

Quantitative  
Applications  
in the  
Social Sciences

153

# Agent-Based Models

SECOND EDITION

Nigel Gilbert



# **AGENT-BASED MODELS**

Second Edition

## Quantitative Applications in the Social Sciences

### A SAGE PUBLICATIONS SERIES

1. **Analysis of Variance, 2nd Edition** *Iversen/Norpoth*
2. **Operations Research Methods** *Nagel/Neef*
3. **Causal Modeling, 2nd Edition** *Asher*
4. **Tests of Significance** *Henkel*
5. **Cohort Analysis, 2nd Edition** *Glenn*
6. **Canonical Analysis and Factor Comparison** *Levine*
7. **Analysis of Nominal Data, 2nd Edition** *Reynolds*
8. **Analysis of Ordinal Data** *Hildebrand/Laing/Rosenthal*
9. **Time Series Analysis, 2nd Edition** *Ostrom*
10. **Ecological Inference** *Langbein/Lichtman*
11. **Multidimensional Scaling** *Kruskal/Wish*
12. **Analysis of Covariance** *Wildt/Ahtola*
13. **Introduction to Factor Analysis** *Kim/Mueller*
14. **Factor Analysis** *Kim/Mueller*
15. **Multiple Indicators** *Sullivan/Feldman*
16. **Exploratory Data Analysis** *Hartwig/Dearing*
17. **Reliability and Validity Assessment** *Carmines/Zeller*
18. **Analyzing Panel Data** *Markus*
19. **Discriminant Analysis** *Klecka*
20. **Log-Linear Models** *Knoke/Burke*
21. **Interrupted Time Series Analysis**  
*McDowall/McCleary/Meidinger/Hay*
22. **Applied Regression, 2nd Edition** *Lewis-Beck/Lewis-Beck*
23. **Research Designs** *Spector*
24. **Unidimensional Scaling** *Mclver/Carmines*
25. **Magnitude Scaling** *Lodge*
26. **Multiattribute Evaluation** *Edwards/Newman*
27. **Dynamic Modeling** *Huckfeldt/Kohfeld/Likens*
28. **Network Analysis** *Knoke/Kuklinski*
29. **Interpreting and Using Regression** *Achen*
30. **Test Item Bias** *Osterlind*
31. **Mobility Tables** *Hout*

32. **Measures of Association** *Liebetrau*
33. **Confirmatory Factor Analysis** *Long*
34. **Covariance Structure Models** *Long*
35. **Introduction to Survey Sampling** *Kalton*
36. **Achievement Testing** *Bejar*
37. **Nonrecursive Causal Models** *Berry*
38. **Matrix Algebra** *Namboodiri*
39. **Introduction to Applied Demography** *Rives/Serow*
40. **Microcomputer Methods for Social Scientists, 2nd Edition**  
*Schrodt*
41. **Game Theory** *Zagare*
42. **Using Published Data** *Jacob*
43. **Bayesian Statistical Inference** *Iversen*
44. **Cluster Analysis** *Aldenderfer/Blashfield*
45. **Linear Probability, Logit, and Probit Models** *Aldrich/Nelson*
46. **Event History and Survival Analysis, 2nd Edition** *Allison*
47. **Canonical Correlation Analysis** *Thompson*
48. **Models for Innovation Diffusion** *Mahajan/Peterson*
49. **Basic Content Analysis, 2nd Edition** *Weber*
50. **Multiple Regression in Practice** *Berry/Feldman*
51. **Stochastic Parameter Regression Models** *Newbold/Bos*
52. **Using Microcomputers in Research** *Madron/Tate/Brookshire*
53. **Secondary Analysis of Survey Data** *Kiecolt/Nathan*
54. **Multivariate Analysis of Variance** *Bray/Maxwell*
55. **The Logic of Causal Order** *Davis*
56. **Introduction to Linear Goal Programming** *Ignizio*
57. **Understanding Regression Analysis, 2nd Edition**  
*Schroeder/Sjoquist/Stephan*
58. **Randomized Response and Related Methods, 2nd Edition**  
*Fox/Tracy*
59. **Meta-Analysis** *Wolf*
60. **Linear Programming** *Feiring*
61. **Multiple Comparisons** *Klockars/Sax*
62. **Information Theory** *Krippendorff*
63. **Survey Questions** *Converse/Presser*
64. **Latent Class Analysis** *McCutcheon*
65. **Three-Way Scaling and Clustering** *Arabie/Carroll/DeSarbo*

66. **Q Methodology, 2nd Edition** *McKeown/Thomas*
67. **Analyzing Decision Making** *Louviere*
68. **Rasch Models for Measurement** *Andrich*
69. **Principal Components Analysis** *Dunteman*
70. **Pooled Time Series Analysis** *Sayrs*
71. **Analyzing Complex Survey Data, 2nd Edition** *Lee/Forthofer*
72. **Interaction Effects in Multiple Regression, 2nd Edition**  
*Jaccard/Turrisi*
73. **Understanding Significance Testing** *Mohr*
74. **Experimental Design and Analysis** *Brown/Melamed*
75. **Metric Scaling** *Weller/Romney*
76. **Longitudinal Research, 2nd Edition** *Menard*
77. **Expert Systems** *Benfer/Brent/Furbee*
78. **Data Theory and Dimensional Analysis** *Jacoby*
79. **Regression Diagnostics, 2nd Edition** *Fox*
80. **Computer-Assisted Interviewing** *Saris*
81. **Contextual Analysis** *Iversen*
82. **Summated Rating Scale Construction** *Spector*
83. **Central Tendency and Variability** *Weisberg*
84. **ANOVA: Repeated Measures** *Girden*
85. **Processing Data** *Bourque/Clark*
86. **Logit Modeling** *DeMaris*
87. **Analytic Mapping and Geographic Databases** *Garson/Biggs*
88. **Working With Archival Data** *Elder/Pavalko/Clipp*
89. **Multiple Comparison Procedures** *Toothaker*
90. **Nonparametric Statistics** *Gibbons*
91. **Nonparametric Measures of Association** *Gibbons*
92. **Understanding Regression Assumptions** *Berry*
93. **Regression With Dummy Variables** *Hardy*
94. **Loglinear Models With Latent Variables** *Hagenaars*
95. **Bootstrapping** *Mooney/Duval*
96. **Maximum Likelihood Estimation** *Eliason*
97. **Ordinal Log-Linear Models** *Ishii-Kuntz*
98. **Random Factors in ANOVA** *Jackson/Brashers*
99. **Univariate Tests for Time Series Models**  
*Cromwell/Labys/Terraza*
00. **Multivariate Tests for Time Series Models**

*Cromwell/Hannan/Labys/Terraza*

01. **Interpreting Probability Models: Logit, Probit, and Other Generalized Linear Models** *Liao*
02. **Typologies and Taxonomies** *Bailey*
03. **Data Analysis: An Introduction** *Lewis-Beck*
04. **Multiple Attribute Decision Making** *Yoon/Hwang*
05. **Causal Analysis With Panel Data** *Finkel*
06. **Applied Logistic Regression Analysis, 2nd Edition** *Menard*
07. **Chaos and Catastrophe Theories** *Brown*
08. **Basic Math for Social Scientists: Concepts** *Hagle*
09. **Basic Math for Social Scientists: Problems and Solutions** *Hagle*
10. **Calculus** *Iversen*
11. **Regression Models: Censored, Sample Selected, or Truncated Data** *Breen*
12. **Tree Models of Similarity and Association** *Corter*
13. **Computational Modeling** *Taber/Timpone*
14. **LISREL Approaches to Interaction Effects in Multiple Regression** *Jaccard/Wan*
15. **Analyzing Repeated Surveys** *Firebaugh*
16. **Monte Carlo Simulation** *Mooney*
17. **Statistical Graphics for Univariate and Bivariate Data** *Jacoby*
18. **Interaction Effects in Factorial Analysis of Variance** *Jaccard*
19. **Odds Ratios in the Analysis of Contingency Tables** *Rudas*
20. **Statistical Graphics for Visualizing Multivariate Data** *Jacoby*
21. **Applied Correspondence Analysis** *Clausen*
22. **Game Theory Topics** *Fink/Gates/Humes*
23. **Social Choice: Theory and Research** *Johnson*
24. **Neural Networks** *Abdi/Valentin/Edelman*
25. **Relating Statistics and Experimental Design: An Introduction** *Levin*
26. **Latent Class Scaling Analysis** *Dayton*
27. **Sorting Data: Collection and Analysis** *Coxon*
28. **Analyzing Documentary Accounts** *Hodson*
29. **Effect Size for ANOVA Designs** *Cortina/Nouri*
30. **Nonparametric Simple Regression: Smoothing Scatterplots** *Fox*

31. **Multiple and Generalized Nonparametric Regression** *Fox*
32. **Logistic Regression: A Primer** *Pampel*
33. **Translating Questionnaires and Other Research Instruments: Problems and Solutions** *Behling/Law*
34. **Generalized Linear Models: A Unified Approach, 2nd Edition** *Gill/Torres*
35. **Interaction Effects in Logistic Regression** *Jaccard*
36. **Missing Data** *Allison*
37. **Spline Regression Models** *Marsh/Cormier*
38. **Logit and Probit: Ordered and Multinomial Models** *Borooah*
39. **Correlation: Parametric and Nonparametric Measures** *Chen/Popovich*
40. **Confidence Intervals** *Smithson*
41. **Internet Data Collection** *Best/Krueger*
42. **Probability Theory** *Rudas*
43. **Multilevel Modeling, 2nd Edition** *Luke*
44. **Polytomous Item Response Theory Models** *Ostini/Nering*
45. **An Introduction to Generalized Linear Models** *Dunteman/Ho*
46. **Logistic Regression Models for Ordinal Response Variables** *O'Connell*
47. **Fuzzy Set Theory: Applications in the Social Sciences** *Smithson/Verkuilen*
48. **Multiple Time Series Models** *Brandt/Williams*
49. **Quantile Regression** *Hao/Naiman*
50. **Differential Equations: A Modeling Approach** *Brown*
51. **Graph Algebra: Mathematical Modeling With a Systems Approach** *Brown*
52. **Modern Methods for Robust Regression** *Andersen*
53. **Agent-Based Models, 2nd Edition** *Gilbert*
54. **Social Network Analysis, 3rd Edition** *Knoke/Yang*
55. **Spatial Regression Models, 2nd Edition** *Ward/Gleditsch*
56. **Mediation Analysis** *Iacobucci*
57. **Latent Growth Curve Modeling** *Preacher/Wichman/MacCallum/Briggs*
58. **Introduction to the Comparative Method With Boolean Algebra** *Caramani*
59. **A Mathematical Primer for Social Statistics** *Fox*

60. **Fixed Effects Regression Models** *Allison*
61. **Differential Item Functioning, 2nd Edition** *Osterlind/Everson*
62. **Quantitative Narrative Analysis** *Franzosi*
63. **Multiple Correspondence Analysis** *LeRoux/Rouanet*
64. **Association Models** *Wong*
65. **Fractal Analysis** *Brown/Liebovitch*
66. **Assessing Inequality** *Hao/Naiman*
67. **Graphical Models and the Multigraph Representation for Categorical Data** *Khamis*
68. **Nonrecursive Models** *Paxton/Hipp/ Marquart-Pyatt*
69. **Ordinal Item Response Theory** *Van Schuur*
70. **Multivariate General Linear Models** *Haase*
71. **Methods of Randomization in Experimental Design** *Alferes*
72. **Heteroskedasticity in Regression** *Kaufman*
73. **An Introduction to Exponential Random Graph Modeling** *Harris*
74. **Introduction to Time Series Analysis** *Pickup*
75. **Factorial Survey Experiments** *Auspurg/Hinz*
76. **Introduction to Power Analysis: Two-Group Studies** *Hedberg*
77. **Linear Regression: A Mathematical Introduction** *Gujarati*
78. **Propensity Score Methods and Applications** *Bai/Clark*
79. **Multilevel Structural Equation Modeling** *Silva/Bosancianu/Littvay*
80. **Gathering Social Network Data** *adams*
81. **Generalized Linear Models for Bounded and Limited Quantitative Variables,** *Smithson and Shou*
82. **Exploratory Factor Analysis,** *Finch*
83. **Multidimensional Item Response Theory,** *Bonifay*



Sara Miller McCune founded SAGE Publishing in 1965 to support the dissemination of usable knowledge and educate a global community. SAGE publishes more than 1000 journals and over 800 new books each year, spanning a wide range of subject areas. Our growing selection of library products includes archives, data, case studies and video. SAGE remains majority owned by our founder and after her lifetime will become owned by a charitable trust that secures the company's continued independence.

Los Angeles | London | New Delhi | Singapore | Washington DC | Melbourne

# AGENT-BASED MODELS

Second Edition

Nigel Gilbert

*University of Surrey, United Kingdom*



Los Angeles | London | New Delhi  
Singapore | Washington DC | Melbourne

Los Angeles

London

New Delhi

Singapore

Washington DC

Melbourne

Copyright © 2020 by SAGE Publications, Inc.

All rights reserved. Except as permitted by U.S. copyright law, no part of this work may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without permission in writing from the publisher.

All third party trademarks referenced or depicted herein are included solely for the purpose of illustration and are the property of their respective owners. Reference to these trademarks in no way indicates any relationship with, or endorsement by, the trademark owner.



FOR INFORMATION:

SAGE Publications, Inc.

2455 Teller Road

Thousand Oaks, California 91320

E-mail: [order@sagepub.com](mailto:order@sagepub.com)

SAGE Publications Ltd.

1 Oliver's Yard

55 City Road

London, EC1Y 1SP

United Kingdom

SAGE Publications India Pvt. Ltd.

B 1/I 1 Mohan Cooperative Industrial Area

Mathura Road, New Delhi 110 044

India

SAGE Publications Asia-Pacific Pte. Ltd.

18 Cross Street #10-10/11/12

China Square Central

Singapore 048423

ISBN: 9781506355603

Printed in the United States of America

This book is printed on acid-free paper.

Acquisitions Editor: Leah Fargotstein

Editorial Assistant: Claire Laminen

Production Editor: Gagan Mahindra

Copy Editor: Michelle Ponce

Typesetter: Hurix Digital

Proofreader: Barbara Coster

Indexer: Integra

Cover Designer: Candice Harman

Marketing Manager: Shari Countryman

# CONTENTS

[Series Editor's Introduction](#)

[Preface](#)

[Acknowledgments](#)

[About the Author](#)

[1: The Idea of Agent-Based Modeling](#)

[1.1 Agent-Based Modeling](#)

[1.1.1 A Computational Method](#)

[1.1.2 Experiments](#)

[1.1.3 Models](#)

[1.1.4 Agents](#)

[1.1.5 The Environment](#)

[1.2 Some Examples](#)

[1.2.1 Urban Models](#)

[1.2.2 Opinion Dynamics](#)

[1.2.3 Consumer Behavior](#)

[1.2.4 Industrial Networks](#)

[1.2.5 Supply Chain Management](#)

[1.2.6 Electricity Markets](#)

[1.2.7 Modeling Policy](#)

[1.2.8 Participative and Companion Modeling](#)

[1.3 The Features of Agent-Based Modeling](#)

[1.3.1 Ontological Correspondence](#)

[1.3.2 Heterogeneous Agents](#)

[1.3.3 Representation of the Environment](#)

[1.3.4 Agent Interactions](#)

[1.3.5 Bounded Rationality](#)

[1.3.6 Learning](#)

[1.4 Other Related Modeling Approaches](#)

[1.4.1 Microsimulation](#)

[1.4.2 System Dynamics](#)

[1.4.3 Discrete Event Simulation](#)

[2: Agents, Environments, and Timescales](#)

[2.1 Agents](#)

[2.1.1 Agents as Objects](#)

[2.1.2 Production Rule Systems](#)

[2.1.3 Agents That Learn](#)

[2.1.4 Cognitive Models](#)

[2.2 Environments](#)

[2.2.1 Features of Environments](#)

[2.2.2 Geography](#)

[2.3 Randomness](#)

[2.4 Time](#)

[2.5 Population Learning](#)

[3: Designing an Agent-Based Model](#)

[3.1 Design Steps](#)

[3.2 An Example of Developing an Agent-Based Model](#)

[3.2.1 Macrolevel Features and Patterns](#)

[3.2.2 Microlevel Behavior](#)

[3.2.3 Designing a Model](#)

[4: Developing an Agent-Based Model](#)

[4.1 Modeling Toolkits, Libraries, Languages, Frameworks, and Environments](#)

[4.2 Using NetLogo to Build Models](#)

[4.3 Building the Collectivities Model Step by Step](#)

[4.4 Verification: Getting Rid of the Bugs](#)

[4.5 Validation](#)

[4.5.1 Abstract Models](#)

[4.5.2 Middle-Range Models](#)

[4.5.3 Facsimile Models](#)

[4.5.4 Complexity](#)

[4.6 Techniques for Validation](#)

[4.6.1 Comparing Theory and the Model: Sensitivity Analysis](#)

[4.6.2 Comparing the Model and Empirical Data](#)

[Appendix: The Features of Simulation Libraries and Environments](#)

[5: Using Agent-Based Models](#)

[5.1 Planning an Agent-Based Modeling Project](#)

[5.2 Reporting Agent-Based Model Research](#)

[5.3 Agent-Based Models for Public Policy](#)

[Resources](#)

[Societies and Associations](#)

[Journals](#)

[Mailing List and Web Sites](#)

[Glossary](#)

[References](#)

[Index](#)



# SERIES EDITOR'S INTRODUCTION

Almost 50 years ago Thomas Schelling published the first agent-based model (ABM) in the social sciences. It showed how relatively modest residential preferences on the part of individual households could result in marked patterns of neighborhood residential segregation. Since then, and especially recently, applications have blossomed in many fields ranging from opinion dynamics to supply chain management, from language evolution to disease epidemiology, from consumer behavior to urban planning. The second edition of *Introduction to Agent-Based Models* targets this broad audience. The author, Nigel Gilbert, is one of the founders of computational social science and an authority on agent-based models.

As Professor Gilbert defines it, agent-based modeling is “a computational method that enables a researcher to create, analyze, and experiment with models composed of agents that interact within the environment.” ABMs range from highly abstract simplified models to facsimile models that attempt to replicate real observations. They explicitly link micro and macro levels of analysis, as illustrated by Schelling’s model of households and neighborhoods. Because agent-based models incorporate dynamic interdependencies among the individual agents, the consequences for macrolevel change in these models are emergent, frequently nonlinear, and sometimes surprising, as was the case with Schelling’s model.

Like the first edition, the second edition of *Introduction to Agent-Based Models* is for beginners. It is suitable as a supplement for undergraduate as well as graduate courses in formal models, simulation, and computational social science; it is also a quick first introduction for any interested social science practitioner. The author carefully defines concepts, outlines the steps involved in planning, building, and reporting ABMs, and includes a helpful glossary. Readers are shown how to use the NetLogo modeling environment, freely available to students, teachers, and researchers worldwide, to

build and run a simple ABM. NetLogo helps readers get their feet wet, even those with little background in coding. The second edition of *Introduction to Agent-Based Models* retains the strengths of the first but updates the material, expands the coverage of verification, validation, and documentation, and addresses some new topics such as the use of ABMs to inform public policy. As was true for the first edition, the goal is to make readers better consumers of published ABMs and to provide the foundation for them ultimately to be creators of these models.

Agent-based modeling is a fast-moving area, especially in breadth of application. In addition, ABMs are increasingly a focus of interdisciplinary collaboration, between social/behavioral scientists from different disciplines (e.g., sociology and geography), between social/behavioral science and natural science (e.g., environmental science), and between social/behavioral science and computer science. Depending on purpose, the rules central to agent-based models can be derived from theory, past empirical research, and/or conversations with local experts. Indeed, ABMs are increasingly used in community-based participatory research. Given these trends, the need for a generally accessible primer is even greater now than when the first edition was published in 2007. This second edition fully satisfies that need.

Barbara Entwisle

*Series Editor*

# PREFACE

Agent-based modeling is a form of computational simulation. Although simulation as a research technique has had a very important part to play in the natural sciences for decades in disciplines from astronomy to biochemistry, it was relatively neglected in the social sciences. This may have been because a computational approach that respected the particular needs of the social sciences was lacking. However, in the early 1990s the value of agent-based modeling began to be realized, and, since then, the number of studies that have used agent-based modeling has grown rapidly (Hauke, Lorscheid, & Meyer, 2017).

Agent-based modeling is particularly suited to topics where understanding processes and their consequences is important. In essence, one creates a computer program in which the actors are represented by segments of program code, and then runs the program, observing what it does over the course of simulated time. There is a direct correspondence between the actors being modeled and the agents in the program, which makes the method intuitively appealing, especially to those brought up in a generation used to computer games. Nevertheless, agent-based modeling stands beside mathematical and statistical modeling in terms of its rigor. Like equation-based modeling, but unlike prose, agent-based models must be complete, consistent, and unambiguous if they are to be capable of being executed on a computer. On the other hand, unlike most mathematical models, agent-based models can include agents that are heterogeneous in their features and abilities, can model situations that are far from equilibrium, and can deal directly with the consequences of interaction between agents.

Because it is a new approach, there are few courses yet available to teach the skills of agent-based modeling, although the number is increasing, and there are few texts directed specifically at the interested social scientist. This short book introduces the subject, emphasizing the decisions that a social scientist needs to make when

selecting agent-based modeling as an appropriate method, and offering some tips on how to proceed. It is aimed at practicing social scientists and graduate students. It has been used as the recommended reading on agent-based modeling for a graduate-level module or doctoral program in computational social science, and it is also suitable as background reading in postgraduate courses on advanced social research methods. It would be a good preparation for any of the textbooks that provide a more in-depth guide to agent-based modeling (e.g., Hamill & Gilbert, 2015; Heppenstall, Crooks, See, & Batty, 2012; O'Sullivan & Perry, 2013; Railsback & Grimm, 2012; Squazzoni, 2012; Wilensky & Rand, 2015).

A knowledge of and experience with computer programming in any language would be helpful but is not essential to understand the book.

The book concludes with a list of printed and Web resources, a glossary, and a reference section. (The glossary terms will appear in bold at first use in the text.) Because the field is growing so rapidly, it has been possible to mention only a few examples of current research and some textbooks that provide more detail on some topics. **There is much more that could have been cited if there had been space.** In particular, the book mentions only briefly two closely linked areas: network models and game theory models, both of which are covered in much more detail in other SAGE volumes such as Knoke and Yang (2008) and Fink, Gates, and Humes (1998).

A website to accompany the book at [study.sagepub.com/researchmethods/qass/gilbert-agent-based-models-2e](http://study.sagepub.com/researchmethods/qass/gilbert-agent-based-models-2e) includes an annotated exemplar model using NetLogo.

## Acknowledgments

This book is born of some 25 years of building agent-based models, both large and small, and in domains ranging from science policy to anthropology. What I know about agent-based modeling has benefited immeasurably from the advice and companionship of many, including Andrew Abbott, Petra Ahrweiler, David Anzola, Robert Axelrod, Rob Axtell, Pete Barbrook-Johnson, Riccardo Boero, François Bousquet, Cristiano Castelfranchi, Edmund Chattoe, Claudio Cioffi-Revilla, Rosaria Conte, Guillaume Deffuant, Bruce Edmonds, Gusz Eiben, Corinna Elsenbroich, Lynne Hamill, Samer Hassan, Wander Jager, David Lane, Scott Moss, Kavin Narasimhan, Gilbert Peffer, Alex Penn, Andreas Pyka, Juliette Rouchier, Mauricio Salgado, Stephan Schuster, Flaminio Squazzoni, Luc Steels, Klaus Troitzsch, Paul Vogt, and Lu Yang. I thank Riccardo Boero, Lars-Eric Cederman, Lynne Hamill, Luis R. Izquierdo, Ken Kahn, Tim Liao, Michael Macy, Lu Yang, and eight anonymous reviewers for their detailed and constructive comments on drafts of the manuscripts for the first and second editions.

SAGE Publishing would like to thank the following reviewers for their feedback on the revision:

*Andrew Crooks, George Mason University*

*Sally Jackson, University of Illinois at Urbana-Champaign*

*James Nolan, University of Saskatchewan*

*Oleg Smirnov, Stony Brook University*

*Garry Sotnik, Portland State University*

## ABOUT THE AUTHOR

**Nigel Gilbert** is professor of sociology at the University of Surrey, Guildford, England. He is the author or editor of 34 books and many academic papers, and was the founding editor of the *Journal of Artificial Societies and Social Simulation*. His current research focuses on the application of agent-based models to understanding social and economic phenomena, especially the emergence of norms, culture, and innovation. He obtained a doctorate in the sociology of scientific knowledge in 1974 from the University of Cambridge and has subsequently taught at the universities of York and Surrey in England. He is one of the pioneers in the field of social simulation and is past president of the European Social Simulation Association. He is a Fellow of the UK Academy of Social Sciences and of the Royal Academy of Engineering.

# CHAPTER 1 THE IDEA OF AGENT-BASED MODELING

This short book explains what agent-based modeling is. It warns of some dangers and describes typical ways of doing agent-based modeling. And it offers a range of examples from many of the social sciences.

This first chapter begins with a brief overview of agent-based modeling before contrasting it with other, perhaps more familiar forms of modeling and describing several examples of current agent-based modeling research. [Chapter 2](#) goes into more detail, considering a range of methodological and theoretical issues and explaining what an **agent** is. [Chapter 3](#) dives into the specifics, showing with a simple example how one can design an agent-based model. [Chapter 4](#) provides some practical advice about developing, verifying, and validating agent-based models. Finally, [Chapter 5](#) discusses planning an agent-based modeling project, publishing the results, and applying agent-based modeling to help formulate and evaluate social and economic policies. The book concludes with a list of resources useful to agent-based modelers on the Web and in print.

Agent-based simulation has become increasingly popular as a modeling approach in the social sciences because it enables one to build models where individual entities and their interactions are directly represented. **In comparison with variable-based approaches using structural equations, or system-based approaches using differential equations, agent-based simulation offers the possibility of modeling individual heterogeneity, representing explicitly agents' decision rules, and situating agents in a geographical or another type of space.** It allows modelers to represent in a natural way multiple scales of analysis, the emergence of structures at the macro or societal level from individual action, and various kinds of adaptation and learning, none of which is easy to do with other modeling

approaches.



## 1.1 Agent-Based Modeling

Formally, agent-based modeling is (1) a computational method that enables a researcher to create, analyze, and (2) experiment with (3) **models** composed of (4) agents that interact within (5) the **environment**. Let us consider each of the five terms in this definition.

### 1.1.1 A Computational Method

First, agent-based modeling is a form of computational social science. That is, it involves building models that are computer programs. The idea of modeling is familiar in most of the social sciences: **One creates** a simplified representation of social reality that serves to express as clearly as possible the way in which one believes that reality operates. For example, **if one has** a dependent variable and one or more independent variables, a regression equation serves as a model of the relationship between the variables. A network of nodes and edges can model a set of friendships. Even an ordinary language description of a relationship, such as that between the strength of protection of intellectual property rights and the degree of innovation in a country, can be considered a model, albeit a simple and rather unformalized one.

Computational models are formulated as computer programs in which there are some inputs (somewhat like independent variables) and some outputs (like dependent variables). The program itself represents the processes that are thought to exist in the social world (Macy & Willer, 2002). For example, we might have a theory about how friends influence the purchasing choices that consumers make. As we will see, we can create a program in which there are individuals (or agents) that buy according to their preferences. The outcome is interesting because what one agent buys will influence the purchasing of a friend, and what the friend buys will influence the first agent. This kind of mutual reinforcement is relatively easy to model using agent-

based modeling.

One of the advantages of computational modeling is that it forces one to be precise: Unlike theories and models expressed in natural language, a computer program must be completely and exactly specified if it is to run. Another advantage is that it is often relatively easy to model theories about processes, **because programs are all about making things within the computer change**. If the idea of constructing computational models reminds you of computer games, especially the kind where the player has a virtual world to build, such as The Sims (<https://www.ea.com/games/the-sims/>), that is no accident. Such games can be very close to computational modeling, although to make them fun, designers often give them fancier graphics and less social theory than they do agent-based models.

### 1.1.2 Experiments

Whereas in physics and chemistry and some parts of biology, experimentation is the standard method of doing science, in most of the social sciences conducting experiments is still rare. An experiment consists of applying some **treatment** to an isolated system and observing what happens. The treated system is compared with another otherwise equivalent system that receives no treatment (the **control**). The great advantage of experiments is that they allow one to be sure that it is the treatment that is causing the observed effects, because it is only the treatment that differs between the treated and the control systems and the systems are isolated from other potential causes of change. However, with social systems isolation is generally impossible, and treating one system while not treating the control is often ethically undesirable. Therefore, it is not surprising that most social scientists do not often use experiments, despite the potential clarity of their results.

A major advantage of agent-based modeling is that the difficulties in ensuring isolation of the human system and the ethical problems of experimentation are not present when one does experiments on

virtual or computational systems. An experiment can be set up and repeated many times, using a range of parameters or allowing some factors to vary randomly. Of course, carrying out experiments with a computational model of some social phenomenon will yield interesting results only if the model behaves in the same way as the human system or, in other words, if the model is a good one, and one may not know whether that is the case. So, experimentation on models is not a panacea.

The idea of experimenting on models rather than on the real system is not novel. For example, when architects put a model tower block in a wind tunnel to investigate its behavior in high winds, they are experimenting on the model for just the same reasons as social scientists might want to experiment on their models: the cost of experimenting on a real tower block is too high. Another reason for experimenting with models is that this may be the only way to obtain results. Deriving the behavior of a model analytically is usually best because it provides information about how the model will behave given a range of inputs, but often an analytical solution is not possible. In these cases, it is necessary to experiment with different inputs to see how the model behaves. The model is used to **simulate** the real world as it might be in a variety of circumstances.

### 1.1.3 Models

Computational social science is based on the idea of constructing models and then using them to understand the social world (Gilbert, 2010). Models have a long history in the social sciences—much longer than the history of using computers—but came to the fore when statistical methods began to be used to analyze large amounts of quantitative data in economics and demography. A model is intended to represent or simulate some real, existing phenomenon, and this is called the **target** of the model. The two main advantages of a model are that it succinctly expresses the relationships between features of the target, and that it allows one to discover things about the target by investigating the model (Epstein, 2008).

One of the earliest well-known social science models is the Phillips (1950) hydraulic model of the economy in which water flowing through interconnected glass pipes and vessels is used to represent the circulation of money. One version of this model can still be admired at the Science Museum, London ([http://en.wikipedia.org/wiki/MONIAC\\_Computer](http://en.wikipedia.org/wiki/MONIAC_Computer)). The effect of changing parameters such as the interest rate can be investigated by changing the rate of flow of water through the pipes.

Models come in several varieties, and it is worth listing some of these to clarify the differences among them:

---

**Scale models** are smaller versions of the target. Together with the reduction in size is a systematic reduction in the level of detail or complexity of the model. So, for example, a scale model of an airplane will be the same shape as its target, but probably would not show the electronic control systems or possibly even the engines of the real plane. Similarly, a scale model of a city will be much smaller than the real city and may model only two dimensions (the distances between buildings but not the heights of buildings, for instance). When drawing conclusions about the target by studying the model, one needs to remember that the results from the model will need to be scaled back up to the target's dimensions, and that it is possible that some of the features not modeled may affect the validity of the conclusions.

An **ideal-type** model is one in which some characteristics of the target are exaggerated to simplify the model. For example, an idealized model of a stock market may assume that information flows from one trader to another instantaneously, and an idealized model of traffic may assume that drivers never get lost. The idealization has the effect of removing one or more complicating factors from the model; if these have negligible effects on how the model works, the model will remain useful for drawing conclusions about the target.

**Analogical models** are based on drawing an analogy between some better-understood phenomenon and the target. The most famous example is the billiard ball model of atoms, but there are also social science examples such as the computer model of the mind (Boden, 1988; Piccinini & Bahar, 2013) and the garbage can model of organizations (March, Cohen, & Olsen, 1972). Such models are useful because well-established results from the analogy can be carried over and applied to the target, but of course the validity of these depends on the adequacy of the analogy.

---

These are not mutually exclusive categories. It is possible, and indeed common, for a model to be a scale model and an analogy (the hydraulic model of the economy mentioned above is such a combination, for instance).

Some models fall into a fourth category that is somewhat different, but also commonly encountered in the social sciences; these are often called mathematical or **equation-based models**. Examples are the structural equation models of quantitative sociology and the macroeconomic models of neoclassical economics. These models specify relationships between variables, but unlike models in the other three categories, they do not imply any kind of analogy or resemblance between the model and the target. Usually, the success of a mathematical model is indicated by the degree to which some data fit the equation, but the form of the equation itself is of little interest or consequence. For example, the Cobb-Douglas production function is a mathematical model of how manufactured outputs are related to inputs (Cobb & Douglas, 1928):

---

$$Y = AL^{\alpha}K^{\beta}$$

---

where  $Y$  = output;  $L$  = labor input;  $K$  = capital input; and  $A$ ,  $\alpha$ , and  $\beta$  are constants determined by technology. The form of this equation

was derived from statistical evidence, not by theorizing about the behavior of firms. Although mathematical models have been very successful in some parts of the social sciences in clarifying the relationships between variables, they are often not very useful in helping to understand *why* one variable is related to another; in other words, these models are not very helpful in expressing ideas about process and mechanism.

### 1.1.4 Agents

Agent-based models consist of agents that interact within an environment. Agents are either separate computer programs or, more commonly, distinct parts of a program that are used to represent social actors—individual people, organizations such as firms, or bodies such as nation-states. They are programmed to react to the computational environment in which they are located, where this environment is a model of the real environment in which the social actors operate.

As will be seen later, a crucial feature of agent-based models is that the agents can interact; that is, they can pass informational messages to each other and act on the basis of what they learn from these messages. The messages may represent spoken dialogue between people or more indirect means of information flow, such as the observation of another agent or the detection of the effects of another agent's actions. The possibility of modeling such agent-to-agent interactions is the main way in which agent-based modeling differs from other types of computational models.

### 1.1.5 The Environment

The environment is the virtual world in which the agents act. It may be an entirely neutral medium with little or no effect on the agents, or it may be as carefully crafted as the agents themselves. Commonly, environments represent geographical spaces, such as in models

concerning residential segregation, where the environment simulates some of the physical features of a city (e.g., Portugali, Benenson, & Omer, 2010), and in ecological models of the location of species (e.g., Watkins, Noble, Foster, Harmsen, & Doncaster, 2015). Models in which the environment represents a geographical space are called **spatially explicit**. In other models the environment could be a space, but one that represents not geography but some other feature. For example, firms in high technology areas can be modeled in knowledge space (Gilbert, Ahrweiler, & Pyka, 2014). In these spatial models the agents have coordinates to indicate their location. Another option is to have no spatial representation at all but to link agents together into a network in which the only indication of an agent's relationship to other agents is the list of the agents to which it is connected by network links. For example, Walbert, Caton, and Norgaard (2018) model the development of defense agreements and wars between countries. The agreements are represented by links forming a global network of countries.

To make these definitions somewhat more concrete, in the next section we introduce some examples of agent-based models in terms of these concepts.

## 1.2 Some Examples

Agent-based models are of value in most branches of social science. The models that are briefly described in the rest of this section have been chosen to illustrate the diversity of the problem areas where they have been used productively.

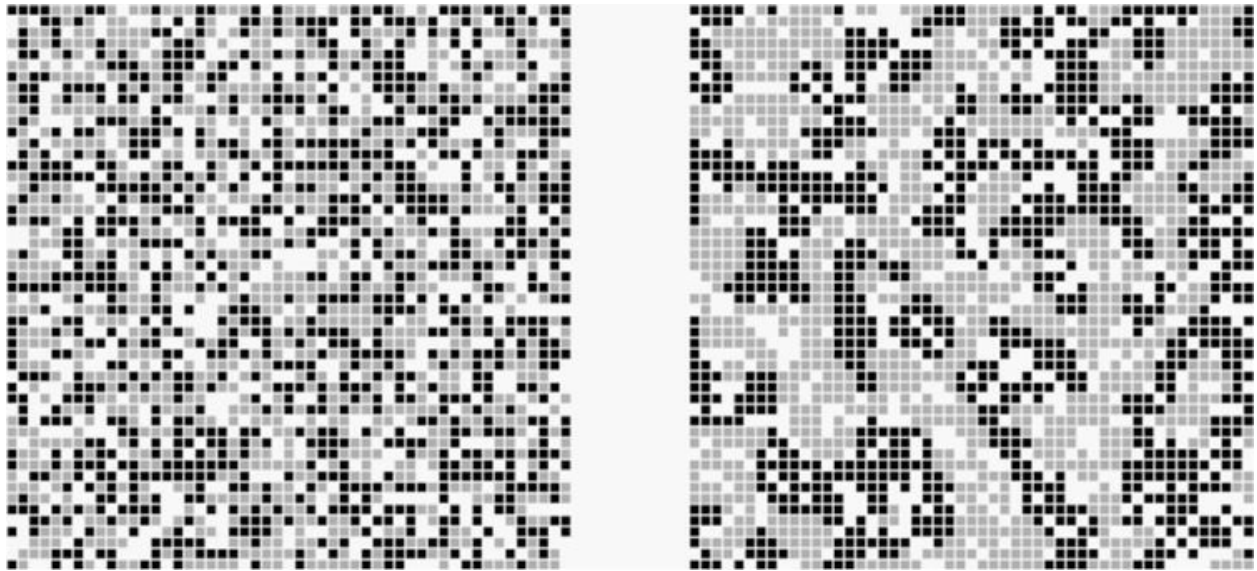
### 1.2.1 Urban Models

In 1971 Thomas Schelling (1971, 1978; see also Hegselmann, 2017; Sakoda, 1971) proposed a model to explain observed racial segregation in American cities. The model is a very abstract one as originally conceived, but it has been influential in work on understanding the persistence of segregation not only in the United States but also in urban centers in other countries. The model is based on a regular square grid of cells representing an urban area on which agents, representing households, are placed at random. The agents are of two kinds (we will call them reds and greens). Each cell can hold only one household agent at a time, and many cells are empty. At each time step, each household surveys its immediate neighbors (the eight cells surrounding it) and counts the fraction of households that are of the other color. If the fraction is greater than some constant threshold tolerance value (e.g., there are more than a fixed proportion of reds surrounding a green, or greens surrounding a red), that household considers itself to be unhappy and decides to relocate. It does so by moving to some vacant cell on the grid.

At the next time step, the newly positioned household may tip the balance of tolerance of its neighbors, causing some of them to become unhappy, which can result in a cascade of relocations. For levels of the tolerance threshold at or above about 0.3, an initially random distribution of households segregates into patches of red and green, with households of each color clustering together ([Figure 1.1](#)). The clustering occurs even when households tolerate living adjacent



to a majority of neighbors of the other color; Schelling interpreted this as indicating that even quite low degrees of racial prejudice could yield the strongly segregated patterns typical of U.S. cities in the 1970s.



**Figure 1.1** The Schelling Model at the Start (Left) and After Equilibrium Has Been Reached (Right), With a Uniform Tolerance of 0.3

SOURCE: Wilensky, U. (1998). NetLogo Segregation model. <http://ccl.northwestern.edu/netlogo/models/Segregation>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

The Schelling model has been influential for several reasons (Batty, 2013). First, the outcome—clusters of households of the same color—is surprising and not easily predictable just from considering the individual agents' movement rule. Second, the model is very simple and has only one parameter, the tolerance threshold. It is therefore easy to understand. Third, the emergent clustering behavior is rather robust. The same outcomes are obtained for a wide range of tolerance values, for a variety of movement rules (e.g., the household agent could select a new cell at random, or use a utility function to select the

most preferred cell, or take into account affordability if cells are priced, and so on), and for different rules about which neighbors to consider (e.g., those in the eight surrounding cells; in the four cells to the north, east, south, and west; or in a larger ring two or more cells away) (Gilbert, 2002). Fourth, the model immediately suggests how it could be tested with empirical data (Benard & Willer, 2007; Benenson & Hatna, 2009; Clark, 1991; Fossett & Waren, 2005; Hatna & Benenson, 2012; Laurie & Jaggi, 2003; Mahdavi Ardestani, O'Sullivan, & Davis, 2018; Pollicott & Weiss, 2001; Sander, Schreiber, & Doherty, 2000; Zhang, 2004), although in practice it has proved quite difficult to obtain reliable and extensive data on household location preferences to calculate ratings of unhappiness. The advantages of the Schelling model over others that had been previously proposed, which were based on equations relating migration flows and the relative values of residential properties (e.g., O'Kelly & Fotheringham, 1989), are that the number of parameters to be estimated is lower and that it is simple to simulate and analyze the model. Current work has focused on making the model more concrete, replacing the abstract square grid with actual urban geographies, and adding further factors, such as the affordability of the locations to which households want to move.

The Schelling model is one example of a class of models that are concerned with changing land use and mobility. There are many examples of models that are concerned with investigating the implications of changes to the landscape (Gotts, Matthews, Gilbert, Polhill, & Roach, 2007), for policy analysis and planning, and for understanding the spatial patterns of land use. Traffic simulation agent-based models are also increasingly being used for planning improvements to roads to improve roadside air quality or reduce congestion (e.g., POLARIS [Argonne National Laboratory, 2018]).

### **1.2.2 Opinion Dynamics**

Another interesting group of models with potentially important implications is concerned with understanding the development of political opinions, such as with explaining the spread of extremist

opinions within a population. We will review just one such study, although there are several that explore the consequences of different assumptions and opinion transmission mechanisms (e.g., Afshar & Asadpour, 2010; Anderson & Ye, 2019; Deffuant, 2006; Deffuant, Amblard, & Weisbuch, 2002; Flache et al., 2017; Krause & Hegselmann, 2002; Kurahashi-Nakamura, Mäs, & Lorenz, 2016; Lorenz, 2006; McKeown & Sheehy, 2006; Stauffer, Sousa, & Schulze, 2004; Stefanelli & Seidl, 2017). Deffuant and colleagues (2002) ask,

---

How can opinions, which are initially considered as extreme and marginal, manage to become the norm in large parts of a population? Several examples in world history show that large communities can more or less suddenly switch globally to one extreme opinion, because of the influence of an initially small minority. Germany in the thirties is a particularly dramatic example of such a process. In the last decades, an initial minority of radical Islamists managed to convince large populations in Middle East countries. But one can think of less dramatic processes, like fashion for instance, where the behavior of minority groups, once considered as extremist, becomes the norm in a large part of the population (it is the case of some gay way of dressing for instance). On the other hand, one can also find many examples where a very strong bipolarization of the population takes place, for instance the Cold War period in Europe. In these cases, the whole population becomes extremist (choosing one side or the other). (Deffuant et al., 2002)

---

In Deffuant and colleagues' (2002) model, agents have an opinion (a real number between  $-1$  and  $+1$ ) and a degree of doubt about their opinion, called uncertainty (a positive real number). An agent's opinion segment is defined as the band centered on the agent's opinion, spreading to the right and left by the agent's value for uncertainty. Agents meet at random; that is, one agent is chosen from the pool of agents and this one interacts with another agent, also chosen

randomly from the remaining agents. When they meet, one agent may influence the other if their opinion segments overlap. If the opinion segments do not overlap, the agents are assumed to be so different in their opinions that they have no chance of influencing each other. If an agent does influence another, the opinion of one agent (agent  $j$ ) is affected by the opinion of another agent (agent  $i$ ) by an amount proportional to the difference between their opinions, multiplied by the amount of overlap divided by agent  $i$ 's uncertainty, minus one. The effect of this formula is that very uncertain agents influence other agents less than those that are certain (for full details, see Deffuant et al., 2002, Equations 1 to 6).

Every agent starts with an opinion taken from a uniform random distribution and with a common level of uncertainty—with the exception of a few extremists, those who have the most positive or negative opinions. The latter are given a low level of uncertainty: the extremists are assumed to be rather certain about their extreme opinions. Under these conditions, extremism spreads, and eventually the simulation reaches a steady state with all agents joining the extremists at one or the other end of the opinion continuum. Restarting the simulation without the politically certain extremists, the population converges instead so that all agents share a middle view. Thus, the model shows that a few extremists with opinions that are not open to influence from other agents can have a dramatic effect on the opinions of the majority. This work has some implications for the development of terrorist movements, where a few extremists have been able to recruit support from substantial proportions of the wider population.

### **1.2.3 Consumer Behavior**

Businesses are naturally keen to understand what influences their customers to buy their products. Although the intrinsic qualities of the product are usually important, so are the influence of friends and families, advertising, fashion, and a range of other social factors. To examine the often-complex interactions between these, some

researchers have started to use agent-based models in which the agents represent consumers. Among the first to report such a model were Jager (2017) and Janssen and Jager (1999), who explored the processes leading to lock-in in consumer markets. Lock-in occurs when one among several competing products achieves dominance so that it is difficult for individual consumers to switch to rival products. Commonly cited examples are VHS videotapes (dominating Betamax), the QWERTY keyboard (dominating other arrangements of the keys), social media platforms such as Facebook, and so on. Janssen and Jager focus on the behavioral processes that lead to lock-in, and therefore they give their agents, which they call consumats, decision rules that are psychologically plausible and carefully justified in terms of behavioral theories of, for example, social comparison and imitation.

An example of using the consumat approach is work by Kangur, Jager, Verbrugge, and Bockarjova (2017) on the diffusion of hybrid and electric cars. Using detailed data collected from 1,795 respondents on their driving behavior, attitudes about how long batteries took to charge, battery range, environmental concern, social susceptibility, prior involvement with purchasing cars, and so on, a population of consumat agents was constructed, one for each respondent. Experiments with the model showed that effective policy requires the implementation of a combination of monetary, structural, and informational measures over a long period, and that the strongest effect on emission reduction was obtained with a policy that supported electric cars, but not hybrids.

Another example of modeling consumers is a study by Izquierdo and Izquierdo (2007) in which the authors consider markets such as the secondhand car market, where there is quality variability (different quality for different items) and quality uncertainty (it is difficult to know the quality of an item before buying it and using it). The study explores how quality variability can damage a market and affect consumer confidence. There are two agent roles: buyer and seller. Sellers sell products by calculating the minimum price they will accept, and buyers buy products by offering a price based on the expected quality of the

product. The expected quality is based on experience, either just of the agent or accumulated from the agent's peers over its social network. There are a finite number of products in the system, buyers and sellers perform one deal per round, and the market is cleared every round as these deals are done. The social network is created by connecting pairs of agents at random, with a parameter used to adjust the number of connections, from completely connected to completely unconnected.

The authors found that without a social network consumer confidence fell to the point where the market was no longer viable, whereas with a social network the aggregation of the agent's own experience and the more positive collective experience of others (which is not as volatile) helped to maintain the market's stability. This shows how social information can aggregate group experience to a more accurate level and so reduce the importance of a single individual's bad experiences, as exemplified in the customer reviews provided by Amazon, eBay, and other online merchants.

### **1.2.4 Industrial Networks**

Most economic theory ignores the significance of links between firms, but there are many examples of industrial sectors where networks are of obvious importance. A well-known instance is the industrial districts of northern Italy, such as the textile production district, Prato. Industrial districts are characterized by large numbers of small firms clustered together in a small region, all manufacturing the same type of product, with strong, but varying, links between them. The links may be those between a supplier and a customer, a collaboration to share techniques, a financial link, or just a friendship or familial relationship (Albino, Giannoccaro, & Carbonara, 2003; Boero, Castellani, & Squazzoni, 2004; Borrelli, Ponsiglione, Zollo, & Iandoli, 2005; Brenner, 2001; El-Tawil, Fang, Aguirre, & Best, 2017; Fioretti, 2001; Squazzoni & Boero, 2002).

Another example is the innovation networks that are pervasive in

knowledge-intensive sectors such as biotechnology and information technology. The firms in these sectors are not always geographically clustered (although there tend to be concentrations in certain locations), but they do have strong networking relationships with other, similar firms, sharing knowledge, skills, and technology with them.

For example, Gilbert, Pyka, & Ahrweiler (2001) developed a model of innovation networks in which agents have genes that symbolize their stock of knowledge and expertise. The genes are used to develop new products that are marketed to other firms in the model. However, a product can be produced only if its components are available to be purchased from other firms, and if some firm wants to buy it. Thus, at one level the model is one of an industrial market with firms trading between each other. In addition, firms can improve their genes either through internal research and development or through incorporating knowledge obtained from other firms by collaborative arrangements. The improved knowledge can be used to produce products that may sell better or require fewer or more cheaply available components. At this level, the model resembles a population that can learn through a type of natural selection (see Section 2.5) in which firms that cannot find a customer cease trading, whereas the fittest firms survive, collaborate with other firms, and produce spin-offs that incorporate the best aspects of their parents (Ahrweiler, Pyka, & Gilbert, 2011; Ahrweiler, Schilperoord, Pyka, & Gilbert, 2014, 2015; Pyka, Ahrweiler, & Gilbert, 2004; Watts & Gilbert, 2014b). For another example, see Pajares, Hernández-Iglesias, & López-Paredes (2004).

### **1.2.5 Supply Chain Management**

Manufacturers normally buy components from other organizations and sell their products to distributors, who then sell to retailers. Eventually, the product reaches the user, who may not realize the complex interorganizational relationships that have had to be coordinated to deliver the product. Maximizing the efficiency of the supply chain linking businesses is increasingly important and increasingly difficult as products involve more parts, drawn more widely from around the

world, and as managers attempt to reduce inventory and increase the availability of goods. Modeling supply chains is a good way of studying order fulfillment processes and investigating the effectiveness of management policies, and agent-based models are increasingly being used for this purpose.

An agent-based model fits well with the task of simulating supply chains because the businesses involved can be modeled as agents, each with its own inventory rules. It is also easy to model the flow of products down the chain and the flow of information, such as order volumes and lead times, from one organization to another. This was the approach taken by Strader, Lin, and Shaw (1998), who described a model they built to study the impact of information sharing in divergent assembly supply chains. Divergent assembly supply chains are typical of the electronics and computer industries and are those in which a small number of suppliers provide materials and subcomponents (e.g., electronic devices) that are used to assemble a range of generic products (e.g., hard disk drives) that are then used to build customized products at distribution sites (e.g., personal computers). Strader and colleagues compared three order fulfillment policies: make-to-order, when production is triggered by customer orders; assembly-to-order, when components are built and held in stock and only the final assembly is triggered by an order; and make-to-stock, when production is driven by the stock level falling below a threshold. They also experimented with different amounts of information sharing between organizations and found that in the divergent assembly supply chains that they modeled, an assembly-to-order strategy, coupled with the sharing of both supply and demand information among organizations along the supply chain, was the most efficient. They also pointed out that their results reinforce the general point that information can substitute for inventory. If one has good information, uncertainty about demand can be reduced, and as a result the required inventory level to satisfy orders can also be reduced.

## **1.2.6 Electricity Markets**



In many developed countries, in recent years, the electricity supply has been privatized. It is now common for there to be two or three electricity utilities that sell power to several distributors that in turn sell the electricity to commercial and domestic users. The change from a monopoly state-owned or state-regulated supplier to one in which there are several supply firms bidding into a market has inspired a range of agent-based models that aim to anticipate the effect of market regulations; changes to the number and type of suppliers and purchasers; and policy changes intended, for example, to reduce the chance of blackouts or to decrease the environmental impact of generation (Bagnall & Smith, 2005; Batten & Grozev, 2006; Bunn & Oliveira, 2001; Guerri, Rastegar, & Cincotti, 2010; Koesrindartoto, Sun, & Tesfatsion, 2005; Ringler, Keles, & Fichtner, 2016).

In these models the agents are the supply companies that make offers to the simulated market to provide a certain quantity of electricity at a certain price for a period, such as a day or an hour. This is also how the real electricity markets work: Companies make offers to supply and the best offer is accepted (different markets have different rules about what is meant by the best offer). Usually, the demand varies continuously, so supply companies have a difficult job setting a price for the electricity that maximizes their profit. A further complication is that the cost of generation can be very nonlinear: Matching peak demand may mean starting up a power station that is used for only a few hours.

By running the simulation, one can study the conditions under which the market price comes down to near the marginal cost of generation; the effect of mergers that reduce the number of supply companies; and the consequences of having different types of market design, such as allowing futures trading. Most of the current models allow the agents to learn trading strategies using a technique known as **reinforcement learning (RL)** (see Section 2.1.3). A supply agent starts by making a bid using a pricing strategy selected at random from a set common to all the suppliers. If the bid is accepted and profitable, the value of this strategy is reinforced and the probability of using it again in similar circumstances is increased, or, if it is

unsuccessful, the chance of using it again is decreased (Sutton & Barto, 2018).

### 1.2.7 Modeling Policy

Computer models are increasingly being used in the formulation and evaluation of public **policy** by national, regional, and local governments and by companies to examine strategic options. For example, governments are regularly using models to assess the distributional impact of tax reforms (Sutherland & Figari, 2013); to guide policies aimed at decreasing air pollution (Ghazi, Khadir, & Dugdale, 2014); to plan new roads and transport policies (Bazzan & Klügl, 2014); to test vaccination strategies to guard against pandemics (Waldrop, 2017); and to quantify risk from flooding (Dubbelboer, Nikolic, Jenkins, & Hall, 2017). Some of these models are agent-based models, and agent-based modeling is likely to become more important in public policy making as it becomes better known among policy analysts.

The development and implementation of public policy is often described in terms of the ROAMEF cycle, which is a process of deciding the *Rationale* for a proposed policy, setting its *Objectives*, undertaking an *Appraisal* of options, implementing and *Monitoring* the policy, *Evaluating* its efficacy, and using *Feedback* to improve the policy. These stages are then repeated indefinitely. Although this description is not only very idealized compared with the messy reality of policy making, but has also been somewhat superseded by alternative approaches (Cairney, Heikkila, & Wood, 2019), it is helpful in indicating how and when modeling can be useful: at the rationale, appraisal, and evaluation stages. When developing a rationale for a policy, a model can be used to assess likely outcomes and to discover potential unintended consequences. At the appraisal stage models can be used to compare costs and benefits of alternative implementations and evaluate the resilience of each option—that is, how resistant the policy would be if the policy environment changes in unexpected ways. To evaluate the impact of a policy, one needs to

know what would have happened if the policy had not been implemented in order to compare this with the actual outcome with the policy. This means that one must have a counterfactual. Counterfactuals are sometimes observed by applying the policy only in some areas (or to only some people) and leaving the others unaffected. However, this is often difficult to do for practical or ethical reasons. Instead, a model can be used to simulate the policy domain without the policy, thus creating a virtual counterfactual to compare with the effect of the implemented policy.

Models are also useful in a policy context as a means of communication. Policy development typically involves many stakeholders—groups with an interest in the policy domain. For example, an agricultural policy might affect landowners, farmers, farm employees, food wholesalers and food retailers, those using rural areas for leisure or commercial activities, and so on, as well as politicians, trade unions, and environmental groups. Each of these is likely to have different expertise and different interests. Developing a model in conjunction with representatives of these stakeholders might help to clarify the differences of perspective and interest between them and help to improve the model.

### **1.2.8 Participative and Companion Modeling**

Agent-based models have been used with success in rural areas in developing countries to help with the management of scarce natural resources such as water for irrigation. This surprising use of agent-based models is due to their fit with participative (or participatory) research methods. As well as being used for research, agent-based models have been used as a support for negotiation and decision making and for training with, for example, farmers in Senegal (D'Aquino, Bousquet, Le Page, & Bah, 2003), foresters and farmers in the Massif Central in France (Étienne, 2003; Étienne, Cohen, & Le Page, 2003), and the inhabitants of an atoll in Kiribati in the South Pacific (Dray et al., 2006).

The approach, also called **companion modeling** (Barnaud, van Paassen, Trébuil, Promburom, & Bousquet, 2010; Barreteau, 2003; D'Aquino, Barreteau, & Le Page, 2003; Étienne, 2014; Ruankaew et al., 2010) involves building an agent-based system in close association with informants selected from the local people. As a preliminary, the informants may be interviewed about their understanding of the situation; they then engage in a role-playing game. Eventually, when enough knowledge has been gained, a computer model is created and used with the participants as a training aid or as a negotiation support, allowing the answering of what-if questions about possible decisions.

For example, Barreteau, Bousquet, and Attonaty (2001) describe the use of participative modeling in order to understand why an irrigation scheme in the Senegal River Valley had produced disappointing results. They developed both a role-playing game and an agent-based system called SHADOC to represent the interactions among the various stakeholders involved in making decisions about the allocation of water to arable plots in the irrigated area. In this instance, the agent-based model was developed first and then its main elements converted to a role-playing game (in which the players were the equivalent of the agents in the agent-based model), partly to validate the agent-based model, and partly because the role-playing game is easier to use in a rural environment. The authors sum up the value of this approach as “enhancing discussion among game participants” and enabling “the problems encountered in the field and known by each individual separately [to be] turned into common knowledge” (Barreteau et al., 2001).

## 1.3 The Features of Agent-Based Modeling

These examples, chosen to illustrate the spectrum of agent-based modeling now being undertaken, also provide examples of some characteristic features of agent-based modeling (Windrum, Fagiolo, & Moneta, 2007).

### 1.3.1 Ontological Correspondence

There can be a direct correspondence between the computational agents in the model and real-world actors, which makes it easier to design the model and interpret its outcome than would be the case with, for example, an equation-based model. For instance, a model of a commercial organization can include agents representing the employees, the customers, the suppliers, and any other significant actors. In each case the model might include an agent standing for the whole class (e.g., employees), or it might have a separate agent for each employee, depending on how important the differences between employees are. The models of electricity markets described above have agents for each of the main players in the market.

### 1.3.2 Heterogeneous Agents

Theories in economics and organization science make the simplifying assumption that all actors are identical or similar in most important respects. They deal, for example, with the typical firm, a representative agent, or the economically rational decision maker. Actors may differ in their preferences, but it is unusual to have agents that follow different rules of behavior; when this is allowed there may be only a small number of sets of such actors, each with its own behavior. This is for the good reason that, unless agents are homogeneous, analytical solutions are very difficult or impossible to find. A computational model avoids this limitation: Each agent can

operate according to its own preferences or even according to its own rules of action. An example is found in the models of supply chains, in which each business can have its own strategy for controlling inventory.

### **1.3.3 Representation of the Environment**

It is possible to represent the environment in which actors are acting directly in an agent-based model. This may include physical aspects (e.g., physical barriers and geographical hurdles that agents must overcome), the effects of other agents in the surrounding locality, and the influence of factors such as crowding and resource depletion. For example, Gimblett (2002) and colleagues have modeled the movement of backpackers in the Sierra Nevada in California to examine the effect of management policies in helping to maintain this area of wilderness. The agents simulated trekking in a landscape linked to a geographical information system that modeled the topology of the area. The environment also plays an important role in the models of industrial districts mentioned in the previous section.

### **1.3.4 Agent Interactions**

An important benefit of agent-based modeling is that interactions between agents can be simulated. At the simplest, these interactions can consist of the transfer of data from one agent to another, typically another agent located close by in the simulated environment. Where appropriate, the interaction can be much more complicated, involving the passing of messages composed in some language, with one agent constructing an utterance and the other interpreting it (and not necessarily deriving the same meaning from the utterance as the speaker intended). The opinion dynamics models (Section 1.2.2) are a good example of the importance of interactions in agent-based models.

### 1.3.5 Bounded Rationality

Many models implicitly assume that the individuals whom they model are rational—that is, that they act according to some reasonable set of rules to optimize their utility or welfare. (The alternative is to model agents as acting randomly or irrationally, in a way that will not optimize their welfare. Both have a place in some models.) Some economists, especially those using rational choice theory, have been accused of assuming that individuals are hyperrational—that is, that people engage in long chains of complex reasoning in order to select optimal courses of action, or even that people are capable of following chains of logic that extend indefinitely. Simon (1955), among others, criticized this as unrealistic and proposed that people should be modeled as **boundedly rational**—that is, as limited in their cognitive abilities and thus in the degree to which they are able to optimize their utility (Kahneman, 2003, 2011). Agent-based modeling makes it easy to create boundedly rational agents. In fact, the challenge is usually not to limit the rationality of agents but rather to extend their intelligence to the point where they could make decisions of the same sophistication as is commonplace among people. Nevertheless, it has been found in several contexts, such as in modeling stock markets, that the aggregate behavior of agents with very little rationality (or zero intelligence) matches the observed behavior at the macro level surprisingly well (Farmer, Patelli, & Zovko, 2005; Poggio, Lo, LeBaron, & Chan, 2001).

### 1.3.6 Learning

Agent-based models can simulate learning at both the individual and population levels. Learning can be modeled in any or all of three ways: as individual learning in which agents learn from their own experience; as population learning in which the set of agents learns because some agents “die” and are replaced by better agents, leading to improvements in the population average; and social learning, in which some agents imitate or are taught by other agents, leading to the sharing of experience gathered individually but distributed over the

whole population (Gilbert et al., 2006). The model of innovation networks summarized above is an example of a model incorporating learning: The individual innovating firms learn how to make better products, and because poorly performing firms become bankrupt to be replaced by better start-ups, the sector can learn to improve its performance.



## 1.4 Other Related Modeling Approaches

The previous section has reviewed some areas where agent-based models have been useful. However, agent-based models are not appropriate for every modeling task. Before starting a new project, it is worth considering the alternatives. This section introduces three styles of modeling used in the social sciences that stand comparison with agent-based modeling: microsimulation, system dynamics, and discrete event simulation.

### 1.4.1 Microsimulation

Microsimulation starts with a large database describing a sample of individuals, households, or organizations and then uses rules to update the sample members as though time were advancing. For example, the database might be derived from a representative national survey of households and include data on variables such as household members' age, sex, education level, income, employment, and pension arrangements. These data would relate to the specific period when the survey was carried out. Microsimulation allows one to ask what the sample would be like in the future. For example, one might want to know how many in the sample would be retired in 5 years' time and how this would affect the distribution of income. If we have some rules about the likely changes in individual circumstances during the year, these rules can be applied to every person in the sample to find what might have changed by the end of the first year after the survey. Then the same rules can be reapplied to yield the state of the sample after 2 years, and so on. After this aging process has been carried out, aggregate statistics can be calculated for the sample as a whole (e.g., the mean and variance of the distribution of income, which can be compared with the distribution at the time of the survey) and inferences made about what changes are to be expected in the population from which the sample was drawn (Li & O'Donoghue, 2013; O'Donoghue, 2014; Orcutt, Quinke, & Merz, 1986; Rutter,

Zaslavsky, & Feuer, 2011; Sutherland, Paulus, & Figari, 2014).

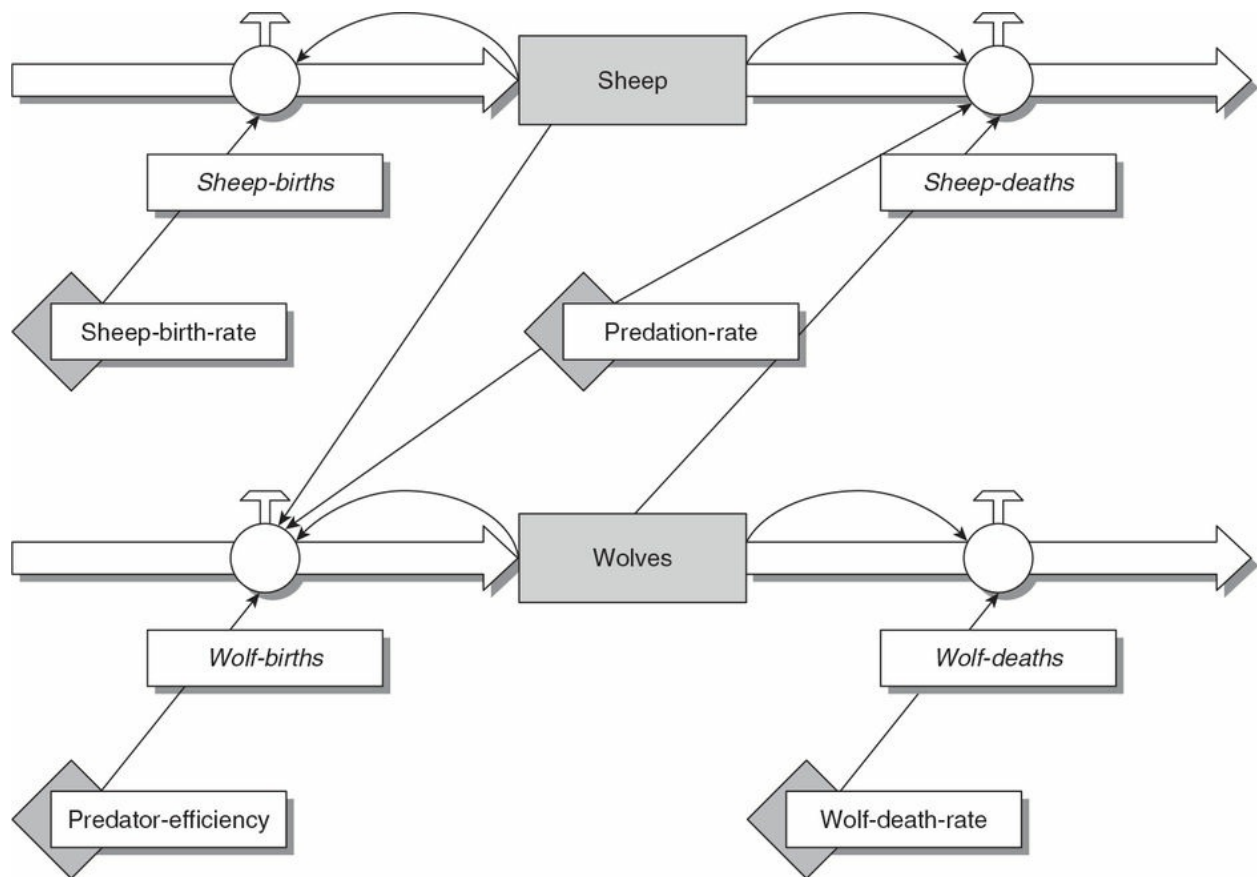
Microsimulation has been used to assess the distributional implications of changes in social security, personal tax, and pensions. For example, it can be helpful in evaluating the effects of changing the income threshold below which state benefits become payable and, more generally, on the effect of taxes on income inequality (Jara & Tumino, 2013; Sutherland & Figari, 2013). Experimental prototypes have also been developed in which there are several databases, describing not only individuals but also firms, and in which the aging process is affected not only by individual characteristics but also by macroeconomic variables such as inflation and the growth in gross domestic product (GDP).

An advantage of microsimulation models is that they start not from some hypothetical or randomly created set of agents but from a real sample, as described by a sample survey. Hence, it is relatively easy, in comparison with agent-based models, to read back from the results of the microsimulation to make predictions about the future state of a real population. There are two main disadvantages. First, the aging process requires very detailed transition matrices that specify the probability that an agent currently in some state will change to some other state in the following year. For example, one needs to know the probability that someone currently employed will become unemployed 1 year later. Moreover, because this transition probability will almost certainly differ between men and women, women with and without children, young people and old people, and so on, one needs not a single probability value but instead a matrix of conditional probabilities, one for each combination of individual circumstances. Obtaining reliable estimates of such transition matrices can be very difficult, requiring estimation from very large amounts of data. Second, each agent is aged individually and treated as though it is isolated in the world. Microsimulation does not allow for any interaction between agents and typically has no notion of space or geography. So, for instance, it is difficult to take account of the finding that the chances of getting a job if one is unemployed are lower if one lives in an area where the unemployment rate is high.

## 1.4.2 System Dynamics

In the system dynamics approach to modeling one creates a model that expresses the temporal cause-and-effect relationships between variables, but does not represent agents directly. One of the earliest and best-known examples is Forrester's model of the world, which was used to make predictions about future population levels, growing pollution, and rates of consumption of natural resources (Forrester, 1971). System dynamics, as its name implies, models systems of interacting variables and is able to handle direct causal links, such as a growth in population leading to increased depletion of resources; and feedback loops, as when population growth depends on the food supply, but food supply depends on the level of the population (Sterman, 2000).

It is often convenient to represent a system dynamics model with a diagram in which arrows represent the causal links between variables. [Figure 1.2](#) shows a typical, although simple, model of an ecosystem in which sheep breed in proportion to their population, wolves eat the sheep, but if there are too few sheep, the wolves starve. The rectangular boxes represent the stocks of sheep and wolves, the tap-like symbols are flows into and out of the stocks, and the diamond shapes are variables that control the rate of flow. The population of sheep increases as sheep are born, and the rate at which this happens is determined by the constant sheep-birth-rate. The diagram shows that sheep die at a rate that is a function of the number of sheep living (the curved arrow from the stock of sheep to the flow control labeled sheep-deaths), the probability that a wolf will catch a sheep (the arrow from the predation-rate variable), and the number of wolves (the arrow from the stock of wolves). Although this illustrative model is concerned with somewhat imaginary wolves and sheep, similar models can be constructed for topics of sociological interest, such as the number of illegal drug users and enforcement agents, public health epidemics, and the spread of crime (Hirsch & Homer, 2006; Jacobsen & Hanneman, 1992; McMillon, Simon, & Morenoff, 2014).



## Description

**Figure 1.2** A System Dynamics Model of a Simple Ecosystem, With Wolves Eating Sheep According to the Lotka-Volterra Equations

SOURCE: Wilensky, U. (2005). NetLogo Wolf Sheep Predation (System Dynamics) model.

<http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation> (System Dynamics). Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

System dynamics is based on the evaluation of sets of simultaneous differential or difference equations, each of which calculates the value of a variable at the next time step given the values of other, causal variables at the current time step. Software such as Stella () and NetLogo (<http://www.iseesystems.com/>) and NetLogo

(<http://ccl.northwestern.edu/netlogo/>, described in more detail in [Chapter 4](#)) can help with drawing the diagrams, and can execute the simulation by computing these equations.

In comparison with agent-based modeling, the system dynamics approach deals with an aggregate, rather than with individual agents. For example, in the wolves and sheep model the simulation will compute the total population of sheep at each time step, but will not represent each individual sheep. This makes it difficult to model heterogeneity among the agents. Although one could, in principle, have a distinct stock for each different type of agent (e.g., a stock of white sheep, a stock of black sheep, a stock of mottled sheep, and so on), in practice this becomes extremely cumbersome with more than a few different types. It is also difficult to represent agent behaviors that depend on the agent's past experience, memory, or learning in a system dynamics model. On the other hand, because it deals with aggregates, the system dynamics approach is good for topics where there are large populations of behaviorally similar agents. Thus, system dynamics was an appropriate method for Forrester's models of the global economy because individual action was unimportant, and the focus was on the state of the world.

### **1.4.3 Discrete Event Simulation**

In typical agent-based models, time is divided into time steps (e.g., a day or an arbitrary unit of time) and the simulation proceeds one tick of the simulation clock at a time, at a constant rate. However, for some models this is a wasteful and unintuitive approach. For example, in a model of the queue at an airport check-in desk, for much of the time nothing happens: the agent at the head of the queue converses with the check-in agent while the rest of the queue simply wait. The alternative, the discrete event approach, is to advance the simulation clock not at a constant rate, but from one event (e.g., one passenger has been checked in) to the next event (the next passenger has been checked in). In this example, other kinds of events might be the arrival of a new person at the tail end of the queue and the clerk leaving the

check-in desk to take a coffee break. In between the events, the system being modeled is assumed to be fixed and unchanging. Each event changes the state of the system.

As well as queues and events, other elements of a discrete event simulation are gates, which allow agents to leave a queue, and servers, which process agents for a period (the desk clerk acts as a server in the check-in example). The simulator has a list of events, each with an associated time, sorted into chronological order. It picks off the next event from the list and simulates its occurrence, which might generate further timed events that are placed on the event list. This continues until the event list becomes empty and the simulation stops.

Discrete event simulation (Banks, Carson, Nelson, & Nicol, 2010; Robinson, 2004) is useful when one needs to model a process in which there are agents that remain in queues until an event occurs. As well as modeling queues of people, discrete event simulation can be used for modeling factory production processes to detect bottlenecks, hospitals to show the effect of decreasing the number of beds on patient throughput, and traffic to show the consequences of traffic light sequencing, and many other issues where what is of interest is a flow of agents or entities.

Although in classic discrete event simulation, the agents, or entities, are passive objects that are merely acted on when events occur, nowadays many discrete event simulations are close to being agent-based models, the only major difference being the use of an event scheduler rather than having a fixed time step (Lawson & Park, 2000).

## Descriptions of Images and Figures

[Back to Figure](#)

The first row on this flow chart has two arrows with taps in them represented by a circle with a lever on top. The arrows point from left to right and have a box in the middle labeled, sheep. An arrow points from the tip of the arrow on the left to the tap in the middle of the same arrow. Another arrow points from the start of the right arrow to the tap in the middle of that arrow.

The second row in the flow chart has two boxes labeled sheep-births and sheep-deaths on the left and right respectively. These are positioned under the taps on the arrows on the first row respectively.

The third row on this flow chart has two boxes with two diamonds under them on the left corner. The first box is aligned to the start of the arrow on the first row and reads, sheep-birth-rate. The second box is aligned to the sheep box on the first row and reads, prediction-rate.

An arrow from the sheep-birth-rate box points to the tap to the left arrow on the first row. This arrow also passes the box labeled sheep-births on the second row.

The fourth row on this flow chart has two arrows with taps in them represented by a circle with a lever on top. The arrows point from left to right and have a box in the middle labeled, wolves.

An arrow points from the tip of the arrow on the left to the tap in the middle of the same arrow.

Another arrow points from the start of the right arrow to the tap in the middle of that arrow. A third double-sided arrow points from the tap on the left tap on the fourth row, to the right tap on the first row. It also passes through the prediction-rate box on the third row. An arrow from the sheep box on the first row also points to the tap on the left arrow on the fourth row.

An arrow from the wolves box points to the tap on the right arrow on the first row.

The fifth row in the flow chart has two boxes labeled wolf-births and wolf-deaths on the left and right respectively. These are positioned under the taps on the arrows on the fourth row respectively.

The sixth row on this flow chart has two boxes with two diamonds under them on the left corner. The first box is aligned to the start of the arrow on the fourth row and reads, predictor-efficiency. The second box is aligned to the right arrow on the fourth row and reads, wolf-death-rate.

An arrow from the predictor-efficiency box points to the left tap on the fourth row. This arrow also passes the box labeled wolf-births on the fifth row. Another arrow points from the wolf-death-rate to the right tap on the fourth row. This arrow passes through the wolf-deaths box on the fifth row.



# CHAPTER 2 AGENTS, ENVIRONMENTS, AND TIMESCALES

We noted in [Chapter 1](#) that an agent-based model consists of a set of agents acting within an environment. In this chapter, we begin by showing how agent-based models can be designed.

## 2.1 Agents

Agents generally have all or most of the following characteristics:

1. Perception. They can perceive their environment, possibly including the presence of other agents in their vicinity. In programming terms, this means that agents have some means of determining what objects and agents are located in their neighborhood.
2. Performance. They have a set of behaviors that they are capable of performing. Often, these include the following:
  - a. Motion. They can move within a space (the environment).
  - b. Communication. They can send messages to and receive messages from other agents.
  - c. Action. They can interact with the environment, for example by picking up food.
3. Memory. They have a memory, which records their perceptions of their previous states and actions.
4. Policy. They have a set of rules, heuristics, or strategies that determines, given their present situation and their history, what behaviors they will now carry out.

Agents can be implemented in many different ways.

### 2.1.1 Agents as Objects

Almost all agent-based models are built using an object-oriented programming language, such as Java, C++, Python, or even Visual Basic. This book cannot teach you object-oriented programming, but because the idea of object-oriented programming is so important to agent-based modeling, here is a short introduction to its main features.

Object-oriented programming develops programs as collections of

objects, each with its own set of things it can do. An object can store data in its own **attributes**, can send messages to other objects, and has **methods** that determine how it is able to process data. The general programming advantage of object orientation is that it provides a high level of modularity. For example, the details of how an object's methods work can be changed without upsetting the rest of the program. An additional advantage for agent-based modeling is that there is an affinity between the idea of an agent and an object: It is natural to program each agent as an object.

There are several basic concepts in object-oriented programming:

- A **class** is an abstract specification of an object, including its attributes and its methods. For example, a program might include a class called Company to represent a firm in a model of an economy. A Company might have attributes such as its capitalization, its number of employees, and the type of product it sells. It might also have methods that describe the processes involved in selling the Company's products to customers and in buying the Company's materials from suppliers, which could be other companies in the model. Classes may be specialized to form more specific classes. For example, the general Company class could be specialized to yield Manufacturing companies and Distribution companies. Each specialized class inherits the attributes and methods of its more abstract class, and may add new attributes and methods or override the ones it inherits. For example, the Manufacturing class would need methods that determine the volume of product that can be made from unit volumes of materials, whereas the Distribution class would not need this method.
- As the program runs, classes are **instantiated** to form objects. For example, the Manufacturing class might be instantiated to yield two objects, one representing XYZ, Inc. and the other ABC, Inc. Although the two objects have the same methods and the same attributes, the values of their attributes (e.g., their names,

size, and type of product) can differ. An object's methods may send messages to another object, thus affecting the second object's state. For example, one of the methods of the XYZ object may send a message to the ABC object requesting it to sell some of its products; a method of the ABC object might respond with the number and price of the products it wishes to sell.

As this summary suggests, it is a short step from object orientation to agent-based modeling. One creates a class for each type of agent, provides attributes that retain the agents' current state, and adds suitable methods that observe the agents' environment and carry out agent actions according to some rules. In addition, one needs to program a scheduler that instantiates the required number of agents at the beginning of the simulation and gives each of them a turn to act. [Chapter 4](#) includes an example of building such a simulation.

## 2.1.2 Production Rule Systems

As we noted in the previous section, the agents in most models need to have the ability to perceive the state of their environment, receive messages from other agents, proactively select behavior to perform depending on their current state, and send messages to other agents. One way to achieve these is to endow the agents with the following:

- A set of rules of behavior. These rules determine what the agent will do. One or more rules from the set will be selected depending on the current state of the agent. Such rules are often called condition-action rules because they include both a condition component (what must be true if the rule is to be used) and an action component (what will be done to carry out the rule). For example, one such rule might be, "If I can see food in the vicinity, then I will move a step toward it."
- A working memory. This will consist of variables that store the

agent's current state. For example, the working memory might store the agent's current location and its current energy reserve.

- A rule interpreter. This is some program code that uses the working memory to select which rule should be activated, or fired. It may need to handle the situation where the condition component of more than one rule is satisfied, and therefore some means of choosing between the rules is needed.
- An input process. This will collect messages and perceptions from the environment and store them in working memory for processing by the rules.
- An output process. This will pass messages to the environment, en route to the agents that are the intended recipients.

Such an arrangement is called a **production (rule) system**, and the rules are production rules (Nilsson, 1998; Russell & Norvig, 2010). The correspondence between the elements of a production system and the desirable features of an agent listed above should be clear. A relatively simple production system can be constructed from a toolkit such as Jess (the Java Expert System Shell, <http://www.jessrules.com/>) (Friedman-Hill, 2003).

### 2.1.3 Agents That Learn

The agents in most of the models mentioned so far are not capable of learning from experience. Simple agents driven by production rule systems have a memory in which their current and past state is recorded, so that they “learn” about the state of the environment as they proceed through it. But usually we mean something much more than this by learning. One kind of learning is learning more-effective rules, and then changing behavior as a result of the learning.

**Reinforcement learning (RL)** is akin to what we do when exploring a new city to find a tourist attraction by trial and error: You try one street,

and if it looks promising you carry on, or if not, you double back to try another. Agents that engage in RL have a state (e.g., their current location) and a small set of possible actions (e.g., move one cell north, east, south, or west). The environment is set up so that it will eventually provide a reward (gratification in reaching the attraction). The agent must find a policy that maximizes the reward it receives (Maini & Sabri, 2018; Sutton & Barto, 2018). Usually, each additional step along the way to the goal reduces the eventual reward; as a result, the goal is not just to find the reward, but also to do so in the most efficient way. The agent can try many times to find the best policy, exploring different routes, and so one of the issues in designing RL algorithms is the balance between exploration (trying new routes, which may be less efficient and waste time) and exploitation (keeping to the old routes, which may not be the best way). RL has been used to model firms that learn about the preferences of their customers (Sallans et al., 2003) and decision making in organizations (Sun & Naveh, 2004), among other phenomena. Several authors have used simple models of RL to explain and predict behavior in the context of experimental game theory with human subjects (e.g., Chen & Tang, 1998; Erev & Roth, 1998; Flache & Macy, 2002; Izquierdo, Izquierdo, & Gotts, 2008).

RL is well suited to problems in which the environment and the reward structure remain constant during the simulation run. On the other hand, RL does badly if the environment is dynamic, especially if all the agents are learning and the actions of one agent affect the state or the rewards of other agents.

### **2.1.4 Cognitive Models**

While relatively simple agents, with behavior defined by condition-action rules or a production system, often serve very well, it is also possible to design agents that make decisions on the basis of ideas more aligned to psychological or social theories of human cognition (An, 2012; Balke & Gilbert, 2014; Groeneveld et al., 2017).

## The Belief-Desires-Intention Model

The Belief-Desires-Intention (BDI) model, which was originally based on ideas expressed by the philosopher Bratman (Bratman, Israel, & Pollack, 1988), is one of the most popular models of agent decision making. In contrast to production rule systems, the idea behind BDI is that agents have a mental state as the basis for their reasoning. As suggested by its name, the BDI model is centered around three mental attitudes: beliefs, desires, and intentions.

Beliefs are the internalized information that the agent has about the world. These beliefs do not need to correspond with reality (e.g., the beliefs could be based on outdated or incorrect information); it is only required that the agent considers its beliefs to be true. Desires are all the possible states of affairs that an agent might like to accomplish. Goals are the subset of desires that an agent actively desires to achieve (Dignum, Kinny, & Sonenberg, 2002). An intention is a commitment to a particular course of action (usually referred to as a plan) for achieving a particular goal (Cohen & Levesque, 1990).

These components are complemented by a library of plans. The plans define procedural knowledge about low-level actions that are expected to contribute to achieving a goal in specific situations and consist of steps that define how to do something. Agents can reason about their plans dynamically. Furthermore, they can reason about their own internal states; in other words, they can reflect on their own beliefs, desires, and intentions and, if required, modify these.

At each reasoning step, a BDI agent's beliefs are updated, based on its perceptions. Beliefs in BDI are usually represented as atomic formulae of first-order logic. The intentions to be achieved are pushed onto a stack, called the intention stack. This stack contains all the intentions that are awaiting achievement. The agent then searches through its plan library for any plans with a post-condition that matches the intention on top of the intention stack. Any of these plans that have their pre-conditions satisfied according to the agent's beliefs are considered possible options for the agent's actions and intentions.

From these options, the agent selects the plans of highest relevance to its goals. Based on these goals and plans, intentions are generated, updated, and then translated into actions that are executed by the agents.

An example may make this clearer. Suppose that we have a model in which there are agents that need to maintain their energy level by seeking out energy sources, which are distributed randomly around a landscape. A production rule system might have rules to move the agent in a random direction, perceive the presence of energy sources, and tap the energy. The agent is defined in terms of what it does. In contrast, a BDI agent would have the goal of maintaining its energy level, beliefs about the locations of energy sources, an intention to obtain energy from any source it finds, and plans about how to move and what to do when energy is found. While in practice the production rule or reactive agent and the BDI or intentional agent might perform exactly the same actions, the BDI agent is designed not in terms of what it does, but instead in terms of what it wants and how its desires can be achieved (Caillou, Gaudou, Grignard, Truong, & Taillandier, 2017).

One of the seminal applications to use an implementation of BDI was a monitoring and fault detection system for the reaction control system on the NASA space shuttle Discovery (Georgeff & Ingrand, 1990). For this purpose, 100 plans and more than 25 meta-level plans (including more than 1,000 facts about it) were designed. Since then, BDI agents have also been used for more psychologically inspired research. They formed the basis for a computational model of child-like reasoning, Children's Reasoning about Intentions, Beliefs, and Behavior, or CRIBB (Wahl & Spada, 2000), and have also been used to develop a rehabilitation strategy to teach autistic children to reason about other people (Galitsky, 2002) and for business games (Farrenkopf, Guckert, Urquhart, & Wells, 2016).

BDI agents differ from those based on production rule systems in that BDI agents are typically goal persistent. This means that if an agent for some reason is unable to achieve a goal through a particular



intention, the agent is able to reconsider the goal in the current context (which is likely to have changed since the agent chose the original course of action). Given the new context, a BDI agent is then able to attempt to find a new course of action for achieving the goal. Only once a goal has been achieved or it is deemed to be no longer relevant does an agent discard it. A detailed discussion on modeling human behavior with BDI agents can be found in Norling (2014).

## Normative Agents

BDI agents act because of a change in their set of beliefs and the establishment of desires to achieve a specific state of affairs (for which the agents then select specific intentions in the form of plans that they want to execute). Their behavior is driven purely by internal motivators such as their beliefs and desires. Normative agents (NoA) also attend to social norms—that is, the standards of behavior that are accepted by and govern social groups. In contrast to beliefs and desires, norms are external to the agent and are established within the society/environment where the agent is situated. Agents that take account of norms are said to be norm-governed or normative. Two examples of NoA are those of the EMIL project (Andrighetto et al., 2013; EMIL Project Consortium, 2008) and NoA (Kollingbaum & Norman, 2004), both of which extend the notion of norms to include legal norms (Boella, van der Torre, & Verhagen, 2007).

EMIL focuses primarily on decisions about which norms to accept and internalize and the effects of this internalization. Factual knowledge (events) and normative knowledge (rules) are treated differently and the agent has a separate interface to the environment for each of these types of knowledge. The agent also has two kind of memories: (i) an event board for facts and events, and (ii) a normative frame for inferring and storing rules from the event board. An agent engages in norm recognition, norm adoption, decision making, and then normative action planning, using an approach similar to BDI, but where the beliefs, goals, and intentions are all based on social norms (Andrighetto, Campennì, Conte, & Paolucci, 2007).

The NoA design of Kollingbaum and Norman (2003) uses a broader definition of norms including organizational concepts and ideas from formal legal systems. The norms governing the behavior of an agent refer to either actions or states of affairs that are obligatory, permitted, or forbidden. Agents have an explicit representation of their normative state. A normative state is a collection of norms (obligations, permissions, and prohibitions) that an agent holds at a point in time that is consulted when the agent wants to determine which plans to select and execute. NoA agents construct plans that are required not to violate any of their internalized norms to achieve their goals.

While much more complicated than the basic if-then rule or production rule agents, these agents are still simple compared with psychologically plausible models of human cognition, such as Soar (Laird, Newell, & Rosenbloom, 1987; Wray & Jones, 2005; Ye & Carley, 1995), CLARION (Sun, 2006), and ACT-R (Taatgen, Anderson, & Lebiere, 2006). However, psychologically plausible models are so rich and so complicated that using them within an agent-based simulation (where each agent would have to run its own cognitive model) can lead to models that are prohibitively slow to run and difficult to manage. (For examples where this was done using CLARION, see Sun and Naveh [2004]; and Naveh and Sun [2006].)

## 2.2 Environments

The previous section has described ways of designing agents. The agents act within an environment, which provides the channel of communication between agents and may also include nonreactive objects, such as obstacles or energy sources.

### 2.2.1 Features of Environments

It is convenient to route all communication between agents through the environment, not only because that is the natural way to do it, corresponding to the role of the environment in human affairs, but also because it makes monitoring the agents easier. It also means that messages from one agent to another can be temporarily stored in the environment (**buffer**), reducing the likelihood that the results of the simulation will depend on the accidents of the order in which agent code is executed. When buffering, messages from agents are stored in a temporary variable until all the agents in the simulation have had their turn. Then, these stored messages are delivered to the recipients.

In many models, the environment will include passive objects, such as landscape barriers, “roads” or links down which agents may travel, resources to provide agents with energy or food, and so on. These can be programmed in much the same way as agents, but more simply, because they do not need any capacity to react to their surroundings.

Because the environment is often straightforward to program, it tends to get neglected, yet environmental influences are generally very important in the real world. Much of the complexity of human life comes from dealing with a complex environment. We also often use the environment as a memory (e.g., placing objects in particular locations to remind us that action needs to be taken), as a store of

value (e.g., money and other forms of wealth), and as a technological aid to make some actions easier (e.g., devices that provide communication and transport services). However, it is not often that researchers recognize this environmental complexity in designing simulation models.

## 2.2.2 Geography

Most agent-based models involve agents moving over a landscape, but often this terrain is a rectangular plane or a toroid. In place of these abstract surfaces, there is also the possibility of creating complex artificial surfaces, or incorporating terrains mapped from real landscapes. This is done by integrating a **geographical information system (GIS)** into the model. A GIS is a software system for managing, storing, and displaying spatial data such as maps (Chang, 2004; Heywood, Cornelius, & Carver, 2011).

GISs store spatial data in specially built or adapted spatially aware database systems. These systems are designed to be efficient at answering the kind of queries that one needs to ask for managing geographical data, such as, “Tell me about all the other objects that are no more than 10 units away from this object.” GIS data are often arranged in layers that contain data on one or a few variables. When displaying or manipulating a map, one can turn some layers on or off to see just the variables in the visible layers. For example, a map might include roads and lakes in separate layers. If one wanted to see the road network but not the lakes, the lake layer could be turned off. There are two kinds of GIS variables: raster and vector. Rasters are regular grids of cells, each cell storing a single value. Raster variables are used for data that vary continuously over the surface, such as height and hours of sunshine. Vector variables are used for data that are in the form of points, lines, or areas (known in GISs as polygons). Spatial data may be displayed using a projection, which is a way of showing the surface of a three-dimensional body such as the earth on a two-dimensional map. There are a wide variety of projections available, although the question of which projection to use is more

likely to trouble a geographer wanting to map the whole earth than a modeler who wants to map just a town or region, small enough that the differences between alternative projections become irrelevant.

With a GIS it is possible to build an agent-based model in which the agents traverse a more realistic landscape—for example, moving only along city streets. This is vital for projects such as modeling traffic flows (e.g., Zheng, Waraich, Axhausen, & Geroliminis, 2012) and the spread of epidemics (Dibble & Feldman, 2004; Dunham, 2005), although it may be a distraction if the intention is to model an abstract process not specifically located in any particular place. Models in which the environment represents geographical space are called spatially explicit. One decision that needs to be made early is whether it is sufficient for the landscape to be unchanging throughout the simulation run, or whether the landscape needs to change dynamically. For example, a static representation of a city is likely to be sufficient for a traffic model because in the period represented by the simulation (a few hours or days) the arrangement of city streets will not change materially. On the other hand, a model of the effects of a hurricane on a city population may need to have its topography updated during the simulation to account for floods, closed streets, and storm damage. However, managing time-varying data is at the state of the art for GISs and a challenge if one wants to integrate it with agent-based modeling.

A second issue that needs to be considered when designing spatially aware agents is how they will detect the features of the terrain they are traversing. For example, if one wants to ensure that pedestrians walk down streets and not through buildings, there needs to be a way for agents to determine that the way ahead is a walkway. This is best done by sending a query to the GIS to determine the locations of the objects ahead of the agent and then decomposing (or parsing) the returned answer to check that forward movement is not obstructed. In principle this is not complicated, but in practice it can take a lot of processing by both the underlying GIS and the agents and may slow down the model.

## 2.3 Randomness

One way of abstracting from the complexity of the real world when designing a model is to build in some randomness. For example, in a model of an industrial network (Section 1.2.2) one might want to abstract from the firms and interfirm linkages to be found in a particular industrial sector in order to develop a general model of networks between firms. However, the question then is, Which linkages should be built into the model? One answer is to choose pairs of firms at random and create links between these pairs. This may be done in several ways, depending on the intended distribution of linkages. For example, the chance of a firm being linked to another could be uniform for all firms (a random network), could be related to the number of firms already linked to the other firm (a preferential attachment) (Barabási, 2003; Barabási & Albert, 1999), or could be arranged so that the links form a local cluster of strongly connected firms with a few long-distance connections between clusters (a small-world network) (Watts, 1999, 2004; Watts & Strogatz, 1998). These alternative schemes provide networks with different structural properties that may have quite different behaviors (Pujol, Flache, Delgado, & Sangüesa, 2005). Personal networks, such as networks of friends, have specific characteristics that may be fitted better with a social circle model (Hamill & Gilbert, 2009).

Randomness may also be used to model communication errors and the influence of noise. For example, Axelrod (1997b) and Axelrod and Dawkins (1990) developed an influential model of the dissemination of culture (Axelrod, 1997c) to explain why, if the beliefs and attitudes of people who interact tend to become more alike, all differences do not eventually disappear. His model displays the emergence of stable regions of homogeneous cultural traits, such as dialects, nationalistic beliefs, or religious customs. In the model, agents have tags, which are a set of five numbers that describe their own cultural traits on five cultural features or dimensions. The chances of two agents interacting depends on the similarity of their tags. If they do interact, one feature

from the five is chosen at random, and one agent adopts the other agent's value for that feature. After several thousand interactions, distinct regions emerge in which all the agents share the same traits and have no traits in common with agents in other regions. Axelrod comments that his model shows how local convergence can give rise to global polarization. Several subsequent papers by different authors have used such tags to explore the emergence of differences between groups of agents (e.g., Edmonds, 2006; Hales, 2000, 2002; Riolo, Cohen, & Axelrod, 2001). However, later work (Klemm, Eguíluz, Toral, & Miguel, 2003) showed that the tendency to global polarization is critically dependent on the agents' copying the value of the other agents' cultural traits exactly, as well as on there being no cultural drift in an agent's traits. If, instead, an agent adopts a slightly distorted copy of the other agent's traits, or if the agent's beliefs occasionally change randomly, a monoculture may emerge rather than there being several distinct regions, each with different cultures. This is an example of how random variation or noise can make a radical difference to the outcome of a model.

## 2.4 Time

In most cases, simulations proceed as though orchestrated by a clock, beat by beat. At each beat, all the agents are given a turn. Thus, time is modeled in discrete time steps. Each time step lasts for the same simulated duration. The simulation starts at time step zero and proceeds as long as necessary, or until all the agents are “dead” or out of action.

There are three issues about time that need consideration when designing agent-based models using discrete time steps:

1. Synchronicity. We have already mentioned in the previous section that one needs to be careful about the timing of messages sent from one agent to another. For example, if Agent A sends a message to Agent B and B replies, and then C sends a message to B, the outcome might be quite different from the results of A sending a message to B, then C sending a message to B, and then B sending a message to A. (Consider these sequences in the context of a model of insider trading, for example.) This is an example of a more general issue of the order of agent invocation (Huberman & Glance, 1993).

With an ordinary computer, because only one thing can happen at once, agents cannot, in fact, engage in action simultaneously. The three possible ways to work around this are as follows:

- a. Invoke each agent in sequential order (Agent 1, Agent 2, Agent 3, Agent 4, Agent 1, Agent 2, Agent 3, Agent 4, ...) (known as sequential asynchronous execution). This is rarely a good solution, though, because the performance of the model may be greatly influenced by the order that is used.
- b. Invoke each agent in a different random order at each time step (known as random asynchronous execution). The advantage of this solution is that the effect of the ordering can be investigated by running the simulation several times,



with a different ordering each time.

- c. Invoke each agent in any convenient order, but buffer all interactions with the agents' environment so that all inputs to agents are completed before all outputs (known as simulated synchronous execution). This is the best option if it can be achieved, because it is closest to actual synchronous execution, although it can be complicated to arrange, and sometimes the requirements of the model prevent it.
2. Event-driven simulation. These three ways of arranging the ordering of agent invocation assume that all the agents need to have a chance for action in each time step, although some agents may do nothing during their slot. A different approach is possible: use an event-driven design in which only those agents that need to act are invoked. The idea of event-driven simulation is that, instead of having a time step of constant duration, the simulation skips from one event to another. The simulation clock is wound forward until the time of occurrence of the next event. For instance, suppose that we are designing a simulation of organizational decision making and have a model in which the agents are decision-making committees. The focus of the model is on the decisions that each committee makes and how these decisions are passed from one committee to another. What happens between committee meetings is of no concern, and the meetings themselves can be considered to be instantaneous. In such a model having a regular time step would be inefficient, because nothing of interest would be happening at most steps. Instead, the model could be designed to jump from the time of one committee meeting to the next.
3. Calibrating time. With both the regular and the event-driven modes of simulation, there is often a problem of matching the simulation time with real time. For example, if one has a model of consumer behavior in which one wants to study the reactions of consumers to the introduction of a new product, a matter of some interest will be the time it takes for a majority of consumers to adopt the product. The model might indicate how many time steps this takes, but how does one translate this into weeks or months of real time? A solution is to observe the process in reality and

match the shape of the adoption trend against the output from the simulation, but this will give only approximate answers. Moreover, in this and other examples there remains a difficulty about what to take as the start or zero time points in the simulated and real worlds. Although a simulation starts at a well-defined moment, in the real world it is rare that any activity (e.g., the marketing of a new product) commences at one clear moment in time. These are issues that one needs to watch out for; there is no general solution that always applies.

## 2.5 Population Learning

A very different approach to learning is that of evolutionary computation, a family of techniques of which the simplest and best known is the **genetic algorithm (GA)**. (Others include Moran processes, replicator dynamics, evolutionary programming, evolution strategy, genetic programming, and **classifier systems**, which is short for learning classifier system.) Evolutionary computation (Michalewicz & Fogel, 2004) is loosely based on natural selection and involves two basic processes: selection and reproduction. Fundamental to evolutionary computation is a population whose members reproduce to form successive generations of individuals who inherit the characteristics of their parents. The probability of successful reproduction is determined by an individual's **fitness**: Fit individuals are more likely to breed and pass on their characteristics to their offspring than unfit ones.

The individuals involved in a genetic algorithm (Holland, 1975) may, but need not, be the agents in the simulation. For example, in a model of an industrial sector, the agents could be firms, with the sector as a whole learning how to be productive in the industrial landscape through successive generations of firm bankruptcies and start-ups (Brenner, 2001; Marengo, 1992). On the other hand, each agent may learn using a genetic algorithm that acts on the agent's rules, each rule being an individual as far as the genetic algorithm is concerned; this is the way that learning classifier systems work (Bull, 2004). Simulations of stock market trading have been built with such agents (Arthur, Holland, LeBaron, Palmer, & Tayler, 1997; Johnson, 2002).

For a genetic algorithm, each individual must have its features encoded in a **chromosome**. The encoding may be in the form of a binary string, with zero representing the absence of a feature, and one representing its presence, or as a more complicated structure. It is also important that it is possible to assess the fitness of every individual—for example, the fitness may be measured by the

accumulated capital stock of a firm agent, or by the effectiveness of a cooperation strategy in a model of altruism. Individuals are selected for reproduction by the algorithm in proportion to their fitness, so that very fit individuals are very likely to breed and very unfit ones very unlikely. The reproduction takes place by combining the parents' chromosomes using a process called **crossover**: Slices are taken from each chromosome and combined to form a new slice made of some sections from one parent's and some sections from the other parent's. In addition, to ensure that there continues to be some variation in the population even after a great deal of interbreeding, some bits from the offspring's chromosomes are randomly changed, or mutated.

The offspring's fitness is evaluated to determine its likelihood to reproduce to yield grandchildren. Eventually, the individuals who are relatively unfit die out and are replaced by individuals who have inherited from fit parents and are likely to be fit themselves. Although no individual does any learning as a result of the genetic algorithm, the population as a whole can be considered to be learning or optimizing as the individuals within it become fitter.

The genetic algorithm is usually a very effective optimization device. Whether it is a good model of any social phenomenon is more debatable (Chattoe, 1998; Reschke, 2001). One problem is that it is often difficult to see what an appropriate way of measuring fitness would be or even whether the concept has any clear meaning when applied to social phenomena. Sometimes this does not matter. For example, in a model of a simple society one does not need a carefully elaborated definition of fitness: Agents can be designed to breed if they have the opportunity and have not previously died from lack of resources. Another common problem with the basic genetic algorithm is the difficulty of coding all the salient features of an agent into a binary string for the chromosome. However, other evolutionary computation techniques allow a wider range of structures. Genetic programming (Banzhaf, 1998; Koza, 1992, 1994; Poli, Langdon, & McPhee, 2008), for example, substitutes program code for the binary string: Individuals evolve a program that can be used to guide their

actions, and learning classifier systems (Bull, 2004; Meyer & Hufschlag, 2006) use condition-action rules as the equivalent of chromosomes.

# CHAPTER 3 DESIGNING AN AGENT-BASED MODEL

Over the past decade agent-based modeling has developed a more or less standardized research process, consisting of a sequence of steps, at each of which design decisions need to be made. Like most social science methods, this process is an idealization of the procedures actually carried out, and, in practice, several of the steps occur in parallel and the whole process is performed iteratively as ideas are refined and developed. Nevertheless, it is useful to have these steps made explicit as a guide to the conduct of agent-based modeling research (see also Axelrod, 1997a; Hammond 2015).

In this chapter we start by listing the steps and then we illustrate each step using an example, so that by the end of the chapter we have a design for a simple model. In [Chapter 4](#) we will show how this design can be implemented to form a running simulation.

## 3.1 Design Steps

At an early stage in the research it is essential to narrow down the general research topic to some specific research question. A **research question** is something that the work should have a realistic chance of answering. If the question is too vague or too general it will not be much use, and the research will be disappointing because it will not be able to provide the hoped-for answers. It is always better to err on the side of specificity: Be too focused rather than too ambitious. It is helpful to think of defining the research question as peeling an onion, from the general area of investigation, through the particular topic, to a question that could be answered in no more than a brief statement of what you have discovered.

As we have noted above, the usual kind of research question that agent-based models are used to study are ones where regularities at the societal or macro level have been observed, and the issue is how these may be explained. Economists often call these regularities **stylized facts** (Kaldor, 1961). For example, the Schelling model described in [Chapter 1](#) starts with the observation that neighborhoods are ethnically segregated and seeks to explain this through modeling individual household decisions. The electricity market models also described in [Chapter 1](#) aim to explain and predict patterns of electricity supply and market pricing in terms of the motivations of suppliers.

After having specified the research question clearly and having identified the macrolevel regularities that are to be explained, the next step is to specify the agents that are to be involved in the model ([Table 3.1](#)). They may be all of one type, or there may be different types. For example, while the models of opinion dynamics reviewed in [Chapter 1](#) involve only one type of agent—the individuals whose opinion changes are being simulated—some of the industrial district models mentioned in [Chapter 1](#) involved several distinct types of firms. For each type of agent, one needs to lay out the agent's behavior in different circumstances, often as a set of condition-action

rules (see Section 2.1.1). It is helpful to do this in the form of two lists: one that shows all the different ways in which the environment (including other agents) can affect the agent, and one showing all the ways in which the agent can affect the environment (again, including other agents). Then one can write down the conditions when the agent has to react to environmental changes, and the conditions when the agent will need to act on the environment. These lists can then be refined to create agent rules that show how agents need to act and react to environmental stimuli. It will then be possible to assess whether simple rules will suffice, or whether a more complicated intentional or cognitive agent design will be needed.

At this stage one will have a good idea of the types of agents and their behaviors that are needed in the model. It will also be necessary to consider what form the environment should take (e.g., does it need to be spatial, with agents having a definite location, or should the agents be linked in a network?) and what features of the model need to be displayed in order to show that it is reproducing the macrolevel regularities as hoped. Once all this has been thought through, one can start to design and develop the program code that will form the simulation. It is often helpful to see whether there are any existing models that could be adapted or that could serve as inspiration. As well as looking at published papers (which sometimes include Web links to program code), it is worth looking at Web sites such as OpenABM (<https://www.comses.net/>) where modelers have deposited their code and documentation for others to use, and the carefully curated demonstration models in the NetLogo Library (<http://ccl.northwestern.edu/netlogo/models/>).

**Table 3.1** Initial Steps When Designing an Agent-Based Model

<b><i>Preliminaries</i></b>
1. Topic



General area of study.

## 2. Users

Who are likely to be the users of the model (e.g., only oneself, other researchers, stakeholders, the public, etc.)?

## 3. Research question

A specific question: one sentence that ends with a question mark (?).

## 4. Background

Is there a literature and, more specifically, a theory relevant to the model? If so, what does it suggest? Are there similar models that can be adapted?

## 5. Macrolevel features and patterns

### 5a. Verification

What is known about the characteristics of the domain at the macro level? Which of these features should the finished model be able to generate? Draw some plots, including some with a time axis, to show how macrolevel variables interact.

### 5b. Type

Will this be an abstract, middle-range, or facsimile model?

*Statics*

## 6. Types of entities

All the types of objects in the simulated world.

#### 6a. Agents

The (pro)active entities, usually corresponding one to one with the kinds of actors (people, firms, organizations, countries, etc.) to be found in the real world. Provide a list of all the types of agent but include as few as possible. In many agent-based models there is only one type of agent.

#### 6b. Resources

Those entities that are used by agents. Any changes in these are caused by agents acting on them. Examples are food and energy.

#### 6c. Other objects

Passive objects that are in the background, unchanged by agents, such as obstructions (walls).

### 7. Environment

Is environment spatial or nonspatial (e.g., a network)?

If spatial ...

size of the environment.

shape (toroidal, bounded, square, rectangular, 3D).

size and shape of units (patches).

If nonspatial or a network ...

directed or undirected links

Are the links static (created at initialization) or dynamic (created and destroyed during the run)?

## 8. Agent attributes

For each type of agent list the attributes (e.g., characteristics, features, or properties) of the agent. Divide the list into those that change or may change during a simulation run, such as wealth, and those that will not change.

Ensure that each type of agent has a different set of properties to all other agent types (if not, they are not different types).

For each attribute, describe the possible values it can take in the model; for example, sex could be male or female, wealth is a real number, and friends is a list of other agents.

If the environment is a network, list the attributes of the links.

## 9. Environment attributes

### 9a. Global attributes

Properties of the whole environment, e.g., the clock time

### 9b. Local attributes

Properties of each location, e.g., color, fertility, history

## 10. Initial values of all attributes

The values of the agent and environment attributes at the start of the simulation.

## *Dynamics*

## 11. Interaction between agents and environment

10a. Ways in which the environment acts on the individual agents

10b. Ways in which the agents act on the environment

12. Agent-agent interactions

Ways in which agents interact (e.g., not bumping into each other; sending messages from one to another).

13. Conditions for action

13a. Conditions

For each of the listed interactions (11 and 12), specify one or more conditions in which that action should take place.

13b. Dependencies

For each condition, check that it refers only to attributes of the agent or the environment. If, in order to evaluate the condition, one needs to know about other values, add the corresponding attributes to the agent or environment.

14. Create rules

For each condition, write out a condition-action rule, such as a rule that says when the condition is true, carry out the action.

15. Agent entry and exit

List the conditions in which an agent is created and those in which it “dies,” if any.

If agents are created (or enter the simulation), specify the initial values of their attributes.

## *User Interface*

### 16. Outputs

Decide on the graphs and other outputs that will allow the (macro) behavior model to be observed. These should include at least those plots shown in Step 5.

If the agents are to be shown on a grid or view, what should they look like? Do they change appearance according to their internal state?

### 17. Parameters

List the parameters of the model (that users can use to alter the way it behaves) and relate these to the research question (Step 3).

### 18. Tests

Define some combinations of parameter values that will allow the model to be tested (part of verification) to ensure that it produces the expected results for known scenarios.

After the model has been constructed, one begins the long process of checking that it is correct. Informally, this is called debugging; more formally, it is called **verification**. Verification is the task of ensuring that a model satisfies the specification of what it is intended to do. It is quite different from **validation**, which is checking that the model is a good model of the phenomenon being simulated. One can have a simulation that satisfies the verification criterion, because it runs as it is supposed to do; if the specification is a poor description of the target in the social world, however, it is not a valid model.

Following successful verification, one can embark on validation. The primary criterion of validation is whether the model shows the macrolevel regularities that the research is seeking to explain. If it does, then this begins to be evidence that the interactions and behaviors programmed into the agents explain why the regularities appear. However, one must guard against alternative explanations. There may be other agent behaviors that are equally or more plausible that lead to the same macrolevel regularities (known as **equifinality**). There is also the possibility that minor changes in the initial state of the model may lead to very different outcomes (known as multifinality). Therefore, one needs to engage in a **sensitivity analysis** to see whether, when model parameters are changed, the outcomes alter, too. It is also important to consider whether a simpler model leads to the same conclusions. If it does, the simpler model should normally be preferred to the more complicated one, using the principle that simple explanations are better than complicated ones if both are equally good at explaining.

Having thus explored the macro behavior of the model, it is then desirable to compare the output of the model with empirical data from the social world. As we will see later, such comparisons between model outputs and data are not easy to carry out and often do not lead to the clear answers that one might expect. Most models are stochastic—that is, they involve random processes, so one does not know whether any difference between the model output and observed data is due to random chance or a bad model. There are also often considerable difficulties in collecting valid and reliable data, especially the data observed over long periods of time that one needs to compare with model outputs. Section 4.5 discusses validation in more detail.

Finally, one can draw some conclusions, hopefully answering the research question that started the process. In addition, if one has confidence in the model, one can experiment with it, perhaps to identify regularities that had been previously unsuspected.

## 3.2 An Example of Developing an Agent-Based Model

In this section the process of developing a simple agent-based model will be described using a simulation of collectivities as an example. In [Chapter 4](#) we will see how this model could be programmed. Because it is a simple and rather abstract model, we will assume that the model user to be targeted is someone like yourself, the reader.

Several related social phenomena are difficult to model, or even to describe, because their boundaries are fluid, the people involved are constantly changing, and there is no single characteristic shared by all those involved. Examples include the following:

- Youth subcultures such as punks (Widdicombe & Wooffitt, 1990) or goths (Hodkinson, 2002)
- Scientific research areas or specialties such as astrobiology (Gilbert, 1997)
- Art movements such as the pre-Raphaelites or the vorticists (Mulkay & Turner, 1971)
- Neighborhoods such as Notting Hill in London or the Bronx in New York (van Ham, Manley, Bailey, Simpson, & Maclennan, 2012)
- Members of armed revolutionary or terrorist movements such as the Red Brigades in Germany (Goolsby, 2006)
- Industrial sectors such as biotechnology (Gilbert et al., 2014)

Although one can easily point to familiar examples, and although they are very common and easily identified, it is difficult to put one's

intuitions about them on a firmer footing. For a start, there is no commonly accepted word with which to name the phenomenon. The terms “subculture,” “area,” “neighborhood,” “specialty,” and “movement” are used for particular types, but none of these terms is appropriate for describing all of them. A closely related concept is the term “figuration” (Elias, 1939), although strictly speaking this should be applied only to individuals, and not to organizations or other types of actors. In this section we use “collectivity” as the generic term, for lack of a better one. Note that the units making up the collectivity may be people (as in most of the examples above) or organizations (e.g., biotechnology firms).

A second barrier to gaining a better understanding of collectivities is that, by definition, there is no definite boundary around them. This means that it is impossible to count their members and therefore to engage in the more common kinds of quantitative analysis of their development over time, their incidence, and so on.

Third, the way in which collectivities arise from the actions of their members is not easily understood. It is the purpose of the model to be developed here to suggest how some plausible assumptions about individual action (**micro foundations**) could yield the collectivities that are observable at a macro level. The research question is, therefore, is it possible to generate collectivities from the individual actions of agents, given just these assumptions.

### **3.2.1 Macrolevel Features and Patterns**

In all collectivities, the following seem to hold, to a greater or lesser extent:

- Although instances of collectivities are usually easily named and described at the aggregate level, precise definitions can prove to be rather slippery and open to negotiation or argument (e.g., there are many slightly different areas that can be described as



Notting Hill, from the official local government area to the locality within which the film of the same name was shot).

- There is no accepted consensual definition that can be used to sort those who are in from those who are out (or members from nonmembers). For example, whereas some might think that someone is a punk because of the way that he or she dresses, this assignment might be contested by others (including the person him- or herself) by pointing to the person's beliefs, behavior, or acquaintances, all of which could alternatively be relevant to decide on membership. In particular, there is no one observable feature that all those who are in and none of those who are out possess. Collectivities are not, for example, formal organizations, where being an employee with a written or verbal contract distinguishes those who are members from those who are not; political parties, where, at a minimum, a formal declaration of support is required and defines membership; or social classes, where externally specified objective criteria are used to sort people (typically based on one's occupation).
- Nevertheless, many of the members will share characteristics in common (e.g., the scientists in a research area may have similar educations, have carried out similar previous research, and be known to each other, even if there is no technique, theory, or object of research with which all of those without exception in the research area are involved).
- Membership of the collectivity entails possessing some related knowledge (e.g., the science of the specialty, or whatever is accepted as cool in a youth culture, or the local geography of Notting Hill). However, no member possesses all the knowledge: Knowledge is socially distributed.
- The features that are thought to be relevant to the collectivity change. For example, researchers do not continue to work on the same problems indefinitely; once they have solved some, they move on to new ones, but still within the same research area. Most political movements change their manifestos over time to

reflect their current thinking and the social problems that they observe, although they remain the same movements, with many of the same adherents. Youth cultures are constantly changing the items that are in fashion.

- Some of the people involved are widely considered (e.g., by the others) as being more central, more influential, of greater status, or as leaders compared with others. For example, some scientists are considered to be more eminent than others, some members of subcultures are cooler than the rest, and so on. (Compare the idea of optimal distinctiveness in social psychology [Brewer, 1991].)

### 3.2.2 Microlevel Behavior

One of the features common to collectivities mentioned in Section 3.2.2 is that the actors (i.e., the people or organizations that make up the collectivity) have some special knowledge or belief. For example, scientists have knowledge about their research area, and youth subcultures have knowledge about what is currently fashionable. Even though this knowledge is socially distributed among the members of the collectivity so that not every member has the same knowledge, possession of it is often a major feature of the collectivity (Bourdieu, 1986). In the model, we assume that all individuals, both members and nonmembers, have some knowledge, but what this knowledge is varies both among actors and over time. We use this knowledge to locate the actors: The position of the actor at a moment in time in an abstract knowledge space represents the knowledge that he or she possesses at that time.

A second assumption is that some actors are of higher status than others and that all actors are motivated to try to gain status by imitating high-status actors (by copying their knowledge). For example, in a collectivity driven by fashion, all actors will want to be as fashionable as they can, which means adopting the clothing styles, musical tastes, or whatever of those whom they perceive to be of the

highest status (Simmel, 1907). However, status is also a function of rarity: An actor cannot remain of high status if there are many other actors with very similar knowledge. For example, a fashion icon must always be ahead of the hoi polloi; a scientist will be heavily cited only if his or her research is distinctive; a revolutionary will earn the respect of colleagues only if he or she stands out in comparison with the foot soldiers.

Third, we assume that actors with the highest status want to preserve this status, which they cannot do if they start to be crowded out by followers who have been attracted to them. In this situation, we assume that high-status actors are motivated to make innovations—that is, to search out nearby locations in knowledge space where there are not yet crowds.

There are thus two countervailing tendencies for actors—on the one hand, they want to get close to the action; on the other hand, they want to be exclusive and can do so by changing the locations that represent the heights of status. As we will see, working out this tension yields patterns at the macro level that are typical of collectivities.

### **3.2.3 Designing a Model**

#### **Related Models**

There are several generic models that deal with similar issues:

1. Boid models (Reynolds, 1987) have agents that try to maintain a desired distance away from all other agents and thus appear to move with coordinated motion. Agents have three steering behaviors: separation, to avoid nearby agents; alignment, to move in the same direction as the average of nearby agents; and cohesion, to move toward the average position of nearby agents. The effect is that agents move as if they were a flock of sheep or a school of fish. These models illustrate the effect of having

agents carrying out actions that are in tension: The separation behavior is in tension with the cohesion behavior, for instance. However, there are no notions of seeking status or innovation in these models.

2. Innovation models (Watts & Gilbert, 2014a) have agents that can learn and act according to their current knowledge. Agents also exchange knowledge and create new knowledge. However, there is no specific idea of collectivity in these models. The set of agents involved in innovation is predetermined.
3. The minority game (Challet, Marsili, & Zhang, 2013) is one example from a large literature. This model, also called the El Farol Bar model, has agents who wish to go to the bar, but only when a minority of the other agents also choose to go there. The agents decide based on their own past experience of the number they previously encountered at the bar. Each agent has several strategies that he or she uses in combination with his or her memory of the outcome of recent trips to the bar to decide whether to visit the bar at the current time step. The strategies are scored according to their success (such as whether the bar is overcrowded when the agent arrives), and unsuccessful strategies are dropped. Over time a dynamic equilibrium can be established, with the number of agents at the bar matching the threshold that agents use to judge that there are too many agents there. This model has some features of the problem addressed here, but there is no representation of a collectivity.

## The Model

The collectivities model consists of a surface over which agents are able to move. The surface is a **toroid** with each point representing one particular body of knowledge or set of beliefs. The agents thus move, not in a representation of physical space, but rather in knowledge space. Although it may be oversimplifying to represent a knowledge space in two dimensions (more exactly, on the surface of a toroid), it makes for easier visualization.

An agent's movement in the knowledge space represents its change

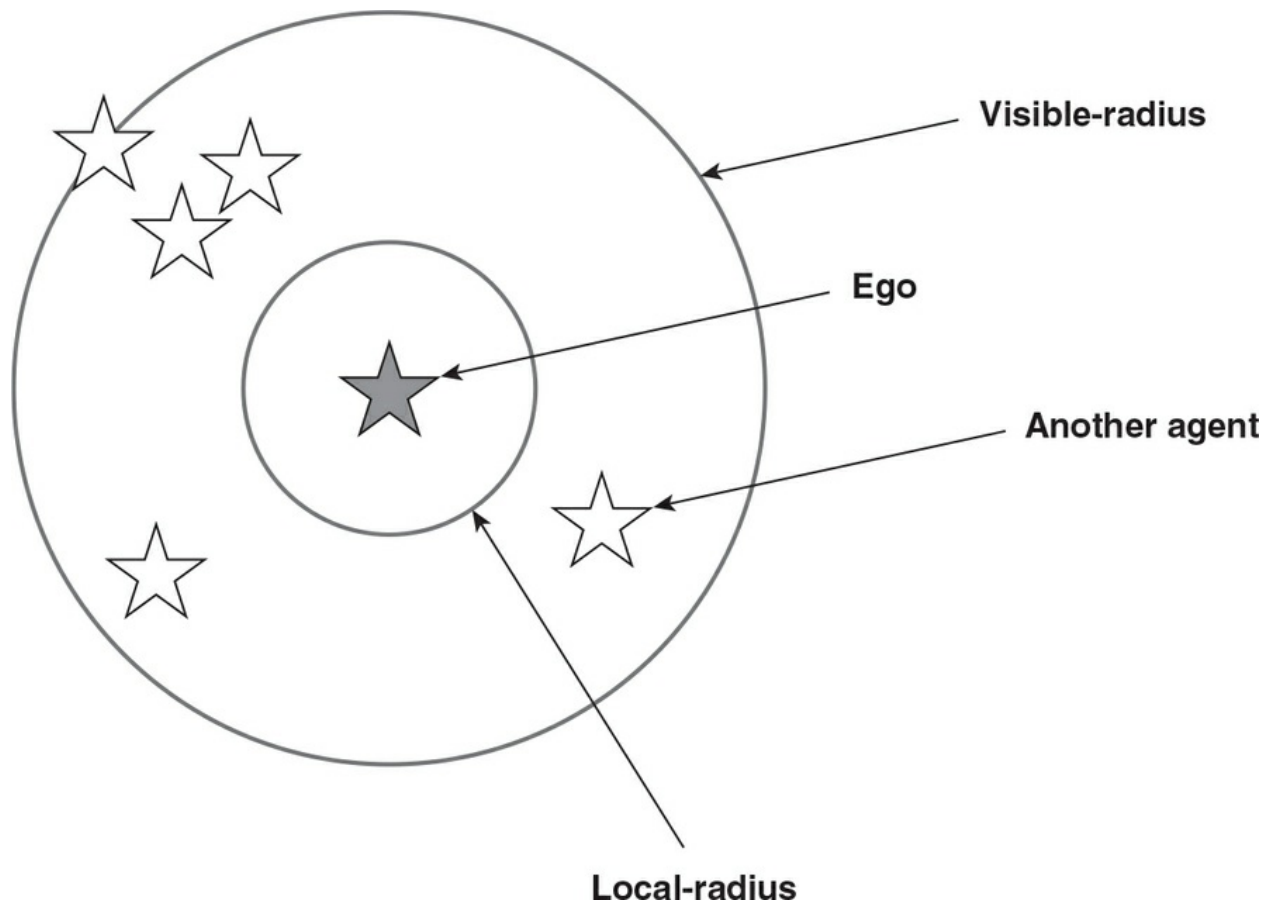
in knowledge. Thus, if an agent imitates another agent, it would move toward that agent in the knowledge space, whereas if it innovates and discovers knowledge that other agents do not have, it would move away from other agents into previously empty areas.

Agents are initially distributed at random on the surface. Agents have no memory of their own or other agents' previous positions. Each agent does the following:

1. It counts how many other agents there are in its immediate neighborhood.
2. If the number of agents is above a threshold, the agent turns to the direction opposite to the average direction of travel of other nearby agents and then moves a random distance.
3. If the number of agents is equal to or below the threshold, the agent looks around the locality to find a relatively full area and then moves a random distance from its present location in the direction of the center of that area.

Each agent acts asynchronously, repeating this sequence of actions indefinitely. There are four parameters required by this algorithm (see [Figure 3.1](#)):

1. The radius of the circular area surrounding an agent within which the number of agents is counted to determine whether the agent is crowded or lonely (local-radius).
2. The threshold number of agents below which the agent is lonely and above which the agent is crowded (threshold).
3. The radius of the circular area surrounding an agent in which the agent, if lonely, counts the number of agents to find where there is a maximum or, if crowded, finds the average direction of agent movement in order to determine the direction in which it is to move (visible-radius).
4. The distance that an agent moves; the distance is chosen randomly from a uniform distribution with this parameter as the maximum (speed).



Description

**Figure 3.1** Local-Radius and Visible-Radius in the Collectivities Model

## Descriptions of Images and Figures

[Back to Figure](#)

This illustration has two concentric circles. The inner circle is labeled local-radius and the other circle is labeled visible-radius. The middle of the inner circle has a dark star labeled, ego. There are four stars in the outer circle – two on the bottom left and right and two grouped together on the top left. The star in the bottom left is labeled, another agent.

# CHAPTER 4 DEVELOPING AN AGENT-BASED MODEL



## 4.1 Modeling Toolkits, Libraries, Languages, Frameworks, and Environments

In this chapter we will consider how, having designed a model, one can implement it in programming code and how one can then check that the design and the code are correct.

Although some modelers build their agent-based models using only a conventional programming language (most frequently Java, although any language could be used), this is a difficult way to start. Over the years, it has become clear that many models involve the same or similar building blocks with only small variations. Rather than continually reinventing the wheel, commonly used elements have been assembled into libraries or **frameworks** that can be linked into an agent-based program. The first of these to be widely used was Swarm (<http://www.swarm.org/>); although this is now more or less completely superseded, its design has influenced more modern libraries, such as Repast (<https://repast.github.io/>) (North, Collier, & Vos, 2006) and Mason (<https://cs.gmu.edu/~eclab/projects/mason/>) (Luke, Cioffi-Revilla, Panait, Sullivan, & Balan, 2005). There is also AnyLogic (<https://www.anylogic.com/features/>), which provides tools not just for agent-based modeling, but also for system dynamics and discrete event simulation. Repast and Mason are open-source, meaning that they are available for free download and noncommercial use, while AnyLogic is a commercial product. These libraries are written in the Java programming language and so link most easily to models that are also written in Java, but they can also be used with other languages. They all provide a similar range of features, such as the following:

- A variety of helpful example models.
- A sophisticated scheduler for event-driven simulations.

- Several tools for visualizing on screen the models and the spaces in which the agents move.
- Tools for collecting results to a file for later statistical analysis.
- Ways to specify the parameters of the model and to change them while the model is running.
- Support for network models (managing the links between agents).
- Links from the model to GISs so that the environment can be modeled on real landscapes.
- A range of debugged algorithms for evolutionary computation (Section 2.5), the generation of random numbers, and sensitivity analysis.
- Can be run on Windows, macOS, or Linux.

In addition, each of them has its own special features (see [Table 4.1](#) in the appendix to this chapter).

Many person-years of effort have gone into the construction of these libraries and using them can greatly reduce the time taken to develop a model and the chance of making errors. However, they are large software packages best suited to those with experience in programming and, even so, it can take some practice before one can take full advantage of the wide range of features they offer.

More suited to the beginner are **[modeling environments](#)** that provide complete systems in which models can be created and executed, and the results visualized, without leaving the system. Such environments tend to be much easier to learn, and the time taken until one has a working model can be much shorter than it would be if one were using the library approach. However, the simplicity comes at the price of less flexibility and slower speed of execution. It is worth investing time to learn how to use a library-based framework if you need the greater

power and flexibility they provide, but often simulation environments are all that is needed. NetLogo (Wilensky, 1999), CORMAS (Bommel, Becu, Le Page, & Bousquet, 2016), and GAMA (Grignard et al., 2013) are examples of simulation environments.

Environments primarily intended for other purposes can also be used for simulation, sometimes quite effectively. For example, simple simulations can be created using the spreadsheet package Microsoft Excel, and the free, open source statistics package R (<https://www.r-project.org/>) can be useful for models that involve processing large amounts of data. Several significant agent-based models have been constructed using the mathematical packages MatLab (<https://uk.mathworks.com/products/matlab.html>) (e.g., Thorngate, 2000) and Mathematica (<http://www.wolfram.com/mathematica/>) (e.g., Gaylord & D'Andria, 1998). Nevertheless, an environment designed specifically for agent-based modeling is usually the first choice.

Currently, the most popular agent-based simulation environment is NetLogo (Wilensky, 1999). It includes a user interface builder and other tools such as a system dynamics modeler. It is available free of charge for use for educational and research purposes (<http://ccl.northwestern.edu/netlogo/>). It will run on all common operating systems: Windows, macOS, and Linux. NetLogo has an active user community that answers users' questions quickly and thoroughly and has users at all levels of education and in the natural as well as the social sciences.

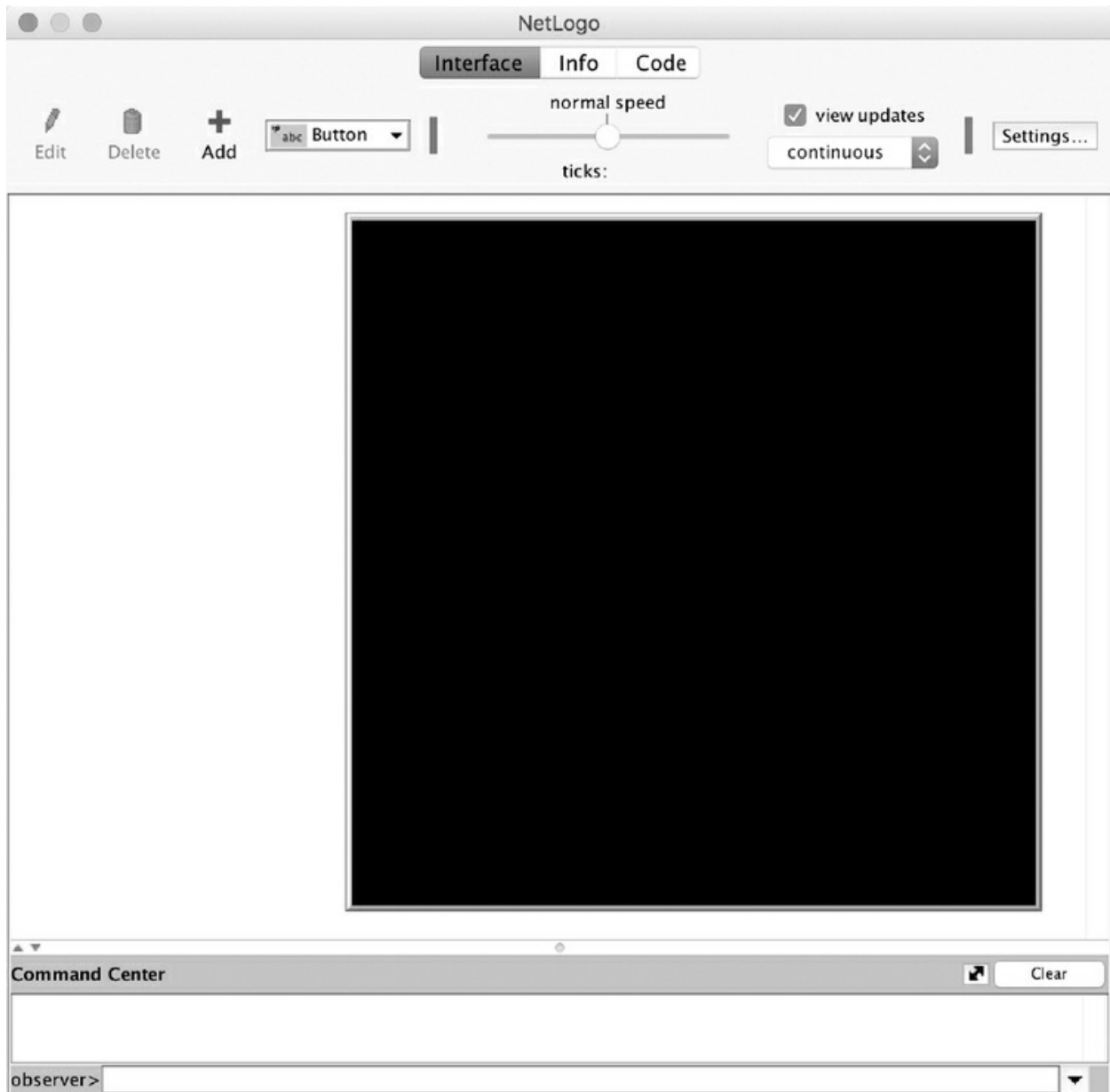
## 4.2 Using NetLogo to Build Models

In the remainder of this book we will use the agent-based simulation environment NetLogo (Wilensky, 1999). NetLogo, like the other environments and libraries mentioned above, is undergoing continuous development, with a major new version appearing more frequently than annually. Wilensky and his team strive to make their improvements to NetLogo upwards compatible, meaning that any corresponding changes needed in your or others' programs are made automatically when you load the code, or may require only minor editing.

The NetLogo system presents the user with three tabs: the Interface tab, the Information tab, and the Code tab. The Interface tab is used to visualize the output of the simulation and to control it (see [Figure 4.1](#)), the Information tab provides text-based documentation of what the simulation is for and what should be observed, and the Code tab is where one writes the simulation program using a special language specific to this environment (the NetLogo language). NetLogo is based on the programming language Logo (Papert, 1983), which was designed for teaching young children about the concepts of procedures and algorithms and was originally used to control small toy robots called turtles. In memory of its origins, NetLogo's agents are still called turtles.

The Interface tab includes a black square called the view, which is made up of a grid of [patches](#). This is the spatial environment in which the agents move: A simulation program can instruct agents to move in any direction from patch to patch, and the agents will be visible on the view (see, e.g., [Figure 4.3](#); the small triangles on the view are agents). Usually, the NetLogo environment is configured so that the left-hand edge joins on to the right-hand one and the top edge to the bottom, so that if an agent moves off the left-hand side of the view, it immediately reappears at the right-hand side (the environment is topologically equivalent to the surface of a toroid, a donut-shaped solid). Patches

start colored black but can easily be recolored so that, for example, one could create a contour map. The number of patches in the view can also be configured: When NetLogo starts, the view consists of 33 by 33 patches, but the number can be increased to many thousands.

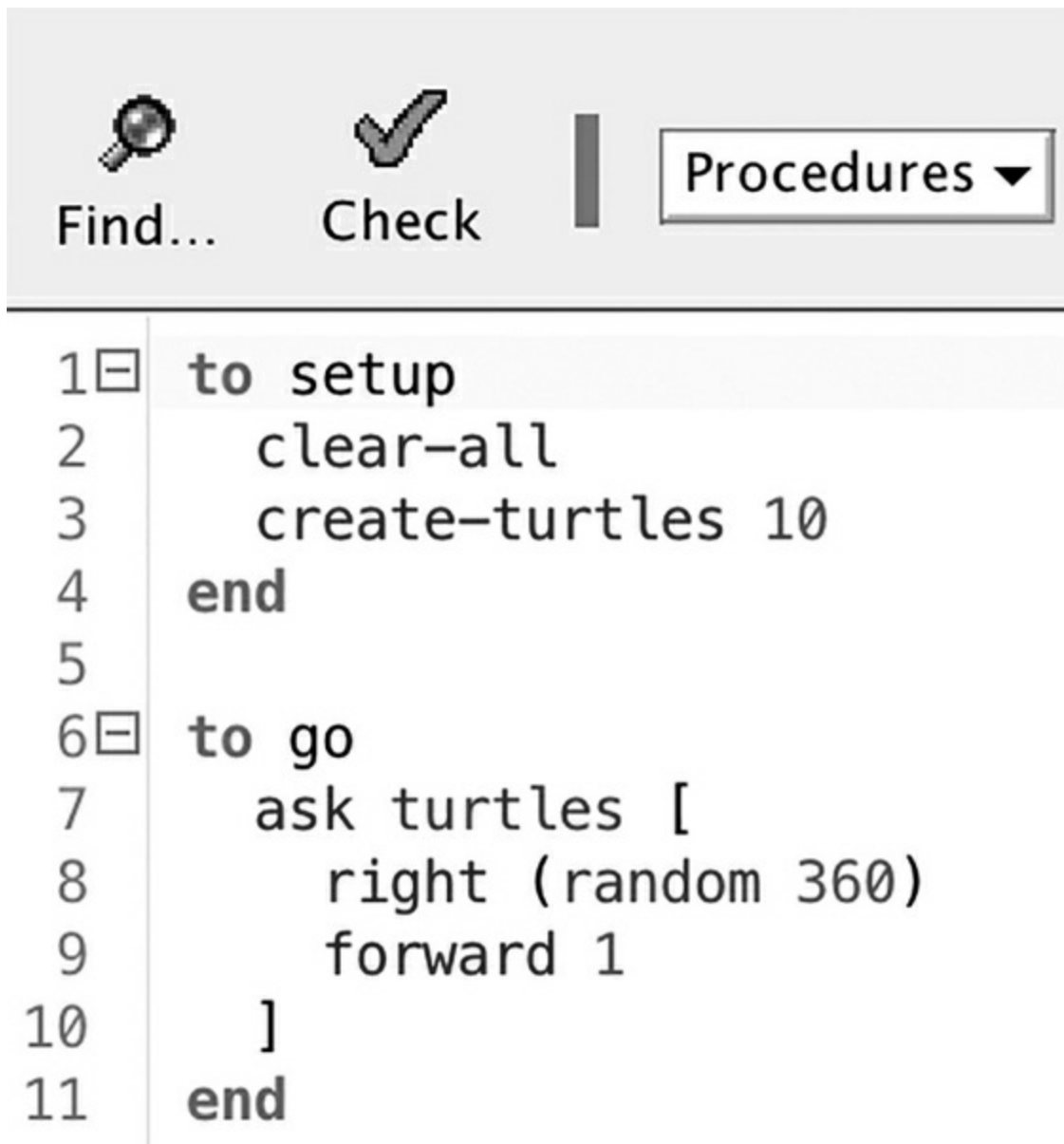


## [Description](#)

**Figure 4.1** The NetLogo Interface

A NetLogo program has three parts. First, there is a section that says what kinds of agents there will be and names the variables that will be available to all agents (the **global variables**). Second, there is a setup procedure that initializes the simulation. Third, there is a go procedure, which is repeatedly executed by the system in order to run the simulation. [Figure 4.2](#) shows a very simple example to give a flavor of a NetLogo program in which 10 agents are created and move randomly around indefinitely.

In this program there are no global variables, so the program starts with the setup procedure. Any turtles left from a preceding run are cleared away, and 10 new turtles are created (these are placed in the center of the NetLogo view). The go procedure tells each turtle (agent) to carry out the commands within square brackets: First, turn to the right (i.e., clockwise) by a random number of degrees, and then move one unit forward, where the unit is the length of the side of a patch. Each turtle moves independently of the others, all at the same time. (Because NetLogo runs on an ordinary computer, the agents cannot all operate at precisely the same time, but NetLogo makes it look as though they do by using an asynchronous random update; see Section 2.3.)



```
1  to setup
2   clear-all
3   create-turtles 10
4 end
5
6  to go
7   ask turtles [
8     right (random 360)
9     forward 1
10  ]
11 end
```

[Description](#)

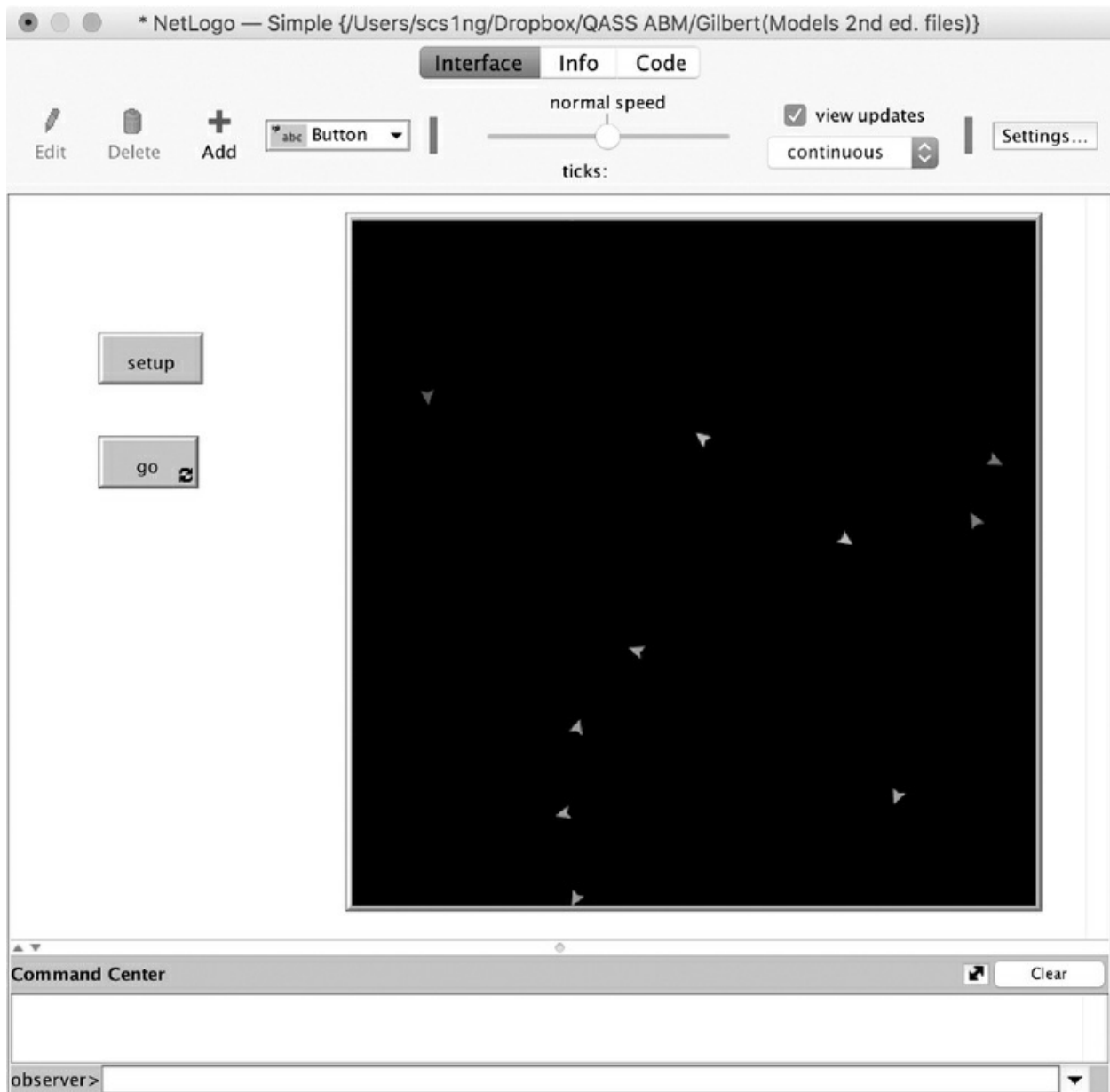
**Figure 4.2** A NetLogo Program to Create 10 Agents and Make Them Move at Random

To run this program on your own computer you would need to download and start up NetLogo. Then click on the Code tab and type in the lines of code shown in [Figure 4.2](#). Move back to the Interface

tab. Click on the Add icon at the top and then on the white area next to the view. NetLogo will open a dialog box. Type setup into the Commands field and click OK. This will draw a button labeled “setup.” Then do the same for a go button, also setting the check box Forever to on. (This will cause the go procedure to be executed in a loop when the go button is clicked, continuing until the button is clicked a second time to stop the run.) Clicking on your setup button will create 10 turtles, shown piled on top of each other in the center of the view. Clicking on the go button will send the agent turtles darting around the view, following a random trajectory. Click on the go button again to stop the program. The NetLogo interface should then look similar to [Figure 4.3](#).

Although this is a very simple example, it does give an idea of how quickly one can develop agent-based simulations in an environment like NetLogo. The graphics for buttons (and sliders, switches, etc.) to control a simulation are available through drag and drop. The view offers many possibilities for the visualization of agents and their environment without doing any programming. NetLogo will also show dynamically changing plots of output variables on the Interface tab. Although the NetLogo programming language is somewhat different from the usual procedural languages, it is both powerful and (mostly) elegant, with the result that complex simulations can be programmed in surprisingly few lines of code.





## [Description](#)

### **Figure 4.3** The Simple Program Running

There is not enough space in this book to provide a detailed tutorial on NetLogo, but the system includes a good built-in tutorial (located under the Help menu) and comes with a large number of demonstration and example models, some of which are relevant to

social science.

## 4.3 Building the Collectivities Model Step by Step

[Chapter 3](#) introduced the collectivities model. In brief, this is a model that simulates the dynamic creation and maintenance of knowledge-based formations such as communities of scientists, fashion movements, and subcultures. The model's environment is a spatial one, representing not geographical space but rather a knowledge space in which each point is a different collection of knowledge elements. Agents moving through this space represent people's differing and changing knowledge and beliefs. The agents have only very simple behaviors: If they are lonely (i.e., far from a local concentration of agents) they move toward the crowd. If they are crowded, they move away.

Thus, formally, there are two agent behavior rules:

<b><i>Condition</i></b>	<b><i>Action</i></b>
The agent is lonely.	Move toward the crowd.
The agent is crowded.	Move away from the crowd.

The first step in building the model is to make some basic decisions about the agents and the environment. The model specification implies that there will be only one type of agent and that the agents will move about in a space. We need to decide the dimensionality of this space: For simplicity, we will use a two-dimensional grid that can be mapped directly onto NetLogo's view. To avoid special effects that might occur at the edges of the grid, we will use a toroid, which has no edges. This is the default arrangement for NetLogo, so nothing extra

is required.

Next, it is helpful to lay out the logic of the model, either graphically or in pseudo-code. To show the logic graphically, it is convenient to use the Unified Modeling Language (UML), a means of representing programs that has been developed as a way of communicating software independently of the details of programming languages (Bersini, 2012; Fowler, 2003). UML provides a range of standardized diagrams that can be used to show the class hierarchy of the objects in the program; a sequence diagram that shows how one thing leads to the next; and an activity diagram, which is similar to a flowchart. UML is very good for describing a model in, for example, a published paper, but the collectivities model is so simple that UML is hardly necessary: There is only one class (for the agents) and only two agent actions (move forward and turn around).

With such a simple model an alternative approach is more helpful: Use pseudo-code, an informal mixture of natural language and programming conventions that makes the structure and flow of a program clear without requiring the reader to be familiar with any particular programming language. [Figure 4.4](#) shows the collectivities program in pseudo-code.

The program is in two parts: the initialization (called Setup in NetLogo) and the execution (Go in NetLogo). The indentation of the pseudo-code helps to clarify which lines go with which. For example, the program loops repeatedly carrying out the lines between Loop forever and End loop. Constant parameters of the model are shown in italics in [Figure 4.4](#).

### *Initialisation*

Create agents and distribute them randomly in knowledge space

### *Execution*

Loop forever

Each agent

counts the number of other agents within its *local-radius*.

Each agent

compares the number of other agents within its *local-radius* with the *threshold*

If the number is greater than the *threshold*

then (the agent is crowded).

The agent locates that agent within *visible-radius* with the most agents surrounding it.

The agent moves a distance proportional to *speed* towards this agent.

Else (the agent is lonely).

The agent locates the agent within *visible-radius* with the most agents surrounding it.

The agent moves a distance proportional to *speed* away from this agent.

End loop

**Figure 4.4** The Collectivities Program in Pseudo-Code

Once one has a pseudo-code version of the program, it is relatively easy to translate it into a programming language such as NetLogo; the

results of doing so are shown on the Web at <https://study.sagepub.com/researchmethods/qass/gilbert-agent-based-models-2e>, where there is also a step-by-step explanation of every line of the program. The model is also available at the OpenABM Web site (Gilbert, 2019).

Running the model shows that the initial uniform random distribution of agents separates into clumps, in which some agents are central and others are distributed around them. The central agents are crowded, and so move. In doing so, they shift the centroid of the clump slightly and may make other agents either crowded or lonely, and they too will move. Thus, the clump of agents, although remaining together for long durations (as measured in time steps), drifts across the view. Lonely agents move toward the clump, sometimes joining it and sometimes continuing to trail behind it. The clumps never merge.

[Figure 4.5](#) illustrates a typical snapshot. In this figure, agents that are crowded are a dark gray and those that are lonely are a lighter gray.



### [Description](#)

## **Figure 4.5** Snapshot of the Simulation

Comparing the behavior of this model with the features of collectivities described in Section 3.2, we see the following:

1. When we run the model, we see clumps, but drawing a boundary

around clumps involves some arbitrary definition, perhaps in terms of local densities of agents.

2. Although a definition of which agents are in and which are out of a clump is possible (e.g., in terms of the distance to the nearest neighbor), any such definition seems arbitrary.
3. Agents in the same clump are close together and so could be thought of as sharing some aspects of their knowledge.
4. The location of the clump, as indicated by the position of its centroid, is constantly changing as some agents move more closely into the clump and others seek new, less-crowded locations.
5. Some agents consider themselves to be crowded, and these behave differently from the other agents in the clump (by innovating or trying to find less-crowded positions by moving through the knowledge space). These agents are located more centrally in the clumps and are influential in setting the direction of travel of the other agents.

The features of collectivities that we observe in society thus emerge in the model as a result of the behavior of the agents. Although other microlevel actions could produce the same or similar macrolevel patterns (Gilbert, 2002), it is useful to know that these do yield the macrolevel behavior that we observe. Specifically, we can conclude from the model that if

1. agents change their ideas in knowledge-space in response to over-crowding (Mulkay & Turner, 1971),
2. some ideas and some agents are considered to be high status or important, and
3. agents are motivated to copy and adopt those ideas or a variation on them, then the phenomenon we have described as a collectivity will emerge from the agents' behavior.

A more detailed comparison of the model output with empirical data is not appropriate for this abstract model (see Section 4.5.5). The value of abstract models is twofold: They can account for the generation of particular phenomena, following Epstein's maxim that "to explain



macroscopic social patterns, we generate—or ‘grow’—them in agent models” (Epstein, 2007, p. 50); and they can help to highlight commonalities and differences between phenomena that otherwise might be considered incomparable. A better criterion for this model’s success is therefore the degree to which it generates further theoretical questions or informs middle-range theory that can be empirically validated. For example, the model suggests the question, “What are the significant similarities and differences between punks and scientists, given that their social formations can be re-created using the same generic model?” It does seem that there are many areas of social life where similar microlevel behaviors may be found and correspondingly many emergent collectivities, and so this model may be applied to account for a wide range of social phenomena.

## 4.4 Verification: Getting Rid of the Bugs

You should assume that, no matter how carefully you have designed and built your simulation, it will contain bugs (code that does something different from what you wanted and expected). When you first run a new simulation model, it is likely that it will have many bugs, most of them easily observable because the simulation gives anomalous results or crashes. More worrying, it is quite likely that even when you have worked on the code to remove the obvious bugs, some will still lurk to catch you unaware. As a rule of thumb, think of the number of bugs remaining as following a negative exponential—the number decreases rapidly at first, but then levels off and never hits zero. Even published simulations sometimes suffer from bugs and misinterpretations (e.g., see Edmonds & Hales, 2003; Galan & Izquierdo, 2005; Rouchier, 2003). Checking that code does not have bugs is called verification.

There are techniques for reducing the chances of including bugs and for making finding them easier. Here are some:

- Code elegantly. When you are writing the simulation program, do it carefully and steadily. Do not rush to get the code working and do not take shortcuts. Any time saved will be lost in extra time needed for debugging. Using an object-oriented language and using variable names that are meaningful in the context of your model will help.
- Include lots of output and diagnostics. It will be difficult to find bugs in sections of code that give no output to show what is happening as the program runs. You should not be satisfied with only displaying the results of the simulation; at least during the debugging phase, you will need to display intermediate values also. Some care will be needed to decide what to display so that you get useful clues, but are not so overwhelmed that you cannot

find these clues because of the amount of other output in which they are buried.

- Code walkthrough, step by step. Run the code one line or one function at a time, observing how the values of variables, parameters, and attributes change and then checking that they alter in the expected way. Although this process can be slow and tedious, it does help to ensure that the code is doing what was intended, at least for the runs that are observed. Often, programming environments provide features to make stepping through code easier to manage.
- Add assertions. If you know that variables must take some values and not others, check for valid values as the simulation runs, and display a warning if the value is out of range; such checks are known as assertions. For example, if two agents cannot occupy the same spatial location at the same time, there should be a check whether this requirement is being violated at every time step.
- Add a debugging switch. You may worry that all the code needed to identify bugs, such as assertions and diagnostics, will slow down the simulation unacceptably. Include a global variable in your program that can be set to a debugging level: from none to maximum. Precede each debugging statement with a test of this variable to see whether the statement should be run at the current debugging level. In some languages it is possible to achieve the same effect with conditional compilation.
- Add comments and keep them up to date. All programming languages allow you to insert comments—text for programmers to read that is not executed as program code. Use this feature to add comments to every function, procedure, method, and object. The comments should describe what the following block of code does and how it does it, but at the conceptual level, not at the implementation level (i.e., do not paraphrase the program code, but state what the code is intended to achieve). As a rule of thumb, there should be about one third as many lines of

comments as there are lines of code. Comments can easily become obsolete, describing the program as it used to be, rather than as it is. Reserve some time to update the comments at regular intervals. In writing comments, assume that the reader is someone who can program as well as you can, but who knows little or nothing about your model. After a couple of months away from the program, this may be a good description of you, so do not avoid writing comments with the excuse that no one else will see your code.

- Use unit testing. Unit testing is an increasingly popular software engineering technique for reducing bugs (Fröhlich & Link, 2003). It consists of writing some test code to exercise the program at the same time as you write the code itself. The idea is to develop the program in small, relatively self-contained pieces, or units. A test harness is created that will supply the unit with a sequence of inputs and that will check the results against a list of expected outputs. The test harness then automatically runs through each input, checking that the expected output is, in fact, generated. Once the unit has passed all its tests, you can move on to writing the next unit. This may involve making some changes to the first unit, which must then be put through its test sequence again to ensure that the changes have not introduced any new bugs. When many units have been written, the test harness is used to automate performing all the tests on all the units, thus giving some guarantee that bugs have not inadvertently crept in as a result of developing the code. As the program develops, additional tests should be written to verify assemblies of several units and the interaction between them.
- Test with parameter values for known scenarios. If there are any scenarios for which the parameters and the output are known with some degree of certainty, test that the model reproduces the expected behavior. This is the test that most people carry out first on a model, but it is a rather weak test and, by itself, will not give much confidence that the simulation is free of bugs.

- Use corner testing. Test the model with parameter values that are at the extremes of what is possible and ensure that the outputs are reasonable. (The name “corner testing” comes from the idea that such parameter values mark the corners of a parameter space enclosing all possible parameter values.) For example, test to see what happens when your simulation is run without any agents and when it is run with the maximum number that your model allows.

There is further excellent advice about how to do scientific computing in Wilson, Bryan, Cranston, Kitzes, Nederbragt, and Teal (2016).

## 4.5 Validation

Once one has developed an agent-based model, it seems obvious that it needs to be checked for validity—that is, whether it is, in fact, a good model of what it purports to represent. However, both the theory and the practice of validation are more complicated, and more controversial, than one might at first expect. The issues are related to the various objectives aimed at by modelers, which imply different criteria for validation, and the sheer difficulty of acquiring suitable social science data in sufficient quantity to allow systematic validation (Troitzsch, 2004). We begin by considering the conceptual issues before discussing some techniques for carrying out validation.

Agent-based models can be directed primarily at formalizing a theory (e.g., Schelling’s residential segregation model; see [Chapter 1](#)), in which case the model is likely to be pitched at a very abstract level; or they can be aimed at describing a wide class of social phenomena, such as the development of industrial districts or the behavior of consumers; or they can be intended to provide a very specific model of a particular social situation, such as some of the models of electricity markets mentioned in [Chapter 1](#), where the precise characteristics of one market, including the location of power plants and pattern of consumer demand, are relevant. Each type of agent-based models requires a different approach to validation (Boero & Squazzoni, 2005).

### 4.5.1 Abstract Models

The aim of abstract models is to demonstrate some basic social process that may lie behind many areas of social life. A good example is Epstein and Axtell’s pioneering book on *Growing Artificial Societies* (Epstein & Axtell, 1996), which presents a series of successively more complex models of the economics of an artificial society. Another example is the model of collectivities introduced earlier in this chapter.

With these models there is no intention to model any particular empirical case, and for some models it may be difficult to find any close connection with observable data at all. For example, Schelling's model is usually built on a toroidal regular grid with agents dichotomized into two classes (e.g., red and green). These characteristics of the model are plainly not intended as empirical descriptions of any real city or real households. How then might such models be validated?

The answer is to see such models as part of the process of development of theory, and to apply to them the criteria normally applied to evaluating theory. That is, abstract models need to yield patterns at the macro level that are expected and interpretable; to be based on plausible microlevel agent behavioral rules; and, most important, to be capable of generating further, more-specific or middle-range theories (Merton, 1968). It is these middle-range theories and the models based on them that may be capable of validation against empirical data. If an abstract model has been created using a deductive strategy, there will already be some hypotheses about the agent's behavior and about the macrolevel patterns that are to be expected. The first validation test is therefore to assess whether the model does indeed generate the expected macrolevel patterns. A more thorough test would be to see what happens when parameters of the model are systematically varied (see Section 4.6.6, on sensitivity analysis). One would hope that either the macrolevel patterns persist unchanged with variation in parameters, or, if they do change, that the changes can be interpreted. For example, in Schelling's model one can alter the tolerance level of the agents. At low values of tolerance, households rarely find a spot where they are happy, and the simulation takes a long time to reach a steady state, if it ever does. At sufficiently high values of tolerance, households are satisfied whatever the color of their neighbors and the initial random distribution hardly changes.

Once these basic tests have been passed, one can evaluate whether the model can be used to inform theories about specific social phenomena, and then test those theories. For example, if one uses

the Schelling model to explain ethnic segregation, one needs to start developing the theory to include the other factors that are of undoubted importance in location decisions in urban areas, including affordability and availability of the housing stock, the presence of more than two ethnic groups and people who belong to none or more than one, and the functional form of ethnic attitudes. Bruch and Mare (2006) suggest that the segregation effect in the Schelling model depends on the agents having a dichotomous attitude of being either happy or unhappy, and that clustering does not result if agents have a smoothly continuous attitude ranging from very unhappy to very happy.

## 4.5.2 Middle-Range Models

Models such as those mentioned in [Chapter 1](#) that simulate consumer behavior, industrial districts, or innovation networks are intended as middle-range simulations. They aim to describe the characteristics of a particular social phenomenon, but in a sufficiently general way that their conclusions can be applied widely to, for example, most industrial districts rather than just one.

The generic nature of such models means that it is not usually possible to compare their behavior exactly with any particular observable instance. Instead, one expects to be satisfied with qualitative resemblances. This means that the dynamics of the model should be similar to the observed dynamics and that the results of the simulation should reveal the same or similar statistical signatures as observed in the real world; in other words, the distributions of outcomes should be similar in shape (Moss, 2002).

For example, the firms that one finds in innovation networks have collaborative links with other firms in the same industrial sector. If one counts the number of partners of each firm and plots the log of the number of partners against the log of the number of firms with that many partners, the graph is approximately a straight line with constant slope (e.g., Powell, White, Koput, & Owen-Smith, 2005, Fig. 3). A



linear relationship between logged variables is the statistical signature of a **power law**, and it is characteristic of many social networks, from utility power networks to the World Wide Web (Barabási, 2003). We would expect that a simulation of an innovation network would also show a power law distribution of interfirm links with a similar slope.

An example of a middle-range model is Malerba, Nelson, Orsenigo, and Winter's (2001) work on the computer industry. They describe their model as history friendly (Windrum, Fagiolo, & Moneta, 2007), by which they mean that while the model does not reproduce the exact history of the computer industry, it does

---

capture in a stylized and simplified way the focal points of an appreciative theory about the determinants of the evolution of the computer industry. It is able to replicate the main events of the industry history with a parameter setting that is coherent with basic theoretical assumptions. (Malerba et al., 2001, para. 6.1)

---

They also note, "Changes in the standard set of parameters actually lead to different results, 'alternative histories' that are consistent with the fundamental causal factors of the observed stylized facts" (Malerba et al., 2001, para. 6.1).

### 4.5.3 Facsimile Models

Facsimile models are intended to provide a reproduction of some specific target phenomenon as exactly as possible, often with the intention of using it to make a prediction of the target's future state, or to predict what will happen if some policy or regulation is changed. For example, a business may be interested in finding the consequences for their inventory level of reducing the interval between sending out restocking orders. It is likely to require a model that precisely represents all their suppliers, the goods each supplies, and the unit

quantities of those goods in order to be able to make reasonable predictions. Another, very different, example is the work by Axtell and colleagues (2002) and Swedlund, Sattenspiel, Warren, and Gumerman (2015) on the Anasazi Indians in the American Southwest. These people began maize cultivation in the Long House Valley in about 1800 BC but abandoned the area 3,000 years later. Axtell and colleagues' model aimed to **retrodict** the patterns of settlement in the valley and match this against the archaeological record, household by household.

If such exact matches can be obtained, they would be very useful, not only as a powerful confirmation of the theory on which the model is based, but also for making plausible predictions. However, there are reasons for believing that simulations that exactly match observations of specific phenomena are likely to be rare and confined to rather special circumstances. Most social simulations contain some element of randomness. For example, the agents may have initial characteristics that are assigned from a random distribution. If the agents interact, their interaction partners may be selected randomly, and so on. The effect of this is that running the model several times will yield different results each time. Even if the results are only slightly different, the best one can hope for is that the most frequent outcome—the mode of the distribution of outputs from the model—corresponds to what is observed (Axelrod, 1997a; Moss, 2002). If it does not so correspond, one might wonder whether this is because the particular combination of random events that occurred in the real world is an outlier and whether, if it were possible to rerun the real world several times, the most common outcome would more closely resemble the outcome seen in the model.

#### **4.5.4 Complexity**

Related to randomness is the idea of complexity. In the technical sense of the word, complexity is found in systems where there are many diverse interacting components with nonproportional interactions between them. Some physical systems are complex, such

as the earth's climate; many ecological systems and social systems, such as organizations, are almost always complex. Many systems are not only complex, but also adaptive, meaning that the components change or learn in response to changes in their environments. Agent-based models are well suited to modeling complex systems, because the components can be represented as agents. Two important implications of a system being complex are that the behavior of the system as a whole is often emergent—that is, not merely the aggregation of the behavior of the components—and that the trajectory of the system over time may be impossible to predict precisely. As an example of the former, the behavior of an organization such as a firm is not simply the sum of the individual activities of its employees but depends on the structure of the organization and how the employees interact. A commonly cited example of the impossibility of precise predictions of the future path of a complex system is that accurate weather forecasts cannot be made beyond a horizon of about 10 days, no matter how powerful the computer used to make them. Similarly, the precise value of a stock exchange index is impossible to predict far ahead.

## 4.6 Techniques for Validation

Two areas need to be examined when validating models: first, the fit between a theory and the model of that theory, and second, the fit between the model and the real-world phenomenon that the model is supposed to simulate.

### 4.6.1 Comparing Theory and the Model: Sensitivity Analysis

The fit between a theory and its model is best evaluated by using the theory to derive several propositions about the form of the relationships expected between variables, and then checking whether the expected distributions do, in fact, appear when the model is run using a variety of parameter settings (Grimm & Railsback, 2012). Each of the parameter settings corresponds to an assumption made by the model. One should aim to check each of the settings either by measuring the value from empirical data or by conducting a sensitivity analysis. Although measuring the value is preferable, there will be many parameters that cannot be checked empirically, and for these some form of sensitivity analysis is essential. For example, models of science and technological innovation often involve representing the flow of knowledge between agents, but rate of knowledge transfer is in practice impossible to observe in the world, although it may be possible to measure within the model.

Sensitivity analysis is aimed at understanding the conditions under which the model yields the expected results. For example, with the opinion dynamics model described in [Chapter 1](#), one might ask, “How extreme do the extremists have to be for all the agents eventually to join one of the extreme parties?” To find out, one needs to run the simulation for a series of values of the uncertainty parameter, perhaps ranging from 0.5 to 1.0 in steps of 0.1 (i.e., six runs). But the model includes random elements (e.g., the order in which agents meet and

exchange opinions is random), so one should not be content with just six runs, but should perform several runs for each value of the uncertainty parameter to obtain a mean and variance. If one does 10 repetitions for each parameter setting, we would need to carry out 60 runs. (The number of repetitions needs to be chosen with the amount of variation in mind, so that one gets a statistically meaningful result.)

To make matters worse, most models include many parameters, and their interaction may affect the simulation (e.g., the number of extremist groups that emerge depends on both the uncertainty of the extremists and the distribution of extremists across the political spectrum, the parameters having an effect both independently and in combination), so that ideally one would want to examine the output for all values of all parameters in all combinations. Even with only a few parameters, this can require an astronomical number of runs and thus is not a practical strategy.

If we think of the range of each parameter as lying on an axis, the set of all parameters defines a multidimensional parameter space, in which each point corresponds to one combination of parameter values. The scale of the task in doing a full sensitivity analysis can then be quantified as the volume of this space, and any way of cutting down the space will reduce the number of simulation runs needed. One obvious way is to use prior empirical knowledge to restrict the range of as many parameters as possible. For instance, we might know that a parameter, although theoretically capable of taking any value between 0 and 100, in fact is never observed to have a value greater than 10. Alternatively, we can limit the applicability of the model by constraining the range of values we test: We may state that the model applies only if the parameter is somewhere between 5 and 10 and not investigate what happens when it is outside this range.

Another approach, which can be used in combination with limiting the range of parameters, is to sample the parameter space. Instead of performing simulation runs at every point in the space, we use only some points. These points may be chosen randomly or purposively to inspect combinations that we think are particularly interesting or that

are close to regions where major changes in the simulation's behavior are expected (**phase changes**).

The choice of which parameters to analyze and which variables to observe can be informed by suggestions from the literature on the Design of Experiments (Kleijnen, 2015; Lorscheid, Heine, & Meyer, 2012). This literature was originally concerned with the design of agricultural experiments to identify the best plant breeds but has now grown to encompass all kinds of experiments, including simulation experiments. Using Design of Experiments one can reduce the number of simulation runs and maximize their effectiveness by a careful and systematic choice of what parameters to vary and by how much, and how many simulations to carry out.

A sophisticated version of this approach uses a learning algorithm, such as the genetic algorithm (see Section 2.5) to search the space to identify regions where some output variable or variables take their maximum or minimum values (Chattoe, Saam, & Möhring, 2000). It is also possible to reverse the process: Instead of varying an input parameter or parameters and observing what changes in the output, one can identify the output behavior one is interested in and search for the combination of input parameters that most strongly generate that output, an approach called Query-Based Model Exploration (Stonedahl & Wilensky, 2011).

Because agent-based models are stochastic—that is, the calculations involve random numbers—the outcome from one run may not be the same as the outcome from the next. Normally, one carries out many runs and then takes the mean outcome, averaging over all the runs. But how many runs should one use? The conceptually simplest answer to this question is to plot the cumulative mean outcome and its standard error (the standard deviation of the mean divided by the square root of the number of runs) against the number of runs so far. Typically, the mean will converge to a stable value, and the standard error of the mean will decrease as more and more runs are added in. Then, one stops the runs when the standard error has decreased to the point where the 95% confidence interval around the mean (which

is the mean  $\pm 1.96$  \* standard error) is sufficiently small for the purpose at hand. Ritter, Schoelles, Quigley, and Klein (2011) explain this well using a psychological simulation based on ACT-R (see Section 2.1.1). Lee and colleagues (2015) discuss other, more-sophisticated methods for determining minimum numbers of runs.

## 4.6.2 Comparing the Model and Empirical Data

As discussed in the previous section, not all models are expected to match empirical data; there may be no data available to compare with models whose objective is the development of theory, and no reason to conduct empirical tests. For middle-range models the criterion is whether the simulation generates outputs that are qualitatively similar to those observed in the social world, but a quantitative match is not expected. It is only with what we have called facsimile models that there are stringent requirements for comparison between data obtained from the simulation and empirical data. This section considers some ways of carrying out such comparisons.

Social scientists are well used to comparing data obtained from models and data collected from the social world. This is what is being done implicitly every time one calculates an  $R^2$  (the coefficient of determination) in an ordinary linear regression (Field & Iles, 2016; Fielding & Gilbert, 2005). The model in this case is the regression equation, which computes the predicted values of the dependent variable. Similarly, one can measure the fit between the values of a variable that are output from a simulation model and the values observed empirically (this is, in fact, just the correlation coefficient between the two sets of values). However, this simple procedure makes several strong assumptions that, although often satisfied for linear regressions, are much less likely to be appropriate for simulation models. These include that the error is approximately normally distributed, but this is often not the case with agent-based models. There are standard techniques for checking normality (e.g., residual plots) and for managing nonlinearity (e.g., take logs and use nonparametric statistics) that are described in statistics textbooks

(e.g., Huet, Bouvier, Poursat, & Jolivet, 2004).

One important characteristic of simulation models is that the values of output variables change as the simulation runs. For example, in a model of consumer behavior the number of purchasers of one brand might be observed growing from zero to a majority during a simulation. The growth trend might then be compared with the growth in sales of an actual product. This can be done visually, comparing both time series, but this can be difficult if there are time lags or different time scales in the two series. A useful technique to deal with this is dynamic time warping (Lee et al., 2015). One can also use statistical techniques to compare time series, although one must allow for the fact that there is autocorrelation: the value at time  $t + 1$  is not independent of the value at time  $t$ . Statistical procedures called AutoRegressive Integrated Moving Average (ARIMA) can be used to compare such time series (Chatfield, 2004).

One also needs to consider what outputs should be validated against data. A helpful way of considering this is known as pattern-oriented modeling (Grimm & Railsback, 2012). In the context of validation, the idea is that one should look for patterns in the simulation and check that each of these patterns is matched in both data and simulation output. A pattern is a relationship between two or more variables, or a spatial or temporal arrangement. For example, if one wanted to validate a segregation model (see Section 1.2.2), one might first look at the patterns of clustering of households by ethnicity. But, in addition, one could compare the distribution of household incomes simulated in the model with the distribution found in a city; the microlevel distribution of attitudes to ethnic minorities; the distribution of location preferences of household; the number of house moves per unit of time; and so on. The more of these patterns that match, the stronger the evidence in favor of the model. While it is possible that a model based on a quite erroneous theory of segregation might match reality with respect to one of these patterns—for example, the clustering of ethnic neighborhoods (due to equifinality; see Section 3.1)—it is unlikely that it will correctly match all of them. Although pattern-oriented modeling makes strong demands on the availability of



data, it can be a powerful approach to validation. Windrum and colleagues (2007) review other approaches to empirical validation, from the perspective of economics.

**Table 4.1** Comparison of some agent-based modelling libraries and environments

	<i>Repast Symphony</i>	<i>Mason</i>
<i>License</i>	GPL	GPL
<i>Documentation</i>	Limited	Improving, but limited
<i>User Base</i>	Large	Moderate
<i>Modeling Language(s)</i>	Java, Python	Java
<i>Speed of Execution</i>	Fast	Fastest
<i>Support for Graphical User Interface Development</i>	Good	Good
<i>Support for Systematic Experimentation</i>	Yes	Yes

<i>Ease of Learning and Programming</i>	Moderate	Moderate
<i>Ease of Installation</i>	Moderate	Moderate
<i>Further Information</i>	<a href="https://repast.github.io/">https://repast.github.io/</a>	<a href="https://cs.gmu.edu/~eclab/">https://cs.gmu.edu/~eclab/</a>

GPL: General Public License, <http://www.gnu.org/copyleft/gpl.html>

GIS: Geographical Information System

Having now discussed how one designs and develops a simulation model, [Chapter 5](#) considers what one can then do with the results of simulation, including tips on project planning, publication, and making an impact on policy.

## Appendix: The Features of Simulation Libraries and Environments

[Table 4.1](#) compares several popular agent-based modeling environments on several criteria, using admittedly subjective judgments. None is ideal for all uses. To choose among them one needs to consider one's own expertise and experience in programming, the likely complexity of the model, and the aims of the project (e.g., Is the project very exploratory and the model likely to be fairly simple? Or does the project intend to build a relatively complicated model and exhaustively test its behavior against data?). They all continue to be developed and are evolving quite rapidly, so the information in [Table 4.1](#) needs to be checked against the current state of each of the systems. Other comparisons and reviews can be found in Abar, Theodoropoulos, Lemarinier, and O'Hare (2017); Allan

(2010); Castle and Crooks (2006); Gilbert and Bankes (2002); Railsback, Lytinen, and Jackson (2006); Tobias and Hofmann (2004); and at <https://www.comses.net/resources/modeling-frameworks/> and [https://en.wikipedia.org/wiki/Comparison\\_of\\_agent-based\\_modeling\\_software](https://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software).

## Descriptions of Images and Figures

[Back to Figure](#)

There are three tabs on the top part of the window that read interface, info and code from left to right. The interface tab is open.

The commands seen below the tab name read:

Edit (with a pencil can icon).

Delete (with a small garbage can icon).

Add (with a plus sign icon).

A button that reads abc, on the left side.

A scale a button at the center. The scale is labeled normal speed.

A checked box that reads, view updates.

A button labeled continuous below the view updates title.

A button labeled settings at the end.

A blank dark box is seen in the next part of the window and it titled, ticks:.

The bottom most section of the box reads, command center and has a clear button on the top right corner.

A field labeled observer> is seen below this section.

[Back to Figure](#)

This image has a banner above with options that read: Find... (with a magnifying glass icon); check (with a check mark icon) and a button labeled procedures with a down arrow head on the right end. The

eleven rows seen below this banner is replicated below:

1. to setup.
2. clear-all.
3. create-turtles 10.
4. end
- 5.
6. to go.
7. ask turtles [.
8. right (random 360).
9. forward 1.
10. ].
11. end.

The numbers 1 and 6 have an option to collapse the rows below them on the right. This option is seen as a minus sign in a square next to both these row numbers.

[Back to Figure](#)

There are three tabs on the top part of the window that read interface, info and code from left to right. The interface tab is open.

The commands seen below the tab name read:

Edit (with a pencil can icon).

Delete (with a small garbage can icon).

Add (with a plus sign icon).

A button that reads abc, on the left side.

A scale a button at the center. The scale is labeled normal speed.

A checked box hat reads, view updates.

A button labeled continuous below the view updates title.

A button labeled settings at the end.

A dark box is seen in the next part of the window and it titled, ticks:. Ten arrow heads in two different colors are seen in this box and are pointing in different directions. Most of them are in the middle and lower parts of the box.

The bottom most section of the box reads, command center and has a clear button on the top right corner.

A field labeled observer> is seen below this section.

### [Back to Figure](#)

The image on the top left shows the tip of a cluster of arrow heads in two colors that predominantly point downward. Most of the arrows are in color 1 (representing agents that are lonely) and a few are in color 2 (representing agents that are crowded).

The image on the top right has a cluster of arrows with most of the top half of the cluster pointing inward and the five rows of arrow heads from the center are seen as five lines pointing outward, below the cluster. One arrow head in the center and a few on the top half are in color 2 while the rest are in color 1.

The image on the lower left has a cluster of arrow heads which raise from bottom to top and spreads outward on top. Most of the arrow heads point up and are in color 1. Quite a few of the arrow heads pointing upward are in color two, throughout the cluster.

The image on the bottom right has a small cluster of arrow heads, arranged in a semicircle, mostly pointing up. All except five are in color 1. These five arrow heads are in color 2.

# CHAPTER 5 USING AGENT-BASED MODELS

## 5.1 Planning an Agent-Based Modeling Project

As with any research project, it is helpful to plan a simulation project step by step in advance. Then you can be more confident that what you plan is likely to be achievable, and you can take remedial action if it becomes clear that you are falling behind schedule. Although most simulation projects are not different in their essentials from projects using other styles of research, there are some special features that need attention.

- Do not underestimate how long it takes. It is tempting to think only about the time spent writing the code, but it often takes as long to design a model as to code it, and frequently longer to debug a program than to write it. Therefore, it is not being pessimistic to estimate the time needed to write a program and then multiply this by at least three to give the total time for model development. Unless one is using a modeling environment such as NetLogo, most of the programming will be taken up with the development of the user interface and output display routines, not with coding the model itself. This is one reason modeling environments are so valuable—they can save a great deal of work.
- Keep a diary. Ideas will occur to you at all stages of the project, and you risk forgetting them unless you jot them down in a diary or lab book. Pay special attention to problems that you encounter while building the model: Difficulties that you initially assume are just technical programming problems may turn out to have a wider significance. For example, if it seems that the results of the simulation are very sensitive to the particular value of a parameter, this may just be an issue in building the model, but it may also suggest some substantive conclusions about the role of this parameter in the real world.



There are some additional points that need consideration in larger projects, where there are several researchers working in a team.

- Find people with appropriate skills. If you are a lone researcher, you will know the extent to which you are already skilled in programming models. If the project is a larger one in which there is some division of labor, you will probably need to recruit people with expertise in modeling. Because agent-based simulation is still a new approach, it is difficult to find researchers with significant experience, and you may need to be content with hiring people with other skills and training them in agent-based modeling. Particularly useful skills are a familiarity with programming in Java. Even if you are not intending to use one of the Java-based libraries (see Section 4.1) the grounding in programming that a Java course gives is very useful. Also useful is some experience in researching in the domain to be modeled; and the ability to write clearly, which is vital for preparing reports and papers.
- Attend to intra-project communication. If more than one person is working on a project, attention needs to be paid to making sure that everyone understands each other and is aware of what the others are doing. Although this is true in all team tasks, in modeling projects it is usual to have some people who are domain experts but know little about modeling, and others who are modeling experts but know relatively little about the domain. Both sides may feel inhibited about asking questions and exposing their ignorance to the others. In larger projects, it may be worth scheduling specific training sessions, where those with greater knowledge of particular aspects of the project teach the others in order to bring everyone up to a common level of knowledge and skill.
- Pay attention to scheduling. Most modeling projects involve some data collection and some model development. This can be tricky to schedule if the specification of the model awaits the collection

and analysis of empirical data, but the collection of data rests on the prior definition of precisely what is to be measured. Unless care is taken, one can get into a situation in which neither modelers nor data collectors can make a move.

- Plan for and write good documentation. Documentation that describes the model, the assumptions the model makes, how to run it, and the interpretation and limitations of the outputs is essential for any project, but especially for large ones. This documentation should serve both as a means of communication between all those involved in the model (commissioning, specifying, building, and using it) and also as a historical record so that at a later date the model can be reused and maintained, and the results justified. Writing comprehensive documentation at the right level of detail takes practice. See Etter (2016) for a short guide.

When writing documentation, it is often helpful to use a standardized template. A popular one is Overview, Design concepts, and Details, or ODD (Grimm et al., 2006; Grimm, Polhill, & Touza, 2017). ODD is “intended to facilitate readability through stipulating a structure for the description with a logical ordering” (Polhill, Parker, Brown, & Grimm, 2008, n.p.). In an ODD description all modeling entities and processes, as well as the purpose of the study, are introduced in the Overview section. In the Design concepts section, the model is discussed according to a list of concepts (e.g., emergence, adaptation) that are derived from the complex adaptive systems literature (Railsback, 2001). Finally, the implementation of the model, including the experiment settings and inputs, are specified in the Details section. Hence, ODD covers not only the description of the model but also its purpose and the initialization and inputs of simulation experiments. ODD + D, an adaptation and extension to the original ODD protocol, adds features to aid describing the agents’ decision making, as well as encouraging documentation of the underlying theoretical foundations of the model, which are both areas where ODD, which originated in ecology, had been found to be lacking

when used for socioeconomic models (Müller et al., 2013).

## 5.2 Reporting Agent-Based Model Research

The best way of learning how to report the results of agent-based modeling is to study how others have done it. Take a sample of papers that you have found to be helpful or interesting and look closely at how the authors constructed them and what makes them persuasive. Although agent-based modeling is too young an area to have a very well-developed set of conventions about how papers should be written, there are some common elements (Axelrod, 1997a). A helpful discussion can be found in Richiardi, Leombruni, Saam, and Sonnessa (2006).

The main sections of an agent-based modeling report or journal article are usually as follows:

1. An abstract. This should indicate (in roughly this order)
  - a. the main research question considered in the paper;
  - b. the findings and conclusions of the paper; and
  - c. the methods used (e.g., agent-based modeling; survey analysis) and, for empirical data, the sample from which data were collected.
2. An introduction that sets out the background to the issue addressed in the paper and explains why it is of interest.
3. A literature review that discusses previous work and shows why the research reported in this paper is a worthwhile addition or improvement to the prior work. Literature on both the research problem or domain and on related models, even if these have not previously been applied to the domain, should be reviewed. This section should make clear in which respects the reported research is an advance and how it is using previous work.
4. A statement of the regularities that you want to explain (this will usually be a summary of the material in the introduction and review). These may be stated as a set of formal hypotheses that you aim to (dis)prove, or they may be presented less formally.
5. A description of the model. The description needs to be

sufficiently detailed that a reader could, in principle, reimplement your model and obtain the same results, but it should not include the program code, although this should be available for download (some readers will not know or understand the programming language you have used). Instead, use diagrams (e.g., UML) or pseudo-code to describe your model (see Section 4.3). Pay particular attention to the sequence in which events occur in the model: This is the most frequent source of problems in reimplementing a model accurately. Do not be afraid of including equations relating variables if these will help to specify your model precisely.

6. A description of the parameters. The values you have chosen for each of the parameters need to be explained and justified. Some may be based on observations of the social world (e.g., the employment rate in a model of the labor market); some may be plausible guesses after you have investigated the effect of varying their values using a sensitivity analysis; and some may have been inferred backwards, because it is only these values that give the patterns of output that you want to demonstrate with the simulation. All this needs to be explained.
7. A description of the results. This will almost certainly involve presenting and commenting on graphs that show how variables that you have observed from the simulation runs are related. Be careful to be clear about the conditions under which these simulations were carried out. For example, do the plots show averages of several runs, and if so, how many runs and how much variation was there between runs? (You might consider using error bars to show the degree of variation.) If you are showing the trend in a variable over time from step zero onward, make sure that you have plotted a run long enough that it is clear that the trend has become established and is unlikely to change drastically just off the graph (Galan & Izquierdo, 2005). If you are relating the values of variables as they are at a particular time step, make sure that you state the time step at which the measurements were made.
8. A discussion of what steps you took to verify (see Section 4.4) and validate (see Section 4.5) the model, and what confidence

the reader should therefore place in your results.

9. A conclusion. This should take the hypotheses listed in Section 4 and clearly state whether the model suggests that they are true, false, or not proven. This section can then develop the ideas from the introduction, proposing a general conclusion and perhaps speculating about the implications. For example, if the paper is about the labor market, what policies might or might not be successful in reducing rates of unemployment?
10. Acknowledgments. Brief thanks to sponsors, funders, and those who have helped you to do the research.
11. A list of references including only those works cited in the paper and no others. As always, you need to be sure to provide full bibliographic details in the format required by the journal in which you hope to publish.
12. Optionally, an appendix in which large tables and possibly the pseudo-code version of your model is placed.
13. A link to a location on the Web from where your model code can be downloaded (Janssen, 2017).

Almost as important as describing your model for publication is making the program code available for others to read, inspect, and use. Unless the code is accessible, others are unlikely to be able to repeat your simulations, try other parameter settings, or replicate your results. Many journals either insist or recommend that model code be made publicly accessible. Although one can merely link the code to a personal Web page, this is not a good idea because the code needs to be available permanently, and personal Web pages and even institutional Web sites get replaced or deleted over time. The best place to lodge code is at a specialized archiving site, such as OpenABM (<https://www.comses.net/>), which caters specifically for agent-based models.

When archiving code, one needs to provide not just the program code itself, but also a copy of the documentation and the meta-data—that is, information about the version of the model; the date it was archived; the operating system and modeling environment that it uses; any libraries or files that are required; instructions about how to install,

start, and run the model; and so on. OpenABM has a user interface that makes it easy to provide all these data in the required format. Uploaded models are reviewed to ensure that they conform to the site's standards for model code and documentation.

## 5.3 Agent-Based Models for Public Policy

One of the main purposes for developing agent-based models is to assist in the formulation of public policy. If you look back at the examples of agent-based models in Section 1.2, many of those mentioned relate to government or commercial policy, such as reducing residential segregation, enhancing commercial innovation, designing the rules for electricity markets, and managing agricultural irrigation. There are several key considerations when one is developing an agent-based model for use in policymaking, or indeed for any use other than pure research (Gilbert, Ahrweiler, Barbrook-Johnson, Narasimhan, & Wilkinson, 2018):

- Models need to be at an appropriate level of abstraction. Although this is a consideration for all modeling, it is especially true for policy models, where there may be pressure from stakeholders to model every detail, entailing the need for much more data than are readily available, and making the modeling take longer making it more difficult than is really necessary. On the other hand, a model that is too abstract (see Section 4.5.5) risks not yielding results that are applicable to the specifics of the target domain.
- The availability of data for calibration and validation. Often time data series (longitudinal data) are not available at all or are inadequate. This should not be an excuse to give up modeling however; the effort of designing and building a model is itself valuable in identifying the important issues in the policy domain, may facilitate more-effective communication between stakeholders, and may reveal that other sources of data than those first considered could be used. In the absence of data, sensitivity analyses may be a partial substitute (see Section 4.6.6).



- Quality assurance and maintenance. Stakeholders are likely to want assurance that the model abides by quality standards, possibly by using an external reviewer to check the model, its documentation, and its validation (HM Treasury, 2015). Also, if the model will continue to have a role in the policymaking process (e.g., a model developed to assist in the appraisal of policy options that may be of value when conducting an evaluation of the policy), there need to be arrangements for maintaining the model so that it continues to work, possibly over a period of years, as personnel, programming languages, hardware, and organizational structures evolve.
- Ethics. Because policy models are part of a process of policymaking that can potentially directly affect people's lives, it is important consider the ethical implications of the modeling. Often, the data used for calibration are personal data and must be processed in accordance with data protection laws and within the terms of the data subject's informed consent. One must also consider whether the data are representative rather than biased (e.g., obtained only from segments of the population such as white males). Turning to the outputs of the model, these must be communicated ethically, with due attention paid to the degree of certainty or uncertainty in the results, and to presenting the findings clearly so that the assumptions and logic of the model can be understood by those using it.

# RESOURCES

## Societies and Associations

There are three regional societies that promote social simulation and agent-based modeling, each with an annual conference. Every other year, they organize a World Congress together.

- Computational Social Science Society of the Americas (CSSSA). Web site: <https://computationsocialscience.org/>
- Pacific Asian Association for Agent-Based Approach in Social Systems Sciences (PAAA). Web site: <http://www.paaa.asia/>
- European Social Simulation Association (ESSA). Web site: <http://essa.eu.org/>

You can join these associations for a modest annual membership fee, and they provide a very useful entry to the agent-based modeling research community.

## Journals

Research using agent-based modeling appears both in discipline-specific journals and in interdisciplinary journals focusing on social simulation. The two most prominent interdisciplinary journals are the following:

- *Journal of Artificial Societies and Social Simulation (JASSS)*. This is an online electronic journal, available only on the Web. It is free with no subscription. On the front page, at <http://jasss.soc.surrey.ac.uk/>, you can sign up to receive an e-mail when each issue is published (four issues per year).
- *Computational and Mathematical Organization Theory (CMOT)*. This is a hard-copy journal, with an online, charged-for version at <https://www.springer.com/business+%26+management/journal/10!>

Other journals with more than occasional agent-based modeling papers include the following:

- *Artificial Life*
- *Complexity*
- *Computational Economics*
- *Ecology and Society*
- *Emergence: Complexity and Organization*
- *Environment and Planning B*
- *Environmental Modeling and Software*

- *Journal of Economic Dynamics and Control*
- *Physica A: Statistical and Theoretical Physics*
- *Simulation and Gaming*
- *Simulation Modeling Practice and Theory*
- *Social Networks*
- *Social Science Computer Review*

## Mailing List and Web Sites

The SIMSOC e-mail distribution list sends out notices of forthcoming conferences, workshops, and jobs of interest to agent-based modelers. You can subscribe to the list or view the list archives at <http://www.jiscmail.ac.uk/lists/simsoc.html>.

There is an excellent Web site maintained by Leigh Tesfatsion called Agent-Based Computational Economics, which has many well-categorized links to a wide range of literature, at <http://www.econ.iastate.edu/tesfatsi/ace.htm>.

This book's Web site is at <https://study.sagepub.com/researchmethods/qass/gilbert-agent-based-models-2e>. There you will find the code for the collectivities model and other resources.

# GLOSSARY

## **agent**

A computer program, or part of a program, that can be considered to act autonomously and that represents an individual, organization, nation-state, or other social actor.

## **analogical model**

A model that is based on an analogy between the target being modeled and the form of the model.

## **attribute**

A characteristic or feature of an agent. It may be set at the beginning of a simulation, and may alter in value during the run to indicate changes in the agent.

## **boundedly rational**

There are many situations where making a perfectly rational decision would involve infinite computation or require infinite amounts of information. It is therefore assumed that people are boundedly rational—that is, that they are limited in the amount of cognitive processing in which they can engage when making decisions.

## **buffer**

An area of computer memory used to store values temporarily, often on a first-in, first-out basis.

## **chromosome**

In a genetic algorithm, a chromosome is a set of parameters that defines a proposed solution. The chromosome is often composed of a sequence of binary digits, or of floating point or integer values.

## **class**

In object-oriented programming, a class is a specification of a

type of object, showing what attributes and methods instances of that class would have.

### **classifier system**

(Short for learning classifier system.) First described by Holland (1975), a classifier system consists of a collection of binary rules. A genetic algorithm modifies and selects the best rules. The fitness of a rule is decided by a reinforcement learning technique.

### **companion modeling**

A form of modeling in which models are developed in close association with the people who might be represented in the model and who might benefit from the knowledge and understanding that the model yields.

### **control**

In the social and medical sciences, an experiment is typically carried out on two similar groups, one of which receives the treatment while the other, the control group, does not. The outcomes in the two groups are then compared.

### **crossover**

In evolutionary computation, a method of creating a new chromosome from corresponding parts of its parents' chromosomes. Crossover is often used in genetic algorithms.

### **environment**

The simulated surroundings in which an agent is located, possibly including simulated physical elements and other agents.

### **equation-based model**

A model consisting of one or more equations that relate variables describing a system (Van Dyke Parunak, Savit, & Riolo, 1998).

### **equifinality**

The idea that similar outcomes may be obtained from different initial conditions and in many different ways.



**fitness**

In evolutionary computation, a measure of the adequacy of an individual within an environment. The fitness is used to determine the likelihood that the individual will reproduce and pass parts of its chromosome to the next generation.

**framework**

A program or library intended to make it easier to develop agent-based models. The framework provides some standardized components and possibly a basic design for the model.

**genetic algorithm (GA)**

A method of simulating evolution. A population of individuals each with a fixed-length chromosome is evolved by employing crossover and mutation operators and a fitness function that determines how likely individuals are to reproduce. GAs perform a type of search in a fitness landscape, aiming to find an individual that has optimum fitness.

**geographical information system (GIS)**

A type of database in which a common spatial coordinate system is the primary means of reference. GISs provide facilities for data input, storage, retrieval, and representation; data management, transformation, and analysis; and data reporting and visualization.

**global variable**

A variable whose value may be accessed and set throughout a program, rather than only within some restricted context.

**ideal type**

An ideal-type model is one that is formed by the one-sided accentuation of one or more characteristics to create a unified analytical construct that abstracts from the variety actually found in concrete social phenomena.

**instantiate**

The process of forming an object (in an object-oriented computer language) by following the specifications represented by a class.

The object is an instance of the class.

**message**

A symbolic communication between two agents, often represented as a string of characters.

**method**

In object-oriented programming, a piece of program code associated with a class that performs some function, often in response to a message received by an instance of the class.

**micro foundation**

Assumptions and theories about individual behavior that inform the design of agents. The actions of the agents are expected to lead to the emergence of features that correspond to real-world macro phenomena.

**model**

A simplified representation of some social phenomenon. Executing or running the model yields a simulation whose behavior is intended to mirror some social process or processes.

**modeling environment**

A computer program that allows the user to create, execute, and visualize the outputs of a simulation.

**patch**

In NetLogo, a cell of the grid that constitutes the environment for a model's agents (called turtles in NetLogo).

**phase change**

An abrupt change in the state of a system as a whole, consequent on a small change in one variable. By analogy to phase transitions in materials, for example, the change from a solid to a liquid when the material's temperature is raised through its melting point.

**policy**

In reinforcement learning a policy maps states of the world to the actions the agent ought to take in those states.

### **power law**

A relationship between two variables such that one is proportional to a power of the other. If one takes logarithms of each variable, the relationship between the logged variables is linear and can be represented as a straight line on a plot of the two logged variables. Many relationships between variables describing complex systems follow a power law.

### **production (rule) system**

A problem-solving system consisting of a knowledge base of rules and general facts, a working memory of facts concerning the current case, and an inference engine for manipulating both. The rules are generally of the form, "If [condition], then [action]."

### **reinforcement learning (RL)**

A type of machine learning concerned with how an agent ought to take actions in an environment to maximize some long-term reward.

### **research question**

A question whose answer can be found by carrying out research. It needs to be sufficiently specific that the research has a reasonable likelihood of obtaining a result, but not so specific that the results will be of limited use or generality.

### **retrodict**

To make predictions about the past. Normal predictions estimate what will happen in the future from the basis of some theory and assumptions; retrodictions use theory and assumptions to estimate a past state. Because the past state can be known empirically through measurements, retrodiction can be used as a method of assessing the validity of the theory and assumptions.

### **scale model**

A model in which the simplifications of reality come mainly from

making the model smaller than the target being modeled.

**sensitivity analysis**

A systematic analysis of changes in simulation results as the model's parameters are changed. Sensitivity analysis is used to assess the extent to which the outcomes are dependent on the precise parameter values that have been assumed.

**simulate**

To run a model and observe its behavior through time.

**spatially explicit**

A spatially explicit model is one in which geography is represented within the model, for example by locating all simulated objects on a grid or other spatial representation. Spatially explicit simulations often use a geographical information system to manage the location of objects.

**stylized fact**

A simplified presentation of an empirical finding that, although broadly true, may ignore particular exceptions. Usually, stylized facts describe whole societies or economies rather than the characteristics of individuals.

**target**

The social phenomenon or process that is represented by a model.

**toroid**

A donut-shaped object that can be constructed by rotating a circle around an axis external to the circle. Topologically, a toroid is formed by connecting the left and right edges, and then the top and bottom edges, of a rectangle.

**treatment**

In an experiment, the application of some process selectively to the treatment group, while leaving the control group unaffected. Changes to the treatment group that are not found in the control

group are considered to be due to the treatment.

**validation**

The process of checking that a model is a good representation of the target.

**verification**

The process of checking that a model conforms to its specification—that is, that it does not include errors, also called bugs.

## REFERENCES

- Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. P. (2017). Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24, 13–33. <https://doi.org/10.1016/j.cosrev.2017.03.001>
- Afshar, M., & Asadpour, M. (2010). Opinion formation by informed agents. *Journal of Artificial Societies and Social Simulation*, 13(4), 5. <https://doi.org/10.18564/jasss.1665>
- Ahrweiler, P., Pyka, A., & Gilbert, N. (2011). A new model for university-industry links in knowledge-based economies. *Journal of Product Innovation Management*, 28(2), 218–235. <https://doi.org/10.1111/j.1540-5885.2010.00793.x>
- Ahrweiler, P., Schilperoord, M., Pyka, A., & Gilbert, N. (2014). Testing policy options for Horizon 2020 with SKIN. In N. Gilbert, P. Ahrweiler, & A. Pyka (Eds.), *Simulating knowledge dynamics in innovation networks* (pp. 155–183). Heidelberg, Germany: Springer. [https://doi.org/10.1007/978-3-662-43508-3\\_7](https://doi.org/10.1007/978-3-662-43508-3_7)
- Ahrweiler, P., Schilperoord, M., Pyka, A., & Gilbert, N. (2015). Modelling research policy: Ex-ante evaluation of complex policy instruments. *Journal of Artificial Societies and Social Simulation*, 18(4), 5. <https://doi.org/10.18564/jasss.2927>
- Albino, V., Giannoccaro, I., & Carbonara, N. (2003). Coordination mechanisms based on cooperation and competition within industrial districts: An agent-based computational approach. *Journal of Artificial Societies and Social Simulation*, 6(4), 3. Available at

<http://jasss.soc.surrey.ac.uk/6/4/3.html>

Allan, R. (2010). Survey of agent based modelling and simulation tools. STFC Daresbury Laboratory, Warrington, UK. Available at <http://www.grids.ac.uk/Complex/ABMS/>

An, L. (2012). Modeling human decisions in coupled human and natural systems: Review of agent-based models. *Ecological Modelling*, 229, 25–36.  
<https://doi.org/10.1016/j.ecolmodel.2011.07.010>

Anderson, B. D. O., & Ye, M. (2019). Recent advances in the modelling and analysis of opinion dynamics on influence networks. *International Journal of Automation and Computing* (Institute of Automation, Chinese Academy of Sciences), 16(2), 129–149.  
<https://doi.org/10.1007/s11633-019-1169-8>

Andrighetto, G., Campenni, M., Conte, R., & Paolucci, M. (2007). On the emergence of norms: A normative agent architecture. *AAAI Symposium, Social and Organizational Aspects of Intelligence*. Available at <http://www.aaai.org/Papers/Symposia/Fall/2007/FS-07-04/FS07-04-003.pdf>

Andrighetto, G., Castelfranchi, C., Mayor, E., McBreen, J., Lopez-Sanchez, M., & Parsons, S. (2013). (Social) norm dynamics. In G. Andrighetto, G. Governatori, P. Noriega, P., & L. W. N. van der Torre (Eds.), *Normative multi-agent systems* (pp. 135–170). Dagstuhl, Germany: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (Dagstuhl Follow-Ups).  
<https://doi.org/10.4230/DFU.Vol4.12111.135>

Argonne National Laboratory. (2018). How Argonne makes super models. Available at <https://www.anl.gov/es/article/how-argonne->

[makes-super-models](#)

Arthur, W. B., Holland, J. H., LeBaron, B. D., Palmer, R. G., & Tayler, P. (1997). Asset pricing under endogenous expectations in an artificial stock market. *SSRN Electronic Journal*, 1001, 48109.  
<https://doi.org/10.2139/ssrn.2252>

Axelrod, R. M. (1997a). Advancing the art of simulation in the social sciences. In R. Conte, R. Hegselmann, & P. Terna (Eds.), *Simulating social phenomena. Lecture Notes in Economics and Mathematical Systems*, vol 456. Springer, Berlin, Heidelberg.  
[https://doi.org/10.1007/978-3-662-03366-1\\_2](https://doi.org/10.1007/978-3-662-03366-1_2)

Axelrod, R. M. (1997b). *The complexity of cooperation: Agent-based models of competition and collaboration*. Princeton, NJ: Princeton University Press (Princeton Studies in Complexity).

Axelrod, R. M. (1997c). The dissemination of culture. *Journal of Conflict Resolution*, 41(2), 203–226.

Axelrod, R. M., & Dawkins, R. (1990). *The evolution of cooperation*. Harmondsworth, UK: Penguin.

Axtell, R. L., Epstein, J. M., Dean, J. S., Gumerman, G. J., Swedlund, A. C., Harburger, J., Chakravarty, S., Hammond, R., Parker, J., & Parker, M. (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.  
<https://doi.org/10.1073/pnas.092080799>

Bagnall, A. J., & Smith, G. D. (2005). A multiagent model of the UK market in electricity generation. *Evolutionary Computation*, 9(5),



522–536.

Balke, T., & Gilbert, N. (2014). How do agents make decisions? A survey. *Journal of Artificial Societies and Social Simulation*, 17(4), 13. <https://doi.org/10.18564/jasss.2687>

Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2010). *Discrete-event system simulation*, 5th ed. London: Pearson.

Banzhaf, W. (1998). *Genetic programming: An introduction; On the automatic evolution of computer programs and its applications*. San Francisco: Morgan Kaufmann.

Barabási, A.-L. (2003). *Linked: How everything is connected to everything else and what it means for business, science, and everyday life*. New York: Plume.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439), 509–512. <https://doi.org/10.1126/science.286.5439.509>

Barnaud, C., van Paassen, A., Trébuil, G., Promburom, T., & Bousquet, F. (2010). Dealing with power games in a companion modelling process: Lessons from community water management in Thailand highlands. *Journal of Agricultural Education and Extension*, 16(1), 55–74. <https://doi.org/10.1080/13892240903533152>

Barreteau, O. (2003). Our companion modelling approach. *Journal of Artificial Societies and Social Simulation*, 6(2), 1. Available at <http://jasss.soc.surrey.ac.uk/6/2/1.html>

Barreteau, O., Bousquet, F., & Attonaty, J.-M. (2001). Role-playing games for opening the black box of multi-agent systems: Method and lessons of its application to Senegal River Valley irrigated systems. *Journal of Artificial Societies and Social Simulation*, 4(2), 5. Available at <http://jasss.soc.surrey.ac.uk/4/2/5.html>

Batten, D., & Grozev, G. (2006). NEMSIM: Finding ways to reduce greenhouse gas emissions using multi-agent electricity modelling. In P. Perez & D. Batten (Eds.), *Complex science for a complex world: Exploring human ecosystems with agents* (pp. 227–252). Canberra, Australia: ANU Press. Available at <https://www.jstor.org/stable/j.ctt2jbhz2.17>

Batty, M. (2013). *The new science of cities*. Cambridge: Massachusetts Institute of Technology Press.

Bazzan, A. L. C., & Klügl, F. (2014). A review on agent-based technology for traffic and transportation. *Knowledge Engineering Review* (Cambridge University Press), 29(03), 375–403. <https://doi.org/10.1017/S0269888913000118>

Benard, S., & Willer, R. (2007). A wealth and status-based model of residential segregation. *Journal of Mathematical Sociology*, 31(2), 149–174. <https://doi.org/10.1080/00222500601188486>

Benenson, I., & Hatna, E. (2009). The third state of the Schelling model of residential dynamics. Available at <http://arxiv.org/abs/0910.2414> (Accessed August 7, 2018).

Bersini, H. (2012). UML for ABM. *Journal of Artificial Societies and Social Simulation*, 15(1), 9. <https://doi.org/10.18564/jasss.1897>

Boden, M. A. (1988). Computer models of mind: Computational approaches in theoretical psychology. Cambridge, UK: Cambridge University Press.

Boella, G., van der Torre, L., & Verhagen, H. (2007). Introduction to Normative Multiagent Systems. In G. Boella, L. van der Torre, & H. Verhagen (Eds.), Normative multi-agent systems. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (Dagstuhl Seminar Proceedings). Available at <http://drops.dagstuhl.de/opus/volltexte/2007/918/pdf/07122.Verhage>

Boero, R., Castellani, M., & Squazzoni, F. (2004). Micro behavioural attitudes and macro technological adaptation in industrial districts: An agent-based prototype. Journal of Artificial Societies and Social Simulation, 7(2), 1. Available at <http://jasss.soc.surrey.ac.uk/7/2/1.html>

Boero, R., & Squazzoni, F. (2005). Does empirical embeddedness matter? Methodological issues on agent based models for analytical social science. Journal of Artificial Societies and Social Simulation, 8(4), 6. Available at <http://jasss.soc.surrey.ac.uk/8/4/6.html> (Accessed August 23, 2018).

Bommel, P., Becu, N., Le Page, C., & Bousquet, F. (2016). Cormas: An agent-based simulation platform for coupling human decisions with computerized dynamics. In T. Kaneda, H. Kanegae, Y. Toyoda, & P. Rizzi (Eds.), Simulation and gaming in the network society (pp. 387–410). Translation. Singapore: Springer. [https://doi.org/10.1007/978-981-10-0575-6\\_27](https://doi.org/10.1007/978-981-10-0575-6_27)

Borrelli, F., Ponsiglione, C., Zollo, G., & Iandoli, L. (2005). Inter-organizational learning and collective memory in small firms

clusters: An agent-based approach. *Journal of Artificial Societies and Social Simulation*, 8(3), 4. Available at <http://jasss.soc.surrey.ac.uk/8/3/4.html>

Bourdieu, P. (1986). *Distinction: A social critique of the judgement of taste*. Abingdon, UK: Routledge.

Bratman, M. E., Israel, D. J., & Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3), 349–355. <https://doi.org/10.1111/j.1467-8640.1988.tb00284.x>

Brenner, T. (2001). Simulating the evolution of localised industrial clusters; An identification of the basic mechanisms. *Journal of Artificial Societies and Social Simulation*, 4(3), 4. Available at <http://jasss.soc.surrey.ac.uk/4/3/4.html>.

Brewer, M. B. (1991). The social self: On being the same and different at the same time. *Personality and Social Psychology Bulletin*, 17(5), 475–482. <https://doi.org/10.1177/0146167291175001>

Bruch, E. E., & Mare, R. D. (2006). Neighborhood choice and neighborhood change. *American Journal of Sociology*, 112(3), 667–709. <https://doi.org/10.1086/507856>

Bull, L. (2004). Learning classifier systems: A brief introduction. In L. Bull (Ed.), *Applications of learning classifier systems* (pp. 1–12). Berlin: Springer. [https://doi.org/10.1007/978-3-540-39925-4\\_1](https://doi.org/10.1007/978-3-540-39925-4_1)

Bunn, D., & Oliveira, F. S. (2001). Agent-based simulation: An application to the new electricity trading arrangements of England and Wales. *Evolutionary Computation*, 2001, 5(5), 493–503. <https://doi.org/10.1109/4235.956713>

Caillou, P., Gaudou, B., Grignard, A., Truong, C. Q., & Taillandier, P. (2017). A simple-to-use BDI architecture for agent-based modeling and simulation. In W. Jager, R. Verbrugge, A. Flache, G. de Roo, L. Hoogduin, & C. Hemelrijk (Eds.), *Advances in social simulation 2015* (pp. 15–28). Berlin: Springer. [https://doi.org/10.1007/978-3-319-47253-9\\_2](https://doi.org/10.1007/978-3-319-47253-9_2)

Cairney, P., Heikkila, T., & Wood, M. (2019). *Making policy in a complex world: Elements in public policy*. Cambridge: Cambridge University Press. <https://doi.org/10.1017/9781108679053>

Castle, C. J. E., & Crooks, A. T. (2006). Principles and concepts of agent-based modelling for developing geospatial simulations. CASA working paper series. Available at <http://discovery.ucl.ac.uk/3342/>

Challet, D., Marsili, M., & Zhang, Y.-C. (2013). *Minority games*. Oxford: Oxford University Press.

Chang, K.-T. (2004). *Introduction to geographic information systems* (2nd ed). Boston: McGraw-Hill.

Chatfield, C. (2004). *The analysis of time series: An introduction* (6th ed.). Boca Raton, FL: Chapman & Hall.

Chattoe, E. (1998). Just how (un)realistic are evolutionary algorithms as representations of social processes? *Journal of Artificial Societies and Social Simulation*, 1(3), 2. Available at <http://jasss.soc.surrey.ac.uk/1/3/2.html>

Chattoe, E., Saam, N. J., & Möhring, M. (2000). Sensitivity analysis in the social sciences: Problems and prospects. In R. Suleiman, G. N. Gilbert, & K. Troitzsch (Eds.), *Tools and techniques for social*

science simulation (pp. 243–273). Heidelberg: Physica-Verlag HD.  
[https://doi.org/10.1007/978-3-642-51744-0\\_13](https://doi.org/10.1007/978-3-642-51744-0_13)

Chen, Y., & Tang, F. (1998). Learning and incentive-compatible mechanisms for public goods provision: An experimental study. *Journal of Political Economy*, 106, 633–662.

Clark, W. A. V. (1991). Residential preferences and neighborhood racial segregation: A test of the Schelling segregation model. *Demography*, 28(1), 1–19. <https://doi.org/10.2307/2061333>

Cobb, C. W., & Douglas, P. H. (1928). A theory of production. *American Economic Review*, 18(Supplement), 139–165.

Cohen, P. R., & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42(2–3), 213–261. Available at <http://www.cs.uu.nl/docs/vakken/iag/CohLev.intention.pdf>

D'Aquino, P., Barreteau, O., & Le Page, C. (2003). Role-playing games, models and negotiation processes. *Journal of Artificial Societies and Social Simulation*, 6(2), 10. Available at <http://jasss.soc.surrey.ac.uk/6/2/10.html>

D'Aquino, P., Bousquet, F., Le Page, C., & Bah, A. (2003). Using self-designed role-playing games and a multi-agent system to empower a local decision-making process for land use management: The self-cormas experiment in Senegal. *Journal of Artificial Societies and Social Simulation*, 6(3), 5. Available at <http://jasss.soc.surrey.ac.uk/6/3/5.html>

Deffuant, G. (2006). Comparing extremism propagation patterns in continuous opinion models. *Journal of Artificial Societies and Social*

Simulation, 9(3), 8. Available at  
<http://jasss.soc.surrey.ac.uk/9/3/8.html>

Deffuant, G., Amblard, F., & Weisbuch, G. (2002). How can extremism prevail? A study based on the relative agreement interaction model. *Journal of Artificial Societies and Social Simulation*, 5(4), 1. Available at <http://jasss.soc.surrey.ac.uk/5/4/1.html>

Dibble, C., & Feldman, P. G. (2004). The GeoGraph 3D Computational Laboratory: Network and terrain landscapes for RePast. *Journal of Artificial Societies and Social Simulation*, 7(1), 7. Available at <http://jasss.soc.surrey.ac.uk/7/1/7.html>

Dignum, F., Kinny, D., & Sonenberg, L. (2002). From desires, obligations and norms to goals. *Cognitive Science Quarterly*, 2(3–4), 407–430.

Dray, A., Perez, P., Jones, N., Le Page, C., D'Aquino, P., White, I., ..., & Dray, A. (2006). The AtollGame experience: From knowledge engineering to a computer-assisted role playing game. *Journal of Artificial Societies and Social Simulation*, 9(1), 6. Available at <http://jasss.soc.surrey.ac.uk/9/1/6.html>

Dubbelboer, J., Nikolic, I., Jenkins, K., & Hall, J. (2017). An agent-based model of flood risk and insurance. *Journal of Artificial Societies and Social Simulation*, 20(1), 6. <https://doi.org/10.18564/jasss.3135>

Dunham, J. B. (2005). An agent-based spatially explicit epidemiological model in MASON. *Journal of Artificial Societies and Social Simulation*, 9(1), 3. Available at <http://jasss.soc.surrey.ac.uk/9/1/3.html>

Edmonds, B. (2006). The emergence of symbiotic groups resulting from skill-differentiation and tags. *Journal of Artificial Societies and Social Simulation*, 9(1), 10. Available at <http://jasss.soc.surrey.ac.uk/9/1/10.html>

Edmonds, B., & Hales, D. (2003). Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation*, 6(4), 11. Available at <http://jasss.soc.surrey.ac.uk/6/4/11.html>

El-Tawil, S., Fang, J., Aguirre, B., & Best, E. (2017). A computational study of the station nightclub fire accounting for social relationships. *Journal of Artificial Societies and Social Simulation*, 20(4), 10. <https://doi.org/10.18564/jasss.3519>

Elias, N. (1939). *The civilising process*. Oxford: Blackwell.

EMIL Project Consortium. (2008). Emergence in the loop: Simulating the two way dynamics of norm innovation; Deliverable 3.3 EMIL-S: The simulation platform. Available at [emil.istc.cnr.it/file\\_download/7/D3.3.Project033841.EMIL.pdf](http://emil.istc.cnr.it/file_download/7/D3.3.Project033841.EMIL.pdf)

Epstein, J. M. (2007). *Generative social science: Studies in agent-based computational modeling*. Princeton, NJ: Princeton University Press.

Epstein, J. M. (2008). Why model? *Journal of Artificial Societies and Social Simulation*, 11(4), 12. <https://doi.org/10.1080/01969720490426803>

Epstein, J. M., & Axtell, R. (1996). *Growing artificial societies: Social science from the bottom up*. Washington, DC: Brookings Institution



Press (Complex adaptive systems).

Erev, I., & Roth, A. E. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, 88(4), 848–881.

Étienne, M. (2003). SYLVOPAST: A multiple target role-playing game to assess negotiation processes in sylvopastoral management planning. *Journal of Artificial Societies and Social Simulation*, 6(2), 5. Available at <http://jasss.soc.surrey.ac.uk/6/2/5.html>

Étienne, M. (ed.). (2014). *Companion modelling*. Dordrecht, the Netherlands: Springer. <https://doi.org/10.1007/978-94-017-8557-0>

Étienne, M., Cohen, M., & Le Page, C. (2003). A step-by-step approach to building land management scenarios based on multiple viewpoints on multi-agent system simulations. *Journal of Artificial Societies and Social Simulation*, 6(2), 2. Available at <http://jasss.soc.surrey.ac.uk/6/2/2.html>

Etter, A. (2016). *Modern technical writing: An introduction to software documentation*. Kindle, Amazon. Available at <https://www.amazon.co.uk/Modern-Technical-Writing-Introduction-Documentation-ebook/dp/B01A2QL9SS>

Farmer, J. D., Patelli, P., & Zovko, I. I. (2005). The predictive power of zero intelligence in financial markets. *Proceedings of the National Academy of Sciences*, 102(6), 2254–2259. <https://doi.org/10.1073/pnas.0409157102>

Farrenkopf, T., Guckert, M., Urquhart, N., & Wells, S. (2016). Ontology based business simulations. *Journal of Artificial Societies and*

Social Simulation, 19(4), 14. <https://doi.org/10.18564/jasss.3266>

Field, A. P., & Iles, J. (2016). *An adventure in statistics: The reality enigma*. London: Sage.

Fielding, J., & Gilbert, N. (2005). *Understanding social statistics*, 2nd ed. London: Sage.

Fink, E. C., Gates, S., & Humes, B. D. (1998). *Game theory topics: Incomplete information, repeated games and n-player games*. Thousand Oaks, CA: Sage (Quantitative Applications in the Social Sciences).

Fioretti, G. (2001). Information structure and behaviour of a textile industrial district. *Journal of Artificial Societies and Social Simulation*, 4(4), 1. Available at <http://jasss.soc.surrey.ac.uk/4/4/1.html>

Flache, A., & Macy, M. (2002). Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 99(Suppl. 3), 7229–7236.

Flache, A., Mäs, M., Feliciani, T., Chattoe-Brown, E., Deffuant, G., Huet, S., & Lorenz, J. (2017). Models of social influence: Towards the next frontiers. *Journal of Artificial Societies and Social Simulation*, 20(4), 2. <https://doi.org/10.18564/jasss.3521>

Forster, J. W. (1971). *World dynamics*. Cambridge: Massachusetts Institute of Technology Press.

Fossett, M., & Waren, W. (2005). *Overlooked implications of ethnic*

preferences for residential segregation in agent-based models. *Urban Studies*, 42(11), 1893–1917.  
<https://doi.org/10.1080/00420980500280354>

Fowler, M. (2003). *UML distilled*, 3rd ed. Boston: Addison-Wesley Professional.

Friedman-Hill, E. (2003). *Jess in action: Rule-based systems in Java*. Greenwich, CT: Manning.

Fröhlich, P., & Link, J. (2003). *Unit testing in Java: How tests drive the code*. San Francisco, Calif.: Morgan Kaufmann.

Galan, J. M., & Izquierdo, L. R. (2005). Appearances can be deceiving: Lessons learned re-implementing Axelrod's "evolutionary approach to norms." *Journal of Artificial Societies and Social Simulation*, 8(3), 2. Available at  
<http://jasss.soc.surrey.ac.uk/8/3/2.html>

Galitsky, B. (2002). Extending the BDI model to accelerate the mental development of autistic patients. In *Proceedings of the 2nd International Conference on Development and Learning (ICDL'02)* (pp. 82–88). <https://doi.org/10.1109/DEVLRN.2002.1011803>

Gaylord, R. J., & D'Andria, L. (1998). *Simulating society: A mathematica toolkit for modeling socioeconomic behavior*. New York: Springer Verlag.

Georgeff, M., & Ingrand, F. (1990). Real-time reasoning: The monitoring and control of spacecraft systems. *Artificial Intelligence Applications, 1990, Sixth Conference on*, 198–204.  
<https://doi.org/10.1109/CAIA.1990.89190>

Ghazi, S., Khadir, T., & Dugdale, J. (2014). Multi-agent based simulation of environmental pollution issues: A review. In International Conference on Practical Applications of Agents and Multi-Agent Systems (pp. 13–21). Cham, Switzerland: Springer-Verlag. [https://doi.org/10.1007/978-3-319-07767-3\\_2](https://doi.org/10.1007/978-3-319-07767-3_2)

Gilbert, N. (1997). A simulation of the structure of academic science. *Sociological Research Online*, 2(2), 3. Available at <http://www.socresonline.org.uk/socresonline/2/2/3.html>

Gilbert, N. (2002). Varieties of emergence. In D. Sallach (Ed.), *Agent 2002: Social agents: Ecology, exchange, and evolution* (pp. 41–56). Chicago: University of Chicago and Argonne National Laboratory. Available at [https://www.researchgate.net/profile/Nigel\\_Gilbert/publication/22879:of-emergence.pdf](https://www.researchgate.net/profile/Nigel_Gilbert/publication/22879:of-emergence.pdf)

Gilbert, N. (2010). *Computational social science, SAGE benchmarks in social research methods series*. London: Sage.

Gilbert, N. (2019). *Collectivities (Version 1.1.0)*. CoMSES Computational Model Library. Available at <https://doi.org/10.25937/qmc0-r354>

Gilbert, N., Ahrweiler, P., Barbrook-Johnson, P., Narasimhan, K. P., & Wilkinson, H. (2018). Computational modelling of public policy: Reflections on practice. *Journal of Artificial Societies and Social Simulation*, 21(1), 14. <https://doi.org/10.18564/jasss.3669>

Gilbert, N., Ahrweiler, P., & Pyka, A. (2014). *Simulating knowledge dynamics in innovation networks*. Berlin: Springer (Understanding Complex Systems). <https://doi.org/10.1007/978-3-662-43508-3>

Gilbert, N., & Banks, S. (2002). Platforms and methods for agent-based modeling. *Proceedings of the National Academy of Sciences*, 99(Supplement 3), 7197–7198.

<https://doi.org/10.1073/pnas.072079499>

Gilbert, N., den Besten, M., Bontovics, A., Craenen, B. G. W. W., Divina, F., Eiben, A. E. E., ..., & Yang, L. (2006). Emerging artificial societies through learning. *Journal of Artificial Societies and Social Simulation*, 9(2). <http://jasss.soc.surrey.ac.uk/9/2/9.html> (Accessed: June 28, 2010).

Gilbert, N., Pyka, A., & Ahrweiler, P. (2001). Innovation networks—A simulation approach. *Journal of Artificial Societies and Social Simulation*, 4(3), 8. Available at

<http://jasss.soc.surrey.ac.uk/4/3/8.html>

Gimblett, H. R. (2002). *Integrating geographic information systems and agent-based modeling techniques for simulating social and ecological processes*. London: Oxford University Press.

Goolsby, R. (2006). Combating terrorist networks: An evolutionary approach. *Computational and Mathematical Organization Theory*, 12, 7–20.

Gotts, N., Matthews, R., Gilbert, N., Polhill, G., & Roach, A. (2007). Agent-based land-use models: A review of applications. *Landscape Ecology*, 22(10), 1447–1459.

Grignard, A., Taillandier, P., Gaudou, B., Vo, D. A., Huynh, N. Q., & Drogoul, A. (2013). GAMA 1.6: Advancing the art of complex agent-based modeling and simulation. In G. Boella, E. Elkind, B. T. R. Savarimuthu, F. Dignum, & M. K. Purvis (Eds.), *PRIMA 2013: Principles and practice of multi-agent systems*. PRIMA 2013 (pp.

117–131). Berlin: Springer. [https://doi.org/10.1007/978-3-642-44927-7\\_9](https://doi.org/10.1007/978-3-642-44927-7_9)

Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., ..., & DeAngelis, D. L. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1–2), 115–126. <https://doi.org/10.1016/j.ecolmodel.2006.04.023>

Grimm, V., Polhill, G., & Touza, J. (2017). Documenting social simulation models: The ODD protocol as a standard. In B. Edmonds & R. Meyer (Eds.), *Simulating social complexity* (pp. 117–133). Berlin: Springer.

Grimm, V., & Railsback, S. F. (2012). Pattern-oriented modelling: A “multi-scope” for predictive systems ecology. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences (The Royal Society)*, 367(1586), 298–310. <https://doi.org/10.1098/rstb.2011.0180>

Groeneveld, J., Müller, B., Buchmann, C. M., Dressler, G., Guo, C., Hase, ..., & Schwarz, N. (2017). Theoretical foundations of human decision-making in agent-based land use models: A review. *Environmental Modelling & Software (Elsevier)*, 87, 39–48. <https://doi.org/10.1016/J.ENVSOFT.2016.10.008>

Guerci, E., Rastegar, M. A., & Cincotti, S. (2010). Agent-based modeling and simulation of competitive wholesale electricity markets. In S. Rebennack, P. M. Pardalos, M. V. F. Pereira, & N. A. Iliadis (Eds.), *Handbook of power systems II* (pp. 241–286). Berlin: Springer. [https://doi.org/10.1007/978-3-642-12686-4\\_9](https://doi.org/10.1007/978-3-642-12686-4_9)

Hales, D. (2000). Cooperation without space or memory: Tags, groups and the prisoner’s dilemma. In P. Davidsson & S. Moss (Eds.),

Multi-agent-based simulation. MABS 2000 (pp. 157–166). Berlin: Springer-Verlag (Lecture Notes in Artificial Intelligence).

Hales, D. (2002). Evolving specialisation, altruism and group-level optimisation using tags. In J. S. Sichman, P. Davidsson, & F. Bousquet (Eds.), *Lecture notes in artificial intelligence* (pp. 26–35). Berlin: Springer-Verlag.

Hamill, L., & Gilbert, N. (2009). Social circles: A simple structure for agent-based social network models. *Journal of Artificial Societies and Social Simulation*, 12(2), 3. Available at <http://jasss.soc.surrey.ac.uk/12/2/3.html> (Accessed: August 20, 2018).

Hamill, L., & Gilbert, N. (2015). *Agent-based modelling in economics*. Chichester, UK: John Wiley & Sons.  
<https://doi.org/10.1002/9781118945520>

Hammond, R. A. (2015). Considerations and best practices in agent-based modeling to inform policy. In R. Wallace, A. Geller, & V. Ogawa (Eds.), *Assessing the use of agent-based models for tobacco regulation* (Appendix A). Washington DC: National Academies Press. Available at <https://www.ncbi.nlm.nih.gov/books/NBK305917/>

Hatna, E., & Benenson, I. (2012). The Schelling model of ethnic residential dynamics: Beyond the integrated-segregated dichotomy of patterns. *Journal of Artificial Societies and Social Simulation*, 15(1), 6. <https://doi.org/10.18564/jasss.1873>

Hauke, J., Lorscheid, I., & Meyer, M. (2017). Recent development of social simulation as reflected in JASSS between 2008 and 2014: A citation and co-citation analysis. *Journal of Artificial Societies and*

Social Simulation, 20(1), 5. <https://doi.org/10.18564/jasss.3238>

Hegselmann, R. (2017). Thomas C. Schelling & James M. Sakoda: The intellectual, technical, and social history of a model. *Journal of Artificial Societies and Social Simulation*, 20(3), 15. <https://doi.org/10.18564/jasss.3511>

Heppenstall, A. J. J., Crooks, A. T., See, L. M., & Batty, M. (2012). Agent-based models of geographical systems. <https://doi.org/10.1007/978-90-481-8927-4>

Heywood, D. I., Cornelius, S., & Carver, S. (2011). An introduction to geographical information systems. Saddle River, NJ: Prentice Hall.

Hirsch, G. B., & Homer, J. B. (2006). System dynamics modeling for public health: Background and opportunities. *American Journal of Public Health*, 96(3), 452–458.

HM Treasury. (2015). The aqua book: Guidance on producing quality analysis for government. Available at <https://assets.publishing.service.gov.uk/government/uploads/system>. (Accessed: August 30, 2018).

Hodkinson, P. (2002). Goth identity, style and subculture. Oxford: Berg.

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press.

Huberman, B. A., & Glance, N. (1993). Evolutionary games and computer simulations. *Proceedings of the National Academy of*



Sciences, 90, 7716–7718.

Huet, S., Bouvier, A., Poursat, M.-A., & Jolivet, E. (2004). Statistical tools for nonlinear regression. New York: Springer-Verlag (Springer Series in Statistics). <https://doi.org/10.1007/b97288>

Izquierdo, S. S., & Izquierdo, L. R. (2007). The impact of quality uncertainty without asymmetric information on market efficiency. *Journal of Business Research*, 60(8), 858–867. <https://doi.org/10.1016/j.jbusres.2007.02.010>

Izquierdo, S. S., Izquierdo, L. R., & Gotts, N. M. (2008). Reinforcement learning dynamics in social dilemmas. *Journal of Artificial Societies and Social Simulation*, 11(2), 1. Available at <http://jasss.soc.surrey.ac.uk/11/2/1.html> (Accessed: August 17, 2018).

Jacobsen, C., & Hanneman, R. A. (1992). Illegal drugs: Past, present and possible futures. *Journal of Drug Issues*, 22(1), 105–120. <https://doi.org/10.1177/002204269202200107>

Jager, W. (2017). Enhancing the Realism of Simulation (EROS): On implementing and developing psychological theory in social simulation. *Journal of Artificial Societies and Social Simulation*, 20(3), 14. <https://doi.org/10.18564/jasss.3522>

Janssen, M. A. (2017). The practice of archiving model code of agent-based models. *Journal of Artificial Societies and Social Simulation*, 20(1), 2. <https://doi.org/10.18564/jasss.3317>

Janssen, M. A., & Jager, W. (1999). An integrated approach to simulating behavioural processes: A case study of the lock-in of

consumption patterns. *Journal of Artificial Societies and Social Simulation*, 2(2), 2. Available at <http://jasss.soc.surrey.ac.uk/2/2/2.html>

Jara, H. X., & Tumino, A. (2013). Tax-benefit systems, income distribution and work incentives in the European Union. *International Journal of Microsimulation*, 1(6), 27–62. Available at <https://econpapers.repec.org/RePEc:ijm:journl:v:1:y:2013:i:issnum:6:62>

Johnson, P. E. (2002). Agent-based modeling: What I learned from the artificial stock market. *Social Science Computer Review*, 20, 174–186.

Kahneman, D. (2003). Maps of bounded rationality: Psychology for behavioral economics. *American Economic Review*, 93(5), 1449–1475.

Kaldor, N. (1961). Capital accumulation and economic growth. In *The Theory of Capital* (pp. 177–222). London: Palgrave Macmillan UK. [https://doi.org/10.1007/978-1-349-08452-4\\_10](https://doi.org/10.1007/978-1-349-08452-4_10)

Kangur, A., Jager, W., Verbrugge, R., & Bockarjova, M. (2017). An agent-based model for diffusion of electric vehicles. *Journal of Environmental Psychology*, 52, 166–182. <https://doi.org/10.1016/J.JENVP.2017.01.002>

Kleijnen, J. P. C. (2015). *Design and analysis of simulation experiments*. Cham, Switzerland: Springer International (International Series in Operations Research & Management Science). <https://doi.org/10.1007/978-3-319-18087-8>

Klemm, K., Eguíluz, V. M., Toral, R., & Miguel, M. S. (2003). Global culture: A noise-induced transition in finite systems. *Physical Review E*, 67(4), 045101.

<https://doi.org/10.1103/PhysRevE.67.045101>

Knoke, D., & Yang, S. (2008). *Social network analysis*, 2nd ed.

Thousand Oaks, CA: Sage. <https://doi.org/10.4135/9781412985864>

Koesrindartoto, D., Sun, J., & Tesfatsion, L. (2005). An agent-based computational laboratory for testing the economic reliability of wholesale power market designs. *Power Engineering Society General Meeting, 2005. IEEE. San Francisco: IEEE Power Engineering Society*, (January), 2818–2823, Vol. 3.

<https://doi.org/10.1109/PES.2005.1489273>

Kollingbaum, M. J., & Norman, T. J. (2003). Norm adoption in the NoA agent architecture. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 1038–1039). New York: ACM.

<http://doi.acm.org/10.1145/860575.860784>

Kollingbaum, M. J., & Norman, T. J. (2004). Norm adoption and consistency in the NoA agent architecture. In M. Dastani, J. Dix, & A. El Fallah-Seghrouchni (Eds.), *Programming multi-agent systems* (pp. 169–186). Berlin: Springer.

[https://doi.org/10.1007/978-3-540-25936-7\\_9](https://doi.org/10.1007/978-3-540-25936-7_9)

Koza, J. R. (1992). *Genetic programming*. Cambridge: Massachusetts Institute of Technology Press.

Koza, J. R. (1994). *Genetic programming 2*. Cambridge: Massachusetts Institute of Technology Press.

Krause, U., & Hegselmann, R. (2002). Opinion dynamics and bounded confidence models, analysis and simulation. *Journal of Artificial Societies and Social Simulation*, 5(3), 2. Available at <http://jasss.soc.surrey.ac.uk/5/3/2.html>

Kurahashi-Nakamura, T., Mäs, M., & Lorenz, J. (2016). Robust clustering in generalized bounded confidence models. *Journal of Artificial Societies and Social Simulation*, 19(4), 7. <https://doi.org/10.18564/jasss.3220>

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1), 1–64.

Laurie, A. J., & Jaggi, N. K. (2003). Role of “vision” in neighbourhood racial segregation: A variant of the Schelling segregation model. *Urban Studies*, 40(13), 2687–2704. <https://doi.org/10.1080/0042098032000146849>

Lawson, B. G., & Park, S. (2000). Asynchronous time evolution in an artificial society model. *Journal of Artificial Societies and Social Simulation*, 3(1), 2. Available at <http://jasss.soc.surrey.ac.uk/3/1/2.html> (Accessed: August 28, 2018).

Lee, J.-S., Filatova, T., Ligmann-Zielinska, A., Hassani-Mahmoei, B., Stonedahl, F., Lorscheid, I., ..., & Parker, D. C. (2015). The complexities of agent-based modeling output analysis. *Journal of Artificial Societies and Social Simulation*, 18(4). <https://doi.org/10.18564/jasss.2897>

Li, J., & O'Donoghue, C. (2013). A survey of dynamic microsimulation models: Uses, model structure and methodology. *International*

Journal of Microsimulation, 6, 3–55.  
<https://doi.org/10.1093/jae/ejm029>

Lorenz, J. (2006). Consensus strikes back in the Hegselmann-Krause Model of continuous opinion dynamics under bounded confidence. *Journal of Artificial Societies and Social Simulation*, 9(1), 8. Available at <http://jasss.soc.surrey.ac.uk/9/1/8.html>

Lorscheid, I., Heine, B.-O., & Meyer, M. (2012). Opening the “black box” of simulations: Increased transparency and effective communication through the systematic design of experiments. *Computational and Mathematical Organization Theory*, 18(1), 22–62. <https://doi.org/10.1007/s10588-011-9097-3>

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., & Balan, G. (2005). MASON: A Java multi-agent simulation environment. *Simulation: Transactions of the Society for Modeling and Simulation International*, 81(7), 517–527.

Luke, S., R. Simon, A. Crooks, H. Wang, E. Wei, D. Freelan, ... & C. Cioffi-Revilla. (2019). The MASON Simulation Toolkit: Past, present, and future. In P. Davidsson & H. Verhagen (Eds.), *Multi-agent-based simulation XIX. MABS 2018. Lecture Notes in Computer Science*, vol 11463. Springer, Cham. [https://doi.org/10.1007/978-3-030-22270-3\\_6](https://doi.org/10.1007/978-3-030-22270-3_6)

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

Mahdavi Ardestani, B., O'Sullivan, D., & Davis, P. (2018). A multi-scaled agent-based model of residential segregation applied to a real metropolitan area. *Computers, Environment and Urban*

Systems (Pergamon), 69, 1–16.

<https://doi.org/10.1016/J.COMPENVURBSYS.2017.11.002>

Maini, V., & Sabri, S. (2018). Machine learning for humans. Online.

Available at

[https://www.dropbox.com/s/e38nil1dnl7481q/machine\\_learning.pdf?dl=0](https://www.dropbox.com/s/e38nil1dnl7481q/machine_learning.pdf?dl=0) (Accessed: August 17, 2018).

Malerba, F., Nelson, R., Orsenigo, L., & Winter, S. (2001). History-friendly models: An overview of the case of the computer industry. *Journal of Artificial Societies and Social Simulation*, 4(3), 6.

Available at <http://jasss.soc.surrey.ac.uk/4/3/6.html>

March, J. G., Cohen, M. D., & Olsen, J. P. (1972). A garbage can model of organizational choice. *Administrative Science Quarterly*, 17(1), 1–25.

Marengo, L. (1992). Coordination and organizational learning in the firm. *Journal of Evolutionary Economics*, 2(3), 313–326.

McKeown, G., & Sheehy, N. (2006). Mass media and polarisation processes in the bounded confidence model of opinion dynamics. *Journal of Artificial Societies and Social Simulation*, 9(1), 11.

Available at <http://jasss.soc.surrey.ac.uk/9/1/11.html>

McMillon, D., Simon, C. P., & Morenoff, J. (2014). Modeling the underlying dynamics of the spread of crime. *PLoS ONE* (edited by M. Perc, Public Library of Science), 9(4), e88923.

<https://doi.org/10.1371/journal.pone.0088923>

Merton, R. K. (1968). *Social theory and social structure* (1968 enl.). New York: Free Press.

- Meyer, M., & Hufschlag, K. (2006). A generic approach to an object-oriented learning classifier system library. *Journal of Artificial Societies and Social Simulation*, 9(3), 9. Available at <http://jasss.soc.surrey.ac.uk/9/3/9.html> (Accessed: August 22, 2018).
- Michalewicz, Z., & Fogel, D. B. (2004). *How to solve it: Modern heuristics*. Berlin: Springer. <https://doi.org/10.1007/978-3-662-07807-5>
- Moss, S. (2002). Policy analysis from first principles. *Proceedings of the National Academy of Sciences*, 99(Suppl. 3), 7267–7274.
- Mulkay, M. J., & Turner, B. S. (1971). Over-production of personnel and innovation in three social settings. *Sociology*, 5(1), 47–61.
- Müller, B., Bohn, F., Dreßler, G., Groeneveld, J., Klassert, C., Martin, R., ..., & Schwarz, N. (2013). Describing human decisions in agent-based models – ODD + D, an extension of the ODD protocol. *Environmental Modelling & Software*, 48, 37–48. <https://doi.org/10.1016/J.ENVSOFT.2013.06.003>
- Naveh, I., & Sun, R. (2006). A cognitively based simulation of academic science. *Computational and Mathematical Organization Theory*, 12(4), 313–337. <https://doi.org/10.1007/s10588-006-8872-z>
- Nilsson, N. (1998). *Artificial intelligence: A new synthesis*. San Francisco: Morgan Kaufmann.
- Norling, E. J. (2014). *Modelling human behaviour with BDI agents* (Doctoral dissertation, Computer Science and Software Engineering, The University of Melbourne, Australia). Available at

<http://hdl.handle.net/11343/37081>

North, M. J., Collier, N. T., & Vos, J. R. (2006). Experiences creating three implementations of the repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 16(1), 1–25. <https://doi.org/10.1145/1122012.1122013>

O'Donoghue, C. (2014). *Handbook of microsimulation modelling*. Bingley, UK: EmeraldInsight. <https://doi.org/10.1108/S0573-855520140000293020>

O'Kelly, M. E., & Fotheringham, A. S. (1989). *Spatial interaction models: Formulations and applications*. Dordrecht, Netherlands: Kluwer.

O'Sullivan, D., & Perry, G. L. W. (2013). *Spatial simulation: Exploring pattern and process*. Chichester, UK: John Wiley. <https://doi.org/10.1002/9781118527085>

Orcutt, G., Quinke, H., & Merz, J. (1986). *Microanalytic simulation models to support social and financial policy*. Amsterdam: North-Holland (Information research and resource reports, v.7).

Pajares, J., Hernández-Iglesias, C., & López-Paredes, A. (2004). Modelling learning and R&D in innovative environments: A cognitive multi-agent approach. *Journal of Artificial Societies and Social Simulation*, 7(2), 7. Available at <http://jasss.soc.surrey.ac.uk/7/2/7.html>

Papert, S. (1983). *Mindstorms: Children, computers and powerful ideas*. *New Ideas in Psychology*, 1(1), 87. [https://doi.org/10.1016/0732-118X\(83\)90034-X](https://doi.org/10.1016/0732-118X(83)90034-X)



- Phillips, A. W. (1950). Mechanical models in economic dynamics. *Economica*, 17(67), 283–305.
- Piccinini, G., & Bahar, S. (2013). Neural computation and the computational theory of cognition. *Cognitive Science*, 37(3), 453–488. <https://doi.org/10.1111/cogs.12012>
- Poggio, T., Lo, A. W., LeBaron, B. D., & Chan, N. T. (2001). Agent-based models of financial markets: A comparison with experimental markets. *SSRN Electronic Journal*.  
<https://doi.org/10.2139/ssrn.290140>
- Polhill, J. G., Parker, D., Brown, D., & Grimm, V. (2008). Using the ODD protocol for describing three agent-based social simulation models of land-use change. *Journal of Artificial Societies and Social Simulation*, 11(2), 3. Available at  
<http://jasss.soc.surrey.ac.uk/11/2/3.html> (Accessed: April 12, 2019).
- Poli, R., Langdon, W. B., & McPhee, N. F. (2008). A field guide to genetic programming. Web site. <http://www.gp-field-guide.org.uk>
- Pollicott, M., & Weiss, H. (2001). The dynamics of Schelling-type segregation models and a nonlinear graph Laplacian variational problem. *Advances in Applied Mathematics*, 27(1), 17–40.  
<https://doi.org/10.1006/aama.2001.0722>
- Portugali, J., Benenson, I., & Omer, I. (2010). Sociospatial residential dynamics: Stability and instability within a self-organizing city. *Geographical Analysis*, 26(4), 321–340.  
<https://doi.org/10.1111/j.1538-4632.1994.tb00329.x>
- Powell, W. W., White, D. R., Koput, K. W., & Owen-Smith, J. (2005).

Network dynamics and field evolution: The growth of interorganizational collaboration in the life sciences. *American Journal of Sociology* (University of Chicago Press), 110(4), 1132–1205.

Pujol, J. M., Flache, A., Delgado, J., & Sangüesa, R. (2005). How can social networks ever become complex? Modelling the emergence of complex networks from local social exchanges. *Journal of Artificial Societies and Social Simulation*, 8(4), 12. Available at <http://jasss.soc.surrey.ac.uk/8/4/12.html> (Accessed August 22, 2018).

Pyka, A., Ahrweiler, P., & Gilbert, N. (2004). Simulating knowledge dynamics in innovation networks. In M. Richiardi & R. Leombruni (Eds.), *Industry and labor dynamics: The agent-based computational economics approach* (pp. 284–296). Singapore: World Scientific Press. Available at <http://link.springer.com/book/10.1007/978-3-662-43508-3>

Railsback, S. F. (2001). Concepts from complex adaptive systems as a framework for individual-based modelling. *Ecological Modelling*, 139(1), 47–62. [https://doi.org/10.1016/S0304-3800\(01\)00228-9](https://doi.org/10.1016/S0304-3800(01)00228-9)

Railsback, S. F., & Grimm, V. (2012). *Agent-based and individual-based modeling: A practical introduction*. Princeton, NJ: Princeton University Press.

Railsback, S. F., Lytinen, S. L., & Jackson, S. K. (2006). Agent-based simulation platforms: Review and development recommendations. *Simulation: Transactions of the Society for Modeling and Simulation International*, 82(9), 609–623. <https://doi.org/10.1177/0037549706073695>

Reschke, C. H. (2001). Evolutionary perspectives on simulations of social systems. *Journal of Artificial Societies and Social Simulation*, 4(4), 8. Available at <http://jasss.soc.surrey.ac.uk/4/4/8.html> (Accessed: August 22, 2018).

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4), 25–34.

Richiardi, M., Leombruni, R., Saam, N., & Sonnessa, M. (2006). A common protocol for agent-based social simulation. *Journal of Artificial Societies and Social Simulation*, 9(1), 15. Available at <http://jasss.soc.surrey.ac.uk/9/1/15.html>

Ringler, P., Keles, D., & Fichtner, W. (2016). Agent-based modelling and simulation of smart electricity grids and markets: A literature review. *Renewable and Sustainable Energy Reviews*, 57, 205–215. <https://doi.org/10.1016/J.RSER.2015.12.169>

Riolo, R. L., Cohen, M. D., & Axelrod, R. M. (2001). Evolution of cooperation without reciprocity. *Nature*, 411, 441–443.

Ritter, F. E., Schoelles, M. J., Quigley, K. S., & Klein, L. C. (2011). Determining the number of simulation runs: Treating simulations as theories by not sampling their behavior. In L. Rothrock & S. Narayanan (Eds.), *Human-in-the-loop simulations* (pp. 97–116). London: Springer. [https://doi.org/10.1007/978-0-85729-883-6\\_5](https://doi.org/10.1007/978-0-85729-883-6_5)

Robinson, S. (2004). *Simulation: The practice of model development and use*. Chichester, UK: Wiley.

Rouchier, J. (2003). Re-implementation of a multi-agent model aimed at sustaining experimental economic research: The case of

simulations with emerging speculation. *Journal of Artificial Societies and Social Simulation*, 6(4), 7. Available at <http://jasss.soc.surrey.ac.uk/6/4/7.html>

Ruankaew, N., Le Page, C., Dumrongrojwattana, P., Barnaud, C., Gajasen, N., van Paassen, A., & Trébuil, G. (2010). Companion modelling for integrated renewable resource management: A new collaborative approach to create common values for sustainable development. *International Journal of Sustainable Development & World Ecology*, 17(1), 15–23. <https://doi.org/10.1080/13504500903481474>

Russell, S., & Norvig, P. (2010). *Artificial intelligence: A modern approach*, 3rd ed. Upper Saddle River, NJ: Prentice Hall.

Rutter, C. M., Zaslavsky, A. M., & Feuer, E. J. (2011). Dynamic microsimulation models for health outcomes. *Medical Decision Making*, 31(1), 10–18. <https://doi.org/10.1177/0272989X10369005>

Sakoda, J. M. (1971). The checkerboard model of social interaction. *Journal of Mathematical Sociology*, 1(1), 119–131.

Sallans, B., Pfister, A., Karatzoglou, A., & Dorffner, G. (2003). Simulation and validation of an integrated markets model. *Journal of Artificial Societies and Social Simulation*, 6(4), 2. Available at <http://jasss.soc.surrey.ac.uk/6/4/2.html>

Sander, R., Schreiber, D., & Doherty, J. (2000). Empirically testing a computational model: The example of housing segregation. In D. Sallach & T. Wolsko (Eds.), *Proceedings at the Workshop on Simulation of Socialagents: Architectures and Institutions* (pp. 109–116). Chicago: ANL/DIS/TM-60, Argonne National Laboratory.

Schelling, T. C. (1971). Dynamic models of segregation. *Journal of Mathematical Sociology*, 1, 143–186.

Schelling, T. C. (1978). *Micromotives and macrobehavior*. New York: Norton.

Simmel, G. (1907). Fashion. *International Quarterly*, 10, 130–155.

Simon, H. A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics*, 69(1), 99–118.

Squazzoni, F. (2012). *Agent-based computational sociology*. Chichester, UK: John Wiley & Sons.  
<https://doi.org/10.1002/9781119954200>

Squazzoni, F., & Boero, R. (2002). At the edge of variety and coordination. An agent-based computational model of industrial district. *Journal of Artificial Societies and Social Simulation*, 5(1), 1. Available at <http://jasss.soc.surrey.ac.uk/5/1/1.html>

Stauffer, D., Sousa, A., & Schulze, C. (2004). Discretized opinion dynamics of the deffault model on scale-free networks. *Journal of Artificial Societies and Social Simulation*, 7(3), 7. Available at <http://jasss.soc.surrey.ac.uk/7/3/7.html>

Stefanelli, A., & Seidl, R. (2017). Opinion communication on contested topics: How empirics and arguments can improve social simulation. *Journal of Artificial Societies and Social Simulation*, 20(4), 3.  
<https://doi.org/10.18564/jasss.3492>

Sterman, J. (2000). *Business dynamics: Systems thinking and*

modeling for a complex world. Boston: Irwin McGraw-Hill.

Stonedahl, F., & Wilensky, U. (2011). Finding forms of flocking: Evolutionary search in ABM parameter-spaces. In T. Bosse, A. Geller, & C. M. Jonker (Eds.), *Multi-agent-based simulation XI* (pp. 61–75). Berlin: Springer. [https://doi.org/10.1007/978-3-642-18345-4\\_5](https://doi.org/10.1007/978-3-642-18345-4_5)

Strader, T. J., Lin, F., & Shaw, M. J. (1998). Simulation of order fulfillment in divergent assembly supply chains. *Journal of Artificial Societies and Social Simulation*, 1(2), 5. Available at <http://jasss.soc.surrey.ac.uk/1/2/5.html>

Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 79–99). New York: Cambridge University Press.

Sun, R., & Naveh, I. (2004). Simulating organizational decision-making using a cognitively realistic agent model. *Journal of Artificial Societies and Social Simulation*, 7(3), 5. Available at <http://jasss.soc.surrey.ac.uk/7/3/5.html>

Sutherland, H., & Figari, F. (2013). EUROMOD: The European Union tax-benefit microsimulation model. *International Journal of Microsimulation*, 1(6), 4–26. Available at [http://www.microsimulation.org/IJM/V6\\_1/2\\_IJM\\_6\\_1\\_Sutherland\\_Fi](http://www.microsimulation.org/IJM/V6_1/2_IJM_6_1_Sutherland_Fi) (Accessed: August 30, 2018).

Sutherland, H., Paulus, A., & Figari, F. (2014). Micro-simulation and policy analysis. In A. Atkinson & F. Bourguignon (Eds.), *Handbook of income distribution*. Vol. 2B (pp. 2141–2221). Amsterdam: Elsevier.

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, 2nd ed. Cambridge: Massachusetts Institute of Technology Press.
- Swedlund, A. C., Sattenspiel, L., Warren, A. L., & Gumerman, G. J. (2015). Modeling archaeology: Origins of the artificial Anasazi project and beyond. In G. Wurzer, K. Kowarik, & H. Reschreiter (Eds.), *Advances in geographic information science* (pp. 37–50). Cham, Switzerland: Springer. [https://doi.org/10.1007/978-3-319-00008-4\\_2](https://doi.org/10.1007/978-3-319-00008-4_2)
- Taatgen, N., Anderson, J., & Lebiere, C. (2006). Modeling paradigms in ACT-R. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 29–52). New York: Cambridge University Press.
- Thorngate, W. (2000). Teaching social simulation with Matlab. *Journal of Artificial Societies and Social Simulation*, 3(1). Available at <http://jasss.soc.surrey.ac.uk/3/1/forum/1.html>
- Tobias, R., & Hofmann, C. (2004). Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), 6. Available at <http://jasss.soc.surrey.ac.uk/7/1/6.html>
- Troitzsch, K. G. (2004). Validating simulation models. In *Proceedings of 18th European Simulation Multiconference on Networked Simulation and Simulation Networks* (pp. 265–270). Wilhelmshaven, UK: SCS Europe.
- Van Dyke Parunak, H., Savit, R., & Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. In J. S. Sichman, N. Gilbert, & R. Conte (Eds.), *Multi-agent*

systems and agent-based simulation. MABS 1998 (pp. 10–25). Paris, July 4–6. Berlin: Springer.  
[https://doi.org/10.1007/10692956\\_2](https://doi.org/10.1007/10692956_2)

van Ham, M., Manley, D., Bailey, N., Simpson, L., & Maclennan, D. (2012). Understanding neighbourhood dynamics: New insights for neighbourhood effects research. In M. van Ham, D. Manley, N. Bailey, L. Simpson, & D. Maclennan (Eds.), *Understanding neighbourhood dynamics* (pp. 1–21). Dordrecht: Springer Netherlands. [https://doi.org/10.1007/978-94-007-4854-5\\_1](https://doi.org/10.1007/978-94-007-4854-5_1)

Wahl, S., & Spada, H. (2000). Children's reasoning about intentions, beliefs and behaviour. *Cognitive Science Quarterly*, 1(1), 3–32.

Walbert, H. J., Caton, J. L., & Norgaard, J. R. (2018). Countries as agents in a global-scale computational model. *Journal of Artificial Societies and Social Simulation*, 21(3), 4.  
<https://doi.org/10.18564/jasss.3717>

Waldrop, M. M. (2017). News feature: Special agents offer modeling upgrade. *Proceedings of the National Academy of Sciences*, 114(28), 7176–7179. <https://doi.org/10.1073/pnas.1710350114>

Watkins, A., Noble, J., Foster, R. J., Harmsen, B. J., & Doncaster, C. P. (2015). A spatially explicit agent-based model of the interactions between jaguar populations and their habitats. *Ecological Modelling*, 306, 268–277. <https://doi.org/10.1016/J.ECOLMODEL.2014.10.038>

Watts, C., & Gilbert, N. (2014a). *Simulating innovation: Computer-based tools for rethinking innovation*. Cheltenham, UK: Edward Elgar. <https://doi.org/10.4337/9781783472536>



- Watts, C., & Gilbert, N. (2014b). Simulating innovation: Comparing models of collective knowledge, technological evolution and emergent innovation networks. In B. Kamiński & G. Koloch (Eds.), *Advances in intelligent systems and computing* (pp. 189–200). Berlin: Springer. [https://doi.org/10.1007/978-3-642-39829-2\\_17](https://doi.org/10.1007/978-3-642-39829-2_17)
- Watts, D. J. (1999). Network dynamics and the small world phenomenon. *American Journal of Sociology*, 105(2), 493–527.
- Watts, D. J. (2004). The “new” science of networks. *Annual Review of Sociology*, 30(1), 243–270.  
<https://doi.org/10.1146/annurev.soc.30.020404.104342>
- Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of “small-world” networks. *Nature*, 393(6684), 440–442.
- Widdicombe, S., & Wooffitt, R. C. (1990). “Being” versus “doing” punk (etc): On achieving authenticity as a member. *Journal of Language and Social Psychology*, 9, 257–277.
- Wilensky, U. (1998). NetLogo Segregation model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. Available at <http://ccl.northwestern.edu/netlogo/models/Segregation>
- Wilensky, U. (1999). NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
- Wilensky, U. (2005). NetLogo wolf sheep predation (system dynamics) model. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

- Wilensky, U., & Rand, W. (2015). An introduction to agent-based modeling: Modeling natural, social, and engineered complex systems with NetLogo. Cambridge: Massachusetts Institute of Technology Press.
- Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2016). Good enough practices in scientific computing. CoRR, abs/1609.0. Available at <http://arxiv.org/abs/1609.00037>
- Windrum, P., Fagiolo, G., & Moneta, A. (2007). Empirical validation of agent-based models: Alternatives and prospects. *Journal of Artificial Societies & Social Simulation*, 10(2), 8. Available at <http://jasss.soc.surrey.ac.uk/10/2/8.html>
- Wray, R. E., & Jones, R. M. (2005). Considering SOAR as an agent architecture. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 53–78). Cambridge: Cambridge University Press.  
<https://doi.org/10.1017/CBO9780511610721.004>
- Ye, M., & Carley, K. M. (1995). RADAR-SOAR: Towards an artificial organization composed of intelligent agents. *Journal of Mathematical Sociology*, 20(2–3), 219–246.
- Zhang, J. (2004). Residential segregation in an all-integrationist world. *Journal of Economic Behavior & Organization*, 54(4), 533–550.  
<https://doi.org/10.1016/j.jebo.2003.03.005>
- Zheng, N., Waraich, R. A., Axhausen, K. W., & Geroliminis, N. (2012). A dynamic cordon pricing scheme combining the macroscopic fundamental diagram and an agent-based traffic model. *Transportation Research Part A: Policy and Practice*, 46(8), 1291–1303. <https://doi.org/10.1016/J.TRA.2012.05.006>

# INDEX

Abstract models, 62–63

Agent-based modeling

- advantages of, 2, 3

- agent-to-agent interactions, 5–6, 17, 48

- characteristic features, 16–18

- companion modeling, 15–16

- complexity, 66

- as computational method, 2

- consumer behaviors, 10–11

- definition, 1–6

- design elements (See Model design)

- energy market models, 13–14

- GIS and, 31–33

- industrial networks, 11–12, 33

- learning simulations, 18

- opinion dynamics, 8–10, 17

- planning considerations, 72–74

public policy decisions, 14–15, 77–78  
research reporting, 74–77  
Schelling model, 6–8, 7 (figure) 38, 63  
supply chain management, 12–13  
traffic simulations, 8  
validation approaches, 62–66  
*See also* Agents; Simulation

## Agents

actions of, 44–45, 47–48  
Belief-Desires-Intention, 28–29  
boundedly rational, 17–18  
characteristics of, 24  
in collectivities model, 56–59, 58 (figure)  
environment and, 17, 30–33  
genetic algorithms, 36–37  
heterogeneous, 16–17  
interaction between, 5–6, 17, 48  
in NetLogo simulations, 52–54  
normative, 29–30

as objects, 24–26  
reinforcement learning, 14, 27  
rules of behavior, 26, 39, 56

Ahrweiler, P., 12

Amblard, F., 9

Analogical models, 4–5

AnyLogic framework, 50, 70 (table)

Attonaty, J.-M., 15

Attributes, object, 25

Axelrod, R. M., 33

Axtell, R. L., 65

Barreteau, O., 15

Belief-Desires-Intention (BDI) model, 28–29

Bockarjova, M., 10

Boid models, 47

Boundedly rational, 17–18

Bousquet, F., 15

Bruch, E. E., 63

Bryan, J., 62

Buffer, 31

Caton, J. L., 6

Children's Reasoning about Intentions, Beliefs, and Behavior (CRIBB) model, 29

Chromosome, 36–37

Class, object, 25

Classifier system, 36–37

Cobb-Douglas production function, 5

Cognitive models, 28–29

Collectivities model

- agents' actions in, 47–48, 49 (figure), 56–59, 58 (figure)

- macrolevel features, 44–45

- microlevel behaviors, 46

- programming code for, 56–59, 57 (figure), 58 (figure)

- research question, 44

Companion modeling, 15–16

Computational methods

- advantages of, 2

experiments and, 2–3

social science models, 3–4

Consumer behavior models, 10–11

Control group, 3

CORMAS simulation environment, 51, 70 (table)

Cranston, K., 62

CRIBB model, 29

Crossover process, 37

Cultural traits, 33–34

Dawkins, R., 33

Dean, J. S., 65

Debugging, 59–62

Deffuant, G., 9

Design of Experiments, 68

Diary, project, 72

Discrete event simulation, 22–23

El Farol Bar model, 47

EMIL project, 30

Energy market models, 13–14

Environment

- agents and, 30–33

- features of, 31, 70 (table), 71

- geographical information system (GIS), 31–33

- reinforcement learning and, 27

- spatially explicit, 6, 17

Epstein, J. M., 59, 65

Equation-based models, 5

Equifinality, 42

Ethical considerations, 78

Etter, A., 73

Experiments, 2–3

Facsimile models, 65

Fagiolo, G., 71

Filatova, T., 68

Fitness, individual, 36

Frameworks, 50, 70 (table)



GAMA simulation environment, 51, 70 (table)

Genetic algorithm, 36–37

Geographical information system (GIS), 31–33

Gilbert, N., 12

Gimblett, H. R., 17

Global variables, 53

*Growing Artificial Societies* (Epstein & Axtell), 62

Gumerman, G. J., 65

Hassani-Mahmoei, B., 68

Hydraulic model of economy, 4

Ideal-type models, 4

Industrial networks, 11–12, 33

Innovation networks, 11–12, 47

Instantiated classes, 25

Izquierdo, L. R., 11

Izquierdo, S. S., 11

Jager, W., 10

Janssen, M. A., 10

Java programming language, 50

Jess (Java Expert System Shell) toolkit, 26

Kangur, A., 10

Kitzes, J., 62

Klein, L. C., 68

Kollingbaum, M. J., 30

Learning, simulation of, 18

Lee, J.-S., 68

Leombruni, R., 74

Ligmann-Zielinska, A., 68

Lin, F., 12

Macrolevel regularities

agent interactions and, 42

in collectivities model, 44–45

stylized facts, 38–39

Malerba, F., 64

Mare, R. D., 63

Mason framework, 50, 70 (table)

Methods, 25

Microsimulation modeling, 18–20

Model design

- agent specifications, 39

- code development, 39

- environment, 30–33

- preliminary steps, 40–42 (table)

- randomness, 33–34, 43

- research question, 38–39

- time considerations, 34–36

- validation, 42–43, 62–69

- verification, 42, 59–62

- See also* Model programming code

Model programming code

- for collectivities model, 56–59, 57 (figure), 58 (figure)

- framework features, 50–51

- modeling environments, 51–52

pseudo-code, 56–57, 57 (figure)

UML, 56

validation techniques, 66–69

verification techniques, 59–62

## Model types

abstract, 62–63

advantages of, 3–4

analogical, 4–5

Belief-Desires-Intention, 28–29

boid, 47

cognitive, 27–30

complexity, 65–66

discrete event simulation, 22–23

environment and, 30–33

equation-based, 5

facsimile, 65

ideal-type, 4

innovation, 47

microsimulation, 18–20

scale, 4

system dynamics, 20–22, 21 (figure)

target of, 3–4

Moneta, A., 71

Nederbragt, L., 62

Nelson, R., 64

NetLogo Library, 39, 51

agent-based simulations in, 52–54, 54 (figure), 55 (figure)

Code tab, 52

features of, 51–52, 70 (table)

Information tab, 52

Interface tab, 52, 53 (figure)

patches, 52

Norgaard, J. R., 6

Norling, E. J., 29

Norman, T. J., 30

Normative agents, 29–30

Object-oriented programs, 24–26

OpenABM web site, 39, 76–77

Opinion dynamics, 8–10, 17

Orsenigo, L., 64

Overview, Design concepts, and Details (ODD), 74

Participative research methods, 15–16

Phillips, A. W., 4

Planning considerations, 72–74

Power law, 64

Production rule systems, 26

Pseudo-code, 56–57, 57 (figure)

Public policy modeling, 14–15, 77–78

Pyka, A., 12

Quality assurance, 78

Query-Based Model Exploration, 68

Quigley, K. S., 68

Randomness, 33–34

Rationality, 17–18

Regression equation, 2, 69

Reinforcement learning, 14, 27

Repast framework, 50, 70 (table)

Research question, 38–39, 44

Research reporting, 74–77

Retrodict, 65

Richiardi, M., 74

Ritter, F. E., 68

ROAMEF cycle, 14–15

Saam, N., 74

Sattenspiel, L., 65

Scale models, 4

Schelling, Thomas, 6

Schelling model, 6–8, 7 (figure), 38, 63

Schoelles, M. J., 68

Sensitivity analysis, 42–43, 66–68

Shaw, M. J., 12

Simon, S. A., 17

The Sims game, 2

## Simulation

discrete event, 22–23

of learning, 18

microsimulation, 18–19

model behavior and, 3

modeling environments, 51–52, 70 (table), 71

planning considerations, 72–74

supply chains and, 12–13

validation techniques, 69–71

Social networks, 11, 33

Sonnessa, M., 74

Spatially explicit environments, 6, 32

Strader, T. J., 12

Stylized facts, 38–39

Supply chain management models, 12–13

Surface, model, 47–48

Swarm framework, 50

Swedlund, A. C., 65

Systems dynamics model, 20–22, 21 (figure)



Systems-based approaches, 1

Teal, T. K., 62

Tesfatsion, Leigh, 80

Time considerations, 34–36

Toroid surface, 47–48

Traffic simulation, 8

Unified Modeling Language, 56

Unit testing, 61

Validation

- for abstract models, 62–63

- data comparisons, 68–71

- for middle-range models, 64

- sensitivity analysis, 42–43, 66–68

Variable-based approaches, 1

Variables

- global, 53

- output, 69

power law relationship, 64

raster and vector, 32

regression equation and, 2

Verbrugge, R., 10

Verification, 42

Verification techniques, 59–62

Virtual experiments, 3

Walbert, H. J., 6

Warren A. L., 65

Weisbuch, G., 9

Wilensky, U., 52

Wilson, G., 62

Windrum, P., 71

Winter, S., 64