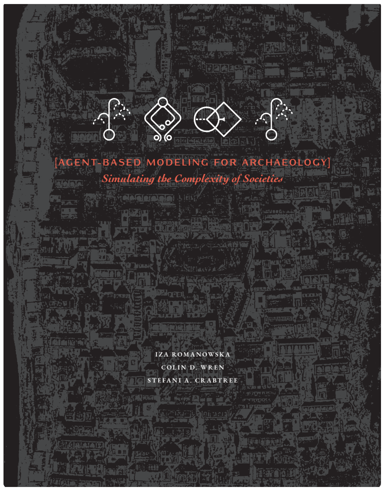
PLEASE NOTE:

The contents of this open-access PDF are excerpted from the following textbook, which is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License:

Romanowska, I., C. D. Wren, and S. A. Crabtree. 2021. *Agent-Based Modeling for Archaeology: Simulating the Complexity of Societies.* Santa Fe, NM: SFI Press.

This and other components, as well as a complete electronic copy of the book, can be freely downloaded at https://santafeinstitute.github.io/ABMA



REGARDING COLOR:

The color figures in this open-access version of *Agent-Based Modeling for Archaeology* have been adapted to improve the accessibility of the book for readers with different types of color-blindness. This often results in more complex color-related aspects of the code than are outlined within the code blocks of the chapters. As such, the colors that appear on your screen will differ from those of our included figures. See the "Making Colorblind-Friendly ABMs" section of the Appendix to learn more about improving model accessibility.

SFI PRESS
THE SANTA FE INSTITUTE PRESS 1399 Hyde Park Road, Santa Fe, New Mexico 87501 | sfipress@santafe.edu

GLOSSARY

agentset: in NetLogo, an unordered set of agents of one type (e.g., turtles or patches).

algorithm: a sequence of instructions given to a computer.

arbitrary values: setting a model's variables not to real-world values but to numbers that reflect ratios between the variables, e.g., X is double Y, X = 10 if Y = 100.

artificial data: model output that can be compared to real archaeological data.

average degree: the average number of edges per node in a network.

average shortest path length: the mean number of steps between all pairs of nodes in a network.

balanced reciprocal exchange network (BRN): type of exchange where individuals may lend to others but expect to be paid back by the individual who borrowed from them.

benchmarking: a computer-science term for measuring the computational speed of an algorithm. Often done before and after a change to the programming for comparison, with many repetitions, and/or across different computers.

betweenness centrality of a network: the average betweenness centrality of all nodes of a network.

betweenness centrality of a node: the number of shortest paths between all pairs of nodes that cross through the node.

Boolean: a variable with only two possible values (e.g., TRUE or FALSE, 0 or 1, heads or tails). Also called a binary variable.

bounded rationality: the idea, first proposed by Simon (1956), that humans will often settle on a "good-enough" solution rather than exert time and resources to find the optimal one.

breed: In NetLogo, a predefined agentset. Can be used with several primitives instead of turtles.

burn-in/warm-up period: the opening phase of the model's run when the dynamics can be affected by the initialization.

calibration: using an external data source to reduce parameter ranges to a narrower and more plausible set.

chaotic system: a type of dynamic system that is highly sensitive to the initial conditions. Chaotic systems, for example weather, are unpredictable beyond the horizon of predictability determined by our ability to measure the initial conditions.

coefficient of variation: the variance divided by the mean of a variable.

conformist bias: probability of adopting a cultural trait proportional to its popularity in the population. The inverse is known as anti-conformist bias.

cultural diffusion: concerns the spread of an idea, such as an innovation, a decorative pattern, or a belief.

cultural trait: different content transmitted between individuals: a skill, reference, piece of information, strategy, or particular behavior.

culture: in this context, any information passed between individuals via social transmission (Mesoudi 2011, p. 2).

data structure: the format and organization of variables and parameters. Common data structures in NetLogo are integers, floats (decimal numbers), Booleans (true/false), lists, agents, and agentsets.

demic diffusion: concerns the physical movement of people, such as in the out-of-Africa dispersal.

diet-breadth model: a model concerned with an individual's acquisition of specific resources within a larger set of resources. Agent actions are determined by the value and the cost associated with acquiring those resources.

digital elevation model (DEM): a datafile with topography of the mapped area recorded as elevation values.

directed edge: one-way connection, such as a predator eating a prey (trophic networks).

edge: the relationships between the entities in a network (the lines).

emergence: a property of model dynamics where an unexpected effect arises from the aggregated interactions of the individual entities, i.e., when the whole becomes more than the sum of its parts.

entity: any object in a model.

equifinality: a principle stating that a given state may be reached through many different routes.

Erdős-Rényi network: a network, also known as a graph in network science, created by repeatedly connecting two random nodes with an edge until the specified number of edges is reached.

exchange: the giving of one object and receiving of another, usually in the same type, and especially goods, services, or ideas

experiment design: the final phase of the modeling process in which the modeler experiments on the modeled world by running the simulation in different configurations (scenarios).

fitness: a measure of an agent's "quality," usually relative to other agents, such as likelihood of reproduction or survivability in a biological system, or presence of preferred traits or behaviors that bring an economic or social benefit in a cultural system.

game theory: a mathematical framework studying models of interaction between competing agents, usually rational decision makers.

generalized reciprocal exchange network (GRN): type of exchange where individuals share their surplus with their kin when asked.

Hamming distance: metric used to compare two sequences of the same length, which sums the number of differences in each position.

hard-coded: entering a parameter value directly into the model's code rather than as an adjustable parameter.

list: an ordered sequence of values. They can be of any type (integer, string, another list, etc.) and can repeat multiple times.

local variables: can be thought of as temporary attributes that are created on the fly.

model: a simplified, often abstract, representation of a system.

model-based bias: probability of adopting a cultural trait based on the characteristics of individuals who carry this trait.

model selection: an idea borrowed from statistics whereby one looks to an assemblage of potential models and identifies the one that best fits the data.

models from first principles: simulate how the world would look under the most basic of assumptions.

modular code development: an approach to coding where code is written incrementally and within many independent procedures.

Moore neighborhood: consists of a central cell and the immediately adjacent eight cells (including diagonals). Note NetLogo's "neighbors" primitive excludes the central cell.

multilevel models: models that explicitly code dynamics that occur at multiple scales (e.g., individual and household).

mutation: a random change of an element in the genome. Here used to denote a random change in the value of a cultural trait.

Nash equilibrium: a set of agent choices such that no agent can benefit by altering its choice, assuming that the choices of the other agents remain unchanged.

negative reciprocity: type of exchange where an individual takes advantage of another without any intention to repay the debt.

network: a set of entities (nodes) connected through a set of relationships (edges or links). Edges between the nodes could represent any type of relationship, from kinship and friendship to trade flows and material culture similarity.

network science: branch of science concerned with relational data and its collection, management, analysis, interpretation, and presentation (Brandes et al. 2013).

node: the entities in a network (the dots).

observer: an "agent" that builds and controls all other simulation objects.

ontology: the full representation of the model's world, including names and definitions of entities and their categories and properties, the relationships between entities, and the rules of behavior.

open science: a philosophy of research transparency in science that advocates for open-access publishing, open-data sharing, and transparent methods.

open source/open code: a pillar of the open-science movement that encourages the publication of all scientific code, including analysis scripts.

optimal foraging theory: a model of resource acquisition behavior that maximizes the net benefit (i.e., after costs have been accounted for).

overfitting: a model is overfitted when it fits the data too closely, thus reflecting noise rather than significant trends.

parameter: a constant used to describe the modeled world and a particular scenario, such as the initial number of agents, population growth rate, or carrying capacity of a patch.

parameter sweep: systematically testing a range of parameter values, i.e., scenarios to determine the range of outcomes.

parameterization: the process of choosing individual or ranges of parameter values.

parsimony: the idea that a model should have as few details as possible while still representing the modeled phenomenon. Often defined as the Occam's Razor principle.

phase space: often multidimensional space of all possible states of the model.

point of equilibrium: a moment in the simulation run when the model stabilizes and nothing new can be learned from it.

pooling: type of exchange where resources are shared among all participants.

primitive: built-in procedure or variable name.

procedure: consists of all code enclosed between {to} and {end}.

pseudocode: a simplified notation of the structure of the code.

random drift: random change in the frequency of traits; often occurs in small populations.

replication: a reimplementation of a model by another researcher to verify the original model's correctness.

reporter: a type of procedure that calculates a value and reports it back.

resilience: the capacity of a system to recover from a perturbation.

roulette wheel algorithm: an algorithm that selects among multiple outcomes, each with a particular probability. As in casino roulette, where getting a red pocket is more likely than a specific number, these probabilities need not be equal.

run: one iteration of a scenario.

scale: the size of spatial units and the duration of temporal units within the modeled system.

scenario: a set of parameter values used to run the model.

seed: a list of numbers generated by a (pseudo-) random number generator used in a run.

self-organized criticality: a phenomenon in which a system undergoes a transition from one semi-equilibrium state to another without any external influence.

sensitivity analysis: running a model repeatedly to determine how and to what degree varying input parameters changes output measures.

simulation: dynamic model of a system. A model with a time axis.

simultaneous encounter: choosing among several different options and weighing their expected returns.

spatial autocorrelation: In his first law of geography, Tobler (1970, p. 236) states that "everything is related to everything else, but near things are more related than distant things."

spatial variability: heterogeneous distribution of an attribute, such as soil type, across space.

state variable: attributes of a model's entities, such as age, energy, color, etc.

stochasticity: degree of randomness in a model's dynamics and outcomes; involving randomly generated numbers.

stylized facts: broad, generalized patterns in data that, although true in general, may not be entirely correct in detail.

subjunctive models: tackle the question, "What would happen if…?"

syntax: in computer programming, the set of rules, or grammar, governing the programming language.

temporal variability: heterogeneous distribution in time of an attribute, such as average rainfall.

theory-driven models: models maximizing generalism, usually exploring relationships and causality and often used to explain phenomena.

topology: in ABM, the relative organization of space/geometry in which agents operate. This may be a 2D plane, cylinder, torus, or network.

torus: a doughnut-shaped landscape topology where an agent going off the top will appear at the bottom, and when traveling out one side will reenter from the other.

toy landscapes: simple landscapes with carefully controlled characteristics designed to help us understand a model's functioning.

tractability: a feature of models denoting how easy it is to understand the internal dynamics, interactions between processes, and causality chain that leads to the patterns in output.

trade: exchange that involves money as intermediary and is market-based.

tragedy of the commons: a phenomenon that extends well beyond collective agreements in small-scale agrarian populations.

uncertainty analysis: running a model through the uncertainty range of the input parameters to determine the robustness of the model outcomes with respect to that uncertainty level.

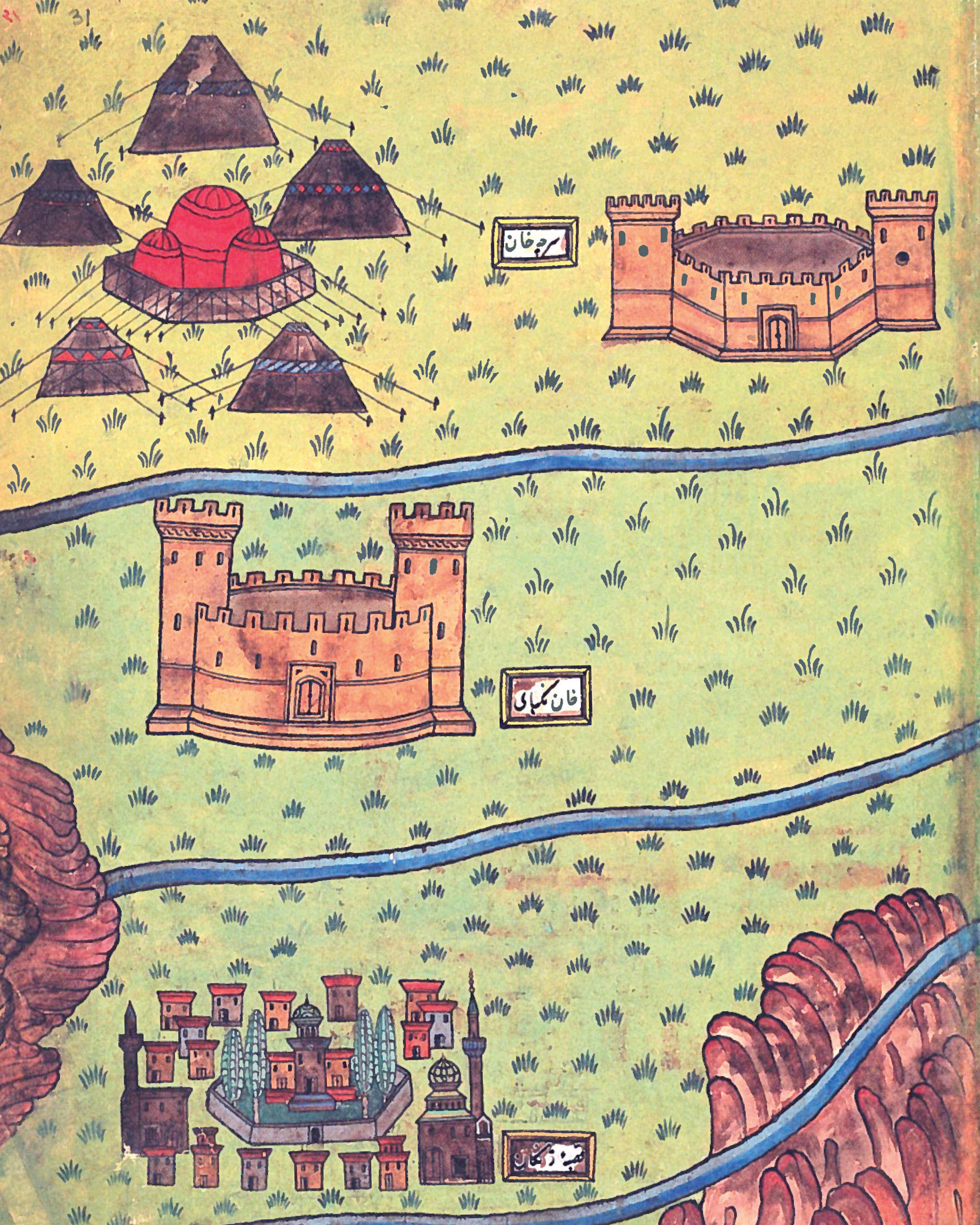
undirected edge: two-way connection, such as a reciprocal relationship.

validation: establishing that the model is conceptually correct with respect to the system being modeled.

variable: attribute of an agent, patch, or the world that can change over the course of a simulation run, such as age, fitness, grass, or number of agents alive.

verification: establishing that the code, as written, works correctly.

von Neumann neighborhood: consists of a central cell and the immediately adjacent four cells (including diagonals). Note NetLogo's "neighbors4" primitive excludes the central cell.


سرچه خان
خان نکبای
قصبه زنگان

THE ABMA MODEL ZOO

This appendix includes a complete list of the models we discuss in this textbook. For each model, we include a citation to the original source, location within the textbook, and a link to the code, if available. In cases where we have either created an original replication of the model or have significantly modified the original code, we also include that model in our ABMA Code Repo.[1]

ACHE HUNTING

▷ M. A. Janssen and K. Hill. 2014. "Benefits of Grouping and Cooperative Hunting Among Ache Hunter–Gatherers: Insights from an Agent-Based Foraging Model." *Human Ecology* 42, no. 6 (August): 823–835. doi:10.1007/s10745-014-9693-1

See chapters 4 and 6.

Code: https://doi.org/10.25937/66d6-kz70

▷ M. A. Janssen and K. Hill. 2016. "An Agent-Based Model of Resource Distribution on Hunter-Gatherer Foraging Strategies: Clumped Habitats Favor Lower Mobility, but Result in Higher Foraging Returns." In *Simulating Prehistoric and Ancient Worlds,* edited by J. A. Barceló and F. D. Castillo, 159–174. Computational Social Sciences. Cham, Switzerland: Springer International Publishing. doi:10.1007/978-3-319-31481-5_3

See chapters 4 and 6.

AGMODEL

▷ L. Barton and I. I. T. Ullah. 2016. "Computer Simulation and the Origins of Agriculture in East Asia." Presented at the Seventh Worldwide Conference of the SEAA June 8–12, 2016, Cambridge/Boston, MA, USA. Boston, MA.

See chapter 6.

▷ I. I. T. Ullah. 2015. *Agmodel: Version 0.3.* doi:10.5281/ZENODO.17551

[1] https://santafeinstitute.github.io/ABMA/

AMPHORABM

See chapters 6 and 9.

▷ S. A. Crabtree. 2016. "Simulating Littoral Trade: Modeling the Trade of Wine in the Bronze to Iron Age Transition in Southern France." *Land* 5, no. 1 (February): 5. doi:10.3390/land5010005

Code: https://comses.net/codebases/c310d351-b629-46ec-a29e-eb365aaa08b4/

THE ANTONINE ITINERARIES

See chapter 5.

▷ S. Graham. 2006. "Networks, Agent-Based Models and the Antonine Itineraries: Implications for Roman Archaeology." *Journal of Mediterranean Archaeology* 19 (1): 45. doi:10.1558/jmea.2006.19.1.45

Code: http://www.graeworks.net/2013/05/16/antonine-itineraries-abm/

ARTIFICIAL ANASAZI

See chapter 3.

▷ R. L. Axtell et al. 2002. "Population Growth and Collapse in a Multiagent Model of the Kayenta Anasazi in Long House Valley." *Proceedings of the National Academy of Sciences* 99 (suppl 3): 7275–7279. doi:10.1073/pnas.092080799

See also SUGARSCAPE.

BIDDING MARKET

See chapter 5.

▷ J. Baker. 2017. *NetLogo Bidding Market Model.* Evanston, IL. https://ccl.northwestern.edu/netlogo/models/BiddingMarket

BOIDS

See chapter 4.

▷ C. W. Reynolds. 1987. "Flocks, Herds, and Schools: A Distributed Behavioral Model." In *SIGGRAPH '87 Conference Proceedings,* 21:25–34. 4. Anaheim, CA: Computer Graphics. doi:10.1145/37402.37406

Code (NetLogo): https://ccl.northwestern.edu/netlogo/models/Flocking

CARDIAL SPREAD

See chapters 4 and 6.

▷ S. M. Bergin. 2016. "Mechanisms and Models of Agropastoral Spread During the Neolithic in the West Mediterranean: The Cardial Spread Model." PhD, Arizona State University. https://search.proquest.com/docview/1845308964/abstract/D9BE13322D14FB5PQ/1

▷ S. Bergin. 2019. *The Cardial Spread Model.* CoMSES Computational Model Library. https://www.comses.net/codebases/5278/releases/1.1.0/

See also MEDLAND.

CORRELATED RANDOM WALK

▷ C. S. Patlak. 1953. "Random Walk with Persistence and External Bias." *The Bulletin of Mathematical Biophysics* 15, no. 3 (September): 311–338. doi:10.1007/BF02476407

See chapter 4.

CULTURAL HITCHHIKING

▷ G. J. Ackland et al. 2007. "Cultural Hitchhiking on the Wave of Advance of Beneficial Technologies." *Proceedings of the National Academy of Sciences* 104, no. 21 (May): 8714–8719. doi:10.1073/pnas.0702469104

See chapter 6.

CULTURAL TRADE

▷ S. Carrignon, T. Brughmans, and I. Romanowska. 2020. "Tableware Trade in the Roman East: Exploring Cultural and Economic Transmission with Agent-Based Modelling and Approximate Bayesian Computation." *PLOS ONE* 15, no. 11 (November): e0240414. doi:10.1371/journal.pone.0240414

Code: https://osf.io/s5mdw/

See chapter 5.

CULTURAL TRANSMISSION ALGORITHMS

▷ Cultural Transmission Algorithms was developed for this textbook by Romanowska.

See chapter 5.

DEMOGRAPHIC CULTURAL EVOLUTION

▷ A. Powell, S. Shennan, and M. G. Thomas. 2009. "Late Pleistocene Demography and the Appearance of Modern Human Behavior." *Science* 324 (5932): 1298–1301. doi:10.1126/science.1170165

See chapter 5.

DIET-BREADTH

▷ C. M. Barton. 2015. *Diet-Breadth Model from Optimal Foraging Theory (Human Behavioral Ecology).* https://www.comses.net/codebases/2225/releases/1.1.0/

See chapter 6.

EDGEWORTH BOX GAME

▷ L. Hamill and N. Gilbert. 2016. *Agent-Based Modelling in Economics.* Chichester, UK: Wiley. doi:10.1002/9781118945520

Code: https://hamill.co.uk/lynne-hamill/abm-in-economics/models

See chapter 5.

FISSION–FUSION

See chapter 6.

▷ E. R. Crema. 2014. "A Simulation Model of Fission-Fusion Dynamics and Long-Term Settlement Change." *Journal of Archaeological Method and Theory* 21 (2): 385–404. doi:10.1007/s10816-013-9185-4
Code, original in R: https://github.com/ercrema/fissionfusion2014

FORAGING MEMORY CAPACITY

See chapter 4.

▷ A. Costopoulos. 2001. "Evaluating the Impact of Increasing Memory on Agent Behaviour: Adaptive Patterns in an Agent-Based Simulation of Subsistence." *Journal of Artificial Societies and Social Simulation* 4, no. 4 (October). https://jasss.soc.surrey.ac.uk/4/4/7.html

GER GROUPER

See chapters 4 and 6.

▷ J. K. Clark and S. A. Crabtree. 2015. "Examining Social Adaptations in a Volatile Landscape in Northern Mongolia via the Agent-Based Model Ger Grouper." *Land* 4, no. 1 (March): 157–181. doi:10.3390/land4010157
Code: https://comses.net/codebases/27f01923-3884-48ca-81ca-55739f976dc0/

HOMININSPACE

See chapter 4.

▷ F. Scherjon. 2019. "Virtual Neanderthals: A Study in Agent-Based Modelling Late Pleistocene Hominins in Western Europe." PhD diss. https://openaccess.leidenuniv.nl/handle/1887/73639
Code: https://www.comses.net/codebases/5294/

LÉVY FLIGHTS

See chapter 4.

▷ D. O'Sullivan and G. L. W. Perry. 2013. *Spatial Simulation: Exploring Pattern and Process.* Chichester, UK: John Wiley & Sons, September. https://patternandprocess.org/
Code: https://patternandprocess.org/category/chapter-4/

LGM ECODYNAMICS

See chapter 6.

▷ C. D. Wren and A. Burke. 2019. "Habitat Suitability and the Genetic Structure of Human Populations during the Last Glacial Maximum (LGM) in Western Europe." *PLOS ONE* 14, no. 6 (June): e0217996. doi:10.1371/journal.pone.0217996
Code: https://doi.org/10.25937/na38-tj46

MAGICAL

▷ M. W. Lake. 2000. "MAGICAL Computer Simulation of Mesolithic Foraging." In *Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes,* edited by T. A. Kohler and G. J. Gumerman, 107–143. Oxford, UK: Oxford University Press.

See chapter 4.

MARKET

▷ L. Hamill and N. Gilbert. 2016. *Agent-Based Modelling in Economics.* Chichester, UK: Wiley. doi:10.1002/9781118945520

Code: https://hamill.co.uk/lynne-hamill/abm-in-economics/models

See chapter 5.

MEDLAND

▷ H. S. Sarjoughian and B. R. Zeigler. 1998. "DEVSJAVA: Basis for a DEVS-Based Collaborative M&S Environment." In *SCS Western Multi-Conference,* 30:29–36. San Diego.

See also CARDIAL SPREAD and SWIDDEN FARMING.

See chapters 4 and 6.

MERCURY

▷ T. Brughmans. 2015. *MERCURY: An ABM of Tableware Trade in the Roman East" (Version 1.1.0).* https://www.comses.net/codebases/4347/releases/1.1.0/

Code: https://www.comses.net/codebases/4347/releases/1.1.0/

See chapter 5.

MULTI-AGENT SYSTEM OF THE TRAGEDY OF THE COMMONS (MASTOC)

▷ J. Schindler. 2012. "A Simple Agent-Based Model of the Tragedy of the Commons." In *ECMS 2012 Proceedings,* edited by K. G. Troitzsch, M. Moehring, and U. Lotzmann, 44–50. ECMS, May. doi:10.7148/2012-0044-0050

Code: https://www.comses.net/codebases/2283/

See chapter 6.

MULTILEVEL POPULATION DYNAMICS

▷ N. Gauthier. 2019. *Multilevel Simulation of Demography and Food Production in Ancient Agrarian Societies: A Case Study from Roman North Africa.* Preprint. SocArXiv, August. doi:10.31235/osf.io/5be6a

Code: https://github.com/nick-gauthier/Silvanus

See chapter 6.

NEUTRAL PROCUREMENT

See chapter 7.

▷ P. J. Brantingham. 2003. "A Neutral Model of Stone Raw Material Procurement." *American Antiquity* 68 (3): 487–510. doi:10.2307/3557105

NEOLITHIC SPREAD

See chapter 4.

▷ J. Fort, T. Pujol, and M. V. Linden. 2012. "Modelling the Neolithic Transition in the Near East and Europe." *American Antiquity* 77 (2): 203–219. doi:10.7183/0002-7316.77.2.203

OUT-OF-AFRICA DISPERSAL

See chapter 4.

▷ I. Romanowska et al. 2017. "Dispersal and the Movius Line: Testing the Effect of Dispersal on Population Density through Simulation." *Quaternary International* 431 (Part B): 53–63. doi:10.1016/j.quaint.2016.01.016
Code: https://github.com/izaromanowska/Modelling-Hominin-Dispersals-Using-Agent-based-Modelling

PALEOSCAPEABM

See chapter 4.

▷ C. D. Wren et al. 2020. "The Foraging Potential of the Holocene Cape South Coast of South Africa without the Palaeo-Agulhas Plain." *Quaternary Science Reviews.* doi:10.1016/j.quascirev.2019.06.012

▷ C. D. Wren et al. 2018. "An Agent-Based Approach to Weighted Decision Making in the Spatially and Temporally Variable South African Palaeoscape." In *Proceedings of the 44th Computer Applications and Quantitative Methods in Archaeology Conference (CAA 2016),* 507–522. Oslo, Norway: Archeopress.
Code: https://doi.org/10.25937/r2qq-fn02

PATAGONIAN HUNTER-GATHERERS

See chapter 5.

▷ J. A. Barceló et al. 2015. "Simulating Patagonian Territoriality in Prehistory: Space, Frontiers and Networks Among Hunter-Gatherers." In *Agent-Based Modeling and Simulation in Archaeology,* edited by G. Wurzer, K. Kowarik, and H. Reschreiter, 217–256. Cham, Switzerland: Springer International Publishing. doi:10.1007/978-3-319-00008-4_10

PATCH-CHOICE

See chapter 6.

▷ C. M. Barton. 2013. *Patch-Choice Model from Optimal Foraging Theory (Human Behavioral Ecology).* https://www.comses.net/codebases/2224/releases/1.0.0/

PIAROA SWIDDEN FARMING

▷ P. Riris. 2018. "Assessing the Impact and Legacy of Swidden Farming in Neotropical Interfluvial Environments through Exploratory Modelling of Post-Contact Piaroa Land Use (Upper Orinoco, Venezuela)." *The Holocene* 28, no. 6 (June): 945–954. doi:10.1177/0959683617752857

See chapter 6.

PRESTIGE TRADE

▷ S. Graham. 2005. "Agent-Based Modelling, Archaeology, and Social Organisation: The Robustness of Rome." *The Archaeological Computing Newsletter* 63:1–6

See chapter 5.

PRISONER'S DILEMMA

▷ U. Wilensky. 2002. *NetLogo PD Basic Evolutionary Model.* Technical report. Evanston, IL: Center for Connected Learning and Computer-Based Modeling, Northwestern University. https://ccl.northwestern.edu/netlogo/models/PDBasicEvolutionary

See chapter 6.

PROCESSION MOVEMENT

▷ K. A. Crawford. 2019. "Rethinking Approaches for the Study of Urban Movement at Ostia." In *Finding the Limits of the Limes: Modelling Demography, Economy and Transport on the Edge of the Roman Empire,* edited by P. Verhagen, J. Joyce, and M. R. Groenhuijzen, 313–327. Cham, Switzerland: Springer International Publishing. doi:10.1007/978-3-030-04576-0_15

See chapter 4.

RANDOM WALK

▷ Random walks are a type of stochastic model used in a wide variety of models across many fields. For a noted archaeological example (also replicated in ch. 1), see:

P. J. Brantingham. 2003. "A Neutral Model of Stone Raw Material Procurement." *American Antiquity* 68 (3): 487–510. doi:10.2307/3557105

See chapter 4.

RESTRICTED WALK

▷ See "Look Ahead" in the NetLogo Models Library.

U. Wilensky. 1999. *NetLogo.* Evanston, IL: Center for Connected Learning / Computer-Based Modeling, Northwestern University. https://ccl.northwestern.edu/netlogo.

See chapter 4.

ROULETTE REPRODUCTION

See chapter 6.

▷ Roulette Reproduction was developed for this textbook by Wren.

SAHLINS'S MODEL OF EXCHANGE

See chapter 8.

▷ S. A. Crabtree. 2015. "Inferring Ancestral Pueblo Social Networks from Simulation in the Central Mesa Verde." *Journal of Archaeological Method and Theory* 22, no. 1 (March): 144–181. doi:10.1007/s10816-014-9233-8

See also VILLAGE ECODYNAMICS PROJECT.

SHEEP

See chapter 4.

▷ I. Romanowska. 2015. "Agent-Based Modelling and Archaeological Hypothesis Testing: The Case Study of the European Lower Palaeolithic." In *Across Space and Time. Papers from the 41st Conference on Computer Applications and Quantitative Methods in Archaeology, Perth, 25–28 March 2013,* 203–214. Amsterdam, Netherlands: Amsterdam University Press.

Code: https://github.com/izaromanowska/SHEEP

SHOPS

See chapter 5.

▷ L. Hamill and N. Gilbert. 2016. *Agent-Based Modelling in Economics.* Chichester, UK: Wiley. doi:10.1002/9781118945520

Code: https://hamill.co.uk/lynne-hamill/abm-in-economics/models

SIMPLE ECONOMY

See chapter 5.

▷ U. Wilensky. 2011. *NetLogo Simple Economy Model.* Evanston, IL. https://ccl.northwestern.edu/netlogo/models/SimpleEconomy

Code: https://ccl.northwestern.edu/netlogo/models/SimpleEconomy

SUSCEPTIBLE–INFECTED–RECOVERED (SIR) FAMILY OF MODELS

See chapter 5.

▷ Susceptible-Infected-Recovered (SIR) was developed for this textbook by Romanowska.

SPATIAL FORESIGHT

See chapter 4.

▷ C. D. Wren et al. 2014. "The Role of Spatial Foresight in Models of Hominin Dispersal." *Journal of Human Evolution* 69:70–78. doi:10.1016/j.jhevol.2014.02.004

Code: https://www.comses.net/codebases/3846/

STEPPINGIN

▷ F. Scherjon. 2013. "SteppingIn—Modern Humans Moving into Europe—Implementation." In *Proceedings of the 40th Conference on Computer Applications and Quantitative Methods in Archaeology Southampton, 26–30 March 2012,* 105–117. Amsterdam, Netherlands: Amsterdam University Press.

See chapter 4.

STEPPING OUT

▷ S. J. Mithen and M. Reed. 2002. "Stepping Out: A Computer Simulation of Hominid Dispersal from Africa." *Journal of Human Evolution* 43 (4): 433–462. doi:10.1006/jhev.2002.0584

See chapter 4.

SUGARSCAPE

▷ J. M. Epstein and R. Axtell. 1996. *Growing Artificial Societies: Social Science from the Bottom Up.* Complex Adaptive Systems. Washington, DC: Brookings Institution Press.

See chapter 3.

See also ARTIFICIAL ANASAZI.

SWIDDEN FARMING

▷ C. M. Barton. 2014. "Complexity, Social Complexity, and Modeling." *Journal of Archaeological Method and Theory* 21, no. 2 (June): 306–324. doi:10.1007/s10816-013-9187-2

See chapter 6.

Code: https://doi.org/2286.0/oabm.3826

See also MEDLAND.

TARGETED WALK

▷ Targeted Walk was developed for this textbook by Romanowska.

See chapter 4.

TIME-AVERAGING

▷ L. S. Premo. 2014. "Cultural Transmission and Diversity in Time-Averaged Assemblages." *Current Anthropology* 55, no. 1 (February): 105–114. doi:10.1086/674873

See chapter 5.

Code: https://www.journals.uchicago.edu/doi/full/10.1086/674873

TRADE DISTANCE

See chapter 2.

▷ I. Romanowska. 2018. "Using Agent-Based Modelling to Infer Economic Processes in the Past." In *Quantifying Ancient Economies. Problems and Methodologies,* edited by J. R. Rodriguez, V. R. Calvo, and J. M. Bermudez Lorenzo, 107–118. *Instrumenta* 60. Barcelona, Spain: University of Barcelona.

TRAIL EVOLUTION

See chapter 4.

▷ D. Helbing, J. Keltsch, and P. Molnar. 1997. "Modelling the Evolution of Human Trail Systems." *Nature* 388 (6637): 47–50. doi:10.1038/40353

VILLAGE ECODYNAMICS PROJECT (VEP)

See chapter 6.

▷ T. A. Kohler et al. 2012. "Modelling Prehispanic Pueblo Societies in their Ecosystems." *Ecological Modelling* 241 (August): 30–41. doi:10.1016/j.ecolmodel.2012.01.002

See also SAHLINS'S MODEL OF EXCHANGE.

WALK THIS WAY

See chapter 4.

▷ A. T. Crooks et al. 2019. *Agent-Based Modelling and Geographical Information Systems.* London, UK: Sage Publishing. https://www.abmgis.org

Code (NetLogo): https://github.com/abmgis/abmgis/tree/master/Chapter10-EvaluatingModels/Models/WalkThisWay

WOLF–SHEEP PREDATION

See chapter 6.

▷ U. Wilensky. 1997. *NetLogo Wolf-Sheep Predation Model.* Evanston, IL.

Code: https://ccl.northwestern.edu/netlogo/models/WolfSheepPredation

YOUNG & BETTINGER (Y&B) DISPERSAL

See chapter 1.

▷ D. A. Young and R. L. Bettinger. 1995. "Simulating the Global Human Expansion in the Late Pleistocene." *Journal of Archaeological Science* 22 (1): 89–92. doi:10.1016/S0305-4403(95)80165-0


آستران نام دیگر
طقوز اولوم
مذکور صوین کوپ نام دیگر
کوشکی سیلان بابا پلاوک
اووه بردان

MAKING COLORBLIND-FRIENDLY ABMS

Stan Rhodes

Accessibility Beyond the Defaults

In this appendix, we discuss how modelers can make the models they write in NetLogo more accessible to people with colorblindness.[1] We also challenge the common use of the default color schemes and scales in NetLogo by explicitly discussing their drawbacks for user experience, and offering alternatives. NetLogo made agent-based modeling accessible worldwide. However, we have to acknowledge that, at the time of this book, its accessibility has improved only modestly and its graphical options are limited. Graphics aren't everything, but they can help tell the story of the model visually. Although this appendix concentrates on making models colorblind-friendly, we hope that the considerations discussed here will help us all think a bit more about our graphical choices, and also what the future of agent-based modeling might—and should—hold.[2]

A Quick Primer on Colorblindness

Colorblindness affects people worldwide. However, "colorblindness" as a term is a bit misleading: most of those with colorblindness have a reduced ability to see differences in color, from just below normal color perception to complete inability to perceive the affected color.

The most common type—*deuteranomaly*—results in reduced perception of green light and difficulty distinguishing between red and green colors. Similarly, *protanomaly* is a reduced sensitivity to red light. Together, deficiencies in red and green cones are called red–green colorblindness. *Trianomaly* and *tritanopia* refer to deficiencies in

[1] You can find the full-color PDF version of this appendix in the ABMA Code Repo: https://github.com/SantaFeInstitute/ABMA/tree/master/appendix

[2] We assume modelers will consider overall visualization issues before, and in conjunction with, this appendix. For guidance on improving visualization for agent-based modeling, please see Kornhauser, Wilensky, and Rand (2009).

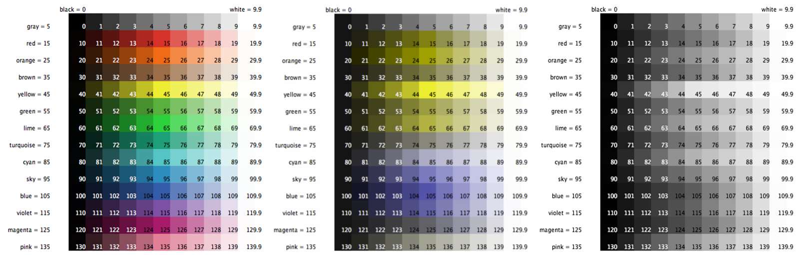


Figure A.0. The NetLogo palette in full-color, simulated deuteranopia, and simulated monochromacy.

perceiving blue. Monochromacy is the inability to distinguish among any colors.

The terminology used indicates the source of the deficiency: for example, deuter*anomaly* indicates a defect in the cones for green, whereas deuter*anopia* indicates the absence of cones for green.

Blue colorblindness occurs at roughly the same rate in males and females, while most people with red–green colorblindness are male. Monochromacy is much rarer than other forms of colorblindness in both males and females.

For a sense of how a person might with colorblindness might experience a graphic, see figure A.0, which simulates these difficulties in distinguishing colors.

Seeing the Problem

We recommend using a program to simulate colorblind views of your model. When adjusting the models in this book to be more colorblind-friendly, we used Color Oracle (available on Windows, Mac, and Linux) frequently.[3]

Color Oracle's grayscale option darkens all colors, including white, so use that option with some caution.

A web search will find many pre-built colorblind-friendly color schemes. We recommend the ColorBrewer website (Harrower and Brewer 2003) for looking at existing color schemes for maps, which are often a good starting point in thinking about possible color schemes for your agent-based model.[4] Also, ColorBrewer and a colorblind simulator can help you develop intuition about why, for example, three distinct colors can work well, but six distinct colors lose distinguishability. After selecting different color schemes and numbers of total distinct data classes in ColorBrewer, use a colorblindness simulator to assess how it

We recommend Cynthia Brewer's book, *Designing Better Maps: A Guide for GIS Users*. Brewer (2016)

[3] https://colororacle.org/

[4] https://colorbrewer2.org/

would appear to different types of colorblindness. Keep a healthy skepticism about colorblind-friendly palettes available on the web, as some work best for only one type of colorblindness, or have weak contrast in part of their scheme. Always do your own visual test of your model with a colorblindness simulator.

The Challenges of Agent-Based Modeling and Color Schemes

Making good color schemes for agent-based models can be hard because we are so constrained in our color options. Models offer many reasons to color everything differently: landscapes may have more than one value per patch; agents may be of different types, with different attributes. When looking at or developing a color scheme for your model, keep in mind that models with patches *and* agents will very likely need the agents to be set in colors distinct from the set of colors used for the patches, otherwise they won't show up well. You'll need to think about agent colors as one or more of the total colors in the overall color scheme.

Keep the Big Picture in Mind

It may be tempting to color everything distinctly, but think about what intervals and which distinctions are essential for the audience to understand the big picture. Consider where distinct colors are essential versus nice-to-have: for example, are intervals for land degradation really important, and if so, what are the largest viable intervals for those values? Difficulty in simplifying and emphasizing parts of the model may reflect areas where model visualization should be improved.

See Kornhauser, Wilensky, and Rand (2009) for guidance and a process to look at the big picture and refine the visualizations of your model.

Shapes Can Help Reduce the Need for Distinct Colors

Shapes expand your ability to distinguish between patches and agents. Do different agents really need different colors, or could you use different shapes instead? A dark triangle on a patch may denote mountainous terrain as well or better than a distinct color. Different or smaller shapes may differentiate juvenile animals from adults. With enough contrast between colors, a smaller shape can be placed on a larger shape. For example, a light-colored spot can be placed on an animal shape to indicate disease, or on a plant shape to indicate flowering or fruiting. These

This appendix includes a section on shapes near the end.

smaller shapes can use colors that are the same or similar to another light color in your palette. The shape and contrast make the distinction, so the color similarity doesn't matter; no one will confuse a yellow dot on a deer for a yellow patch denoting sandy soil.

Which Colors Work, Which Don't

You may be asking "Which colors should I use, then?" Unfortunately the answer is, "It depends." It depends on the model, and it depends on what distinctions you want users to be able to make when viewing the model. We'd recommend using a few heuristics to guide your process.

CONSIDER WHICH GRADIENTS ARE CRUCIAL

Consider any crucial gradient(s) you want to display first, as it will be the biggest constraint on your remaining colors. What color makes the most sense for the maximum value of the gradient? What are the largest viable intervals for that value, and thus the fewest intervals that will work to show differences? Would a lighter or darker gradient make sense, and would it be possible to only use a light or dark gradient, rather than the whole range? Consider the full range of the gradient you want to display and then how many shades within the gradient are important to include. Fewer shades with larger intervals may be sufficient to show the differences you need.

IF GREEN IS ESSENTIAL, START WITH IT

Consider if green is a crucial color in your model, because that will constrain your options. This really holds true for any crucial color. However, green is special: green is used in agent-based models all the time as an indicator of vegetation, yet it's also a problematic color for most types of colorblindness. Rethinking the use of plain, pure green in models is a core challenge of making them colorblind friendly, and often a good starting point for considering the palette of the model. Would lighter greens or darker greens be a better choice? Could you keep the meaning of the green while changing the hue to a bluer or yellower green?

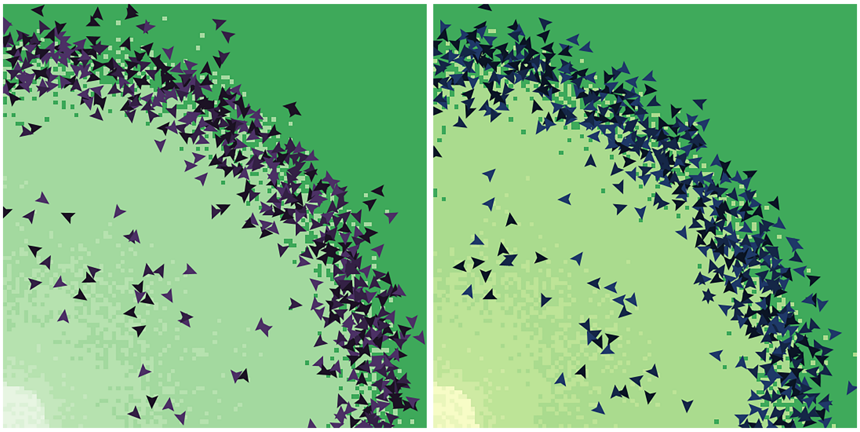


Figure A.1. If a green gradient is essential for the model, the best option for two contrasting color gradients is light green and dark violet (left). Yellow-green and dark blue (right) is a second, but much less ideal, option.

SPLIT THE COLOR SCHEME INTO LIGHT AND DARK OPTIONS

If having many colors seems unavoidable, assemble your color scheme options by dividing it up into two parts: lighter and darker. These may be gradients or discrete sets. If only one gradient is needed, light agents can be used on the dark gradient, or vice versa. If two gradients are needed, consider using only black and/or white agents.

Try to avoid brown if you can. Being a combination of red and green, it's usually not worth the trouble unless it really is essential to visually communicate an aspect of the model.

Gradients in colors schemes

If using a green gradient, the best option for two contrasting color gradients is light green and dark violet. See figure A.1. Yellow-green and dark blue can be used, but is not ideal for tritanopia, as the hues look similar.

In this example, we apply the NetLogo named color to the turtles and we use the palette extension to apply the "Greens" color map from ColorBrewer to the patches.

We will cover use of the palette extension in more detail later in the appendix.

CODE BLOCK A.0

```
create-turtles 500 [
  set size 4
  set color (random 3 + violet - 4)
]
```

CODE BLOCK A.0 (cont.)

```
ask patches [
  set pcolor palette:scale-gradient
    palette:scheme-colors
    "Sequential" "Greens" 5 soil_quality 0 100
]
```

Discrete values in color schemes

Discrete values need to be farther apart in contrast than colors in a gradient, which means that while four discrete values can be colorblind friendly, five or more discrete values will become indistinguishable for people with colorblindness. You can use ColorBrewer to find discrete values that will work if you use their given RGB values with NetLogo's `set color` and `set pcolor`. ColorBrewer does not have colorblind-friendly discrete sets above four discrete values.

The use of any gradient or background colors constrains your available colors. Although in the model's VIEW, the patches' colors are the likely constraint, NetLogo's white-background plots are another common constraint. We'll provide a few options here for discrete value sets that are distinguishable on plots. If the colors aren't to be used on plots, you have slightly more options.

If you're plotting discrete colors, you'll need to avoid light colors like yellow so that the lines are visible on the plot. Five discrete colors inevitably results in one being light-colored.

▷ *Discrete values using NetLogo's named colors*

- Four-color discrete set:

  `red`, `orange + 2`, `blue`, `black`.

- Five-color discrete set (see fig. A.2):

  `red`, `orange + 2`, `blue`, `black`, `lime + 3`.

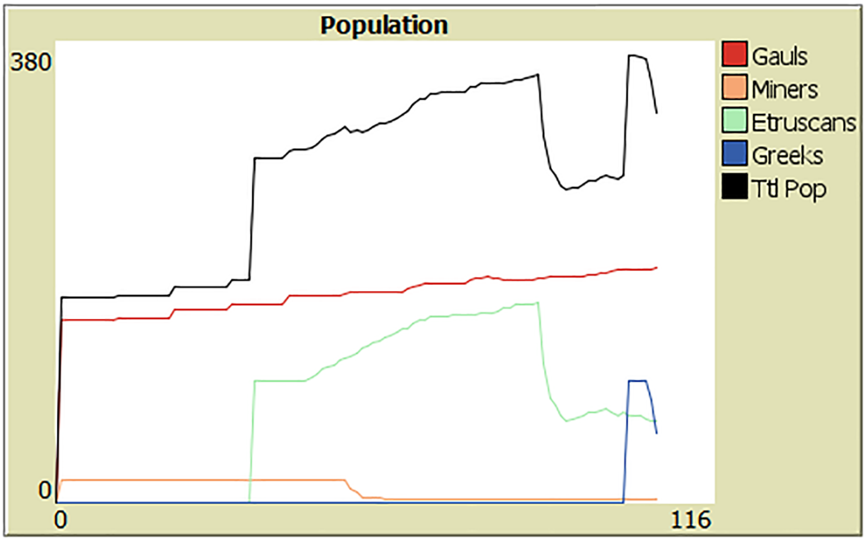


Figure A.2. This set of five discrete values can be used in a plot, but adding a further color will make two of the plot colors indistinguishable. Ideally, we would use only four.

▷ *Discrete values using* RGB *values from viridis*

- Five-color discrete set:

CODE BLOCK A.1

```
[252 228 30]
[144 214 45]
[31 146 115]
[42 72 122]
[54 0 65]
```

NEEDING MANY COLORS SUGGESTS HIGH MODEL COMPLEXITY

If you find yourself needing a lot of colors to distinguish between elements and components of the model, that suggests the model may be too complex to be easily understood by others. In that case, using a lot of colors is unlikely to improve the comprehensibility of the model, and you should consider simplifying what you're communicating about the model. You may be able to divide your visualization into separate concepts and find a reduced color option for each of them.

In cases where just one or two more colors seem needed to distinguish elements of the model, take another look at whether using shapes

Figure A.3. Setting the color bounds to match the bounds of the the variable range often leads to a confusing mess.

might help. If not, or if you are using a lot of shapes already, reassess the complexity of the model in light of what you want to communicate.

The Challenges of Default NetLogo Color Gradients

NetLogo's default color scales present some challenges for user interpretations of models, which then make colorblind-friendly considerations a little extra tricky. Consider a model that has patches with a soil fertility score between 0 and 100. We might normally map that to a color using `scale-color`:

CODE BLOCK A.2

```
ask patches [ set pcolor scale-color
    green fertility 0 100 ]
```

The result is low-fertility patches that are nearly black, and high-fertility patches that are nearly white.

Visually, these defaults cause patches to blend with agents, and conflict with a basic intuition that fertile soil produces very green grass (instead, here the highest fertility is depicted as white). One way to adapt NetLogo's default behavior for an improved visual gradient is to set the range limits to be outside the fertility variable's range. We can use this for

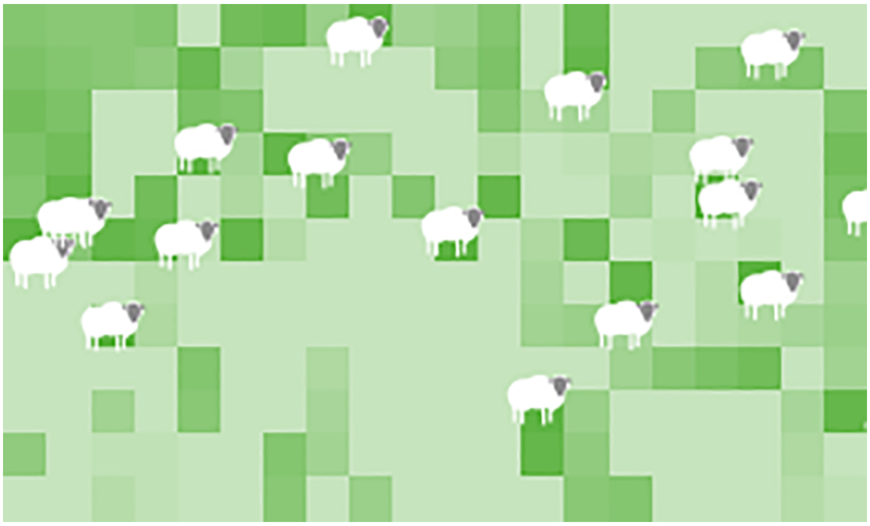


Figure A.4. Setting the color bounds out of range enables more intuitive color gradients than matching the bounds with the variable range.

both light and dark gradients because we can invert the minimum and maximum values to reverse the color gradient. For example, we might want low fertility to be light green and high fertility to be dark green. Before we increase the range limits, we place the larger number first to invert the scaling so that low values are lighter and high values are darker (`100 0` instead of `0 100`). Then, with the color gradient now going from light to darker, we extend the range by adding some padding (e.g., ~20) to the resulting numbers so that the extremes are outside the variable's range:

CODE BLOCK A.3

```
ask patches [ set pcolor scale-color
    green fertility 120 -20 ]
```

Guessing at how much padding to add is, well, guessing. If we could normalize the range between 0 and the max of a variable for any variable, we could avoid NetLogo's color variant extremes consistently without guesswork. And we can!

AVOIDING EXTREME LIGHT AND DARK COLORS IN VANILLA NETLOGO

Here we provide a more programmatic and consistent way of setting the appropriate outside-the-variable-range bounds to avoid light and dark extremes, which also uses a vivid color (one of NetLogo's named colors) at one end of the gradient instead. In figure A.5 we show four gradients: vivid-to-light, light-to-vivid, vivid-to-dark, and dark-to-vivid. The code for each requires a variable, we use `fertility`, that ranges from 0 to a maximum value. We also set `varmax` to be the maximum value of `fertility`, which should be set in the code (e.g., `set varmax 100`). Here is the code for each in turn:

▷ Vividest color = 0, light color = variable's max value:

CODE BLOCK A.4

```
let upperbound (varmax + varmax / 2)
let lowerbound (-1 * (varmax + varmax / 2))
set pcolor scale-color green
    fertility lowerbound upperbound
```

▷ Light color = 0, vividest color = variable's max value:

CODE BLOCK A.5

```
let upperbound (2 * varmax + varmax / 2)
let lowerbound -1 * (varmax / 2)
set pcolor scale-color green
    fertility upperbound lowerbound
```

▷ Vividest color = 0, dark color = variable's max value:

CODE BLOCK A.6

```
let upperbound (-1 * (varmax + varmax / 2))
let lowerbound (varmax + varmax / 2)
set pcolor scale-color green
    fertility lowerbound upperbound
```

▷ Dark color = 0, vividest color = variable's max value:

CODE BLOCK A.7

```
let upperbound -1 * (varmax / 2)
let lowerbound (2 * varmax + varmax / 2)
set pcolor scale-color green
    fertility upperbound lowerbound
```

These are an improvement over the default behavior, and are colorblind friendlier. Often they will do the trick if you need to use a NetLogo word color; you can use any of NetLogo's named colors in place

Figure A.5. Programmatically setting the color bounds out of range enables fine, consistent control of the color gradient with any range of variable values. From left to right: vivid-to-light, light-to-vivid, vivid-to-dark, dark-to-vivid.

of `green`. However, we can usually do as well or better with the palette extension because it can use the predefined, and multi-color, ColorBrewer palettes.

USING THE COLOR-PALETTE EXTENSION

The palette extension is a default extension included in NetLogo, and its functions can be made available within a NetLogo by including `extensions [palette]` at the start of the code. Here is an example of using the palette extension to set a color gradient:

CODE BLOCK A.8

```
ask patches [
  set pcolor palette:scale-gradient
    palette:scheme-colors
    "Sequential" "Reds" 3 artifact_density 0 100
]
```

The NetLogo palette extension doesn't have the sequential ColorBrewer class "YlGn" (which is a really useful one!), but we provide an implementation of it below.

The easiest way to use the color-palette extension is with ColorBrewer settings. Make sure to test the qualitative ColorBrewer schemes in your model with a colorblindness simulator tool. Some of the schemes work better than others, especially on plots.[5]

[5] The original documentation on the palette extension is a bit scattered; please see:
https://ccl.northwestern.edu/netlogo/docs/palette.html
http://ccl.northwestern.edu/papers/ABMVisualizationGuidelines/palette/
http://ccl.northwestern.edu/papers/ABMVisualizationGuidelines/palette/doc/NetLogo%20Color%20Howto%202.htm

Advanced Palette Extension: Making Custom Palettes

The palette extension also gives us advanced options for colors where we can specify the entire scheme with RGB values. In our experience, this is often not worth the effort, and also makes for very hefty code. Use it sparingly and only when you can't find another viable alternative.

The nine-data class version would have nine RGB colors in the list.

Currently the YlGn palette from ColorBrewer has not been implemented in NetLogo's palette extension, so we will implement it here as an example using ColorBrewer's YlGn color scheme with five data classes. To get these values, we went to the ColorBrewer website[6] and selected the YlGn sequential scheme, then made sure we copied the RGB (versus HEX or CYMK) values. The copied text will require some cleanup to make the RGB values the format NetLogo requires.

CODE BLOCK A.9

```
ask patches [
  set pcolor palette:scale-gradient
        [
          [255 255 204]
          [194 230 153]
          [120 198 121]
          [49 163 84]
          [0 104 55]
        ]
        variable-name 0 100
  ]
```

We recommend using a text editor with good find-and-replace functionality when working with large color palettes to clean up commas and insert the square brackets for NetLogo-bound code. This includes programs such as Notepad++ (Windows) or Atom (Windows, Mac, Linux).

Once you assign colors in an RGB format (e.g., `255 255 204`), the typical NetLogo approach of adding or subtracting values (`set pcolor pcolor +3`) to lighten or darken a color will no longer work. The color's data structure is now different; RGB is a list of three 0-to-255 values rather than single number from 0 to 140. Adjusting color in this way is bad coding practice as well; updating the color *only* would mean it no longer represented the underlying value. When the color is set from an underlying variable initially, every subsequent color change should use that underlying value to set the color.

[6] https://colorbrewer2.org/

You might wonder if you can increment RGB values in the same way you would increment single-value or named colors in NetLogo. Unfortunately, changing RGB values isn't a simple linear process, so adding or subtracting values from one of the three RGB values (or all of them) won't give consistent results.

CAN WE USE SOME OPTIMAL COLOR SCHEME?

If we were considering gradients for data visualization only, in most cases we would want to use a color scheme that was uniform to all users, but which also maximized the perceptual range of all users (colorblind or not). A number of color maps fit these criteria well enough, including viridis, magma, inferno, and plasma (created by Stéfan van der Walt and Nathaniel Smith) and cividis (created by Jamie R. Nuñez, Christopher R. Anderton, and Ryan S. Renslow). Unfortunately, implementing these color maps in NetLogo is no easy task: you'll need to convert the map to a long list of RGB values for the palette extension (e.g. code block A.9).

In some cases, these color maps will need to be found inside a package. For example, with viridis, you can load the package in R and look at the color map as a dataframe. The values in the dataframe can then be parsed into RGBs and written to a string that can be used in NetLogo.

Keep in mind that agent-based models are usually more complicated than a single gradient, which means that as more colors are needed for agents, fewer colors are available for gradients. The gradients, then, will need to be pared down to just a few colors; you will need to pick a subrange in the color map that works well for your other color choices.

An RMarkdown document for creating a NetLogo list from the viridis colors is available in the ABMA Code Repo as *viridis_color_calcs.rmd.*

Working with Shapes

Shapes offer a lot of possibilities for improving your model visually, both for distinguishing different types of agents and for clarifying landscape or environmental features.

OUTLINING SHAPES FOR BETTER VISIBILITY

Generally, lighter-colored landscapes with vivid-colored agents work well. However, sometimes a landscape may make agents a little harder to see. Heterogeneous landscapes using color gradients can cause visual

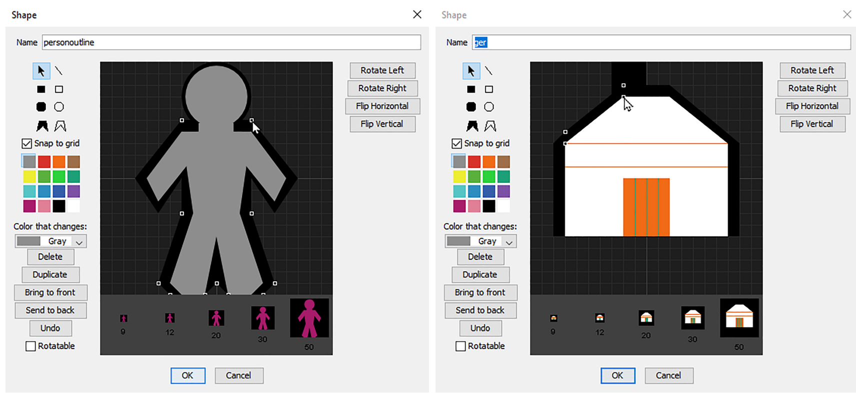


Figure A.6. The shapes editor can be used to outline existing shapes in black for higher contrast and for avoiding agent colors "bleeding" into patch colors.

confusion when part of the gradient has a similar contrast to the agent. In these situations, the agents need more visual "pop."

Remember that you can import shapes from the shape library or import them from another NetLogo model. See https://ccl.northwestern.edu/netlogo/docs/shapes.html

A black outline should give agents the necessary contrast to work on any landscape. Unfortunately, most default shapes have no black outline; fortunately, adding a black outline to most shapes is straightforward, see figure A.6. NetLogo comes with a shapes editor built in, accessible via TOOLS > TURTLE SHAPES EDITOR.

USING SHAPES FOR STATES OR TERRAIN

Shapes may better communicate a terrain type than patch color alone: for example, a triangle may better represent a mountain than trying to pick the "right" mountain color, as in figure A.7.

Shapes may also help denote landscape states better than color alone, particularly when the landscape has events such as snow, rain, seeds germinating, or plants fruiting. See figure A.7.

WHEN USING SHAPES, USER EXPERIENCE IS STILL KING

Kornhauser, Wilensky, and Rand (2009) discuss visual design principles and considerations in great depth.

How many shapes are too many? Again, it depends entirely on the model. A good general approach is to modify one feature at a time, run the model, and ask yourself whether the visualization feels busy. Sometimes taking a little time off from the model before revisiting it can help you see it afresh. If you're still not sure, show it to someone for feedback.

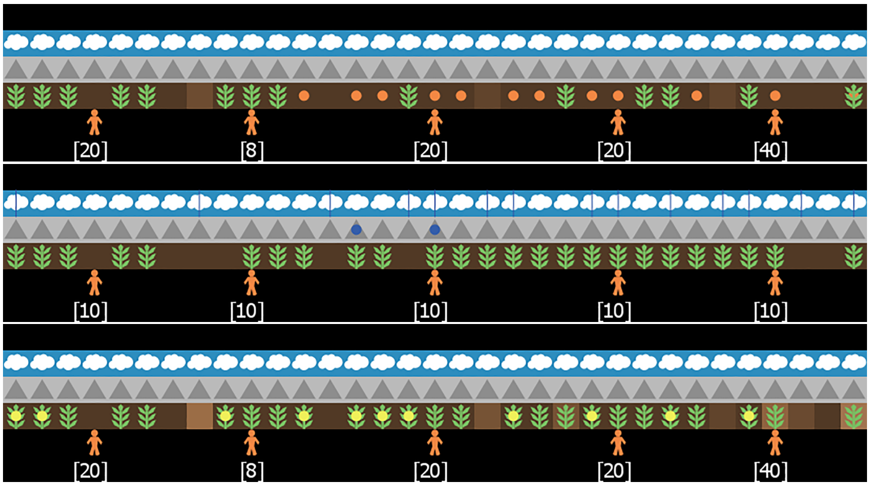


Figure A.7. Shapes can tell part of the story when the paired colors provide sufficient contrast. Top: Seeds, as circles, sprout into plants. Middle: Lines in the clouds represent rain, and water running down to plants are dots on the mountains. Bottom: Plants with mature ears of maize have yellow dots with high contrast.

As Accessibility Evolves, So Does Our Thinking

Keep in mind that although it would be nice to "do it right the first time," that is an unrealistic expectation for any modeler. Developing a clear, concise, and accessible model is an iterative process. Making models colorblind-friendly involves trade-offs in meaningful color-based communication, but we have found that understanding these trade-offs and working within these constraints has improved the depth of our understanding of the use of color, and our approach to modeling overall. We hope you will find the same benefits!