DOI: 10.1177/0956797615605798

Supplemental Material

Agent-Based Modeling

Agent-based modeling (ABM) is a computational approach to simulate large social systems. ABM is widely employed in the social sciences, particularly within economics (e.g., Tesfatsion & Judd, 2006), sociology (e.g., Schelling, 1971; Granovetter, 1978), and political science (e.g., Axelrod, 1984). ABM is also employed within fields such as evolutionary biology, epidemiology and public health, and computer science. Use of ABM within psychology has been somewhat more limited, likely due to a greater emphasis on the processes that operate within individuals (see Goldstone & Janssen, 2005; Smith & Collins, 2009; Richie, Yang, & Coppola, 2014, for exceptions). At the heart of any ABM is a computational description of three inter-related elements: the individual agents, how those agents interact with one another, and the environment the agents inhabit. Once these descriptions are constructed and initial conditions are specified, the model is run, allowing the system to evolve on its own.

ABM thus consists of computational descriptions of individuals but ultimately yields phenomena at the aggregate level. For this reason, ABM is particularly useful in testing assumptions about individuals with respect to the consequences such assumptions have on collective behavior. In sociology, such linkages are sometimes referred to as micro-macro theories (Sawyer, 2003) and non-trivial micro-macro linkages are often referred to as emergent phenomena. For many, the ability to relate separate levels of analysis represents one of the major strengths of ABM and makes it clear why researchers in a variety of fields have found value in ABM. Economists can simulate existing markets (Bonabeau, 2002) and evaluate counterfactuals scenarios such as changes in existing policy. Political scientists can evaluate theories of political communication and persuasion (e.g., Muis, 2010). Sociologists can evaluate

DOI: 10.1177/0956797615605798

the spread of trends, both beneficial (Centola, 2010, 2011) and maladaptive (Schelling, 1971; Granovetter, 1978).

The goals of a specific ABM endeavor can fall into one (or more) of several categories, each of which could hold value within the field of psychology. Axelrod and Tesfatsion (2006), for example, suggest that goals can be empirical, normative, or heuristic in nature (though there are undoubtedly many other potential goals). Empirical goals would consist of investigations into the "how" and "why" associated with empirically observed phenomena, typically phenomena that exist at the level of collective behavior. For example, people generally act in accordance with social norms even when doing so is personally costly. A variety of ABM work (e.g., Axelrod, 1997) has gone into addressing how such regularities arise and why they persist even in the absence of top-down control (e.g., government policy). Normative endeavors, in contrast, seek to address how to intervene on social interactions and environments so as to achieve particular objectives. For example, economists might be interested in evaluating policies intended to encourage costly contributions to a public good (e.g., Korobow, Johnson, Axtell 2007). Epidemiologists might seek practices that minimize the spread of disease (e.g., Meyers, Newman, Martin, & Schrag, 2003).

A third type of goal discussed by Axelrod and Tesfatsion (2006) is heuristic. Heuristic ABM endeavors seek to arrive at general principles about the fundamental factors underlying collective patterns of behavior. Importantly, heuristic goals do not require the ABM to be absolutely correct in its details. For example, Schelling's classic ABM model (1971) investigated the processes that underlie residential segregation, finding that absolute segregation can arise in the absence of both centralized control (e.g., without governmental policies) and local intent (e.g., without overly malicious preferences about racial mixing). Schelling's model is extremely simplistic, and thus unlikely to capture the rich details of psychology of racism. However, the ultimate impact of the model was to demonstrate that dramatic racial segregation can arise as an emergent property of weak racial preferences.

Below we provide additional details of the ABM presented in the current article, using it as an illustrative example of ABM more generally. Here, we provide additional technical details, particularly details of the model's implementation. We also highlight implementation choices that were made and alternatives choices that could have been made. In doing so, we hope that the materials provide guidance for readers unfamiliar with ABM but who might be contemplating its use in their own work.

The Current Study

Model Details

In the current study, as with all ABMs, we specified the three elements of the model: the agents, their interactions, and the environment. Because the current simulations emphasize the social influences on memory, our agents are essentially simple models of memory. As with most computational models of cognitive processes, our agent model specifies two, inter-related components: the representations that allow the agents to store information and the processes that operate over those representations.

Each agent has two related representations. The first reflects the elements that may be retrieved and the probability with which they will be retrieved. We refer to these as activations and the set of activations as the activation vector, denoted as *A*. We have chosen to allow our agents to represent a total of 40 items, but this choice was essentially arbitrary (though it is well within the range used in typical behavioral experiments). The second component of the memory model represents the associations between individual items. These associations represent pre-experimental knowledge such as the semantic associations between words (e.g., knowing that "cat" and "dog" are both common pets). These associations can be thought of as a square matrix (40 x 40), with the element in column *i* and row *j* representing the (directional) association between items *i* and *j*. This matrix is denoted as *S*. Because the current simulations do not directly explore the influence of pre-experimental knowledge per se, elements in *S* were assigned small, random values between –0.2 and 0.2. As we mention in the article, *S* could be formulated to reflect more interesting prior knowledge. For example, an experiment using categorized words could be simulated by constraining within-category associations (e.g., those between "cat" and "dog") to be relatively high and constraining between-category associations (e.g., those between "cat" and "wrench") to be relatively low.

The processes that operate over these representations are invoked when agents engage in either of the two behaviors they agents possess. These processes are discussed in the article. However, we briefly summarize them here to situate them within the larger description of our implementation. Agents can encode items and can retrieve items. Encoding of an item, *i*, essentially increases the activation of A_i. However, the encoding process first reduces the activation of the maximally active item is first reduced if *i* is not the maximally active item. Finally, *A* is normalized so that *A* to be interpreted as a proper probability distribution. During retrieval, an agent successfully retrieves an item with a fixed probability. If retrieval is successful, an item is selected in proportion to the activations in *A* with more active items being more likely and less active items being less likely. Finally, the activations in *A* are modified in response to successful retrieval. Semantic associates of the successfully retrieved item activations modified. If the retrieved item was not the most active item in *A*, the activation of the

DOI: 10.1177/0956797615605798

most active item as well as its semantic associates are modified. The successfully retrieved item is then encoded.

The details of the encoding and retrieval could be modified in a variety of ways.

Implementation Details

There are software packages specifically designed for the purposes of constructing, running, and analyzing ABMs (discussed below). The simulations reported in the current article, however, were written in Python. Python is a general-purpose programming language that has gained tremendous popularity in recent years, particularly within the field of scientific computing (e,g., Oliphant, 2007; Perez & Granger, 2007). The syntax of the language is relatively straightforward compared to more traditional programming languages (such as C and Java) and thus presents somewhat less of a learning curve for new programmers (or those only familiar with other languages). Furthermore, Python is an interpreted programming language, meaning that code can be run immediately once it is written rather than requiring a relatively lengthy compile-build-run cycle. When compared to compiled languages, interpreted languages are typically at a disadvantage performance-wise, however, there are a variety of Python packages specifically designed to boost performance (see below for details). In addition, with nearly limitless computational power and processing time within easy reach (e.g., Amazon Web Services, Microsoft Azure), the speed at which specific programming languages run is not the issue it once was.

The agent model is a relatively straightforward implementation of the mathematical expressions described above. The interactions that our agents engaged in differed across the three simulations reported in the article. In all three simulations, agents engaged in group interactions with each agent in the group repeatedly taking turns throughout the interaction. In

DOI: 10.1177/0956797615605798

Study 1, the group interactions involved groups of three agents. In Study 2, the group interactions involved groups of variable size. Despite the large number of agents involved in the simulations themselves, interactions were exclusively dyadic (involving groups of two agents). In Study 3, we formalized the agents' environment by specifying which agents were permitted to interact by describing each agent's location within a larger social network. Agents were only permitted to interact with agents they were directly connected to in the network. These networks were implemented using the NetworkX package (Hagberg, Schult & Swart, 2008). This package provides tools for constructing, representing, and analyzing networks. The networks, often referred to as "graphs" within fields such as computer science, each describe a set of nodes connected by a set of edges (i.e., links). NetworkX includes a wide array of tools for network analysis including standard measures of centrality, clustering, distance, and degree distributions among others. NetworkX is also incredibly flexible. It is capable of handling both directed networks (i.e., those with edges that "point" from node A to node B, but not necessarily from B to A) and undirected networks (i.e., those in which edges simply represent a non-directional connection). Edges themselves can either be unweighted (e.g., representing adjacency) or weighted (e.g., representing a communication channel between two individuals) and nodes can be connected with parallel edges (e.g., semantic associations, temporal associations, etc.). One of the most useful aspects of NetworkX is the fact that both nodes and edges can be associated with arbitrary, user-defined data. For example, nodes could be numbers (e.g., agent ID numbers) or strings (e.g., the names of individuals). Edges could be associated with categories to allow for multiplex networks or networks with different categories of edges (e.g., work relationships vs. social relationships vs. family relationships). If the network is evolving over time, edges could be associated with information about when the relevant interaction occurred.

DOI: 10.1177/0956797615605798

In the present study, NetworkX was useful for two particular reasons. First, NetworkX has a variety of tools for constructing networks. For example, there are methods for generating random graphs such as the “small world” networks used in Study 3 (i.e., using the `connected_watts_strogatz_graph()` function). NetworkX also has the ability to instantiate several classic networks including Zachary’s karate club network (using the `karate_club_graph()` function). The details of the co-authorship network were downloaded (from the UCI Network Data Repository, see below for more detail) in GML format (Graph Modeling Language), a file format for describing network structures. NetworkX has the ability to read (and write) a variety of file formats, including GML (using the `read_gml()` function). Because the co-authorship contains several separate, unconnected subgraphs (what are known as components), the largest component was selected from the entire network by using the longest entry returned by the `connected_component_subgraphs()` function.

The collaborative groups simulated in Studies 1 and 2 were also implemented as interactions over network structures, despite the fact that the interactions in these studies do not resemble what readers might think of as networks. The networks that describe these group interactions are what are referred to as complete or fully connected networks (in which every node is connected to every other node) and are easily constructed using the `complete_graph()` function in NetworkX.

Interestingly, NetworkX was also used in the current study to implement the association matrix, *S*, described above. Recall that *S* represents the pre-experimental associations that exist between items. These associations were represented as a complete network (each item had some association with every other item), edges were directional (i.e., the degree to which retrieving item A influenced the activation of B was not necessarily the same as the opposite situation), and the edges were weighted (with the weights representing the strength of the association. By implementing the associations as a network, associations can easily be chained together simply by traversing the network edge by edge (e.g., as in spreading activation).

The last element of NetworkX that we wish to note is its ability to visualize network structures. The networks depicted in Figure 4 of the article were created using NetworkX, although these illustrate only a small portion of NetworkX's capabilities. Nodes and edges can be plotted separately and the visual attributes (e.g., color, shape, size, transparency) of individual nodes/edges can be controlled. Nodes and edges can also be labeled and the visual attributes of the labels can be similarly customized. One of the most useful aspects of NetworkX's drawing capabilities is the ability to construct a variety of different layouts. A layout specifies where the nodes in the network are plotted relative to one another. For simple networks, there may be some obvious, natural layout. For example, for a complete, three-node network (i.e., like those used in Study 1 of the current article), the network is naturally plotted as an equilateral triangle. However, as the number of nodes grows, determining how to place nodes so as to ease visual comprehension becomes increasingly difficult. NetworkX thus provides a variety of layout options. These include random layouts (in which nodes are randomly arranged), circular layouts (in which the nodes are arranged along a circle), and, perhaps most useful, spring layouts (among others). A spring layout arranges the nodes by placing them in initial positions (which are customizable) and then moving them as if a) nodes exert repulsive forces on one another and b) edges act as springs. The algorithm then iteratively evaluates the forces acting upon each nodes, updates each node's new position (the number of iterations is customizable). The result tends to be a layout that makes certain features of the network quite salient. For example, heavily interconnected portions of the network will tend to be arranged as densely packed subsets of nodes (as can be seen in Figure 3 in the current article). The networks in the in Figure 3 of the current article were constructed by first constructing a random layout and using this layout as the initial positions in the subsequent construction of a spring layout. This combination seems to provide reasonable layout, with nodes arranged to easily visualize their place in the larger network, but with the network itself occupying a reasonably compact, regular (e.g., circular) space.

ABM Literature

Here, we provide a brief overview of literature on ABM. This section should be taken as a starting point for curious readers unfamiliar with ABM rather than as an exhaustive review. The general topic has been covered in substantially greater detail and depth elsewhere and points to such resources are included. Before diving into the literature itself, it is important to note two related points. First, the terminology used in discussing ABM is diverse. Some, as in the current article, refer to agent-based modeling (ABM), but others instead prefer terms such as agent-based modeling and simulation (ABMS), or multi-agent systems (MAS). There are also disciplinary differences in terminology. For example, the term agent-based computational economics (ACE) allows researchers to distinguish such approaches from more traditional approaches within the field of economics. Second, how ABM is employed varies across disciplines. ABM is an approach that makes it inherently attractive to researchers in a variety of disciplines. However, each of these disciplines brings a different perspective on what value ABM provides, what answer ABM is best suited to provide, and the how ABM (and/or computational modeling more generally) fits into the discipline. For this reason, there is wide variability in how ABM studies are reported across disciplines. As just one obvious example, ABM reported in computer science might include little discussion about ABM as an approach, whereas similar articles in anthropology might include greater justification of their methods.

We would point readers unfamiliar with ABM to the variety of general texts available. These would include Epstein and Axtell (1996), Gilbert and Troitzsch (2005), and Shoham and Leyton-Brown (2008). For a more compact overview, we might suggest Macal and North (2006) or Bonabeau (2002). For more of a conceptual overview of how ABM is of use within psychology, readers might consider Goldstone and Janssen (2005) for a somewhat more cognitive psychological perspective, Smith and Collins (2009) for a somewhat more social psychological perspective, or Richie, Yang, & Coppola, (2014) for an application in the context of language. If readers are instead looking for further examples of ABM as it has been applied to specific research questions, we would strongly suggest Schelling (1971), Axelrod (1997), and Carley (2002).

We would also like to make special note of Tesfatsion & Judd (2006). This is a large, edited volume that contains a variety of perspectives on how ABM can be applied, largely in the field of economics. The chapters include both descriptions of various applications, but also historical perspectives on how the field has evolved. One of the most useful parts of this volume is the appendix, entitled “A guide for newcomers to agent-based modeling in the social sciences” (Axelrod & Tesfatsion, 2006). This guide includes discussion of the “whys” and “hows” of ABM as well as its own reading list.

ABM-related Software

Python

DOI: 10.1177/0956797615605798

Python itself can be downloaded from python.org. However, the true power of Python ultimately lives in the thriving ecosystem of packages written by independent developers. These packages are typically distributed freely by the developers themselves and can be installed individually. However, it is likely more straightforward to instead use a more full-featured Python product, one that comes with both a base installation of Python as well as a large number of individual packages. One major advantage of these sorts of products is that the various interdependencies between packages will have been sorted out by the developers and will thus require little effort on the part of the user. Such products include Canopy, developed by Enthought (enthought.com) and Anaconda, developed by Contiuum (continuum.io). There are also similar products available from others (including Python(x,y)).

Regardless of how Python is acquired, there are specific packages that would be of particular interest to researchers engaged in ABM. These would include scipy (scientific tools, including statistical methods), numpy (numeric tools), NetworkX (the previously mentioned package providing a variety of network-related tools). There are also many packages that are more generally useful in scientific computing roles. These would include matplotlib and seaborn (both including plotting tools), pandas (spreadsheet-like data structures). IPython is also extremely useful in scientific contexts, allowing for interactive code to be written and the results (and code) to be saved for later distribution (e.g., to collaborators). As mentioned above, there are also tools that aim to increase the computational performance of Python. These would include Cython, Numba, and PyPy.

ABM-Specific

This section lists other software tools that are specifically intended to construct, run, and analyze agent-based models. These tools are potentially less flexible than a model implemented

DOI: 10.1177/0956797615605798

in a generic programming language (e.g., Python), but will ultimately be less intimidating for those with little programming experience.

- NetLogo: Probably one of the more common packages for implementing ABMs. The software allows models to be constructed and the interface provides the ability to visualize the model behavior and can provide access to model parameters so that they can be tweaked as the model is running.
- FLAME (Flexible Large-scale Agent Modeling Environment): Flame occupies a niche similar to Swarm (see below). Models consist of components (agents, agent interactions, environments, etc.), each of which can be constructed as the research wishes. Models are specified in XML (and thus do not require "programming" in the traditional sense) and thus should be more accessible to those with less programming experience.
- Swarm: Developed by the Sante Fe Institute, Swarm represents a middle ground between a generic tool like Python and an ABM-specific tool such as NetLogo. Models consist of components (agents, objects, organizations, time, space, etc.), each of which can be specified as the researcher wishes. The software does require programming (C or Java), but comes with tutorials and example code to get new users started.
- Cellular Automaton Explorer: This is a convenient interface to build, explore, and visualize cellular automata (simple ABMs). Frequently used for teaching purposes, cellular automata are convenient for demonstrating the complex emergent properties that can emerge from simple, interacting components. There are several

implementations of this, including a popular one from Wolfram (demonstrations.wolfram.com/CellularAutomatonExplorer).

- OpenABM (openabm.org): OpenABM.org represents a clearinghouse for ABM models, news, and information. Of particular interest is the model library, which collects models, many of which are implementations of models described in previously published papers. Each entry in the library has downloadable versions of the model itself as well as information about its construction, authorship, and any associated publications. The library also provides the ability to cite the model directly. The library includes models implemented in a variety of languages and is thus a good place to start for those new to ABM.

Network-Related Resources

Network Data Sets

This list represents a small sample of repositories making network-related data freely available. The co-authorship network used in the current study was obtained from the first resource in the list.

- http://networkdata.ics.uci.edu/
- http://networkrepository.com/
- http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/ucidata.htm
- http://vlado.fmf.uni-lj.si/pub/networks/data/

Network Analysis and Visualization Software

DOI: 10.1177/0956797615605798

As with the network-related data resources listed above, the network software packaged listed below is by no means exhaustive. However, these represent some of the most popular packages and will thus allow for loading/analyzing/visualizing many of the network-related data listed above.

- UCINet (sites.google.com/site/ucinetsoftware/home)
- Graphviz (graphviz.org)
- Gephi (gephi.github.io)
- Pajek (mrvar.fdv.uni-lj.si/pajek)

References

Axelrod, R. (1984). *Evolution of Cooperation*. Basic Books, New York, New York, USA.

Axelrod, R. (1997). *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton, New Jersey: Princeton University Press.

Axelrod, R., & Testfatsion, L. (2006). A guide for newcomers to agent-based modeling in the social sciences. In Eds. Tesfatsion., L, Judd, K.L., *Handbook of Computational Economics,* Amsterdam, The Netherlands, Elsevier.

Bonabeau, E., 2002. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, *99*, 7280–7287.

Carley, K. M. (2002). Computational organization science: A new frontier. *Proceedings of the National Academy of Sciences*, *99*(suppl 3), 7257-7262.

Centola, D. (2011). An experimental study of homophily in the adoption of health behavior. *Science*, *334*, 1269–1272.

Centola, D. (2010). The spread of behavior in an online social network experiment. *Science*, *329*, 1194–1197.

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

Gilbert, N., & Troitzsch, K. (2005). *Simulation for the Social Scientist*. McGraw-Hill Education, Berkshire, UK.

Goldstone, R. L., & Janssen, M. A. (2005). Computational models of collective behavior. *Trends in Cognitive Sciences, 9*, 424-430.

Granovetter, M. (1978). Threshold models of collective behavior. *American Journal of Sociology, 83*, 1420-1443.

Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th Python in Science Conference (SciPy2008)* (11–15).

Korobow, A., Johnson, C., & Axtell, R. (2007). An agent–based model of tax compliance with social networks. *National Tax Journal*, *60*, 589-610.

Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, *4*(3), 151-162.

Muis, J. (2010). Simulating political stability and change in the Netherlands (1998-2002): An agent-based model of party competition with media effects empirically tested. *Journal of Artificial Societies and Social Simulation*, *13*, 4.

Meyers, L. A., Newman, M. E. J. Martin, M., & Schrag, S. (2003). Applying network theory to epidemics: Control measures for outbreaks of Mycoplasma pneumoniae, *Emerging Infectious Diseases*, *9*, 204–210.

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering, 9*, 10-20.

DOI: 10.1177/0956797615605798

Perez, F., & Granger, B. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, 9, 21–29.

Richie, R., Yang, C.. & Coppola, M.. (2014). Modeling the emergence of lexicons in homesign systems. *Topics in Cognitive Science*, *6*, 183-195.

Sawyer, R. K. (2003). Artificial societies: Multi-agent systems and the micro-macro link in sociological theory. *Sociological Methods and Research*, *31*, 325-363.

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

Shoham, Y., & Leyton-Brown, K. (2008). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge, UK.

Smith, E. R., & Collins, E. C. (2009). Contextualizing person perception: Distributed social cognition. *Psychological Review*, *116*, 343-364.

Tesfatsion, L., & K. L. Judd. (2006). *Handbook of Computational Economics: Agent-Based Computational Economics*, Vol. 2. Elsevier, Oxford, UK.