Netlogo vs. Julia programming: The best way to simulate opinion formation

First Author[1 [0000−1111−2222−3333]], Second Author[2,3 [1111−2222−3333−4444]], and Third Author[3 [2222−−3333−4444−5555]]

Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
{abc,lncs}@uni-heidelberg.de

Abstract Keywords: Simulation · Agent-based modells · Programming languages · Netlogo · Julia programming.

We used the following packages to create this document: `knitr` [6], `tidyverse` [4], `rmdformats` [1], `kableExtra` [**R-kableExtra**], `scales` [5], `psych` [3], `rmdtemplates` [2].

1 Abstract

2 Introduction

When humans interact with each other or with digitized technology we speak of complex systems. The interaction of humans in such systems, for example in opinion-forming processes, leads to consequences that we cannot yet overlook or understand. An important component of socio-technical complex systems are single individuals that appear as human-in-the-loop [**Valdez2018human**]. To look at the humans, their interactions and the resulting overall behavior, different simulation approaches using different programming languages are chosen.

3 Related Work

In this study, we consider whether it is possible tp create an Agent-based model with the programming language Netlogo and the programming language Julia. We further consider, how the two languages differ, which are the strenghths for creating Agent-based models of each programming language and which are the X. Contentwise, we built a bounded rationality model to simulate opinion formation.

Therefore, we introduce

explain what is

what influences how messages are spread in online social networks using an agent-based simulation. Therefore, we explain what is known in theory about the spread of information and the formation of opinions in this section. We further

introduce the latent process model, on which we built our simulation and explain further aspects that are important for the agent-based simulation.

Using two different programming languages (Julia language and Netlogo), we created two identical agent-based models that simulate opinion formation. Since our primary aim was to find out whether agent-based models could be implemented equally well in the two programming languages, we chose the most basic model of opinion-forming: bounded rationality.

3.1 Complexity

When examining opinion-forming processes, we look at a complex system. Such complex systems lead to emergent phenomena. Complex systems and emergent phenomena are difficult to understand, because while it is easy to observe the individual system components, the resulting overall system cannot be considered as the sum of its parts. Instead, understanding the system behavior requires more than understanding the individual parts of the system [**Valdez2018human**].

To analyse complex systems we need a suitable approach, such as simulations, which enable to model the individual parts of a system and thus make the overall behavior visible. For the simulation of complex systems, Agent-based models are very well suited [**Epstein2007**]. Agent-based models always consist of the agents or individuals and the environment in which the agents reside [**Bonabeau2002Agentbased**].

To create agent-based models, Netlogo [**Wilensky1999**] is the language most commonly used. Nevertheless, there are some other programming languages that are also suitable for creating agent-based models and that seem to be partly more intuitive, at least for people with programming experience. Therefore, in this study we compare two programming languages with respect to their suitability for creating agent-based models.

3.2 Complex systems/Complexity

4 Method

Using two different programming languages (Julia language and Netlogo), we created two identical agent-based models that simulate opinion formation. Since our primary aim was to find out whether agent-based models could be implemented equally well in the two programming languages, we chose the most basic model of opinion-forming: bounded rationality.

In our bounded rationality model, we defined the maximum number of agents, the maximum steps of the simulation, the seed and an epsilon in the beginning. The epsilon indicates how different the opinions of two people can be, so that they still include the other person's opinion in their opinion making. We further defined from the beginning, that each agent has an (floating) opinion between 0 and 1. In each simulation step, every agent compares his opinion with the opinion of an other agent. For example, if Anna compares her opinion with Ralf and

the distance between the opinion of Anna and Ralf is smaller than the defined epsilon, then the two converge in their opinions.

We build the agent based models using either the Julia programming language or Netlogo. For the following analysis of the results we used R Markdown.

4.1 What do we compare

To find out whether both programming languages are euqually suitable to simulate our bounded rationality model, we look at severeal measurable criteria. These criteria include the outcomes and performance of both models. They further include how many lines of code are necessary to program the simulation. Another aspect, that we take into consideration, is, if learning Julia and Netlogo is equally difficult. For this aspect we consider both computer scientists who are familiar with other programming languages and a person who has no previous experience with programming languages. We further compare the explorability and scalability of both languages.

5 Results

Following, we first present the outcome of the bounded rationality model created with julia. Afterwards we show, if the model created with Netlogo showed the same or different results. Based on these results, we compare the two considered programming languages and show their advantages and disadvantages.

5.1 Julia simulation

5.2 Comparison of Julia and Netlogo

When comparing both programming languages to create an agent-based model that simulates the bounded rationality model, Julia proved to be a faster language less code lines were required to simulare our model. In addition, Julia turned out to be easier to learn for people with previous programming skills. In contrast, Netlogo proved to be a language that is easier to learn for people without programming skills.

6 Discussion

In our study, no language turned out to be the perfect programming language for creating agent-based models, but the choice of language seems to be a trade-off between various advantages and disadvantages. Of course, we have only focused on one very simple bounded rationality model, so that we would have to create further simulations with both languages to be able to make statements about the generality.

7 Conclusion and outlook

The results of our research have shown that Julia is the better choice for creating an agent-based model in some aspects, but also that Netlogo is better suited in some aspects. In the future, we plan to create more agent-based models using both programming languages, thus extending our comparison.

References

[1] Julien Barnier. *rmdformats: HTML Output Formats and Templates for 'rmarkdown' Documents*. R package version 0.3.6. 2019. URL: https://CRAN.R-project.org/package=rmdformats.

[2] André Calero Valdez. *rmdtemplates: RMD Templates*. R package version 0.1.0.0. 2019.

[3] William Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. R package version 1.8.12. 2019. URL: https://CRAN.R-project.org/package=psych.

[4] Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.3.0. 2019. URL: https://CRAN.R-project.org/package=tidyverse.

[5] Hadley Wickham and Dana Seidel. *scales: Scale Functions for Visualization*. R package version 1.1.0. 2019. URL: https://CRAN.R-project.org/package=scales.

[6] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.26. 2019. URL: https://CRAN.R-project.org/package=knitr.