Agent- and Individual-based Modeling with NetLogo: Introduction and new NetLogo Extensions

Authors: Jan C. Thiele, Winfried Kurth, Volker Grimm

Published in:
Die Grüne Reihe 2011
22. Tagung, Göttingen, 20.-21. Sep. 2010
Editors: Klaus Römisch, Arne Nothdurft, Uwe Wunn
ISSN 1860-4064

Please cite as:
Thiele, J.C., Kurth, W. & Grimm, V. (2011) Agent- and Individual-based Modelling with NetLogo: Introduction and New NetLogo Extensions. In: K. Römisch, A. Nothdurft & U. Wunn (eds.): Die Grüne Reihe 22. Tagung der Sektion Forstliche Biometrie und Informatik des Deutschen Verbandes Forstlicher Forschungsanstalten und der Arbeitsgemeinschaft Ökologie und Umwelt der Internationalen Biometrischen Gesellschaft - Deutsche Region, 20-21th September 2010 in Göttingen (Germany), pages 68-101, ISSN 1860-4064.

Agent- and Individual-based Modeling with NetLogo: Introduction and new NetLogo Extensions

Jan C. Thiele[1,2], Winfried Kurth[1] & Volker Grimm[2]

[1]Dep. of Ecoinformatics, Biometrics and Forest Growth, Georg-August University Göttingen
[2]Dep. of Ecological Modelling, Helmholtz Centre for Environmental Research - UFZ, Leipzig

Zusammenfassung

Die agentenbasierte Modellierung (ABM) oder individuenbasierte Modellierung (IBM), wie sie in der Ökologie und Biologie genannt wird, ist inzwischen ein weitverbreiteter Modellierungsansatz, wenn lokale Interaktionen auf Mikrolevel entscheidend für die Beschreibung von Mustern auf dem Makrolevel sind. Der vorliegende Beitrag gliedert sich in vier Abschnitte. Im ersten Teil wird ein Überblick über die Entwicklung und Definition von ABM in den Forschungsgebieten Informatik, Soziologie, Ökonomie und Ökologie gegeben. Dieser Abschnitt schließt mit einer Diskussion der Gemeinsamkeiten und Unterschiede von agentenbasierter Modellierung in den unterschiedlichen Forschungsgebieten und einer Darstellung der gegenwärtigen Probleme und Herausforderungen im Bereich der agentenbasierten Modellierung. Eine dieser Schwierigkeiten ist das Fehlen anerkannter Standards im Bereich der Kommunikation und Programmierung agentenbasierter Modelle. Der zweite Abschnitt nimmt diesen Punkt mit einer Vorstellung verschiedener, weitverbreiteter Programmbibliotheken (Swarm, Mason, Repast, NetLogo) auf und geht in den dritten Abschnitt über. Dort wird NetLogo als eine mögliche Standardsprache bzw. ein Standardwerkzeug für die Kommunikation von ABM vertieft dargestellt. Im letzten Abschnitt werden Erweiterungen für NetLogo, entwickelt von den Autoren dieses Artikels, vorgestellt. Dies beinhaltet die MultiView-Extension, die R-Extension und ein NetLogo-Plug-In für den Pygments Syntax Highlighter. Der Beitrag schließt mit einem Ausblick auf weitere Werkzeuge und Erweiterungen für NetLogo ab, die NetLogos Relevanz als Kandidat für ein(e) Standardsprache/-werkzeug in der agentenbasierten Modellierung noch ausbauen werden.

Summary

Agent-based models (ABM) or individual-based models (IBM), as they are called in ecology and biology, are a widely used modeling approach when local interactions on the micro level are essential for the description of patterns on the macro level. This chapter is divided into four sections. In the first section, the history and definitions of ABMs in various research disciplines, namely computer science, social science, economics and ecology, are reviewed. This section closes with a discussion of similarities and differences in the different research fields and a discussion of current challenges in agent-based modeling. One of these difficulties is the lack of accepted standards for communication and programming. The second section refers to this point by a presentation of some widely used ABM libraries, namely Swarm, Mason, Repast and NetLogo and is followed by a more detailed description of NetLogo as a potential standard tool in ABM communication. In the last section extensions to NetLogo, developed by the authors of this chapter, are presented. This includes the MultiView-Extension, the R-Extension and the NetLogo plug-In for the Pygments syntax highlighter. The chapter closes with an outlook to further tools for NetLogo which aim at making NetLogo even more relevant as a standard tool in ABM.

I. Agent-/individual-based modeling

Introduction

Building and using models is part of everybody's life. For example, if we wait for the train, we decide where to stay so that we can get into the train fast and get a good seat. We include our former experience about where it is best to find a seat: in the middle, the front or the back of the train. Furthermore we check the other passengers waiting on the track. People carrying heavy luggage are slower than others and so on. But it will be impossible to include all information and details. Therefore, simplification and aggregation of the real system (abstraction) is the key in modeling. STARFIELD et al. (1990) called this "purposeful representation", which means that the real system is represented only by those elements which are important for answering the question the model is designed for. Thus, the problem to solve should stand at the beginning of the model building process. The second step is the definition of the system and its boundaries which leads to a verbal, conceptual model (BOSSEL 1994) followed by the selection of the formal model structure with scales, state variables, processes and parameters. If such a model is too complex to calculate its outcome analytically, it has to be implemented as a computer model, called computer simulation model, before the model can be analyzed and developed (see Modeling Cycle in GRIMM & RAILSBACK 2005).

There are different reasons to perform a computer simulation. They are useful to test theories and to try understanding system behavior, but they are also used to make forecasts and to design experiments which could not be done in real life (e.g. for ethical or technical reasons).

With increasing computer power and growing criticism of conventional modeling methods, a new approach to simulation models, agent-based modeling (ABM), established in different kinds of scientific fields. Since the development of ABM took place more or less independently in different scientific fields, they differ in definitions, history and context. Therefore, we will give a short overview over the most important sectors: computer science, sociology, economics and ecology. But it should be mentioned that ABM is now increasingly applied interdisciplinarily, which makes the following separate descriptions slightly obsolete.

ABM in Computer Science

The basis for agent-based modeling in many other fields has been the development of so-called **multi-agent systems** (MAS) in **computer science** as a part of the **distributed artificial intelligence** (DAI) research area (GREEN et al. 1997, SYCARA 1998). The development of MAS started around the late 1970s (e.g. HEWITT 1976) but their wider use began only in the 1990s, mainly to distribute large computational problems over multiple computers (LUCK et al. 2003, WOOLDRIDGE 2005, WEISS 1999). Later influenced and adapted by different disciplines, it is not surprising that many different definitions of the term MAS and what an agent is exist (e.g. JENNINGS et al. 1998, JENNINGS 1999). In a very general case a MAS is defined as a system which is composed of multiple (semi-) autonomous components to reach a common goal

(JENNINGS et al. 1998). Many modular software systems could be summarized under this weak definition. A narrower definition is given by WOOLDRIDGE & JENNINGS (1995). They added the requirement of using intelligent agents to their definition of MAS. Their (computational) agents have been characterized as autonomous, social, reactive and pro-active entities. In this concept, an agent takes input from the environment and produces actions as outputs that affect the environment but with incomplete information at the agent level and without a global control mechanism (WOOLDRIDGE 2002). Some classical examples for the use of MAS with intelligent agents can be found in computer games, computer networks, robotics for manufacturing, or traffic-control systems (for examples, see OLIVEIRA 1999, LUCK et al. 2003, SHEN et al. 2006, MOONEN 2009). But the more autonomous the local agents became, the more important the question about coordination and cooperation within the system was. These questions had strong similarities to the research questions in sociology (CONTE et al. 1998).

ABM in Social Sciences

Since some researches from the DAI area adapted ideas from **social sciences,** sociologists became aware of the MAS techniques in the 1990s, encouraged by the advent of personal computers and the development of object-oriented programming languages (EPSTEIN & AXTELL 1996, GILBERT 1999, SQUAZZONI 2010). As a part of computational sociology, the approach is called **agent-based social simulation** (ABSS) or just **agent-based modeling** (ABM) (CONTE et al. 1998, GILBERT 2008, MACY & WILLER 2002). In such ABMs, agents typically represent social actors which can be individual persons, organizations (e.g. companies), or countries (GILBERT 2008) which are simulated as an artificial society (EPSTEIN & AXTELL 1996). GILBERT (2008) defined an ABM as a "computational method that enables a researcher to create, analyze and experiment with models composed of agents that interact within an environment", which is very similar to the definition given by WOOLDRIDGE (2005) for MAS.

The ABM approach differs substantially from former microsimulation and system dynamics approaches in social computations. The latter typically consists of sets of differential equations which describe the change of stock variables and is known as "top-down" approach. In this way, individual social actors are averaged and their behavior is aggregated. A famous example for that kind of models is the world model of FORRESTER (1971), which was used by MEADOWS (1974) for predicting the ecological limitation of economic and demographic growth. Such models have been mainly used for quantitative forecasts, are often highly sensitive to the (frequently unknown) model parameters, and ignore the heterogeneity of individuals. Therefore, they are rarely appropriate for social sciences, which are mainly interested in understanding and explanation (GILBERT & TROITZSCH 1999).

The classical micosimulation approach incorporates individual heterogeneity but not interactions between individuals (MACY & WILLER 2002). This approach is based on transition probabilities for each individual, which are typically derived from empirical data, and delivers quantitative forecasts (GILBERT & TROITZSCH 1999). Such data-driven microsimulations have been applied mainly to forecast the effects of policy changes and are highly related to macroeconomic analysis but also well established in non-

social sciences like traffic/transportation analysis and economic research. Example applications in social sciences include tax-benefit analysis, analysis of demographic developments, social security and labour analysis. Reviews can be found in SPIELAUER (2007), ZAIDI & RAKE (2001) or ALGERS et al. (1997).

With the introduction of ABM, interactions between complex, adaptive individuals (MACY & WILLER 2002) and the opportunity to take spatial effects into account (GILBERT 2008) were added to social computer simulations. The main idea is to analyze the way in which macro-level system properties emerge from interaction of the agents on micro-level to describe social systems (DAVIDSSON 2002). Therefore, it is mainly used for theory testing and development (MACY & WILLER 2002, CONTE 2009). Furthermore, the development of an ABM constrains the social scientist to clearly define his assumptions of local behavior which helps to clearly communicate basic ideas (GILBERT 2004).

In general, ABMs are a good choice for studying social processes that do not include central coordination (MACY & WILLER 2002). ABMs in social science have been applied to the simulation of differentiation, diffusion and emergence of social order in social systems (for examples see listing in MACY & WILLER 2002 and references in LI et al. 2008 and SQUAZZONI 2010) as well as to questions about demographic behavior (BILLARI & PRSKAWETZ 2003). Most famous models in social sciences are SCHELLING'S (1969, 1971) segregation model and the Sugarscape model of EPSTEIN & AXTELL (1996), which have often been cited and extended. However, as stated by MACY & WILLER (2002) and recently by HAMILL (2010), ABMs are still considered very skeptically by many sociologists and have been used rather rarely compared to other disciplines, although SQUAZZONI (2010) reported about a constant increase of ABM awareness in recent years. This could be the result of the increasing use of ABM techniques in interdisciplinary projects with socio-ecological and socio-economical contexts.

ABM in Economics

The neoclassical Walrasian general equilibrium (GE) model as well as the (New) Keynesian framework are still the most used fundamental paradigms in (Macro-) **economics** (TESFATSION 2006, OEFFNER 2008). They simplify the economic system by representing (averaged) agents with perfect rationality, information and foresight. Furthermore, interactions between agents (e.g. firms and households) take place only indirect, in case of Walrasian GE model, for example, by pricing using the concept of "Walrasian auctioneer" (OEFFNER 2008).

Such top-down approaches were developed despite ADAM SMITH (1776) and others stated that economic processes are the result of parallel, local interactions between large numbers of individuals. Therefore, GUN (2004) criticized macroeconomic approaches as he wrote: "How can any reasonable person admit, that, for example, the evolution of the US aggregates' results from decisions made by a single individual who owns all factories and who decides how much to produce, how much labor to use, how production will be distributed between consumption and investment, and so on?". Consequently, to overcome these weaknesses the ABM approach was adapted to economic modeling by using heterogeneous, interacting (and learning) agents (FARMER & FOLEY 2009).

The ABM approach in economics is called **agent-based computational economics** (ACE) and is related to the field of cognitive and evolutionary economics. ARTHUR (2006) described this as follows: "Standard neoclassical economics asks what agent's actions, strategies, or expectations are in equilibrium with (consistent with) the outcome or pattern these behaviors aggregatively create. Agent-based computational economics enables us to ask a wider question: how agent's actions, strategies or expectations might react to – might endogenously change with – the pattern they create. In other words, it enables us to examine how the economy behaves out of equilibrium, when it is not at a steady state." The beginnings of ACE are going back to the work of the Santa Fe Institute which started in the late 1980s with its work on describing and analyzing the economy as an evolving and complex system encouraged by the development of personal computers and object- and agent-oriented programming languages (RICHIARDI 2007).

TESFATSION (2006) has defined ACE as "the computational study of economic processes modeled as dynamic systems of interacting agents" and an agent as "bundled data and behavior methods" which could represent individuals, social grouping and institutions, biological or physical entities. This definition is nearly identical to the definition of social ABMs given by GILBERT (2008), which is not surprising since there is no strict boundary between both disciplines. Even the aims of ACE have similarities to those of social ABMs. They can be divided into four main categories: empirical understanding, normative understanding, qualitative insight as well as theory generation and methodological advancement (for details see TESFATSION 2006). Therefore, ACE can be used complementarily to mathematical theorizing as well as a substitute for it (PHAN 2004, AXTELL 2000). ACE have been used for example for the reproduction of classical cobweb theorem (e.g. ARIFOVIC 1994), for modeling financial/stock markets (see LEBARON 2000 for a review) as well as for simulating industry and labor dynamics (e.g. LEOMBRUNI & RICHIARDI 2004). Nevertheless, many economists still prefer the conventional mathematical models by tradition (BUCHANAN 2009).

ABM in Ecology

In contrast, in the field of **ecology** the agent-based approach has a slightly longer tradition and is well established nowadays (GRIMM & RAILSBACK 2005), although there are critics as well (CASHWELL 2001). The development of so called **individual-based models** (IBM) is less closely related to the developments of MAS as it is the case in social sciences, because ecologists early became aware of the restrictions in classical population models (differential equation models) and looked for alternatives. The most obvious approach for the simulative reproduction of observed population-level effects was to take the heterogeneous properties of the individuals into account.

Since sufficient computational power became available in the early 1970s, first ecologists started to develop such IBMs (e.g. MYERS 1976, DEANGELIS et al. 1980). One of these pioneer works is the forest succession model JABOWA of BOTKIN et al. (1972). In his review, however, GRIMM (1999) distinguishes this work, which often was pragmatically motivated, from the later paradigmatic motivations of the IBM approach,

in which IBMs are not used because of the limitation of more aggregate models but because of the motivation to overcome the limitations of the population-level paradigm of theoretical ecology. GRIMM (1999) refers to KAISER (1974) and ŁOMNICKI (1978, 1988) as the pioneers of the paradigmatic motivation, but noticed that still only a minority of ecological IBMs directly addressed theoretical issues. The main driver of the success of IBMs in ecology was and still is the pragmatic motivation.

Milestones in the establishment of the IBM approach in ecology were the review of HUSTON et al. (1988), followed by the substantial conference proceedings of DEANGELIS & GROSS (1992) and the influential articles of HOGEWEG & HESPER (1990), JUDSON (1994), UCHMANSKI & GRIMM (1996), GRIMM (1999), ŁOMNICKI (1999), DEANGELIS & MOOIJ (2005) as well as the monograph of GRIMM & RAILSBACK (2005). As GRIMM & RAILSBACK (2005) noticed, up to mid-1990s, a clear definition of IBMs was still missing and the use of the term became fuzzy.

Therefore, UCHMANSKI & GRIMM (1996) defined criteria for the classification of ecological models which allow separating IBMs from classical state-variable approaches (for details see UCHMASKI & GRIMM 1996 or GRIMM & RAILSBACK 2005): IBMs include (potentially) heterogeneous, discrete entities which represent real individuals. These individuals can be adaptive in behavior and life history. Moreover, IBMs include feedbacks between dynamic (food) resources and individuals. In contrast, age- or state-structured population models and models defining resources via constant carrying capacities do not fulfill these conditions. GRIMM & RAILSBACK (2005) defined such models in the middle between unstructured population models and IBMs.

Over the last three to four decades hundreds of IBMs were developed in ecology (DEANGELIS & MOOIJ 2005). For reviews see for example GRIMM (1999), GRIMM & RAILSBACK (2005), HOGEWEG & HESPER (1990), DEANGELIS et al. (1990), DEANGELIS & GROSS (1992), DEANGELIS et al. (1994) and DEANGELIS & MOOIJ (2005). See also BUNSING & MAILLY (2004) and LIU & ASHTON (1995) on IBMs of forest dynamics. In case of modeling forests, these IBMs are sometimes called **individual-tree models** (e.g. by PRETZSCH 2009 and COATES et al. 2003).

A family of models in biology which has some common features with IBMs is that of functional-structural plant models (FSPM) (GODIN & SINOQUET 2005, VOS et al. 2007). In an FSPM, a plant individual is decomposed into morphological units like internodes, leaves, root segments etc., which all have their own functions and state variables and which interact with each other. The three-dimensional architecture of a plant and its functioning are both represented in the same model. Since the morphological units are modeled as entities with certain autonomy, this approach can be considered as an extension of IBMs to a spatial scale level below that of the individual. However, FSPMs can also be used to model interactions between several individuals and thus the behavior of whole plant stands (HEMMERLING et al. 2008, COURNÈDE et al. 2010).

Synopsis

As we have seen, there are strong similarities in the definitions of ABMs in the different fields but there are also some differences. The narrowest definition gave UCHMANSKI & GRIMM (1996) in the field of ecology while no strict line can be drawn between ACE and

ABSS. In the last two fields, however, the influence of MAS from computer sciences was much stronger than in ecology. Whilst ABMs in social science and economy are often more paradigmatic and abstract, they are usually pragmatic and specific in ecology. Nevertheless, they have in common that they present an alternative to aggregated state-variable (equilibrium) models; they do not impose system-level results but are dedicated to answer the question how system-level patterns emerge from adaptive behavior and local interactions.

Although we here focussed on the research fields computer science, sociology, economy and ecology, there are many other disciplines in which ABMs are increasingly used, often within an interdisciplinary context. Examples are **ecological economics** (e.g. HECKBERT et al. 2010 and DRECHSLER et al. 2007), **marketing/socio-psychology** (e.g. NORTH et al. 2010 and BEN SAID et al. 2002), **archaeology/anthropology** (e.g. GRIFFIN & STANISH 2007, PREMO 2007 and PREMO & KUHN 2010), **microbiology** (e.g. FERRER et al. 2008 and GINOVART et al. 2002), **biomedicine/epidemiology** (e.g. AN 2009 and CARPENTER & SATTENSPIEL 2009), **criminology** (strongly related to ABSS, e.g. MALLESON et al. 2010), **land-use management** (e.g. POLHILL 2009 and MATTHEWS et al. 2007) and **forest management planning** (e.g. SIMON & ETIENNE 2009).

The wide use of the ABM approach in all these disciplines has in common that it was strongly related to the increasing availability and power of personal computers and the development of object- and agent-oriented programming languages.

Current Challenges in ABMs

As with every method there is no light without shadow. There are several challenges as well as stereotypes related to the ABM approach. Some of them are shortly discussed in the following.

ABMs are often seen as very **data hungry** (e.g. REED et al. 2002). It is true that data are needed when building ABMs, but it is wrong to believe that an ABM should not be built without complete information (STARFIELD 1997). As GRIMM & RAILSBACK (2005) and SQUAZZONI (2010) noticed, ABMs can be used to test hypothesis about unknown parts of a system and can help empiricists to identify which are the important things to measure. Furthermore, it can be easier to collect data at the level of the individual than to collect them at an (abstract) macro-level (HOGEWEG & HESPER 1990, HUSTON et al. 1988), and the model might react less sensibly on parameter variations at the level of the individual (BRECKLING 2002). The main problem with the measurements is that results of laboratory experiments with sometimes isolated individuals/humans could be invalid for natural conditions. In general, GRIMM (1999) reported about a tendency towards an overuse of empirical knowledge in IBMs in the sense of putting all available data into a model. He advised to find the appropriate level of aggregation through the modeling process and not through availability of data or the aim to let the model (not the outcome) look more ‘realistic’.

This leads directly to the next point, the **difficulties in analyzing, validating and understanding** the outcomes of an ABM due to its complexity. ABMs should be kept as simple as possible otherwise they could contain unnecessary details to answer the research question (MACY & WILLER 2002, HIEBELER 1994). Nevertheless, even in simple

ABMs outcomes can be hard to understand. Different guidelines about how to analyze ABMs are available (e.g. RICHIARDI et al. 2006, GILBERT 2008, WINDRUM et al. 2007, GALÁN et al. 2009); GRIMM & RAILSBACK (2005) dedicated a whole chapter to this topic. But an accepted standard way of analyzing ABMs is still missing.

The **communication** of ABMs is also very difficult, since ABMs are simulation models and cannot be described completely by means of mathematics. Therefore, several communication protocols have been proposed. Some are based on an extension of the Unified Modeling Language (UML) but such approaches are usually too technical for non-computer scientists. The ODD protocol of GRIMM et al. (2006) is much simpler and allows providing a precise overview over the general properties of the model. It has the potential to become a standard protocol for ABMs (GRIMM et al. 2010, POLHILL 2010, JANSSEN et al. 2008).

However, since the ODD protocol only provides a structure for verbal model descriptions but does not specify how to document the details of the model it will be just one half of an overall model documentation. It does not yet provide a "lingua franca" by itself, which via some kind of pseudo code would allow describing all submodels unambiguously. In the next two sections we will discuss how the software platform NetLogo (WILENSKY 1999) might fill this gap.

The last point to be discussed here is the high risk of programming "**bugs**" in ABM implementations (LOREK & SONNENSCHEIN 1999, GILBERT 2008). A bug means that the computer is doing something different to what the programmer/modeler intended. Such bugs can lead to misinterpretations of the model outcomes (AXELROD 1997).

Simulation platforms for agent-based models can help reducing the risk of programming bugs and can increase the understandability of the source code. Furthermore, such platforms will make the implementation process much easier (HAMILL 2010). HEATH et al. (2009) surveyed 279 ABM articles of which just 175 offered details about their programming language/tool. He identified 68 different tools/languages. HEATH et al. (2009) conclude that there will never be one standard programming language.

We think, however, that the fact that modelers usually are not computer scientists and the challenges of ABMs described so far should encourage us to try and find a common language and standard procedures or template (sub) models. There is actually no point in implementing an ABM from scratch, using a general purpose programming language like Java or C++ because ABM libraries exist which provide standard software designs and code. Standardized model descriptions and implementations would facilitate to understand the underlying concepts, structure and algorithms of ABMs. Without such standards, it is difficult or even impossible to reproduce the results of published models, which undermines the scientific credibility of the ABM approach (JANSSEN et al. 2008).

Therefore, the next part of this chapter focuses on software platforms and languages for ABMs and is followed by a more detailed description of the NetLogo language and platform.

II. Software libraries, environments and languages for ABMs

As discussed above, to decrease implementation time and the risk of making errors and to increase re-usability, traceability and communicability, it is helpful to use established software libraries or languages that were especially designed for implementing ABMs. One of the first programming languages which provided features for object-oriented modeling and simulation was SIMULA (DAHL et al. 1967). It was indeed used for ABMs in ecology (e.g., Reuter 1998), but required still considerable skills in mastering a general-purpose language, and its supportive features for ABMs were rather limited. Since that time, many general and special purpose libraries and platforms for the development of ABMs have been developed. WIKIPEDIA (2010) lists 69 different agent-based modeling software libraries/platforms and several reviews can be found in the literature (e.g. ALLAN 2009, NIKOLAI & MADEY 2009, BERRYMAN 2008, GILBERT 2008, RAILSBACK et al. 2006, TOBIAS & HOFMANN 2004). The most popular, non-proprietary general-purpose ones are Swarm (MINAR et al. 1996), MASON (LUKE et al. 2005), Repast (COLLIER et al. 2003) and NetLogo (WILENSKY 1999), which are briefly presented in the following. Besides we would like to mention that any review of ABM software platforms is due to be outdated within a year or two because some platforms develop rapidly while others are no longer maintained. Therefore, in the following we do not list software features in detail but describe the history and the main concepts underlying the platforms.

Swarm can be seen as the mother of many other ABM libraries. The project was initiated in 1994 at the Santa Fe Institute, New Mexico (HIEBELER 1994) and is maintained since 1999 by the Swarm Development Group (SWARM 2010a). The aim of Swarm is to provide a vocabulary and a set of standard computer tools for the development of multi-agent simulation models (SWARM 2010b). It comes as an open-source library collection for Objective-C and has a port for using it in Java, which both follow the object-oriented programming (OOP) paradigm. Applications using Swarm often contain a hierarchical structure, with an *observerSwarm* object responsible for the creation of screen displays and a *modelSwarm* object, which manages the individual agents and schedules their activities in discrete time intervals as well as delivers information to the observer (SWARM 2010b). Agents are instances of user-written classes derived/extended from the class *SwarmObject*. Furthermore, the Swarm libraries provide classes for the creation of Graphical User Interfaces (GUI) for controlling simulations and for visualization (SWARM 2010b). Since Swarm consists just of libraries, the models are written in the language of the chosen library (Objective-C or Java). This requires strong programming skills but gives a maximum of flexibility and extendibility to the (experienced) user.

MASON is considerably younger as the project was initiated in 2003 at the George Mason University, USA (BALAN et al. 2003). It is written as an open-source library in Java and is conceptually strongly inspired by Swarm. The main development goal was a good computational performance to allow simulations with many runs, a large number of agents and a good support for 3D-simulations and visualizations (LUKE et al. 2004). It is, like Swarm, a pure but very concise library. The implementation of a simulation model in MASON is encapsulated completely from the code for visualization. The highest level of a simulation model is the *SimState* super-class which holds an instance

of the *Schedule* class for managing the time and sub-schedules in the model. Agents can be added to the simulation/schedule by instancing user-written classes, which have to implement the interface *Steppable* with the agent schedule method *step*. Space can be represented by using the grid-based or continuous space classes whereas networks are created by using the classes *Network* and *Edge* (MASON 2010). The same as for Swarm holds about flexibility and required programming skills.

Repast (Recursive Porous Agent Simulation Toolkit) is slightly older than MASON as the first release was available in 2000 offered by a group of researches of the University of Chicago (COLLIER et al. 2003). Started as a pure Java library, it was available for the programming languages Java (Repast J), Microsoft .Net (Repast .Net) and Python (Repast Py) in the meantime since Version 3.0, which was released in 2004 (ROAD 2010a). In 2008 the Repast development team, more precisely the Repast Organization for Architecture and Development (ROAD), introduced **Repast Simphony** (called Repast S), a customized version of the integrated development environment (IDE) Eclipse, which enables the user to build models using graphical editing tools via drag and drop. At the end of 2010, a beta version of Repast S 2.0 was released and opened different ways to build models: the point-and-click flowcharts, Java, Groovy and **ReLogo** as a Logo dialect and an import and translation routine for NetLogo models (ROAD 2010b). In December 2010 as well, a first beta version of **Repast High Performance Computing** (HCP) was published, designed to run simulations in parallel on large-scale distributed computer platforms such as clusters and supercomputers. Models for Repast HCP can be implemented in standard or Logo-style C++ (ROAD 2010c). Although the Repast toolkit has its origins in social sciences, it is usable for the implementation of most agent-based models. The basis of a classical Repast model is a *Context* which is built by the *ContextBuilder* class. A *Context* manages a set of *Agents*, like a population. *Agents* are user-defined classes for each type of *Agents* (e.g. one for wolves and another for sheeps) which implements behaviors by methods and state variables by class members. If spatial aspects are required, a *Projection* can be added to a *Context*, which places the *Agents* into a *Grid* or *ContinousSpace*. To create relations between *Agents*, the *NetworkBuilder* class is used (COLLIER & NORTH 2010). The ReLogo part is conceptually equivalent to that of NetLogo, what is not surprising, since ReLogo is adapted from/inspired by NetLogo. Therefore, details about the ReLogo concept can be left out here as NetLogo is described in detail later. The Java/Groovy and ReLogo part of Repast deliver an extendable standard window for simulation visualization, a way for creating stand-alone programs and an option to run them in batch mode as well as a couple of interfaces to helpful programs like GNU R, GRASS GIS or MATLAB.

By now, the presented modeling libraries are conceptually very similar, except for ReLogo. Models are written in a common high level programming language by implementing interfaces or extending classes of the used libraries. There is always a top-level class coordinating the simulation and bundling the simulation entities. The class or interface which are extended or implemented by agent classes are unspecific, i.e. do not differentiate between different types of agents. Space is added explicitly to

agent classes. Although some helping methods for implementing the behavioral rules for the agents exist, a lot of code has to be written by hand.
Because **NetLogo** is not a library but a complete language especially designed for the implementation of ABM, it provides many pre-defined methods for the implementation of the behavioral rules of the agents. Furthermore, its Logo-like syntax (see next section) and different standard agent types (Turtles, Patches, Links) in combination with a point-and-click GUI-building mechanism make it possible to learn the language very fast without programming experiences. No less important are the excellent and comprehensive documentation of the NetLogo language and simulation environment and the different tutorials. Moreover, there is a very large library of sample models and code examples as well as a very active mailing list. Furthermore, textbooks with practical introductions to agent- and individual-based modeling using NetLogo are available (e.g. RAILSBACK & GRIMM in press and WILENSKY & RAND in press).
Because of these features, NetLogo is very popular and widely used. It has the potential to become a standard language for describing ABMs (as pseudo language) and is ideal for prototyping since in no other language ABMs can be written and changed as fast as in NetLogo. This hypothesis is underlined by the ReLogo project, which adapted the NetLogo language as described above and opens the possibility to translate NetLogo models to Repast, which allows to run also models which are computational very expansive. Therefore, in the following NetLogo is in focus as it has the potential to help to overcome the weakness of missing standards in ABMs in respect to the description of model details.

III. NetLogo: Modeling language and simulation platform

History of NetLogo

The development of NetLogo started in 1999 (WILENSKY 1999) and the first beta version was released in 2000 by Uri Wilensky (TISUE & WILENSKY 2004a). It is now maintained by the Center for Connected Learning and Computer-based Modeling at the Northwestern University, Illinois (NETLOGO 2010a). NetLogo is the successor of the StarLogo simulation environment family. Although StarLogo was mainly developed for the use in education it was used more and more by researches. This led to the development of NetLogo, which was designed for educational and research purposes (TISUE & WILENSKY 2004b). Milestones in the development of NetLogo were

- the release of the first stable version 1.0 in 2002,
- the introduction of the HubNet functionality in 2003,
- the extension and controlling API as well as a headless mode for GUI-free command line runs in 2004,
- the System Dynamics Modeler for models including ordinary differential equations and a 3D view in 2005,
- an improved compiler that speeded up many models and the introduction of links as own agent type for network models in 2007,
- a GIS extension published in 2008 and

- BehaviourSpace, a tool for running NetLogo repeated with different parameter values, became open source and supports running multiple simulations with the BehaviourSpace in parallels in 2009 (NETLOGO 2010b).

The current version, released in December 2010, is 4.1.2.

A drawback of NetLogo is that although it is available free of charge, its source code is not available (with exception for BehaviourSpace and the extensions). This is often criticized in reviews as it means that the proper functioning of built-in NetLogo procedures, called “primtives”, can only be validated by testing and not by inspecting and directly testing the underlying source. On the other hand, this is more a theoretical than a practical discussion: NetLogo includes a compiler (SONDAHL et al. 2006) whose implementation is presumably beyond the programming skills of most modelers, who are often not computer scientists. Nevertheless, it is not good practice to make use of open source libraries, like the MersenneTwisterFast from the Repast project or the MovieEncoder from the MASON project, without opening the own project (NETLOGO 2010c). But on the FAQ page of the NetLogo project the NetLogo team wrote, that “We are working on eventually releasing the source under an open source license.” (NETLOGO 2010d).

NetLogo Programming Language

The NetLogo language is especially designed for the implementation of ABMs. As the name suggests, it is an extension of the Logo language. Logo was designed as a functional programming language for educational use in the 1960s and is a dialect of Lisp (LOGO FOUNDATION 2010). Logo is often known for its turtle, which is used to create graphics on the monitor by giving movement commands to the turtle. NetLogo adapted this turtle approach and combined it with the concept of multiple turtles/agents and concurrency from *Lisp (“StarLisp”). The design goal of the Logo language was "low threshold and no ceiling", which was borrowed for the development of NetLogo (TISUE & WILENSKY 2004a). Low threshold stands for easy to learn also by people with less or without modeling and programming experiences. No ceiling means that it should not be restrictive for advanced users, who need high flexibility.

The basic entity in NetLogo is the agent. There are different pre-defined types of agents: the observer, patches, turtles, and links. The **observer** is the global instance which provides global variables and manages and has access to the other agents. There is only one observer. **Patches** are immobile agents, i.e. spatial units with a location in space. All patches together form the grid of the world and define the extent of the world. Patches have pre-defined variables, such as x- and y-coordinates, a color and a label, but the user can add further variables to the patches. Patches are, like all other agents, programmable. **Turtles** are, contrary to the patches, mobile agents. They can move on the patches in continuous space within the world defined by the patches. Turtles have, like patches, pre-defined variables, like their position, shape etc. but can get user-defined variables. It is possible to declare different types of turtles, called breeds; breeds inherit all variables of the turtles, but can have additional own variables. The last agent type is the **link**, which is a connection between two turtles. All agents can communicate and interact with each other.

The available data types in NetLogo are numbers (double and integer), boolean, string, color, agent, agentset and list. The **agentset** data type is a collection of agents and the list data type is a vector to store multiple values. **Blocks** of code are defined by embedding them into squared brackets and **comments** are available just as line comments beginning with a semicolon.
Variable declarations and **assignments** can be done by using the operator *set* or by using *let* for local variables, as shown in Listing 1. Advanced programmers need to get used to it, but for the novice this is a very natural way for assignments, as it is written like a spoken command.

➔ Insert Listing 1 anywhere around here

The NetLogo language distinguishes between **commands** and **reporters**. Commands are instructions to an agent whereas reporters calculate and return a value. There are around 400 built-in commands and reporters, called **primitives**. Some of them have to be executed in a specific agent type context, for example *move-to* is just for turtle context since it does not make sense to ask a patch for moving away as they are immobile. User-defined commands and reporters are called **procedures** and can consist of sequences of commands and reporters. They are defined by the opening keyword *to* or *to-report* followed by the name given to the procedure, which includes the things to do, and are closed by the keyword *end*. An example is given in Listing 2.
If necessary, input variables for procedures can be defined. Commands and reporters are executed by using their name. If inputs should be passed to the command or reporter, the user has to write them directly after the name of the command or reporter in a whitespace separated list. Because reporters return values on the right side of a reporter call, there has to be an output command or an assignment. Typical for Logo languages is that commands and reporters, although comparable with functions or methods in other languages, do not have parentheses after the name containing the input variables. There is no terminal character, like the semicolon in Java or C++. Everything is separated just by whitespaces. In some cases, when a primitive gets optional or repeatable inputs, the primitive and its inputs have to be declared as belonging together by using parentheses.

➔ Insert Listing 2 anywhere around here

A very important command is *ask* which iterates over the given agentset and executes the commands and reporters given in a block with the context of the current agent of the iterated agentset. This ask command is very powerful in combination with the *with* reporter, which creates a new agentset containing only those agents that satisfy a given condition. A scheduler for a simulation model could then look like the procedure *go* in Listing 3 and would be executed in observer context.

➔ Insert Listing 3 anywhere around here

NetLogo Integrated Simulation Environment

As mentioned before, NetLogo comes with an integrated, interactive simulation environment. There are three basic tabs, one for the model source code (**Procedures Tab**), one for the model description (**Information Tab**) and one for the Graphical User Interface (**Interface Tab**), as shown in **Fehler! Verweisquelle konnte nicht gefunden werden.**.

➔ Insert Figure 1 anywhere around here

When leaving the Procedures Tab or when dropping the Check Button, NetLogo runs the **spellchecker**, reports errors, if found (see Figure 2) and tokenizes as well as compiles parts of the code for execution/interpretation (for details on the combined compiler-interpreter architecture with the bytecode inlining technique see SONDAHL et al. 2006).

➔ Insert Figure 2 anywhere around here

Graphical elements can be placed somewhere on the Interface Tab by drag-and-drop. The user can add **Buttons**, **Sliders**, **Switches**, **Choosers**, **Inputs**, **Monitors**, **Plots** (Scatter, Line, and Bar plots), **Outputs** and **Notes**. Some of them take additional NetLogo commands or reporters. For example, a button, when pushed, executes the command the user gave to it, like the execution of a procedure.
One graphical element which is always there is the **View**, which visualizes the patches and the turtles. If one wants to implement a non-spatial model, it is possible to hide the turtles and reduce the world to just one remaining patch, which could be leaved unused and hidden behind a plot. If a spatial model is implemented, users can choose a wrapping world, choose the number and size of the patches and set the location of the origin of the coordinates. Within the world, patch, link and turtle variables can be inspected by clicking on the desired agent with the mouse. The variables of this agent are shown in a new window and can be watched and changed during and after the simulation. If required, NetLogo offers a 3D view of the world for perspective visualizations.
Furthermore, users can control the speed of a simulation via a slider which opens the possibility to slow down the simulation to observe the turtle's movement in detail.
With the **Command Center**, NetLogo delivers an interpreter. It is possible to execute any command or reporter during the simulation within the context of the observer, patch, turtle or link.

NetLogo Extensions and Controlling API

Due to NetLogo's design philosophy "low threshold, no ceiling", the developers included an interface for everyone to extend the NetLogo language. The Extension API offers a way to extend the language by adding user-defined primitives (STONEDAHL et al. 2008). Extensions can be written in Java or Scala and can access NetLogo objects, like turtles or lists.

NetLogo comes with a bundle of standard extensions, like the GIS extension, which adds functionality to import raster and vector datasets into a NetLogo model or the array, table and matrix extensions, which add multi-dimensional data storages.
The controlling API allows running the NetLogo application by remote control. It is possible to open models and execute NetLogo commands and reporters from other Java and Scala programs.

IV. Extensions to NetLogo

This last section of this chapter gives an overview over new extensions to and tools for NetLogo written by the authors of this chapter. The extensions were developed to extend the functionality of NetLogo for making this software more relevant as a standard tool of ABM and, in case of the R-Extension, to combine ABMs with standardized methods from statistics and to avoid hand-written solutions.

MultiView

As described above, NetLogo visualizes the patches in the world widget. For this, the built-in patch variable *pcolor* is used by default. The value of *pcolor* or any other patch variable, if defined, determines the color of each patch. But there is no way to create more than one world widget to visualize more than one patch variable at a time. For example, if we take a model like BEFORE (RADEMACHER et al. 2004) this could be a restriction. BEFORE simulates the dynamic of natural beech forests. It distinguishes four horizontal layers which are characterized by their coverage percentage. The forest itself is divided into quadratic patches, which are represented in NetLogo by the patches of the NetLogo world. The coverage percentage of each horizontal layer is stored in a user-defined patch variable. There is no useful way to visualize the four patch variables simultaneously.
Another example is a model where we want to compare different temporal stages at a time. For example, if one defines a patch variable which saves the value of the variable of interest of the last simulation step there is no way to see the current spatial pattern of the patch variables and the pattern of the last simulation step simultaneously.
Therefore, we developed an extension which adds new windows to the simulation showing the patches of the world. The windows contain a copy of the view, i.e. uses the settings of the view regarding patch size and the number of patches. The user can define the patch variable which is to be used for colorization of the patches in the new view window. There is also the right mouse click functionality within the view window available to inspect patches and to export the view into an image file (see Figure 3 and Listing 4). The user can add as many additional view windows as desired.

➔ Insert Figure 3 and Listing 4 anywhere around here

The extension adds just four new primitives to the NetLogo language: One for the creation of a new window, one for repainting/updating the view, another for renaming the window and a last for (tidy) closing the window. The extension is available for

download at the official NetLogo web page: http://ccl.northwestern.edu/netlogo/resources.html. It is released under the GNU GPL v2 open source license and comes with a short documentation and the source code.

R-Extension

NetLogo already provides some primitives for data analyses, like *mean*, *median*, *mode*, *variance* and *standard-deviation* but it lacks advanced methods. Therefore, we developed an extension to link NetLogo directly to the statistical software GNU R (THIELE & GRIMM 2010). There are two typical use cases of this extension that we had in mind while developing it. First, the integration of advanced statistical methods within the model itself and second the integrated and immediate stepwise analysis of simulation results.

The first goal includes methods like regression analysis or just random numbers from special random distributions. Imagine, for instance, that the amount of food intake of an agent is dependent on the expectations for the future which is based on the experiences of the past and current circumstances. Here, non-linear regression models could be fitted to the past values and used to make the forecast. In such a case, the stepwise fitting of the regression model and the forecast could be computed using GNU R and used by the NetLogo simulation. This would require an interactive use of both software packages.

An example of the second intended use of our R extension of NetLogo is the analysis of the spatial distribution of agents using spatial point pattern statistics like Ripley's K. Another example is the calculation of diversity indices. Furthermore, as R delivers advanced plotting functionalities, the R-Extension can be used to extend the limited plotting capabilities of NetLogo. Moreover, connections to all common databases could be established via GNU R.

By using the R-Extension, the modeler can save a lot of work and time by using R functions instead of programming statistical analysis from scratch. This guarantees to use reliable functions implemented and tested by the R programmers/contributors and users. Because R is a standard tool in statistical analysis, listing the R functions used describes the used methods comprehensively. Therefore, using the R-Extension keeps the NetLogo code short, clear, traceable and reduces the chance of doing bugs.

The extension makes use of the rJava-Package for GNU R and the Extension API of NetLogo. Via the rJava-Package, or more precisely the JRI library within the rJava-Package, it is possible to create R data types from Java via the Java Native Interface (JNI) and C. The NetLogo data types are accessed via the Extension API and converted into R data types within the R-Extension. R commands are evaluated within the R-Extension and return values are converted into NetLogo data types. Even NetLogo turtles, links, patches and lists are supported as well as R vectors, lists and dataframes.

The R-Extension adds nine new primitives to the NetLogo language for the interaction between NetLogo and R and six additional primitives for debugging purposes. Table 1 lists the different basic primitives.

➔ Insert Table 1 anywhere around here

A nice, but often not recognized, feature is the *r:interactiveShell* primitive. If NetLogo has been started via the shell/command line/ms-dos prompt and uses the R-Extension the user can redirect the connection to the underlying R session to the shell. Then, the user can work directly in the R session, can access the variables created with the R-Extension in NetLogo, can execute R commands and access newly created R variables within NetLogo using the R-Extension (see Figure 4).

➔ Insert Figure 4 anywhere around here

Figure 5 shows an example for the usage of the R-Extension in spatial statistics. The point pattern of the agent positions is used to calculate the L function (based on Ripley's K) with Monte-Carlo-Simulations for the null hypotheses test (complete spatial randomness). For this, the agent positions are sent to an R dataframe via the primitive *r:putagentdf*. By using the R-Package *spatstat* this agent positions dataframe is transformed into a planar point pattern object (ppp) and used as input for the *envelope* function, which performs 99 replicated simulations (Monte-Carlo-Simulation) for the hypotheses test. The pointwise critical envelopes for L(r) are then plotted using the standard plot function of R, which creates a gray band for the envelope with a dotted line for the theoretical value under complete spatial randomness and a solid line for the observed pattern. The code for this operation in NetLogo is shown in Listing 5. In Figure 5 you can see that it is also possible to send the result of the envelope calculation back to NetLogo and make a simple plot there, but without the nice gray bands and dotted lines. The NetLogo code for this step, which requires some data transformations for the plot routine, can be found in the examples which are delivered with the R-Extension.

➔ Insert Figure 5 and Listing 5 anywhere around here

The R-Extension is specific to the version number of GNU R/r-Java and NetLogo. It comes with documentation and different examples as well as the source code under the GNU GPL v2 open source license. It is available for download at the Berlios repository: http://netlogo-r-ext.berlios.de/. The Extension is tested on Windows XP, Windows Vista, Windows 7, Linux (Ubuntu, SuSe) and Apple Macintosh. It runs also on 64-bit systems. Since the release at the beginning of 2010 until December of 2010, it was downloaded more than 300 times. The most frequent problem which has been reported was the installation/configuration process. To create the connection to R, the user has to set two environment variables, which turned out to be too difficult for some users.

Pygments parser

Not a real extension but a small supporting tool is the NetLogo language definition for Pygments (POCCO 2010). Pygments is an open source syntax highlighting engine

written in Python which takes source code and produces output in different formats that contain syntax highlighting markup. Output formats include HTML, LaTeX, RTF, GIF, PNG, JPEG and others. It can be used as a library or as a command-line tool.
The NetLogo language definition makes it possible to generate automatically NetLogo model source code in different formats, as mentioned above, directly from the original source file. The output looks like in the Procedures Tab of NetLogo with respect to the colorization as well as the indentation and is therefore easily available and editable for publications in text processing software or for using it on websites.
It is written as a Plug-In for Pygments with a lexer based on regular expressions and keyword lists as well as a style definition for the colorization. It works for the primitives of the bundled extensions as well as for the MultiView- and R-Extension.
The Plug-In includes a setup script which automatically adds the Plug-In to Pygments. A command line call for creating an HTML output could look as shown in Listing 6. The user can choose between an embedded css-style (Cascading Style Sheet) definition within the HTML file or without. It is possible to create a separate css-file based on the style definition.

➔ Insert Listing 6 anywhere around here

The Pygments Plug-In for NetLogo language is available at http://www.uni-goettingen.de/de/72779.html.

Outlook

To further extend the functionality of NetLogo, which will help to strengthen its potential as a standard tool, further tools for NetLogo are currently in preparation. One of these tools is RNetLogo, a package for GNU R to include NetLogo simulations within R. This is the reverse connection of the R-Extension with its own strength. It will overcome the difficulties in the setup process of the R-Extension with the creation of environment variables and will have the functionalities of the Mathematica Link for NetLogo described in BAKSHY & WILENSKY (2007). It could be used to establish a standard protocol for calibrating and analyzing ABMs. GNU R with its hugh amount of packages is the ideal basis for designing simulation experiments and analyzing their results.
Another important functionality, which is currently missing in NetLogo, is a stepwise debugger, as mentioned by RAILSBACK et al. (2006). Such a tool is currently under development and will fill this gap.

Acknowledgements

We would like to thank Michael Henke for some helpful comments on the manuscript.

References

ALGERS, S., BERNAUER E., BOERO, M., BREHERET, L., DI TARANTO, C., DOUGHERTY, M., FOX, K., GABARD, J.-F. (1997): *A review of micro-simulation models*, Project Report SMARTEST/D3, Institute of Transport Studies, University of Leeds.

ALLAN, R. (2009): *Survey of agent based modelling and simulation tools*, Technical Report, STFC Daresbury Laboratory, http://epubs.cclrc.ac.uk/bitstream/3637/ABMS.pdf (accessed 2010/12/20).

AN, G. (2009): Dynamic knowledge representation using agent-based modeling: ontology instantiation and verification of conceptual models, *Methods in Molecular Biology*, 500, 445-68.

ARIFOVIC, J. (1994): Genetic algorithm learning and the cobweb model, *Journal of Economic Dynamics and Control*, 18, 3-28.

ARTHUR, W.B. (2006): Out-of-Equilibrium Economics and Agent-Based Modeling, in L. TESFATSION and K.L. JUDD (eds.), *Handbook of computational Economics Vol. 2: Agent-based computational economics*, 1551-1564, Amsterdam a.o.: Elsevier.

AXELROD, R. (1997): *The complexity of cooperation: agent-based models of competition and collaboration*, Princeton: Princeton Univ. Press.

AXTELL, R. (2000): Why agents? On the varied motivations for agent computing in the social sciences, Working Paper No. 17, Center on Social and Economic Dynamics, The Brookings Institution.

BAKSHY, E. and U. WILENSKY (2007): Turtle Histories and Alternate Universes; Exploratory Modeling with NetLogo and Mathematica, in M. J. NORTH, C. M. MACAL and D. L. SALLACH (eds.), *Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence,* 147-158, IL: Argonne National Laboratory and Northwestern University.

BALAN, G.C., CIOFFI-REVILLA, C., LUKE, S., PANAIT, L. and S. PAUS (2003): MASON: A java multi-agent simulation library, in *Proceedings of the Agent 2003 Conference*, http://cs.gmu.edu/~eclab/projects/mason/publications/Agent2003.pdf (accessed 2010/12/22).

BEN SAID, L., BOURON, T. and A. DROGOUL (2002): Agent-based interaction analysis of consumer behavior, in *Proceedings of AAMAS 2002*, First international joint conference on Autonomous agents and multiagent systems, 184-190.

BERRYMAN, M. (2008): *Review of Software Platforms for Agent Based Models*, Technical Report, Australian Government – Department of Defence, Defence Science and Technology Organisation, DSTO-GD-0532,

http://dspace.dsto.defence.gov.au/dspace/bitstream/1947/9306/1/DSTO-GD-0532%20PR.pdf (accessed 2010/12/15).

BILLARI, F.C. and A. PRSKAWETZ (2003): *Agent-based computational demography: using simulation to improve our understanding of demographic behavior*, Heidelberg: Physica-Verl.

BOSSEL, H. (1994): *Modeling and simulation*, Wellesley: A.K. Peters.

BOTKIN, D.B., JANAK, J.F. and J.R. WALLIS (1972): Some ecological consequences of a computer model of forest growth, *Journal of Ecology*, 60, 849-872.

BRECKLING, B. (2002): Individual-based modelling – Potentials and limitations, *TheScientificWorldJOURNAL*, 2002 (2), 1044-1062.

BUCHANAN, M. (2009): Economics: Meltdown modelling, *Nature*, 460, 680-682.

BUNSING, R.T. and D. MAILLY (2004): Advances in spatial, individual-based modelling of forest dynamics, *Journal of Vegetation Science*, 15, 831-842.

CARPENTER, C. and L. SATTENSPIEL (2009): The design and use of an agent-based model to simulate the 1918 influenza epidemic at Norway House, Manitoba, *American Journal of Human Biology*, 21, 290-300.

CASWELL, H. (2001): *Matrix population models: Construction, analysis and interpretation*, 2nd Edition, Sunderland, Mass.: Sinauer.

COATES, K.D., CANHAM, C.D., BEAUDET, M., SACHS, D.L. and C. MESSIER (2003): Use of a spatially explicit individual-tree model (SORTIE/BC) to explore the implications of patchiness in structurally complex forests, *Forest Ecology and Management*, 186, 297-310.

COLLIER, N., HOWE, T. and M.J. NORTH (2003): Onward and upward: The transition to Repast 2.0, in *First Annual North American Association for Computational Social and Organizational Science Conference*, Pittsburgh, PA USA, North American Association for Computational Social and Organizational Science.

COLLIER, N. and M. NORTH (2010): Repast Java getting started, http://repast.sourceforge.net/docs/RepastJavaGettingStarted.pdf (accessed 2010/12/22).

CONTE, R. (2009): From simulation to theory (and backward), *Epistemological Aspects of Computer Simulation in the Social Sciences*, Lecture Notes in Computer Science, 5466/2009, 29-47.

CONTE, R., GILBERT, N. and J.S. SICHMAN (1998): MAS and social simulation: A suitable commitment, *Multi-Agent Systems and Agent-Based Simulation*, Lecture Notes in Computer Science, 1534/1998, 1-9.

COURNÈDE, P.-H., GUYARD, T., BAYOL, B., GRIFFON, S., DE COLIGNY, F., BORIANNE, P., JAEGER, M. and P. DE REFFYE (2010): A forest growth simulator based on functional-structural modelling of individual trees, in B. Li, M. Jaeger and Y. Guo (eds.), *Plant Growth Modeling, Simulation, Visualization and Applications*, 34-41, Proceedings PMA09, Los Alamitos: IEEE Computer Society.

DAHL, O.-J., MYHRHAUG, B. and K. NYGAARD (1967): *SIMULA 67, Common Base Language*, Oslo: Norwegian Computing Center.

DAVIDSSON, P. (2002): Agent based social simulation: A Computer Science View, *Journal of Artificial Societies and Social Simulation*, 5(1)7.

DEANGELIS, D.L., BARNTHOUSE, L.W., van WINKLE, W. and R.G. OTTO (1990): A critical appraisal of population approaches in assessing fish community health, *Journal of Great Lakes Research*, 16, 576-590.

DEANGELIS, D.L., COX, D.K. and C.C. COUTANT (1980): Cannibalism and size dispersal in young-of-the-year largemouth bass: Experiment and model, *Ecological Modelling*, 8, 133-148.

DEANGELIS, D.L. and L.J. GROSS (1992): *Individual-based models and approaches in ecology: Populations, communities and ecosystems*, New York: Chapman and Hall.

DEANGELIS, D.L. and W.M. MOOIJ (2005): Individual-based modeling of ecological and evolutionary processes, *Annual Reviews of Ecology and Evolutionary Systematics*, 36, 147-168.

DEANGELIS, D.L., ROSE, K.A. and M.A. HUSTON (1994): Individual-oriented approaches to modeling ecological populations and communities, in S. A. Levin (ed.), *Frontiers in Mathematical Biology*, Lecture Notes in Biomathematics 100, 390-410, New York: Springer-Verlag.

DRECHSLER, M., GRIMM, V., MYSIAK, J. and F. WÄTZOLD (2007): Differences and similarities between ecological and economic models for biodiversity conservation, *Ecological Economics*, 62, 232-241.

EPSTEIN, J.M. and R. AXTELL (1996): *Growing artificial societies: Social science from bottom up*, Washington DC a.o.: Brookings Inst. Press.

FARMER, J.D. and D. FOLEY (2009): The economy needs agent-based modeling, *Nature*, 460, 685-686.

FERRER, J., PRATS, C. and D. LÓPEZ (2008): Individual-based modelling: an essential tool for microbiology, *Journal of Biological Physics*, 34, 19-37.

FORRESTER, J.W. (1971): *World dynamics*, Cambridge: MIT Press.

GALÁN, J.M., IZQUIERDO, L.R., IZQUIERDO, S.S., SANTOS, J.I., DEL OLMO, R., LÓPEZ-PAREDES, A. and B. EDMONDS (2009): Errors and artefacts in agent-based modelling, *Journal of Artificial Societies and Social Simulation*, 12(1)1.

GILBERT, N. (1999): Simulation: A new way of doing social science. *American Behavioral Scientist*, 40, 1485-1487.

GILBERT, N. (2004): Agent-based social simulation: dealing with complexity, http://cress.soc.surrey.ac.uk/resources/ABSS%20-%20dealing%20with%20complexity-1-1.pdf (accessed 2010/12/22).

GILBERT, N. (2008): *Agent-based models*, Quantitative Applications in the Social Sciences, Los Angeles a.o.: SAGE Publ.

GILBERT, N. and K.G. TROITZSCH (1999): *Simulation for the social scientist*, Buckingham a.o.: Open Univ. Press.

GINOVART, M., LÓPEZ, D. and J. VALLS (2002): INDISIM, an individual-based discrete simulation model to study bacterial cultures, *Journal of Theoretical Biology*, 214, 305-19.

GODIN, C. and H. SINOQUET (2005): Functional-structural plant modelling, *New Phytologist*, 166, 705-708.

GREEN, S., HURST, L., NANGLE, B., CUNNINGHAM, P., SOMERS, F. and R. EVANS (1997): *Software agents: A review*, Computer Science Technical Report TCD-CS-1997-06, Trinity College Dublin, Department of Computer Science.

GRIFFIN, A.F. and C. STANISH (2007): An agent-based model of prehistoric settlement patterns and political consolidation in the Lake Titicaca Basin of Peru and Bolivia, *Structure and Dynamics*, 2, 1-46.

GRIMM, V. (1999): Ten years of individual-based modelling in ecology: What have we learned, and what could we learn in the future?, *Ecological Modelling*, 115, 129-148.

GRIMM, V., BERGER, U., BASTIANSEN, F., ELIASSEN, S., GINOT, V., GISKE. J., GOSS-CUSTARD, J., GRAND, T., HEINZ, S.K., HUSE, G., HUTH, A., JEPSEN, J.U., JØRGENSEN, C., MOOIJ, W.M., MÜLLER, B., PE'ER, G., PIOU, C., RAILSBACK, S.F., ROBBINS, A.M., ROBBINS, M.M., ROSSMANITH, E., RÜGER, N., STRAND, E., SOUISSI, S., STILLMAN, R.A., VABØ, R., VISSER, U. and D.L. DEANGELIS (2006): A standard protocol for describing individual-based and agent-based models, *Ecological Modelling*, 198, 115-126.

GRIMM, V., BERGER, U., DEANGELIS, D.L., POLHILL, J.G., GISKE, J. and S.F. RAILSBACK (2010): The ODD protocol: A review and first update, *Ecological Modelling*, 221, 2760-2768.

GRIMM, V. and S.F. RAILSBACK (2005): *Individual-based modeling and Ecology*, Princeton: Princeton Univ. Press.

GUN, O. (2004): Why do we have separate courses in 'Micro' and 'Macro' economics, in E. Fullbrook (eds.), *A guide to what's wrong with economics*, London: Anthem Press.

HAMILL, L. (2010): Agent-based modelling: The next 15 years, *Journal of Artificial Societies and Social Simulation,* 13(4)7.

HEATH, B., HILL, R. and F. CIARALLO (2009): A survey of agent-based modeling practices (January 1998 to July 2008), *Journal of Artificial Societies and Social Simulation*, 12(4)9.

HECKBERT, S., BAYNES, T. and A. REESON (2010): Agent-based modeling in ecological economics, *Annals of the New York Academy of Sciences*, 1185, 39-53.

HEMMERLING, R., KNIEMEYER, O., LANWERT, D., KURTH, W. and G. BUCK-SORLIN (2008): The rule-based language XL and the modelling environment GroIMP illustrated with simulated tree competition, *Functional Plant Biology*, 35, 739-750.

HEWITT, C. (1976): *Viewing control structures as patterns of passing messages*, A.I.MEMO 410, Cambridge: MIT Press.

HIEBELER, D. (1994): The Swarm simulation system and individual-based modeling, in J.M. POWER, M. STROME and T.C. DANIEL (eds.), Decision Support 2001. *17th Annual Geographic Information Seminar and the Resource Technology '94 Symposium*, 474-494, American Society for Photogrammetry and Remote Sensing.

HOGEWEG, P. and B. HESPER (1990): Individual-oriented modelling in ecology, *Mathematical and Computer Modelling*, 13, 83-90.

HUSTON, M., DEANGELIS, D. and W. POST (1988): New computer models unify ecological theory, *BioScience*, 38, 682-691.

JANSSEN, M.A., ALESSA, L.N., BARTON, M., BERGIN, S. and A. LEE (2008): Towards a community framework for agent-based modelling, *Journal of Artificial Societies and Social Simulation*, 11(2)6.

JENNINGS, N.R. (1999): On agent-based software engineering, *Artificial Intelligence*, 117, 277-296.

JENNINGS, N.R., SYCARA, K. and M. WOOLDRIDGE (1998): A roadmap of agent research and development, *Autonomous Agents and Multi-Agent Systems*, 1, 275-306.

JUDSON, O.P. (1994): The rise of individual-based model in ecology, *Trends in Ecology and Evolution*, 9, 9-14.

KAISER, H: (1974): Populationsdynamik und Eigenschaften einzelner Individuen, *Verhandlungen der Gesellschaft für Ökologie*, 4, 25-38.

LEBARON, B. (2000): Agent Based Computational Finance: Suggested Readings and Early Research, *Journal of Economic Dynamics and Control*, 24, 679-702.

LEOMBRUNI, R. and M. RICHIARDI (2004): Industry and labor dynamics: the agent-based computational economics approach, *Proceedings of the Wild@Ace 2003 Conference*, Singapore: World Scientific Publ.

LI, X., MAO W., ZENG, D. and F.-Y. WANG (2008): *Agent-Based Social Simulation and Modeling in Social Computing*, Lecture Notes in Computer Science, 5075/2008: 401-412.

LIU, J., and P.S. ASHTON (1995): Individual-based simulation models for forest succession and management, *Forest Ecology and Management*, 73, 157-175.

LOGO FOUNDATION (2010): The Logo programming language, http://el.media.mit.edu/logo-foundation/logo/programming.html (accessed 2010/12/22).

ŁOMNICKI, A. (1978): Individual differences between animals and the natural regulation of their numbers, *Journal of Animal Ecology*, 47, 461-475.

ŁOMNICKI, A. (1988): *Population ecology of individuals*, Princeton: Princeton University Press.

ŁOMNICKI, A. (1999): Individual-based models and the individual-based approach to population ecology, *Ecological Modelling*, 115, 191-198.

LOREK, H. and M. SONNENSCHEIN (1999): Modelling and simulation software to support individual-based ecological modelling, *Ecological Modelling*, 115, 199-216.

LUCK, M., MCBURN, P. and C. PREIST (2003): *Agent technology - Enabling next generation computing: A roadmap for agent based computing*, AgentLink, Southampton: University of Southampton.

LUKE, S., CIOFII-REVILLA, C., PANAIT, L. and K. SULLIVAN (2004): MASON: A new multi-agent simulation toolkit, in *Proceedings of the 2004 SwarmFest Workshop*, http://cs.gmu.edu/~eclab/projects/mason/publications/SwarmFest04.pdf (accessed 2010/12/22).

LUKE, S., CIOFFI-REVILLA, C., PANAIT, L., SULLIVAN, K. and G. BALAN (2005): MASON: A multi-agent simulation environment, *Simulation*, 82, 517-527.

MACY, M. and R. WILLER (2002): From factors to actors: Computational sociology and agent-based modeling, *Annual Review of Sociology*, 28, 143-166.

MALLESON, N., HEPPENSTALL, A. and L. SEE (2010): Crime reduction through simulation: An agent-based model of burglary, Computers, *Environment and Urban Systems*, 34, 236-250.

MASON (2010): MASON Class Overview, http://cs.gmu.edu/~eclab/projects/mason/docs/overview.html (accessed 2010/12/22).

MATTHEWS, R.B., GILBERT, N.G., ROACH, A., POLHILL, J.G. and N.M. GOTTS (2007): Agent-based land use models: a review of applications, *Landscape Ecology*, 22, 1447-1459.

MEADOWS, D. (1974): *The limits to growth: A report for the Club of Rome's project on the predicament of mankind*, 2nd Edition, New York: Universe Books.

MINAR, N., BURKHART, R., LANGTON, C. and M. ASKENAZI (1996): *The Swarm simulation system: a toolkit for building multi-agent simulations*, Working Paper 96-06-042, Santa Fe Institute, Santa Fe.

MOONEN, J.M. (2009): *Multi-agent systems for transportation planning and coordination*, ERIM Ph.D. series research in management 177, Rotterdam.

MYERS, J.H. (1976): Distribution and dispersal in populations capable of resource depletion – a simulation study, *Oecologia*, 23, 255-269.

NETLOGO (2010a): NetLogo, http://ccl.northwestern.edu/netlogo/index.shtml (accessed 2010/12/22).

NETLOGO (2010b): What's new?, http://ccl.northwestern.edu/netlogo/docs/versions.html (accessed 2010/12/22).

NETLOGO (2010c): Copyright and license information, http://ccl.northwestern.edu/netlogo/docs/copyright.html (accessed 2010/12/22).

NETLOGO (2010d): FAQ (Frequently asked questions), http://ccl.northwestern.edu/netlogo/faq.html (accessed 2010/12/22).

NIKOLAI, C. and G. MADEY (2009): Tools of the trade: A survey of various agent based modeling platforms, *Journal of Artificial Societies and Social Simulation*, 12(2)2.

NORTH, M.J., MACAL, C.M., AUBIN, J.S., THIMMAPURAM, P., BRAGEN, M., HAHN, J., KARR, J., BRIGHAM, N., LACY, M.E. and D. HAMPTON (2010): Multiscale agent-based consumer market modeling, *Complexity*, 15, 37-47.

OEFFNER, M. (2008): Agent–Based Keynesian Macroeconomics - An Evolutionary Model Embedded in an Agent–Based Computer Simulation, Ph.D. diss., University of Würzburg.

OLIVEIRA, E. (1999): Applications of intelligent agent-based systems, in: *Proceedings of SBAI - 4. Simpósium Brasileiro de Automação Inteligente*, 51-58, São Paulo.

PHAN D. (2004): From Agent-Based Computational Economics towards Cognitive Economics, in P. BOURGINE and J.P. NADAL (eds.), *Cognitive Economics*, 371-398, Berlin a.o.: Springer Verlag.

POCCO (2010): Pygments Syntax Highlighter, http://www.pocoo.org/projects/pygments/#pygments (accessed 2010/12/28).

POLHILL, J.G. (2009): Agent-based modeling of socio-economic processes related to the environment: Example of land-use change, in P.C. Baveye, M. Laba and J. Mysiak (eds.), *Uncertainties in Environmental Modelling and Consequences for Policy Making*, NATO

Science for Peace and Security Series C: Environmental Security, Theme I, 61-76, Dordrecht: Springer Science + Business Media.

POLHILL, J.G (2010): ODD updated, *Journal of Artificial Societies and Social Simulation*, 11(2)3.

PREMO, L.S. (2007): Exploratory agent-based models: Towards an experimental ethnoarchaeology, in J. T. Clark and E. M. Hagemeister (eds.), *Digital discovery: Exploring new frontiers in human heritage*, 29-36, CAA 2006 Computer Applications and Quantitative Methods in Archaeology, Budapest: Archeolingua Press.

PREMO, L.S. and S.L. KUHN (2010) Modeling effects of local extinctions on culture change and diversity in the Paleolithic, *PLoS ONE*, 5, 1-10.

PRETZSCH, H. (2009): *Forest dynamics, growth and yield: From measurement to model*, Berlin a.o.: Springer-Verlag.

RADEMACHER, C., NEUERT, C., GRUNDMANN, V., WISSEL, C. and V. GRIMM (2004): Reconstructing spatiotemporal dynamics of central European beech forests: The rule-based model BEFORE, *Forest Ecology and Management*, 194, 349-368.

RAILSBACK, S.F., LYTINEN, S.L and S.K. JACKSON (2006): Agent-based simulation platforms: Review and development recommendations, *Simulation, 82, 609-623.*

RAILSBACK, S.F. and V. GRIMM (in press): *Agent-based and Individual-based Modeling: A Practical Introduction*, Princeton: Princeton Univ. Press.

REED, M.J., MILLS, L.S., DUNNING, J.B., MENGES, E.S., MCKELVEY, K.S., FREYE, R., BEISSINGER, S.R., ANSTETT, M.-C. and P. MILLER (2002): Emerging issues in population viability analysis, *Conservation biology*, 16, 7-19.

REUTER, H. (1998): Multidimensional biotic community interaction of small rodents: Assessment through object oriented modelling, *ASU Newsletter*, 24 Suppl., 27-40.

RICHIARDI, M. (2007): Agent-based Computational Economics: A Short Introduction, Working Paper No. 69, LABORatorio R. Revelli, Centre for Employment Studies.

RICHIARDI, M., LEOMBRUNI, R., SAAM, N.J. and M. SONNESSA (2006): A common protocol for agent-based social simulation, *Journal of Artificial Societies and Social Simulation*, 9(1)15.

ROAD (2010a): Repast files, http://sourceforge.net/projects/repast/files/Repast/ (accessed 2010/12/20).

ROAD (2010b): Repast Simphony, http://repast.sourceforge.net/repast_simphony.html (accessed 2010/12/21).

ROAD (2010c): Repast for High Performance Computing, http://repast.sourceforge.net/repast_hpc.html (accessed 2010/12/21).

SCHELLING, T. (1969): Models of segregation, *American Economic Review*, Papers and Proceedings of the Eighty-first Annual Meeting of the American Economic Association, 59, 488-493.

SCHELLING (1971): Dynamic models of segregation, *Journal of Mathematical Sociology*, 1, 143–186.

SHEN, W., HAO, Q., YOON, H.J. and D.H. NORRIE (2006): Applications of agent-based systems in intelligent manufacturing: An updated review, *Advanced Engineering Informatics*, 20, 415-431.

SIMON, C. and M. ETIENNE (2009): A companion modelling approach applied to forest management planning with the Société Civile des Terres du Larzac, *Environmental Modelling and Software*, 25, 1371-1384.

SMITH, A. (1776): *An Inquiry into the Nature and Causes of the Wealth of Nations*, London: Printed For W. Strahan; And T. Cadell.

SONDAHL, F., TISUE, S. and U. WILENSKY (2006): Breeding faster turtles: Progress towards a NetLogo compiler, *Paper presented at Agent 2006*, Chicago, IL.

SPIELAUER, M. (2007): Dynamic microsimulation of health care demand, health care finance and the economic impact of health behaviours: Survey and review, *International Journal of Microsimulation*, 1, 35-53.

SQUAZZONI, F. (2010): The impact of agent-based models in the social sciences after 15 years of incursions, *History of Economic Ideas*, xviii/2010/2, 197-233.

STARFIELD, A.M. (1997): A pragmatic approach to modeling for wildlife management, *Journal of Wildlife Management*, 61, 261-270.

STARFIELD, A.M., SMITH, K.A. and A.L. BLELOCH (1990): *How to model it: Problem solving for the computer age*, New York: McGraw-Hill.

STONEDAHL, F., KORNHAUSER, D., RUSSELL, E., BROZEFSKY, C., VERREAU, E., TISUE, S. and U. WILENSKY (2008): Tinkering with turtles: An overview of NetLogo's Extensions API, *Paper presented at the annual meeting of the Swarm Development Group*, Chicago, IL.

SWARM (2010a): Swarm Development Group, http://www.swarm.org/index.php/Talk:Swarm_Development_Group (accessed 2010/12/22).

SWARM (2010b): Documentation set for Swarm 2.2, http://www.swarm.org/swarmdocs-2.2/set/set.html (accessed 2010/12/22).

SYCARA, K.P. (1998): Multiagent system, *AI Magazine*, 19, 79-92.

TESFATSION, L. (2006): Agent-based computational economics: A constructive approach to economic theory, in L. TESFATSION and K.L. JUDD (eds.), *Handbook of computational Economics Vol. 2: Agent-based computational economics*, 833-880, Amsterdam a.o.: Elsevier.

THIELE, J.C. and V. GRIMM (2010): NetLogo meets R: Linking agent-based models with a toolbox for their analysis, *Environmental Modelling & Software*, 25, 972-974.

TISUE, S. and U. WILENSKI (2004a): NetLogo: Design and Implementation of a Multi-Agent Modeling Environment, *Presented at SwarmFest*, Ann Arbor, May 9-11. 2004.

TISUE, S. and U. WILENSKI (2004b): NetLogo: A simple environment for modeling complexity**,** *Paper presented at the International Conference on Complex Systems*, Boston, May 16 - 21.

TOBIAS, R. and C. HOFMANN (2004): Evaluation of free Java-libraries for social-scientific agent based simulation, *Journal of Artificial Societies and Social Simulation*, 7(1)6.

UCHMANSKI, J. and V. GRIMM (1996): Individual-based modelling in ecology: What makes the difference?, *Trends in Ecology and Evolution*, 11, 437-441.

VOS, J., MARCELIS, L. F. M. and J. B. EVERS (2007): Functional-structural plant modelling in crop production – adding a dimension, in J. Vos, L. F. M. Marcelis, P. H. B. de Visser, P. C. Struik and J. B. Evers (eds.), *Functional-Structural Plant Modelling in Crop Production*, 1-12, Dordrecht: Springer.

WEISS, G. (1999): *Multiagent systems: A modern approach to distributed artificial intelligence*, Cambridge: MIT Press.

WIKIPEDIA (2010): Comparison of agent-based modeling software, http://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software (accessed 2010/12/22).

WILENSKY, U. (1999): NetLogo, http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

WILENSKY, U. and W. RAND (in press): *An introduction to agent-based modeling: Modeling natural, social and engineered complex systems with NetLogo*, Cambridge, MA: MIT Press.

WINDRUM, P., FAGIOLO, G. and A. MONETA (2007): Empirical validation of agent-based models: Alternatives and prospects, *Journal of Artificial Societies and Social Simulation*, 10(2)8.

WOOLDRIDGE, M. and N.R. JENNINGS (1995): Intelligent agents: theory and practice, *The Knowledge Engineering Review*, 10, 115-152.

WOOLDRIDGE, M.J. (2002): *An introduction to multiagent systems*, Chichester a.o.: Wiley.

ZAIDI, A. and K. RAKE (2001): *Dynamic microsimulation models: A review and some lessons from SAGE*, SAGE Discussion Paper No. 2.

Figures

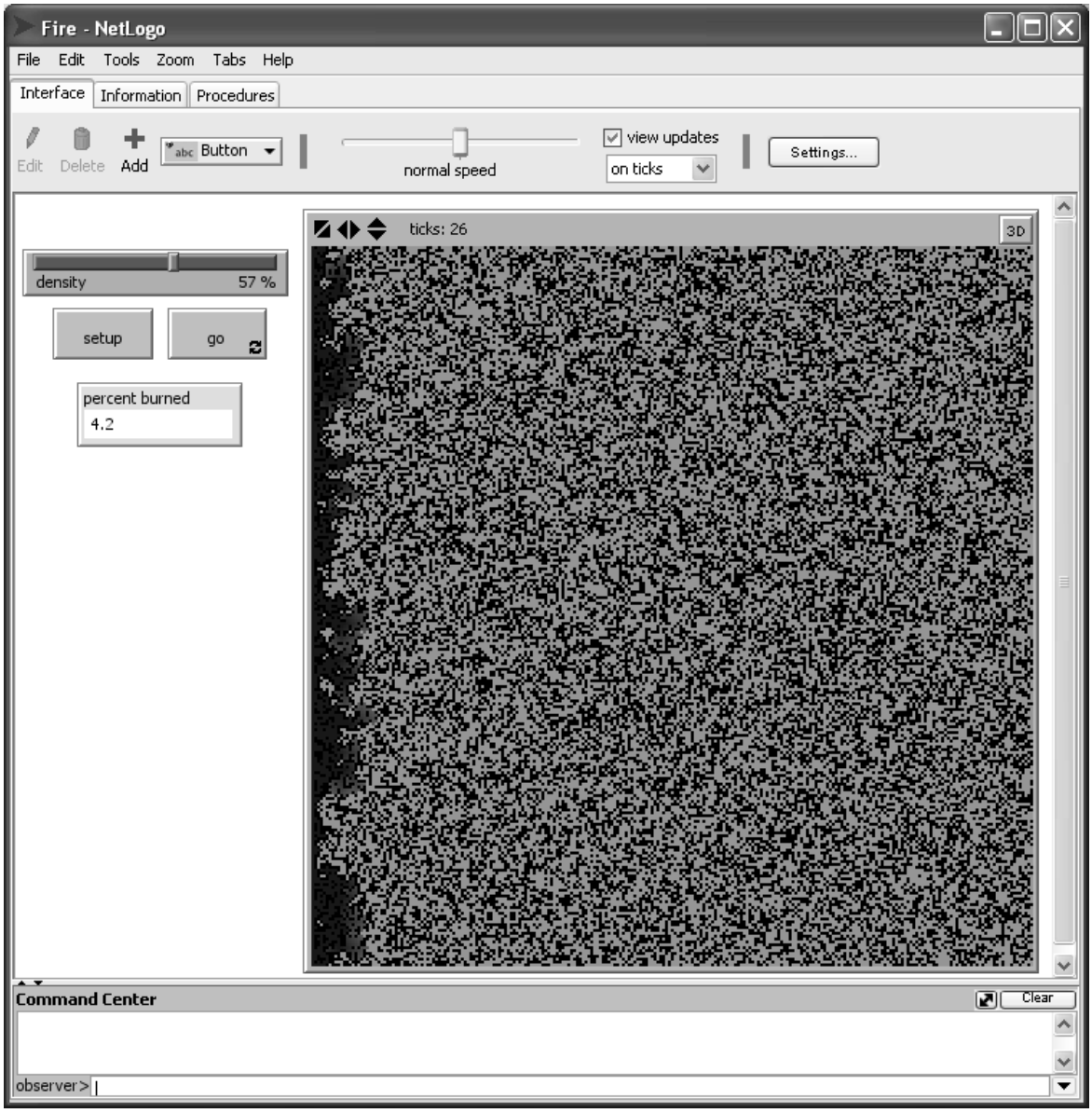

Figure 1: NetLogo GUI, Interface Tab with the View/World on the right, some control widgets on the left and Command Center at the bottom.

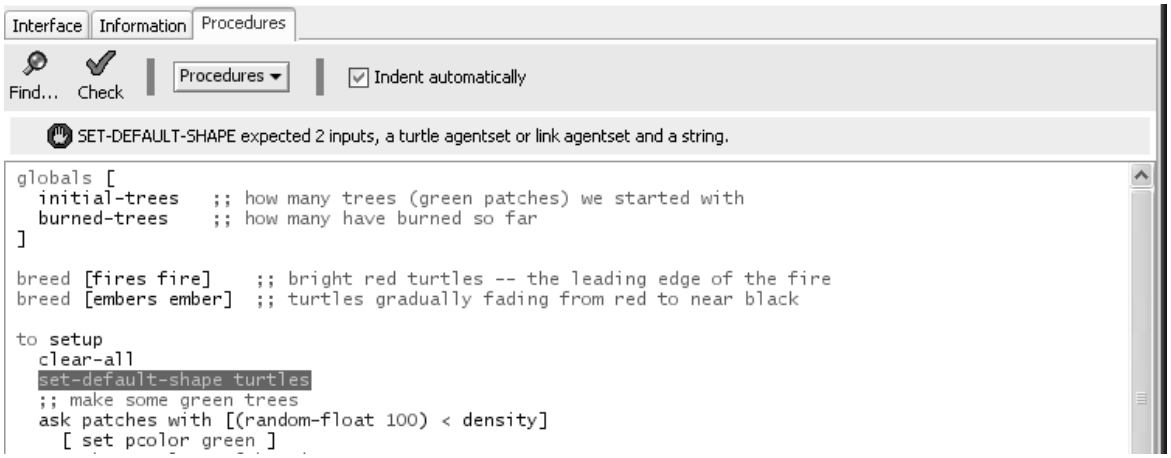

Figure 2: NetLogo GUI, Procedures Tab with Syntax Checker indicating an error.

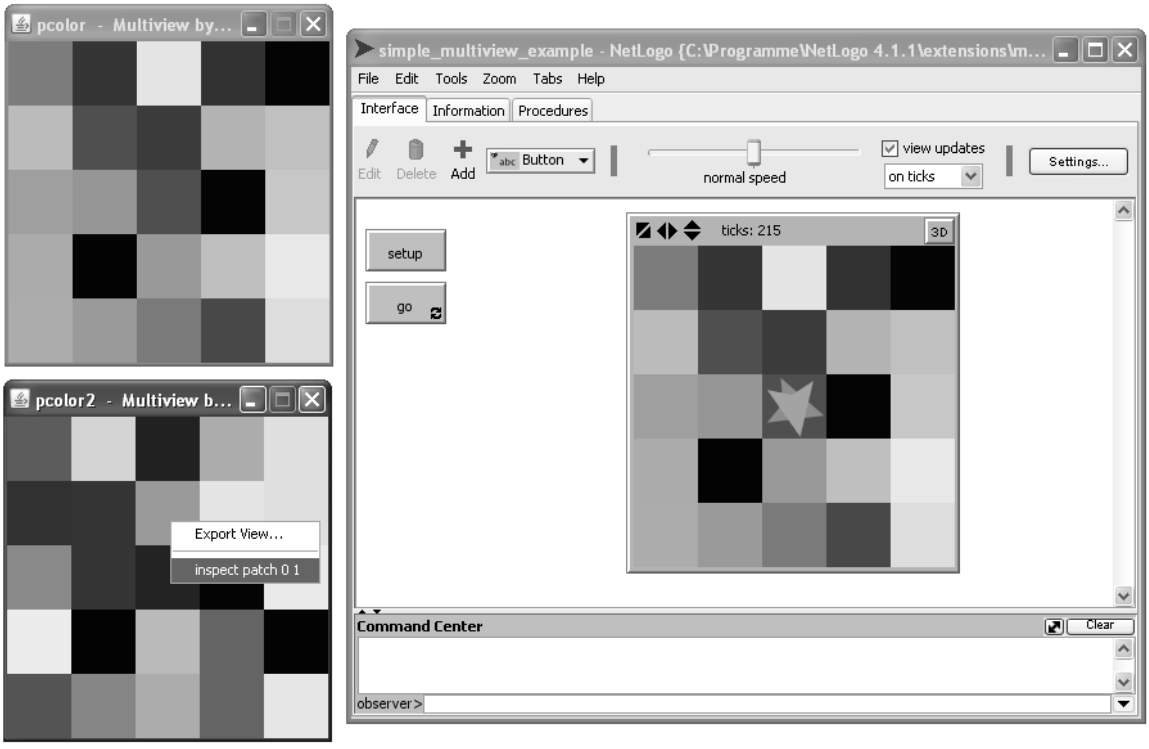

Figure 3: NetLogo GUI on the right with two additional view windows on the left created by the MultiView-Extension. The additional view window on the top uses the pcolor variable for colorization and is therefore a duplication of the original view, the bottom view uses the *pcolor2* variable for colorization as shown in Listing 4. On the bottom view, the right click functionality is shown. The user can export the current view into an image file and can inspect the selected patch.

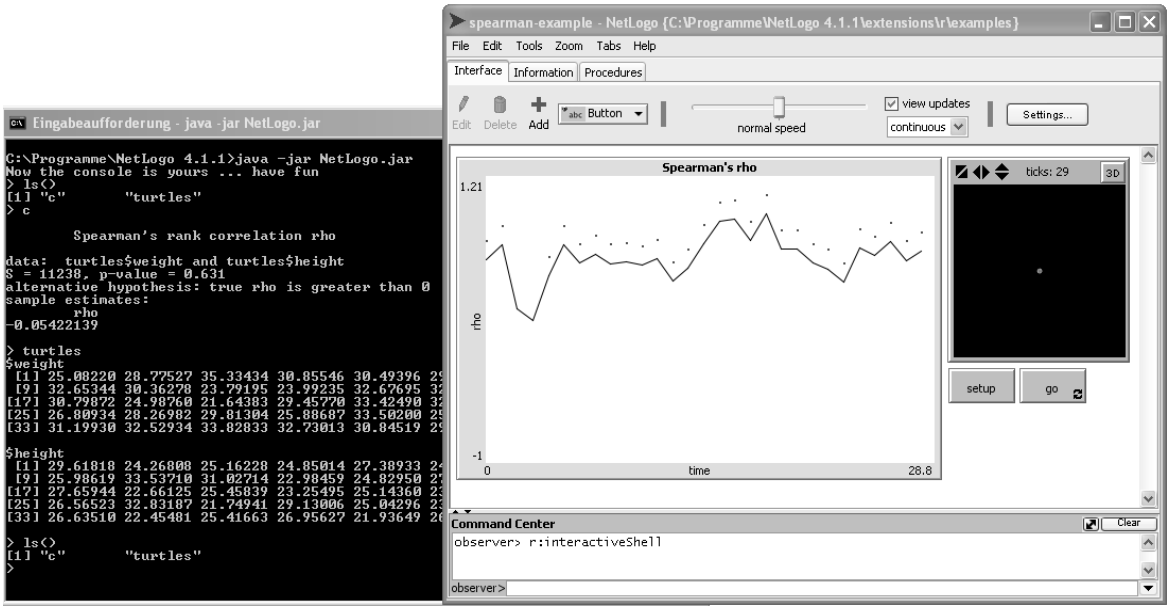

Figure 4: NetLogo started from an MS DOS prompt with calculation of Spearman's Rho over time using the R-Extension (on the right hand side); using the R-Extension's *interactiveShell* primitive to get access to the R session in the MS DOS prompt (on the left hand side) and checking the content of the *turtles* variable created by the R-Extension in NetLogo as well as listing the content of the R session.

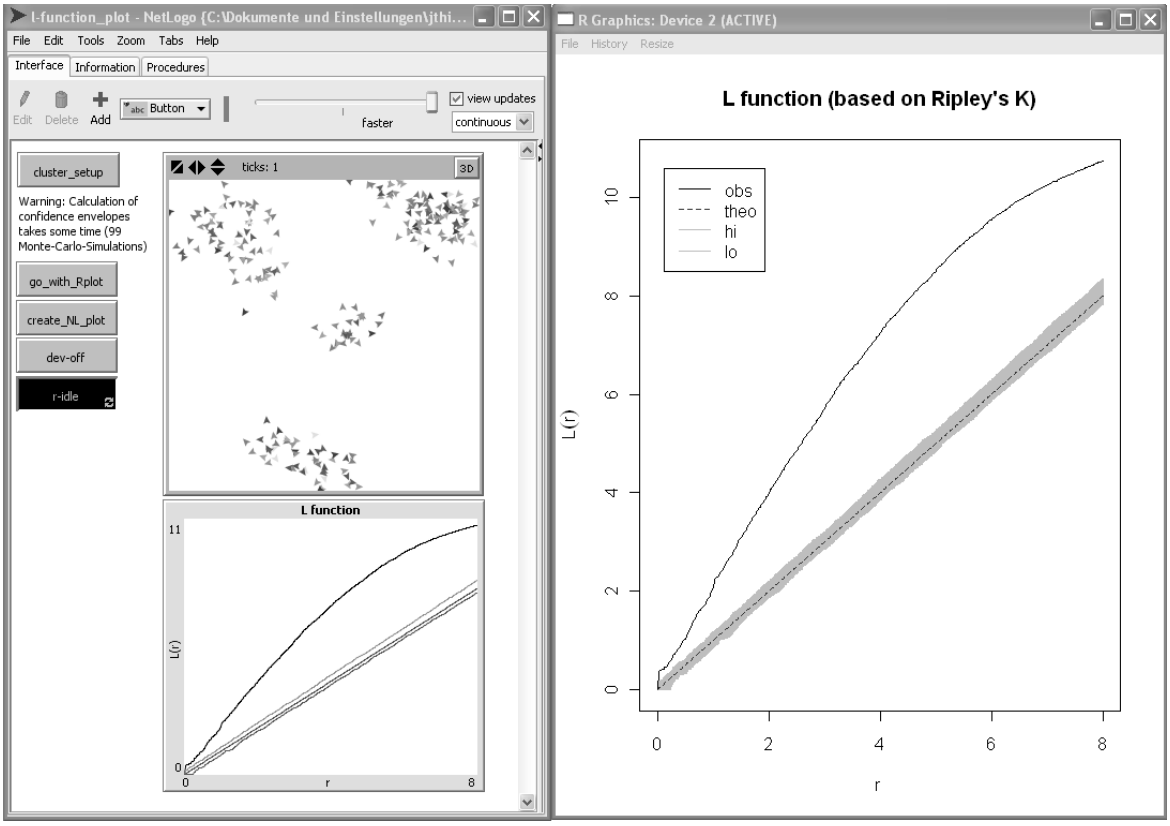

Figure 5: NetLogo GUI on the left, GNU R plot window called from NetLogo R-Extension on the right; the results of the L function calculation are sent back to NetLogo and are visualized in a plot on the bottom of the NetLogo GUI as well.

Tables

Table 1: Primitives added by the R-Extension with a short description (without the additional debugging primitives).

Primitive	Function
r:clear	Clears the R workspace, deletes all variables.
r:eval	Evaluates the submitted string in R, used for R function without a return value.
r:get	Gets a value from R; either submits a string including an R function with return value, or gets the value of a variable; return values can be strings, numbers, booleans, lists, or a list of lists.
r:put	Creates a new variable in R with value(s) from NetLogo; submitted values/variables can be strings, numbers, booleans, or lists (NetLogo lists become R vectors, to create R lists see putlist).
r:putagent	Creates a list in R from variables of a set of NetLogo agents; collections of agent-variables can be stored in a single named list.
r:putagentdf	Same as putagent, but creates an R dataframe instead of a list because many R functions requires dataframes as input.
r:putlist	Creates a new R list based on the submitted NetLogo variables and values.
r:putnamedlist	Same as putlist, but creates a named R list, i.e. the columns of the list can be called via their names.
r:putdataframe	Same as putnamedlist, but creates an R dataframe instead of a list because many R functions require dataframes as input.

Listings

```
let my-local-var 10 ; declaration and first assignment
set my-local-var 20 ; re-assignment
```

Listing 1: Declaration (and assignment) of a new local variable in the first line and re-assignment in the second line. The assignment of global variables is the same as the re-assignment of local variables. This code fragment could be embedded into a user defined procedure.

```
to setup
        clear-all
        create-turtles 2
end
```

Listing 2: A user defined procedure with the name setup which resets everything and creates two turtles.

```
breed [beeches beech]          ; create a new breed (turtle subtype) named beech

to setup                       ; a user-defined procedure setup, to initialize simulation
…
end

to go                          ; procedure, executed in observer context
        tick                   ; increment internal simulation time
        ask patches [          ; do for all patches
                update-soil-water ; call procedure update-soil-water in patch context
        ]
        ask beeches [          ; to for all beeches
                grow
                mortality
        ]
end

to update-soil-water
        …
end

to grow
        …
end

to mortality
        …
end
```

Listing 3: A (fragmented) example of a model source code with a user-defined turtle type (beech breed), a setup procedure to initialize the simulation, a go procedure as a scheduler for one simulation step which iterates over all beech-turtles and some other procedures called from the go procedure.

```
; load the extension
extensions [multiview]

; a patch variable to save the 2nd color (beside pcolor) for the 2nd view window
patches-own [ pcolor2 ]

globals [
  view1 ; a variable to save the reference to a view window
  view2 ; a 2nd variable to save the reference to a 2nd view window
]

to setup
  ; clear-all, closes the view windows as well, view1 & view2 has value 0
  ca
  ; create two turtles
  crt 2 [ set xcor random xcor set ycor random ycor]
  ; set the colors of the patches, pcolor and pcolor2 will be the same initially
  ask patches [
    set pcolor random 300
    set pcolor2 pcolor
  ]
  ; open the 1st view window, using pcolor (as in the NetLogo world view) to
  ; colorize the patches
  ; first parameter "pcolor" is the title of the window
  ; second parameter "pcolor2" is the patch variable used for coloring the patches
  ; save the reference to the view window in the variable "view1"
  set view1 multiview:newView "pcolor" "pcolor"
  ; open the second view window, using pcolor2
  set view2 multiview:newView "pcolor2" "pcolor2"
end

to go
  tick
  ; change the color values of the patches
  ask patches [
    set pcolor random 300
    set pcolor2 random 100
  ]
  ; update the colors of the view windows (using the predefined patch variables,
  ; see above: multiview:newView)
  multiview:repaint view1
  multiview:repaint view2
end
```

Listing 4: An example for the usage of the MultiView-Extension which creates two additional view windows, one as a copy of the original NetLogo world view and a second with a colorization using variable *pcolor2*.

```
extensions [r]

to cluster_setup
  ; clear all
  ca
  r:clear
  ask patches [ set pcolor 9.9]
  random-seed 1234567

  ; load R package spatstat for spatial statistics
  r:eval "library(spatstat)"

  ask n-of 10 patches [
    ask neighbors [
      sprout 3 [
        right random 360
        forward random 2
      ]
    ]
  ]
end

to go_with_Rplot
  tick
  ; let the turtles walk around randomly
  ask turtles [
    right random 360
    forward random 4
  ]

  ; send agent variables into an R dataframe
  (r:putagentdf "agentset" turtles "who" "xcor" "ycor")

  ; create point pattern with vectors of x- and y-coordinates of turtles and the
  ; dimension of the world
  let revalstring (word "agppp <- ppp(agentset$xcor, agentset$ycor,
                    c("min-pxcor","max-pxcor"), c("min-pycor","max-pycor"))")
  r:eval revalstring

  ; calculate L function with critical bands from 99 Monte-Carlo-Simulations
  r:eval "Lsim <- envelope(agppp, Lest)"
  ; plot the L function
  r:eval "plot(Lsim, main=\"L function (based on Ripley's K)\")"
end
```

Listing 5: An example for the usage of the R-Extension. In the setup procedure some turtles are created with a clustered spatial distribution. Furthermore, the R package spatstat for spatial statistics is loaded. In the go procedure the turtles walk around randomly first. Then, the positions of the turtles are sent to R into a dataframe. This dataframe is used to create a point pattern object which is used to calculate the L function with critical bands. The result of the L function calculation/simulation is plotted using R.

a. with embedded css-style:

```
pygmentize -l NetLogo -O full,style=NetLogo -f html -o test1.html test1.nlogo
```

b. with extra css-file:

Create the html file:

```
pygmentize -l NetLogo -f html -o test1.html test1.nlogo
```

Export the style to css file:

```
pygmentize -f html -S NetLogo -a .syntax > netlogosyle.css
```

Listing 6: An example for the usage of the Pygment NetLogo Plug-In to create an HTML file (test1.html) from a NetLogo model file (test1.nlogo) with embedded css-style (a) and separated css-file (b).