Preface

Welcome to the second edition of *Agent-Based and Individual-Based Modeling*. In the six years since the original edition was published, we have been using and supporting the book, collecting feedback from instructors and people using it to teach themselves, supervising young modelers, and continuing to build, use, and publish our own models. The authors of the software platform we use, Wilensky's NetLogo (Wilensky 1999), meanwhile continued to improve it in important ways. And the NetLogo user community has continued to develop new software tools that can also be extremely helpful. As a result of all this experience, this second edition includes well over 200 revisions.

Many of the revisions were to improve parts of the book that caused particular difficulty to students and instructors, to implement many excellent suggestions from these users, and to simply make things clearer and easier to learn. A few were necessary to update code examples and techniques to correspond to the current version (6.0.4) of NetLogo. A number of revisions, especially in part IV, address new techniques and software tools that have become available. To aid instructors, many of the exercises have been revised, replaced, or newly added. Those of you familiar with the first edition should be aware that we made major revisions to the Butterfly model used in chapters 4 and 5, replaced the software testing exercise model presented in section 6.5, made important changes to the Business Investor model used in chapters 10–12, changed how the Wild Dog model of chapter 16 is programmed, completely revised the style of sensitivity analysis presented in section 23.2.1, and made other updates to chapters 23 and 24 to reflect advances since 2012.

As we revised the book, we also updated the chapter exercises and the solutions available to instructors (discussed below). The new instructor materials are more complete and detailed than they were for the first edition.

We like to believe that the first edition was successful: it was adopted and used as a text at many universities, and we have been contacted by hundreds of individual users. Agent-based and individual-based models (A/IBMs) continue to grow in popularity, in their use in policy- and decision-making, and in their acceptance as a tool that scientists use themselves and teach to others. However, our original goal for the book remains valid. That goal is to address the gap between the need and desire to use A/IBMs and the availability of instructors and instructional

materials for learning how. More university departments want to offer classes in this kind of modeling, but there are still few instructors with the experience to develop courses by themselves. Therefore, we formulated this book to help inexperienced instructors and self-learners access this knowledge.

This book is designed so it can be used by itself, but we think many of its users would benefit from reading our first book, *Individual-Based Modeling and Ecology* (Grimm and Railsback 2005) either beforehand or concurrently with this one. Our first book focused on conceptual aspects of how to design A/IBMs and analyze them to do science, while this book focuses more on the details of implementing and analyzing models. One difference is that this book is not specific to ecology; while it still reflects our own backgrounds in ecology, the book uses many examples from social and human sciences and is designed for the many fields in which a textbook on A/IBMs is needed. This multidisciplinarity is possible because the principles of modeling in general, and agent-based modeling in particular, are independent of scientific domains.

In disciplines other than ecology, IBMs are more often referred to as ABMs, so we use the term "agent-based" in this book more than "individual-based." There have been historical differences between individual- and agent-based models: IBMs focused on individual variability and local interactions, whereas ABMs focused on decision-making and adaptive behavior. But these differences are fading away so that we use both terms interchangeably, as we did in our first book. (In fact, the fascinating bibliographic analysis by Vincenot 2018 indicates that these two terms became interchangeable in part *because of* standards we promote in this book, especially the ODD protocol introduced in chapter 3.) Likewise, we also implicitly include and address "multiagent systems," which are just a branch of agent-based modeling that originated from computer science and research on artificial intelligence and artificial life.

Book Objectives

This book is designed to support introductory classes—or independent study—in agent-based modeling for scientists, including courses with instructors new to simulation modeling and computer programming. The course is targeted at graduate students and advanced undergraduates who are starting their research careers, but it is also appropriate for experienced scientists who want to add agent-based modeling to their toolkit. Students can expect to learn about both the conceptual and theoretical aspects of using ABMs and the details of implementing models on the computer using NetLogo software. Among the topics covered are

- When and why to use ABMs, and how they are different from other models;
- How to design an ABM for a particular system and problem;
- A conceptual foundation for designing and describing models;
- Programming models and conducting simulation experiments in NetLogo; and
- How to analyze a model to solve scientific problems, including development of theory for complex systems.

Our objective is to provide a good foundation in these topics, especially those unique to ABMs. We chose to strategically limit the book to materials that students can reasonably be expected to absorb in one course, and not to delve deeply into technical details that are covered in the very extensive and established literature on general simulation modeling.

Throughout the course we emphasize several themes about doing science with ABMs:

- *Using models for solving research problems.* The primary characteristic of scientific models is that they are designed to solve a specific problem about a system or class of systems.

These problems might include predicting how the system responds to novel conditions, or just understanding the mechanisms that drive the system.

- *Basing models on theory, and using models to develop theory.* By theory, in the context of agent-based complex systems, we mean models of the individual characteristics and behaviors from which system behaviors emerge.
- *Learning and following the conventions and theory of scientific modeling.* Modeling is not simply an intuitive process that lacks standard procedures and theory. There is in fact much that modelers need to learn from our predecessors. Examples include knowing the importance of appropriate space and time scales and how to conduct standard kinds of model analysis.
- *Documenting models and testing software.* These tasks are often treated by novices as tedious distractions, but they are in fact essential—and productive, and sometimes even fun—parts of scientific modeling.
- *Employing standardization.* One of the historical difficulties with ABMs is that the standard "languages" we have for thinking about and describing other kinds of models (e.g., differential equations, statistics) are not sufficient for formulating ABMs. A great deal of recent work has gone into developing standards for ABMs, and we emphasize their use. Throughout this book we use a standard protocol (called ODD) for describing models and a set of standard concepts for thinking about and designing ABMs; and NetLogo itself is a standard language for programming ABMs.

Why Netlogo?

Choosing which software platform to use in this book was a critical decision for us. There are many platforms for agent-based modeling, and they vary in many ways. We learned to use the most popular platforms and tried teaching several of them. This experience led us to two conclusions. First, there is no single ideal platform; platforms are inevitably compromises that cannot be best for all applications. Second, though, NetLogo (Wilensky 1999) clearly stands apart as the best platform for both beginners and serious scientific modelers.

NetLogo provides a programming language that is simple yet extremely powerful for ABMs, along with the graphical interfaces that are essential for testing and understanding them. Therefore, we can spend far less time on programming and much more time doing science. Now, many if not most published scientific ABMs are implemented in NetLogo. Just as importantly, the NetLogo team at Northwestern University provides an extremely complete, helpful, and professional set of documentation and tutorial materials. NetLogo was originally designed as an educational tool, but its use in science has grown very rapidly, and NetLogo itself has changed to better meet the needs of scientists. Further, NetLogo is available for all popular operating systems and is free, although we encourage others to join us in contributing to its support (via the NetLogo web site) from projects and programs that depend on it.

NetLogo has had a reputation as computationally slow, and we used to assume that especially large or complex models would need to be programmed in another language. But this reputation is no longer deserved, as we discuss in chapter 24. While we still introduce general software concepts throughout the book, we now have sufficient experience with large models implemented and analyzed productively in NetLogo that we no longer assume students will need to transition to another platform for "serious" models. Many NetLogo models are initially quite slow, but for reasons that would affect any programming language. In a separate publication and supporting materials (at www.railsback-grimm-abm-book.com/JASSS-models.

html), we explain how to identify and remedy slow parts of NetLogo programs, and we illustrate the many very large models and analyses that have been implemented with NetLogo.

Overview and Suggested Course Structure

The book has four parts. The first provides the foundation that students need to get started: a basic understanding of what agent-based modeling is and of the modeling process, and basic skills in implementing ABMs in NetLogo. Part II introduces specific model design concepts and techniques while widening and deepening the student's NetLogo skills. In part III we move on to broader scientific issues: using knowledge of the system being modeled to design models with the right level of complexity, develop theoretical understanding of the system, and calibrate models. Finally, part IV focuses on analyzing models: how to learn from and do science with ABMs. At the end we provide suggestions for moving on to become a successful modeler without losing momentum.

A course using this book is expected to include both lecture and hands-on computer lab time. We present many modeling and software concepts that are best explained in a lecture format. But most of the learning will take place in a computer laboratory as students work on exercises and projects of increasing complexity and independence. For a college class, we envision each week to include one or two lecture hours plus at least one computer lab of two to four hours.

At the start of the course, chapters 1, 3, and 6 are designed for lecture, while chapters 2, 4, and 5 are designed as introductory computer labs led by the instructor; chapter 6 also includes an important lab exercise. For the remainder of the course, each chapter includes modeling concepts and programming techniques to be introduced in lecture, followed by exercises that reinforce the concepts and techniques. In the computer labs for parts II–IV, students should be able to work more independently, with the instructor spending less time in front of the class and more time circulating to help individual students. Exercises started in lab can be completed as homework.

The exercises in parts I–II are generally short and focused, and it is probably best to have the entire class do the same exercises. Starting with part III, the exercises are more open-ended and demand more creativity and independence. It is natural to transition to more project-like work at this point, and it should be easy to develop independent projects that accomplish the same objectives as the exercises we provide. We strongly encourage instructors to replace our exercises with ones based on their own experience and discipline wherever they can.

Our experience has been that working on exercises in teams, usually of two students, is productive at least early in a class. It is great when students teach and help each other, and it is good to encourage them to do so, for example, by providing a computer lab where they can work in the same room. Anyone who programs regularly knows that the best way to find your mistake is to explain it to someone else. However, it is also very common for beginning programmers to get stuck and need just a little help to get moving again. Programming instructors often provide some kind of "consulting service"—office hours, tutors, email, and the like—to students needing help with their assignments outside of regular class hours.

Even for a graduate-level class, it may be ambitious to try to work through the entire book. Especially in part II, instructors should be able to identify subsections, or perhaps even a whole chapter or two, to leave out, considering their particular interests. We certainly discourage instructors from spending so much time in parts I and II that the key later chapters (especially 18, 19, and 22) are neglected; that would be like teaching art students to mix pigments and stretch canvas but never to paint pictures.

What We Expect of Instructors

We wrote this book with the expectation that many instructors using it will have little or no prior experience with ABMs. This approach is unusual for a textbook, but we had little choice, given how few experienced agent-based modelers there are.

What do we expect of instructors? First, we believe it will help them to digest our first book, *Individual-Based Modeling and Ecology*, fairly thoroughly, especially because it presents many of the ideas in this book in more detail and provides more explanation for why we do things the way we do. The book of O'Sullivan and Perry (2013) also provides valuable and accessible background in the kind of modeling we address.

Second, instructors and the people who assist them need to develop enough experience with the NetLogo platform to stay ahead of their students. This platform can be very easy to learn, but it has a distinct style that takes getting used to. Prior experience with programming may or may not be helpful: NetLogo uses some standard programming techniques (e.g., declaring variables, writing subroutine-like procedures), but if you try to write NetLogo code the way you write C or Java or MatLab, you will miss most of its advantages. Luckily, there is no reason to fear NetLogo. Its excellent documentation and example models will be a great help to you and your students. And NetLogo is fun: if you try to learn it completely by yourself, your primary problem (after an initial 1–2 hours of confusion and frustration) will be making yourself stop. Also, NetLogo is quite popular and it is likely that you can find experienced users on campus, or perhaps even a computer science professor or student willing to learn and help. Finally, there is an online NetLogo users community, programming support on stackoverflow.com, and an online forum for users of this book (discussed below). If you start working your way through NetLogo's tutorial materials and parts I and II of this book with some focus a few weeks before starting to teach, you should be OK.

The third thing we hope for from instructors is to work on keeping students from developing bad habits that are unfortunately common among novice and self-taught modelers. NetLogo encourages an experimental approach: it is very easy and fun to make little code changes and see what happens. This experimentation is fine, but students need to learn that they are not really modeling until they do three things. First is to stop changing and expanding their model—even before it seems "realistic" or "finished"—so they can proceed with learning something from it. Second is to write down what their model is to do and why; and third is to provide solid evidence that their software actually does what they wrote down. After building many ABMs, we know that software that has not been tested seriously is very unlikely to be mistake-free, even with a platform as simple as NetLogo. And a written description of the model is necessary for both scientific communication and software testing. But neither of those tasks can be completed until the student stops changing the model's design. We hope instructors will do what computer programming instructors do: treat assignments as incomplete unless software is accompanied with documentation of (a) its purpose and design and (b) a serious attempt to find mistakes in it. You will find your students making much more rapid progress with modeling after these steps become habit.

In courses using our first edition, some students struggled with programming much more than others. Prior training may or may not help: people simply vary in how rapidly and intuitively they learn to program. We think this course in agent-based modeling should be valuable even to those who go on not to build models but to use the science developed via modeling. (In our work as applied modelers, we find familiarity with the modeling process a very valuable skill in scientists and managers even if they never develop models themselves.) Therefore, we would not require students to have prior programming experience because that could discourage some from even attempting the class. However, instructors should be prepared to provide

extra support to students who are not natural programmers. This support could include consulting by instructors or peers, additional exercises and examples, or even simply providing working code so students can conduct the model analysis steps that are actually more important than programming.

For the Independent Learner

As much as we hope that classes on agent-based modeling will be offered at every university, we realize that this will not happen immediately and some people will need to teach themselves. If you are one of these people, we kept you in mind as we designed the book.

How should you proceed? We recommend you read a chapter and then work through its exercises until you are comfortable moving on. If you are nervous about learning NetLogo by yourself, keep in mind that we will point you to many learning materials and sources of support provided with NetLogo and by its users. The online user community may be especially helpful for you, and you should be able to find other NetLogo users nearby to consult with. And you should feel free to contact us for access to exercise solutions and example code.

Course Web Site and Supporting Materials

Materials supporting this book and its users are available through http://press.princeton.edu/titles/14270.html. Please check this site for news and updates as you use the book. The information available through the site includes

- Any changes to the text, example code, and exercises that result from new versions of NetLogo (this edition is current with NetLogo 6.0.4);
- Input files, documents, etc., used in the text and exercises;
- Supplementary materials, referred to in the book, that we plan to update;
- Corrections and clarifications;
- Materials to help instructors and people teaching themselves; and
- Ways to communicate with us, provide feedback, and share ideas.

Thanks to Dr. Jeremy Wojdak of Radford University, there is also an online forum for this book, part of the Quantitative Undergraduate Biology Education and Synthesis network. This QUBES forum is designed to help instructors using the book interact with each other and the authors, and share instructional materials and experience. There is a link to the QUBES site from the book's web site.