Agent-Based Modeling for Archaeologists: Part 1

Q1 *Iza Romanowska, Stefani A. Crabtree, Kathryn Harris, and Benjamin Davies*

ABSTRACT

Formal models of past human societies informed by archaeological research have a high potential for shaping some of the most topical current debates. Agent-based models, which emphasize how actions by individuals combine to produce global patterns, provide a convenient framework for developing quantitative models of historical social processes. However, being derived from computer science, the method remains largely specialized in archaeology. In this paper and the associated tutorial, we provide a jargon-free introduction to the technique, its potential and limits as well as its diverse applications in archaeology and beyond. We discuss the epistemological rationale of using computational modeling and simulation, classify types of models, and give an overview of the main concepts behind agent-based modeling.

Keywords: agent-based modelling, simulation, complex systems, complexity science, computational modelling, NetLogo

Modelos cuantitativos robustos de sociedades humanas en el pasado tienen el potencial de informar los temas de debate actual, particularmente modelos informados por estudios de arqueología. Modelos basados en sistemas multiagente proveen un marco práctico para explorar modelos cuantitativos de sociedades en el pasado. Aun así, al ser un método de informática no es aún bien establecido entre la mayoría de arqueólogos. En este artículo y el tutorial que lo acompaña, proveemos una introducción a estos métodos, libres de jerga técnica, su potencial y sus límites, y también las diversas aplicaciones en arqueología. Además, discutimos la epistemología de utilizar modelos computacionales y de simulación, clasificamos los tipos de modelos, y proveemos un resumen de los conceptos principales de los modelos multiagente.

Q2 **Palabras clave:**

Archaeology is uniquely positioned to use modern technology to understand the long trajectory of human history. The time-depth of archaeological inquiry allows researchers to investigate long-term and large-scale trends in human behavior, such as the evolution of social hierarchy (Crabtree et al. 2017), the changes in subsistence strategies (Powers and Lehmann 2014), or the resilience of human groups in the face of natural disasters (d'Alpoim Guedes et al. 2016). Just as material culture studies greatly benefited from the introduction of formal statistical tools, many current conceptual models— often grouped under the umbrella term of "theory building"—would benefit from a systematic and formal approach of computational modeling (d'Alpoim Guedes et al. 2016; Lake 2014).

One class of computational models that has been used increasingly in archaeology over the past two decades is agent-based modeling. Researchers studying archaeological systems worldwide have adopted this formal modeling technique to approach their research questions (e.g., Cegielski and Rogers 2016; Kohler 2012; Linde and Romanowska 2018; Madella et al. 2014; Perry et al. 2016; Rogers and Cegielski 2017; Romanowska 2015; Wurzer et al. 2015). Archaeologists use agent-based modeling to understand archaeological patterns across a range of temporal and spatial settings (e.g., Angourakis et al. 2014; Balbo et al. 2014; Morrison and Allen 2017; Perrault and Brantingham 2011; Premo 2015; Wren et al. 2014).

Our own experiences with simulation reflect this topical diversity, as we have used simulation to explore the development of the wine industry during the Bronze to Iron Age transition in Littoral France (Crabtree 2016), exchange practices and the development of hierarchy in the US Southwest (Crabtree 2015; Crabtree et al. 2017), Pleistocene hominin dispersals (Romanowska et al. 2017), and the formation of archaeological landscapes in Australia (Davies et al. 2015). Although the scenarios are simulated, the implications for research are real. For example, Lake (2014) discusses how formal reaction-diffusion models and long-term evolutionary models have already helped to move research forward on different aspects of human origins studies, noting that agent-based models have the potential to address many other areas of inquiry.

In this article and in the accompanying tutorial, we walk the reader through the process of building an agent-based simulation using

DOI:10.1017/aap.2019.6

an example of a model used to understand toolkit richness (Brantingham 2003). Lithics are commonly used to develop hypotheses about the behavior of their makers, but the causal relationship between any particular foraging strategy and the composition of lithic assemblages remains unclear. Using this example, we show how Brantingham's model enables researchers to use archaeological data (patterns in lithic assemblages) to identify behavior of people in the past (foraging strategies).

This is the first in a series of three articles and tutorials on agent-based modeling. A recent survey (Davies and Romanowska 2018) showed that the majority of archaeological modelers had to depend on self-teaching and peer support to acquire skills necessary to build their simulations. Although other agent-based modeling tutorials exist (e.g., Grimm and Railsback 2011) this series is unique in that it presents a case study of an archaeological system. We have also kept it largely jargon free with the intention of presenting the method to researchers with no previous experience in computational modeling.

In Part 1, we discuss the definition and function of agent-based models and introduce some key concepts in simulation. In the associated tutorial, we show how to build a simple hypothesis-testing agent-based model using a user-friendly, open-source, cross-platform simulation framework—NetLogo (Wilensky 1999)—and provide an outline of programming concepts. Part 2 (Davies et al. 2019) builds on this tutorial and incorporates realistic geographic information systems dataplanes to move the model from abstract to more realistic. Finally, in Part 3 (Crabtree et al. 2019), we demonstrate how agent-based models can be used for outreach to explain archaeological patterns to the public, whereas the associated tutorial will focus on analyzing the results.

WHY MODEL? AND IF SO, HOW TO MODEL?

Simulation has been hailed as the third leg of the scientific tripod: a qualitatively new scientific method falling between theoretical and empirical research (Axelrod 2006; Epstein 2006; Hartmann 1996; Kohler 2012). For example, Axelrod says:

> Simulation is a third way of doing science. Like deduction, it starts with a set of explicit assumptions. But unlike deduction, it does not prove theorems. Instead, a simulation generates data that can be analyzed inductively. Unlike typical induction, however, the simulated data comes from a *rigorously specified set of rules* rather than direct measurement of the real world [Axelrod 2006:95, emphasis added].

In fact, as Whitley (2016) points out, archaeologists commonly engage in what can be called "an analogue simulation." For instance, flintknapping experiments aimed at replicating past techniques or reenactments of medieval battles do not differ from computer simulation in the normative sense. Both start with a model—that is, a set of basic assumptions (e.g., knapping was performed using hands)—and are validated by comparing the simulation results (e.g., the shape and dimensions of a knapped stone tool) with the available data (e.g., archaeological artifacts).

Simulation is an established scientific tool, widely used across the natural and social sciences, as well as outside of academia, where it is commonly applied in industry, economics, and policy making (e.g., Abergel et al. 2014; Chattoe-Brown 2013; Davidsson and Verhagen 2013; Farmer and Foley 2009; Hammond 2015; Hartmann 1996; Mitchell 2009; Pyka and Werker 2009). Although there are many different types of simulation techniques, they share a number of characteristics.

At the core of every simulation is a model—a simplified representation of a real-world system, composed of entities and the relationships between them. In the philosophy of science, a model is defined simply as a set of assumptions (Godfrey-Smith 2003). Some models can be built using observations derived from experiments or from systems that can be directly studied. For example, a model of a preindustrial village may assume that people who lived in a single household were in some way related because of what we know from the observations of modern human groups and their family-forming behaviors. In other cases, the dependencies or the importance of different types of entities and processes are theorized about (known as "conceptual modeling"). If we were interested in how different degrees of social cohesion may lead over time to different habitation patterns, a conceptual model of social interaction would form the basis of the simulation.

Similar to a model, simulation is an artificially constructed and simplified representation of a real-world system with all relationships formally defined (known as "ontology"), but with the additional dimension of time (Hofman et al. 2011; Smith 2000). Therefore, simulation investigates changes occurring in a system over time and space as a result of external (exogenous) factors or the internal dynamics of the system.

The rapid rise of simulation in the 1990s came hand-in-hand with the recognition that many (if not most) real-world systems are complex; that is, governed by nonlinear processes, which escape more traditional, reductionist scientific methods focused on detailed description of system elements. Instead, it was the interaction between these elements that explained the system (Ladyman et al. 2013). For example, Schelling (1971) showed how a complex pattern (racially segregated neighborhoods) may emerge from relatively small sets of simple rules (slight incline of urban dwellers toward settling down among people similar to them), often defying expectations or common sense ("intolerance"—that is, having a low tolerance for neighbors of a different type—actually decreases segregation; see Chattoe-Brown 2013 for discussion).

This shift in research focus from the "detail" to the "whole," coupled with rapidly increasing computer power available to researchers, led to the development of complexity science. The core idea behind complexity science is the observation that simple interactions of multiple entities may lead to surprising global patterns and that this connection could not be easily deduced from studying each of the system's elements in isolation. This process, known as "emergence," is often summarized by the emblematic motto: the whole is greater than the sum of its parts (Mitchell 2009:13). The emergent properties of complex systems mean that simulation is the primary tool for studying them.

DIFFERENT TYPES OF MODELS

Hartmann (1996) recognized five main applications of simulation techniques used across different scientific disciplines: (1) simulations as a technique used to investigate the detailed dynamics of a system; (2) simulations as a heuristic tool for developing hypotheses, models, and theories; (3) simulations as a substitute for an experiment when performing it is unfeasible or not practical; (4) simulations as a tool for experimentalists used to support empirical experiments; and (5) simulations as a pedagogical tool allowing students and wider audiences to gain understanding of a process. Four of them are directly applicable in the context of traditional archaeological inquiry, whereas application (4) has been shown to be a valuable tool in archaeologically inspired anthropological research (Mesoudi and O'Brien 2008).

Simulation as a Technique

Many scientific disciplines can study systems by directly observing them; for example, microbiologists may observe an organism as it undergoes a change, and sociologists can ask a sample of the population to share their thoughts. However, in many cases, the actual dynamics of the system cannot be observed because the process takes too long (e.g., macroevolution of a species), the scale is too small (e.g., quantum physics), or the system does not exist anymore (e.g., past societies). In those cases, simulation is the appropriate research tool.

Simulation is a formal computational tool that reveals causal relationships between system entities and the evolution that the system undergoes. Equally, it allows the researchers to investigate the impact of specific events, such as a particular initial state or rare events, on the evolution of the system (known as "historical contingency" or "hysteresis"). Finally, exploratory models (Premo 2010), built on the minimum set of assumptions (known as "null models" or models from "first principles"), enable researchers to test their beliefs about the system's dynamics and the relative importance of different factors influencing it. We will see the value of this particular methodology on the archaeological example explored in the tutorial.

Simulation as a Theory-Building Tool

Simulation is commonly used as a tool for hypothesis development. Di Paolo and colleagues (2000) call simulation an "opaque thought experiment" because it represents a computer-based theoretical exercise in examining what-if scenarios (known as "subjunctive models"; David et al. 2013). The advantage of using a computer tool is that it can explore more complex, multiscalar and multivariate scenarios than can ever be reliably entertained in one's brain. In addition, as numerous examples have shown (e.g., Reynolds 1987; Resnick 1997; Schelling 1971), even simple models can unfold into surprising and counterintuitive patterns. As already mentioned, this phenomenon is known as "emergence" (Epstein 2006). The counterintuitive nature of such conclusions means that, by definition, they would be unlikely to be proposed as a result of conceptual modeling "in one's head." Second, modeling allows researchers to produce archaeologically testable predictions out of existing conceptual hypotheses, articulated in natural language. Premo describes the role of simulation as a "virtual lab" for "eliminat[ing] the plausible scenarios that are unlikely to have occurred, given observed characteristics of empirical data" (2006:108). As a result, instead of producing new theoretical models, which do not surpass the already existing ones, formal methods such as simulation build an increasingly strong framework based on "knowledge that over time is cumulative at both a theoretical and empirical level" (Neiman 1995:30).

Simulation as a Substitute for an Experiment

Simulation can replace an experiment in situations when practical constraints or ethical issues come into play (for example, if the investigated social process takes decades to evolve or if the experiment would subject the experimental population to prolonged hardship). Creating an artificial society and pestering it with climatic fluctuations, social upheavals, or natural disasters is a way of approaching such topical subjects as long-term social change, resilience, evolution, and impact of innovation without referring to modern and historical analogues and anecdotal evidence, or needing extensive Institutional Review Board/Ethics Committee oversight.

This process can be exemplified by comparing middle-range theory to simulation (Binford 1982; Kosso 1991; Premo 2007; Raab and Goodyear 1984). Ethnoarchaeologists study modern human groups because their behaviors and relationships can be directly observed. The material record generated by a modern group can then be compared with the archaeological record. If the two resemble each other, it is concluded that it is likely that these archaeological remains have been generated by processes similar to the ones driving the modern human group.

Similarly, an agent-based modeler constructs an artificial society governed by a strictly defined set of behavioral rules, making processes and causal relationships directly observable. The consequences of the simulated processes are then compared to the patterns in archaeological data. The aim of both types of research is to understand the dynamics of an accessible and, therefore, well-understood system well enough to be able to infer whether similar processes might have taken place in the past.

While modern scientists have critiqued the use of ethnoarchaeology (Fahlander 2004), cautioning that modern hunter-gatherers are not archaeological groups frozen in an early development stage (Kelly 2013), most archaeologists use ethnographic analogy explicitly or not. This is mostly done with appropriate caution, as researchers understand that all societies are dynamic and changing (Martelle Hayter 1994) and realize the limitations of their models. Agent-based modelers, who are usually archaeologists themselves, do the same.

Simulation as a Pedagogical Tool

Simulation can be used as an education tool. For example, Resnick (1997) used StarLogo (an early version of NetLogo) to explore the nonintuitive phenomena of emergence, decentralization feedback, self-organization, and criticality among high school students. Similarly, the interactive visualizations (known as "explorable explanations") of Hart and Case (2015) guide the participants through the process of simulating social phenomena, such as segregation, thus elucidating the real-life societal consequences of seemingly innocuous individual decisions. Because of its game-like properties, agent-based models are engaging and fun for specialists and non-specialists alike. They can, therefore, be an easy and cost-effective way to represent complex relationships and ideas and to let members of the general public, stakeholders, or students gain deep understanding of complex concepts by performing their own experiments. For example, van Havre (2018) has built a model of an archaeological landscape that archaeology students can use to explore how successful different sampling strategies will be in finding archaeology depending on the original distribution of artifacts. Part 3 will further explore these themes.

AGENT-BASED MODELING (ABM)

Agent-based modeling (ABM) is considered a "bottom up" simulation approach because it comprises heterogeneous individuals whose actions and interactions (both with each other and with the environment) lead to emergent population-level patterns. This method is often contrasted with Equation-based modeling (EBM), or the "top down" approach, where the individual actors are treated in aggregation as a homogenous population, whose characteristics are defined by a set of variables and whose interactions are described in equations (Macal and North 2010; Railsback and Grimm 2011).

In agent-based models, agent behavior is described in a set of rules (algorithms; Grimm et al. 2005; Ahrweiler and Gilbert 2005) and often modeled using probabilities (i.e., stochastically). Thus, ABM allows the researcher to model individual-driven mechanisms—such as cognitive processes, cultural transmission, and communication—and to introduce heterogeneity in the population, be it genetic/cultural diversity or even simple age and sex differences. As a result, the method enables a crossover between two levels of analysis: an individual perspective, which is very much at the heart of archaeological interest, and the global or population-level patterns representing the consequences of aggregated individual actions, which can be compared to the archaeological record. ABM provides a platform that facilitates the integration of the spatial environment as one of the primary model entities considerably (O'Sullivan and Perry 2014; see also Part 2 in the series), and the ability to construct models out of familiar entities (people, groups, households, etc.) rather than in the non-natural language of equations makes agent-based models easier to consult with and communicate to the archaeological (and public) audience (see also Part 3 in this series). Finally, the explicit focus on individuals and agency is a particularly important feature for archaeologists, who for decades have been concerned with the lack of "the individual" in the focus of archaeological practice. For example, Gamble and Porr argue that "the individual needs to be seen as the center of causality in order to understand why change and variation occur. It is individuals that make decisions and deal with choices" (2005:7). Computational modeling, and ABM in particular, holds great potential for addressing this issue as it provides a formal environment for testing the relationships between individual decisions, aggregated actions, and the consequences of these actions that are represented by the archaeological record.

Often, CRM archaeologists are tasked with finding explanations for the distribution of artifacts in their project area. These explanations are built on established theory, but many CRM projects do not allow for a full-scale analysis of an entire region. Many projects bisect sites, and archaeologists are forced to work within a down-sampled geographic area so that they do not explore beyond the project area. This can hamper their ability to draw meaningful conclusions from these partial datasets. ABM enables the use of partial data to test models and provide predictions. For example, if a researcher wants to understand the distribution of finds along an alluvial plain, writing a simple agent-based model simulating the transportation of artifacts might elucidate the patterns of deposition and could further predict where artifacts would be found. Thus, ABM can serve as a type of "behaviorally driven" predictive modeling that incorporates our knowledge of people's behaviors rather than one-to-one correlations. In doing so, it could counteract the common criticism of predictive models as being a self-fulfilling prophecy (Wheatley 2004).

For example, the members of the Village Ecodynamics Project, despite having a survey coverage of less than 20 percent (Kohler and Varien 2012:18), were able to create an agent-based model that examined the growth of population and the placement of households on the landscape. This model has been successful in testing hypotheses on the lifeways of Ancestral Pueblo people and has aided greatly in our understanding of the prehistory of the area despite the less than complete survey coverage. These themes will be explored further in the tutorial and in Part 2.

THE TUTORIAL

The tutorial based on the model by Brantingam (2003) that accompanies this article has been written with the general archaeological audience in mind. It does not assume any previous knowledge or skill of the reader and has been presented in a very informal and jargon-free style. In our demonstration of agent-based modeling, we will use an existing exploratory-type model framework to approach a new problem and apply it to a specific case study. Tutorial 1 (Supplementary Text 1) will focus on replicating the original model; Tutorial 2 (Supplementary Text in Part 2) will place it in a semirealistic landscape; finally, Tutorial 3 (Supplementary Text in Part 3) will explore how the results can be interpreted and communicated to stakeholders and the general public. In addition, we have prepared a document providing a more extended description of NetLogo structure and features that can be used as a glossary and for further help (Supplementary Text 2).

While simple, the model used in the tutorials is a way to start testing the validity of common archaeological assumptions regarding behavioral strategies presumably employed by people in the past. This model was also chosen because it reflects one of a few instances in which an agent-based model has been subjected to a number of published reevaluations (known as "replications"; Pop 2015; Oestmo et al. 2016). Each of them expands on the base model and leads to new insights, thereby showing how computational modeling facilitates the building up of our understanding in cumulative fashion (later models build upon and improve earlier ones rather than compete with them). For example, Pop (2015) revisited Brantingham's model, arguing that the original model did not fully appreciate the difference between the assemblage of a living forager and an archaeological assemblage, which might have undergone significant changes since the moment of being deposited. Although we acknowledge the usefulness of this extension and find his review helpful, our tutorial focuses on Brantingham's original model. A further test of good modeling practice would be applying additions from Pop (2015) and Oestmo and colleagues (2016) to the tutorial presented below, and we encourage students to do just that.

CONCLUSIONS

The benefits of using formal models for theory building and hypothesis testing in an academic environment are manifold. Although conceptual modeling will always have a place in science, building formal models, such as simulations, can enable a deeper understanding of complex processes that incorporate a temporal and spatial dimension. Simulations can also enable researchers to eliminate hypotheses that sound plausible yet do not concur with the archaeological record by thoroughly examining simulated output. Finally, simulations can produce predictions that can be tested on the ground with carefully focused empirical research. Consequently, field-based, lab-based, and computer-based research are not in competition but rather complement each other. We argue that using them together is the best way to bring us closer to understanding the lives of peoples in the past.

Supplementary material

For supplemental material accompanying this article, visit https://doi.org/10.1017/aap.2019.6

Supplementary Text 1. Tutorial 1: The Base Model

Supplementary Text 2: Summary of NetLogo

Acknowledgments

This manuscript and tutorial have been developed out of a number of ABM workshops and courses, and we extend our thanks to the many participants who have helped us to develop these materials over the years. The authors declare no conflicts of interest. IR received funding from the European Research Council (ERC) (EpNet Project, grant agreement n° ERC-2013-ADG340828). SC acknowledges support from an NSF Graduate Research Fellowship DGE-080667, an NSF GROW fellowship, and a Chateaubriand Fellowship. We thank two anonymous reviewers for comments that benefited the manuscript. We also thank Colin Wren and Andre Costopoulos for providing detailed consideration of the manuscript and tutorial materials.

Data Availability Statement

No data has been used in preparation of this manuscript. The software used in the tutorial is open access and open source (https://ccl.northwestern.edu/netlogo/).

REFERENCED CITED

Abergel, Frédéric, Hideaki Aoyama, Bikas K. Chakrabarti, Anirban Chakraborti, Asim Ghosh
2014 *Econophysics of Agent-Based Models*. Springer, New York.

Ahrweiler, Petra, and Nigel Gilbert
2005 Caffè Nero: The Evaluation of Social Simulation. *Journal of Artificial Societies and Social Simulation* 8(4):1–14.

Angourakis, Andreas, Bernardo Rondelli, Sebastian Stride, Xavier Rubio-Campillo, Andrea L. Balbo, Alexis Torrano, Verònica Martinez, Marco Madella, and Josep M. Gurt
2014 Land Use Patterns in Central Asia. Step 1: The Musical Chairs Model. *Journal of Archaeological Method and Theory* 21:405–425.

Axelrod, Robert
2006 Advancing the Art of Simulation in the Social Sciences. In *Handbook of Research on Nature-Inspired Computing for Economy and Management*, edited by Jean-Philippe Rennard, pp. 90–100. Idea Group, Hersey, Pennsylvania.

Balbo, A. L., B. Rondelli, C. Lancelotti, A. Torrano, M. Salpeteur, N. Lipovetzky, and M. Madella
2014 Agent-Based Simulation of Holocene Monsoon Precipitation Patterns and Hunter-Gatherer Population Dynamics in Semi-Arid Environments. *Journal of Archaeological Method and Theory* 21:426–446.

Binford, Lewis
1982 Objectivity-Explanation-Archaeology. In *Theory and Explanation in Archaeology*, edited by C. Renfrew, M. Rowlands, and B. Seagraves, pp. 125–138. Academic Press, New York.

Brantingham, P. Jeffrey
2003 A Neutral Model of Stone Raw Material Procurement. *American Antiquity* 68(3):487–509.

Cegielski, Wendy H., and J. Daniel Rogers
2016 Rethinking the Role of Agent-Based Modeling in Archaeology. *Journal of Anthropological Archaeology* 41:283–298. DOI:10.1016/j.jaa.2016.01.009.

Chattoe-Brown, Edmund
2013 Why Sociology Should Use Agent Based Modelling. *Sociological Research Online* 18(3):3.

Crabtree, Stefani A.
2015 Inferring Ancestral Pueblo Social Networks from Simulation in the Central Mesa Verde. *Journal of Archaeological Method and Theory* 22 (1):144–181. DOI:10.1007/s10816-014-9233-8.
2016 Simulating Littoral Trade: Modeling the Trade of Wine in the Bronze to Iron Age Transition in Southern France." *Land* 5(1):5. DOI:10.3390/land5010005.

Crabtree, Stefani A., R. Kyle Bocinksy, Paul L. Hooper, Susan C. Ryan, and Timothy A. Kohler
2017 How to Make a Polity (in the Central Mesa Verde Region). *American Antiquity* 82(1):71–95.

Crabtree, Stefani A., Kathryn Harris, Benjamin Davies, Iza Romanowska
2019 Outreach in Archaeology with Agent-based modeling. *Advances in Archaeological Practice* 7(2). DOI:10.1017/aap.2019.4

d'Alpoim Guedes, Jade A., Stefani A. Crabtree, R. Kyle Bocinsky, Timothy A. Kohler
2016 Twenty-First Century Approaches to Ancient Problems: Climate and Society. *PNAS* 113(51):14483-14491.

David, Nuno, Nuno Fachada, and Agostinho Rosa
2013 Verifying and Validating Simulations. In *Simulating Social Complexity: A Handbook*, edited by Bruce Edmonds and Ruth Meyer, pp. 135–172. Springer-Verlag, Berlin.

Davidsson, Paul, and Harko Verhagen
2013 Types of Simulation. In *Simulating Social Complexity: A Handbook*, edited by Bruce Edmonds and Ruth Meyer, pp. 23–38. Springer-Verlag, Berlin.

Davies, Benjamin, and Iza Romanowska
2018 An Emergent Community? Agent-Based Modelers in Archaeology. *The SAA Archaeological Record* 18(2):27–32.

Davies, Benjamin, Simon J. Holdaway, and Patricia C. Fanning
2015 Modelling the Palimpsest: An Exploratory Agent-Based Model of Surface Archaeological Deposit Formation in a Fluvial Arid Australian Landscape. *The Holocene* October 19:1–14.

Davies, Benjamin, Iza Romanowska, Kathryn Harris, Stefani A. Crabtree
2019 Combining Geographic Information Systems and Agent-Based Models in Archaeology. *Advances in Archaeological Practice* 7(2). DOI:10.1017/aap.2019.5

Di Paolo, Ezequiel A., Jason Noble, and Seth Bullock
2000 Simulation Models as Opaque Thought Experiments. In *Artificial Life VII:*

Proceedings of the Seventh International Conference on Artificial Life, edited by N. Packard and S. Rasmussen M. A. Bedau, J. S. McCaskill, pp. 497–506. MIT Press, Cambridge, Massachusetts.

Epstein, Joshua M.
2006 Agent-Based Computational Models and Generative Social Science. In *Generative Social Science: Studies in Agent-Based Computational Modeling*, edited by Joshua M. Epstein, pp. 41–60. Princeton University Press, Princeton, New Jersey.

Fahlander, Fredrik
2004 Archaeology and Anthropology–Brothers in Arms? On Analogies in 21st-Century Archaeology. In Material Culture and Other Things: Post-Disciplinary Studies in the 21st Century, edited by Fredrik Fahlander and Terje Oestigaard, pp. 185–212. Bricoleur Press, Lindome, Sweden.

Farmer, J. Doyne, and Duncan Foley
2009 The Economy Needs Agent-Based Modeling. *Nature* 460:685–686.

Gamble, Clive, and Martin Porr
2005 From Empty Spaces to Lived Lives. Exploring the Individual in the Palaeolithic. In *The Hominid Individual in Context: Archaeological Investigations of Lower and Middle Palaeolithic Landscapes, Locales, and Artefacts*, edited by Clive Gamble and Martin Porr, pp. 1–12. Routledge, Oxon, United Kingdom.

Godfrey-Smith, Peter
2003 *Theory and Reality: An Introduction to the Philosophy of Science*. Science and Its Conceptual Foundations series. The University of Chicago Press, Chicago.

Grimm, Volker, Eloy Revilla, Uta Berger, Florian Jeltsch, Wolf M. Mooij, Steven F. Railsback, Hans-Hermann Thulke, Jacob Weiner, Thorsten Wiegand, and Donald L. DeAngelis
2005 Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology. *Science* 310(5750):987–91. DOI:10.1126/science.1116681.

Grimm, Volker, and Stephen F. Railsback
2011 Agent-Based and Individual-Based Modeling: A Practical Introduction. Princeton University Press, Princeton, New Jersey.

Hammond, Ross A.
2015 Considerations and Best Practices in Agent-Based Modeling to Inform Policy. In *Assessment of Agent-Based Models to Inform Tobacco Policy*. Institute of Medicine, National Academy of Sciences Press, pp. A1–A27.

Hart, Vi, and Nicky Case
2015 Parable of the Polygons: A Playable Post on the Shape of Society, http://ncase.me/polygons/, accessed March 15, 2019.

Hartmann, Stephan
1996 The World as a Process: Simulations in the Natural and Social Sciences. In *Modelling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, edited by Rainer Hegselmann, Ulrich Mueller, and Klaus G. Troitzsch, pp. 77–100. Kluwer, Dordrecht, Netherlands.

Hofmann, Marco, Julia Palii, and Goran Mihelcic
2011 Epistemic and Normative Aspects of Ontologies in Modelling and Simulation. *Journal of Simulation* 5(3):135–146.

Kelly, Robert L.
2013 *The Lifeways of Hunter-Gatherers: The Foraging Spectrum*. 2nd edition. Cambridge University Press, Cambridge.

Kohler, Timothy A.
2012 Complex Systems and Archaeology. In *Archaeological Theory Today*, edited by Ian Hodder, pp. 93–123. Polity Press, Cambridge.

Kohler, Timothy A., and Mark D. Varien
2012 *Emergence and Collapse of Early Villages: Models of Central Mesa Verde Archaeology*. University of California Press, Berkeley.

Kosso, Peter
1991 Middle-Range Theory as Hermeneutics. *American Antiquity* 56(4):621–627.

Ladyman, James, James Lambert, and Karoline Wiesner
2013 What Is a Complex System? *European Journal for Philosophy of Science* 3(1):33–67.

Lake, Mark W.
2014 Trends in Archaeological Simulation. *Journal of Archaeological Method and Theory* 21:258–287.

Linde, Lennart, and Iza Romanowska
2018 The-ABM-in-Archaeology-Bibliography. bit.ly/ABMBiblio. DOI:10.5281/zenodo.1343332

Macal, Charles M., and Michael J. North
2010 Tutorial on Agent-Based Modelling and Simulation. *Journal of Simulation* 4(3):151–162.

Madella, Marco, Bernardo Rondelli, Carla Lancelotti, Andrea Balbo, and Debora Zurro
2014 Introduction to Simulating the Past. *Journal of Archaeological Method and Theory* 21:251–257.

Martelle Hayter, Holly
1994 Hunter-Gatherers and the Ethnographic Analogy: Theoretical Perspectives. *Totem: The University of Western Ontario Journal of Anthropology* 1(1): Article 8.

Mesoudi, Alex, and Michael J. O'Brien
2008 The Cultural Transmission of Great Basin Projectile-Point Technology II: An Agent-Based Computer Simulation. *American Antiquity* 73(4):627–644.

Mitchell, Melanie
2009 *Complexity. A Guided Tour*. Oxford University Press, Oxford.

Morrison, Alex E., and Melinda S. Allen
2017 Agent-Based Modelling, Molluscan Population Dynamics, and Archaeomalacology. *Quaternary International* 427A:170–183.

Neiman, Fraser D.
1995 Stylistic Variation in Evolutionary Perspective: Inferences from Decorative Diversity and Interassemblage Distance in Illinois Woodland Ceramic Assemblages. *American Antiquity* 60(1):7–36.

Oestmo, Simen, Marco A. Janssen, and Curtis W. Marean
2016 Testing Brantingham's Neutral Model: The Effect of Spatial Clustering on Stone Raw Material Procurement. In *Simulating Prehistoric and Ancient Worlds*, edited by Juan Antonio Barceló and Florencia Del Castillo, pp. 175–188. Springer, Cham, Switzerland.

O'Sullivan, David, and George Perry
2013 *Spatial Simulation: Exploring Pattern and Process*. Wiley-Blackwell, Chichester, United Kingdom.

Perreault, Charles, and P. Jeffrey Brantingham
2011 Mobility-Driven Cultural Transmission along the Forager-Collector Continuum. *Journal of Anthropological Archaeology* 30(1):62–68.

Perry, George L. W., John Wainwright, Thomas R. Etherington, and Janet M. Wilmshurst
2016 Experimental Simulation: Using Generative Modeling and Paleoecological Data to Understand Human-Environment Interactions. *Frontiers in Ecology and Evolution* 4(109):1–14.

Pop, Cornel M.
2015 Simulating Lithic Raw Material Variability in Archaeological Contexts: A Re-evaluation and Revision of Brantingham's Neutral Model. *Journal of Archaeological Method and Theory* 23:1127. doi:10.1007/s10816-015-9262-y.

Powers, S. T., and L. Lehmann
2014 An Evolutionary Model Explaining the Neolithic Transition from Egalitarianism to Leadership and Despotism. *Proceedings of the Royal Society B* 281(1791):20141349–20141349. DOI:10.1098/rspb.2014.1349.

Premo, Luke S.
2006 Agent-Based Models as Behavioral Laboratories for Evolutionary Anthropological Research. *Arizona Anthropologist* 17:91–113.
2007 Exploratory Agent-Based Models: Towards an Experimental Ethnoarchaeology. In *Digital Discovery: Exploring New Frontiers in Human Heritage*, edited by J. T. Clark and E. M. Hagemeister, pp. 91–113. Archaeolingua, Budapest.
2010 Equifinality and Explanation: The Role of Agent-based Modeling in Postpositivist Archaeology. In *Simulating Change. Archaeology into the Twenty-First Century*, edited by Andre Costopoulos and Mark Lake, pp. 28–37. University of Utah Press, Salt Lake City.
2015 Mobility and Cultural Diversity in Central-Place Foragers: Implications for the Emergence of Modern Human Behavior. In *Learning Strategies and Cultural Evolution during the Palaeolithic*, edited by Alex Mesoudi and Kenichi Aoki, pp. 45–65. Springer Japan, Tokyo.

Pyka, Andreas, and Claudia Werker
2009 The Methodology of Simulation Models: Chances and Risks. *Journal of Artificial Societies and Social Simulation* 12(4): paper number 1. Q10

Raab, L. Mark, and Albert C. Goodyear
1984 Middle-Range Theory in Archaeology: A Critical Review of Origins and Applications. *American Antiquity* 49(2):255–268.

Railsback, Steven F., and Volker Grimm
2011 *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, Princeton, New Jersey.

Resnick, Mitchel
1997 Turtles, Termites, and Traffic Jams. *Explorations in Massively Parallel Microworlds*. MIT Press, Cambridge, Massachusetts.

Reynolds, Craig W.
1987 Flocks, Herds and Schools: A Distributed Behavioral Model. *ACM SIGGRAPH Computer Graphics* 21(4):25–34.

Rogers, J. Daniel, and Wendy H. Cegielski
2017 Opinion: Building a Better Past with the Help of Agent-Based Modeling. *Proceedings of the National Academy of Sciences* 114(49):12841–12844. DOI:10.1073/pnas.1718277114.

Romanowska, Iza
2015 So You Think You Can Model ? A Guide to Building and Evaluating Archaeological Simulation Models of Dispersals. *Human Biology* 87(3):169–192.

Romanowska, Iza, Clive Gamble, Seth Bullock, and Fraser Sturt
2017 Dispersal and the Movius Line: Testing the Effect of Dispersal on Population Density through Simulation. *Quaternary International* 431:53–63. DOI: 10.1016/j.quaint.2016.01.016

Schelling, Thomas C.
1971 Dynamic Models of Segregation. *Journal of Mathematical Sociology* 1: 143–186.

Smith, Roger D.
Q11 2000 *Simulation. Encyclopedia of Computer Science*. Nature Publishing Group.

van Havre, Grégoire
Q12 2018 ArcheoBM, https://github.com/gvanhavre/ArcheoBM. DOI: 10.5281/zenodo.1342668

Wheatley, David
2004 Making Space for an Archaeology of Place. *Internet Archaeology* 15. DOI:10.11141/ia.15.10. Q13

Whitley, Thomas G
2016 Archaeological Simulation and the Testing Paradigm. In *Uncertainty and Sensitivity Analysis in Archaeological Computational Modeling*, edited by Marieka Brouwer Burg, J. H. M. Peeters, and William A. Lovis, pp. 131–156. Springer International Publishing, Switzerland.

Wilensky, Uri
1999 *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, Illinois, https://ccl.northwestern.edu/netlogo/, accessed March 15, 2019.

Wren, Colin D., Julian Z. Xue, Andre Costopoulos, and Ariane Burke
2014 The Role of Spatial Foresight in Models of Hominin Dispersal. *Journal of Human Evolution* 69:70–78.

Wurzer, Gabriel, Kerstin Kowarik, and Hans Reschreiter (editors).
2015 Agent-Based Modeling and Simulation in Archaeology. Springer International Publishing, Switzerland.

AUTHOR INFORMATION

Iza Romanowska ■ Barcelona Supercomputing Center, Carrer de Jordi Girona, 29-31, 08034 Barcelona, Spain (iza.romanowska@bsc.es) https://orcid.org/0000-0002-9487-2111

Stefani A. Crabtree ■ Department of Anthropology, The Pennsylvania State University, 410 Carpenter Building, University Park, PA 16803 (sac376@psu.edu); Center for Research and Interdisciplinarity, 8bis rRue Charles V, 75004 Paris, France; Crow Canyon Archaeological Center, [illegible] C R K, Cortez, CO 81321 https://orcid.org/0000-0001-8585-8943

Kathryn Harris ■ Science & Technology Policy Fellow, The American Association for the Advancement of Science and The American Geophysical Union, 2000 Florida Avenue. NW, Washington, DC 20009 (kaharris@wsu.edu)

Benjamin Davies ■ Department of Anthropology, University of Utah, 260 S. Central Campus Drive, Room 4625, Salt Lake City, U[illegible]84[illegible]12 Q1

Tutorial 1: The base model

This tutorial is based on the Neutral Model of Stone Raw Material Procurement developed by Jeffrey Brantingham. Brantingham's (2003) highly influential article was the first application of agent-based modeling to the topic of procurement, curation and spatial distribution of raw material used for lithics production in prehistory. Brantingham sought to develop from first principles a model of raw material curation, that is, a base model stripped of any behavioral assumptions that could be then compared to the lithics assemblages found at archaeological sites. Essentially, he asked how these assemblages would look if the processes leading to their creation were random (Brantingham 2003: 490).

The main premise behind the model is to establish the pattern of assemblage variability under neutral conditions of no behavioural biases (for example, without a preference for any particular raw material type and no specific type of mobility). In the simulation, a single agent-forager follows a random walk (see O'Sullivan and Perry 2013, ch 4) through a uniform landscape dotted with raw material sources. The agent collects raw material indiscriminately whenever s/he comes across a source. When a raw material source is encountered their toolkit is filled up until it contains 100 pieces and the agent continues their journey. At every step one piece is deposited on the current grid cell. There is no specific foraging/movement strategy or any preference for a particular type of raw material.

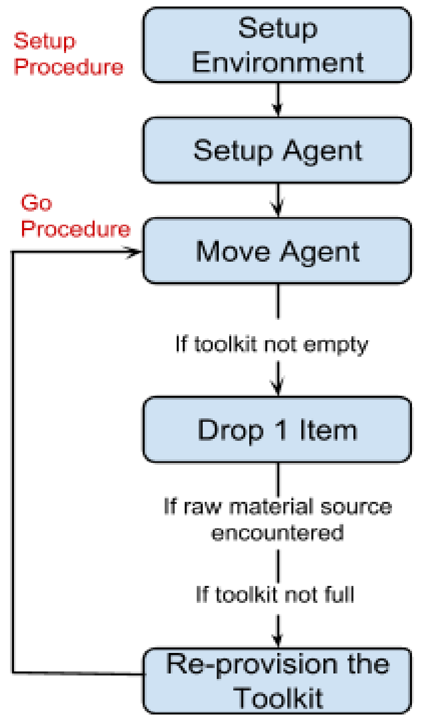


Figure 1.0.1. Flowchart of the Neutral Model. Adapted and simplified from Brantingham 2003, fig. 5.

The outline of the simulated processes is given in Fig. 1. In the setup phase a 500x500 cells world is seeded with 5000 unique raw material sources. Each raw material type is present at only one cell and their distribution is random. In the second phase the agent is initialised with an empty toolkit. Once the setup phase is completed the time clock is started. In each time step the agent either moves to one of the 8 neighbouring cells chosen at random or stays put. If the toolkit is not empty, the agent drops one randomly chosen item. If the cell the agent is on is a raw material source the agent reprovisions the toolkit unless it is full. The cycle move-drop-reprovision repeats until the simulation is stopped. The richness of the toolkit (the number of unique raw materials) as well as the variability of assemblages composed of items dropped by the agent are recorded throughout the simulation.

In this tutorial we will try to replicate Brantingham's model as accurately as possible. It will take approximately two hours to complete. It has been built and tested in NetLogo 6.0.1. For installing instruction and more in-depth explanations see Romanowska et al. 2019 Supplementary Information B.

1.1 NetLogo Interface

The NetLogo interface (Fig. 2) consists of three tabs: **Interface**, **Info** and **Code**. Let's look at them in turn. The Interface window consists of: the **View Panel** for watching the simulation, a few buttons along the top of the window with settings and the **Command Center** towards the bottom of the window, which you can use to directly alter or inspect any element of the simulation.

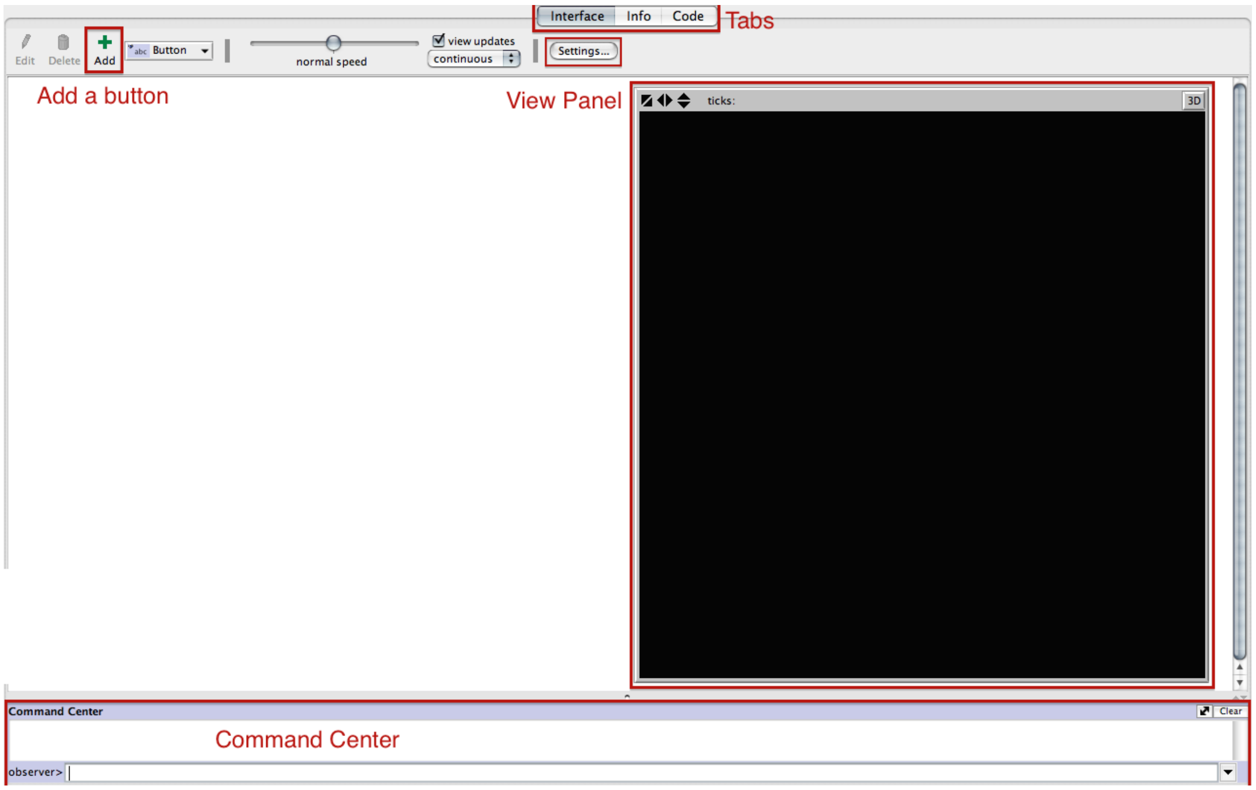
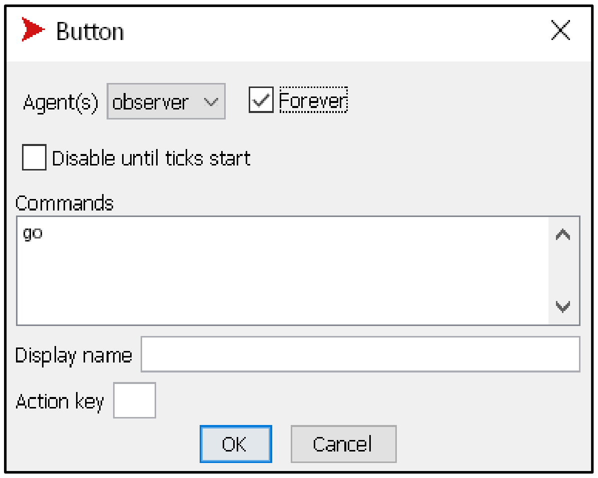


Figure 1.2. The Netlogo Interface.

We are first going to change the size of our simulation window to accommodate a much larger view. Click on the 'Settings' button in the top menu to adjust the view panel. First change the 'Location of origin:' to 'Corner' and choose 'Bottom left' from the drop down menu. We need a much larger area than the default so replace the values in max-pxcor and max-pycor to 499 (the coordinates of the first patch start at 0 0 so setting maximum x and y to 499 gives a 500x500 square). However, this means that if we keep the size of **patches** (grid squares) as large as it is now the screen will be enormous. Change the 'Patch size' to 1.0 and hit 'Apply'. You might have noticed the two tick boxes 'World wraps horizontally' and 'World wraps vertically'. If ticked they provide continuity between the edges of the screen, i.e., if the agent moves right while standing on the right-most patch it will appear on the left-most patch; this is known as a torus world as it doesn't have edges. Check that both tick boxes are ticked and hit 'OK'. This will get you back to the main "home" image. If you do not like the size of the view panel, right click anywhere on it, choose 'Select' from the dropdown menu and drag one of the corners until the size is ok - note that this only changes the size of the patches, their number (500 by 500) remains the same.

1.2 The setup and the go procedures

The backbone of almost all NetLogo simulations are two procedures: ‘setup’ and ‘go’. The **setup procedure** is used to initialise the simulation, i.e., to create the starting population of agents and to their environment. The **go procedure** is the main simulation loop where in each time step the agents and the environment undergo a series of actions.



```
go
```

Figure 1.3. The button window.

Click on the ‘Add’ button and then click anywhere on the white space. A dialogue box (Fig. 3) should pop up. Write `setup` in the ‘Commands’ box and click OK. Follow the same steps to create a second button and write `go` in the ‘Commands’ box. This time also tick the ‘Forever’ box. This means that this action will be repeated until the simulation ends.

You can see that the text on both buttons has instantly turned red, indicating an error - this is because we have not yet defined what we mean by ‘setup’ and ‘go’ within the code. Let’s move to the Code Tab to fix it. The Code Tab consists mostly of a white text box, the **code box**, and a few buttons towards the top, which we will inspect in a moment. We will start with setting up the two procedures. Type the following in the code box:

```
to setup
end

to go
end
```

The words `to` and `end` delimit all NetLogo procedures. If you now click on the ‘Procedures’ button at the top of the screen, you will find that ‘setup’ and ‘go’ are already there. To the left is the debugger button ‘Check’ - if you click on it, it will check if the basic syntax of the code is correct.

1.3 The setup procedure

Logo - the language of NetLogo - was developed to resemble a natural language as much as possible, which means that it is very easy to understand the code. It was also developed with educational goals in mind (read: teaching kids), which means that it is equally easy to write. We will start with setting up the environment by asking each patch to set a number of variables: colour, whether they are a raw material source patch or not and the list of dropped lithics it contains. In addition, we will use the standard NetLogo functions that ensure that every time we hit the setup button the remains of the previous runs are removed. All **commands** directed at turtles and patches are initiated by the word `ask` and enclosed in square brackets `[]`, here we will make use of them for the first time. Type inside the setup procedure (i.e., between `to setup` and `end`), so that it looks like this:

```
to setup
      clear-all
      ask patches[
            set pcolor white
            set source? false
            set assemblage []
      ]
      reset-ticks
end
```

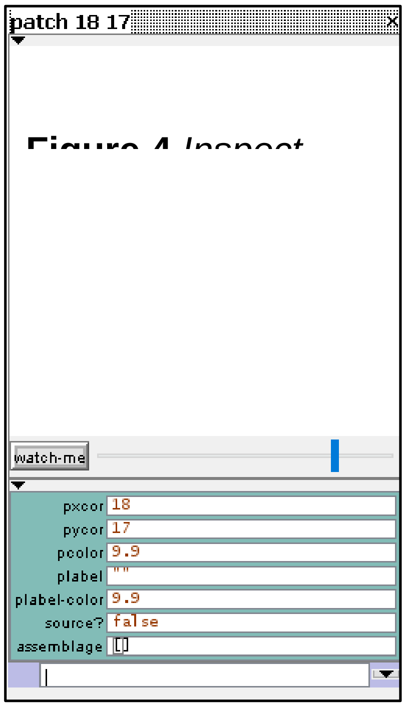
In the first line we use the `clear-all` primitive to wipe clean any remnants of the previous runs. Similarly the last line of the setup procedure is always dedicated to resetting the time counter using the `reset-ticks` primitive. **Primitive** is NetLogo jargon for an in-built function, that is defined in the NetLogo library. Check out the [NetLogo dictionary](https://ccl.northwestern.edu/netlogo/docs/dictionary.html) (https://ccl.northwestern.edu/netlogo/docs/dictionary.html) for a full list of primitives. Coming back to the code, we set up the environment by asking patches: 1) to set their color to white, 2) to set their status as a source of raw material (we will ask them all to be a no-source for now), and 3) to start a list in which we will record whether any artefacts have been dropped on this particular patch during a run. The flow of the program is governed by brackets and it is very easy to lose track of how many you have opened already. To avoid confusion once you open a bracket, immediately close it and write the code inside. The indentation does not matter, but it makes the code easier to read so we recommend using it. Hit the 'Check' button to see if there are any errors.

And there are. There always are. The message: 'Nothing named SOURCE? has been defined.' appeared at the top of the screen. Indeed, we tried to set the variable `source?` to false without defining it first. Congratulations on seeing your first code error!

Variables are often used to describe the characteristics of agents, patches and the world. For example, an agent can have age, gender, energy, cultural marker or any other feature relevant to the model. These variables may change throughout the simulation run (e.g., age, energy) or remain static (e.g., gender, cultural marker). There are two types of variables in NetLogo's syntax: 1) **global variables**, used throughout the code, which must be listed at the beginning of the code or defined using Interface items (we will come back to this), and 2) **local variables**, defined by the `let` primitive, which are only valid within one procedure. We will see the use of local variables later on, but the `source?` is a global one - it is a characteristic of all the patches. We will set its value in the setup procedure and it will remain unchanged throughout the run. The same applies to the `assemblage` list so we will also define it as a global variable. Type at the beginning of the code (before "`to setup`"):

```
patches-own [ source? assemblage ]
```

And hit the 'Check' button; there should be no errors. If you now go to the Interface tab and click on the 'setup' button, you will see that the screen went white. Right click anywhere within the view panel and choose 'inspect patch ...' from the drop down menu. It will show you (Fig. 4) a list of patch variables, including some of the built-in ones such as the x and y coordinates and the patch color, but also the two we have defined ourselves: `source?` and `assemblage`.



```
pxcor 18
pycor 17
pcolor 9.9
plabel ""
plabel-color 9.9
source? false
assemblage []
```

Figure 1.4. Inspect patch window.

The white screen is not very exciting so let's set up the patches that are raw material sources. Because we do not want ALL the patches to be a source we will use the `n-of` (*number of*) primitive. Each raw material source needs to be unique so we will give them a different id. Type inside the setup command, after the first command block but before the `reset-ticks`.

```
let r 1
ask n-of 5000 patches [
             set source? true
             set material_type r
             set r r + 1
             set pcolor black
  ]
```

The first thing we do here is to define a local variable `r` and set it up as 1. We then ask 5000 patches 1) to change their `source?` status to true, 2) to set the `material_type` as the unique id `r`, 3) to then add 1 to `r` so that the next patch gets the next (`r+1`) id, and finally 4) we will change their colour to black to see where the raw material source patches are. Hit the 'Check' button. A familiar error message appears. But this time you know what to do! Add `material_type` in the list of patch variables (`patches-own`) at the beginning of the code:

```
patches-own [ source? assemblage material_type ]
```

Move to the Interface tab and hit the 'setup' button. You can now inspect one of the source patches by right clicking on it and choosing 'Inspect Patch ...' from the drop down menu. You will see in the pop-up window that the value of `source?` is `true`.

We have now created the environment, but not the agents. We actually only need one but we should give her/him quite a few variables such as the initial location and looks as well as a maximum number of lithics s/he can carry and a list to keep track of them. Type after the patches setup procedures but before `reset-ticks`:

```
  crt 1 [
    setxy random max-pxcor random max-pycor
    set color red
    set size 10
    set shape "person"
    set max_carry 100
    set toolkit []
  ]
```

`crt` stands for ‘create agents’, in our case, one agent. Inside the brackets we set their initial position to a random patch (i.e., with x and y coordinates between 0 and the current maximum - `max-pxcor` and `max-pycor`) and add a few variables: color (notice that `color` applies to agents and `pcolor` to patches), size and shape. We also initiate a list of all the raw material types the agent carries in its toolkit and set up how much s/he can carry at any one time. Just like `source?` or `assemblage` the `toolkit` list and the `max_carry` variable are global variables (you can hit ‘Check’ if you want to see the error message). However, this time they apply to agents not patches so we need to make an turtle-specific variables lists. Write the following line at the beginning of the code:

```
turtles-own [ toolkit max_carry ]
```

Here, we finally drop the bombshell - in NetLogo jargon agents are called **turtles**. This is the legacy of being originally developed as an educational tool for kids. It makes for a fun code development and all NetLogo developers sooner or later learn to love their turtles.

If you now go to the Interface tab and hit ‘setup’ you should find a red human-shaped agent on one of the patches (Fig. 5). If you keep on pressing the setup button you will see that each time the simulation is reset, the agent and source patches are initialised at a different (random) location.

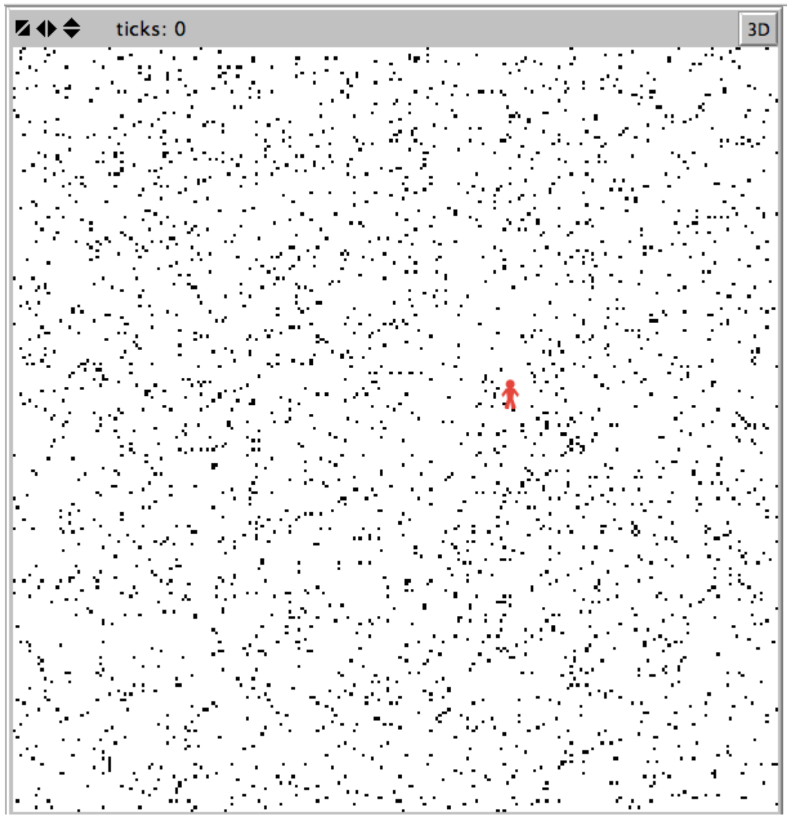


Figure 1.5. The initialized simulation.

1.4 The go procedure

With the setup complete let's move on to the `go` procedure, i.e., the main body of the simulation which will be repeated until you click on the 'go' button again. The first thing we want the agent to do is to move around the landscape. We have established that in each time step the agent will move to one patch in its Moore neighborhood (that is, to the S, N, E, W or SE, SW, NE, NW of the current location) or stay put. You can put it in a more mathematical terms as: the agent has a 1 in 9 chance of staying where it is or moving to any one of the surrounding patches. In order to code the agent's movement, we start the command with the keyword `ask turtles` and enclose a list of functions, e.g., the `move-to` primitive in brackets. Type the following inside the go procedure:

```
to go
      ask turtles [
            if random 9 > 0 [ move-to one-of neighbors ]
      ]
      tick
end
```

Let's look at the code a bit closer. We ask all turtles (in our case there is only one) that if a random integer (whole number) between 0 and 8 is higher than 0 then the agent should perform the functions that are inside the brackets: `move-to one-of neighbors`. If it's 0 then nothing happens - the agent stays put. `move-to`, `one-of` and `neighbors` are all NetLogo primitives and we encourage you to check them in the NetLogo dictionary (https://ccl.northwestern.edu/netlogo/docs/dictionary.html). We also added the `tick` primitive at the end of the go procedure that moves the time counter by one. Go to the Interface tab and click first on 'setup' and then on the 'go' button. You should see the red agent running around the world like crazy. Use to speed slider at the top of the window to slow it down a bit. You should be able to see that the agent moves by one patch as the time counter underneath the speed slider is ticking forward. Click on the 'go' button to stop the simulation.

The next thing to do according to the flow diagram of the model (Fig. 1) is for the agent to drop one item at each step whenever its toolbox is not empty. We will use an if-loop to check whether there is anything to drop in the toolkit and then update both the assemblage of the patch and the agent's toolkit. Write the following code inside the `go` using another '`ask turtles`' command (after the final closing bracket of the movement function but before `tick`):

```
ask turtles[

   if length toolkit > 0 [
     let i random length toolkit
     ask patch-here [
            set  assemblage  fput  (item  i  [  toolkit  ]  of  myself)
     assemblage
     ]
     set toolkit remove-item i toolkit
   ]
]
```

Let's go through the code line by line. Like in the previous code snippet we use the 'if' conditional loop. This time we will only perform the functions enclosed by the first set of square brackets if the length of the toolkit list is more than zero, i.e., the toolkit is not empty. If that is the case, we choose at random an item with an index `i` from the toolbox. `i` is an index number between zero and the number of items currently present in the toolkit (`length toolkit`) which denotes its position in the list (as in: first, third, tenth etc.). In the next line we ask the patch on which the agent stands (note the special primitive `patch-here`) to add the item (using the `fput` list primitive) `i` of the `toolkit` of the turtle asking, referred to with the primitives `of myself`, to the patch's list of dropped items - `assemblage`. We then remove the same item from the agent's toolkit.

If you run the simulation and inspect the agent (click on the 'go' button to pause the simulation, then right click on the agent and choose 'inspect turtle 0'), you will notice that despite all the coding we have just done the agent's toolkit remains empty. That's not what we want! But it is easy to understand why. We have no code for picking up raw material! In short, the agent never got a chance to pick anything up! This cycle of writing small modular code bits and then checking the simulation (by just running it and by inspecting its elements) is the best way of writing code. If you try to write everything at once, chances are there will be errors and it will be much harder to find them.

We recognise patches which contain a raw material source by their variable `source?` set as `true`. Whenever the agent comes across one of them we want it to restock the raw material. We will again use an if-loop to check whether the patch is a 'source patch' and if so fill up the agent's toolkit until it is full (using the `while` loop). Type the following code to make a new `ask turtles` command inside the `go` procedure (after dropping procedure but before `tick`):

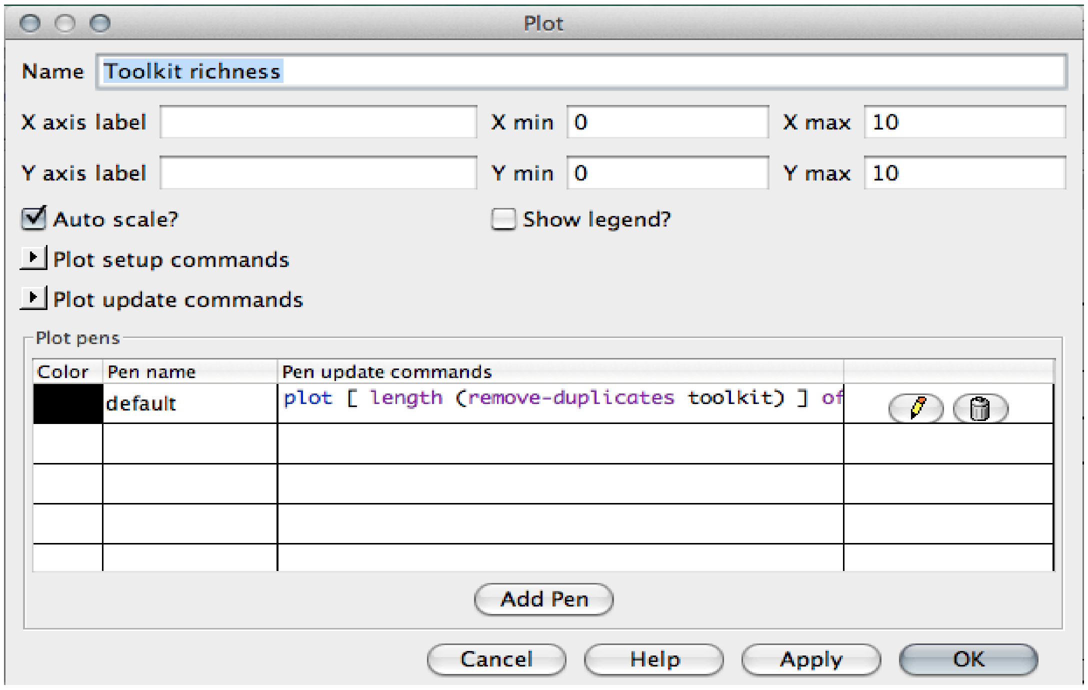
```
ask turtles[

    if [ source? ] of patch-here = true [
        let raw_material [ material_type ] of patch-here
        while [ length toolkit < max_carry ] [
            set toolkit fput raw_material toolkit
        ]
    ]
]
```

In the first line we ascertain that the patch is indeed a source patch. If it is not, the block of code enclosed in the square brackets will be ignored and the program will move to the next statement (in this case: `tick`). Do you remember that each raw material source has a unique ID? We need that ID so that we know what type of raw material is added to the toolkit. The `let` statement sets up a local variable `raw_material` to the same value as the ID (`material_type`) of the patch the agent is standing on (`patch-here`). The variable raw_material is local, meaning it is only recognised inside the `ask turtles` brackets. If you try to use is anywhere else, our favourite error message ('Nothing named raw_material has been defined') would pop up. In the next line a 'while-loop' adds the `raw_material` to the `toolkit` list until the maximum capacity of the agent (`max_carry`) is reached. Note the difference between the if- and while-loops here. If the given condition is fulfilled (e.g., the patch-here is a source or a randomly drawn number is higher than zero) an if-loop will perform the actions defined inside its brackets once. A while-loop will keep on repeating them until the given condition is reached (e.g., the toolkit length is equal to the maximum capacity of the agent).

Go to the Interface tab and run the simulation. Inspect the turtle - if you keep the inspect window open during the run you should be able to see how the toolkit changes every time the agent comes across a raw material source patch.

It is a bit difficult to see the raw material toolkit in the small box, so let's create a plot that will show the changes in the frequencies of different raw material types present in the toolkit. Click on the drop-down list next to the 'Add' button at the top of the Interface tab and choose 'Plot'. Click anywhere on the white area outside of the view panel. A new pop-up window will appear. Write in the 'Name' box: 'Toolkit richness'. The 'Plot pens' box is where we specify what should be plotted. The default value of `plot count turtles` counts the number of agents and in many cases is very useful but since we only have one agent it does not make much sense. Delete it and type:

```
plot [ length (remove-duplicates toolkit) ] of turtle 0
```



Name: Toolkit richness
X axis label:
X min: 0
X max: 10
Y axis label:
Y min: 0
Y max: 10
Auto scale? ✓
Show legend? ☐

Color	Pen name	Pen update commands
■	default	plot [length (remove-duplicates toolkit)] of

Figure 1.6. The plot interface.

This will plot the size (`length`) of the toolkit list once all duplicates are removed, that is the number of unique raw material sources present in the agent's toolkit. Every turtle has an in-built unique number assigned to it when it is created. Since we only have one agent, its number is zero. We specify this (`of turtle 0`) because otherwise the plotting function would not know which agent's toolkit to plot. Run the simulation (slow it down). If you compare your plot with figure 7 in Brantingham's paper (2003) you will notice that they strongly resemble each other.

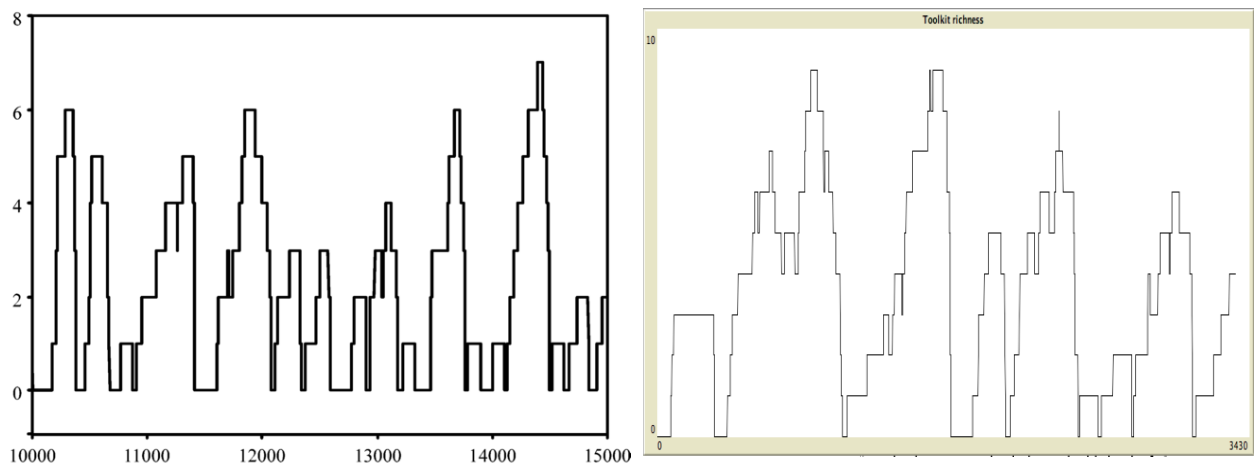
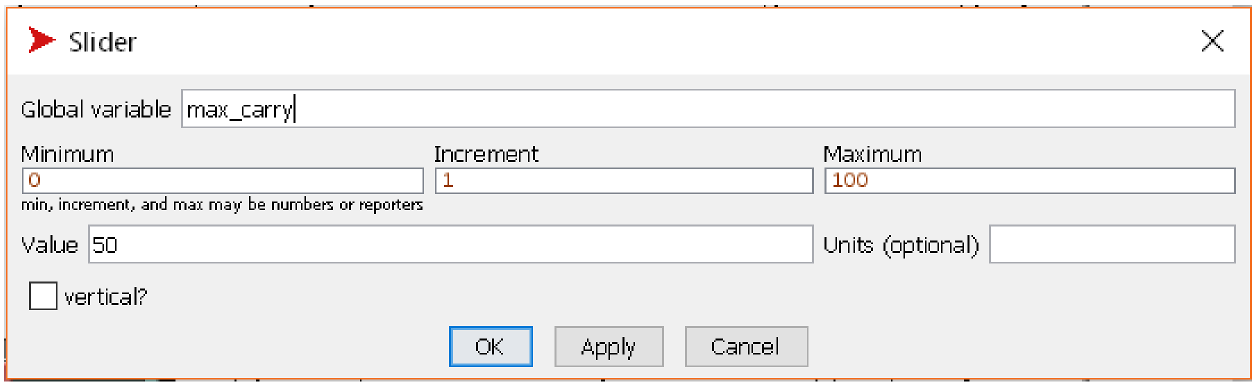


Figure 1.7. The plots, Brantingham versus our model.

Congratulations you have successfully replicated a famous model!

To finish off, we will slightly extend the neutral model. Brantingham notes that the dynamics of the simulation will change if the maximum size of the toolkit that the agent can carry is altered. We will set up a slider to help running a series of experiments that will test it. As mentioned before, global variables can be defined at the beginning of the code in the variables lists (`patches-own`, `turtles-own`). However, you can also define them in the Interface tab by using a slider, a chooser or a box. Go to the interface and click on the drop-down list next to the 'Add' button and choose 'Slider'. Then click anywhere on the white field outside of the view panel. A pop-up window will appear. Type `max_carry` in the 'Global variable' box at the very top. You can leave the rest of the boxes unchanged and hit OK. You immediately get an error message saying that 'There is already a global variable called MAX_CARRY'.


Slider

Global variable max_carry

Minimum 0

Increment 1

Maximum 100

min, increment, and max may be numbers or reporters

Value 50

Units (optional)

vertical?

Figure 1.8. The slider variable interface.

This is because we have already defined `max_carry` at the beginning of the code in the `turtles_own` list. Now we have two sources of a variable called `max_carry` (the slider and the `turtles-own` list) and NetLogo does not know which one to use. Simply delete `max_carry` from the `turtles_own` list. It should look like this now:

```
turtles-own [ toolkit ]
```

Also, go to the `setup` procedure and delete the line `set max_carry 100` in the command used to create the agent. If you forget to remove it from the setup procedure what will happen is NetLogo will read from the slider the value of `max_carry` (say 80) and then start executing the setup procedure, but when it comes across the line in which you `set max_carry 100` it will overwrite the value set on the slider (80) to the one in the code (100). This is a common error because it does not produce an error message (since you have not done anything illegal according to the NetLogo syntax). Hit the 'Check' button and there should be no more errors. You can now use the slider to vary how much the agent can carry in each run. Although you can use the slider during the run, it is discouraged as the results will not be replicable. Instead change the value before each run and compare the output of the plot. Have you noticed how the peaks of the toolkit richness are lower and less frequent if the `max_carry` is set to a lower number?

This is the end of Tutorial 1. In the next one (Davies et al. 2019), we will move our model into a real landscape generated from GIS layers. Tutorial 3 (Crabtree et al. 2019) will focus on how to better collect the simulation output (just looking at a plot is not the greatest method) and how to automate running the experiments (so you don't spend days moving sliders).

To cite this document:
Romanowska, I., S. Crabtree, B. Davies, and K. Harris. 2019. "Agent-based Modeling for Archaeologists. A step-by-step guide for using agent-based modeling in archaeological research (Part I of III)." Advances in Archaeological Practice 7 (2).

References cited

Brantingham, P. Jeffrey
2003 A Neutral Model of Stone Raw Material Procurement. *American Antiquity* 68(3): 487–509.

Crabtree, Stefani, Kathryn Harris, Benjamin Davies, Iza Romanowska
2019 "Outreach in Archaeology with Agent-based modeling. A step-by-step guide for using agent-based modeling in archaeological research (Part III of III)." Advances in Archaeological Practice 7 (2).

Davies, Benjamin, Iza Romanowska, Kathryn Harris, Stefani Crabtree
2019 "Combining Geographic Information Systems and Agent-Based Models in Archaeology: A step-by-step guide for using agent-based modeling in archaeological research (Part II of III)." Advances in Archaeological Practice 7 (2).

O'Sullivan, David, and George Perry
2013 *Spatial Simulation: Exploring Pattern and Process*. Wiley-Blackwell, Chichester.

```
; _________________ GLOBAL VARIABLES _________________
turtles-own [ toolkit ]
patches-own [ source? assemblage material_type ]

; _________________ TO SETUP _________________

to setup
  ;;; The setup procedure is run only once at the beginning of each experiment.
  clear-all                      ; remove any residuals of previous experiments

; _________________ 1. Environment Setup _________________
  ;;; setup patches
  ask patches[
    set pcolor white
    set source? false            ; initially all cells are set as having no raw material
    set assemblage []            ; start a list of items that the agent dropped at this location
  ]                              ; (this would be an equivalent to an archaeological 'find spot')

   ;;; setup patches with raw material
   let r 1
   ask n-of 5000 patches [       ; 5000 random patches become raw material sources
      set source? true           ; set the variable source? as true
      set material_type r        ; each will have a different raw material type
      set r r + 1                ; marked as a number between 1 and 5000
      set pcolor black
  ]
 ; _________________ 2. Agent Setup _________________
   ;;; create the agent and place him on a random patch, set color, size and shape
   crt 1 [
    setxy random max-pxcor random max-pycor
    set color red
    set size 10
    set shape "person"
    set toolkit []               ; start a list of items that the agent carries

  ]
   reset-ticks                   ; reset the time counter
end                              ; end of the SETUP procedure
```

```
; _________________ TO GO _________________

to go
;;; agent procedure: 1. move to one of 4 neighbouring cells; 2. drop an item from the toolkit if not
empty 3. reprovision the toolkit if a raw material source patch is encountered

  ask turtles [
    ;_________________ 1. Move _________________
     if random 9 > 0 [                     ; if a randomly drawn no between 0 - 9 is higher than 0
       move-to one-of neighbors          ; move to any one of the 8 neighbouring patches
     ]                                     ; otherwise (it is 0) don't do anything
  ]

    ;_________________ 2. Drop a random item from the toolkit _________________
  ask turtles [
     if length toolkit > 0 [               ; if the toolkit is not empty
       let i random length toolkit         ; determine which item (i) will be dropped

       ask patch-here [                    ; add item i to the 'assemblage' of the patch
         set assemblage fput (item i [ toolkit ] of myself) assemblage
     ]

     set toolkit remove-item i toolkit    ; remove the item (i) from the toolkit
   ]
 ]
     ;_________________ 3. Reprovision if on a source patch  _________________
  ask turtles [
     if [ source? ] of patch-here = true [         ; if you come across a patch with raw material
         let raw_material [ material_type ] of patch-here          ; determine the type of raw material
         while [ length toolkit < max_carry ] [    ; while you still have capacity to carry more...
           set toolkit fput raw_material toolkit     ; keep on adding that raw material to your toolkit
       ]
     ]
   ]

  tick                                              ; time + 1

end                                                 ; end of the GO procedure
```

Summary of NetLogo

This document provides a general overview of NetLogo structure and syntax and can be used as a glossary alongside the tutorials (Romanowska et al. 2019; Crabtree et al. 2019; Davies et al. 2019) or as a quick ‘cheat-sheet’.

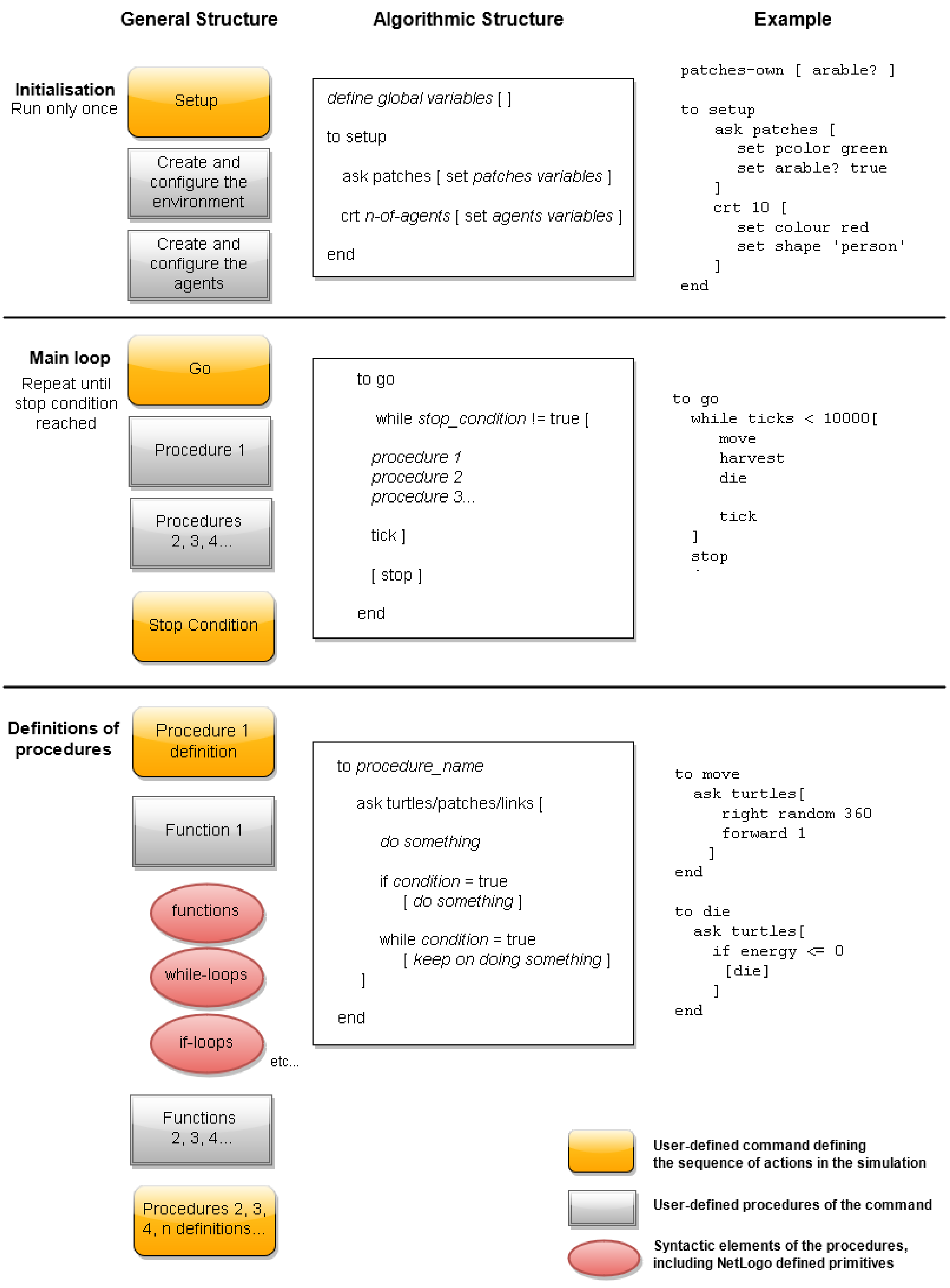
NetLogo (Wilensky 1999) is a user-friendly simulation platform commonly used for agent-based modeling in social and natural sciences. It is based on the Logo language originally designed as an educational tool for teaching programming to kids, making it a ‘low threshold, high ceiling’ platform. NetLogo combines ease of use and quick development with high level capacity and a wide suite of built-in tools such as visualizations, automated scenario running, etc. There are a number of other ABM platforms (RePast, Mason, AnyLogic; overview: Abar et al. 2017) and simulations can also be built using any of the general use programming languages (Python, C++, Java). NetLogo is by far the most dominant ABM platform in Archaeology (Davies and Romanowska 2018) and is very popular in social sciences and ecology. However, like every tool, NetLogo has some limitations which we discuss at the end of the document.

1.1. Installation

NetLogo can be downloaded from https://ccl.northwestern.edu/netlogo/. It is available for Windows, Mac OS X and Linux. The installation is a simple “point and click” and in most cases is unproblematic. In case of any issues it is worth consulting the FAQ of the NetLogo User Manual (https://ccl.northwestern.edu/netlogo/requirements.html).

1.2. Code building blocks

The four main entities in Netlogo are the agents (‘turtles’), the grid squares (‘patches’), the connections between agents (‘links’) and the observer. The observer governs the simulation flow, for example, by progressing the time counter or scheduling the order of actions. The building blocks in NetLogo consist of commands and reporters. Built-in commands are called ‘primitives,’ user defined ones are called ‘procedures’ and ‘reporters’. The latter calculate a value and then report it. Most NetLogo simulations are composed of two main procedures: **`to setup`** and **`to go`**. In the `setup` procedure the world and the agents are initialized and it is executed only once at the beginning of a run. Setup may include loading up the GIS raster, creating the initial population of agents and giving them a set of characteristics (e.g., sex, age, profession). The `go` procedure is the core of a model and consists of a list of actions (often themselves procedures) that are repeated at each time step of the simulation until the stop condition is reached. All procedures come from the observer environment and are defined using the `to` and `end` keywords. If not specific to the observer commands apply to one of NetLogo entities: turtles, patches, or links. Therefore, they are always enclosed between square brackets following the keywords: `ask` *`turtles/patches/links`*. Figure 1 shows the common structure of NetLogo code with a list of procedures inside the `go` and the ‘`ask turtles/patches`’, etc. command inside each of the procedure’s definitions. It is important not to have ‘`ask turtles`’ inside another ‘`ask turtles`’, e.g., do not use `ask turtles [move]` if there is another ‘`ask turtles`’ inside the ‘`move`’ procedure - it will inevitably give you an error.


General Structure
Algorithmic Structure
Example
Initialisation
Run only once
Setup
Create and configure the environment
Create and configure the agents
Main loop
Repeat until stop condition reached
Go
Procedure 1
Procedures 2, 3, 4...
Stop Condition
Definitions of procedures
Procedure 1 definition
Function 1
functions
while-loops
if-loops
etc...
Functions 2, 3, 4...
Procedures 2, 3, 4, n definitions...
User-defined command defining the sequence of actions in the simulation
User-defined procedures of the command
Syntactic elements of the procedures, including NetLogo defined primitives

```
define global variables [ ]

to setup

  ask patches [ set patches variables ]

  crt n-of-agents [ set agents variables ]

end
```

```
patches-own [ arable? ]

to setup
    ask patches [
       set pcolor green
       set arable? true
    ]
    crt 10 [
       set colour red
       set shape 'person'
    ]
end
```

```
    to go

      while stop_condition != true [

      procedure 1
      procedure 2
      procedure 3...

      tick ]

      [ stop ]

    end
```

```
to go
  while ticks < 10000[
      move
      harvest
      die

      tick
  ]
  stop
```

```
  to procedure_name

    ask turtles/patches/links [

       do something

       if condition = true
          [ do something ]

       while condition = true
          [ keep on doing something ]
    ]

  end
```

```
to move
  ask turtles[
      right random 360
      forward 1
     ]
end

to die
  ask turtles[
     if energy <= 0
       [die]
     ]
end
```

1.3. Variables

There are two types of variables in NetLogo: global and local variables. Global variables are defined at the beginning of the code and ascribed to one of the NetLogo entities: the observer (`globals []`), agents (`turtles-own[]`), grid squares (`patches-own[]`) or links (`links-own[]`). In the Interface tab, button, slider, and chooser also define global variables. Global variables can be used throughout the code and accessed via any procedure. Usually, we use global variables for things that are set at the beginning of the run and do not change during it, such as the initial number of agents, or the capacity of agents' backpack.

In contrast, local variables, defined by the `let` primitive, can only be accessed from within the procedure it has been defined in. For example, the following line of code:

```
let old_turtles turtles with [age > 50]
```

defines `old_turtles` as all turtles whose attribute `age` is greater than 50. We can then ask all 'old turtles' to do a specific task that is different from the one performed by youngsters. However, in the next time step, the list of 'old turtles' will change as some of them would have died while others would have reached the required age. Thus, local variables are dynamic and changeable throughout the run.

1.4. Control statements

Similar to all programming languages, NetLogo supports three main types of loops: if-loops, while-loops and for-loops (the latter will be discussed in section 1.4, Lists). If the conditional statement of an if-loop evaluates to 'true' a list of actions specified in the square brackets is performed. For example, the following command:

```
ask turtles[
      if energy <= 0 [die]
]
```

can be read as 'ask each turtle: if your energy is equal or lower than zero, please die'. In contrast, the while loop keeps on repeating the set of actions enclosed by the square brackets until the specified condition is reached. For example, this command:

```
ask turtles [
      while energy > 0 [move]
]
```

can be read as 'ask each turtle to move as long as their energy is greater than zero'. It is important to ensure that the while loop can (and will) reach the condition, otherwise the code will be run forever (or rather, until your computer's memory limit is reached - this is known as the 'stack overflow'). For example, this will give the while-loop a closure:

```
ask turtles [
      while energy > 0 [move]
      set energy energy - 1
]
```

1.5. Lists

A list is an object that stores multiple pieces of information. Lists are useful for keeping track of groups of things where group membership might change over time; for example, a forager's toolkit. A list can be initiated using the `set` primitive:

```
set example_list [10 20 30 40 50]
```

Alternatively, an agentset can be used to construct a list through the primitive `of`, e.g.:

```
set turtle_ages [ age ] of turtles
```

In this case the list `turtle_ages` contains the values of the `age` variable of each turtles.

Although lists are immutable, new lists can be created on the basis of existing lists, again using the `set` primitive.

```
set example_list replace-item 2 example_list 25
```

The '`2`' in the example above represents the index of the list, i.e. the position of the item which is to be replaced with the value '`25`'. Similar, to most programming languages the indexing in NetLogo starts with zero, i.e., the index of the first element of the list is 0, the second: 1, the third: 2, etc.

The for-loop mentioned above allows the user to perform an action on each element of the list. If we would like to inspect the content of the `example_list` after it was altered we can use:

```
foreach example_list show
```

and the result should be (if you executed the previous example):

```
10
25
30
40
50
```

We can also use `foreach` in conjunction with the arrow reporter to iterate through a list. For example

```
(foreach example_list example_list

  [ [a b] -> show word "the sum is: " (a + b) ])
```

Which tells NetLogo to use example_list as input and add corresponding integers together (e.g., 10+10) and report that in the command line.

The result should be (if you executed the previous example):

```
20
50
60
80
100
```

The examples above are intended as a general reference only. We will guide the reader through the process of building a simulation in NetLogo and discuss the code elements in a more comprehensive manner in the tutorial.

1.6. NetLogo resources

There are many freely-available learning resources for ABM and NetLogo on the Internet. NetLogo documentation (NetLogo 2018), which includes tutorials, a programming guide and a full dictionary of NetLogo primitives is usually the first port of call for any technical inquiries. However, there are many other ABM- and NetLogo- dedicated websites, blogs, code repositories and user groups. **simulatingcomplexity.wordpress.com** run by the authors of this tutorial is a specialized blog on archaeological applications of computational modeling and complexity science. The Special Interest Group in Complex Systems Simulation holds an annual beginner workshop in NetLogo as well as sessions and roundtables at the Computer Application and Quantitative Methods in Archaeology - CAA conference (https://caa-international.org/), while the European Social Simulation Association - ESSA (http://www.essa.eu.org/) organizes a week-long summer school in social simulation as a satellite to its annual conference. In addition, the Complex Systems Society annual conference - CSS (https://cssociety.org/) usually features at least one session (satellite) dedicated to archaeology. There are other, domain specific training courses available.

There are a number of university-level textbooks which use NetLogo to show the principles of complex systems modeling in ecology (Railsback and Grimm 2011), geography (O'Sullivan and Perry 2014), social science (Gilbert and Troitzsch 2005; Miller and Page 2007) and economy (Hamill and Gilbert 2016).

Not all simulations need to be written from scratch. NetLogo comes with a large and growing library of models. With over two hundred agent-based models the Models Library (accessible through the user interface of NetLogo) contains many examples, which, although developed for other disciplines from mathematics and physics to ecology and transport, could be adapted to archaeological research. For instance, epidemiological models simulating the spread of a disease in human populations under different conditions share many characteristics with theoretical models of the diffusion of innovations. In addition, many authors working on archaeological case studies share their model's code after publication in the OpenABM (https://www.comses.net/) repository and increasingly also on GitHub (https://github.com/).

1.7. When to switch to a different platform

Given how easy it is to develop agent-based models in NetLogo and how popular it is among social scientists and in other disciplines, you may ask yourself: 'why would anyone use any other software?'

Like every tool, NetLogo has its tradeoffs and it is important to know its 'weaknesses' in order to minimize their impact on one's research. The three main limitations of NetLogo are: 1. The high level of the programming language, 2. The lack of some of the standard software development tools, and 3. The performance.

The high level of the programming language is the exact reason why it is not difficult to learn to code in NetLogo. You 'ask turtles' to 'forward 1'. It's easy to write, it's easy to read, but what exactly does it do? The simplicity of coding comes at a price of not having a full control over the code. For example, it is all too common to (erroneously) assume that the primitive 'forward 1' makes all turtles move to the next patch ahead. It looks so obvious that hardly anyone checks the documentation to see whether this is actually the case (it is not). But these checks should be done. NetLogo documentation is extensive, thorough and openly available. It is important to constantly test the code during development to be sure it is doing exactly what one thinks it is doing. Unfortunately, NetLogo lacks some of the standard software development tools, for example, for testing code. This is a weakness that we hope will be soon addressed by its developers, but in the meantime it is crucial that the modeler runs a wide variety of tests to minimize the risk of code errors. It is commonly done using the ‘print’ primitive (e.g., by asking one of the turtles to print out the results of calculations it is performing and checking that they are within the expected ranges) or by running ‘special’ scenarios (e.g., with no agents or only one agent to check that the algorithms function correctly). Also, the ‘inspect’ and ‘watch’ functions as well as the ‘pen-down’ primitive may come very handy in checking the code.

Finally, the performance issue. NetLogo is not a tool for highly optimized simulations. Also, it has no code parallelization capacity or any other support for HPC (High Performance Computing) so even a supercomputer may not be able to save you. If you need to run your simulation 300,000 times, NetLogo is not going to cut it. This limits, for example, the parameter ranges one can test or the number of runs performed to deal with the stochasticity of the model. It is common that the modeler finds themself (usually about 6 months before the end of a project) with a model that takes 20 minutes to run from start to finish. If they need to run scenarios with 4 parameters, and each one has 10 values that need to be tested, plus each run needs to be repeated 10 times because of stochasticity then we are talking about three and a half years before the results come in and let’s hope that the code does not have a bug and needs to be run again. However, there are tools that can help optimize the code, in particular, the profiler (https://simulatingcomplexity.wordpress.com/2015/03/23/netlogo-profiler/)

On the other hand, it is much easier to implement a working simulation developed in NetLogo in one of the fast programming languages, such as Python or Java, than to develop it there from scratch. Thus, it is not unusual for archaeologists to start their ABM adventure with NetLogo and then move to other programming languages and simulation platforms while still using NetLogo to prototype their models. As one of the reviewers of this paper noted: “I would not make the argument that learning Netlogo will make you an ABM expert over all platforms. It is a good start though.” We hope that this set of tutorials will help you kick start such a journey.

To cite this document:
Romanowska, Iza, Stefani Crabtree, Benjamin Davies, and Kathryn Harris
2019 “Agent-based Modeling for Archaeologists. A step-by-step guide for using agent-based modeling in archaeological research (Part I of III).” Advances in Archaeological Practice 7 (2).

References cited

Abar, Sameera, Georgios K. Theodoropoulos, Pierre Lemarinier, and Gregory M.P. O’Hare
2017 “Agent Based Modeling and Simulation Tools: A Review of the State-of-Art Software.” *Computer Science Review* 24: 13-33. Elsevier Inc.: 13–33. doi:10.1016/j.cosrev.2017.03.001.

Crabtree, Stefani, Kathryn Harris, Benjamin Davies, Iza Romanowska
2019 “Outreach in Archaeology with Agent-based modeling. A step-by-step guide for using agent-based modeling in archaeological research (Part III of III).” Advances in Archaeological Practice 7 (2).

Davies, Benjamin, Iza Romanowska, Kathryn Harris, Stefani Crabtree
2019 “Combining Geographic Information Systems and Agent-Based Models in Archaeology: A step-by-step guide for using agent-based modeling in archaeological research (Part II of III).” Advances in Archaeological Practice 7 (2).

Davies, Benjamin, and Iza Romanowska

2018 “An Emergent Community? Agent Based Modelers in Archaeology.” *The SAA Archaeological Record* 18(2): 27–32.

Gilbert, Nigel G., and Klaus G. Troitzsch
2005 *Simulation for the Social Scientist*. Open University Press, Maidenhead.

Hamill, Lynne, and Nigel Gilbert
2016 *Agent-based modeling in economics*. Wiley, Chichester.

Miller, John H., and Scott E. Page
2007 *Complexity in Social Worlds*. Princeton University Press, Princeton.

O’Sullivan, David, and George Perry
2013 *Spatial Simulation: Exploring Pattern and Process*. Wiley-Blackwell, Chichester.

OpenABM
2014 Open Agent Based Modeling Consortium. A node in the CoMSES Network. https://www.openabm.org

Railsback, Steven F., and Volker Grimm
2011 *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, Princeton.

Romanowska, Iza, Stefani Crabtree, Benjamin Davies, and Kathryn Harris
2019 “Agent-based Modeling for Archaeologists. A step-by-step guide for using agent-based modeling in archaeological research (Part I of III).” Advances in Archaeological Practice 7 (2).

Wilensky, Uri
1999 “NetLogo.” Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston. https://ccl.northwestern.edu/netlogo/, accessed February 2, 2016.