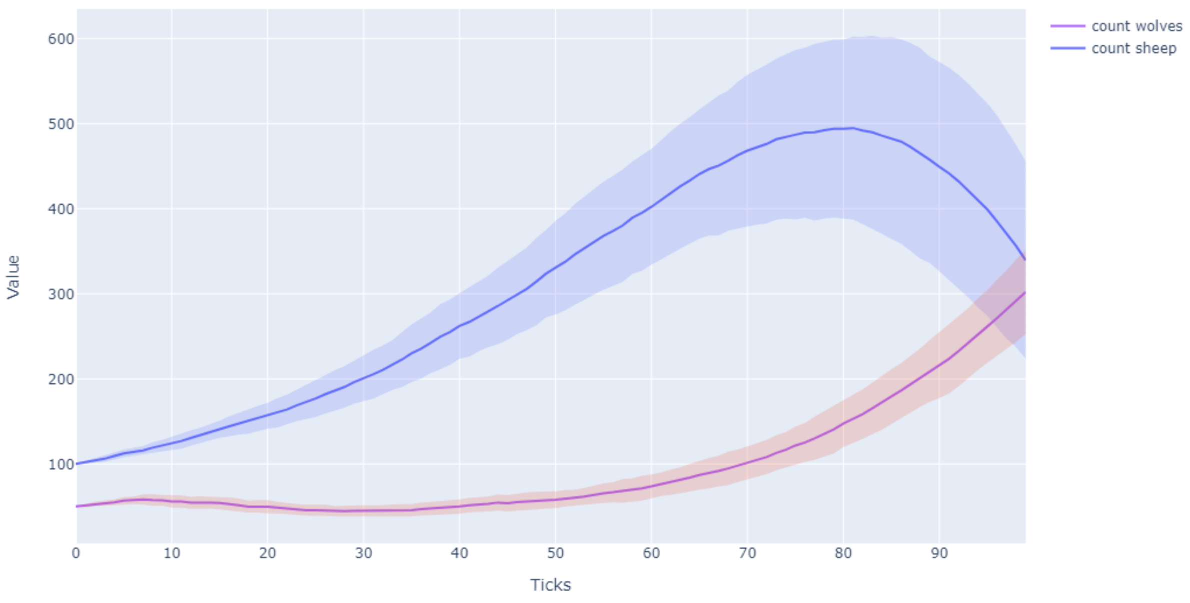


Figure 4.4: Timeseries plot showing the progression of the ”count sheep” and ”count wolves” reporters throughout the simulation.

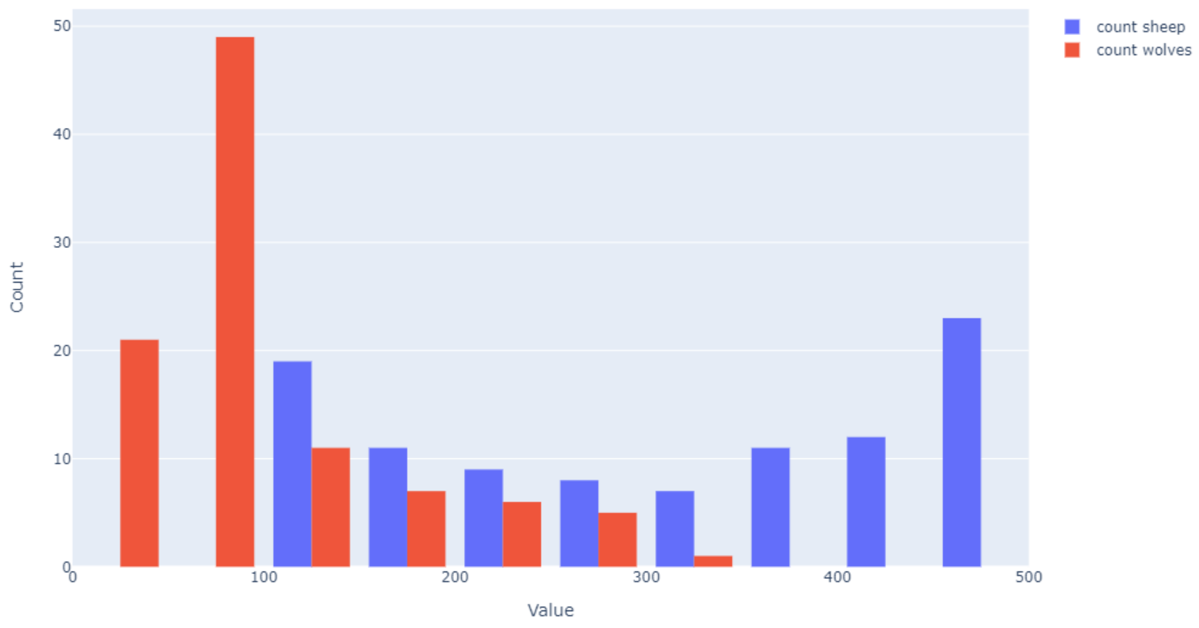


Figure 4.5: Histogram plot showing the distribution of the ”count sheep” and ”count wolves” reporters throughout the simulation.

parameter configuration. the interactivity of Plotly can also be used here again to zoom in, drag axes, toggle the display of lines on and off and show tooltips with the values of the graph. This interactivity, along with the shaded areas and multiple repetitions in a single graph provides much more functionality than the standard NetLogo plots, and thus also makes the analysis of models easier.

Histogram & Distribution plot A histogram can plot the distribution of the values of a variable by allocating them to buckets [29]. Every bucket has a lower and upper bound and the value of each bucket is the number of values that fall in that range. The data plotted is the same as with the timeseries, so the total number of data points is equal to the number of ticks in the simulation. Similar to this is the distribution plot, which also gives an overview of the distribution of values, but also includes a line curve and rug plot next to the histogram. All three of these can be turned on and off when generating the graph, so the user also has the ability to plot the distribution with one or two of the three options. Figure 4.5 and Figure 4.6 show the same data as presented in Figure 4.4, but then represented as a histogram and complete distribution plot respectively.

The goal of these two visualisations are to get an easier overview of the distribution of the values of a reporter during a simulation. Even though the same data is used for the timeseries, that plot type is more useful for the progression of the value, rather than the distribution of it. With these two plots, it is very easy to see how often a reporter has a certain value during a simulation.

Box plot & Violin plot Box plots and violin plots are two graph types that are used to give a more statistical overview of a distribution of values. A box plot shows the distribution by displaying the 25th, 50th, 75th percentile in a box and the 0th and 100th percentile as whiskers [34]. A violin plot takes a more direct approach to the visualisation by displaying the distribution as a line curve, similar to the line curve from the distribution plot. Examples of the box plot and violin plot are shown in Figures 4.7 and 4.8 respectively.

These two visualisations can be used to determine the variance of a certain reporter throughout a simulation. The data from all repetitions is averaged at each tick and then plotted. With these graphs, a good indication of which values can be expected from a reporter is given. In the plots, the outliers are also shown if there are any, which may provide the user with some additional interesting information.

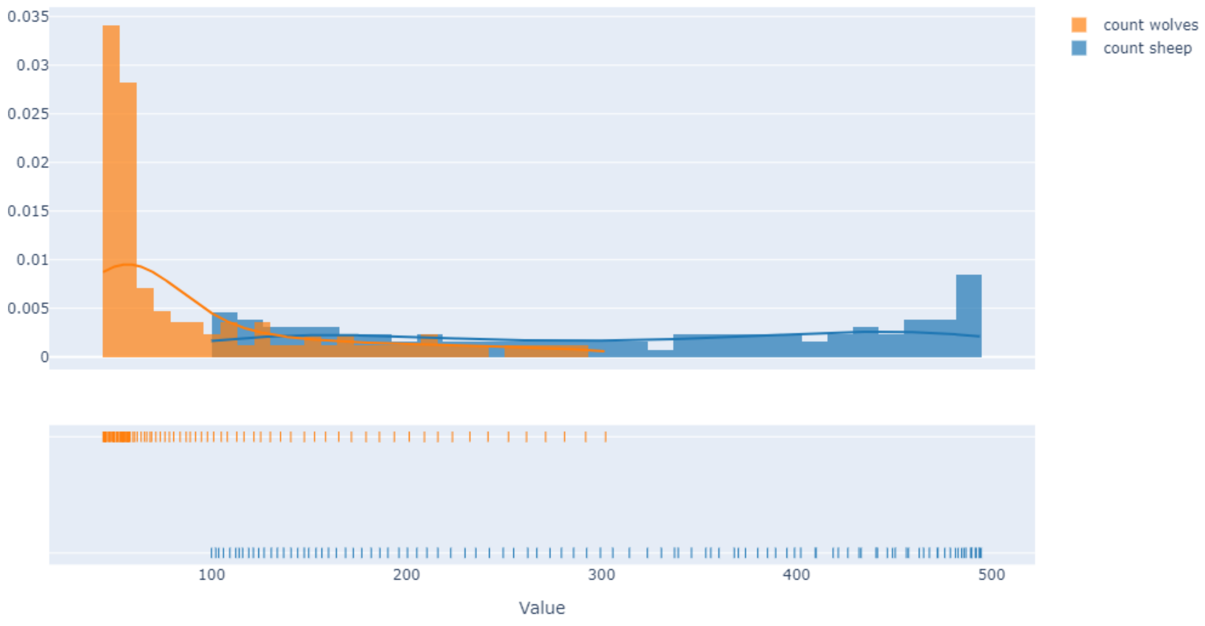


Figure 4.6: Distribution plot showing the distribution of the ”count sheep” and ”count wolves” reporters throughout the simulation with a histogram, line curve and rug plot.

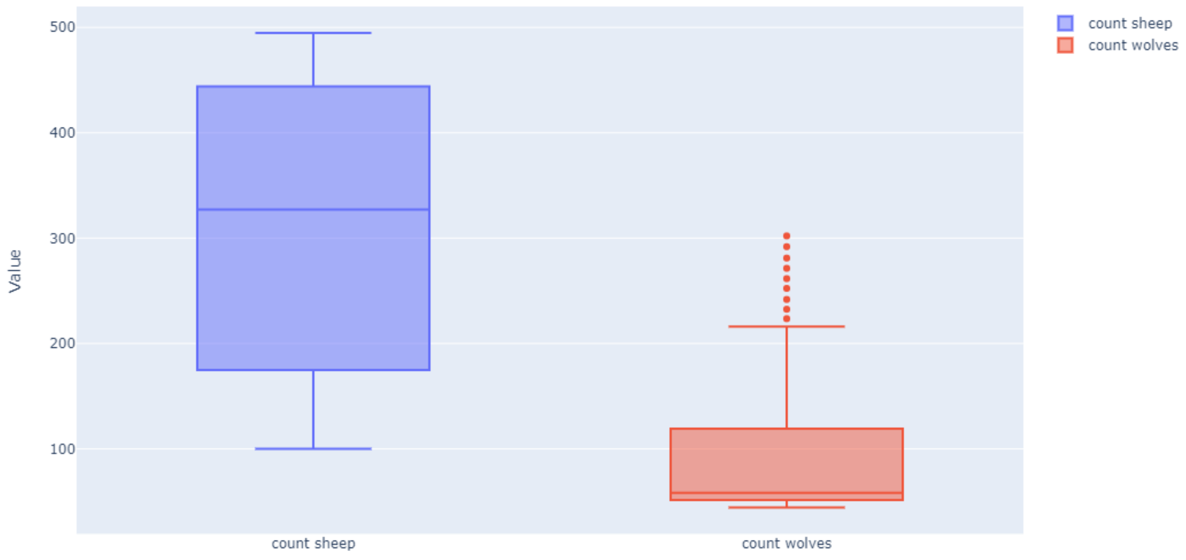


Figure 4.7: Box plots showing the variance of the ”count sheep” and ”count wolves” reporters throughout the simulation.

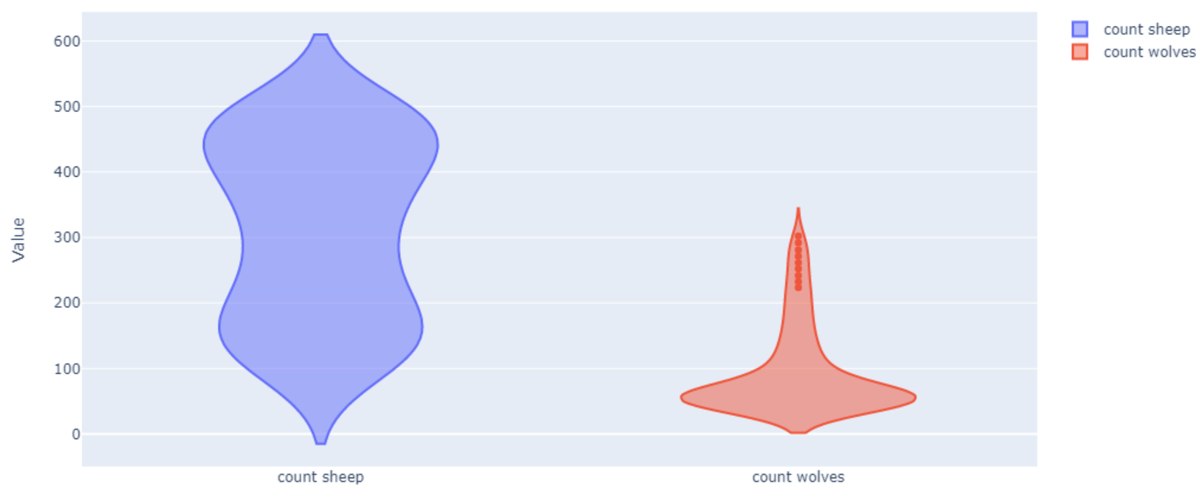


Figure 4.8: Violin plots showing the distribution of the ”count sheep” and ”count wolves” reporters throughout the simulation.

5 Usability Study

The goal of this usability study is to perform a formal evaluation of the DM tool that was developed during this master thesis. There are two primary purposes of usability studies, namely to identify usability problems or to gather usability measurements [48]. With this usability study, the primary goal is to gather usability measurements with metrics and questionnaires. With these usability measurements, a definitive statement can be made about the usefulness and usability of the tool. The results of this study can be used to determine the success of the thesis and to be able to place the tool in the context of relevant literature and alternatives.

This chapter is structured as follows. Section 5.1 will describe the setup of the usability study, including the recruitment of participants. The procedure is discussed in Section 5.2. In Section 5.3, the measures that were recorded during the study will be given, along with the motivation for these measures. Finally, Section 5.4 will describe the measure goals, which will determine whether the usability study is successful.

5.1. Setup

The participants for the study were found through a combination of the SIMSOC mailing list, NetLogo forums and indirectly through friends/acquaintances of my supervisor and people that already agreed to participate. Using these methods, a total of 14 people were found that were willing to help with the usability study. Beforehand, the decision was made to have between 10-15 people for the study. This range was chosen to have a good balance between the reliability of the results and the time it would take to do all the usability studies. There was only a single inclusion criteria for the study, namely that the participant had to have moderate to high experience with NetLogo. This was done because beginners are usually not experienced with experimental design for model analysis and thus would not be able to properly assess the usefulness of the tool.

5.2. Procedure

When the participants are found, it is important to define a procedure that is able to answer the main question of the usability study [98]. In our case, the main question is the usability of the tool, so the procedure must result in some measures of the usability that allow the question to be answered. In short, the procedure consists of the participant executing three specified tasks during and after which the measures are recorded. These tasks are chosen in such a way that they access nearly all components of the tool in order to obtain the most reliable data of the usability. The procedure is discussed in detail in Appendix C, along with the instructions that were given to the participant.

5.3. Measures

The measures are the values that are recorded during the usability study, which make up the main results of the study. The measures need to be chosen in a way where they provide useful information towards answering the main research question of the usability study. In this case, the main question is the usability of the tool, so the measures need to be able to provide some indication of the usability of the system. Usability is defined as ”the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use” [43]. There are two types of measures that will be recorded during the study, namely metrics and questionnaires. The metrics will give an indication of the effectiveness and efficiency, while the questionnaires will be used to judge the user satisfaction. Both of these measures will be explained in the following sections.

5.3.1. Metrics

The metrics are usability measures that are recorded while the participant is executing the tasks [47]. For this usability study, two metrics have been defined:

1. Success rate of a scenario X (number of times it was successfully completed).
2. Amount of time it takes to accomplish scenario X.

The motivation of these two metrics is to provide a measure of the effectiveness and efficiency [48]. The success rate and time taken are the two most commonly used metrics usability studies [39]. Due to the simplicity and linear workflow of our tool, other metrics such as the number of wrong menu choices and usage patterns would not provide relevant information about the usability. Furthermore, the most important part of this usability study was the user satisfaction, so more emphasis was put on the questionnaires than the metrics. For these reasons, using only these two metrics was sufficient within the context of this study.

5.3.2. Questionnaires

Not all aspects of the tool can be put into concrete numbers by defining a metric for it. For this reason, the test subjects will also be asked to fill in a questionnaire to give a more opinion based evaluation of the tool. The questionnaires that will be used are standardized usability questionnaires that are widely used across a variety of fields:

1. After-Scenario Questionnaire (ASQ): A short 3 item questionnaire that the participants fills in after each scenario. All 3 questions are answered using a 5 point Likert scale [44]. This questionnaire is used to get a good understanding of the individual components of the tool that have to be used during a scenario, rather than the tool as a whole. This can expose issues that would otherwise potentially get lost in obscurity of a full questionnaire. The ASQ has been shown to provide a useful measure for usability studies [53].
2. Software Usability Scale (SUS): A 10 item questionnaire that the participant fills in after all scenarios have been completed. All 10 questions are answered using a 5 point Likert scale [44]. SUS is widely used and focuses mainly on the ease of use of the system, which is exactly what we are interested in. It is easy to understand for both the participant and the evaluator and has been shown to be reliable across a wide range of domains [74].

These two questionnaires were selected because they are widely used and thus provide a good basis to compare the results with. The combination of these two questionnaires within the same usability study has been shown to give reliable results [51], because it provides an indication of the usability for each scenario as well as the system as a whole. The results from these two questionnaires will make up the most important part of the results.

5.3.3. Open Questions

Along with the standardized questionnaires, some open questions will also be answered by the participant. These questions are about system specific things that would not necessarily be answered in a standardized questionnaire. This includes things such as the installation of the tool and the comparison to the NetLogo BehaviorSpace. The three open questions that will be asked are:

1. How do you feel this tool compares to the NetLogo BehaviorSpace in terms of functionality and usability?
2. Would you say the installation of the tool was straight-forward or difficult?
3. Do you have any final comments or advice for improvements?

The answers to these open questions will not be formally analysed, but the feedback will be taken into account for future improvements of the tool. Examples of such improvements could be easier installation, more visualisations or more customisation. However, since this feedback is obtained relatively late in the project, there is no guarantee that all of it can be implemented. For this reason, the answers the the open questions will be taken more as inspiration rather than hard requirements.

5.4. Measure Goals

In order to determine whether the system performs on a satisfactory level, measure goals need to be set before the usability study starts [90]. With these goals, the results can be used to determine whether the system meets the standard for usability. For this study, measure goals need to be set for both of the two numerical metrics, as well as the questionnaires.

5.4.1. Metrics

The metric goals are the goals that are set for the metrics described in the previous section. For this study, one goal is defined for the success rate along with a time goal for each scenario, which results in a total of four metric goals:

1. The success rate of all scenarios is 90% or more.
2. The average completion time of scenario 1 is less than 600 seconds (10 minutes).
3. The average completion time of scenario 2 is less than 480 seconds (8 minutes).
4. The average completion time of scenario 3 is less than 240 seconds (4 minutes).

The goals for the times were set at these values by looking at how long it took myself to execute these tasks, along with some estimation and comparison to the current alternatives such as BehaviorSpace. Obviously I know exactly how the tool works, so those times are much lower than what is expected from the participants. The first time the participants are using the tool will take much longer than when they are familiar with it, so the goals were set relatively high. However, 10, 8 and 4 minutes respectively are still respectable scores for a first time user, since it is still much faster than manually executing and visualising experiments.

Because the number of participants is relatively low (<15), taking the mean value of the results is better than a percentile because they are less variable [27]. So instead of ”95% of the participants completes the task in 10 minutes”, the goal is set as ”the average completion time is less than 10 minutes”. However, since the success rate produces a binary value, using the average value is not possible, so for this metric a percentile is set as the goal. Because the number of participants is less than 15, a success rate of 90% means that at most 1 person can fail each scenario, which is a reasonable goal to set.

5.4.2. Questionnaires

Since both of the questionnaires that are used are standardized, there is a lot of information readily available about interpreting the results of them. Since there is a very limited time to get familiar with the system during the tasks, any usability rated above neutral is good because the tool will become easier to use with experience. The expectation is that people will rate lower during a usability study due to them being new to the tool, so the measure goals for the questionnaires are set at just above average. As people get more familiar with it, it will become easier to use as well so the real usability score should be even higher than the results from the usability study [15].

For ASQ, we are using a 5 point Likert scale as opposed to the standard 7 point, because the tasks are relatively fast and easy and thus should also be easier to grade. However, the interpretation of the results can still be done the same way by averaging the scores and checking whether they are above neutral. Since a neutral score would be 3, the goal for this study will be to be at 4 or above.

For SUS, the industry standard for being above average in usability is defined by a score of 68 [54]. We want to be above this score at least, but because the tool is heavily focused on ease of use, the goal will be set to be at an average of 75 or above.

6 Results

In this chapter, the results of the usability study described in the previous chapter will be presented. The results consist of three main parts, namely the recorded metrics, the ASQ questionnaire and the SUS questionnaire. The results from each of these parts will be given in Sections 6.1, 6.2 and 6.3 respectively. Finally, Section 6.4 will discuss the most common answers to the open questions and general feedback that was given by the participants.

For clarity, the results from all three parts of the usability study are summarized in Table 6.2. In this table, the results of all metrics and questionnaires are shown for each participant, along with the mean and standard deviation of each column.

6.1. Metrics

Only two metrics were defined during this usability study, namely the success rate of a scenario and the amount of time taken to execute a scenario. The success rate for all scenarios was 100%, which means that every participants managed to successfully finish each of the three tasks. However, since the usability studies were conducted in an online environment with screen sharing, I did assist the participant sometimes if they were confused, stuck or explicitly asked a question. By doing this, the success rate is almost guaranteed to be perfect for each scenario and thus does may not provide a meaningful insight into the usability. For this reason, the success rate will not be weighted heavily when analysing the results and has also been excluded from Table 6.2.

The amount of time it takes for a user to execute a scenario is more insightful than the success rate, although it is also affected by interference similar to the success rate. However, the time taken actually gives an indication of the ease with which a scenario was completed, as opposed to being a binary value like the success rate. This means that the effect of the interference is not nearly as large, because a difference between 100 and 120 seconds has a much smaller impact on the results than the difference between success and failure.

The time taken for the three scenarios for all participants are shown in Figure 6.1. From these graphs it can be seen that the times were quite similar for all participants, with very few large outliers. In Figure 6.2, a boxplot is shown of this same data grouped by scenario. In this graph it can be seen that only two times a point is classified as outlier, both in scenario 1. Besides these points however, the variance is relatively low with the difference between the 25th and 75th percentile being less than 200 seconds for all three scenarios.

The mean times for scenario 1, 2 and 3 are 520, 370 and 175 seconds respectively (rounded to the nearest multiple of 5). In Section 5.4, the goals for the mean execution times per scenario were set at 600, 480 and 240 respectively. This means that the goals for the execution times were achieved with a wide margin of at least a minute under the target for all three scenarios. This is a very good result because it proves two main points. The first one is that the participants were able to learn and use the system quickly, which is a good indication of the ease of use. The second point is that the functionality of the tool provides a good alternative to other options such as the BehaviorSpace, because people are able to design, run and visualise NetLogo experiments within an average of 9 minutes. Doing all of these actions manually would take a considerably longer amount of time, certainly if the user has to

	Scenario 1		Scenario 2		Scenario 3		SUS
	Time (s)	ASQ 1	Time (s)	ASQ 2	Time (s)	ASQ 3	
Participant 1	454	4.33	242	5.00	120	5.00	92.5
Participant 2	582	4.67	361	4.67	129	5.00	77.5
Participant 3	972	5.00	465	5.00	239	5.00	82.5
Participant 4	568	5.00	248	5.00	259	5.00	100.0
Participant 5	543	4.67	271	5.00	220	5.00	92.5
Participant 6	403	5.00	378	5.00	156	5.00	92.5
Participant 7	545	4.67	573	4.33	209	5.00	85.0
Participant 8	286	4.67	286	4.00	125	4.67	75.0
Participant 9	371	3.67	425	4.33	280	4.67	92.5
Participant 10	377	5.00	387	3.33	154	5.00	60.0
Participant 11	414	4.00	258	4.00	99	4.33	72.5
Participant 12	569	5.00	410	5.00	156	5.00	92.5
Participant 13	823	4.33	429	4.67	204	4.67	82.5
Participant 14	400	4.67	454	4.67	119	5.00	95.0
Mean	521.92	4.62	370.5	4.57	176.35	4.88	85.17
Standard Deviation	185.39	0.41	98.64	0.51	58.10	0.21	10.93

Table 6.2: Table showing all the results from the metrics and questionnaires gathered during the usability study.

do the data processing and visualisation themselves.

6.2. After Scenario Questionnaire

The ASQ questionnaires are taken directly after the participant completes a scenario and consists of three questions about the ease, time and support information of the scenario. The answers are given with a 5-point Likert scale ranging from strongly agree to strongly disagree. The score of a single ASQ is obtained by averaging the scores of the three questions, which results in a final score between 0 and 5, 5 being the best. In Figure 6.3, the average ASQ scores of all three scenarios for all participants are shown, similar to Figure 6.1. From these graphs it can be seen that the average scores are very good, rarely going below 4 for most participants. Out of the 42 averages in total, only two were lower than 4, being 3.33 and 3.67.

In the left graph of Figure 6.4, a boxplot is shown for the average ASQ scores per scenario. From this graph it is visible that there is a relatively large variance at the bottom for the first two scenarios. This is largely caused by the scores of 3.67 and 3.33 for scenario 1 and 2 respectively. However, since there are also some scores of 4 and 4.33, these lower values are not seen as outliers and thus cause a large whisker in the boxplot. The variance at the top of the box is 0 for all scenarios because the 75th percentile is already 5.0, which is the maximum score.

The average ASQ scores for scenario 1, 2 and 3 are 4.62, 4.57 and 4.88 respectively, as shown in Table 6.2. Sadly, there is no specific industry standard to which the results can be compared. However, the goal of this study as specified in Section 5.4 was to be above an average of 4 for all scenarios. Since a value of 3 would be neutral, a score of 4 means that the participants were satisfied with the ease of the task, which gives an indication of the ease of use of the system itself as well. For all three scenarios the average ASQ scores were well above 4, which shows that the participants generally thought the tasks were easy to execute with our tool.

6.3. System Usability Scale

When the participant has completed all three scenarios, they fill in the SUS questionnaire, which consists of 10 questions about the usability of the system. Similar to ASQ, the answers are given on a 5-point Likert scale ranging from strongly agree to strongly disagree. The final score of the SUS questionnaire is calculated in a different way however, which results in a score between 0-100 with 100 being the best. The explanation of the calculation of this score, along with an example, is given in Appendix D.

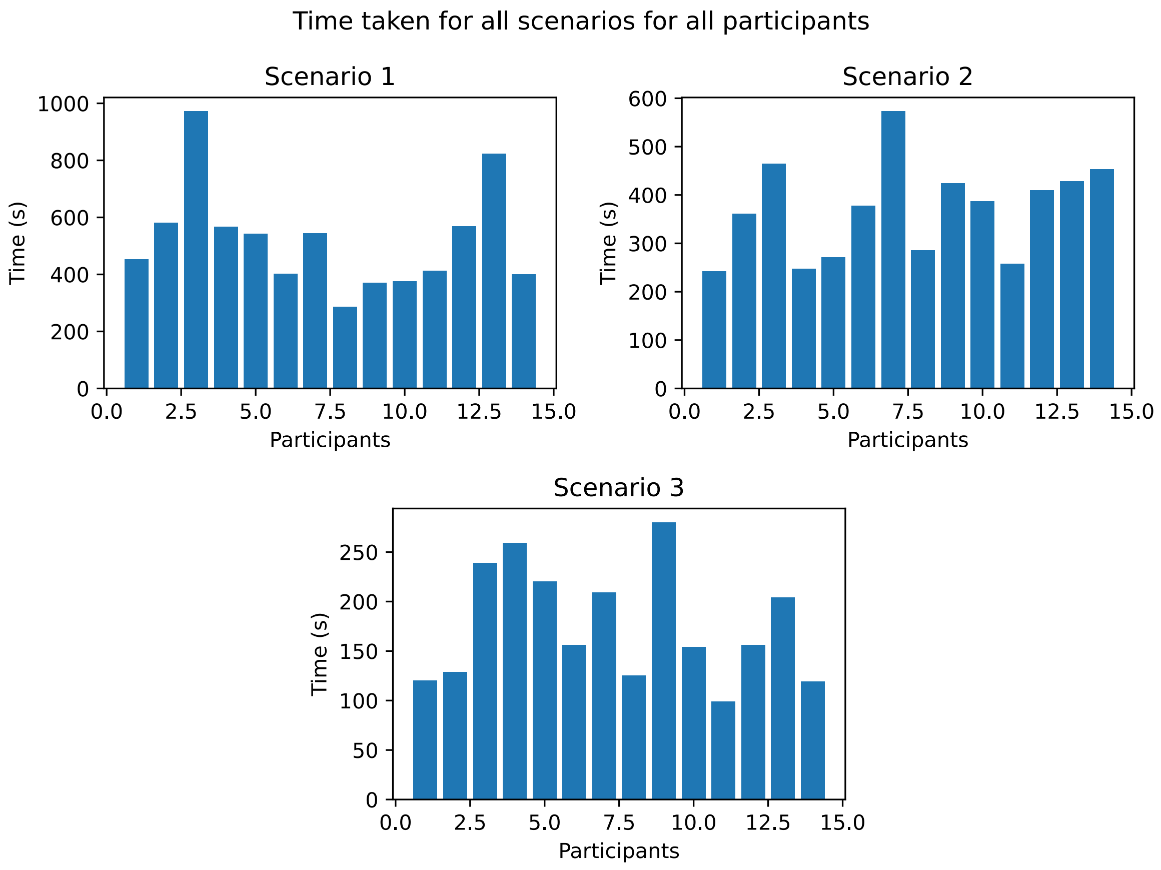


Figure 6.1: Execution times for the three scenarios for all participants.

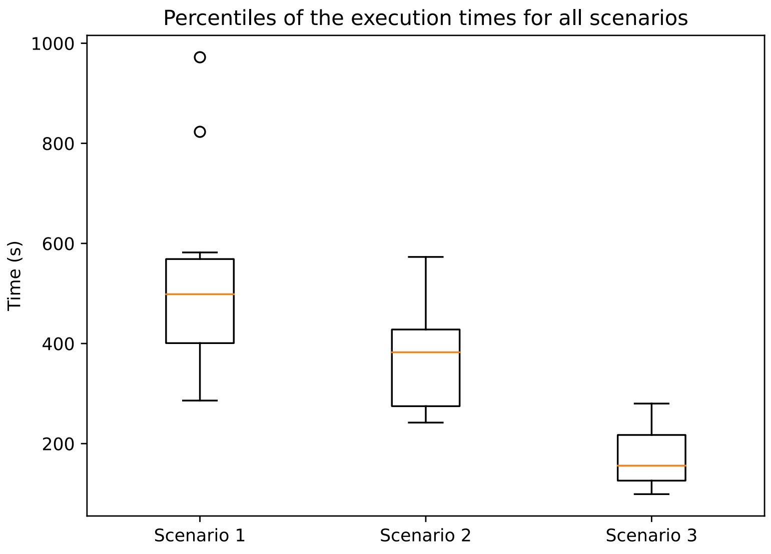


Figure 6.2: Boxplot showing the execution times for all three scenarios.

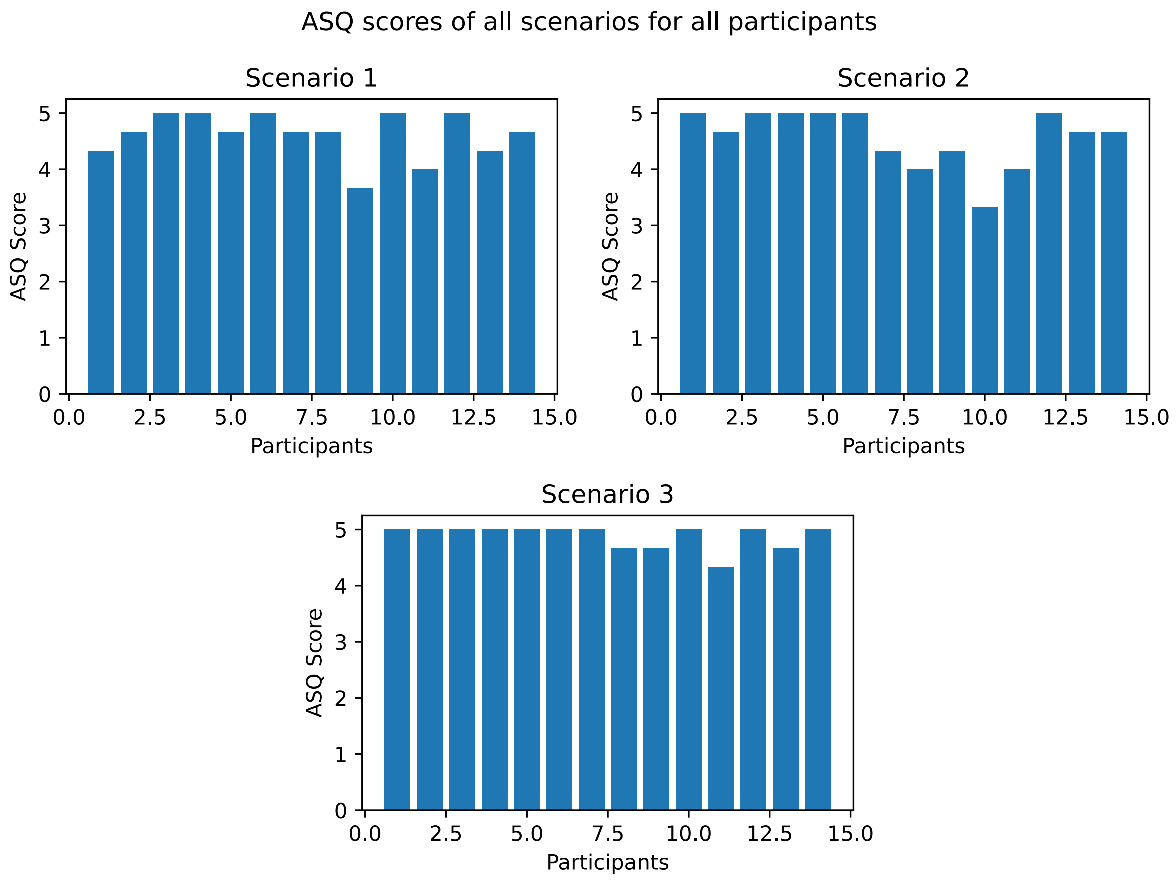


Figure 6.3: Average ASQ scores for the three scenarios for all participants.

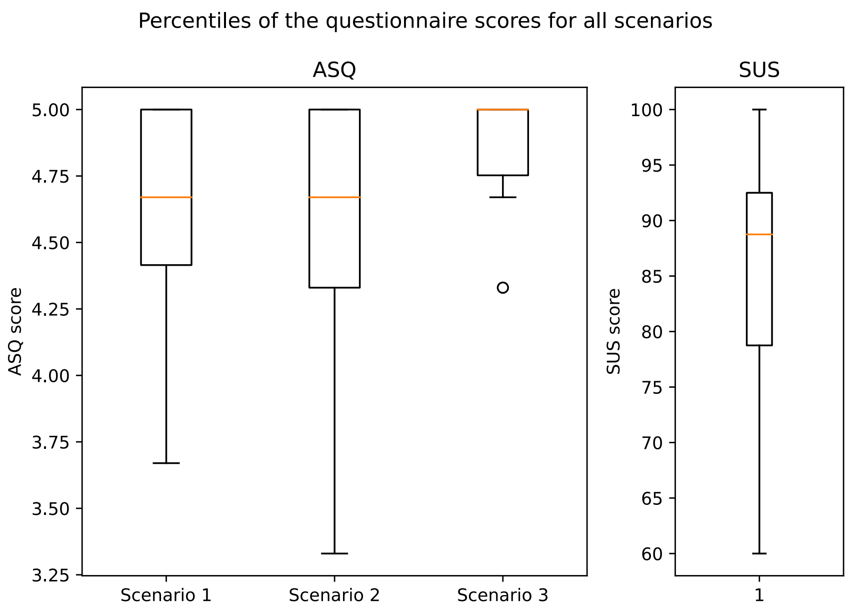


Figure 6.4: Boxplot showing the average ASQ scores for each scenario (left) and the SUS scores (right)

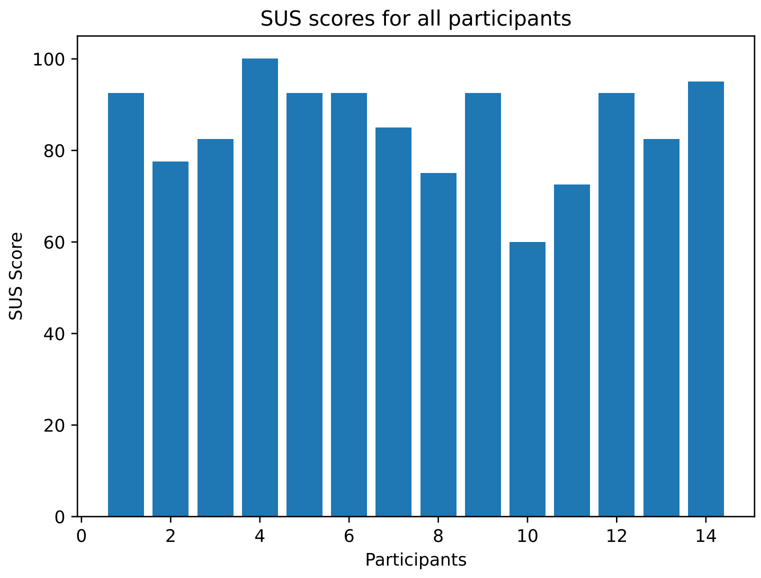


Figure 6.5: SUS scores for all participants.

The SUS scores that were obtained from all participants are shown in Figure 6.5. In general, the scores were quite good, with all of them except one being above the industry standard of 68. On the right side in Figure 6.4, a boxplot of all the SUS scores is given. From this graph it can be seen that there are no large outliers, but there is a large whisker going down to 60. The reason that this score is not an outlier is because there are a few other scores in the 70's that are close enough to it that it can not be seen as an outlier.

In Table 6.2, the mean SUS score is given as 85. It is important to note that these scores should not be interpreted as percentages, despite them being scores from 0-100. The 50th percentile (median) of SUS scores is defined as 68, which can be considered as average usability. Our mean score of 85 is 17 points above this benchmark, which shows that participants considered the tool to be very easy to use while executing the tasks.

6.4. Open Questions & Feedback

Besides the questionnaires, the participant was also asked three open questions, which are discussed in Section 5.3.3. These questions were meant to gain more information about the opinion of the participant on the tool on aspects that were not covered by the standardized questionnaires. This included questions about the comparison to the BehaviorSpace, installation of the tool and general feedback.

The general consensus among the participants was that the tool was similar to the BehaviorSpace in terms of usability, but much better in terms of functionality. The usability being similar is logical, since the interface of the GUI was heavily inspired by the BehaviorSpace. The biggest upside of the tool according to the majority of the participants was the visualisations that could be accessed instantly after running the experiments.

The opinions on the installation of the tool were very mixed. In order to run the GUI, the user needs to have Python, Java and pip. The full installation guide that was given to the participant is shown in Appendix F. Some people found it quite difficult to set up everything, while others did it by themselves without any problems. From this we can conclude that the difficulty of the installation is largely determined by the skills of the user. Some participants were already familiar with Python and had it installed, while others had never used it before.

A common point of general feedback was to make a user manual with examples and explanations of each feature. Since the exact values and format of the input fields can be unclear to some people, it would be useful to be able to look at some documentation. There are already help buttons that explain the basics in the GUI itself, but writing a separate document dedicated to the explanation might be easier to understand. This would also allow us to explain the visualisations in more detail, so people know exactly what the graphs mean and how to interpret them.

7
Discussion

The main goal of this thesis was to determine how data mining tools could be made accessible for people in the field of ABMS. From the results of the usability study, we saw that all the measure goals were achieved, often with a wide margin for both the times and questionnaires.

The most important measure is the SUS questionnaire in my opinion, because it allows the participant to give their opinion on the usability of the entire tool. The other two measures, times and ASQ, are both related to a specific scenario and thus may not provide a complete picture of the entire tool. In Figure 7.1, a grading scale for SUS is shown, which was developed by analysing the SUS scores for 964 participants across different domains [10]. The average SUS score for our tool was 85.17, which can be classified as excellent. The usability study was done with a total of 14 participants, which has been shown to be enough to produce reliable results [96]. This conclusion, along with the positive results for the times and ASQ scores, proves that we have reliable results that show that the participants from the target group considered the tool as very easy to use.

The research question of this thesis is defined as: How can data mining tools be made accessible to inexperienced programmers in order to analyse NetLogo simulations? With the information gathered during this thesis with the survey, implementation and usability study, we can answer this question. There are two important aspects that make a tool accessible. The first one is to ensure that the user does not need to learn new things in order to use the tool. For this research, this mostly relates to programming languages, since that is the most common way to analyse data. In order to ensure that the user does not need to do any programming, a GUI can be developed where the user can design, execute and analyse experiments by simply inputting the necessary values in text and pressing buttons. The second aspect of accessibility is to make the usage of the tool similar to what the target group is already familiar with. In our case, this was the NetLogo BehaviorSpace, since that is currently the most commonly used technique to execute NetLogo experiments. This approach increases the ease of use and thus the accessibility, because the users can simply perform the actions that they are used

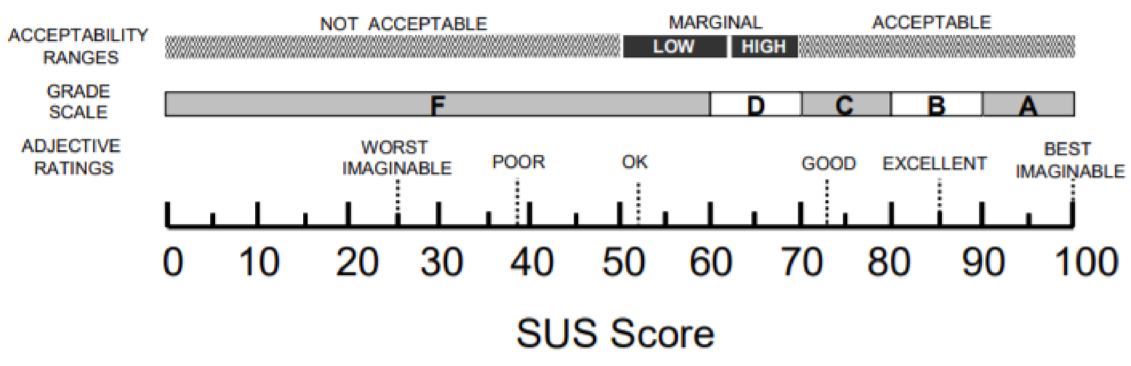


Figure 7.1: A comparison of the adjective ratings, acceptability scores, and school grading scales, in relation to the average SUS score. *Image & caption obtained from [10].*

to already.

This thesis has identified the primary shortcomings in the field of DM in ABMS and attempted to find a solution for these shortcomings. As explained in the introduction, little research has been done towards finding a useful combination of the field of DM and ABMS [6]. To the best of my knowledge, the only papers discussing the combination of these methods in detail are [6, 11, 78]. However, these papers are written purely from a theoretical standpoint and do not provide usable implementations to supplement their research. There are existing libraries that provide functionality for DM in ABMS, such as EMA Workbench and OpenMOLE, but these all lack in either functionality or usability.

The main cause of the current shortcomings in my opinion is the lack of UCD during the development of tools for ABMS, specifically NetLogo. All of the tools that allow users to design, execute and analyse experiments are targeted towards a user base that is experienced with programming in Python, R or MATLAB. Due to the ease of use and low skill threshold for NetLogo itself, this can not always be expected from the users [95]. In my opinion, this discrepancy between the required skills and actual skills is caused by the fact that most of the tools start off as a project that the developer creates for themselves. This means the usage of the tool is tailored towards their own skill set and not that of other people who might also benefit from its functionality. The results from this thesis have shown that developing a tool that is accessible to the entire NetLogo user base is certainly possible. The priority of the developers of such tools needs to be on the end-users and their skills, not only for the implementation, but also for the design, evaluation and maintenance. A large part of this accessibility can be obtained by providing the user with a GUI, since it circumvents the need to code completely and has been proven to increase the usability of systems [36].

8 Conclusion

During this masters thesis, we have attempted to answer the research question: *How can data mining tools be made accessible to inexperienced programmers in order to analyse NetLogo simulations?* The biggest limitation with the currently available tools is the fact that they require programming experience in data analysis languages such as Python or R. Many model designers lack this programming experience, which causes a barrier between them and the data mining tools that many do not overcome.

In order to address this problem, a tool was developed during this thesis which makes these data mining tools available to the user without having to do any programming themselves. The methodology of UCD was used during the design and implementation of the tool, which was chosen due to the high priority on ease of use and usability. This approach resulted in a GUI that the user interacts with by simply filling out text fields and clicking buttons. With the GUI, the user is able to design, run and visualise NetLogo experiments without writing a single line of code themselves. This addresses all of the problems with the currently available tools, which often lack in functionality or usability and provides a usable alternative for model analysis besides the NetLogo BehaviorSpace.

The tool was evaluated by performing a formal usability study with participants from the target group. During this study, the participants executed three predefined tasks with the tool after which they gave their opinion on the usability by filling out questionnaires. Every measurement goal that was set before the usability study was achieved, most of them with wide margins. The ASQ questionnaires obtained an average result of 4.69 out of 5, where the target was set at 4. Furthermore, the SUS questionnaires obtained an average result of 85 out of 100, with the target being 75. These results prove that our tool is easy to use and thus that we successfully made advanced data mining methods accessible to the target group.

During the 9 months of this thesis, I have learned a lot about doing academic research, user interaction, software development and ABMS. The main improvement I would have liked to make on the process is to handle the thesis more as academic research earlier on, rather than a software development project. In the early stages of the thesis, I focused mainly on the implementation and description of the tool itself, rather than the research and impact behind it. Despite this, I think the end result of this thesis can be considered as successful. We have answered all the research questions, provided the target group with an excellent alternative to the current tools for the analysis of ABMS and produced meaningful and relevant results that can be used to improve the development of such tools in the future. The results show that making DM accessible for ABMS is certainly possible, so I hope that this work will inspire others to adopt the approach of UCD and consider the needs of their end-users during development.

8.1. Future Work

Despite the good results that were found during this research, there are more directions in this field that could be interesting to explore in future work. These directions are related to the tool and usability study, as well as recommendations for the approach and development of DM tools in general.

The first direction is related to the fact that the tool is implemented in Python and thus also needs to be run from Python. Since many people in the target group have never used Python before, the installation and running of it could be difficult. An interesting direction for future research here would be to develop a similar tool that is built into NetLogo, so it can easily be accessed and used by everyone that uses NetLogo without installing external software. This would increase the usability and accessibility drastically, since users have access to the analysis tool within the NetLogo environment that they are familiar with, similar to the BehaviorSpace. This would also help with the distribution of the tool, since currently it is only available on the GitHub repository that the code is published on. Being able to access the tool from within NetLogo itself could lead to more widespread use.

The second limitation is the fact that the usability study was mainly focused on the ease of use and not as much on the functionality itself. The main goal of the usability study was to determine whether the tool could be considered accessible to the target group, which it did achieve. However, the secondary goal of identifying weaknesses or gaps in terms of the functionality of the tool was not achieved in a formal way that allows us to evaluate this aspect. The participants did answer some open questions after completing the tasks about the functionality, but these answers were more considered as inspiration rather than concrete requirements. I think there could be value in a formal evaluation of the functionality on top of the usability in order to gather measures and feedback that can be used to make improvements to the tool in the future.

The last recommendation for future work is to incorporate UCD into the development process of DM tools. This thesis has proven that it is possible to develop an accessible and easy to use tool for ABMS by considering the needs and skills of the user first. I think currently too many developers focus on the functionalities without thinking about the usability of the software, which often makes tools inaccessible to a large part of the intended user base. By using UCD throughout the entire development process, the usability of the tools can be guaranteed, which leads to a wider range of available analysis options for model developers.

A
Target Group Survey

Which of these tools for experimental analysis of ABMS have you heard of before?

1. I have done my analytics without coupling them to NetLogo and used programs that I write in C when necessary.
2. NLExperiment (R)
3. PyNetLogo (Python)
4. My own R package for DOE experiments
5. PyNetLogo (Python)
6. -

Which of these tools have you used for experimental analysis of ABMS?

1. same as above
2. -
3. None
4. My own R package for DOE experiments
5. -
6. -

What is currently preventing you from using external data analysis tools?

1. The ones you mention need more work for familiarising than writing ad hoc programs in C
2. They are language specific.
3. They are usually broken / buggy by the time you try them and the developer has left the building.
4. I used to develop in Java and analyse in R which seemed sufficient. I haven't done ABM for some time recently.
5. I couldn't get them to work with Netlogo
6. I don't know what these tools are.

Which requirements would you define to make the tool easy to use for non programmers?

1. It would be good if one could specify them much like the specifications in the BehaviorSpace

2. Not having to learn a new programming language.
3. Updatable, non-buggy, supported by Netlogo / community
4. Easy-to-use interfaces with existing ABM software (Netlogo, Repast, mesa). good documentation and feature description, examples, tutorials
5. Very clear documentation in plain language with lots of examples, graphical interface
6. Well, NetLogo has a ”Behavior Space” that seems to fit your definition. It’s enough for me.

What type of information would you like my tool to present? (e.g variables, locations, agent counts)

1. Distributions, histograms, parameters of distributions (e.g. the two parameters of a beta distribution that best fits the simulated distribution of an agent variable, but also regressions, cross tabulations etc. for sensitivity analyses of multi-runs in BehaviorSpace).
2. I would like the tool to be applicable to a database of results (which could be about anything that comes from a simulation)
3. Any variable, any quality available in Netlogo
4. interaction with parameters / agent properties with meso/macro level result variables
5. Summary statistics about agents and environmental variables with ability to deep dive into those and visualise/analyse them in different ways; time series; ability to calculate other types (custom) of population and individual-level summary statistics ’on the go’
6. That’s already in NetLogo’s Behavior Space.

How should this information be presented? (e.g, types of graphs/tables/visualisation)

1. Distributions as histograms, 2d- and 3d-scattergrams, heatmaps, violin maps, box plots.
2. As many forms/visualisations as possible
3. Graphs and tables - integrating with nineplot or similar
4. Starting from overview figures, details should be explorable (interative use by limiting agent set whose properties is analyses, e.g.)
5. Tables, time series plots, histograms, etc. - depending on the statistic
6. I prefer to create my own graphs.

Do you have any final comments/advice?

1. -
2. -
3. Please do this :)
4. Make a sound survey on existing software, also Java-based, and try to extend and improve existing software rather than starting from scratch with very limited time.
5. Really interesting idea! There are so many directions you could take this, but no reason it can’t be ongoing/open access after your Masters finishes, so others can pick it up and add a module or make suggestions too.
6. Rather than tools for analyzing data, it would be good to have tools that add computational functionalities to NetLogo. [*redacted for privacy*] has an on-going project on this.

What are the domain(s) you have made models for (in keywords)?

1. opinion formation, organisation theory (task allocation), normative behaviour
2. Many! Political Science, social norms, negotiation, voting, mutual influence
3. Transport systems health systems social systems
4. environmental psychology, land use, energy
5. Socio-ecological systems
6. Social Science

B
Experimental Design

In Figure B.1, the GUI screen for the experimental design of parameter space search runs is shown. The design screen for the reporter value analysis runs is similar, except that it removes the sampling options and the number of scenarios. In the figure it can be seen that most of the fields have a question mark button next to them. If the user clicks this, a new window will pop up with more explanation on what a field is and what it is used for. This method is better than putting the explanation directly on the GUI in my opinion, since it reduces clutter and only explains what the user does not already know. The rest of the GUI screens other than the experimental design are shown in Appendix E.

The first thing is the experiment name, which is not used for much other than allow the user to give a clear name for a configuration when saving and importing it. The next field is the model file, which needs to be a valid path to a NetLogo model. This can either be typed out, or the import button can be used which opens a file explorer and imports the path when the user selects a file. The most important input is the multiline which allows the user to input the parameter settings for the run. This is followed by the sampling options and number of scenarios, which are only options for the parameter space search runs. The next two inputs are the number of repetitions that will be run for each parameter configuration and the maximum number of ticks that a simulation can run before it is stopped. After this, the NetLogo reporters that will be recorded during the simulation can be specified, similarly to the BehaviorSpace. This is followed by the setup commands field, which are all the commands that will be ran before the simulation starts. Finally, the number of parallel executors can be specified, which can speed up the experiment by running independent simulations in parallel.

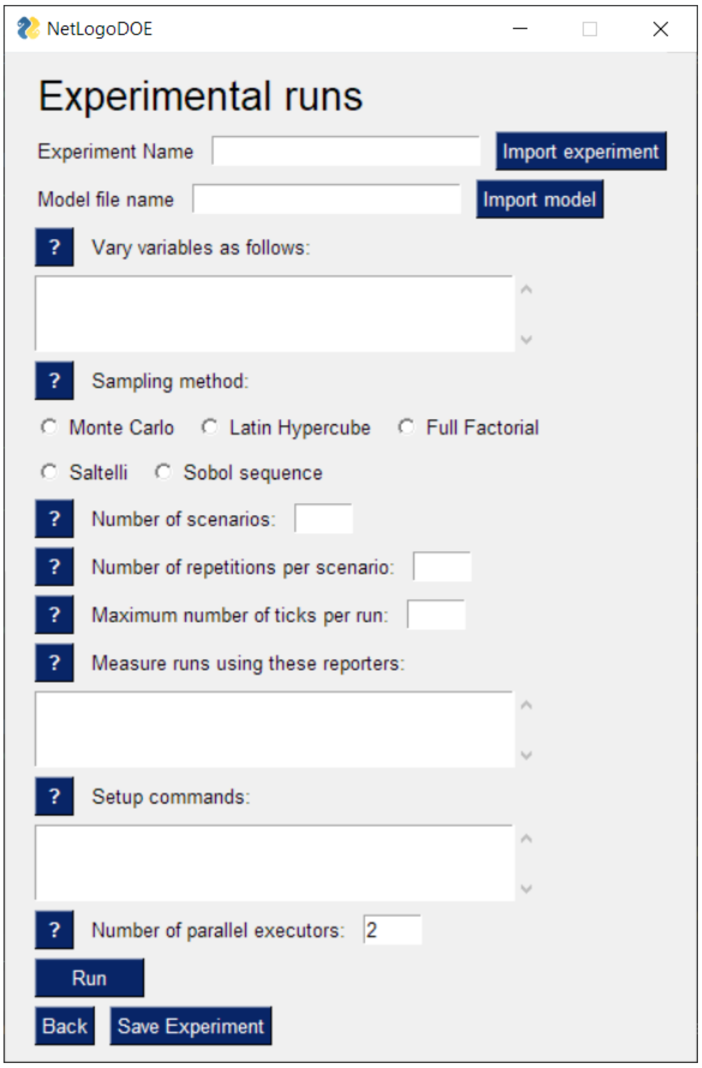


Figure B.1: The screen in the GUI that allows the user to input an experimental design for parameter space search.

C Usability Study Procedure

The main question for the usability study is the usability of the tool, so the procedure must result in some measures of the usability that allow the question to be answered. This is done by defining scenarios that the user will execute during the study. It is important for these scenarios to be representative of real-world applications of the tool in order for the evaluation to have meaningful results [48].

In order to obtain metrics of the usability for the entire system, the scenarios have been defined such that all of them combined use nearly all components of the tool. In this study, every scenario starts with the user launching the GUI and ends with the user closing the GUI. The following 3 scenarios have been defined:

1. Fill in a valid parameter space search configuration, run the model, visualize the result with parallel coordinates and a scatterplot.
2. Fill in a valid reporter value analysis configuration, run the model, visualize the result with a timeseries plot and a boxplot.
3. Fill in a valid experiment configuration, save the configuration, restart the GUI, load the saved configuration.

In Section C.1, the full instructions that the user was given for all three of these scenarios are shown. All studies were done with the same model, which was the Wolf Sheep predation model from the NetLogo model library. This model was chosen because it is a well known NetLogo model, so most people experienced with NetLogo will be relatively familiar with it. The values that the participants had to input were specified in the scenarios, rather than letting them build their own experiment from scratch. The main reason for giving them the input is to make it easier for the user, because they may not know the exact names and values of the model parameters. This also ensures that the execution times will be similar for all participants, thus making it easier to compare the results. In a real-world scenario, the user would also have an idea of what input they want in their experiment when they are using the tool, so these tasks are still representative of the intended use case.

All usability studies were done online in a voice call with the user (Skype, Zoom etc.). During the study, the user would share their screen and execute the three scenarios. I did not interfere during the execution of the scenarios, unless the user seemed to be stuck or explicitly asked for help. After each scenario, the user would fill in a short questionnaire about that scenario, along with a larger questionnaire when all scenarios were finished.

C.1. User Instructions

Scenario 1

1. Launch the GUI.
2. Navigate to parameter space search.
3. Import the Wolf Sheep predation model (the input needs to be a file path to the .nlogo model file).

4. Set the following variables:
    - initial-number-sheep with a lower bound of 50 and an upper bound of 150.
    - initial-number-wolves with a lower bound of 40 and an upper bound of 60.
    - sheep-reproduce with a value of 6
5. Choose appropriate values for the number of scenarios, repetitions and ticks (somewhere around 20, 10, 100 respectively).
6. Choose reporters that will count the number of sheep and wolves during the run.
7. Run the configuration.
8. Visualise average of the results with parallel coordinates.
9. Visualise the last tick of the results with a scatter plot with the initial-number-sheep on the x-axis and the sheep count on the y-axis.
10. Close the GUI.

Scenario 2

1. Launch the GUI.
2. Navigate to reporter value analysis.
3. Import the Wolf Sheep predation model (the input needs to be a file path to the .nlogo model file).
4. Set the following variables:
    - initial-number-sheep with a value of 70.
    - initial-number-wolves with a value of 80.
    - sheep-reproduce with a value of 6.
    - wolf-reproduce with a value of 7.
5. Choose appropriate values for the number of repetitions and ticks (somewhere around 10, 100 respectively).
6. Choose reporters that will count the number of sheep and wolves during the run.
7. Run the configuration.
8. Visualise the average value of the results with a time series that shows shaded areas for the variance, for all reporters.
9. Visualise the values of the sheep count with a boxplot.
10. Close the GUI.

Scenario 3

1. Launch the GUI.
2. Navigate to reporter value analysis.
3. Fill in any valid configuration for the Wolf Sheep predation model (for example the one from scenario 2).
4. Save this experiment in a .txt file on your PC.
5. Close the GUI.
6. Launch the GUI.
7. Navigate to reporter value analysis.
8. Import the saved experiment from the .txt file.
9. Close the GUI.

D System Usability Scale Score Calculation

The SUS questionnaire consists of 10 questions about the usability of the system. The answers are given on a 5-point Likert scale ranging from strongly agree to strongly disagree. The 10 questions can be split into two groups, namely positive and negative questions. Every odd numbered question (1, 3, 5, 7, 9) is positively phrased, such as *"I thought the system was easy to use"*. Every even numbered question (2, 4, 6, 8, 10) is negatively phrased, for example *"I found the system unnecessarily complex"*. In order to calculate the final result, the score of every even numbered question is subtracted from 5 and 1 is subtracted from the score of each odd numbered question. After this, the scores from all 10 questions are summed and multiplied by 2.5, which gives a value between 0 and 100. An example of this calculation is given in the diagram in Figure D.1.

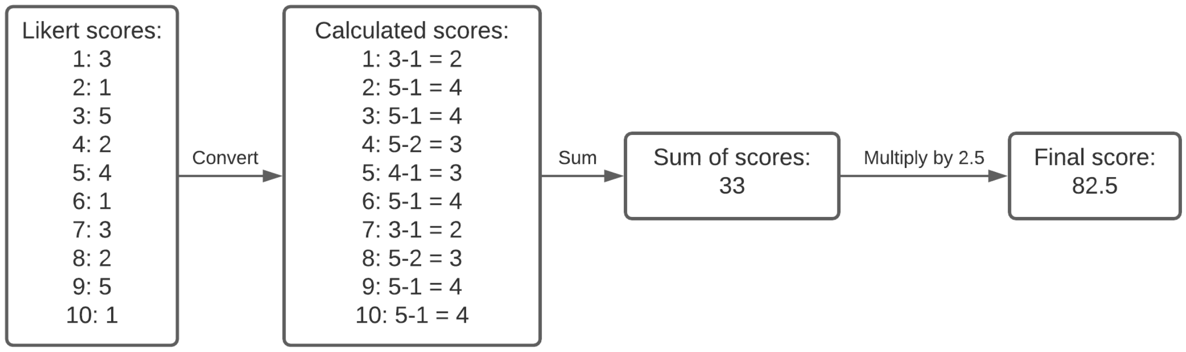
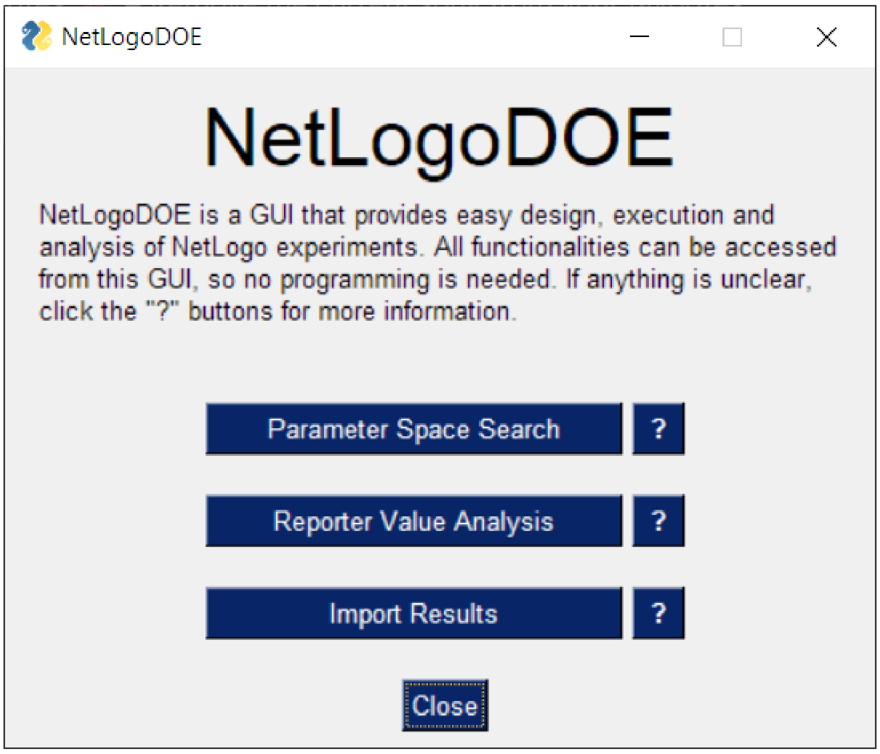


Figure D.1: Example of the calculation of the SUS questionnaire score.

E Graphical User Interface

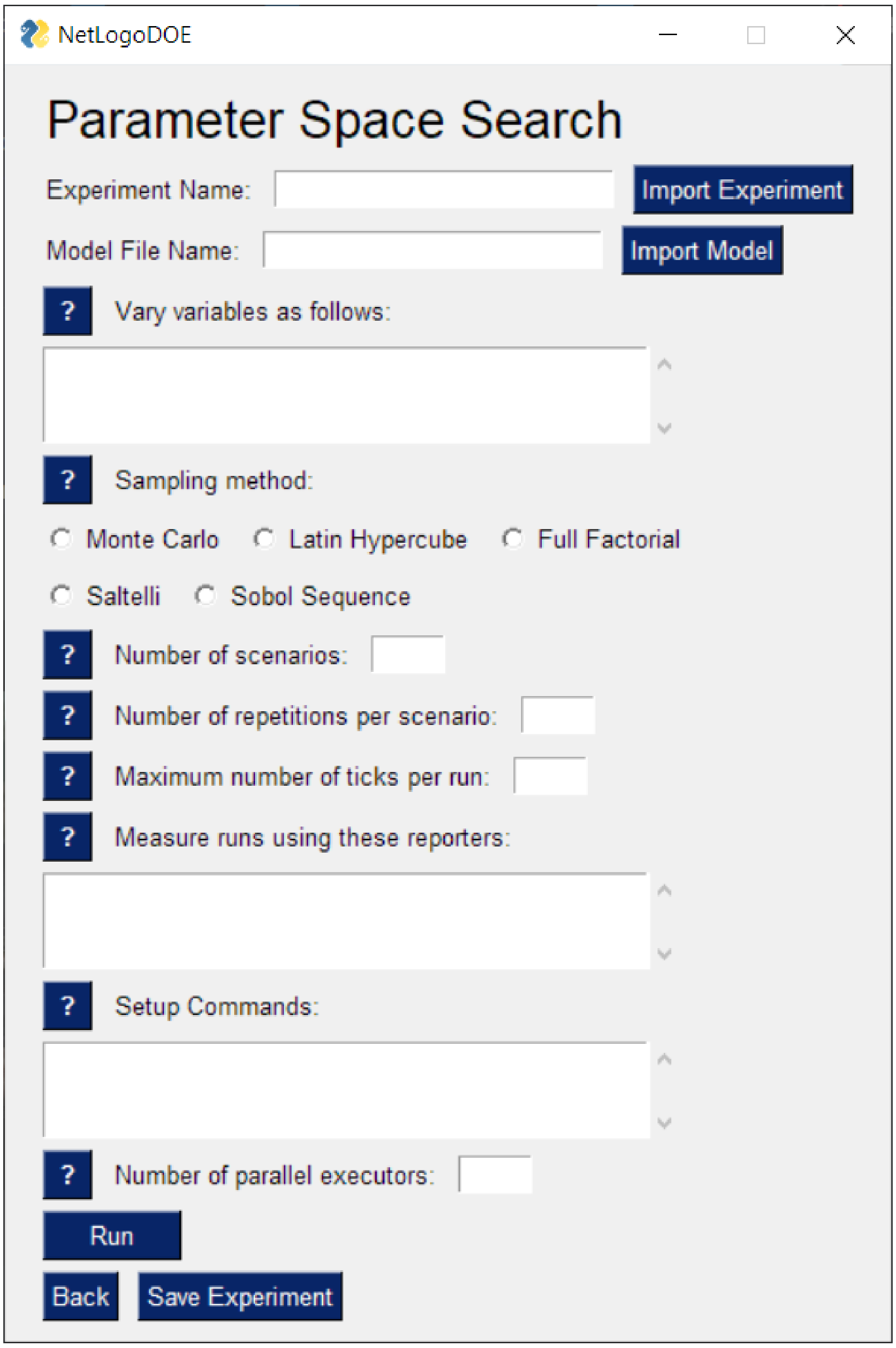
This appendix will show the different screens that the user has access to within the GUI. These screens can be used for the general navigation, experimental design and visualisations. The GUI itself is available with the NetLogoDOE Python package from: `https://pypi.org/project/NetLogoDOE/`. The GitHub repository for the package can be found at: `https://github.com/robinfaber97/NetLogoDOE`.


NetLogoDOE

NetLogoDOE

NetLogoDOE is a GUI that provides easy design, execution and analysis of NetLogo experiments. All functionalities can be accessed from this GUI, so no programming is needed. If anything is unclear, click the "?" buttons for more information.

Figure E.1: The index screen that is shown when the GUI is lauched. This provides a small explanation of the tool along with the two experimental design options and the ability to import previous results.



Experiment Name:

Model File Name:

Vary variables as follows:

Sampling method:

Monte Carlo ○ Latin Hypercube ○ Full Factorial ○ Saltelli ○ Sobol Sequence

Number of scenarios:

Number of repetitions per scenario:

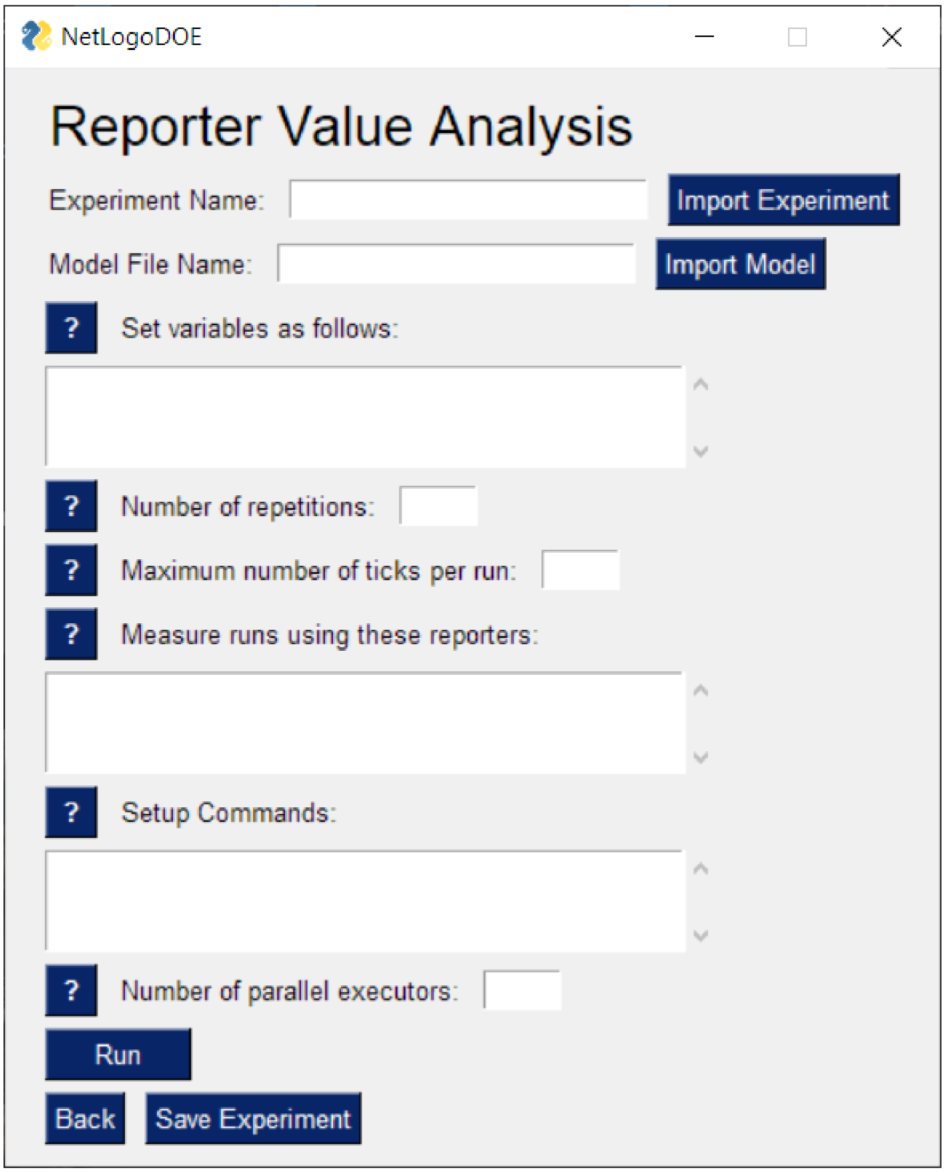
Maximum number of ticks per run:

Measure runs using these reporters:

Setup Commands:

Number of parallel executors:

Figure E.2: Experimental design screen for the Parameter Space Search run type, as explained in 4.2.1.


NetLogoDOE

Reporter Value Analysis

Experiment Name:

Model File Name:

Set variables as follows:

Number of repetitions:

Maximum number of ticks per run:

Measure runs using these reporters:

Setup Commands:

Number of parallel executors:

Figure E.3: Experimental design screen for the Reporter Value Analysis run type, as explained in 4.2.1.

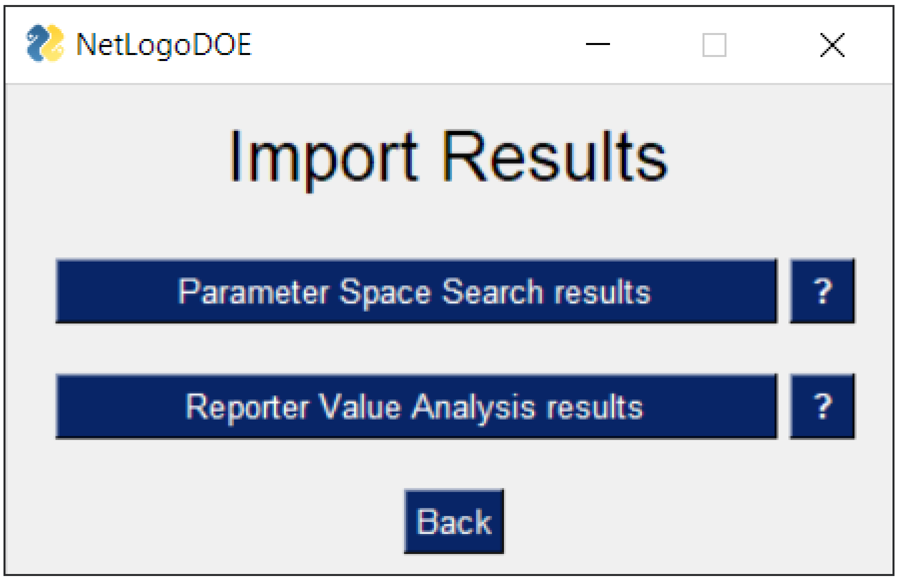

NetLogoDOE
Import Results
Parameter Space Search results
?
Reporter Value Analysis results
?
Back

Figure E.4: Screen where the user can import the results from previous experiments to get access to the visualisations, for both run types.

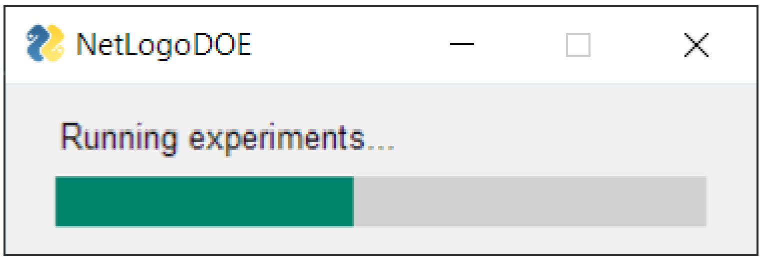


Figure E.5: Screen the user sees while the experiments are running in the background. An interactive progress bar is shown to give an indication of the progress of the experiments.

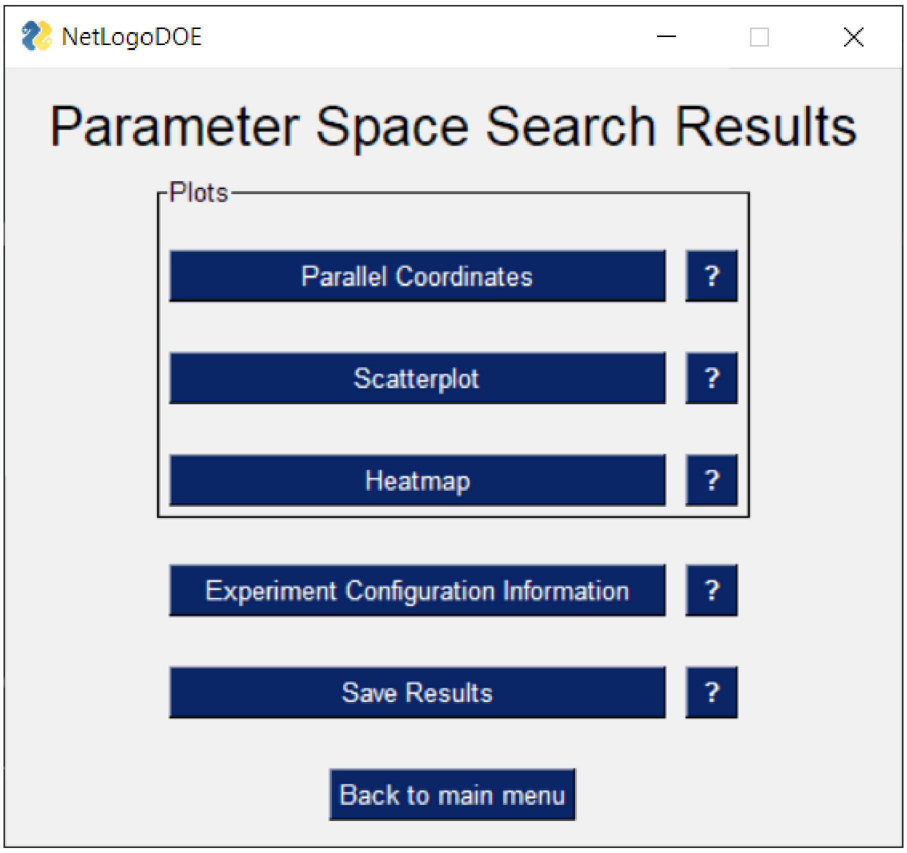


Figure E.6: The result screen that is shown when the PSS experiments are finished running, or when a user imports results. Here, the user can access the visualisations and save the results for future use.

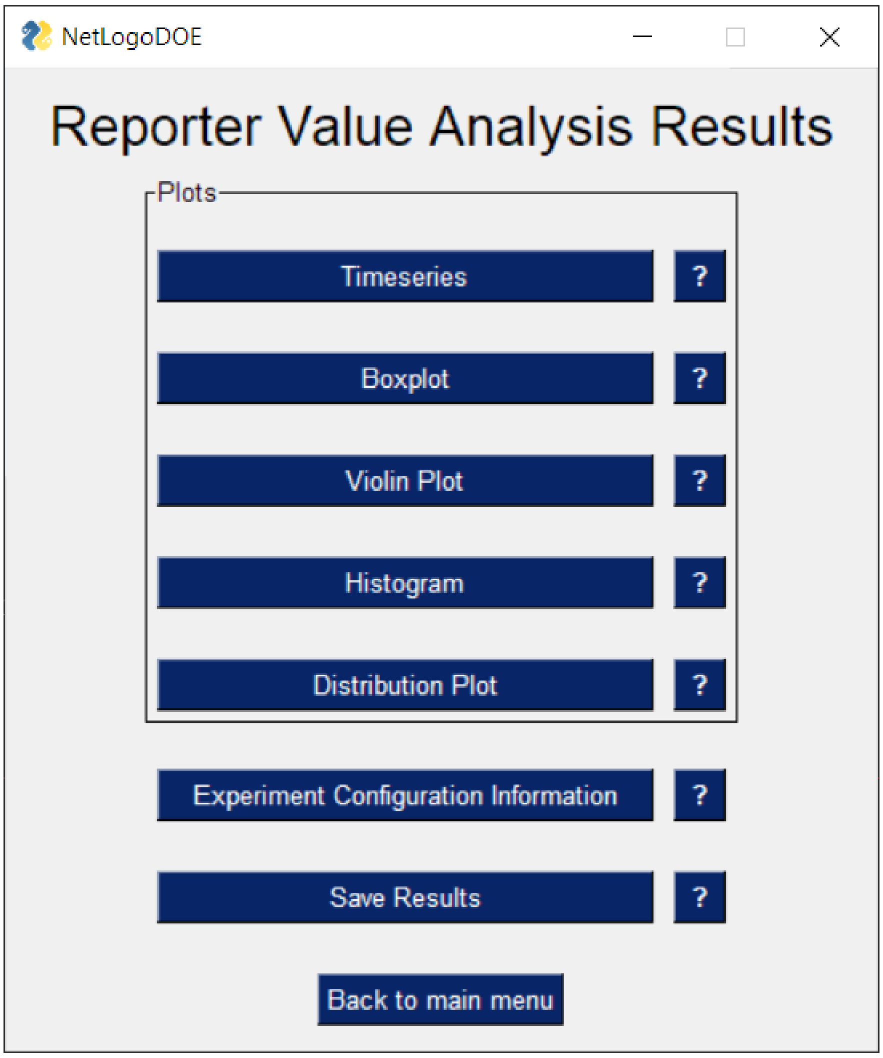


Figure E.7: The result screen that is shown when the RVA experiments are finished running, or when a user imports results. Here, the user can access the visualisations and save the results for future use.

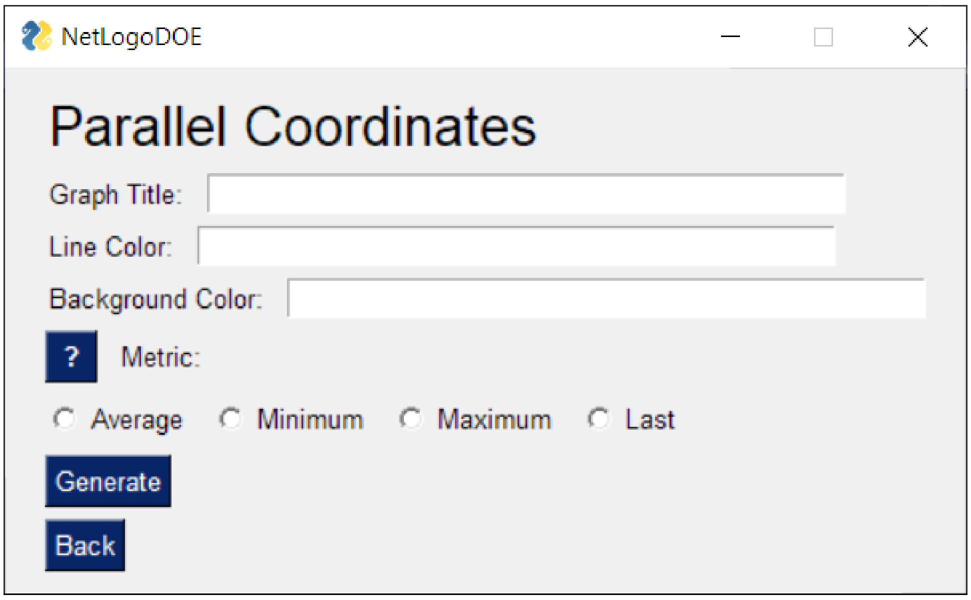


Figure E.8: Example of a visualisation screen, in this case for the parallel coordinates for the PSS run type. The user can customize the graph here and decide which metric should be used to merge the data.

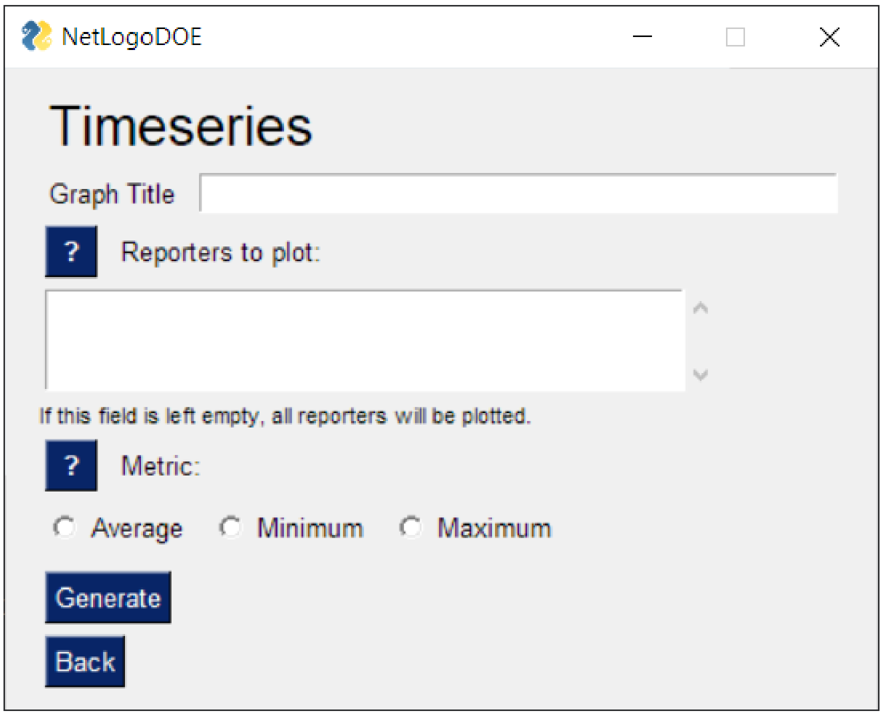
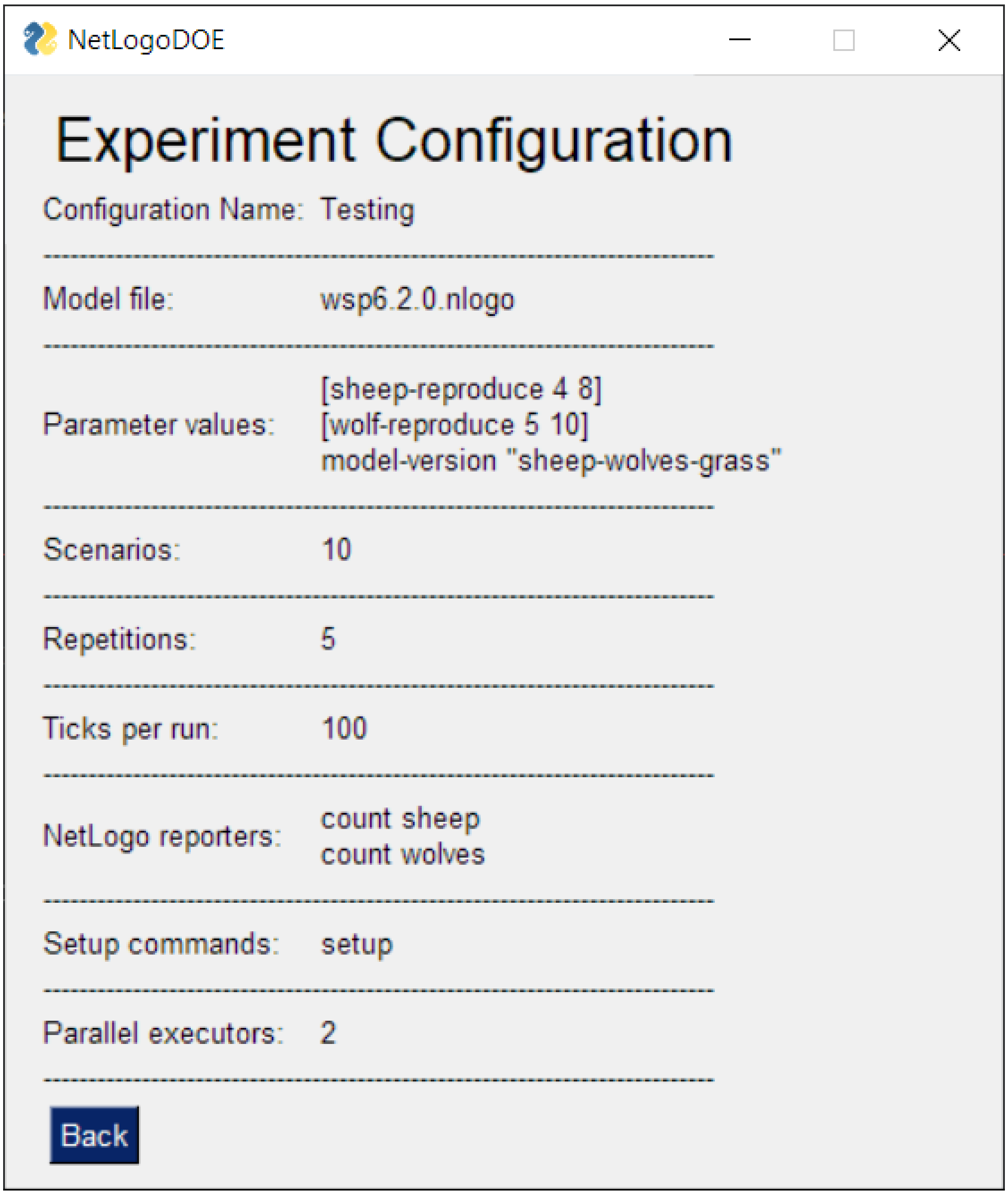


Figure E.9: Example of a visualisation screen, in this case for the timeseries plot for the RVA run type. The user can select which of the reporters should be plotted and which metric should be used to merge the data.



Experiment Configuration

Configuration Name:	Testing
Model file:	wsp6.2.0.nlogo
Parameter values:	[sheep-reproduce 4 8] [wolf-reproduce 5 10] model-version "sheep-wolves-grass"
Scenarios:	10
Repetitions:	5
Ticks per run:	100
NetLogo reporters:	count sheep count wolves
Setup commands:	setup
Parallel executors:	2

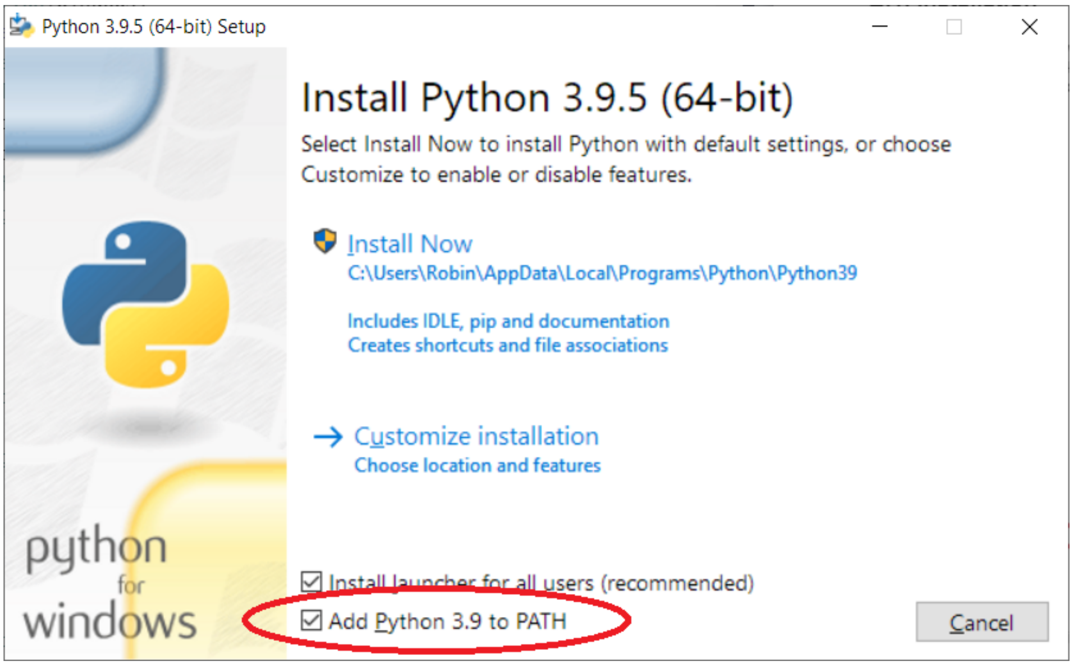
Figure E.10: Configuration information that the user can access from the result screen of both run types. This screen shows the experimental design that the user inputted when running the experiments.

F
Installation Guide

This appendix will give a step by step guide on the installation of the tool, both for Windows and MacOS. All of the steps, except 4 and 5, are independent of the others and can be done in any order. If one of the steps is already completed on your computer, you can simply skip it.

F.1. Windows

1. Install Python (Windows installer (64-bit)): `https://www.python.org/downloads/windows/`. Make sure to click the checkbox to add it to the PATH environment variable to allow you to run Python anywhere on your PC, as shown in Figure F.1.


Python 3.9.5 (64-bit) Setup

Install Python 3.9.5 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Robin\AppData\Local\Programs\Python\Python39

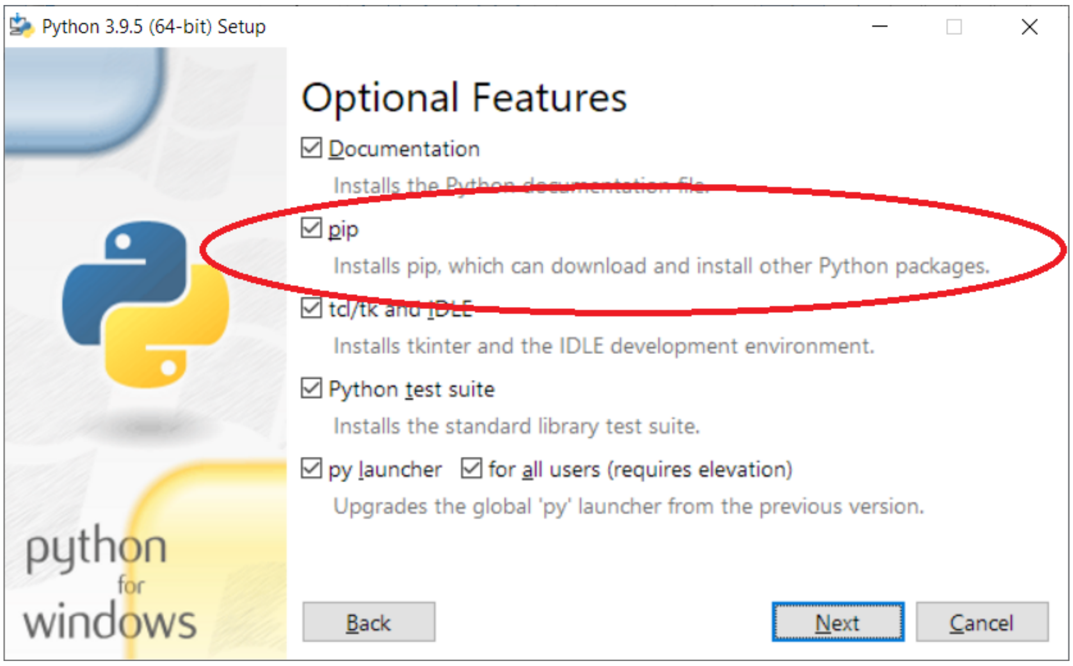
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

Figure F.1: Python installer screen showing the checkbox that ensures your Python installation gets added to PATH.

Make sure to go to customize installation and make sure pip is checked (it should be by default), as shown in Figure F.2. After this go back and click Install Now.


Python 3.9.5 (64-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Upgrades the global 'py' launcher from the previous version.

Figure F.2: Python installer screen showing the checkbox that ensures your Python installation includes pip.

2. Open a command prompt by searching for cmd in Windows, it should be the first result as shown in Figure F.3.

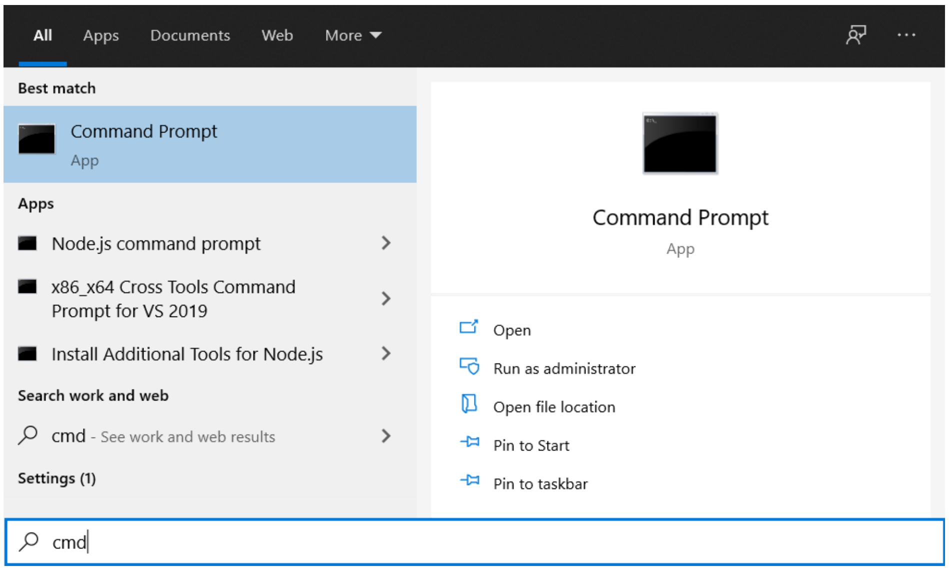
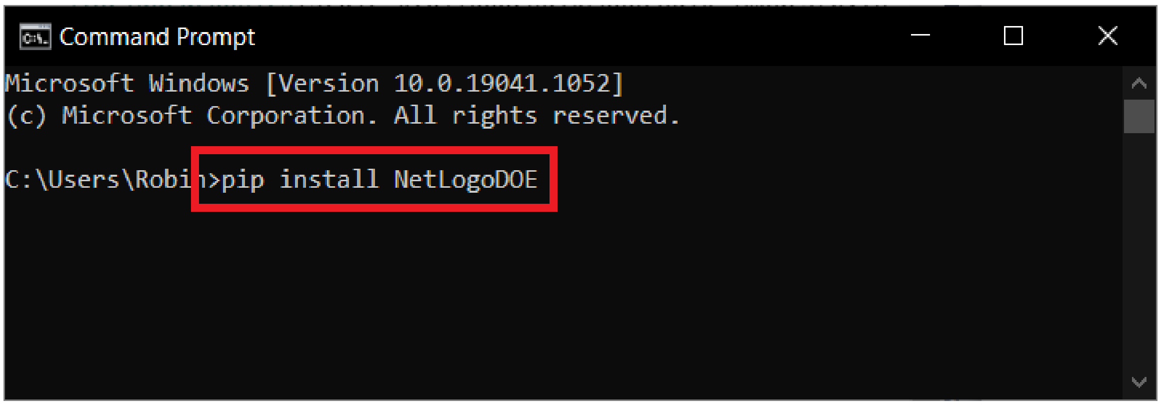


Figure F.3: Windows search showing how to open a command prompt.

In this command prompt you can install NetLogoDOE, which is the Python package that contains the GUI. This can be done by running the command: pip install NetLogoDOE, as shown in Figure F.3.

3. Install NetLogo: `https://ccl.northwestern.edu/netlogo/download.shtml`. Any version should be compatible, but it is always recommended to upgrade to the latest stable version, which is 6.2.0 at the time of writing this. I can not guarentee compatibility for any NetLogo version later than 6.2.0. If you already have this version installed on your PC, you can skip this step.

4. Install Java by downloading the installer from: `https://www.java.com/en/download/`.



```
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Robin>pip install NetLogoDOE
```

Figure F.4: Command prompt showing how to install the GUI package.

Follow the instructions in the installer, the default values should be correct for everything.

5. Set the JAVA_HOME environment variable on your computer, this allows Java to be executed from anywhere on your PC, similar to the PATH variable with Python. The instructions for this process can be found on the following link: `https://javatutorial.net/set-java-home-windows-10`.

F.2. MacOS

1. Install Python (macOS 64-bit Intel installer): `https://www.python.org/downloads/mac-osx/`. Make sure to click the checkbox to add it to the PATH environment variable to allow you to run Python anywhere on your PC, as shown in Figure F.1.

   Go to customize installation and make sure pip is checked (it should be by default), as shown in Figure F.2. After this go back and click Install Now, this will take a few minutes.

2. Open a terminal. This can be done searching for terminal in the search menu, it should be the first result.

   In this terminal you can install NetLogoDOE, which is the Python package that contains the GUI. This can be done by running the command: pip install NetLogoDOE, as shown in Figure F.3. This figure shows the Windows command prompt, but the terminal in MacOS should look similar and requires the same command.

3. Install NetLogo: `https://ccl.northwestern.edu/netlogo/download.shtml`. Any version should be compatible, but it is always recommended to upgrade to the latest stable version, which is 6.2.0 at the time of writing this. I can not guarentee compatibility for any NetLogo version later than 6.2.0. If you already have this version installed on your PC, you can skip this step.

4. Install Java by downloading the installer from: `https://www.java.com/en/download/`. Follow the instructions in the installer, the default values should be correct for everything.

5. Set the JAVA_HOME environment variable on your computer, this allows Java to be executed from anywhere on your PC, similar to the PATH variable with Python. The instructions for this process can be found on the following link: `http://www.sajeconsultants.com/how-to-set-java_home-on-mac-os-x/`.

Bibliography

[1] Sameera Abar, Georgios K Theodoropoulos, Pierre Lemarinier, and Gregory MP O'Hare. Agent Based Modelling and Simulation Tools: A Review of the State-of-Art Software. *Computer Science Review*, 24:13–33, 2017.

[2] Chadia Abras, Diane Maloney-Krichmar, Jenny Preece, et al. User-Centered Design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456, 2004.

[3] Samuel Soma Ajibade and Anthonia Adediran. An Overview of Big Data Visualization Techniques in Data Mining. *International Journal of Computer Science and Information Technology Research*, 4(3):105–113, 2016.

[4] Simone Alfarano, Thomas Lux, and Friedrich Wagner. Estimation of a Simple Agent-Based Model of Financial Markets: An Application to Australian Stock and Foreign Exchange Data. *Physica A: Statistical Mechanics and its Applications*, 370(1):38–42, 2006.

[5] Jiju Antony. Full Factorial Designs. In *Design of Experiments for Engineers and Scientists*, pages 63–85. Elsevier, Oxford, 2nd edition, 2014.

[6] Javier Arroyo, Samer Hassan, Celia Gutiérrez, and Juan Pavón. Re-Thinking Simulation: A Methodological Approach for the Application of Data Mining in Agent-Based Modelling. *Computational and Mathematical Organization Theory*, 16(4):416–435, 2010.

[7] Ivo Babuska and J Tinsley Oden. Verification and Validation in Computational Engineering and Science: Basic Concepts. *Computer methods in applied mechanics and engineering*, 193(36): 4057–4066, 2004.

[8] Jennifer Badham and Judy-anne Osborn. Zombies in the City: A Netlogo Model. *Mathematical Modelling of Zombies*, pages 209–232, 2014.

[9] Allison H Baker, Haiying Xu, John M Dennis, Michael N Levy, Doug Nychka, Sheri A Mickelson, Jim Edwards, Mariana Vertenstein, and Al Wegener. A Methodology for Evaluating the Impact of Data Compression on Climate Simulation Data. In *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*, pages 203–214, 2014.

[10] Aaron Bangor, Philip Kortum, and James Miller. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal of usability studies*, 4(3):114–123, 2009.

[11] Omar Baqueiro, Yanbo J Wang, Peter McBurney, and Frans Coenen. Integrating Data Mining and Agent Based Modeling and Simulation. In *Industrial Conference on Data Mining*, pages 220–231. Springer, 2009.

[12] Behzad Behdani, Zofia Lukszo, Arief Adhitya, and Rajagopalan Srinivasan. Performance Analysis of a Multi-Plant Specialty Chemical Manufacturing Enterprise Using an Agent-Based Model. *Computers & Chemical Engineering*, 34(5):793–801, 2010.

[13] Marco Better, Fred Glover, and Manuel Laguna. Advances in Analytics: Integrating Dynamic Data Mining With Simulation Optimization. *IBM journal of research and development*, 51(3.4):477–487, 2007.

[14] RB Bhise, SS Thorat, and AK Supekar. Importance of Data Mining in Higher Education System. *IOSR Journal Of Humanities And Social Science (IOSR-JHSS)*, 6(6):18–21, 2013.

[15] Jennifer Boger, Tammy Craig, and Alex Mihailidis. Examining the Impact of Familiarity on Faucet Usability for Older Adults With Dementia. *BMC geriatrics*, 13(1):1–12, 2013.

[16] Eric Bonabeau. Agent-Based Modeling: Methods and Techniques for Simulating Human Systems. *Proceedings of the national academy of sciences*, 99(suppl 3):7280–7287, 2002.

[17] Thomas F Brady and Edward Yellig. Simulation Data Mining: A New Form of Computer Simulation Output. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 5–pp. IEEE, 2005.

[18] Manuel Brhel, Hendrik Meth, Alexander Maedche, and Karl Werder. Exploring Principles of User-Centered Agile Software Development: A Literature Review. *Information and software technology*, 61:163–181, 2015.

[19] Saskia LGE Burgers, Gert Jan Hofstede, Catholijn M Jonker, and Tim Verwaart. Sensitivity Analysis of an Agent-Based Model of Culture's Consequences for Trade. In *Progress in Artificial Economics*, pages 253–264. Springer, 2010.

[20] Steven Burrows, Benno Stein, Jörg Frochte, David Wiesner, and Katja Müller. Simulation Data Mining for Supporting Bridge Design. In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*, pages 163–170, 2011.

[21] Benoit Calvez and Guillaume Hutzler. Parameter Space Exploration of Agent-Based Models. In *international conference on knowledge-based and intelligent information and engineering systems*, pages 633–639. Springer, 2005.

[22] Francesca Campolongo, Jessica Cariboni, and Andrea Saltelli. An Effective Screening Design for Sensitivity Analysis of Large Models. *Environmental modelling & software*, 22(10):1509–1518, 2007.

[23] Guglielmo Maria Caporale, Antoaneta Serguieva, and Hao Wu. Financial Contagion: Evolutionary Optimisation of a Multinational Agent-Based Model. 2008.

[24] Jean-Christophe Castella, Tran Ngoc Trung, and Stanislas Boissau. Participatory Simulation of Land-Use Changes in the Northern Mountains of Vietnam: The Combined Use of an Agent-Based Model, a Role-Playing Game, and a Geographic Information System. *Ecology and Society*, 10(1), 2005.

[25] Karen Chan, Andrea Saltelli, and Stefano Tarantola. Winding Stairs: A Sampling Tool to Compute Sensitivity Indices. *Statistics and Computing*, 10(3):187–196, 2000.

[26] William S Cleveland and Robert McGill. The Many Faces of a Scatterplot. *Journal of the American Statistical Association*, 79(388):807–822, 1984.

[27] Richard E Cordes. The Effects of Running Fewer Subjects on Time-on-Task Measures. *International Journal of Human-Computer Interaction*, 5(4):393–403, 1993.

[28] Jonathan D Cryer and Kung-Sik Chan. *Time Series Analysis: With Applications in R*. Springer Science & Business Media, 2008.

[29] Susan Dean and Barbara Illowsky. Descriptive Statistics: Histogram. *Retrieved from the Connexions Web site: http://cnx. org/content/m16298/1.11*, 2009.

[30] Alexis Drogoul, Edouard Amouroux, Philippe Caillou, Benoit Gaudou, Arnaud Grignard, Nicolas Marilleau, Patrick Taillandier, Maroussia Vavasseur, Duc-An Vo, and Jean-Daniel Zucker. GAMA: Multi-Level and Complex Environment for Agent-Based Models and Simulations. In *12th International Conference on Autonomous agents and multi-agent systems*, pages 2–p. Ifaamas, 2013.

[31] Joseph S Dumas, Joseph S Dumas, and Janice Redish. *A Practical Guide to Usability Testing*. Intellect books, 1999.

[32] Stefano Federici and Simone Borsci. *Usability Evaluation: Models, Methods, and Applications*, pages 1–17. Center for International Rehabilitaion Research Information and Exchange (CIRRIE), 01 2010.

[33] Enrique Frias-Martinez, Graham Williamson, and Vanessa Frias-Martinez. An Agent-Based Model of Epidemic Spread Using Human Mobility and Social Network Information. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 57–64. IEEE, 2011.

[34] Michael Frigge, David C Hoaglin, and Boris Iglewicz. Some Implementations of the Boxplot. *The American Statistician*, 43(1):50–54, 1989.

[35] Arnaud Grignard, Patrick Taillandier, Benoit Gaudou, Duc An Vo, Nghi Quang Huynh, and Alexis Drogoul. GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In *International conference on principles and practice of multi-agent systems*, pages 117–131. Springer, 2013.

[36] Thomas Haslwanter. Useful Programming Tools. In *Hands-on Signal Analysis with Python*, pages 197–204. Springer, 2021.

[37] Julian Heinrich and Daniel Weiskopf. State of the Art of Parallel Coordinates. In *Eurographics (STARs)*, pages 95–116, 2013.

[38] Julian Heinrich, Yuan Luo, Arthur E Kirkpatrick, Hao Zhang, and Daniel Weiskopf. Evaluation of a Bundling Technique for Parallel Coordinates. *arXiv preprint arXiv:1109.6073*, 2011.

[39] Kasper Hornbæk. Current Practice in Measuring Usability: Challenges to Usability Studies and Research. *International journal of human-computer studies*, 64(2):79–102, 2006.

[40] Bertrand Iooss and Paul Lemaître. A Review on Global Sensitivity Analysis Methods. In *Uncertainty management in simulation-optimization of complex systems*, pages 101–122. Springer, 2015.

[41] Marc Jaxa-Rozen and Jan H Kwakkel. Pynetlogo: Linking Netlogo With Python. *Journal of Artificial Societies and Social Simulation*, 21(2), 2018.

[42] Tony Jenkins. On the Difficulty of Learning to Program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer, 2002.

[43] Timo Jokela, Netta Iivari, Vesa Tornberg, and Polar Electro. Using the ISO 9241-11 Definition of Usability in Requirements Determination: Case Studies. In *Proceedings of HCI2004: Design for life, the 18th British HCI group annual conference, Leeds Metropolitan University, UK. Eds. A. Dearden & L. Watts*, volume 2, pages 155–156, 2004.

[44] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert Scale: Explored and Explained. *Current Journal of Applied Science and Technology*, pages 396–403, 2015.

[45] Hamdi Kavak, Jose J Padilla, Christopher J Lynch, and Saikou Y Diallo. Big Data, Agents, and Machine Learning: Towards a Data-Driven Agent-Based Modeling Approach. In *Proceedings of the Annual Simulation Symposium*, pages 1–12, 2018.

[46] Jackie Kazil, David Masad, and Andrew Crooks. Utilizing Python for Agent-Based Modeling: The Mesa Framework. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 308–317. Springer, 2020.

[47] Alfred Kobsa, Rahim Sonawalla, Gene Tsudik, Ersin Uzun, and Yang Wang. Serial Hook-Ups: A Comparative Usability Study of Secure Device Pairing Methods. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, pages 1–12, 2009.

[48] Gerald P. Krueger. Book Review: Handbook of Human Factors and Ergonomics. *Ergonomics in Design*, 24(1):35–35, 2016.

[49] Satsuki Kumatani, Takayuki Itoh, Yousuke Motohashi, Keisuke Umezu, and Masahiro Takatsuka. Time-Varying Data Visualization Using Clustered Heatmap and Dual Scatterplots. In *2016 20th International Conference Information Visualisation (IV)*, pages 63–68. IEEE, 2016.

[50] Ahmed Laatabi, Nicolas Marilleau, Tri Nguyen-Huu, Hassan Hbid, and Mohamed Ait Babram. Odd+ 2d: An Odd Based Protocol for Mapping Data to Empirical ABMs. *Journal of Artificial Societies and Social Simulation*, 21(2), 2018.

[51] Emily G Lattie, Michael Bass, Sofia F Garcia, Siobhan M Phillips, Patricia I Moreno, Ann Marie Flores, JD Smith, Denise Scholtens, Cynthia Barnard, Frank J Penedo, et al. Optimizing Health Information Technologies for Symptom Management in Cancer Patients and Survivors: Usability Evaluation. *JMIR formative research*, 4(9):e18412, 2020.

[52] Dirk J Lehmann, Georgia Albuquerque, Martin Eisemann, Marcus Magnor, and Holger Theisel. Selecting Coherent and Relevant Plots in Large Scatterplot Matrices. In *Computer Graphics Forum*, volume 31, pages 1895–1908. Wiley Online Library, 2012.

[53] James R Lewis. Psychometric Evaluation of an After-Scenario Questionnaire for Computer Usability Studies: The ASQ. *ACM Sigchi Bulletin*, 23(1):78–81, 1991.

[54] James R Lewis and Jeff Sauro. Item Benchmarks for the System Usability Scale. *Journal of Usability Studies*, 13(3), 2018.

[55] Qi Li. Overview of Data Visualization. In *Embodying Data*, pages 17–47. Springer, 2020.

[56] Arika Ligmann-Zielinska. Spatially-Explicit Sensitivity Analysis of an Agent-Based Model of Land Use Change. *International Journal of Geographical Information Science*, 27(9):1764–1781, 2013.

[57] Wei-Liem Loh et al. On Latin Hypercube Sampling. *Annals of statistics*, 24(5):2058–2080, 1996.

[58] Guonian Lü, Michael Batty, Josef Strobl, Hui Lin, A-Xing Zhu, and Min Chen. Reflections and Speculations on the Progress in Geographic Information Systems (GIS): A Geographic Perspective. *International journal of geographical information science*, 33(2):346–367, 2019.

[59] Charles Macal and Michael North. Introductory Tutorial: Agent-Based Modeling and Simulation. In *Proceedings of the Winter Simulation Conference 2014*, pages 6–20. IEEE, 2014.

[60] Martin Maguire and Nigel Bevan. User Requirements Analysis. In *IFIP World Computer Congress, TC 13*, pages 133–148. Springer, 2002.

[61] David Masad and Jacqueline Kazil. Mesa: An Agent-Based Modeling Framework. In *14th PYTHON in Science Conference*, pages 53–60. Citeseer, 2015.

[62] Tim P Morris, Ian R White, and Michael J Crowther. Using Simulation Studies to Evaluate Statistical Methods. *Statistics in medicine*, 38(11):2074–2102, 2019.

[63] Aditya Nagori, Raghav Awasthi, Vineet Joshi, Suryatej Reddy Vyalla, Akhil Jarodia, Chandan Gupta, Amogh Gulati, Harsh Bandhey, Keerat Kaur Guliani, Mehrab Singh Gill, et al. Less Wrong COVID-19 Projections With Interactive Assumptions. *medRxiv*, 2020.

[64] Atsushi Niida, Takanori Hasegawa, and Satoru Miyano. Sensitivity Analysis of Agent-Based Simulation Utilizing Massively Parallel Computation and Interactive Data Visualization. *PloS one*, 14 (3):e0210678, 2019.

[65] Cynthia Nikolai and Gregory Madey. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. *Journal of Artificial Societies and Social Simulation*, 12(2):2, 2009.

[66] Michael J North, Eric Tatara, Nick T Collier, J Ozik, et al. Visual Agent-Based Model Development With Repast Simphony. Technical report, Citeseer, 2007.

[67] Michael J North, Nicholson T Collier, Jonathan Ozik, Eric R Tatara, Charles M Macal, Mark Bragen, and Pam Sydelko. Complex Adaptive Systems Modeling With Repast Simphony. *Complex adaptive systems modeling*, 1(1):1–26, 2013.

[68] Kamila Olševičová, Richard Cimler, and Tomáš Machálek. Agent-Based Model of Celtic Population Growth: Netlogo and Python. In *Advanced Methods for Computational Collective Intelligence*, pages 135–143. Springer, 2013.

[69] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, Zachariah Hughes, et al. MatLab vs. Python vs. R. *Journal of Data Science*, 15(3):355–371, 2017.

[70] Michael K Painter, Madhav Erraguntla, Gary L Hogg, and Brian Beachkofski. Using Simulation, Data Mining, and Knowledge Discovery Techniques for Optimized Aircraft Engine Fleet Management. In *Proceedings of the 2006 Winter Simulation Conference*, pages 1253–1260. IEEE, 2006.

[71] Dhirendra Pandey, Ugrasen Suman, and A Kumar Ramani. An Effective Requirement Engineering Process Model for Software Development and Requirements Management. In *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 287–291. IEEE, 2010.

[72] Jonathan Passerat-Palmbach, Mathieu Leclaire, Romain Reuillon, Zehan Wang, and Daniel Rueckert. Openmole: A Workflow Engine for Distributed Medical Image Analysis. In *International Workshop on High Performance Computing for Biomedical Image Analysis (part of MICCAI 2014)*, 2014.

[73] M Hammad Patel, Mujtaba Ahmed Abbasi, Muhammad Saeed, and Shah Jamal Alam. A Scheme to Analyze Agent-Based Social Simulations Using Exploratory Data Mining Techniques. *Complex Adaptive Systems Modeling*, 6(1):1, 2018.

[74] S Camille Peres, Tri Pham, and Ronald Phillips. Validation of the System Usability Scale (SUS) SUS in the Wild. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 192–196. SAGE Publications Sage CA: Los Angeles, CA, 2013.

[75] Primoz Podrzaj. A Brief Demonstration of Some Python Gui Libraries. In *The 8th International Conference on Informatics and Applications (ICIA2019)*, page 1, 2019.

[76] Andy Pryke, Sanaz Mostaghim, and Alireza Nazemi. Heatmap Visualization of Population Based Multi Objective Algorithms. In *International conference on evolutionary multi-criterion optimization*, pages 361–375. Springer, 2007.

[77] Marco Remondino and Gianluca Correndo. Data Mining Applied to Agent Based Simulation. In *Proceedings of the 19th European Conference on Modelling and Simulation, Riga, Latvia*, pages 1–4, 2005.

[78] Marco Remondino and Gianluca Correndo. Mabs Validation Through Repeated Execution and Data Mining Analysis. *International Journal of Simulation: Systems, Science & Technology*, 7(6): 10–21, 2006.

[79] Romain Reuillon, Mathieu Leclaire, and Sebastien Rey-Coyrehourcq. Openmole, a Workflow Engine Specifically Tailored for the Distributed Exploration of Simulation Models. *Future Generation Computer Systems*, 29(8):1981–1990, 2013.

[80] Eric Roberts. An Overview of MiniJava. *ACM SIGCSE Bulletin*, 33(1):1–5, 2001.

[81] Duncan Robertson. Agent-Based Modeling Toolkits NetLogo, RePast, and Swarm. *The Academy of Management Learning and Education*, 4:524–527, 12 2005.

[82] Jan Salecker, Marco Sciaini, Katrin M Meyer, and Kerstin Wiegand. The Nlrx R Package: A Next-Generation Framework for Reproducible Netlogo Model Analyses. *Methods in Ecology and Evolution*, 10(11):1854–1863, 2019.

[83] Andrea Saltelli. Sensitivity Analysis for Importance Assessment. *Risk analysis*, 22(3):579–590, 2002.

[84] Andrea Saltelli, Stefano Tarantola, and KP-S Chan. A Quantitative Model-Independent Method for Global Sensitivity Analysis of Model Output. *Technometrics*, 41(1):39–56, 1999.

[85] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance Based Sensitivity Analysis of Model Output. Design and Estimator for the Total Sensitivity Index. *Computer physics communications*, 181(2):259–270, 2010.

[86] N. Scott, M. Livingston, A. Hart, J. Wilson, D. Moore, and P. Dietze. SimDrink: An Agent-Based NetLogo Model of Young, Heavy Drinkers for Conducting Alcohol Policy Experiments. *J. Artif. Soc. Soc. Simul.*, 19, 2016.

[87] Alexander Shapiro. Monte Carlo Sampling Methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

[88] Ilya M Sobol', Danil Asotsky, Alexander Kreinin, and Sergei Kucherenko. Construction and Comparison of High-Dimensional Sobol Generators. *Wilmott*, 2011(56):64–79, 2011.

[89] Patrick Taillandier, Duc-An Vo, Edouard Amouroux, and Alexis Drogoul. GAMA: A Simulation Platform That Integrates Geographical Information Data, Agent-Based Modeling and Multi-Scale Control. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 242–258. Springer, 2010.

[90] Dan Tamir, Oleg V Komogortsev, and Carl J Mueller. An Effort and Time Based Measure of Usability. In *Proceedings of the 6th international workshop on Software quality*, pages 47–52, 2008.

[91] Ghazal Tashakor and Remo Suppi. Agent-Based Model for Tumour-Analysis Using Python+ Mesa. *arXiv preprint arXiv:1909.01885*, 2019.

[92] Eric Tatara, M North, T Howe, N Collier, Jerry Vos, et al. An Indroduction to Repast Simphony Modeling Using a Simple Predator-Prey Example. In *Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects*, pages 83–93. Citeseer, 2006.

[93] Guus Ten Broeke, George Van Voorn, and Arend Ligtenberg. Which Sensitivity Analysis Method Should I Use for My Agent-Based Model? *Journal of Artificial Societies and Social Simulation*, 19 (1), 2016.

[94] Jan C Thiele and Volker Grimm. Netlogo Meets R: Linking Agent-Based Models With a Toolbox for Their Analysis. *Environmental Modelling & Software*, 25(8):972–974, 2010.

[95] Seth Tisue and Uri Wilensky. Netlogo: A Simple Environment for Modeling Complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA, 2004.

[96] Thomas S Tullis and Jacqueline N Stetson. A Comparison of Questionnaires for Assessing Website Usability. In *Usability professional association conference*, volume 1, pages 1–12. Minneapolis, USA, 2004.

[97] Koen H Van Dam, Igor Nikolic, and Zofia Lukszo. *Agent-Based Modelling of Socio-Technical Systems*, volume 9. Springer Science & Business Media, 2012.

[98] Rachel Walton. Looking for Answers: A Usability Study of Online Finding Aid Navigation. *The American Archivist*, 80(1):30–52, 2017.

[99] Dirk Zimmermann and Lennart Grötzbach. A Requirement Engineering Approach to User Centered Design. In *International Conference on Human-Computer Interaction*, pages 360–369. Springer, 2007.