Article

Strategic Organization
1–25
© The Author(s) 2018
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1476127018815295
journals.sagepub.com/home/soq

Replicating agent-based models: Revisiting March's exploration–exploitation study

Sasanka Sekhar Chanda
Indian Institute of Management Indore, India

Kent D Miller
Michigan State University, USA

Abstract
To validate prior agent-based models, public disclosure of model code is necessary, but not sufficient. Conceptual model replication, involving independent reproduction of a model without referring to the originator's code, is crucial for detecting nonconformity between the publication text and the model implemented. We frame and evaluate four scenarios to expose the shortfalls of replication efforts that use the original study's program code and neglect the published model description. We identify, evaluate, and offer recommendations regarding disparities between the text and the program code that generated the results reported in March's classic study of exploration and exploitation in organizational learning. Based on our experience replicating his model, we suggest a two-step method of model verification—beginning with replicating the model from the published description, then turning to the program code of the original implemented model to account for divergent results. Exclusive reliance on either the published model description or the original program code does disservice to theory advancement, because disparities between the original conceptual and implemented models go unaddressed.

Keywords
agent-based modeling, conceptual model replication, exploration–exploitation model, model verification, organizational learning

Introduction

Computational modeling provides a laboratory for developing strategic organization theory. Through agent-based modeling, experiments that may be impossible to carry out in real life can be simulated (Burton, 2003). Compared to verbal theory development, computational modeling calls

Corresponding author:
Sasanka Sekhar Chanda, Indian Institute of Management Indore, C-101 Academic Block, Prabandh Shikhar, Rau-Pithampur Road, Indore 453556, Madhya Pradesh, India.
Email: sschanda@iimidr.ac.in

for more disciplined precision and logical consistency (Adner et al., 2009; Davis et al., 2007; Harrison et al., 2007; Vancouver and Weinhardt, 2012). Furthermore, agent-based modeling scores over analytical modeling in that there are fewer unappealing assumptions driven by the need for mathematical tractability (Muldoon, 2007). As a theory-building exercise, agent-based modeling complements empirical theory-testing research (Miller, 2015). These advantages make a case for agent-based modeling in strategic organization research.

However, beneath the surface, there are challenges regarding how models get implemented that are rarely discussed. Errors can arise in describing a model and programming it. Reviewers and editors rely upon author-provided descriptions of models, but rarely verify conformity between programs and model descriptions. Contrary to the claims regarding potential precision and consistency of computational studies, disparities between published model descriptions and implemented models can go undetected. Given that the program code normally is not reviewed by outsiders, discrepancies from the theoretical framing in the text of an article can remain through publication. These shortcomings can persist in follow-on research.

The main issue in moving between model descriptions and computer programs concerns the occurrence of one-to-many cardinality in translating natural language process descriptions into precise formal code. Model descriptions—even when written carefully—may admit a variety of potential operationalizations (Maguire et al., 2006). Any sample of program code has multiple branches (often signaled by language such as IF … ELSEIF … ELSE …). The presence of just 10 nested branches with binary options at each node translates into over a thousand possible alternatives. Model summaries, comprising natural language process descriptions, often fall short of detailing the exact logic used at each of these branches. The sequence for nesting branches may be ambiguous, yet the order in which the branches arise may affect model outcomes (Wilensky and Rand, 2007).

In short, the findings reported in a study applying agent-based modeling are the product of a set of specific programming choices, from potentially a vast number of alternatives that qualify based on the natural language description of the model in the text of the publication. The model description undergoes rigorous review along the path to journal publication; the same cannot be said about the program code, which generally does not undergo external review. Even if a publishing outlet sought and reviewed the model program, there may not be a guarantee that the particular computer instructions implementing the natural language model are the most appropriate. The submitted code would orient the reviewers' minds regarding potential ways to implement the natural language instructions in program code, thereby shutting out consideration of other reasonable interpretations (Wilensky and Rand, 2007). Outside reviewers would be hard-pressed to suggest improvements because they lack firsthand experience working through potential operationalizations *for that particular model* and discovering the implications of alternative approaches. Furthermore, reviewers lack incentives to take on the burden of this additional work.

Recognizing the need for replication to verify prior models, some have called for making computer programs available to other researchers (e.g. Donoho, 2010; Ince et al., 2012). Following this recommendation would enhance accountability and facilitate replication, but does it go far enough toward advancing theory development through replication? Wilensky and Rand (2007) make an intriguing suggestion that researchers should attempt to recreate independently a pioneer's model *without looking at the original program code upfront*, even if the code is available. They argue that variances between the logic in the published model description (i.e. the conceptual model) and that implemented in the program code that generated the outputs can be unearthed when a researcher attempts to write fresh program code, unbiased by knowledge of the original program. Wilensky and Rand's observation raises an interesting question: *what can researchers learn from an attempt to replicate an agent-based model without looking at the pioneer's program that cannot be known by merely checking conformity of the logic of the pioneer's program with the published model description?*

To gain some practical insight, we recount our own experience attempting an exact replication of March's (1991) classic study of exploration and exploitation in organizational learning. Despite the prominence of March's article, no prior published study has investigated all of the details required to verify the model. We initially sought to replicate March's model by writing computer code with the published paper as the sole source. When our results failed to match those in the original study, we requested the program code, which Professor March kindly made available. Despite having the code, it took us a while to obtain convergence. The process of arriving there was illuminating. Over time it occurred to us that what we learned would be worthwhile to share with a larger audience. Hence, this study identifies, evaluates, and offers recommendations regarding disparities between March's model description (conceptual model) and the program code (implemented model) that generated his reported results. Our experience not only clarifies the rationale for pursuing replication studies in agent-based modeling but it also motivates a method for doing so. This method serves the interests of agent-based modelers in carefully aligning theoretical arguments and formal models and provides added assurance to those who use their theory and results in subsequent research.

Replication and verification: four scenarios

Agent-based models—that simulate interactions among agents representing social actors (Fioretti, 2013)—make up a distinct subset of computational research. Modeled agents engage in activities characterized by autonomy and interdependence, typically have some adaptive learning capability, and pursue goals related to desired states (such as fitness). Some examples are *NK* (Kauffman and Weinberger, 1989; Levinthal, 1997), genetic algorithm (Holland, 1975; March, 1991), multi-arm bandit (Posen and Levinthal, 2012), and simulated annealing (Carley and Svoboda, 1996) models. Whereas some kinds of computational models—such as Monte Carlo models and system dynamics models—can be specified with mathematical precision, agent-based modeling developed as a method for exploring the implications of assumptions that cannot be fully represented through equations or solved analytically. For many complex social phenomena, a mathematical model—in the sense of a formula—cannot be specified, yet an agent-based model can be fashioned and implications derived by means of computational simulation. However, relaxing the constraint of mathematical representation may introduce imprecision into authors' descriptions of agent-based models that potentially imperils subsequent interpretations and attempts to replicate models.

Verification certifies that the theoretical mechanisms posited by a study's authors account for the results (Galán et al., 2009; Midgley et al., 2007). Replication contributes to verification by confirming that an author's stated model and program code align with each other. Conceptual model replication does not, however, address whether the model aligns in every way with the broader theoretical arguments motivating it. For example, Bendor et al. (2001) critiqued the assumption built into the garbage can model (Cohen et al., 1972) that all decision makers follow the same heuristic to select decision areas to which they attend. Their critique follows from the inconsistency between the setting of organized anarchy, in which agents are expected to act in idiosyncratic ways, and the formal model as originally implemented in computer code. Such critiques leading to model reform are essential to progress in theory development but beyond the scope of our focus here on model replication. Likewise, we exclude from the ambit of conceptual replication all text and arguments—in the pioneer's publication—other than those pertaining directly to the specifications of the computational model.

Following the terminology given by Wilensky and Rand (2007), let us designate the model description in a pioneer's published work as the *conceptual model* (C) and the program code used by the pioneer to generate the output displayed in tables and graphs in the article as the *implemented model* (I). Assume that the narration in the published paper (C) has a total of m components, $c_1, c_2, \ldots, c_m$, that are distinguishable substantive features written in a natural language, such as English. Assume also that the original program code has a total of n components, $i_1, i_2, \ldots, i_n$. Typically, the program code would be written in a formal language, such as C/C++, Java, MATLAB, Perl, Python, and so forth. Furthermore, we designate the fresh computer code developed by a researcher attempting conceptual model replication as I^*.

In this section, we compare two approaches to model verification, *conceptual model replication* and *implemented model replication*. The first approach—which we represent as $C \rightarrow I^*$—concerns developing independent computer code (I^*) solely from the model description provided in the publication text (C). The second approach—labeled $I \rightarrow I'$—concerns working from the pioneer's program code (I) to develop a replicating model (I'). In either case, researchers compare the output from their replicating model with those in the publication to assess alignment with the original model. Our purpose is to determine the circumstances under which either of these approaches is better suited to detect divergence between a published model description and the associated computer code.

We describe four potential scenarios arising from the relation between a published model description (C) and the associated computer code (I), and their implications for model verification. Our four scenarios are logically exhaustive and plausible in practice. They are particularly likely to occur when the programmer developing the computer code is not the theorist who writes the model specification found in the publication. The theorist may not articulate all of the details necessary for developing the program logic in an unambiguous way. If not a proficient programmer, the theorist may be unable to follow all of the details of the computer code. Likewise, the programmer—for lack of familiarity with the domain of the theory—may omit, misinterpret, or misrepresent portions of the conceptual specification. Coupled with a division of labor between theorist and programmer, the iterative process of revising the model description and the computer program easily can produce misalignment in particular features between the two representations of the model. Even a single researcher who is both theorist and programmer may introduce disparities inadvertently when working iteratively between the conceptual model and the computer code.

Scenario 1: exact match

In the first scenario, every code component i_k ($k = 1, \ldots, n$) has a corresponding narration component c_j ($j = 1, \ldots, m$). Furthermore, there is no relevant narration component lacking a corresponding code component. In this scenario, we have the ideal case of one-to-one correspondence between formal language and natural language components.

Given that ambiguities and redundancies invariably arise in the narration of agent-based models, scenario 1 establishes a very high standard for reporting and implementing the model design. However, we include within this scenario a less-stringent form, namely, models for which readers can readily identify the computer code components that *need not* make it into the publication text (for instance, operation of count variables, error checking routines, and so forth). We also include cases where readers are able to classify natural language statements as redundant or irrelevant, for implementation purposes. For example, authors may repeat theoretical assumptions and arguments for reinforcement and include references to prior literature, yet these features are irrelevant to writing a suitable computer program. Whether a published study fulfills even these less restrictive assumptions is, however, a matter for empirical testing, rather than something that can be ascertained a priori. This scenario results in the following:

> Conclusion 1: Either conceptual model replication or implemented model replication serves equally well to verify a model when natural language specification in a published text conforms sufficiently to the formal language implementation.

Scenario 2: substitutions between the conceptual and implemented models

The description of certain model features (in C) may be at variance with the program code implemented (in I) in the original study. In this case, the conceptual model and the implemented model share the same number of features ($m = n$), but for one or more narration components in the published article, c_p (where $p \in \{1,\ldots,m\}$), there is a non-matching formalization, i_q ($q \in \{1,\ldots,n\}$), in the pioneer's code. In other words, there is a lack of meaning conformance between the natural language and formal language components. For example, conceptual and implemented models may include unaligned assumptions about the properties of agents or their relationships, or the environment in which they interact. Programs may include unintended violation of closed system (or open system) assumptions stated in the model description. There may be unintended interdependence among modeled agents where none ought to exist or unintended independence among agents that ought to be interdependent, and so forth.

Conceptual model replication ($C \rightarrow I^*$) will produce outcomes consistent with the theoretical assumptions in the publication text. When the underlying logic in the implemented model is different from that in the conceptual model, outputs obtained by a follow-on researcher performing conceptual model replication will be at odds with the results reported in the original publication. Thus, problem detection is feasible because the output from the researcher's code (I^*) will not match with the original results. However, in the absence of the original code, it will not be possible to identify the specific source(s) of the disparity, thwarting problem rectification.

In contrast, replicating the implemented model ($I \rightarrow I'$) will not uncover disparities between the original conceptual (C) and implemented (I) models. For example, if the text of the publication is ambiguous—that is, admitting more than one interpretation—the researcher will ratify the meaning suggested by the code in the implemented model, never realizing that those not seeing the code may derive an entirely different meaning from the theory discussed in the publication.

> Conclusion 2: Lack of conformance between one or more natural language and formal language components can be detected—but not corrected—by attempting a conceptual replication. Rectifying nonconformity can take place when researchers subsequently refer to the original program.

Scenario 3: omissions from the conceptual model

In the third scenario, the text of the conceptual model description (C) omits a subset of features present in the model code (I). This situation can arise readily as a programmer introduces features without reporting them to others on the research team. Alternatively, when the programmer reports minute details, other researchers may fail to grasp their implications for theory and refrain from including the details in the publication text. Furthermore, limits on article length force concision that can lead to omissions from the conceptual model description.

Stated formally, for one or more feature implemented in the pioneer's code, i_q, there exist no corresponding narration component, c_p (where $q \in \{1,\ldots,n\}$ and $p \in \{1,\ldots,m\}$). Common omissions include under-specification of the explanatory mechanisms in a model and missing explanations regarding the sequence of process steps. For example, a conceptual model description might omit details about the order of agents' actions. In such a case, the mechanisms for updating values of intermediate or final outcome variables (e.g. fixed, random, or endogenously determined order) become unclear or idiosyncratic from the point of view of a subsequent researcher.

Conceptual model replication admits nothing that is not explicit in the published model description. If a code component that lacks a counterpart in the conceptual model plays a substantive role in fashioning the reported results, conceptual model replication will promptly surface a problem, because outputs from the newly developed code (I^*) will not match the pioneer's output displayed in the publication text. As in scenario 2, detecting the sources of failure to verify based on conceptual model replication requires taking the further step of accessing the original program and carefully comparing its features to those of the conceptual model. In contrast, working from the original program ($I \rightarrow I'$) will, quite likely, ratify the outcomes from the original study but leave gaps in the original conceptual model undetected. Hence, we have the following:

> Conclusion 3: Conceptual model replication generally fails to verify its predecessor if the conceptual model suffers from omissions. Only when outcomes are robust to component omissions or substitutions, will conceptual model replication verify a model. Rectifying nonconformity can occur once researchers refer to the original program.

Scenario 4: omissions from the implemented model

Certain features described in the conceptual model (C) may be absent in the implemented model (I). Here, for one or more feature in the article's description of the model, c_p, there exists no corresponding program component, i_q (where $p \in \{1, \ldots, m\}$ and $q \in \{1, \ldots, n\}$). Examples arise in any argument invoked by the original author that does not have a corresponding representation in the implemented model. For example, the original study may have failed to include analyses incorporating alternative values of a moderator identified in its theoretical arguments. Such a simplifying assumption may be reasonable, but if it went unstated and unjustified in the original article, then the implemented model suffers an omission.

As with the previous two scenarios, replication based on the previously implemented model ($I \rightarrow I'$) presents no opportunity to detect omissions from the implemented model. Anchoring on the program code may lead researchers to dismiss components in the theoretical arguments as background that fails to reference explanatory mechanisms needed in the model (i.e. noise that does not require implementation in computer code).

In contrast, conceptual model replication ($C \rightarrow I^*$) will, most likely, flag omissions from the implemented model. Even if there is only one omission and it is salient, conceptual model replication alerts researchers to variables missing from the prior analysis by producing results that diverge from those in the published study. Divergence in results may be qualitative, as well as quantitative, and will lead the replicating researcher to conclude that the outputs shown in the original publication are not the product of the assumptions stated in the conceptual model description. This suggests the following:

> Conclusion 4: Conceptual model replication detects omissions of important theoretical assumptions, whereas replicating the implemented model does not. Identifying specific omissions requires careful work between the model description and the original program code.

Conclusions from scenarios

This section framed four scenarios based on a typology of the possible logical relations between a prior study's conceptual and implemented models. In only one of these scenarios, would researchers undertaking replications based on the conceptual and implemented models reach the same conclusion. In scenario 1, where there is sufficient conformity between the conceptual and implemented models, starting from either source would verify the original model. In each of the other three scenarios, in which there is some disparity between the conceptual and implemented models, our reasoning supports a preference for working from the conceptual model to detect inconsistencies and omissions. Conceptual model replication can alert researchers to disparities between the original conceptual and implemented models but does not pinpoint their sources. Lack of match with the original study's output acts as an overall cue for intensive search for discrepancies between the conceptual (C) and implemented (I) models, as well as errors in the new model program (I^*).

These arguments provide the rationale for Wilensky and Rand's (2007) suggestion to replicate models without referring to the original code. Taking this approach, researchers setting out to replicate an agent-based model have to perform independently a series of mental and computational "what-if" exercises leading to identification of a logically compelling and robust implementation. The task of finding a "good" implementation from myriad candidates is the essence of theory-building through agent-based modeling (Miller, 2015). The graphical output from the new computer program can be checked for conformance with the graphs presented in the original publication. Divergences stimulate further review of the publication text. Researchers performing the replication eventually reach either convergence or a confident judgment that there exist disparities between the conceptual and implemented models of the original study. Convergent findings from independently constructed programs based on a common model description are reassuring, and disparate findings are telling. Alternatively, if subsequent researchers work from the pioneer's code, they may simply ratify properties idiosyncratic to the initial computer program (I), rather than test the implications of other alternatives consistent with the model description in the original article.[1] Replications based solely on the original program code can lead researchers to fallacious conclusions about model verification.

Replicating March's (1991) exploration–exploitation model

Overview of March's model

We chose March's (1991) exploration–exploitation model as our illustrative case for two primary reasons. First, his model is simple—but not too much so. The textual model description in March's article is quite brief. March was interested in qualitative outcomes (i.e. proof of concept), rather than drawing inferences about quantitative outcomes in some empirical setting. As such, he favored parsimony and did not pursue calibration to support external validity. Despite its essential simplicity, March's agent-based model provides a rich platform for studying the emergence of macro-level patterns from stylized conceptualizations of micro-level dynamics. In this respect, it contrasts with the bandit model, which is a single-agent model reflecting an atomistic entity. Another possible candidate for our replication effort, the *NK* model, is versatile and open to use in multiple theoretical domains; however, it is often deployed to represent an organization as a single agent rather than as a collection of many agents.

Second, March's (1991) model merits replication based on its prominence. Web of Science reports over 6500 citations of this article (accessed 12 September 2018). Citing articles include studies of organizational learning, knowledge management, ambidexterity, and innovation (Wilden et al., 2018). A stream of modeling work followed this pioneering study as well. This includes prior, although limited, efforts at replication. Despite the prominence of March's study, no published study has attained an exact replication of his model. Modeling research has broadly supported March's key qualitative findings but has done so without achieving quantitative agreement.

March's model of organizational learning features an external reality consisting of m dimensions, each having a value of 1 or -1 assigned at random (i.e. with equal probability) at the outset of a model run. There also is an organization composed of n individuals. Each organizational member holds m beliefs of 1, 0, or -1, with initial beliefs assigned at random (i.e. with equal probability). Truthful knowledge occurs when beliefs match reality. Beliefs of 1 or -1 may or may not align with the values for the corresponding dimensions of the external reality; beliefs of 0, indicating neutrality, never match the external reality.

March's (1991) model drew from work on genetic algorithms introduced by Holland (1975) to characterize organizational learning as a recombinant social process. March's (1991) model is "a specific form of a genetic algorithm in the sense that individuals do not recombine attributes of other individuals directly, but rather do so indirectly, mediated by the 'organizational code'" (Gavetti et al., 2012: 29). In addition to n members, the modeled organization includes one additional entity standing for the organization's memory. This construct, designated the *organizational code*, serves as a repository for the authorized knowledge in the organization. Initially, the organizational code has values of 0 for each of its m dimensions. It adapts to the beliefs of those members whose beliefs correspond to reality on more dimensions than does the code (i.e. those with superior knowledge). March implemented two concurrent learning mechanisms. One mechanism is learning by members from the organizational code, at a rate given by p_1 ($0 \leqslant p_1 \leqslant 1$). Members learn only from nonzero values (1 or -1) in the code. The second mechanism is the code's learning from the organizational members with superior knowledge at the rate p_2 ($0 < p_2 < 1$).[2]

March observed that if organizational members learn rapidly, differences in knowledge between them and the organizational code disappear quickly. In such a situation, members and the organizational code reach a suboptimal long-run knowledge level. Conversely, higher organizational knowledge is attained if organizational members learn slowly—and particularly if the organizational code updates rapidly to the superior beliefs in the organization. Although managers may be inclined toward exploiting current knowledge through rapid diffusion within the organization, the central finding from March's (1991) model emphasizes the long-run value of broad exploration through experimentation with many possible combinations of diverse beliefs.

In a variant of his model invoking member turnover, March (1991) illustrated the effect of infusing diverse beliefs from outside the organization. In any period the probability of any member leaving the organization is p_3 ($0 \leqslant p_3 \leqslant 1$). When individuals leave the organization, they are replaced by others from outside who bring m beliefs (1, 0, or -1) assigned at random. Turnover sets up two-way learning between newcomers and the organization. If rates of turnover are very low, newcomers will be socialized into the established beliefs. If the turnover rate is high, limited progress can be made in raising the level of knowledge through learning among the organization's members. Overall, a moderate rate of turnover supports cumulative organizational learning.

In a stable environment, it is quite likely that, over time, an organization will discover beliefs that fit the requirements of its context. Turbulent environments are characterized by changes in environmental demands that make aspects of an organization's knowledge obsolete. March (1991) allowed any given dimension of reality to shift between 1 and -1 with probability p_4 ($0 \leqslant p_4 \leqslant 1$) in each period. If the rate of environmental change is very high, organizational learning fails to keep up with environmental demands. Knowledge quickly becomes obsolete, leaving little time for productive exploitation by the organization. The point of interest, therefore, lies in assessing what it takes to balance exploitation and exploration efforts under moderate environmental change.

Replication effort and findings

We set out to recreate March's (1991) results by programming his model in MATLAB, taking the conceptual model described in his original publication as our sole guide. Our first attempt produced a reasonable degree of *qualitative* match with Figures 1 to 4 of March's article. However, we found a significant mismatch for one curve reported in his Figure 5.

We wrote to some of the researchers who had published work involving March's model—seeking to know how they may have circumvented the problems encountered—but not much useful information could be gleaned. In part, this was because quite a few of them had used significantly modified versions of March's model in their studies and, therefore, they did not attempt to perform exact replication. Finally, we wrote to Professor March himself, apprising him about our predicament and requesting his original program. Professor March kindly provided his original code, written in *Basic* programming language. We ran the programs in a *QuickBasic* environment and observed output almost identical to the figures in the original article. However, given our rudimentary familiarity with *Basic*, it was not possible to determine right away, what part of the logic was different between our replication attempt and March's original program. Therefore, we decided to embark on a process of translating March's programs from *Basic* to MATLAB. There were six distinct programs in all, one corresponding to each of the six published figures. The programs for the first five figures are interrelated, containing some overlaps in logic. The sixth program reflects an altogether different model and was out of scope for our study.

Hearteningly, our MATLAB translation of March's code produced a close match with the results that appear in the figures of the 1991 article. Our next step was to merge the five distinct MATLAB programs (that were direct translations of the five *Basic* programs from March) into one program. This program, when run with distinct parameter values, also produced March's graphical results. At the conclusion of this effort, we had two MATLAB programs, one from our initial replication attempted (I^*) from the conceptual model (C) and the other from translating March's implemented model (I) from *Basic* to MATLAB (producing an I'); yet, the results from I^* did not match those from I'! Nevertheless, the process of merging five distinct programs into one program brought to our attention certain similarities and differences in logic between March's code and the code from our original replication attempt. Eventually, we identified three places where the conceptual model described in the text of March's (1991) article differs from the code implementation underlying the published figures.

Subsequently, we revised our MATLAB program in such a way that we could turn on and turn off the undocumented features of the model. Two possible settings for each of these three features made possible experiments with eight distinct versions of the exploration–exploitation model. To limit the length of this article, we present a selected set of five model versions. We start from our conceptual model replication (I^*) which omits the features that went undocumented in March's (1991) article. Thereafter, we introduce independently each of the three undocumented features, thereby showing the effects of each relative to our conceptual model replication. Finally, we present results for the model (I') with all three undocumented features operating together.

For each model variant, panels a, b, c, d, and e correspond, respectively, to Figures 1 to 5 in March's (1991, pp. 76–80) original study.[3] Panel a plots the equilibrium level of knowledge attained when organizational members learn at the same rate. Panel b compares long-run equilibrium outcomes for organizations in which all members learn at the same rate vis-à-vis organizations in which members learn at heterogeneous rates. Heterogeneous learning rates are motivated by having varying proportions of slow ($p_1=0.1$) and fast ($p_1=0.9$) learners. Panel c shows knowledge at period 20 for varying proportions of slow ($p_1=0.1$) and fast ($p_1=0.9$) learners. The panel d analysis assumes a homogeneous member learning rate and introduces turnover of organizational members (at rate p_3). Panel e allows turbulence in the external reality (at rate p_4) along with turnover. Consistent with March's precedents, the time horizon is through period 250 in panels a and b, through period 20 in panels c and d, and through period 100 in panel e. Each of our figures presents averages over 10,000 runs.

Conceptual model replication. Figure 1 presents the results obtained in our initial replication effort following the conceptual description in March's published article. The trajectories of the curves in Figure 1(a) ratify March's conclusion that the highest equilibrium knowledge is attained when the

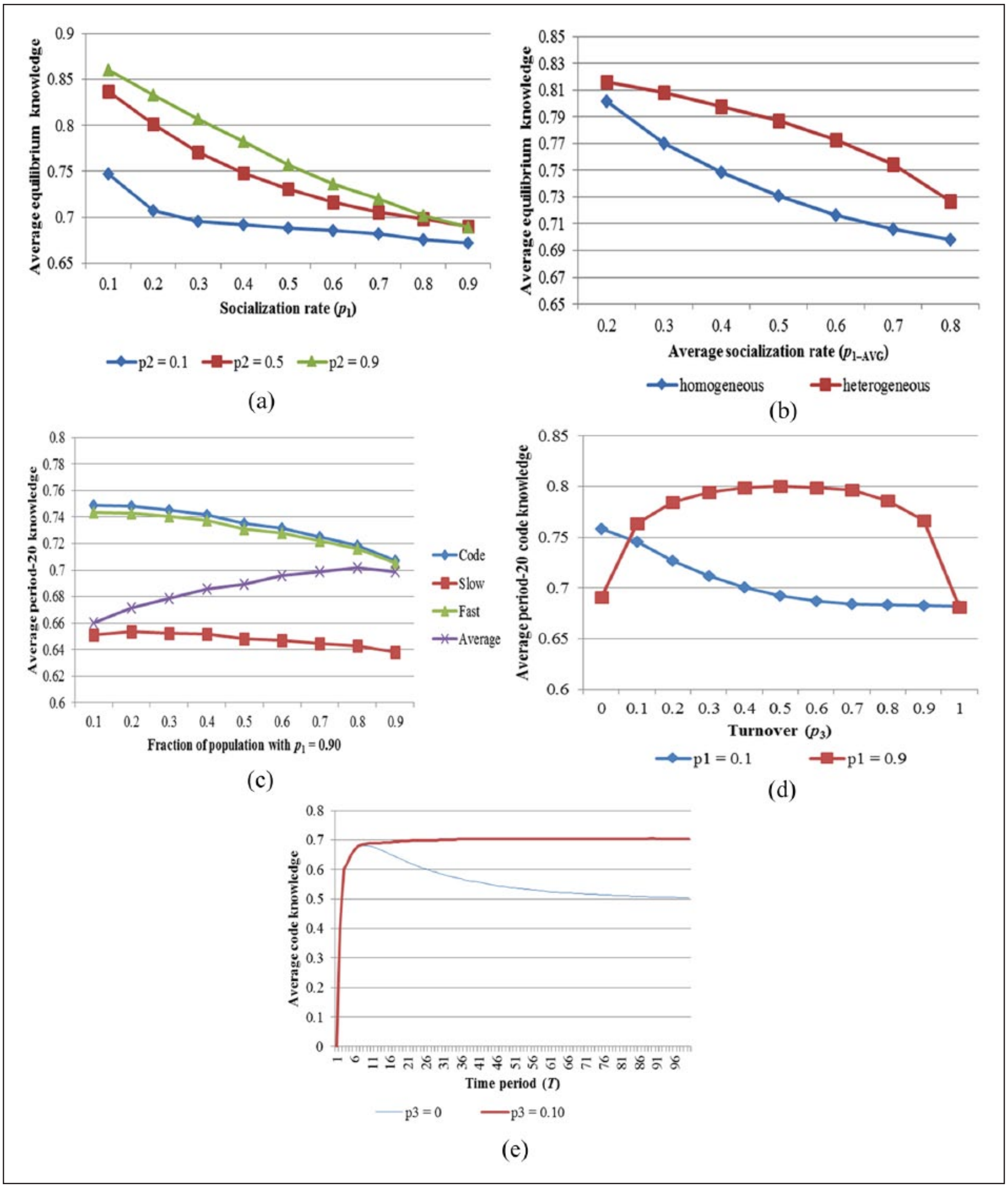

Figure 1. Conceptual model replication. (a) effect of learning rates (p_1, p_2) on equilibrium knowledge, (b) effect of heterogeneous socialization rates ($p_1 = 0.1, 0.9$) on equilibrium knowledge, (c) effect of heterogeneous socialization rates ($p_1 = 0.1, 0.9$) on period-20 knowledge, (d) effect of turnover (p_3) and socialization rate (p_1) on period-20 code knowledge, and (e) effect of turbulence (p_4) on code knowledge over time, with and without turnover (p_3).

code learns quickly and members learn slowly. The results in Figure 1(a) uphold a general negative relation better than March's own graphical output (Figure 1 on p. 76, replicated here in Figure 5(a)), which included some positively sloped segments in each of the lines. In Figure 1(a), all results for slow code learning ($p_2 = 0.1$) are lower than those for fast code learning ($p_2 = 0.5$ and $p_2 = 0.9$), exactly as March theorized, but his reported results did not demonstrate.

Figure 1(b) retains March's key result concerning the superiority of knowledge attained by heterogeneous learners over outcomes from homogeneous learners—albeit with slightly different quantitative outcomes. Likewise, Figure 1(c) and (d) shows qualitative similarities to March's (1991) results, but always with quantitative differences.

The most dramatic deviation from March's published output is in Figure 1(e). The curve for no turnover ($p_3 = 0$) does not fall to near zero (as reported by March); instead, it hovers just above 50%. By studying how I^* differs from I', we found that this difference occurs primarily because March's computer code calculates the dependent variable in the analyses—organizational knowledge—in a manner inconsistent with the description in the text. Our conceptual model replication followed March's (1991: 75) reported approach of computing "the proportion of reality that is represented" in individual beliefs or the organizational code. As such, our program used a counter to quantify matches between a belief string and the exogenous reality, and it ignored mismatches.

Net correct beliefs. March's program effectively ended up subtracting the number of incorrect nonzero beliefs from the number of matches. In March's code, program statements computed organizational or member knowledge as the product of two vectors: (1) either code beliefs or individual beliefs and (2) the external reality. This product reflects the extent to which the two arrays coincide, with mismatched nonzero values scored negatively. In computing this product, incorrect nonzero beliefs cancel out an equivalent number of correct beliefs. Hence, the proportion of *net correct beliefs* appears in the graphical output shown in March's paper (Figures 1 to 5: 76–80).

It is instructive to note that, in their replication of March (1991), Blaschke and Schoeneborn (2006) reported results similar to those in our Figure 1(a) and (e) (see their Figure 1, p. 104 and Figure 2, p. 105). Similar to us, they computed organizational knowledge as *number of correct beliefs*. They acknowledge having access to March's code. However, they do not report whether they found the discrepancy—in the formula used to calculate organizational knowledge—between March's code and publication text. In contrast, Rodan (2005) derived his definition of the dependent variable from March's code and calculated organizational knowledge as *net correct beliefs*. However, he too does not report having noticed that March's article suggests a formula for calculating organizational knowledge that is different from the formula implemented in the program code.

Shifting from a count of correct beliefs to net correct beliefs as the outcome of interest, we obtained the results presented in Figure 2. In Figure 2(a), we find that the curve for $p_2 = 0.1$ rises with the rate of learning by organizational members for $p_1 \geqslant 0.2$. This is problematic, since it deviates from the consistent negative effect found for $p_2 = 0.5$ and $p_2 = 0.9$. The other graphs in Figure 2 share many qualitative similarities to the corresponding graphs in Figures 1, but differ in their details. A comparison of the lines corresponding to $p_3 = 0.10$ in Figures 1(e) and 2(e) is instructive. In Figure 1(e), this line attains a value of approximately 0.7, whereas in Figure 2(e), which plots net correct beliefs, the long-run value is near 0.4, which is simply the difference between the proportion of correct beliefs and incorrect beliefs (0.7 – 0.3). Likewise, the plot for $p_3 = 0$ in Figure 1(e) indicates that the proportion of correct beliefs nears 0.5 in the long run, whereas net correct beliefs drops to near zero in Figure 2(e).

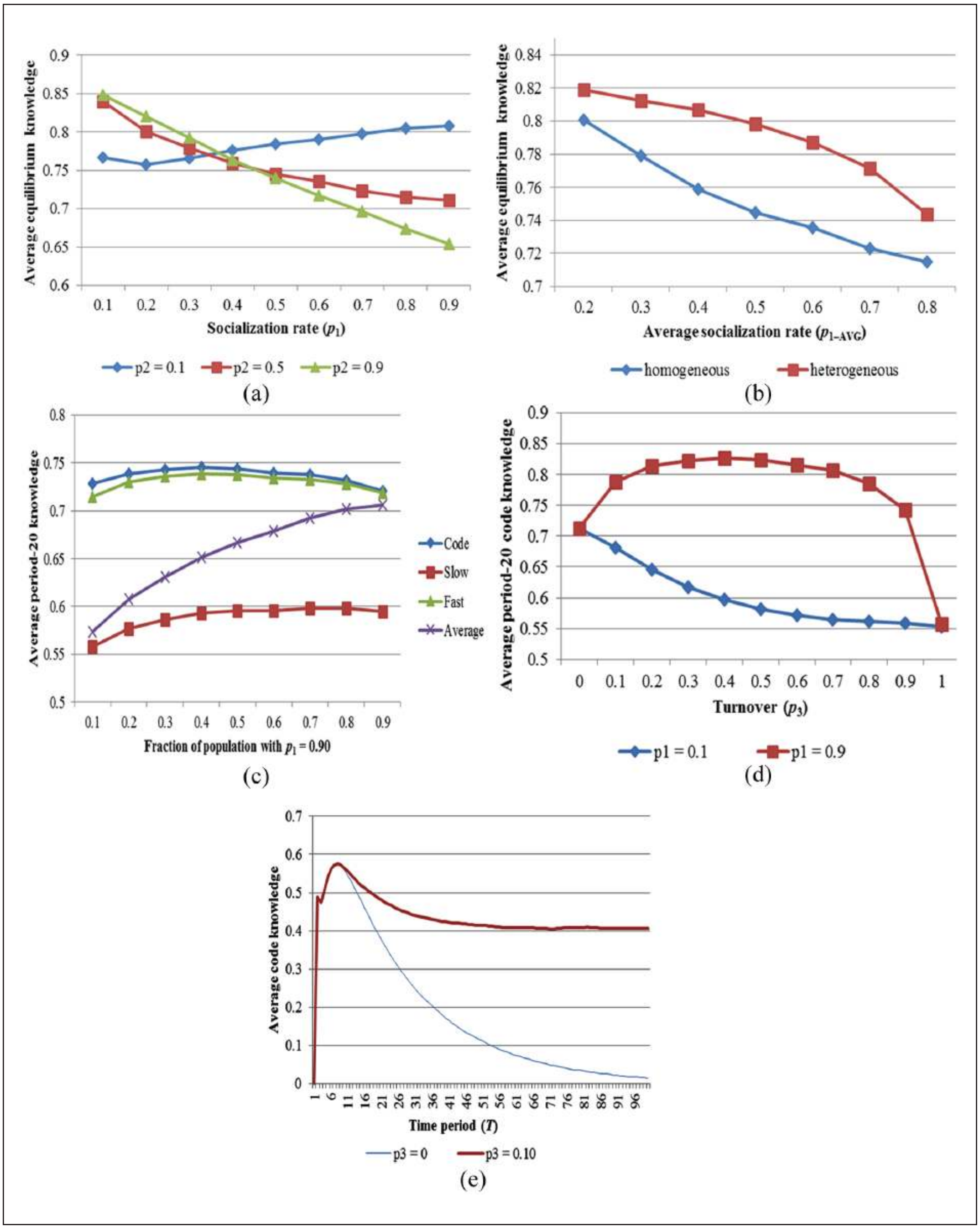

Figure 2. Net correct beliefs. (a) effect of learning rates (p_1, p_2) on equilibrium knowledge, (b) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on equilibrium knowledge, (c) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on period-20 knowledge, (d) effect of turnover (p_3) and socialization rate (p_1) on period-20 code knowledge, and (e) effect of turbulence (p_4) on code knowledge over time, with and without turnover (p_3).

Random classification of zero beliefs. In March's model, organizational members have belief-strings including elements with values −1, 0, and 1. When the organizational code identifies knowledgeable members to learn from, the implemented program (*I*) randomly assigns 1 or −1 values to their zero beliefs. These randomly assigned beliefs, in turn, contribute to the recommendation from the most knowledgeable members reflected in the organizational code. In our original replication attempt, we did not give any weight to the "neutral beliefs" (March, 1991: 75) because this process is not documented in the published article.

Figure 3 reports the model results after adding random classification of zero beliefs to our initial conceptual model replication. Comparing Figure 3(a) with Figure 1(a), we observe that the curves for $p_2 = 0.5$ and $p_2 = 0.9$ have shifted up in the former, whereas the curve for $p_2 = 0.1$ has shifted down. Random assigning nonzero values (1 and −1) to neutral beliefs (0) injects heterogeneity into the beliefs from which the code learns. This adds exogenous variation into a model, somewhat compromising the portrayal of the organization as a *closed system*. This added variability in the beliefs from which the code learns slows the convergence of the code and members' beliefs to a uniform set of beliefs. Slowing the learning process when the code learns rapidly ($p_2 = 0.5$ or $p_2 = 0.9$) enhances learning in the long run. In contrast, when the code learns slowly ($p_2 = 0.1$), randomization of zero beliefs increases the likelihood that incorrect beliefs persist in the long run.

Figures 3(b) to (d) shows upward shifts in knowledge relative to the corresponding Figures 1(b) to (d). For each of these, the code learning rate of $p_2 = 0.5$ is sufficiently high that adding random variability into the beliefs from which the code learns slows convergence to long-run beliefs. Random classification of zero beliefs works together with modest turnover ($p_3 = 0.1$) to promote broader experimentation with possible combinations of beliefs, delay belief convergence, and arrive eventually at enhanced knowledge. In Figure 3(e), the organization with modest turnover achieves greater knowledge than in Figure 1(e).

Two-step updating. March's code performed two-step updating of members' knowledge as agents learn from the organizational code. Our reading of March's article suggested that when a belief of an individual is selected for updating, it acquires the value that is present in the corresponding position in the organizational code (if the latter is nonzero and different from the current value held by the member). For example, if position number 3 in a member's belief string has a value −1, and it is selected for updating in a given period, that position of the member's belief string will be updated with a value of 1 if the third position of the organizational code has a value of 1. In March's code—in a situation where a member's (nonzero) belief does not conform to the organizational code's (nonzero) belief—updating an organizational member's belief is a two-step process occurring over two distinct periods. During the first learning opportunity, the member's belief gets updated from −1 to 0 (when the value in the corresponding dimension in the organizational code is 1). On a subsequent learning opportunity, the member's belief converts from 0 to 1. In general, only zero values for members' beliefs can get overwritten by nonzero values from the organizational code.

Figure 4 shows the results after adding two-step updating of beliefs to our initial conceptual model replication. Comparing Figures 4(a) and 1(a), we find that changing the way that the model program updates member beliefs (by a two-step process in Figure 4 versus by a one-step process in Figure 1) generally affirms the negative relation between members' learning rate (p_1) and equilibrium knowledge. However, in Figure 4(a), the organization with slow learning by the code ($p_2 = 0.1$) achieves superior knowledge (relative to organizations with $p_2 = 0.5$ and 0.9) when members learn rapidly ($p_1 \geqslant 0.7$). This result does not have a theoretical motivation in the conceptual arguments. Furthermore, quantitative outcomes differ between Figures 4(a) and 1(a). Two-step updating lowers equilibrium knowledge when the code learns rapidly ($p_2 = 0.5$ or 0.9). When the

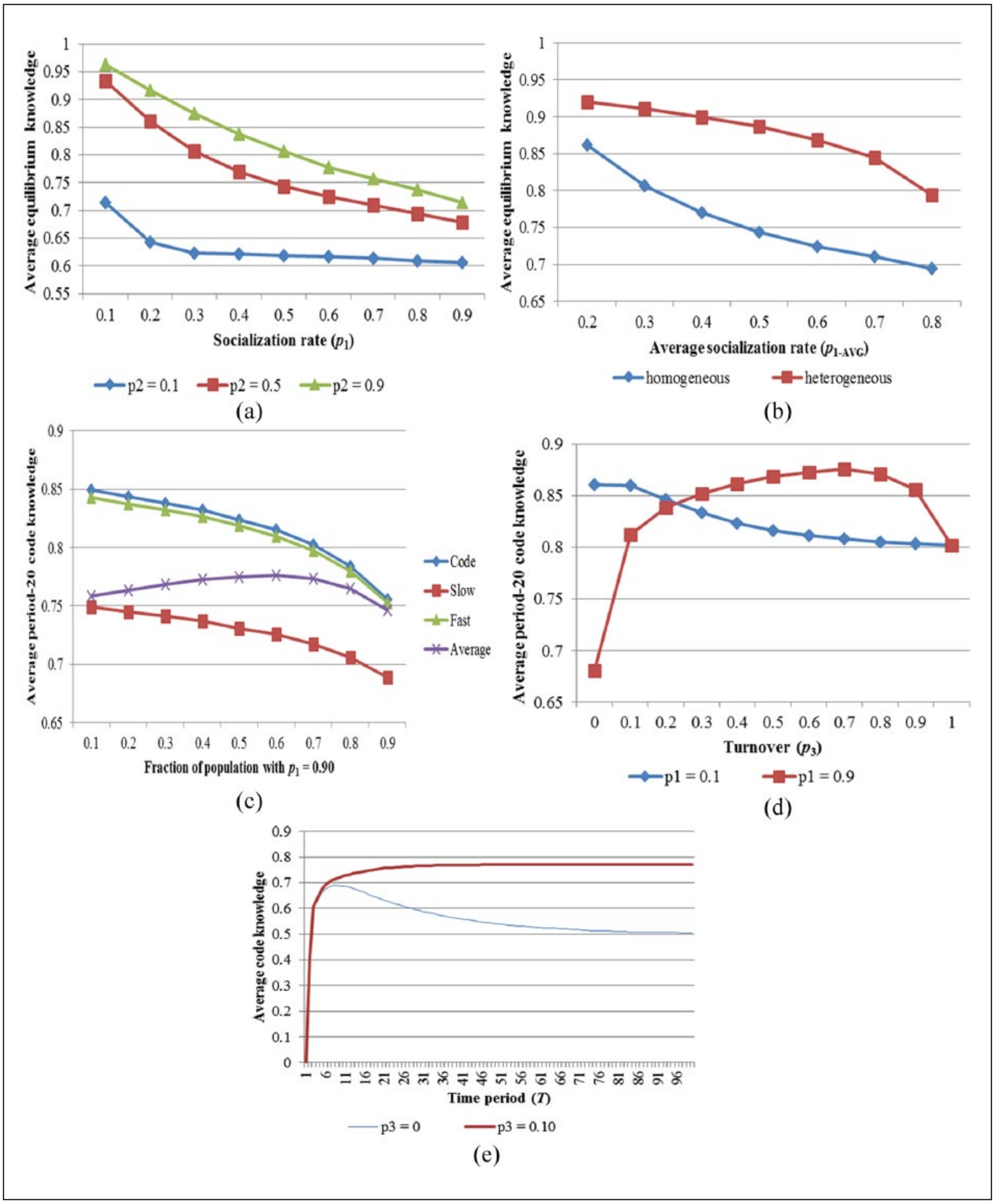

Figure 3. Random classification of zero beliefs. (a) effect of learning rates (p_1, p_2) on equilibrium knowledge, (b) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on equilibrium knowledge, (c) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on period-20 knowledge, (d) effect of turnover (p_3) and socialization rate (p_1) on period-20 code knowledge, and (e) effect of turbulence (p_4) on code knowledge over time, with and without turnover (p_3).

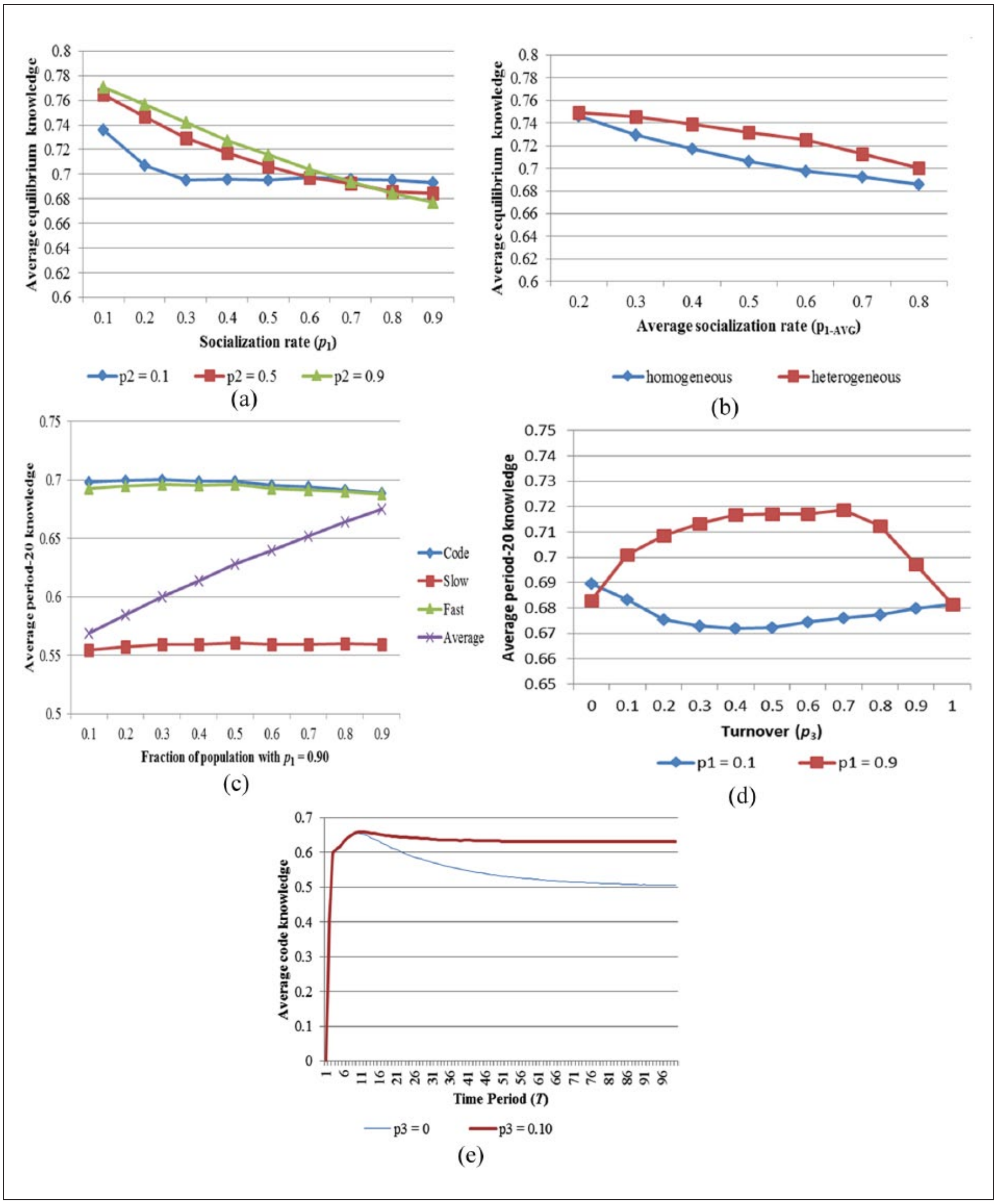

Figure 4. Two-step updating. (a) effect of learning rates (p_1, p_2) on equilibrium knowledge, (b) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on equilibrium knowledge, (c) effect of heterogeneous socialization rates ($p_1 = 0.1$, 0.9) on period-20 knowledge, (d) effect of turnover (p_3) and socialization rate (p_1) on period-20 code knowledge, and (e) effect of turbulence (p_4) on code knowledge over time, with and without turnover (p_3).

organizational code learns slowly ($p_2=0.1$), two-step updating restricts the range of knowledge outcomes relative to one-step updating.

Comparing Figure 4(b) with Figure 1(b), we observe that the gap between outcomes from heterogeneous and homogeneous learning rates has narrowed significantly. Two-step updating diminishes—but does not overturn—an important result undergirding the continuum conception of exploration and exploitation (Chanda et al., 2018), that "there might be some advantage to having a mix of fast and slow learners in an organization" (March, 1991: 76).

Comparisons between the other graphs in Figures 4 and 1 show a mix of implications associated with two-step updating of beliefs. In Figure 4(c), knowledge of fast learners and the code is relatively insensitive to the proportion of members who are fast learners. Fast learners' knowledge declines under two-step learning, but not as dramatically as it does for slow learners. The qualitative patterns in Figures 4(d) and 1(d) are similar, but quantitative outcomes are lower under two-step updating. Figure 4(e) shows longer time to equilibrium knowledge under two-step updating, as we would expect, as well as lower long-run knowledge in an organization with modest turnover ($p_3=0.1$).

Model combining three features. Our model combining all three features outside March's conceptual model produced the results found in Figure 5. We considered our efforts to account for disparities between March's conceptual and implemented models successful when, from the combined MATLAB program, we could reproduce March's quantitative results nearly exactly. Allowing for random processes within the model, which produce variation across runs, invoking all three features outside March's conceptual model generated results consistent with those in his published article. Averaging over a larger number of runs (10,000 versus 80) leads to graphs that are smoother, relative to those found in the original article.

Conclusions from replicating March's (1991) model

Replacing single-step updating of beliefs with a two-step learning process is an example of the earlier-described scenario 2—a substitution between the conceptual and implemented models. Another instance of scenario 2 pertains to computing average knowledge as the net correct beliefs instead of the proportion of correct beliefs. The other inconsistency between the published text of March's (1991) article and the details of his implemented model evident in his program code exemplifies scenario 3—omission from the conceptual model. Randomizing zero beliefs in code learning is a feature found exclusively in the implemented model; it has no counterpart in the article's model description.

When working solely from March's article, we experienced one instance of ambiguity where the text seemed to admit several possible interpretations. In March's model, in each period, the organizational code learns from the members with superior knowledge and the organizational members learn from the organizational code. It is not clear whether the code learns first (after which the members learn from the enhanced code), or the members learn first (after which the code learns from the updated member beliefs), or some other arrangement applies. During our early replication effort (when we did not have access to March's code), we vacillated between implementing code learning prior to member learning or vice versa. Eventually, and rather felicitously,[4] we settled for the option that March used: the organizational code learns from superior-knowledge members identified from the prior period. Likewise, individuals learn from the previous period's image of the organizational code. Had we not come to the judgment that this was a reasonable sequence, we easily might have introduced a fourth disparity between our interpretation of the conceptual model and March's implemented model. In other words, we could have introduced another occurrence of scenario 2 by substituting an idiosyncratic interpretation for an ambiguous

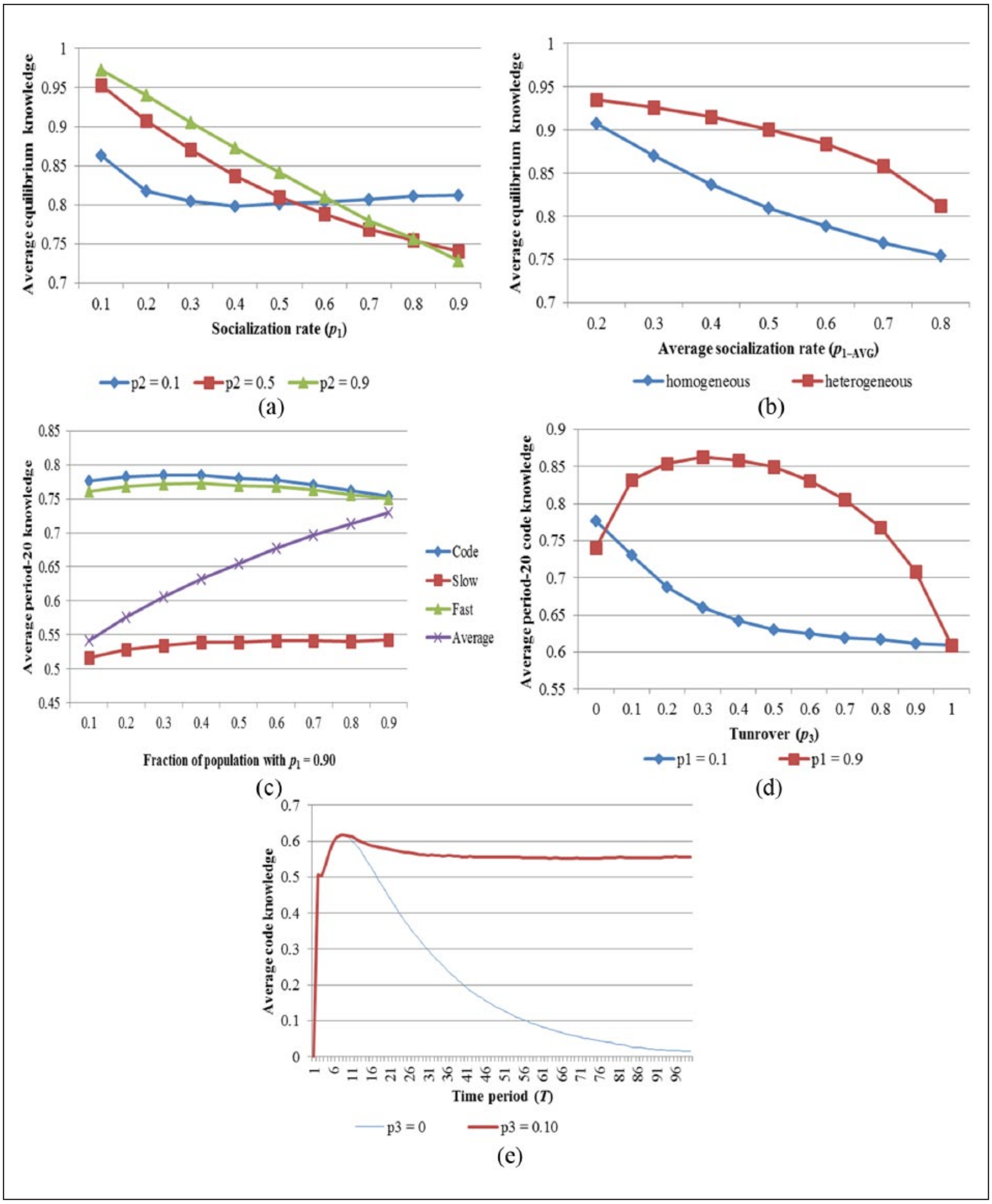

Figure 5. Model combining three features. (a) effect of learning rates (p_1, p_2) on equilibrium knowledge, (b) effect of heterogeneous socialization rates ($p_1 = 0.1, 0.9$) on equilibrium knowledge, (c) effect of heterogeneous socialization rates ($p_1 = 0.1, 0.9$) on period-20 knowledge, (d) effect of turnover (p_3) and socialization rate (p_1) on period-20 code knowledge, and (e) effect of turbulence (p_4) on code knowledge over time, with and without turnover (p_3).

feature in the conceptual model. This experience points out how conceptual model replication alerts researchers to ambiguities in the model description. If one has access to the original program code, it is unlikely that the mind will register that alternative code implementations of the same (ambiguous) text may be reasonable. We share the apprehension expressed by Dent (2012) that researchers may not pay careful attention to lack of congruence between the published text and the program code, if the latter is easily available and the focus of attention.

All three features uncovered by our replication effort are in conflict with the logic in March's conceptual model. If these features are dropped, all key conclusions from the original article remain valid. *Thus, going forward, we propose that the results presented in our Figure 1 should be considered the baseline for future replication and extension work based on March's model*, and we offer supporting arguments below.

Our responses to the particular disparities arising in March's work favor the conceptual model, yet we anticipate that substitutions and omissions in conceptual models will often lead to corrections derived from the implemented models. Efforts to reconcile conceptual and implemented models should attempt to identify and incorporate the best of both. Although we argue for starting from the conceptual model when performing replications, we accept that the discrepancies uncovered during a replication may be resolved in favor of the implemented model when warranted based on logic, prior theory, or empirical evidence.

Disregard zero beliefs in voting. March (1991: 74) states simply that the organizational code adjusts probabilistically "to conform to the dominant belief within the superior group." Substituting random beliefs for members' zero beliefs when determining the majority beliefs among superior-knowledge individuals has no counterpart in the conceptual model description. Unmotivated randomization introduced into the beliefs of individuals with superior knowledge contradicts March's characterization of the organization as a *closed system* and has a substantive effect on the learning process and outcome. Furthermore, randomization contradicts the stated nature of zeros as reflecting "neutral beliefs" or "no knowledge" (March, 1991: 75). Neutrality or ignorance should elicit voting behavior such as abstention or choosing "no opinion," which does not necessarily translate into random beliefs. Allowing zeroes to remain neutral in the voting process ratifies their unique status (relative to nonzero beliefs) in a way that is consistent with March's expressed theory.

Single-step updating of member beliefs. March (1991: 74) wrote, "In each period in which the code differs on any particular dimension from the belief of an individual, individual belief changes to that of the code with probability, p_1." This quote implies that individuals' beliefs can switch between nonzero values within a single period, rather than requiring two-step updating of agents' beliefs. Indifference or ignorance is not a necessary intermediate state, introducing a temporal lag, when people switch between beliefs. Hence, we recommend that replication and extension studies adopt single-step updating, which aligns with the conceptual model described in March's paper.

Knowledge as the proportion of correct beliefs. March (1991: 74) specified that elements of the environment were assigned *independently* and that organizational members' beliefs about elements of the environment were updated *independently* through the learning process. Nothing in his conceptual model description indicates interdependence among beliefs such that erroneous beliefs could have an offsetting effect on correct beliefs in determining knowledge. Unlike *NK* models following Levinthal (1997)—where, in an effort to reflect interactive complexity, interdependence among organizational features determines performance—we find an emphasis on *independence* of beliefs in the learning process in March's conceptual model. The penalty for incorrect nonzero beliefs, but not incorrect zero beliefs, in March's implemented model introduces unstated assumptions about costs and benefits into the computation of organizational knowledge.

Moreover, a comparison between Figures 1(a) and 2(a) shows that negative marking for incorrect beliefs is the primary cause of the upward slope of the curve associated with slow learning by the code ($p_2 = 0.10$). All three curves decrease monotonically in the agent learning rate (p_1) when the computation of organizational knowledge is the simple proportion of correct beliefs (Figure 1(a)).

Our suggestion to report the proportion of correct beliefs as the model outcome aligns with March's (1991: 75) statements that

> First, the proportion of reality that is correctly represented in the organizational code can be calculated for any period. This is the knowledge level of the code for that period. Second, the proportion of reality that is correctly represented in individual beliefs (on average) can be calculated for any period. This is the average knowledge level of the individuals for that period.

Prior exploration–exploitation model studies

Following the line of reasoning associated with our conclusions regarding scenarios 2 and 3, subsequent conceptual model replication could have surfaced concerns about results inconsistent with those reported originally and, hence, failure to verify the model. Did this occur? We reviewed a set of ten prior computational studies based on March's (1991) model. We examined all ten studies to find out how they handled the three discrepancies between the conceptual and implemented models. Table 1 provides a summary of our findings. Its columns report (1) whether each study randomized agents' zero beliefs for the purpose of code learning or treated zero beliefs as neutral, (2) whether agents' learning of the alternative nonzero belief occurs in one or two steps, and (3) whether knowledge was measured in terms of the net correct beliefs or correct beliefs as a proportion of total beliefs. Seven of the studies—Kane and Alavi (2007), Kim and Rhee (2009), Fang et al. (2010), Schilling and Fang (2014), Chanda and Ray (2015a), Chanda (2017), and Chanda et al. (2018)—make assumptions that are in alignment with March's (1991) conceptual model description. In other words, their conceptual description suggests that agents' zero beliefs are ignored rather than randomized, updating members' beliefs occur in one step, and they compute organizational knowledge as proportion of correct beliefs.

Table 1. Handling of undocumented features in modeling studies building on March (1991).

Study	Zero beliefs in code learning	Updating members' beliefs	Knowledge measure
Rodan (2005)[a]	Randomize (new construct, "self-restrained experimentation")	Two-step	Net correct
Blaschke and Schoeneborn (2006)[a]	Neutral	Two-step (new construct, "forgetting by an individual")	Correct
Miller et al. (2006)	Neutral	One-step	Net correct
Kane and Alavi (2007)	Neutral	One-step	Correct
Kim and Rhee (2009)	Neutral	One-step	Correct
Fang et al. (2010)	Neutral	One-step	Correct
Schilling and Fang (2014)	Neutral	One-step	Correct
Chanda and Ray (2015a)[a]	Neutral	One-step	Correct
Chanda (2017)[a]	Neutral	One-step	Correct
Chanda et al. (2018)[a]	Neutral	One-step	Correct

[a]Article acknowledges access to March's program.

Out of the remaining three, Miller et al. (2006) and Blaschke and Schoeneborn (2006) each adopted one of the undocumented features from March's implemented model and Rodan (2005) incorporated two. For computing knowledge held by an individual, the formula used by Miller et al. corresponds to net correct beliefs. Similarly, Rodan used net correct beliefs to compute knowledge. From his Figure 3 (p. 421), we deduce that he implemented two-step updating of member beliefs in his model. He makes no reference to random classification of zero beliefs in the process of updating the organizational code. However, Rodan introduced random replacement of zero beliefs as a new construct, "self-restrained experimentation," which alters agents' beliefs rather than merely substituting a nonzero belief temporarily when voting for the organizational code. In similar vein, Blaschke and Schoeneborn (2006) use the feature concerning two-step updating as a new construct "forgetting by an individual," which turns a nonzero belief into a zero (cf. Miller and Martignoni, 2016).

Five of these published studies—Rodan (2005), Blaschke and Schoeneborn (2006), Chanda and Ray (2015a), Chanda (2017), and Chanda et al. (2018)—acknowledge that the authors had access to March's original program. For the rest, we are not in a position to judge whether they did. They may have been unaware that the original program was available upon request. Whether these researchers accessed the program code or not, their general inclination was to anchor on the conceptual model, thereby omitting the features in the code undocumented in the published study. Their model implementations (as summarized in Table 1) ratify our interpretation that the three program features that we identified are indeed nonconforming with respect to the text of March's article.

Most importantly, no prior replication conformed fully to March's implemented model, so his model remained unverified. Disparities between March's conceptual and implemented models went unexamined and unexplained in published follow-on research. If verification of the original model had been a priority, conceptual model replication should have elicited efforts to examine details in the program code. However, each of the studies following March's model offered extensions, rather than rigorous attempts at verification. Extensions test the sensitivity of results to other theoretical considerations (Edmonds and Hales, 2003; Hales et al., 2003), but through acceptance of the original study, they leave model verification unexamined.

Discussion

Axtell et al. (1996) point out that alignment of computational models is essential to support cumulative progress in research (see also Burton, 2003). Only if subsequent models align with their predecessor, can researchers run critical experiments to test whether changes in assumptions advance a model's explanatory potential or whether a subsequent model subsumes its predecessors as special cases. Replications also can be precursors to model alterations or extensions because they give researchers occasions to reflect critically on the assumptions of the preceding model, as well as consider potential uses in wider domains. Beyond the obvious benefit that comes from verifying the results of prior research, replication studies foster shared language, practices, and understanding among members of a modeling community (Wilensky and Rand, 2007).

Nevertheless, research studies that seek to undertake exact replication of prior work have a tenuous standing in strategic organization scholarship. Researchers' careers advance more readily through publishing innovative results than by reproducing results already in the literature (Nosek et al., 2012). Tsang and Kwan (1999) contend that replications of empirical studies do not find their way into academic journals because editors believe that they lack novelty and, as such, fail to make a compelling contribution to theory development and testing. Replications are likewise rare in computational modeling research in the field of strategic organization. From time to time, we come across instances of (1) success in obtaining convergence between models developed by different researchers (e.g. Axtell et al., 1996), (2) partial success in replicating an earlier model (e.g. Blaschke and Schoeneborn, 2006), or (3) unveiling of shortcomings in a prior model (e.g. Edmonds and Hales, 2003). However, the evidence from published research shows few attempted replications by computational researchers. This dearth of replications may reflect a lack of shared appreciation, objectives, and methods for replication studies.

Our study offers guidance that is (1) particular to March's (1991) widely cited study and (2) general to modelers and those who review and use their research. Regarding the first subject, we identified, evaluated, and offered recommendations regarding disparities between March's (1991) text and the program code that generated his results. March's article serves as an example of how extensively a model and its results can influence subsequent research without careful review of details in the described and implemented models. Narrating our personal experience replicating March's (1991) model illustrates the learning that can come from applying a two-step replication method that starts from the conceptual model (*C*) and then turns to the original computer program (*I*) to uncover the sources of disparate results.

For the broader subject and audience (consisting of modelers, reviewers, and research users), we offer a case for independent replication based on a study's conceptual model. Laying out four possible scenarios and their implications established the value of replications proceeding from natural language specifications to produce formal computer code. The main requirement that we advocate is that a later researcher performs independently the mental tasks to derive a formal (computer code) model based on the natural language (conceptual) description in the pioneer's publication text. Such work is necessary to verify that the computer code conforms to the model specification given in the publication text.

We alert reviewers, editors, and users of computational modeling research to problems arising from disparities (i.e. substitutions, omissions, and additions) between a study's conceptual model and the implementation in the program code. The realistic possibility of such errors going undetected by authors, reviewers, and editors introduces a note of caution into claims regarding the precision and utility of formal computational models for advancing theory in strategic organization research (e.g. Adner et al., 2009; Burton, 2003; Carley, 1995; Davis et al., 2007; Harrison et al., 2007; Miller, 2015).

Clearly communicating the conceptual model is essential. Authors should aim for thoroughness and minimal ambiguity. To test whether a model description fulfills these criteria, researchers may provide the conceptual model to colleagues outside the research team for their scrutiny. A key test is whether an outsider can develop independently computer code that aligns with the original program. Authors of original models may hire independent programmers to replicate their work prior to publishing. Interested readers may refer to the work by Chanda and Ray (2015b) for some further guidance on developing simulation studies.

In the realm of theory development through agent-based modeling, we question a dominant belief that asking the originator of a study to make the program code publicly available is sufficient to advance science. Advocating that authors make their code public (Donoho, 2010; Ince et al., 2012) is an important but incomplete stance because it leaves open *how* replicators will use prior programs. If replicators simply work from the pioneer's program, errors can go undetected and persist within the follow-on stream of modeling research. Independent development of computer code from the pioneer's publication text will unearth issues that cannot be found by working from the program code to validate prior findings.

These observations both affirm and clarify Wilensky and Rand's (2007) contention that researchers attempting to replicate a pioneer's work should do so initially without looking at the pioneer's code, even if publicly available. Their suggestion has value, in that working from the conceptual model description avoids replicating errors and idiosyncrasies in the pioneer's program. A fresh implementation attempt has the potential to certify the translation of the natural language description to formal specification. Achieving successful replication when working only from the conceptual model description alone reassures all researchers interested in the original article that the stringent conditions of scenario 1 hold. However, if researchers fail to achieve exact quantitative replication, the only efficient recourse for unearthing reasons is examining the original program together with the text of the conceptual model. Thus, Wilensky and Rand's (2007) advice is an important *first step*, but frequently replication research calls for a *second step* requiring access to the original model program.

When developing an original model or attempting a replication, we recommend writing computer code in modular fashion to facilitate the diagnosis of problems in complex programs. The program code for one functionality is written exactly once (in a module or function or subroutine), even if the functionality is invoked from multiple locations (branches) of the main program. A conscious orientation toward developing code in a modular fashion forces researchers and programmers to design each processing block according to the Input–Process–Output paradigm. Anomalous outputs from experiments with a controlled set of inputs signal a faulty process, the source of which can be diagnosed and remedied much more readily in a stand-alone block of code than in a fully integrated program. Lead researchers and programmers should work together during testing of program modules to satisfy themselves that the theorized mechanisms are functioning properly. The expected outputs from a set of inputs may be computed in a spreadsheet; thereafter, the program module is checked for faithful performance on the same inputs.

If journal editors commit, as a matter of policy, to giving space to conceptual model replication studies, they will contribute to resolving flaws that hinder theory development. Appropriate incentives will be in place. Even journal space for less-than-full-length articles—as accorded to research notes—will go a long way toward motivating needed replication efforts. Unsuccessful replications (due to scenarios 2, 3, and 4 or their combination) should be published as readily as successful replications (scenario 1) because disparate results can carry important lessons. We expect that the wisdom of the crowd will be able to advance theory development, even if the original code is unavailable.

Conclusion

In a research community focused primarily on verbal theorizing and empirical testing, a deleterious chain of events can occur when published articles contain disparities between the model as described and as actually implemented. Many researchers draw upon the model description and the conclusions from the implemented model and, taking them at face value, assume that the latter follow logically from the former, but this may not be so. False inferences that go undetected can have ripple effects in theory development and empirical testing. We propose conceptual model replication as a method to check the diffusion of errant explanations and implications. Such replication of agent-based models assures strategic organization theorists and empirical researchers that findings follow from the written model description and engenders confidence among modelers to reuse an established model and build further theory through modifications and extensions.

Acknowledgements

The authors express their deep gratitude to Professor James March for his role in making this study possible. They also thank Ann Langley, Thorbjorn Knudsen, James LeBreton, and Olav Sorenson, who provided helpful comments on earlier drafts.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

Notes

1. In sum, for $C \rightarrow I^*$, persistent differences between the graphical output from I^* and the graphs provided in the publication text provoke researchers to thoroughly investigate the conceptual model (C). This assists detection of discrepancies arising from omissions in the conceptual or formal models and those arising from ambiguity in the specification text. In contrast, for $I \rightarrow I'$, even if redundancies in C or I are observed, there is a good chance that they get dismissed as irrelevant detail or noise. Ambiguities in C stay undetected as well because the formalization in I gets readily accepted. There is little impetus for checking each statement in C for admissible alternate meanings and specifications.
2. In a footnote, March (1991: 74) explained,

   More precisely, if the code is the same as the majority view among those individuals whose overall knowledge score is superior to that of the code, the code remains unchanged. If the code differs from the majority view on a particular dimension at the start of a time period, the probability that it will be unchanged at the end of period is $(1 - p_2)^k$, where k ($k > 0$) is the number of individuals (within the superior group) who differ from the code on this dimension minus the number who do not. This formulation makes the effective rate of code learning dependent on k, which probably depends on n.

3. The parameter settings specific to each panel were: (a) $p_3 = 0, p_4 = 0$; (b) $p_2 = 0.5, p_3 = 0, p_4 = 0$; (c) $p_2 = 0.5$, $p_3 = 0, p_4 = 0$; (d) $p_2 = 0.5, p_4 = 0$; (e) $p_1 = 0.5, p_2 = 0.5, p_4 = 0.02$. These parameter values hold for all presented figures (1 through 5). For all panels, $m = 30$ and $n = 50$.
4. We noticed that, in Figure 5 of March (1991: 80), the value of code knowledge at time period **1** has a value of zero. This led us to infer that the fruits of learning by the code get reflected in the next period, that is, not in the current period. We reasoned that the fruits of learning by the members (from the organizational code), likewise, should reflect in a subsequent period.

ORCID iD

Sasanka Sekhar Chanda https://orcid.org/0000-0002-1155-2199

References

Adner R, Pólos L, Ryall M, et al. (2009) The case for formal theory. *Academy of Management Review* 34(2): 201–208.

Axtell R, Axelrod R, Epstein JM, et al. (1996) Aligning simulation models: A case study and results. *Computational & Mathematical Organization Theory* 1(2): 123–141.

Bendor J, Moe TM, and Shotts KW (2001) Recycling the garbage can: An assessment of the research program. *American Political Science Review* 95(1): 169–190.

Blaschke S and Schoeneborn D (2006) The forgotten function of forgetting: Revisiting exploration and exploitation in organizational learning. *Soziale Systeme* 11(2): 99–119.

Burton RM (2003) Computational laboratories for organization science: Questions, validity and docking. *Computational & Mathematical Organization Theory* 9(2): 91–108.

Carley KM (1995) Computational and mathematical organization theory: Perspective and directions. *Computational & Mathematical Organization Theory* 1(1): 39–56.

Carley KM and Svoboda DA (1996) Modeling organizational adaptation as a simulated annealing process. *Sociological Methods & Research* 25(1): 138–168.

Chanda SS (2017) Inferring final organizational outcomes from intermediate outcomes of exploration and exploitation: The complexity link. *Computational & Mathematical Organization Theory* 23(1): 61–93.

Chanda SS and Ray S (2015a) Optimal exploration and exploitation: The managerial intentionality perspective. *Computational & Mathematical Organization Theory* 21(3): 247–273.

Chanda SS and Ray S (2015b) Formal theory development by computational simulation modeling: A tale of two philosophical approaches. *Decision* 42(3): 251–267.

Chanda SS, Ray S, and McKelvey B (2018) The continuum conception of exploration and exploitation: An update to March's theory. *M@n@gement* 21(3): 1050–1079.

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

Davis JP, Eisenhardt KM, and Bingham CB (2007) Developing theory through simulation methods. *Academy of Management Review* 32(2): 480–499.

Dent T (2012) Comment #41837 on Ince et al. (2012). The case for open computer programs. *Nature: Perspectives*. Available at: http://www.nature.com/nature/journal/v482/n7386/full/nature10836.html (accessed 20 December 2017).

Donoho DL (2010) An invitation to reproducible computational research. *Biostatistics* 11(3): 385–388.

Edmonds B and Hales D (2003) Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation* 6. Available at: http://jasss.soc.surrey.ac.uk/6/4/11.html (accessed 20 December 2017).

Fang C, Lee J, and Schilling MA (2010) Balancing exploration and exploitation through structural design: The isolation of subgroups and organizational learning. *Organization Science* 21(3): 625–642.

Fioretti G (2013) Agent-based simulation models in organization science. *Organizational Research Methods* 16(2): 227–242.

Galán JM, Izquierdo LR, Izquierdo SS, et al. (2009) Errors and artefacts in agent-based modelling. *Journal of Artificial Societies and Social Simulation* 12. Available at: http://jasss.soc.surrey.ac.uk/12/1/1.html (accessed 20 December 2017).

Gavetti G, Greve HR, Levinthal HA, et al. (2012) The behavioral theory of the firm: Assessment and prospects. *The Academy of Management Annals* 6(1): 1–40.

Hales D, Rouchier J, and Edmonds B (2003) Model-to-model analysis. *Journal of Artificial Societies and Social Simulation* 6(4). Available at: http://jasss.soc.surrey.ac.uk/6/4/5.html (accessed 20 December 2017).

Harrison JR, Lin Z, Carroll GR, et al. (2007) Simulation modeling in organizational and management research. *Academy of Management Review* 32(4): 1229–1245.

Holland JH (1975) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Ince DC, Hatton L, and Graham-Cumming J (2012) The case for open computer programs. *Nature* 482(7386): 485–488.

Kane GC and Alavi M (2007) Information technology and organizational learning: An investigation of exploration and exploitation processes. *Organization Science* 18(5): 796–812.

Kauffman S and Weinberger E (1989) The NK Model of rugged fitness landscapes and its application to the maturation of the immune response. *Journal of Theoretical Biology* 141(2): 211–245.

Kim T and Rhee M (2009) Exploration and exploitation: Internal variety and environmental dynamism. *Strategic Organization* 7(1): 11–41.

Levinthal DA (1997) Adaptation on rugged landscapes. *Management Science* 43(7): 934–950.

Maguire S, McKelvey B, Mirabeau L, et al. (2006) Complexity science and organization studies. In: Clegg SR, Hardy C, Lawrence TB, et al. (eds) *The SAGE Handbook of Organization Studies*, 2nd edn. London: SAGE, pp. 165–214.

March JG (1991) Exploration and exploitation in organizational learning. *Organization Science* 2(1): 71–87.

Midgley D, Marks R, and Kunchamwar D (2007) Building and assurance of agent-based models: An example and challenge to the field. *Journal of Business Research* 60(8): 884–893.

Miller KD (2015) Agent-based modeling and organization studies: A critical realist perspective. *Organization Studies* 36(2): 175–196.

Miller KD and Martignoni D (2016) Organizational learning with forgetting: Reconsidering the exploration-exploitation tradeoff. *Strategic Organization* 14(1): 53–72.

Miller KD, Zhao M, and Calantone RJ (2006) Adding interpersonal learning and tacit knowledge to March's exploration-exploitation model. *Academy of Management Journal* 49(4): 709–722.

Muldoon R (2007) Robust simulations. *Philosophy of Science* 74(5): 873–883.

Nosek BA, Spies JR, and Motyl M (2012) Scientific utopia: II. Restructuring incentives and practices to promote truth over publishability. *Perspectives on Psychological Science* 7(6): 615–631.

Posen HE and Levinthal DA (2012) Chasing a moving target: Exploitation and exploration in dynamic environments. *Management Science* 58(3): 587–601.

Rodan S (2005) Exploration and exploitation revisited: Extending March's model of mutual learning. *Scandinavian Journal of Management* 21(4): 407–428.

Schilling MA and Fang C (2014) When hubs forget, lie, and play favorites: Interpersonal network structure, information distortion, and organizational learning. *Strategic Management Journal* 35(7): 974–994.

Tsang EWK and Kwan KM (1999) Replication and theory development in organizational science: A critical realist perspective. *Academy of Management Review* 24(4): 759–780.

Vancouver JB and Weinhardt JM (2012) Modeling the mind and the milieu: Computational modeling for micro-level organizational researchers. *Organizational Research Methods* 15(4): 602–623.

Wilden R, Hohberger J, Devinney TM, et al. (2018) Revisiting James March (1991): Whither exploration and exploitation? *Strategic Organization* 16(3): 352–369.

Wilensky U and Rand W (2007) Making models match: Replicating an agent-based model. *Journal of Artificial Societies and Social Simulation* 10(4). Available at: http://jasss.soc.surrey.ac.uk/10/4/2.html (accessed 20 December 2017).

Author biographies

Sasanka Sekhar Chanda is an Associate Professor in strategic management in IIM Indore. His research interests are in strategic decision-making, managerial intentionality, and complexity theory. Earlier, he worked in industry in a range of roles spanning engineering, consulting, and management for a period of 15 years.

Kent D Miller is a Professor of management at the Eli Broad Graduate School of Management, Michigan State University. He received his PhD from the University of Minnesota. His research examines organizational learning and strategic change, as well as methodological and philosophical issues in management and organization studies.